A Scientific Understanding of Keystroke Dynamics

Kevin S. Killourhy

CMU-CS-12-100
January 2012

School of Computer Science
Computer Science Department
Carnegie Mellon University
Pittsburgh, PA 15213

Thesis Committee:
Roy A. Maxion, Chair
Daniel P. Siewiorek
Christos Faloutsos
David L. Banks (Duke University)

Submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy.

This research was sponsored by the National Science Foundation under grant numbers CNS-0430474, CNS-0435382, CNS-0716677, Defense Advanced Research Projects Agency under grant number FA8721-05-C-0003; U.S. Army Research Office under grant number DAAD-190210389, W911NF0910273; Department of Interior under grant number NBCHC-030109; and Commonwealth of Pennsylvania under grant number C000016682.

Keywords: Keystroke dynamics, Mixed-effects models, Science of security

Abstract

Keystroke dynamics—technology to distinguish people based on their typing rhythms—could revolutionize insider-threat detection. Insiders accessing backdoors, using shared accounts, or masquerading as other users would be exposed by their unique typing rhythms. In the past thirty years, dozens of classifiers have been proposed for distinguishing people using keystroke dynamics; many have obtained excellent results in evaluation. However, when evaluations are replicated, the results are often wildly different; one classifier's error rate jumped from 1% to 85% upon replication.

Classifier error rates depend on a multitude of factors; until the effects of these factors on error rates are understood, keystroke dynamics cannot realize its promise. To tackle this multitude-of-factors problem, we developed the following methodology: (1) evaluate multiple classifiers under systematically ranging conditions; (2) analyze the results with linear mixed-effects models (LMMs), a technique for inferential statistics well suited to understanding how various factors affect classifier error rates; and (3) validate the models, demonstrating that they accurately predict error rates in subsequent evaluations.

In three investigations using this methodology, we found that while some classifiers had lower error rates than others, the differences were overshadowed by the effects of factors other than the classifier. For the best classifier, error rates vary from 0% to 63% depending on the user. Impostors triple their chance of evading detection if they touch type. On the bright side, the best combination of timing features (hold times and up-down times) reduces error rates by over 50%. Other configuration tweaks, such as increased training and an updating strategy, offer further opportunity to significantly reduce error rates. On a scientific level, this work establishes that understanding these factors is critical to making progress in keystroke dynamics.

By understanding the influential factors, we can deploy the best classifier given the environment, accurately estimate its error rate, and know where to direct future efforts to improve performance. For the first time in keystroke dynamics, we can reliably predict classifier error rates, and our approach is general. Since other computer-security technologies (e.g., intrusion, worm, and malware detection) face analogous multitude-of-factors problems, they would similarly benefit from our methodology.

Acknowledgments

I could not have completed this work without the help of many colleagues, family, and friends. My advisor, Roy Maxion, taught me what it means to be a researcher; his mentorship has been invaluable. Kymie Tan was instrumental, patiently educating me on research and writing; Pat Loring has been equally instrumental, not only running experiments but also sharing her unique and valuable perspective. David Banks provided much needed encouragement in addition to statistical advice. Christos Faloutsos and Dan Siewiorek guided me in this work, incorporating both machine learning and computer systems.

Mark Stehlik, Jim Roberts, and Paul Mazaitis provided friendship and advice at critical instances throughout my time at CMU; without their intervention, I would never have gone to graduate school, never mind finished. Many other colleagues provided substantial assistance in myriad ways: Rachel Roberts, Fahd Arshad, Tahlia Townsend, Shing-hon Lau, Chao Shen, Bob Olszewski, Rob Reeder, Missy Harvey, Harvey Vrsalovic, Ian Welsh, Mary Theofanos, Howard Seltman, Bill Eddy, Will Guthrie, Cristiano Giuffrida, Lorenzo Cavallaro, and John McHugh. Deborah Cavlovich, Sharon Burks, and Catherine Copetas acted as friendly navigators through the CSD graduate-student experience.

My mother and father—Sheila and Stephen Killourhy—have always encouraged me. Many relatives also provided support (and helped me keep things in perspective): Peg Kelley and Robb Jackson; Peg, Ellis, Josh, and Ethan Goldman; and Kevin, Noriko, and Kento Riendeau. My friends also provided encouragement, advice, and understanding when I disappeared into the lab or the library for nights and weekends: Megan DeBertrand, Rachel Peck, Khalid El-Arini, Sean Westmoreland, Jeremy Murphy, Meclina Hunt, Adrienne Walsh, Meghan Kyle-Miller, Erica Cosler, and Betty and Jim Hanigan. As should be evident from the long lists of names, there are many who supported me, academically and personally, in this work. Without their help, I could not have done it. I am honored to have such great friends, family, and colleagues.

Contents

1 Introduction **1**
1.1 Statement of problem . . . 2
1.2 Proposed solution . . . 4
1.3 Thesis statement . . . 6
1.4 Outline of approach . . . 7
1.5 Glossary of terminology . . . 9

2 Background on Keystroke Dynamics **13**
2.1 History of keystroke dynamics . . . 13
2.2 Keystroke-dynamics classifier evaluations . . . 17
2.3 Statistical inference in keystroke dynamics . . . 20
2.4 Survey methodology . . . 22
2.5 Survey results . . . 25
2.5.1 Group 1: No statistical inference . . . 26
2.5.2 Group 2: One-factor-at-a-time inference . . . 27
2.5.3 Group 3: Multiple-factor inferences . . . 30
2.6 Related work on evaluation methodology . . . 32
2.7 Discussion . . . 33
2.8 Summary . . . 34

3 Linear Mixed-Effects Modeling **35**
3.1 Background . . . 35
3.2 Example evaluation . . . 37
3.3 Variance-stabilizing transformation . . . 40
3.4 LMM #1: Classifier and feature effects . . . 42
3.5 LMM #2: Classifier/feature-set interaction . . . 48
3.6 Parameter estimation . . . 49
3.7 Model selection . . . 50
3.7.1 Constructing a family of models . . . 50
3.7.2 Selecting the best model in a family . . . 53
3.8 Testing hypotheses . . . 55
3.9 Computational details . . . 59
3.10 Validation . . . 60
3.11 Summary . . . 65

4 Benchmarking Keystroke-Dynamics Classifiers 67
4.1 Background . . . 67
4.2 Aim and approach . . . 71
4.3 Experimental method . . . 71
4.3.1 Password-data collection . . . 72
4.3.2 Classifier implementation . . . 74
4.3.3 Evaluation procedure . . . 82
4.4 Empirical results . . . 85
4.5 Statistical analysis . . . 88
4.5.1 Model selection . . . 89
4.5.2 Parameter estimation . . . 90
4.5.3 Statistical hypothesis testing . . . 92
4.6 Validation . . . 95
4.7 Discussion . . . 98
4.8 Summary . . . 101

5 Personal Traits and Keystroke Dynamics 103
5.1 Background . . . 103
5.2 Aim and approach . . . 104
5.3 Experimental method . . . 105
5.3.1 Demographic survey . . . 105
5.3.2 Merged demographic and evaluation results . . . 107
5.4 Empirical results . . . 108
5.5 Statistical analysis . . . 111
5.5.1 Model selection . . . 111
5.5.2 Parameter estimation . . . 112
5.5.3 Statistical hypothesis testing . . . 114
5.6 Validation . . . 116
5.7 Discussion . . . 120
5.8 Summary . . . 122

6 Screening Keystroke-Dynamics Factors 125
6.1 Background . . . 125
6.2 Aim and approach . . . 126
6.3 Experimental method . . . 127
6.3.1 Selecting factor values. . . . 127
6.3.2 Typing-data collection . . . 129
6.3.3 Evaluation procedure . . . 131
6.4 Empirical results . . . 133
6.5 Statistical analysis #1: Feature sets . . . 137
6.5.1 Model selection . . . 138
6.5.2 Parameter estimation . . . 139
6.5.3 Statistical hypothesis testing . . . 141
6.6 Validation #1: Feature sets . . . 143

6.7 Statistical analysis #2: Many-factor analysis . . . 146
6.7.1 Model selection . . . 147
6.7.2 Parameter estimation . . . 148
6.7.3 Statistical hypothesis testing . . . 151
6.8 Validation #2: Many-factor analysis . . . 156
6.9 Discussion . . . 157
6.10 Summary . . . 158

7 Summary, Contribution, and Future Work 161
7.1 Summary of findings . . . 161
7.2 Impact for keystroke dynamics . . . 163
7.3 Supplemental materials . . . 166
7.4 Contributions . . . 166
7.5 Limitations . . . 167
7.6 A science of security: Beyond keystroke dynamics . . . 170

Bibliography 173

Keystroke-Dynamics Bibliography 185

List of Figures

1.1 Keystroke dynamics holds tentative promise as a security technology 2

3.1 Example keystroke-dynamics classifier evaluation results. 37
3.2 Variance-stabilization transformation for binomial random variables 42
3.3 Two model equations that define a family of model-selection candidates . . 51
3.4 Example results from secondary evaluation for model validation 62
3.5 Model predictions and validation data for the example evaluation 63
3.6 QQ-plots for validating LMM predictions from the example evaluation. . . 65

4.1 Screenshot of the collection software for typing data 73
4.2 Contours of the decision surfaces for the 10 implemented classifiers 75
4.3 Example ROC curve with the 5% false-alarm operating point marked. . . . 83
4.4 Per-user miss rates for the first 5 classifiers in the benchmark evaluation. . . 86
4.5 Per-user miss rates for the last 5 classifiers in the benchmark evaluation. . . 87
4.6 Equations for model selection in the benchmark-evaluation analysis 89
4.7 Model predictions and validation data for the benchmark evaluation 97
4.8 QQ-plots for validating LMM predictions from the benchmark evaluation. . 98

5.1 Miss rates by user and impostor age and gender 109
5.2 Miss rates by user and impostor dominant hand and typing style 110
5.3 Equations for model selection in the personal-trait investigation 111
5.4 Model predictions and validation data for personal-traits study (part 1) . . . 118
5.5 Model predictions and validation data for personal-traits study (part 2) . . . 119
5.6 QQ-plots for validating LMM predictions from the personal-traits study . . 120

6.1 Average miss rates for different typing tasks 134
6.2 Average miss rates for different timing features 135
6.3 Average miss rates for different updating strategies 136
6.4 Average miss rates for different impostor-familiarity levels 137
6.5 Equations for model selection in the feature-set analysis 138
6.6 Model predictions and validation data for feature-set analysis 144
6.7 QQ-plots for validating LMM predictions from the feature-set analysis . . . 146
6.8 Equations for model selection in the many-factor analysis 147
6.9 Model predictions and validation data for many-factor analysis (part 1) . . . 154
6.10 Model predictions and validation data for many-factor analysis (part 2) . . . 155

6.11 QQ-plots for validating LMM predictions from the many-factor analysis . . 156

List of Tables

2.1 Breakdown of the number of sources found in survey 24
2.2 Categories of inferential statistics found in literature survey 26

3.1 LMM #1—Possible LMM built from example data 43
3.2 LMM #2—Another possible LMM built from example data 47
3.3 Hypothesis tests comparing different classifiers and feature sets. 58

4.1 Different studies use substantially different evaluation procedures 69
4.2 Average error rates for the classifiers on the benchmark data 85
4.3 LMM for the results of the benchmark evaluation 91
4.4 Hypothesis tests comparing miss rates in the benchmark evaluation 94
4.5 Average error rates for classifiers on the validation set 96

5.1 Summary of the subjects' age, gender, dominant hand, and typing style . . . 106
5.2 LMM for the results of the personal-traits study 113
5.3 Hypothesis tests comparing miss rates for touch-typists and others 115
5.4 Hypothesis tests comparing miss rates across genders 116

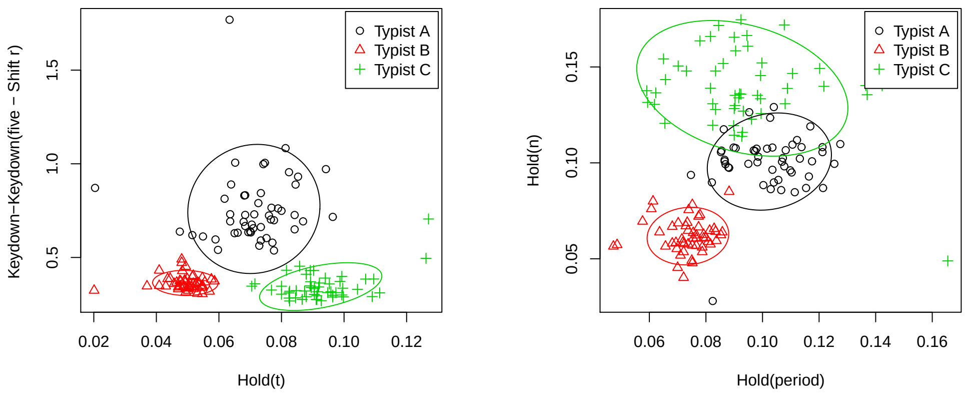
6.1 Summary of the number of subjects who completed each typing task 131
6.2 LMM model equation for the results of the feature-set analysis 139
6.3 LMM parameter-estimate table for the feature-set analysis 140
6.4 Hypothesis tests comparing miss rates on different feature sets 143
6.5 LMM model equation for the results of the many-factor analysis 148
6.6 LMM parameter estimates table for the many-factor analysis 149
6.7 Hypothesis tests comparing error rates as amount of training increases . . . 152
6.8 Hypothesis tests comparing miss rates for familiar and unfamiliar impostors 153
6.9 Hypothesis tests comparing miss rates with and without updating 153

Chapter 1

Introduction

If keystroke dynamics—technology to distinguish people by their typing rhythms—were demonstrably reliable, it would significantly advance computer security. For criminal investigations, keystroke dynamics could tie a suspect to the "scene" of a computer-based crime much like a fingerprint does in real-world crime. For access control, keystroke dynamics could act as a second factor in authentication; an impostor who compromised a password would still need to type it with the correct rhythm. For insider-threat detection, keystroke dynamics could detect when a masquerader is using another user's account; the technology could even identify who is using a backdoor account.

Figure 1.1 shows the results of a simple exercise to test the feasibility of keystroke dynamics. Three typists each typed the same password (.tie5Roanl) 50 times. The left panel compares the length of time they held the t key (horizontal axis), and the length of time between pressing the 5 key and the R key (vertical axis). The ellipses represent regions that a Mahalanobis classifier (described in Chapter 4) would associate with each of the three typists. Since these regions enclose most of the relevant typing samples and do not overlap, the figure suggests the promise of keystroke dynamics. However, the right panel shows that if we looked at different timing features, the typists would be more difficult to distinguish. Based on the hold times for the period key (horizontal axis) and the n key (vertical axis), Typist B is still usually easy to distinguish, but the timings of Typists A and C overlap substantially. These results suggest that keystroke dynamics may work better under some conditions than others (e.g., better for some users and timing features).



(a): Features with good separation *(b)*: Features with moderate separation

Figure 1.1: Keystroke dynamics holds tentative promise as a security technology. Three users typed the password .tie5Roanl 50 times. Panel (a) plots the t hold time and the 5-R down-down time for each repetition of the password. The three ellipses represent the typing regions learned for each user (by a Mahalanobis classifier); the regions effectively separate the typists. However, Panel (b) pairs period (.) and n hold times. The overlap between two of the regions suggests that keystroke dynamics may be more effective for some typists and timing features.

1.1 Statement of problem

Keystroke-dynamics research has been ongoing for 30+ years. Many different classification methods have been proposed during that time. Methods based on traditional statistics—such as mean typing times and their standard deviations—are common (Forsen et al., 1977; Gaines et al., 1980; Joyce and Gupta, 1990; Araújo et al., 2005). Over the years, different pattern-recognition methods have come into vogue and been applied to keystroke dynamics: neural networks (Brown and Rogers, 1993; Obaidat and Sadoun, 1997; Cho et al., 2000), fuzzy logic (Hussien et al., 1989; de Ru and Eloff, 1997; Haider et al., 2000; Mandujano and Soto, 2004; Tran et al., 2007), and support-vector machines (Yu and Cho, 2003; Sung and Cho, 2005; Loy et al., 2007; Giot et al., 2009a), among others.

A typical study of keystroke dynamics proposes a new classification algorithm to distinguish people based on their typing, collects some typing data, and uses the typing data to evaluate the accuracy of the classifier. Accuracy results are often expressed in terms of false-alarm and miss rates. The false-alarm rate is the percentage of typing samples from the legitimate user that are mistaken as impostor typing; the miss rate is the percentage of typing samples from other typists mistaken as the legitimate user's typing. There are also derivative metrics such as the equal-error rate (EER): the miss and false-alarm rate when the classifier is tuned so that the two error rates are equal.

Despite the massive research effort, or perhaps because of it, results for these classifiers are inconsistent. For instance, in one evaluation of a particular kind of neural network, the average false-alarm rate was 1% (Cho et al., 2000). When the classifier was re-implemented and re-evaluated, the average false-alarm rate increased to 86% (Killourhy and Maxion, 2009). In both cases, the classifier was tuned to have the lowest possible false-alarm rate with the miss rate constrained to 0%. A security officer might take a chance deploying this neural network if assured of a 1% false-alarm rate, but the 86% false-alarm rate would raise too many doubts.

Discrepancies between evaluation results for the same classifier can be explained if other factors affect a classifier's ability to distinguish typists. In general, the problem can be stated as follows:

> **In keystroke dynamics, a classifier does not have *an* error rate; it has *many* error rates, depending on a multitude of factors in the evaluation environment. Without identifying and understanding the effects of these factors, we cannot understand classifier behavior.**

Researchers effectively choose a particular evaluation environment (or set of environments) when they design the evaluation. The researcher decides the parameters of the evaluation when he or she recruits subjects, selects typing tasks, schedules typing sessions, picks a keyboard, implements a keystroke-timing mechanism, extracts features from the raw timestamps, tunes the classifier, and so on. All the decisions that go into conducting the experiment could affect the results. Any choice that affects the results restricts (perhaps unintentionally) the environments for which the results are valid.

Some factors in the evaluation environment seem to affect classifier error rates. While there have been a few attempts to understand the effects of some of these factors, the findings are always preliminary and sometimes contradictory. For instance, when a researcher proposes a new classifier, he might have subjects type some strong passwords and some English words. He might report the classifier's error rate under both conditions. While such empirical results can be useful, their overall utility is impeded by two obstacles: potential interactions and an absence of inferential statistics.

As discussed in detail in Chapter 2, experimental results in keystroke dynamics are almost never analyzed statistically. Without drawing inferences using methods like hypothesis testing or confidence-interval estimation, it is impossible to establish that any perceived effect is real. For instance, if a classifier achieves a 1% error rate with English words and a 10% error rate with the strong password, how likely is it that English words are easier for keystroke dynamics? Answering such questions is the purview of inferential statistics: drawing general conclusions from empirical data Dodge (2003).

Even if statistical analysis finds a significant difference between the English-word error rate and the strong-password error rate, questions remain about how closely the results are tied to the constants of the experiment. If fewer training samples were used, would the difference still be significant? If different subjects were recruited, or different words and passwords used, would we see the same difference? Do English words produce better error rates for all classifiers? Certainly no experiment can answer all questions about all possible interactions, but in current practice, most experiments answer no questions about any possible interactions.

When, as in the example above, one evaluation finds that a classifier is nearly perfect, and another evaluation finds that the same classifier might as well be flipping a coin, the field faces a serious problem: explaining the discrepancy. Perhaps one of the evaluations was flawed; perhaps some factor was present in the second evaluation that hampered the classifier; perhaps the classifier is just wildly inconsistent. Sorting out the correct explanation is crucial to understanding keystroke dynamics. A classifier that fails inexplicably cannot be trusted in security applications; the stakes are too high. If the status quo continues, and keystroke-dynamics classifiers remain inexplicably inconsistent, the technology is unlikely ever to be deployed. Eventually people will stop investing in uncertain and inconsistent research.

1.2 Proposed solution

While the multitude-of-factors problem is serious, it is not unique to keystroke dynamics or even computer security. Understanding the behavior of a complex system is a challenge in biology, ecology, medicine, and even quality control (Burnham and Anderson, 2002; Mayer, 2010; Wadsworth Jr. et al., 1986; Zuur et al., 2009). All of these disciplines must account for the myriad environmental and physiological factors that affect the results of experiments. Approaching computer security and keystroke dynamics as one would approach a physical- or life-science experiment may seem unconventional at first. However, security data such as network packets and programs are manifestations of people's behavior—keystroke timings especially so. As such, looking to other sciences for solutions to the problem of understanding keystroke-dynamics classifiers is reasonable.

These other sciences all rely heavily on good experimental designs and inferential statistics to make sense of complicated behavior. By designing experiments that systematically vary multiple factors, researchers in these other sciences discover factors that interact with other factors.

> We have usually no knowledge that any one factor will exert its effects independently of all others that can be varied, or that its effects are particularly simply related to variations in these other factors. [...] If the investigator, in these circumstances, confines his attention to any single factor, we may infer either that he is the unfortunate victim of a doctrinaire theory as to how experimentation should proceed, or that the time, material or equipment at his disposal are too limited to allow him to give attention to more than one narrow aspect of his problem. (Fisher, 1949)

Despite the availability of time, materials, and equipment, keystroke-dynamics research typically focuses on a single factor (e.g., the classifier), neglecting other factors and the possibility of interactions among factors.

Of course, when multiple factors are varied in an experiment, one must be careful to avoid accidentally *confounding* factors. Factors are said to be confounded when, in the course of an experiment, they vary in lock-step. For instance, in keystroke dynamics, if right-handed subjects are all asked to type an English word, while left-handed subjects are asked to type a strong password, the subject's dominant hand and the typing task would be said to be confounded. When two factors are confounded, the experiment cannot differentiate the effects of one from the effects of the other.

We rarely see such obvious cases of confounded variables within a single study, but we frequently see confounded variables across studies. For example, if one study evaluates Classifier A on Data Set 1, and another study evaluates Classifier B on Data Set 2, the classifier and data set have been confounded. One cannot compare the two classifiers soundly since any difference could be explained by the different data sets. Despite their lack of soundness, such comparisons are often made across studies.

With a well-designed experiment, researchers are still faced with the task of analyzing and interpreting the results. Linear mixed-effects models (LMMs) are a statistical technique for understanding and interpreting experimental results. With LMMs one can discover and understand the influences of many factors (and their interactions) on the results of an experiment. This technique can potentially enable us to make sense of classifier behavior.

The formal description of linear mixed-effects models and the procedure for using them to analyze keystroke-dynamics evaluation data will be explained in detail in Chapter 3. In the meantime, the following description may provide some intuition. LMMs relate explanatory factors to a response variable. The response variables in keystroke-dynamics evaluations are miss and false-alarm rates. The explanatory factors include, for example, the classifier, the typing task, the subject acting as a legitimate user, the subject acting as an impostor, the amount of training data, and the mechanism used to timestamp keystrokes.

LMMs can capture interactions among these explanatory factors. For instance, an analysis might find that the miss rate is higher when the impostor is familiar with the typing task (e.g., after having typed a password hundreds of times). However, further analysis might discover an interaction between impostor familiarity and the specific typing task (e.g., passwords or English words). If the task involves English words, then impostor familiarity may have less effect on the miss rate. Discovering such an interaction would be of immediate practical interest. Deploying classifiers for certain typing tasks and not others would mitigate the risk of impostors becoming familiar with a typing task to evade detection.

LMMs can also identify the sources of uncertainty in a model. Statistical models rarely make perfect predictions; there always remains some residual uncertainty or noise. LMMs can decompose this residual noise, for instance, into the uncertainty that comes from per-user variation and per-impostor variation. It may be that a classifier trained to recognize one user has lower error rates than the same classifier trained to recognize another user. Accordingly, we might might describe some users as "easy" and others as "hard." Knowing how much variation to expect between easy and hard users would be useful; if the variation is large, researchers can try to understand what makes some users easy. If they can be identified, the classifier can be deployed for the easy users, and researchers can study other classifiers for the hard users. Research effort would move in a different direction than if the classifier were simply mediocre for all users.

As explained above, keystroke-dynamics classifiers have many different error rates, depending on a multitude of factors in the evaluation environment. LMMs offer a solution to this problem by identifying the influential factors and explaining their effect. The resulting models can be used to understand which factors cause the classifier's error rate to change, and to explain when the classifier will succeed and how it might fail.

1.3 Thesis statement

Using linear mixed-effects models (LMMs) to explain classifier behavior, (a) is novel for keystroke dynamics, and (b) offers better understanding of the behavior (e.g., changing error rates) than current practices.

How will we show that LMMs offer better understanding? One might hope to show that

LMMs will produce a true description of classifier behavior, but assessing the ultimate truth of a model is beyond the realm of statistics. As Box et al. (2005) famously stated: “All models are wrong; some models are useful.” Rather than showing that an LMM is true, we will show that they are useful. Specifically, it would be useful if an LMM could accurately predict a classifier’s error rate in subsequent evaluations. Such an LMM would need to identify which factors cause the error rate to increase or decrease, and to accurately predict the magnitude of the change.

Prior to the current work, such predictions were not possible. Most prior work built no models, and so they offered no predictions. Inferential statistics in general, not just LMMs, are rarely used to analyse keystroke-dynamics evaluation results. The few uses of inferential statistics do not adequately investigate interactions among multiple factors. Predictions based on LMMs are more accurate, incorporating the effects of more factors and more interactions among factors.

1.4 Outline of approach

Chapter 1: Introduction

The problem facing keystroke-dynamics research is outlined: a classifier does not have *an* error rate; it has *many* error rates, depending on a multitude of factors in the evaluation environment. LMMs are offered as a potential solution to this problem. They can identify factors that cause a classifier’s error rate to change, explaining when the classifier will succeed and how it might fail.

Chapter 2: Background on Keystroke Dynamics

Prior work in keystroke dynamics is reviewed, providing many examples of how different researchers make different choices when designing their evaluations. These different choices make it difficult to compare results across studies. A systematic survey of the extant literature demonstrates that researchers rarely use statistical models, and that when models are used, they never capture influential interactions among factors. The lack of inferential statistics makes it difficult to draw conclusions from individual studies.

Chapter 3: Linear Mixed-Effects Modeling

The statistical concepts behind LMMs are reviewed, and the particular procedures used to analyze keystroke-dynamics evaluation data with LMMs are explained. Topics include variance-stabilizing transformations, maximum-likelihood parameter estimation, model selection, and validation.

Chapter 4: Benchmarking Keystroke-Dynamics Classifiers

Ten classifiers are evaluated using typing data from 51 subjects; the error rates are analyzed using LMMs. The analysis not only compares the classifiers but also establishes how their error rates change across users and impostors. We establish not only which classifiers have the lowest error rates but also how much easier some users and impostors are to distinguish than others. These per-user and per-impostor effects increase the uncertainty of classifier error rates.

Chapter 5 Personal Traits and Keystroke Dynamics

The variation in per-user and per-impostor error rates might be attributable to personal traits of the typist (e.g., dominant hand, touch-typing skill, and possibly even age or gender). A demographic survey was administered to subjects from the benchmarking investigation (Chapter 4). The demographic data are combined with a subset of the benchmarking results and analyzed using LMMs. We identify the user's and impostor's typing styles—whether they are touch typists—as playing a significant role in the miss rate. This investigation partially addresses the question of why some users and impostors have higher miss rates than others.

Chapter 6 Screening Keystroke-Dynamics Factors

A top-performing subset of classifiers from the benchmark evaluation (Chapter 4) are subjected to another series of evaluations to understand how other factors in the evaluation environment—beyond the classifier, user, and impostor—affect their performance. Factors include the typing task, number of samples used for training, timing features, updating strategy, and impostor familiarity. This many-factors-at-once experiment reveals that interactions among the factors do affect classifier error rates and could be responsible for wild changes in those error rates across evaluations.

Chapter 7 Summary, Contributions, and Future Work

The findings of the three investigations are compiled and presented as evidence of what can be learned through analysis with LMMs. The contributions of this work are enumerated, including the public release of the classifiers, data, evaluation scripts, and analytical procedures so that future research might build upon them. Future work and implications of this work for keystroke dynamics and computer security are discussed. Of note, LMM-based analysis could be used in security research beyond keystroke dynamics. Many fields struggle with similar multitude-of-factors problems.

1.5 Glossary of terminology

This dissertation uses many technical terms from a variety of fields including keystroke dynamics, statistics, machine learning, and signal-detection theory. Certain terms are overloaded across these fields (and, in some cases, within a field). Consequently, this glossary provides definitions for many of the technical terms used in this work, *as* they are used in this work.

Classifier: A program (or function) with a training phase and a classification phase. During training, samples labeled with a class name are presented to the classifier which builds a profile describing the samples associated with each label. During classification, a new sample is presented to the classifier, which uses its profile to predict the correct label for the sample. Its output is the predicted class, accompanied by a score indicating (often in unit-less terms) the certainty in the class label. In this work, the samples are typically vectors of typing times collected while subjects type passwords, and the class labels are one of genuine user or impostor.

Factor: A measurable variable in an experiment that may have an effect on the outcome of the experiment. In this work, the experiments are typically evaluations and comparisons of keystroke-dynamics classifiers. The factors in these experiments are the classifiers themselves and various characterizations of the evaluation environment. For example, the particular typing task, the user's typing style, the impostor's typing style, and the particular combination of timing features used to train and test a classifier are all factors. In this work, every factor is categorical, taking two or more values. For instance, when subjects type a strong password, a simple English word, and a numeric passcode, the values of the typing-task factor are Strong, Simple, and Numeric, respectively.

Error: The misclassification of a sample by a classifier. For instance, both a miss (i.e., the misclassification of an impostor's typing sample as genuine) and a false alarm (i.e., the misclassification of a genuine user's typing sample as an impostor's) are specific kinds of error. In statistics, error often refers to the residual-error term in a statistical model. We will avoid that usage, and instead refer to such a residual-error term as a *noise term* or *residual effect*.

Equal-error rate (EER): A single-number measure of classifier performance. The equal-error rate aggregates two other error rates (i.e., miss and false-alarm rates) into a single number. Many classifiers, including all those presented in this work, can be tuned to exchange misses for false alarms. One tuning results in high-miss/low-false-alarm rates; another tuning results in a low-miss/high-false-alarm rates. The equal-error rate is the miss and false-alarm rate when the classifier has been tuned so that the two are equal (Bolle et al., 2004).

False-alarm rate: A measure of how accurately a keystroke-dynamics classifier recognizes a genuine user's typing. The *true* false-alarm rate is the long-run frequency with which typing samples from genuine users are misclassified as having been typed by an impostor. The *empirical* false alarm rate is the actual proportion of typing samples misclassified as being typed by an impostor in an evaluation. The false-alarm rate is often presented under other names: false-positive rate (FPR), false-reject rate (FRR), and Type I error. We will only use false-alarm rate to denote this concept.

Feature set: A set of timing features used by keystroke-dynamics classifiers to compare typists. The feature sets used in this work are *hold times*, the time between when a key is pressed and released; *down-down times*, the time between when the first key in a digraph is pressed and the second key is pressed; and *up-down times*, the time between when the first key in a digraph is released and the second key is pressed. The up-down time can be negative if the second key is pressed before the first one is released.

Fixed effect: A parameter in a statistical model that represents a deterministic relationship between a factor and the response variable. For instance, in a model that predicts a 50 percentage point drop in the average miss rate when Classifier A is used instead of Classifier B, -50 is a fixed effect. Fixed effects are used to model factors when our interest is in understanding how changing the value of the factor changes the average (mean) outcome (e.g., how changing the classifier changes the miss rate).

Interaction term: A parameter in a statistical model that represents the combined effect of two or more factors. Specifically, interaction terms are necessary when the effects of one factor depends on the value of the other factor. For instance, it could be that classifier error rates are lower when the number of training samples increases, but the reduction is greater when the typing task is a password and less when it is an English word. Accurately describing this effect on the error rate would require a typing-task/training-amount interaction term.

Linear mixed-effects models (LMMs): A family of statistical models that represent a stochastic relationship between multiple explanatory variables and a response variable. The explanatory variables include both fixed effects and random effects (hence the *mixed-effects* descriptor). The relationship is modeled as linear, but since our explanatory variables are typically categorical, linearity is not a strong assumption.

Miss rate: A measure of how accurately a keystroke-dynamics classifier recognizes an impostor's typing. The *true* miss rate is the long-run frequency with which typing samples from an impostor are misclassified as having been typed by the genuine user. The *empirical* miss rate is the actual proportion of typing samples misclassified as being typed by a genuine user in an evaluation. The miss rate is often presented under other names: false-negative rate (FNR), false-accept rate (FAR), impostor-pass rate (IPR), and Type II error. We will only use miss rate to denote this concept.

Evaluation environment: The entire context in which a classifier is evaluated. For keystroke dynamics, the environment includes not only the physical environment (e.g., the location, lighting, desk position, and keyboard), but also the parameters of the evaluation, including the typist, the typing task, how the keystroke times are recorded, the timing features extracted, and the amount of the enrollment data. Informally, the evaluation environment comprises all the "stuff" in an evaluation other than the classifier itself.

Random effect: A parameter in a statistical model that represents a stochastic relationship between a factor and a response variable. For instance, in a model that predicts an average change of ± 10 percentage points in the average miss rate because some typists are harder to distinguish than others, the per-user variation is a random effect. Random effects are used to model factors when our interest is in understanding how much uncertainty (i.e., variance) the factor adds to the outcome.

Chapter 2

Background on Keystroke Dynamics

We present a brief history of the 30+ years of keystroke-dynamics research, and we specifically concentrate on the many different choices that have been made when evaluating new classifiers. These different choices have led us to the present, problematic situation where classifier error rates may be explained by any of a multitude of factors. We explain why inferential statistics might solve this problem, and we conduct a survey to establish how rarely inferential statistics are used. Across 170 journal articles, conference papers, and other research reports, we find that 87.1% drew no statistical inferences; 10.0% used inferential statistics to investigate one factor at a time; only 2.9% used inferential statistics to investigate multiple factors and their interactions. This review establishes that keystroke dynamics has a *multitude-of-factors problem*, and that the solution proposed in this work—inferential statistics using linear mixed-effects models—is novel for keystroke dynamics.

2.1 History of keystroke dynamics

Keystroke-dynamics research was inspired by much older work that distinguished telegraph operators by their keying rhythms. This capability—which was allegedly quite useful during World War II for identifying radio operators and tracking troop movements (Gladwell, 2005)—was first formally investigated by Bryan and Harter (1897, 1899) as part of a study on skill acquisition in telegraph operators.

Keyboard typing rhythms were first considered as a means of distinguishing typists in the mid 1970s. Spillane (1975) suggested in an IBM technical bulletin that typing rhythms might be used for identifying the user at a computer keyboard. That bulletin described keystroke dynamics in concept; it laid out no specific classifier and reported no empirical evaluation results.

Forsen et al. (1977) conducted preliminary tests of whether keystroke dynamics could be used to distinguish typists. Their work considered dozens of different means of authentication, with keystroke dynamics among those considered. A small group of subjects typed their own and each others' names, and the authors presented summary statistics showing that a subject typing his or her own name was distinguishable from another subject typing the same name.

Gaines et al. (1980) produced an extensive report of their investigation into keystroke dynamics. They recorded seven typists over two sessions separated by months. In each session, the subjects transcribed three pages of words and sentences. The authors performed a variety of statistical analyses. They showed that down-down times (i.e., the time between the key-down event of the first key in a digraph and the key-down event of the second key) are log-Normally distributed. They found that a subject's down-down times do not change substantially from the first session to the second. They derived a statistical test that could be used to decide whether a transcription record was typed by a particular subject or not. The test perfectly distinguished the seven subjects, but the authors explained that follow-up work would be necessary because of the small number of subjects and the large amount of transcription required.

In the intervening 30+ years, hundreds of classifiers have been proposed. Survey papers by Peacock et al. (2004) and Karnan et al. (2011) review many of these classifiers. We present a bibliography of 170 studies of keystroke-dynamics classifiers at the end of this document (assembled as part of a survey described later in this chapter). It should be noted that these classifiers do not all offer solutions to the same keystroke-dynamics problems. Within keystroke-dynamics research, classifiers can actually be used for a variety of different purposes.

The early works by Forsen et al. and Gaines et al. provide good examples of two broad categories of keystroke-dynamics research: login-type and in-session authentication. With login-type applications, classifiers are typically presented with short typing samples similar to what would be seen at login: user IDs, passwords, names, and passphrases. The work by Forsen et al. falls into this category. In contrast, with in-session applications, classifiers are presented with longer and more free-form typing samples, that a classifier might be expected to encounter if monitoring a user's normal typing activities (e.g., writing emails and word processing). The work by Gaines et al. falls into this category.

Many classifiers, based on a variety of different technologies, have been proposed for login-type authentication. Brown and Rogers (1993, 1994) proposed using neural networks to identify impostors, achieving a 0% miss rate and 12.0% false-alarm rate. A classifier based on fuzzy logic was built by de Ru and Eloff (1997) to reason about whether a typing sample belonged to the genuine user; the classifier achieved a 2.8% miss rate and 7.4% false-alarm rate. Haider et al. (2000) combined multiple classification technologies (neural network, fuzzy logic, and statistics) into an ensemble classifier and reported a 6% miss rate and 2% false-alarm rate. Araújo et al. (2004) used a Scaled Manhattan classifier, finding a 1.89% miss and 1.45% false-alarm rate. Chen and Chang (2004) and Chang (2005) used a hidden Markov model (HMM) and obtained miss and false-alarm rates between 1% and 1.5% (depending on the tuning). Bartlow and Cukic (2006) used a random forest and found miss and false-alarm rates of 2%. Azevedo et al. (2007a,b) proposed a support vector machine (SVM) wrapped in a feature-selection algorithm that used genetic algorithms and particle-swarm optimization. They reported miss and false-alarm rates between 1.1% and 1.2%. In summary, nearly every promising pattern-recognition or machine-learning technology has been applied at some point to distinguishing users based on how they type during login authentication; many report promising results.

Notable among researchers who developed login-type classifiers are Bleha et al. (1990) who had subjects type their names and the phrase “University of Missouri Columbia” and used classifiers based on Scaled Euclidean and Mahalanobis distance. In that work, they were among the first to identify different sub-problems in login-type keystroke dynamics. In the first problem, a classifier is given a set of training samples from a known typist and a test sample from an unknown typist. The classifier is asked to determine whether the known typist produced the test sample. On this problem, their classifiers achieved a 2.8% miss rate and an 8.1% false-alarm rate. In the second problem, the classifier is given a set of training samples from multiple known typists and a test sample from an unknown typist. The classifier is asked to determine which of the known typists produced the test sample. On this problem, the classifiers achieved a 1.17% misclassification rate. In biometrics, these two problems are called *verification* and *identification* respectively (Mansfield and Wayman, 2002), and they represent another distinction among keystroke-dynamics classifiers.

Obaidat and Sadoun (1997) had 16 subjects type their own and each others’ user IDs. They constructed neural networks and a variety of other pattern-recognition classifiers, and they trained each classifier to distinguish each subject’s typing from the other subjects’. Their work is of particular note because they achieved perfect results (i.e., 0% misclassification). This work demonstrates the promise of login-type keystroke dynamics, but since their classifiers trained on thousands of typed repetitions of each user ID from each subject, more work is required before a practical classifier would be ready for deployment.

Monrose et al. (2002) considered the logistics of using keystroke-dynamics for login-type authentication without compromising information about the password in the process. If a UNIX password file is stolen, cryptographic hashes make it challenging to recover the passwords. These researchers developed a classifier that could store its typing profiles in such a way that if the stored information were stolen, the thief would be unable to recover the details of the password or the legitimate user's typing rhythms.

Fewer classifiers have been proposed for in-session authentication. Ahmed and Traore (2005) use a neural network, achieving a miss rate of 0.651% and a false-alarm rate of 1.312%. Changshui and Yanhua (2000) used auto-regressive time series in combination with k-NN (k-Nearest Neighbors), but only achieved misclassification rates of 36.34%. Janakiraman and Sim (2007) used statistical methods like Naive Bayes and the Bhattacharyya distance to estimate the probability that two typing samples came from the same typist. They obtained perfect accuracy (i.e., 0% misclassification rate) for common English digraphs such as an and in. Harun et al. (2010) returned to neural networks, using more modern radial-basis function (RBF) networks, but only obtained miss and false-alarm rates of 22.9%.

Much of the work on in-session authentication has been done by Bergadano et al. (2002, 2003) and Gunetti and Picardi (2005). They developed an algorithm for comparing the similarity of two typing samples based on the typing times. The algorithm compared the relative speeds at which different digraphs were typed (e.g., whether the th digraph is typed faster than the he digraph). They obtained a miss rate below 0.005% with a false-alarm rate below 5%. Gunetti et al. (2005a,b) refined the algorithm and examined whether it worked across languages (e.g., comparing samples typed in Italian and English by the same subject) and with samples typed months apart. They found no substantial change in the error rates.

It seems that the focus of most keystroke-dynamics research has been on finding the right classifier for each problem. Nearly every paper proposes a new classifier, and many propose several new classifiers. Yet, we also see the same families of classifier appearing in multiple studies. Neural networks have been used in over a dozen studies. Support vector machines (SVMs) and k-NNs have been independently proposed multiple times. Many different papers propose statistical methods that differ only slightly (e.g., using Scaled Manhattan vs. Scaled Euclidean as a distance metric). Each of these classifiers is evaluated and empirical results are reported (e.g., miss and false-alarm rates).

When we look at how a family of classifiers has fared in multiple evaluations, we see wildly different results. For neural networks, reported miss rates range from 0.0% to over 22% (Brown and Rogers, 1993; Haider et al., 2000), and false-alarm rates range from 0.0%

to 22.9% (Ali et al., 2009; Harun et al., 2010). For k-NNs, reported equal-error rates range from 0.9% to 23.61% (Zack et al., 2010; Loy et al., 2007). For SVMs, misclassification rates range from 1.58% to 15.3% (Azevedo et al., 2007b; Giot et al., 2011). Since the classifiers are essentially the same, perhaps differences in the evaluations explain the different results.

2.2 Keystroke-dynamics classifier evaluations

Nearly all keystroke-dynamics evaluations involve (1) recruiting subjects, (2) presenting them with a typing task, (3) recording keystroke-timing information, (4) extracting features suitable for training and testing a classifier, and (5) training the classifier using one portion of the typing data and testing it using another. In each of these five steps, researchers make a lot of choices. When different researchers make different choices, they introduce a factor (other than the classifier) that may explain any difference in their results. In this section, we consider some of the different choices that researchers have made regarding each of the five evaluation steps noted above.

1. Recruiting subjects. A researcher must choose how many subjects to recruit. Subjects can be recruited to act as genuine users, impostors, or both. Bartmann et al. (2007) used 87 genuine-user subjects, while Cho and Hwang (2005) used 1. Jiang et al. (2007) used 257 impostor subjects when Bleha et al. (1990) used 9. One could imagine a per-user or per-impostor effect, whereby some users and impostors are just easy to distinguish using almost any classifier, and others are hard to distinguish. If the per-user or per-impostor variability is very high, researchers might need many subjects to accurately estimate a classifier's miss rate. Per-user and per-impostor variability, if they exist, could explain wildly different results.

Different subjects have different traits, of course. Some are touch typists, others are not; some are right handed, others left handed. Researchers typically attempt to recruit a diverse sample of subjects. For instance, Bartmann et al. (2007) reported that "both genders" were represented. Giot et al. (2009b) presented a detailed breakdown of their subjects' age and gender. Bartlow and Cukic (2006) described their subjects as ranging "from the most inept 'hunt and peck' typists to individuals with professional training." Having a diverse set of subjects in an evaluation helps to ensure that the evaluation results will generalize. However, if personal traits like touch typing affect classifier error rates (e.g., because error rates increase when both user and impostor are touch typists), then results will depend on the particular proportion of subjects in each trait group. Differences in these proportions may explain different error rates.

2. Typing task. As noted above, some researchers work on login-type authentication while others work on in-session authentication. Among research on login-type authentication, where subjects type the same sequence repeatedly, the sequence ranges from a 7 character password to a 50 character sentence (Cho et al., 2000; Bartmann et al., 2007). Among research on in-session authentication, where subjects type long spans of text, some researchers have subjects transcribe text (e.g., a passage from a novel), while others monitor keystrokes during subjects' day-to-day activities (Bergadano et al., 2002; Dowland and Furnell, 2004). Because research has found some digraphs to be better than others for accurate keystroke dynamics (Gaines et al., 1980; Janakiraman and Sim, 2007), we know that the error rates depend on the typing task. Perhaps these different typing tasks explain why different researchers get different error rates.

3. Recording timestamps. Researchers typically give very little detail about the instrumentation they used to collect keystrokes and keystroke timestamps while subjects perform the typing task. From those that have reported these details, we know that a range of keyboards have been used, from standard IBM PC keyboards to laptop keyboards (Umphress and Williams, 1985; Araújo et al., 2004). Researchers rarely specify whether the keyboard communicates using USB or PS/2, each of which introduces distinctive timing differences (Plant and Turner, 2009). A range of operating systems has been used: MS-DOS, SunOS, and Windows (Obaidat and Sadoun, 1997; Cho et al., 2000; Bartmann et al., 2007). Timestamps have been collected using X Windows, Java, Javascript, and by directly polling the high-resolution Pentium timestamp counter (Cho et al., 2000; Bartlow and Cukic, 2006; Jiang et al., 2007; Wong et al., 2001). While most research is conducted in a laboratory setting, some researchers distribute the data-collection programs to their subjects (Hosseinzadeh and Krishnan, 2008; Bartlow and Cukic, 2006; Jiang et al., 2007).

Different keyboards, operating systems, and timestamping software may affect the timestamp accuracy. Since those timestamps become the classifier inputs, differences in how they are collected could manifest as differences in classifier error rates. Analogous problems have been discovered in speaker-recognition research when subjects use different microphones (Doddington et al., 2000).

4. Extracting features. Having obtained raw timestamps for each keystroke, researchers must decide how to extract timing features suitable for input to their classifiers. The usual feature sets are hold times, digraph down-down times, digraph up-down, or some combination of the three (Joyce and Gupta, 1990; Cho et al., 2000; Araújo et al., 2004). Occasionally, digraph up-up times are used (Hosseinzadeh and Krishnan, 2008), and some work uses timing features for longer keystroke sequences like trigraphs (Bergadano et al., 2002).

Past researchers have compared their classifier's error rates with different combinations of features and found that error rates depend on the particular combination (Napier et al., 1995; Araújo et al., 2004). Consequently, we know that when different researchers extract different features, their choice has likely affected how well their classifier will do. If two classifiers are evaluated with different feature sets, any difference in their error rate could be caused by the different feature sets.

5. Training and testing. Once the timing features are ready for input to the classifier, researchers must choose some portion for training and another portion for testing its accuracy. In many cases, one subject or set of subjects are designated as genuine users and the rest as impostors. Some genuine-user samples are used to train the classifier. The remaining genuine-user samples are used to measure the classifier's false-alarm rate, and some impostor samples are used to measure the miss rate.

Different researchers make different choices when choosing training and testing data. Within login-type evaluations, the number of per-subject samples used to train a classifier ranges from 6 to 325 (Joyce and Gupta, 1990; Cho et al., 2000). With in-session evaluations, the number of characters typed in the training sample varies from 300 to 2200 (Bergadano et al., 2002; Furnell et al., 1996). Typically, researchers use subjects in different roles in multiple evaluation trails. A subject acts as the genuine user in one trial and as an impostor in trials where each of the other subjects acts as the genuine user. However, some researchers choose to recruit genuine-user and impostor subjects separately (Bergadano et al., 2002). The amount of training data, test data, and the split between the two always plays a role in the measured error rate of a classifier (Hastie et al., 2001). Different choices in how the data are presented to the classifier for training and classification would certainly result in different reported error rates.

Multitudes of factors in evaluations. We have considered keystroke-dynamics classifier evaluation as a five-step procedure, and we examined some of the choices that researchers must make when conducting an evaluation. We have shown that different researchers make different choices, and that those different choices could explain why they get different error rates. Since we see similar classifiers with wildly different error rates, it appears that some of these choices *must* affect error rates, but which ones? How do we discover which factors among the multitudes affect error rates?

As Mansfield and Wayman (2002) point out when discussing the evaluation of biometric systems, innumerable factors could affect performance. They acknowledge that evaluators divide factors into four categories (perhaps implicitly):

1. factors to be systematically varied in the evaluation to observe their effect;
2. factors to be controlled as constants during the evaluation to limit their effect;
3. factors to be randomized—either by recruiting a diverse set of subjects or by assigning conditions randomly—to average out their effect;
4. factors to be ignored or assumed to have negligible effect either because they cannot be observed, they are difficult to control, or their effect is deemed implausible.

We find this factor-handling taxonomy useful when questioning how to identify which factors affect error rates, and how. The first category offers the only way to test whether a factor has an effect with any certainty. The second and third categories offer ways to prevent factors beyond the scope of a test from affecting its results. The final category offers a reminder that we can never control everything. Future researchers might discover previously unknown factors, and evaluation methods must adjust to compensate for the discovery.

Researchers are certainly aware that a classifier's error rate may depend on factors in the operating environment, and some have conducted comparative experiments to measure how different choices in the evaluation affect the results. However, the experiments alone have not been sufficient to really understand which of these factors have an effect, or what its effect is. For that, one must not only perform an experiment in which the factor of interest is systematically varied; one must analyze and interpret the results correctly.

2.3 Statistical inference in keystroke dynamics

Statistical inferences enable a researcher to understand empirical results and explain what they mean. Without inferential statistics, one must either avoid drawing any conclusions or risk jumping to conclusions that are not supported by the data. In addition to being standard scientific practice, inferential statistics are necessary for understanding which factors are responsible for changing classifier error rates.

Inferential statistics include confidence-interval estimation, hypothesis testing, and information-theoretic model selection. With any of these techniques, a researcher can draw general inferences from specific experimental results. Moreover, the inferences are based on impersonal statistical models rather than the wishful thinking of the researcher. Any of these techniques can be used to understand which factors influence the outcome of an experiment, and which had either no effect or negligible effect.

For instance, Araújo et al. (2004) evaluated a classifier seven times, each with a different feature set (i.e., hold times, down-down times, up-down times, and various combinations of these features). They reported the error rates under each condition, but they did not use any statistical technique to understand and explain the results. Consequently, it is difficult to interpret their results. Is one combination of features really better than all the others, or are several combinations roughly equivalent? (We conduct our own comparison of these features in Chapter 6.)

They interpreted the results to mean that combining all three feature sets produces the best results. While that combination was empirically shown to be best for their particular data set, they provided no evidence that it will also be true in subsequent evaluations. One of the goals of inferential statistics is to establish how uncertain the empirical results are, and to predict the range of results one can expect in the future. If we make predictions without using inferential statistics, we are effectively making a leap of faith that we can reason about uncertainty without doing a formal analysis; unfortunately, people are typically quite bad at such reasoning (Nisbett, 1993).

Given their results, were Araújo et al. (2004) correct that using all three feature sets improved performance? In other words, should other researchers who used only hold times and down-down times expect their error rates to improve if they used up-down times as well? If so, this important result would offer a way to improve many previously proposed classifiers. If not, the result would be important in a different way, ruling out an unprofitable line of further research. Any answer to these questions would be an important result, but the questions are not answered by a report of the empirical error rates without any statistical analysis.

As noted above, statistically based inferences are not only good general practice, but they are necessary to address the problem raised in this dissertation; they enable an understanding of which factors in the operating environment affect classifier error rates, and how. Without inferential statistics, even if a factor is systematically varied across a series of evaluations and the results are reported, one cannot say with any precise certainty what effect the factor had.

It is certainly possible with some particularly strong results for an author to assert that a factor has an effect simply by eyeballing the numbers. Such authors are mentally judging that the outcomes are different enough and the sample size is large enough, making a formal statistical test unnecessary. They are effectively using inferential statistics implicitly rather than explicitly. With simple hypothesis testing (e.g., establishing that a factor matters) and with a large difference in the outcomes, an ad-hoc approach to statistical inference may not run serious risk of error.

However, even if these ad-hoc approaches arrive at the correct answer, they are inadvisable because of how many more questions could be answered if a more rigorous approach were used. For example, a researcher comparing classifiers might look at the evaluation results and see that Classifier A had a 1% equal-error rate while Classifier B had a 25% equal-error rate. Looking at these numbers, the researcher might conclude that he does not need inferential statistics to tell him that Classifier A is better than Classifier B. If the evaluation involved a sufficiently large and diverse set of subjects, the researcher is right. A hypothesis test would only confirm what his intuition is telling him.

However, formal statistical inferences could enable the researcher to place confidence intervals around Classifier A's 1% equal-error rate. Future researchers could use these intervals to predict how the classifier will perform in subsequent evaluations. They could compare error-rates across evaluations to determine whether new results are consistent with the predictions. Inconsistencies could be investigated, leading to a better understanding of Classifier A and the factors that affect its performance. None of this would be possible without doing the formal investigation.

Formal statistical modeling would also enable researchers to discover interactions among factors. Interactions (e.g., different classifiers are better for users with different typing styles) would be too complicated to find using ad-hoc assessments of results. In summary, while one can sometimes forgo a hypothesis test without much risk of error, doing so is inefficient and under-utilizes the experimental results.

2.4 Survey methodology

We believe that inferential statistics (and linear mixed-effects models in particular) offer a solution to the multitude-of-factors problem in keystroke-dynamics research. To establish that our solution is novel, we conduct a survey of the literature to determine how often previous sources have used inferential statistics, and what kind of inferential technique was used. Note that we refer to these previous works as *sources*, because they include papers, articles, book chapters and theses. To obtain a large and representative sample, we consulted four databases of computer-science journals and conference proceedings: IEEE Xplore, ACM Digital Library, SpringerLink, and ScienceDirect; the last is a search tool that includes books and journals published by Elsevier. In our experience, IEEE, ACM, Springer, and Elsevier publish the vast majority of research papers on computer science and, in particular, computer security (where the majority of keystroke-dynamics papers are published).

Using each of the four databases, we conducted keyword searches for (1) *keystroke dynamics*, (2) *keystroke biometrics*, and (3) *keystroke authentication*. With each database's particular search syntax, we searched for sources that included both words of a search (e.g., *keystroke* and *dynamics*), rather than the two-word string itself. Note for the ACM Digital Library, we limited the search to titles and abstracts with the ACM itself as the publisher (to avoid overlap). For Science Direct, we limited the search to sources with the search terms appearing in the title, abstract, or keywords. Science Direct does not include results with the same stem as the search string, so alongside searches for *dynamics* and *biometrics*, we searched *dynamic* and *biometric*.

Searching these four databases yielded 215 sources, including journal articles, conference papers, and book chapters. However, some important and high-profile work in keystroke dynamics was not among the sources returned. For instance, some early work, technical reports, and theses are not stored in the publishers' databases. Some important journal articles and conference papers that are stored in the databases were not returned through our particular queries. To make the set of prior work as complete as possible, we supplemented the sources returned by the searches with 29 additional sources that we believe are relevant. With these additional sources, the total number under consideration is 244.

We screened these sources to identify the relevant subset: those sources that describe the empirical evaluation of a keystroke-dynamics classifier and report the results. This screening excluded 74 sources; 170 sources remained. The majority of excluded sources were surveys which mentioned keystroke dynamics but did not conduct a technology evaluation. A few of the sources describe a new keystroke-dynamics classifier but no empirical evaluation is reported. Since our intent is to learn how often inferential statistics are used in keystroke-dynamics classifier evaluations, those sources that do not conduct evaluations are out of scope.

The full set of sources in the survey are listed separately in the references at the end of this document, under the heading *Keystroke-Dynamics Bibliography*. Table 2.1 presents a count of the number of within-scope sources obtained from each of the search databases.

	Article	Proceedings	Chapter	Masters Thesis	PhD Thesis	Tech Report	Total
ACM Digital Library	3	9	-	-	-	-	12
IEEE Xplore	10	74	-	-	-	-	84
Science Direct	13	2	-	-	-	-	15
SpringerLink	2	25	3	-	-	-	30
Other Sources	15	5	-	5	2	2	29
Total	43	115	3	5	2	2	170

Table 2.1: Breakdown of the number of sources found in survey. The total is cross classified by whether the source is a journal article, paper in a conference proceedings, book chapter, thesis, or technical report (column) and which database contained the source (row). Through experience, we knew of several relevant research efforts that were not in any of the databases, and these were added to the survey sample.

The database providing the most sources is IEEE Xplore, and most sources are conference papers.

For the 170 sources that reported the results of a keystroke-dynamics classifier evaluation, we assessed whether any statistical analysis was performed in order to draw inferences from the results. A paper was recognized as having performed a statistical inference if, in the section describing the evaluation results and analysis (including tables and figures), the researchers reported the results of a hypothesis test (e.g., a p-value), reported confidence intervals, or applied model-selection criteria such as AIC (Akaike's Information Criterion) or BIC (Schwartz's Bayesian Information Criterion). These statistical techniques are a principled way to draw general inferences from the results gathered during empirical evaluations.

A few sources used the word "significant" without, in our judgment, meaning it in a statistically precise way. For instance, a paper might claim that one technique "had a significantly higher error rate" than another (Alghathbar and Mahmoud, 2009), but unless a p-value or test procedure is reported, we would not recognize it as an instance of statistical inference. Likewise, when tabulating classifier error rates, several sources printed the best error rate in bold font or marked them with asterisks. In some disciplines, these conventions are used for results that are statistically significant, but in these sources, we did not recognize such font changes and symbols as signs of statistical inference without an additional description of the test.

In reviewing these sources, we discovered many that used inferential statistics when analyzing the keystroke-timing features. For instance, Gaines et al. (1980) used goodness-of-fit tests to ascertain whether down-down typing times are Normally distributed, and they used t-tests to determine whether typing changed after a six month interval. Modi and Elliott (2006) used mixed-effects models—the very models that we have proposed for this work—when selecting typing features to use with their classifier. Nevertheless, most of these researchers who used inferential statistics when analyzing typing times did not use them to analyze their classifier's behavior. For this survey, we are interested in whether statistical analysis was used to draw inferences from the classifier-evaluation results, and papers that only drew inferences from typing times were not recognized as having met the necessary criteria.

Also note that statistical procedures can be used correctly or incorrectly (e.g., failure to check assumptions or to account for multiple testing). However, our intent in this survey is to estimate how often statistical inferences are used *at all*, not to assess whether a particular inferential procedure was used correctly. We recognized papers as having performed statistical inferences even if the particulars of the analysis raise questions.

The sources were divided into those that used inferential statistics and those that did not. A source was recognized as using inferential statistics if it reported a hypothesis test, estimated confidence intervals, or performed model selection. As noted above, inferential statistics are necessary for understanding with any certainty whether a factor has an effect on the results, and if so, what effect it has. When authors used inferential statistics, we made a note of which techniques were used (e.g., t-tests, Kruskal-Wallace, or bootstrap confidence-interval estimation), which factors were studied, and what conclusions were drawn.

2.5 Survey results

We have categorized sources into three groups: those that used no inferential statistics, those that used inferential statistics to investigate one factor at a time, and those that used inferential statistics to investigate multiple factors. Table 2.2 presents a breakdown of the number of sources in each group. By far the largest group, containing 148 sources (87.1%), did not use any statistical analysis to draw inferences from the results.

The remaining groups, containing 22 sources, used some sort of statistical inferences to investigate classifier error rates. The second group, containing 17 sources (10.0%), used statistical analysis to investigate the effects of individual factors, one factor at a time. The final group, containing 5 sources (2.9%), used statistical analyses that enabled study not

	Article	Proceedings	Chapter	Masters Thesis	PhD Thesis	Tech Report	Total
No Inferential Statistics	30	108	3	3	2	2	148
One-factor-at-a-time Inferences	9	6	-	2	-	-	17
Multiple-factor Inferences	4	1	-	-	-	-	5

Table 2.2: Categories of inferential statistics found in literature survey. One group performed no inferential statistics on the classifier-evaluation results. The second group used statistical methods to make inferences about individual factors (e.g., two classifiers or different amounts of training but not both). The third group used statistical methods to investigate multiple factors and any possible interactions.

just of individual factors but also interactions between factors.

We review the studies in each group, and we discuss each of the papers in the last two groups. The authors of studies that belong in either of these two groups used inferential statistics to understand factors that affect keystroke-dynamics error rates. Since those authors have undertaken to solve a similar problem as in this dissertation, we have a responsibility to present and acknowledge their related efforts individually.

2.5.1 Group 1: No statistical inference

Based on this survey, most papers that evaluate keystroke-dynamics classifiers perform no statistical analysis to draw inferences from the results. These 148 sources report empirical error rates (e.g., miss and false-alarm rates or equal-error rates). As noted in the glossary of terminology (Section 1.5), the empirical miss rate is the fraction of impostor test samples classified as the genuine-user's; the empirical false-alarm rate is the fraction of genuine-user test samples classified as the impostor's.

While these studies did not use inferential statistics, many asked the kinds of research questions that we ask in this work. Some researchers, sometimes implicitly, acknowledged that a classifier's error rate depends on factors in the operating environment. In their evaluations, these researchers performed a series of trials, systematically varying factors of interest (e.g., the feature set or amount of training data) and reported the changing empirical error rates.

The purpose of inferential statistics would be to make general statements about the true error rates of the classifier (or classifiers) based on the empirical error rates. Unfortunately, these journal articles, conference papers, and other sources applied no inferential statistic to their empirical results. Two empirical error rates may differ even if the true error rates are the same.

In our experience, almost any change in evaluation conditions causes some change in the error rates. For instance, a classifier trained with 100 samples might have an empirical EER of 10%; at 101 samples, the empirical EER may drop to 1%; and, at 102 samples, the empirical EER returns to 10%. Have we discovered some *sweet spot* at 101 training repetitions? No, the drop is likely to be a result of random chance (unless further analysis proves otherwise).

When empirical error rates are used to support claims that one classifier is better than another, or some factor improves classifier accuracy, the researcher is making a general statement on the basis of the experimental results. Unfortunately, without support of a proper statistical analysis, these claims are unscientific. Even if they turn out to be true, the research effort did not provide a statistically sound argument in support of the claim. One might argue over the practical value of research without scientifically supported conclusions.

2.5.2 Group 2: One-factor-at-a-time inference

The 17 research efforts that drew inferences from one factor at a time used a variety of methods: t-tests, Pearson's χ^2 (chi-square) tests, 1-way ANOVAs, Wilcoxon signed-rank tests, and Kruskal-Wallis tests. Researchers used these tests to compare not only the classifiers themselves, but also different typing tasks, different keyboards, and even the effectiveness of artificial rhythms to increase a typist's consistency and distinctiveness.

Cho et al. (2000) supported their claim that an Auto-Associative Neural Network outperformed a Mahalanobis k-NN classifier using a t-test. They tuned both detectors so that the miss rate was zero, and they conducted a paired test on the false-alarm rates for each subject. They found the Auto-Associative Neural Network to be significantly better, with a p-value < 0.01.

Ngugi et al. (2011) investigated whether miss and false-alarm rates change with different typing tasks. Specifically, they had subjects type two different 4-digit numbers (i.e., 1234 and 1324) on the keypad. They conducted t-tests on the miss and false-alarm rates, and found a significant difference in the error rates between the typing tasks.

Bartlow and Cukic (2006) also used t-tests in their analysis of whether different typing tasks have different error rates. Their two typing tasks were (1) 8-character, lowercase English words and (2) 12-character passwords containing upper-case and special characters

(Bartlow, 2005; Bartlow and Cukic, 2006). According to their analysis, the long password had a lower error rates (false-alarm, miss, and EER) for multiple classifiers and most tunings of a random-forest classifier.

Hwang et al. (2009a,b) investigated whether classifier error rates can be reduced by priming the typists with rhythms to mimic. Subjects were asked to choose passwords with different properties (e.g., high familiarity or high randomness); subjects were grouped into one-handed and two-handed skill levels. In some typing tasks, the researchers provided rhythms for the subjects to imitate, while in others, the typists were allowed to use their natural rhythms. While this experiment involves many factors, in their analysis, the researchers used a series of t-tests. Each test considered only one factor at a time (e.g., EER with vs. without the artificial rhythms, or EER with familiar vs. random passwords).

In several investigations, Lee and Cho (2006, 2007) and Kang and Cho (2009) evaluated between 6 and 14 classification algorithms (in different studies) and used hypothesis testing to identify the one with the lowest error rates. They did not specify the precise test used, but they explain that the best classifier outperforms the others "at a significance level of 10%" (Kang and Cho, 2009, p.3131). This description suggests that they used either a 1-way ANOVA or an equivalent non-parametric technique.

To answer a similar question about whether one classifier was significantly better than the others in an evaluation, Giot et al. (2011) parametrically estimated 95% confidence intervals for each classifier (i.e., by assuming the error rates were distributed binomially). They also used Kruskal-Wallis to test whether different keyboards affected error rates and whether per-user thresholds reduced error rates. Neither were found to have a statistically significant effect.

In some work that launched the work in Chapter 4, Killourhy and Maxion (2009) used the Wilcoxon signed rank test (with a multiple-testing adjustment) when benchmarking 14 classifiers. Note that two of the classifiers evaluated in that research were re-implementations of the two classifiers evaluated by Cho et al. (2000); in the earlier work, the Auto-Associative Neural Net significantly outperformed the Mahalanobis k-NN, while in the more recent work, the reverse was true. We believed the discrepancy might be due to differences in the evaluation environment, leading to our current interest in identifying the factors that influence classifier error rates.

Curtin et al. (2006), Villani et al. (2006), and Tappert et al. (2010) used a χ^2 test in their investigation of several factors. Their subjects typed free and transcribed text on two different kinds of keyboards (i.e., desktop and laptop). By our reading, the authors conducted over 13 analyses for different combinations of factors. Within each analysis, the authors constructed cross-classification tables for the factors; in each cell of the table, the corresponding accuracy percentage was recorded. Pearson's χ^2 test was repeatedly used to test whether differences between cells, rows, and columns were statistically significant. Based on this series of tests, the authors concluded that classifier accuracy is higher (1) for laptop keyboards than desktop keyboards, (2) for transcribed text than free text, and (3) when the same keyboard is used for training and testing. Of note, they found a significant keyboard effect where Giot et al. (2011) did not. Perhaps the keyboard effect depends on other factors which varied between these investigations, bolstering our claim that error rates depend on a multitude of (potentially interacting) factors in the operating environment.

In their work, Epp et al. investigated whether keystroke dynamics could be used to identify the emotions of a typist (Epp, 2010; Epp et al., 2011). In this work, they used the κ (kappa) statistic to establish whether the output of the classifier was significantly better than random guessing. While the κ statistic is not technically the product of a hypothesis test, it is conventionally used to draw inferences. For instance, a κ score near the top of its range (1.0) is treated as strong evidence that a classifier is producing accurate labels, while a score near zero is evidence that the classifier would do no better than chance. The authors explained the critical values used to draw inferences. Based on their results, they were successful in identifying emotional states such as confidence and tiredness.

In biometrics research, subjects are informally grouped into categories named after animals. For instance, *goats* have higher-than-average false-alarm rates, making them seem inadvertently uncooperative, and *wolves* induce higher-than-average miss rates when they act as impostors for other users, making them seem threatening. Yager and Dunstone (2010) used Kruskal-Wallis tests to establish whether these and other animal groups truly exist in subject populations (e.g., whether goats' similarity scores are significantly lower than the average subject's). For keystroke dynamics, they found significant evidence that wolves are real.

All of these efforts used statistical analysis to make general claims from experimental results. In doing so, they provided concrete answers to research questions. These 17 research efforts should be recognized for providing a scientific contribution to our understanding of keystroke-dynamics classifiers. However, as we explained in Chapter 1, when researchers investigate one factor at a time, in isolation, they miss the opportunity to understand how that factor interacts with other factors. Simply by comparing the inferences drawn by different research projects, we have seen some conflicting results. To understand these discrepancies, we must look at more than a single factor at a time.

2.5.3 Group 3: Multiple-factor inferences

Our survey uncovered only 5 studies (out of 170) that drew statistical inferences from multiple factors. These researchers not only investigated multiple factors but also performed the statistical analysis necessary to make general statements about them. Two of these sources used confidence intervals or error bars, and three fit parametric statistical models.

Bartmann et al. (2007) evaluated their classifier systematically in a variety of different operating environments: with/without Shift-key features, various numbers of samples, various numbers of users, various numbers of impostors, and various lengths of the typing sample. For each of these factors, they used bootstrapping to estimate 95% confidence intervals and presented error bars in their plots. From these error bars, one can draw inferences about which factors have an effect on the error rates. Since some plots compare multiple factors, the error bars can be used to draw inferences about whether the factors interact. For instance, EERs with error bars are plotted for the classifier with/without Shift-key features and for various lengths of the typing sample. The error bars suggest that both factors have an effect, but they do not appear to interact. Shift-key features lower the error rate, and so does increasing the length of the typing sample. Together, these effects appear to be additive (i.e., with no interaction).

Everitt and McOwan (2003) developed a classifier that used both keystroke dynamics and mouse movements. They reported 95% confidence intervals with their empirical error rates across a range of factors. Using these confidence intervals, one can compare multi-factor interactions as well as the effects of individual factors. For instance, one factor is the mouse-movement task: "signing" their name vs. drawing a picture. Another factor is the level of impostor information: given no information about the genuine user's signature vs. given copies of the genuine user's signature and picture samples. The empirical average error rates suggest that the miss rate increased both when impostors had samples of genuine-user signatures and also when the input was a picture rather than a signature. However, all of the confidence intervals overlap, implying that these different conditions may not have much effect in subsequent evaluations.

The two papers by other researchers most related to the current work are by Mahar et al. (1995) and Napier et al. (1995). In one investigation, these researchers described and evaluated two classifiers (similar to the Outlier Count and Mahalanobis ones benchmarked in this work). They ran the evaluation twice: once using down-down times, and once using both hold and up-down times. They had 67 subjects. In serial, each subject was designated the genuine user, the others designated an impostor, and the empirical error rates recorded. To analyze their evaluation results, these researchers used repeated-measures

ANOVA. With this statistical technique, the per-subject error rates for each of the four conditions (i.e., 2 classifiers $\times$ 2 feature sets) are considered to be repeated measurements from the same subject. The effects of the classifier and feature set are estimated "within" each genuine-user subject, and the between-user variation is also estimated. They found significantly lower error with the Mahalanobis-like classifier and the hold and up-down times. They also found a significant interaction between the classifier and feature set, meaning that changing the feature set affected the error rates of the different classifiers differently.

The repeated-measures ANOVA used by these researchers is a form of linear mixed-effects model. In this work, we propose LMMs as a solution to the multitude-of-factors problem in keystroke dynamics, so one might ask whether this earlier work already solved the problem. While we recognize the important contribution these authors made, the scope of the current work is much broader. Napier et al. (1995) used repeated-measures ANOVA to investigate two factors, both concerning the classifier and its tuning (i.e., which features to use). In the current work, we consider many more factors, and the factors extend beyond the classifier and its configuration. We consider how different typing tasks and different impostor-evasion strategies affect the error rate. While they considered the per-user variation, we also consider the per-impostor variation. Investigating both sources of variation requires more complicated LMMs than repeated-measures ANOVA. Computation may be part of the reason why these earlier researchers could not do the sort of investigation we have undertaken. LMMs are computationally intensive, and crossed random effects (needed for estimating per-user and per-impostor variation) have only recently been incorporated into publicly available tools for statistical analysis (Bates, 2005).

Finally, in other research that lead directly to the current work, Killourhy and Maxion (2010) evaluated three classifiers in a series of operating environments, including different numbers of training samples, with/without updating, different feature sets (e.g., hold times and down-down times vs. hold times and up-down times). In that work, as in the current work, linear mixed-effects models were used to capture the effects of these factors on the error rate. Model-selection criteria—specifically BIC, which will be described in Chapter 3—were used to determine which factors and interaction terms had a substantial effect on the results. Much of that work is revisited in Chapter 6 of the current work. Unlike the earlier work, we investigate different typing tasks and we abandon EER as a performance metric in favor of miss rates, having tuned the classifier to have a 5% false-alarm rate (as justified in Chapter 4).

These earlier efforts to draw inferences regarding multiple factors are important predecessors to the current work. Together, they support our claim that a classifier's error rate depends on a multitude of factors in the operating environment. They have helped to identify some of these factors. They also suggest that LMMs may be key to identifying and understanding these factors. Of all the prior work in keystroke dynamics, these 5 papers provide the foundation on which the current work is built.

2.6 Related work on evaluation methodology

The current work concerns keystroke dynamics in general, and evaluation methodology in particular. Two prior efforts have been aimed at improving and standardizing keystroke-dynamics classifier-evaluation methodology. To put the current work in context, we consider the arguments and recommendations of these other papers. Their conclusions sometimes support and sometimes contradict our own recommendations.

Hosseinzadeh and Krishnan (2008) recommend the use of inferential statistics, just as we do in this work. The earlier authors explain two heuristics that are recommended as best practices for conducting biometric evaluations (Mansfield and Wayman, 2002): "The Rule of 3" and "The Rule of 30." The first rule is a heuristic that enables one to establish confidence intervals around an empirical 0% error rate. Specifically, when the empirical error rate is 0%, the upper bound of the 95% confidence interval is $3/N$ where N is the sample size. The second rule offers a more general heuristic for estimating confidence intervals when the results contain more than 30 errors. Specifically, under such situations, the true error rate is within 30% of the empirical error rate at a 90% confidence level. Beyond citing it as a best practice, the authors offer no evidence that applying these rules produces better results. Unfortunately, they do not appear to use these rules later in their paper to estimate confidence intervals for their own results.

Crawford (2010) sharply criticizes earlier work wherein flawed evaluation methodology "renders useless any reported results." Six recommendations are offered. Two of the recommendations concern keystroke dynamics evaluations in general and are discussed below, one is an endorsement of neural-network technology, and the three others concern the specific application of keystroke-dynamics to mobile-phone applications.

The first of the two relevant recommendations is that equal-error rates (EERs) should be reported alongside false-alarm and miss rates. We agree that the EER can be a useful single-number summary, enabling easy comparison of different classifiers. However, we caution against over-dependence on EERs. If two papers report EERs for different classifiers, one might be tempted to compare the classifiers using the EERs. However, if other factors differ in the two evaluations, such a comparison has confounded classifier and evaluation effects. Moreover, EERs are are one of many possible single-number summaries of miss and false-alarm rates, and they are not always the most useful. For this reason, the current work does not use EERs.

The author's second recommendation forbids the collection of genuine-user and impostor data from the same pool of subjects. A classifier trained and tested on the same impostors may have better performance than one trained and tested on different sets of impostors. One can avoid this source of bias by using different subjects as impostors. We appreciate the author's concern, but in some settings, using the same subjects' as genuine users and impostors does not bias the results. For instance, in the current work, classifiers are only trained on typing samples from the designated genuine-user subject. Typing samples from the other subjects are used to evaluate the classifier, but since they were not used in training, they do not bias the outcome.

These works (and our own) on keystroke-dynamics methodology are in agreement on at least one point: there is much room for improvement. We seek to improve understanding of classifier error rates by using inferential statistics on multiple factors. This topic is not considered in earlier efforts to improve keystroke-dynamics evaluation methodology.

2.7 Discussion

One of the key features of this review is the survey of statistical methods used in other keystroke-dynamics classifier evaluations. In that survey, we estimated that 87.1% of classifier evaluations do not use any inferential statistics at all. In many other scientific disciplines, researchers are required to use inferential statistics to justify conclusions. Clearly, in keystroke dynamics, no such requirements exist.

Nevertheless, we wish to be clear that our survey is not intended to criticize earlier researchers for not using inferential statistics to analyze classifier evaluation results. The shortcoming is in the current research methodology, not individual papers. Some of our own papers are among those which did not use any inferential statistics. When the standards are such that inferential statistics are not necessary, one cannot expect researchers to use them (or to always report the outcome even when they are used).

In other scientific disciplines, one sign of progress is that weaknesses of methodology are found and corrected. One famous example occurred with the discovery of the *Hawthorne Effect*, named after a study of worker productivity at Western Electric Company's Hawthorne site. An increase in worker productivity was eventually traced to workers' excitement and enthusiasm over being studied. The discovery spurred changes in the research methods of behavioral science (Shadish et al., 2002). Previous studies that did not account for the Hawthorne Effect were recognized as flawed. Future studies adjusted their methodologies to accommodate the effect.

The lack of inferential statistics is a shortcoming of keystroke-dynamics methodology (and in other areas of security research such as intrusion detection). As a result, conclusions drawn in papers that do not use inferential statistics may be flawed. Nevertheless, just as behavioral research methods changed with the discovery of the Hawthorne Effect, our intention in publicizing how few papers use inferential statistics is to influence future methodology, not to criticize existing papers.

2.8 Summary

In this chapter, we presented a brief history of keystroke-dynamics classification research, and we argued that inferential statistics are necessary to draw meaningful conclusions from classifier evaluations. We conducted a survey of 170 sources that evaluated classifiers, to understand how often inferential statistics were used. According to the survey, 148 sources (87.1%) used no inferential statistics at all. Even if such research draws conclusions from the empirical results of the evaluation, without proper statistical methods, we cannot be certain that the conclusions are supported by the data.

The remaining sources used some kind of inferential statistics, but 17 sources (10.0%) used analytical procedures appropriate for investigating one factor at a time. While certainly better than nothing, keystroke-dynamics classifiers may be affected by many factors and the interactions between them. Conclusions drawn about individual factors in isolation may only be of limited use. The remaining 5 sources (2.9%) did investigate multiple factors and their interactions. These prior research efforts were more limited in scope than the current effort, but they establish that multiple factors do matter.

Chapter 3

Linear Mixed-Effects Modeling

The previous chapter argued that keystroke-dynamics research must start using inferential statistics to understand classifier behavior. The current chapter offers linear mixed-effects models (LMMs) as the appropriate statistical technique for understanding the many factors that affect classifier behavior. We explain what LMMs are and provide some background on their development. A running example—using LMMs to understand which of 3 classifiers performs the best and whether hold times or down-down times are more useful features—will keep the explanation grounded in the practical application at hand. Finally, we explain some computational details and conventions for using LMMs in this work. These conventions ensure that a uniform statistical procedure is used throughout the work.

3.1 Background

A classic t-test could be used to establish whether one classifier's miss rate is significantly lower than another's. As discussed in Chapter 2, such an analysis was used by Cho et al. (2000). However, more advanced statistical methods are needed to handle data involving many classifiers, other factors, and interactions among factors. Linear mixed-effects models (LMMs) offer a powerful framework for analyzing complicated data sets where many different factors affect the experimental outcome (Pinheiro and Bates, 2000). LMMs are regularly employed when simpler methods are not adequate (e.g., when t-tests, analysis of variance, or linear regression cannot capture the complexity of the data).

An LMM may be thought of as the intellectual descendant of analysis-of-variance (ANOVA) models. In fact, as discussed in Chapter 2, a repeated-measures ANOVA is one form of LMM, though not the only one. In both ANOVA models and LMMs, one or more explanatory factors are related to a response variable. For instance, one can analyze the relationship between two explanatory variables—the classifier and the typist—and a response variable, the equal-error rate. Presumably some classifiers have lower EERs than others, but perhaps some typists are just harder for all classifiers to recognize correctly. Both ANOVA and LMMs can investigate how the two explanatory factors, classifier and subject, affect the EER.

Compared to ANOVA models, LMMs allow a greater number and variety of explanatory factors, and more complex relationships with the response variable. The *mixed-effects* descriptor indicates that explanatory factors fall into one of two categories: *fixed effects* and *random effects*. The difference between fixed and random effects is sometimes subtle, but the following rule of thumb is typically applied. If we primarily care about the effect of *each* value of a factor, the factor is a fixed effect. If we primarily care about the variation *among* the values of a factor, the factor is a random effect (McCulloch et al., 2008).

For instance, with keystroke-dynamics evaluation results, we treat the classifier as a fixed effect and the typist (i.e., the subject) as a random effect. If the experiment involves three classifiers, practitioners will want to directly compare the error rates of the three classifiers. Such a comparison can be used to identify the best-performing classifier, so a practitioner could then decide to deploy it in his or her environment. Since a comparison among the three actual values is desired, the classifier would be modeled as a fixed effect.

In contrast, the subjects in the evaluations are meant to be representative of other typists. If an experiment involves ten subjects, practitioners have less interest in comparing error rates between those particular ten subjects. Knowing which subject had the lowest (or highest) error rate is less useful to the practitioner than knowing how much variation exists across subjects. That variation can be used to predict how much per-user variability will be seen when the classifier is deployed. Since we desire an estimate of how the factor affects variability, the subject would be modeled as a random effect rather than a fixed effect. Ultimately, mixed-effects models can reveal which classifiers work best, how accurate they are, and whether there is substantial variation among the subjects.

LMMs—and their even-more versatile descendants called *generalized* linear mixed-effects models—are powerful statistical techniques used to understand complex behavior across scientific disciplines. In ecology, Bolker et al. (2009) used mixed-effects models to understand whether the number of fruits produced by a plant depends more on nutrient levels or genotype. In psychology, Baayen et al. (2008) used mixed-effects models to control experimental effects in understanding decision response times. Specifically, when subjects are presented with a series of words and asked to state which are made up, mixed-effects models can capture per-subject effects, per-word effects, and order-of-presentation effects.

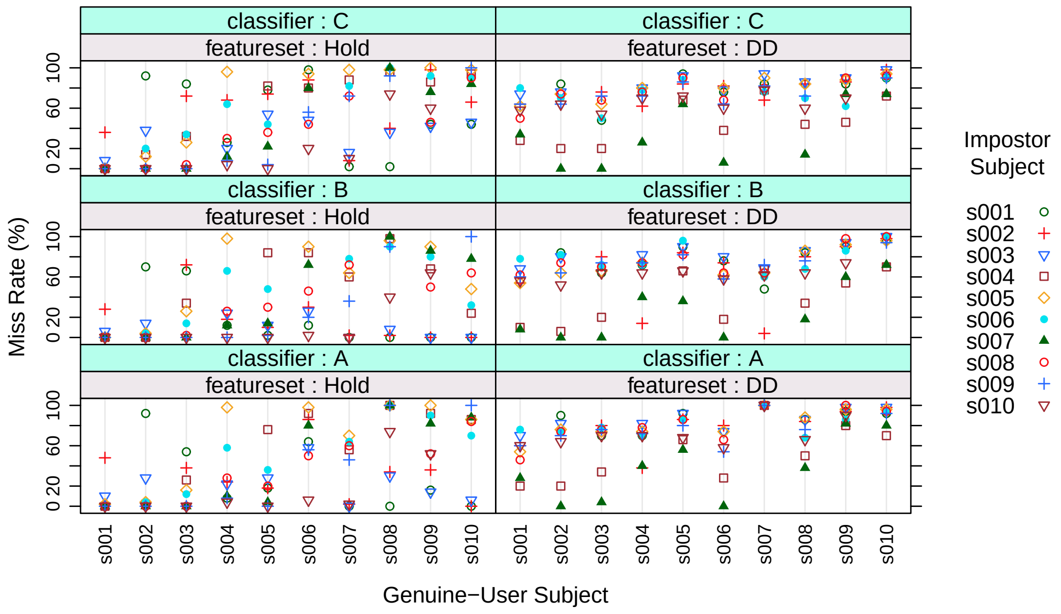


Figure 3.1: Example keystroke-dynamics classifier evaluation results. Classifiers A, B, and C were evaluated, once with hold times (Hold) and once with down-down (DD) times as features. Each run involved 10 subjects; one subject was treated as the genuine user, and the classifier was tested at recognizing the other subjects as impostors. The miss rate for each pair of genuine-user and impostor subjects are plotted. The graph shows the complexity of understanding the results. Any differences between the classifiers and feature sets are partly obscured by variation in the miss rate across users and impostors.

In pharmacology, Davidian and Gallant (1993) used LMMs to investigate the effects of birth weight and Apgar score on the blood concentration of Phenobarbital in newborns treated with the drug for seizures. As will be demonstrated throughout this work, mixed-effects models are equally suitable for understanding the behavior of keystroke-dynamics classifiers (and the influence of the evaluation environment on their behavior).

3.2 Example evaluation

In this section, we introduce an example evaluation that will help us to illustrate the use of LMMs and their application to understanding keystroke-dynamics evaluation results. Suppose that we are interested in understanding three classifiers. For the sake of simplicity, we will just refer to the three classifiers as A, B, and C. Suppose also that we have the option of using two different feature sets when analyzing keystroke dynamics: down-down times vs. hold times. We intend to find out which feature set works best.

An evaluation was conducted for each of the three classifiers and the two feature sets. Data was collected for the evaluation by having ten subjects type the same password (.tie5Roanl) repeatedly. Each subject typed the password 200 times. During data collection, both down-down and hold times were recorded for each repetition of the password.

Together, there were six pairings of classifier with feature set (3 classifiers $\times$ 2 feature sets). For each pairing, we ran 10 evaluation trials, one for each subject. Each trial corresponds to one combination of classifier (A, B, or C), feature set (hold or down-down times), and subject (Subject 1–10). The given subject is designated the genuine user and the classifier for that trial is trained on the first 100 typing samples from the genuine-user subject. The remaining 100 typing samples from the genuine-user subject were used to tune the classifier to have a 5% false-alarm rate.

Once trained and tuned, the classifier was presented with 50 samples from each of the other 9 subjects. These 9 subjects were designated impostors for the trial. We recorded whether each sample was recognized as having been typed by someone other than the genuine user, and for each impostor we calculated the miss rate as the fraction of the 50 samples that were not classified as impostor samples.

The accuracy of a classifier can be expressed in terms of misses and false alarms. These two dimensions of accuracy complicate analysis since a classifier can be tuned to have a lower false-alarm rate at the expense of a higher miss rate or vice versa. In this example—and in the remainder of this work—we tuned classifiers so that the false-alarm rate is fixed at 5%, and we analyzed the miss rates. As explained in more detail in Chapter 4, tuning classifiers to operate at a particular false-alarm rate makes more practical sense than tuning to a particular miss rate. In brief, genuine-user typing samples are easier to obtain than impostor samples, so tuning the classifier based on false-alarm frequency is easier.

Figure 3.1 presents these evaluation results as a lattice of dotplots. We ran 60 evaluation trials (3 classifiers $\times$ 2 feature sets $\times$ 10 genuine-user subjects), and from each trial, we record 9 per-impostor miss rates. Each panel corresponds to the results for one combination of classifier and feature set. Within a panel, each column corresponds to the results for one genuine-user subject. The symbol in each column denotes the impostor subject, and its vertical position is the miss rate. For example, when Classifier C was trained on Subject 1's hold times, and tested with the 50 samples from Subject 2, the miss rate was 48.0%. This combination of classifier (C), feature set (Hold), genuine-user subject (`s001`), and impostor subject (`s002`) corresponds to the red cross in the first column of the top-left panel of Figure 3.1.

The key observation to make from this figure is that miss rates vary a lot. Simply by looking at the figure, it can be difficult to tell whether one classifier or feature set is better than the others. Within every panel, some miss rates are near 0% and others are near 100%. We could ignore this variability if we looked at the average miss rate over all genuine-user and impostor subjects, for each combination of classifier and feature set. However, the variability itself seems worthy of study.

Keystroke dynamics may work better for some users than for others. Looking in any one panel, the miss rates for some genuine-user subjects are lower than others. For instance, in the top-left panel (Classifier C, Hold features), all the miss rates when Subject 1 is the genuine user are lower than all the miss rates when Subject 10 is the genuine user. Across panels, some genuine-user subjects tend to have lower-than-average miss rates, while others have higher-than-average ones. Subject 1's miss rates are usually low and Subject 10's are usually high, when each is the genuine user. With a somewhat unfocused eye, one can see a slope to the cloud of points in each panel, progressing from low on the left to high on the right. For this example, we assigned subject numbers so as to ensure this visual effect. In later chapters, we occasionally reorder the subjects and classifiers in plots to obtain a similar opportunity for visualization (e.g., sorting subjects from left to right in the dotplot panels based on average miss rate).

Keystroke dynamics may also work better against some impostors than others. Looking at the symbols in each panel, miss rates for some impostor subjects are usually lower than others. Some symbols more frequently appear toward the bottoms of panels, and others toward the tops. For instance, Subject 7's green triangle often corresponds to low miss rates, at least when down-down features are used, suggesting the subject would be comparatively easy to detect when impersonating other users. In contrast, Subject 2's red cross often corresponds to high miss rates, suggesting the subject is more likely to be missed when impersonating other users.

Typically, researchers do not present per-user or per-impostor error rates as we have. Instead, they present aggregate statistics such as a classifier's average miss rate across all genuine-user and impostor subjects. While those aggregate statistics can help to understand which classifier is best, they sometimes hide other important details. In this example, the average error rates would hide the fact that one should expect high variability even from the best classifier.

Having acknowledged that the raw evaluation results are difficult to interpret because of how much error rates vary across users and impostors, we can still try to compare classifiers and feature sets. The clouds of points in the panels on the left seem lower than the clouds in the panels on the right. As such, we might expect that hold times are better features for keystroke dynamics than down-down times. Likewise, the clouds of points in the middle row may be lower than the clouds in the top or bottom rows. As such, Classifier B may be more accurate than the other two classifiers. Such observations are not the same as a formal statistical analysis, but they suggest what we might expect to find through more formal analysis.

An interaction effect is even more difficult to observe in the raw evaluation results than classifier or feature-set effects. An example of an interaction effect would be if switching from down-down times to hold times improves the accuracy of Classifiers A and B *more* than the switch improves the accuracy of Classifier C. Such a classifier/feature-set interaction may be present in the evaluation results, but the effect is not readily apparent from Figure 3.1. Whether the interaction effect is absent or present and just obscured by the other effects requires further analysis.

This example evaluation and its results should demonstrate the difficulty of understanding the behavior of keystroke-dynamics classifiers. Other factors—including the feature set, the genuine user, and the impostor—can affect whether the classifier has a low or high error rate. Possible interactions between these other factors and the classifier add further complexity. To obtain a more complete understanding of classifier behavior and the influences of these other factors, we will use linear mixed-effects models (LMMs).

3.3 Variance-stabilizing transformation

When statistical models are used to estimate the effect of a factor (e.g., how changing from down-down times to hold times will affect the miss rate), they often assume that the effect may be hidden by the addition of some residual noise in the experimental results. An assumption made by many statistical models in general and LMMs in particular is that any residual noise is Normally distributed and equal across all values of a factor. For instance, the average miss rate may be higher for down-down times than hold times, but individual times will vary from the average by the same amount. In other words, the mean miss rates may be different but the noise in individual miss rates is assumed to be Normal and have equal variance for both down-down and hold times.

While linear models have been shown to be somewhat robust to assumption violations (Madansky, 1988), statisticians often employ transformations to make the data adhere closer to modeling assumptions. For instance, if modeling assumptions require that a response variable is Normally distributed and the variable is skewed, statisticians might apply a logarithmic transformation to the variable and study the transformed variable instead. This transformed response variable (i.e., the log of the original response variable) has a distribution closer to Normal than the original.

In our evaluation data, the response variables are miss rates and can be modeled using a Binomial distribution. A binomial distribution is appropriate for modelling proportions (e.g., y successes out of n trials). In our example evaluation, we had 50 samples from each impostor, so $n = 50$, and y is the number of misses; y/n is the miss rate. With binomial data, the variance changes with the mean, and so if two classifiers have different average error rates, the variances of those error rates may also differ, violating the equal-variance assumption of LMMs and other statistical models.

To accommodate the equal-variance assumption, analysts often employ a *variance-stabilizing transformation* (VST). For binomial data, the recommended variance-stabilizing transformation is

$$\mathrm{VST}(y/n) = \frac{200}{\pi} \arcsin \sqrt{y/n}.$$

In this formula, the error rate is represented as a proportion (i.e., y errors on n total samples). The kernel of the transformation is $\arcsin \sqrt{y/n}$. The scaling factor ($200/\pi$) is technically unnecessary, but it converts angles from radians to *grads*, a unit of measure that may be uncommon but helps with intuition in this case because grads range from 0 to 100 (Box et al., 2005).

Figure 3.2 illustrates the variance-stabilizing transformation as a function. The domain is the region [0,1] and the range is the region [0,100]. When y/n is near 0.0 or 1.0, the variance is much smaller than when y/n is near 0.5. This transformation stabilizes the variance by spreading values near 0.0 or 1.0 over a wider range of the output space than values near 0.5. This property of the variance-stabilizing transformation can be seen in its slope; the steep slope on the left and right sides of the figure mean that the extreme values of y/n are mapped to a wider range, while the comparatively shallow slope in the middle of the figure means that the middle values of y/n are mapped to a narrower range.

When analyzing our example evaluation data, we apply the variance-stabilizing transformation to the miss rates before employing LMMs. While this transformation accommodates the LMM equal-variance assumption, it somewhat sacrifices interpretability during the model-fitting process. Estimates of the factors' effects (e.g., different classifiers) are expressed in the transformed space. A change of $+5$ in the variance-stabilized space means a different change in the miss rate depending on the starting point before the change. For instance, if the starting miss rate is 0%, a change of $+5$ in the variance-stabilized space corresponds to a resulting miss rate of 0.6%; if the starting miss rate is 50%, a change of

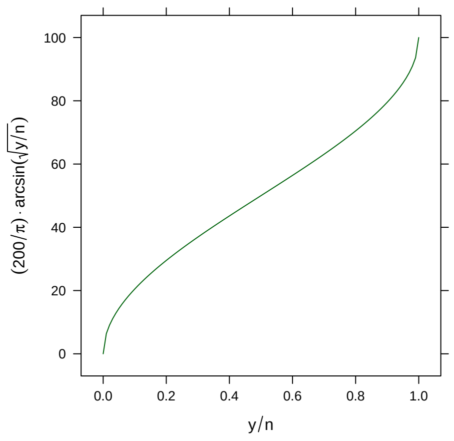


Figure 3.2: Variance-stabilization transformation for binomial random variables, where y is the number of successes in n trials. For binomials, the variance of y/n is smaller when y/n is near the ends of the 0.0–1.0 range than when it is in the middle of the range. A variance-stabilizing transformation can be used to spread the extreme values of y/n over a wider range, so the transformed variable has a more consistent variance across its whole range. For binomial random variables, the recommended variance-stabilizing transformation is depicted as a function.

$+5$ in the variance stabilized space corresponds to a resulting miss rate of 57.8%. This non-linearity can make effects in the variance-stabilized space difficult to interpret. Fortunately, in many cases, we can map results back to the original response-variable space (i.e., miss rates) by applying the inverse transformation:

$$\text{VST}^{-1}(x) = \sin^2(x \cdot \pi/200).$$

3.4 LMM #1: Classifier and feature effects

To better explain what LMMs are and how they work, we present a model built during an analysis of the example data from Section 3.2. First, the variance-stabilizing transformation was applied to the individual miss rates, and a model with four factors was built: the classifier, the feature set, the genuine-user subject, and impostor subject. The classifiers and feature sets were treated as fixed effects, and the genuine-user and impostor subjects were treated as random effects.

We will discuss shortly how to construct an LMM from the raw evaluation results, but first we explain what an LMM looks like after construction. Table 3.1 presents an LMM built from the example data. Like all LMMs, this model has two components: a

Model Equation:

$$\begin{aligned}
\mathrm{VST}(\textit{miss rate})_{ijklm} &= \mu + (\textit{Classifier})_i + (\textit{Feature Set})_j + (\textit{user})_k + (\textit{impostor})_l + \varepsilon_m \\
(\textit{user})_k &\sim N(0, \sigma^2_{(\textit{user})}) \\
(\textit{impostor})_l &\sim N(0, \sigma^2_{(\textit{impostor})}) \\
\varepsilon_m &\sim N(0, \sigma^2_{\varepsilon})
\end{aligned}$$

Parameter Estimates:

Parameters	classifier	feature set	estimate
(μ) baseline	A	Hold	40.72
$(\textit{Classifier})_i$	B		-6.79
	C		1.63
$(\textit{Feature Set})_j$		Down-Down	24.91
$\sigma_{(\textit{user})}$			12.98
$\sigma_{(\textit{impostor})}$			7.92
σ_{ε}			22.06

Table 3.1: LMM #1—Possible LMM built from example data. The LMM includes a model equation (top) and a parameter-estimate table (bottom). The model equation establishes the form of the probabilistic model and which factors have an effect on miss rates. The parameter-estimate table quantifies the effects of those factors. Section 3.4 describes this particular model in detail.

model equation and a *parameter-estimate table*. The notation in the model equation and parameter-estimate table may seem daunting at first, but we will walk through the process of reading and interpreting a model. In brief, the model equation expresses which factors have an effect on the response variable, and the parameter-estimate table explains what those effects are.

Model equation. A model equation consists of (1) an actual equation relating the response variable to various factors in the experiment, and (2) distributional assumptions for the random effects and residuals. The first line of the model equation in Table 3.1 presents the actual equation, relating the variance-stabilized miss rate to six terms: μ, $(\textit{Classifier})_i$, $(\textit{Feature Set})_j$, $(\textit{user})_k$, $(\textit{impostor})_l$, and ε_m. The μ and ε terms will be present in every model, representing the average miss rate and the residual noise, respectively. The other four terms represent fixed and random effects that have an effect on the average miss rate.

The presence of a term in the model equation means that it has *some* effect. For instance, the presence of the classifier term means that, based on this model, different classifiers have different average miss rates. If the classifier term were not in the model equation, the model would describe all classifiers as having the same miss rate. Likewise, the presence of the feature-set term, user term, and impostor term means that, based on this model, these factors also affect the miss rate. To know how much of an effect a factor has, we would have to consult the parameter-estimate table (described below).

The classifier and feature set are treated as fixed effects; the user and impostor are treated as random effects. Both fixed and random effects appears as terms in the model equation. Operationally, the difference between fixed and random effects is that fixed effects are modeled as constants while random effects are modeled as random variables. Consequently, for each random effect, the model equation includes a line describing the distribution of the random variable. Per-user effects are modeled as having a Normal distribution with zero mean and variance equal to $\sigma^2_{(user)}$. Per-impostor effects are modeled as having a Normal distribution with zero mean and variance equal to $\sigma^2_{(impostor)}$.

The residual-noise term (ε_m) can be thought of as a special kind of random effect. Like the per-user and per-impostor random effects, the residual noise is modeled as a random variable with a Normal distribution with zero mean and some variance (σ^2_ε). This term is special in that it is present in all LMMs (and many other statistical models); it represents the effect of all the uncontrolled and unknown factors that influence the miss rate in addition to those factors whose influence is made explicit as terms in the model equation.

The variances of each of the random effects are assumed to be greater than zero, meaning that the random effect introduces some amount of variability into the response variable. The presence of the per-user term means that error rates are systematically lower for some users and higher for others, across all classifiers, feature sets, and impostors. The presence of the per-impostor term means that error rates are systematically lower for some impostors and higher for others, across all classifiers, feature sets, and users. The model equation does not tell us *how much* lower or higher the error rate can be because of these factors. To know the magnitude of the variability, we would have to consult the parameter-estimate table.

Parameter-Estimate Table. Whereas the terms in the model equation tell us which factors have an effect, the parameter estimates in the parameter-estimate table tell us how much effect each factor has. In Table 3.1, the first line of the parameter-estimate table gives the estimated value of the baseline μ. This parameter is called the baseline because it denotes the average response for one particular combination of fixed-effect factor values, called the baseline values. In this example, we have two fixed effects: the classifier and the feature set. The baseline corresponds to Classifier A using hold times as features. The estimated average variance-stabilized miss rate for this classifier/feature-set combination is $\mu = 40.72$. By applying the inverse of the variance-stabilizing transformation, we can convert the baseline to an actual miss rate: $\mathrm{VST}^{-1}(40.72) =$ 35.6%.

The second and third lines in the parameter-estimate table both correspond to a single term in the model equation. In the model equation, the $(Classifier)_i$ term denotes that the classifier has an effect, but since there are three classifiers, the parameter-estimate table contains multiple estimates, one for each of the non-baseline classifiers. The line with B in the classifier column corresponds to Classifier B, and the estimate -6.79 represents the change to the variance-stabilized miss rate when Classifier B is substituted for Classifier A. Likewise, the line with C in the classifier column corresponds to Classifier C, and the estimate 1.63 represents the change when Classifier C is substituted for Classifier A.

As noted earlier, because of the variance-stabilizing transform, interpreting these estimates can be tricky without going through the inverse-transformation calculations. At a coarse level, the numbers give us some intuition about the estimated effects on miss rates. Since the estimate for Classifier B is negative, we know that the estimated miss rate for B is lower than A, and since the estimate for Classifier C is positive, we know that the estimated miss rate for C is higher than for A.

To get more specific, we need to map the effects back to miss rates. To calculate the estimated average miss rate for Classifiers B or C, one must add the corresponding effect to the baseline value and apply the inverse of the variance-stabilizing transformation. The average miss rate for Classifier B is calculated as $\mathrm{VST}^{-1}(40.72 + -6.79) =$ 25.8%. The average miss rate for Classifier C is $\mathrm{VST}^{-1}(40.72 + 1.63) =$ 38.1%. Note that these estimates apply with the baseline feature set (hold times).

The fourth line in the parameter-estimate table corresponds to the feature-set term in the model equation. Since only two feature sets were used in our evaluation—the baseline hold times and the alternative down-down times—only one parameter needs to be estimated: the effect of switching from hold to down-down times. This parameter estimate is 24.91, meaning that the variance-stabilized miss rate is expected to increase quite a lot if down-down times were substituted for hold times. For the baseline classifier, the miss rate is expected to increase to $\mathrm{VST}^{-1}(40.72 + 24.91) =$ 73.6%.

So far, we have only used the model to estimate the miss rate for the baseline combination of values and with a single substitution (e.g., from Classifier A to B, or from hold times to down-down times, but not both). To estimate the miss rate for Classifier B using down-down times with this model, we would add both the classifier and the feature-set effect to the baseline and invert the variance-stabilizing transformation. Specifically, we would calculate $\mathrm{VST}^{-1}(40.72 + -6.79 + 24.91) =$ 63.7%. Note that miss-rate estimates are dramatically worse for all classifiers when down-down times are used instead of hold times. Based on the relative magnitude of effects in the parameter-estimate table, we would conclude that the feature set (i.e., hold times vs. down-down times) has a bigger effect on the miss rate than the classifier, at least in this example.

For the random effects, the parameter-estimate table contains an estimate of the standard deviation (e.g., $\sigma_{(user)}$, $\sigma_{(impostor)}$, and σ_{ε}). The standard deviation is the square root of the variance, and we report standard deviations simply because we find them more intuitive than variances. For a Normal distribution with standard deviation σ, one can expect that 95% of values will be within $\pm 1.96\sigma$ of the mean. Consequently, we can use the estimated per-user standard deviation ($\sigma_{(user)} = 12.98$) to place prediction intervals around the classifier's average miss rates. For instance, for Classifier A, we can calculate an interval in which we expect that 95% of users' average miss rates will lie:

$$\text{VST}^{-1}(40.72 \pm 1.96 \cdot 12.98) = [5.6\%, 74.3\%]$$

Obviously, this interval is very wide; it spans nearly two thirds of the viable range (0%–100%). While somewhat discouraging for keystroke dynamics, this prediction confirms what we suspected when we looked at the lattice plot of the data in Figure 3.1. Miss rates depend as much on the user and impostor as on the classifier.

Just as with the per-user variation, we can estimate intervals based on the per-impostor variation ($\sigma_{(impostor)} = 7.92$). A per-impostor effect means that, regardless of the genuine user, some impostors are harder to detect than others. In other words, for every user, the miss rate is substantially higher for one impostor's samples than for another impostor's samples. We can estimate the effect of the per-impostor variation on Classifier A by substituting the per-impostor standard deviation into the Normal interval estimates:

$$\text{VST}^{-1}(40.72 \pm 1.96 \cdot 7.92) = [14.9\%, 59.7\%]$$

This interval is not quite as wide as the per-user interval, but it is still substantial.

Let us call attention to a point of possible misunderstanding. According to this model, the presence of a per-user and per-impostor effect means that different genuine-user and impostor subjects have different miss rates. This statement expressed by the model is stronger than the statement that, due to random chance, when we calculate per-user and per-impostor miss rates, some will be higher than others. Such happenstance differences in empirical miss rates are to be expected of any model. These effects are incorporated into the residual noise term. The presence of large per-user and per-impostor random effects

Model Equation:

$$\begin{aligned}
\mathrm{VST}(\mathit{miss\ rate})_{ijklm} &= \mu + (\mathit{Classifier})_i + (\mathit{Feature\ Set})_j \\
&\quad + (\mathit{Classifier} : \mathit{Feature\ Set})_{ij} + (\mathit{user})_k + (\mathit{impostor})_l + \varepsilon_m \\
(\mathit{user})_k &\sim N(0, \sigma^2_{(\mathit{user})}) \\
(\mathit{impostor})_l &\sim N(0, \sigma^2_{(\mathit{impostor})}) \\
\varepsilon_m &\sim N(0, \sigma^2_\varepsilon)
\end{aligned}$$

Parameter Estimates:

Parameters	classifier	feature set	estimate
(μ) baseline	A	Hold	38.52
$(\mathit{Classifier})_i$	B		-6.51
	C		7.96
$(\mathit{Feature\ Set})_j$		Down-Down	29.31
$(\mathit{Classifier} : \mathit{Feature\ Set})_{ij}$	B	Down-Down	-0.55
	C	Down-Down	-12.66
$\sigma_{(\mathit{user})}$			12.99
$\sigma_{(\mathit{impostor})}$			7.93
σ_ε			21.90

Table 3.2: LMM #2—Another possible LMM built from example data. This model equation includes not only a classifier and feature-set effect (as in Table 3.1), but also an interaction between the two factors. The parameter-estimate table includes estimates for these new effects. Section 3.5 describes this LMM in detail.

means that there are systematic differences in the average miss rates of different genuine users and impostors. Miss rates are uniformly higher for some users and impostors and systematically lower for other users and impostors.

The final term in the model equation (ε) is the noise term representing the unknown influences of additional factors on the miss rate. Like the random effects, ε is a Normally distributed random variable. Its variance, σ^2_ε, expresses a measure of the residual uncertainty in the miss rate. Looking at the parameter-estimate table, the residual standard deviation in this model is 22.06. Comparing the three standard deviations ($\sigma_{(user)}$, $\sigma_{(impostor)}$, and σ_ε), we see that the residual standard deviation is higher than the others. Based on this model, there are substantial per-user and per-impostor effects, but a great deal more residual noise remains; the residual noise represents the sum effect of other uncontrolled or unknown factors that also influence the miss rate.

3.5 LMM #2: Classifier/feature-set interaction

When we looked at the example data in Section 3.2, we acknowledged that different classifiers might be differently affected by a switch from hold times to down-down times. Looking back at Figure 3.1, we observed that hold times might reduce the miss rate for Classifiers A and B more than C. The model we presented in Section 3.4 is unable to express such a dependency between the effect of changing classifiers and changing the feature set. For every classifier, substituting down-down times for hold times increased the variance-stabilized miss rate by the same amount; the feature-set effect is the same for every classifier. To express a feature-set effect that is different for different classifiers, we must introduce an interaction term into the model equation.

Table 3.2 presents a second LMM. This LMM differs from the previous one in that the model equation contains a $(\mathit{Classifier} : \mathit{Feature\ Set})_{ij}$ term, and the parameter-estimate table has new entries related to this term. This new term denotes a classifier/feature-set interaction. The presence of this term in the model equation means that the effect of switching the feature set *depends* on the classifier. (Equivalently, the effect of switching the classifier *depends* on the feature set.)

In the model equation, the other terms have the same meaning as in the previous LMM. The presence of a term means that the corresponding factor has *some* effect. Likewise, in the parameter-estimate table, the other parameter estimates have the same meaning as in the previous LMM (though the estimated values themselves may differ between the models). The baseline (μ) is still an estimate of the variance-stabilized miss rate for the baseline values of the fixed effects: Classifier A with hold-time features. The two estimates for $(\mathit{Classifier})_i$ still denote the change in the variance-stabilized miss rate when Classifiers B or C are substituted for Classifier A. The estimate for $(\mathit{Feature\ Set})_j$ still denotes the change when down-down times are substituted for hold times.

However, when Classifier B or C is substituted for Classifier A *and* down-down times are substituted for hold times, the change is not simply the sum of the classifier effect and the feature-set effect. The new estimates for $(\mathit{Classifier} : \mathit{Feature\ Set})_{ij}$ denote adjustments to the sum of the classifier and feature-set effects when substitutions are made to both the classifier and feature set. The adjustment for substituting Classifier C for A *and* down-down for hold times is -12.66, a large negative value. To use this new model to estimate the miss rate for Classifier C and down-down times, we must include the interaction term in the calculation. Specifically, we start with the baseline estimate and add, not only the Classifier C estimate, and down-down estimate, but also the Classifier C /

Down-Down feature-set interaction estimate. Specifically, the miss rate is estimated as $\text{VST}^{-1}(38.52 + 7.96 + 29.31 + -12.66) = 70.0\%$. The negative adjustment from the classifier/feature-set interaction means that, when down-down times are used instead of hold times, the increase in the miss rate is *less* for Classifier C than for Classifier A (but still substantial).

The first LMM, presented in Section 3.4, assumed that the feature set did not interact with the classifier in determining the miss rates. Since there was no term in the model equation for the classifier/feature-set interaction, there could be no interaction. This second LMM, has terms that enable it to express a classifier/feature-set interaction. One might ask whether this second, more complicated model is better since it is more expressive. The challenge of choosing among multiple possible models—balancing model simplicity and expressiveness—will be addressed shortly when model-selection criteria are discussed. First, we will discuss how to construct an LMM model equation and parameter-estimate table from a set of evaluation results. Then, we will describe the model-selection procedure.

3.6 Parameter estimation

The parameter-estimate table is typically created after the model equation is selected, but it is easier to describe this step first. We simply have to assume that a model equation has already been selected. Given a model equation like either of the ones presented already, an analyst has a variety of options for estimating the parameters (e.g., μ, $(\mathit{Classifier})_B$, $(\mathit{Classifier})_C$, $(\mathit{Feature\ Set})_{\mathsf{Hold}}$, $\sigma_{(user)}$, $\sigma_{(impostor)}$, and σ_ε) from the data. A traditional parameter-estimation approach, and the one used in this work, is maximum-likelihood estimation. From any set of parameter estimates, one can calculate the probability of the data given those estimates; this probability is called the *likelihood* of the data. The likelihood calculation can be considered a function of the parameter estimates, and the maximum-likelihood estimates are those parameter estimates for which the likelihood function achieves its maximum value (Searle et al., 2006).

For simple model equations, one can derive a maximum-likelihood solution to the probability density equations through mathematical analysis (i.e., taking derivatives of the likelihood function, setting them to zero, and solving for all the parameters). However, for most mixed-effects model equations, the solution must be obtained through iterative optimization and numeric integration procedures. A more complete discussion of these computational issues will be presented in Section 3.9.

Note that maximum-likelihood methods have been shown to produce biased estimates of the variance parameters (e.g., σ_s^2). The favored method is a slight elaboration called REML estimation (for restricted or residual maximum likelihood) which addresses the bias in maximum-likelihood estimation (Pinheiro and Bates, 2000). For this work, we use REML estimates whenever possible. As detailed below, maximum-likelihood estimates must be used during model selection.

Maximizing the likelihood function is not the only approach to parameter estimation. Bayesian approaches to parameter estimation are also possible for LMMs, although when taking that approach LMMs are usually just called *hierarchical models*. The distinction between fixed and random effects is murky (or even murkier) in a Bayesian framework (Gelman, 2004). Concerns over the philosophical differences between likelihood-based and Bayesian approaches to parameter estimation are mitigated by the fact that the actual estimates are often very similar (Wasserman, 2003). Maximum-likelihood methods are considered more traditional and less controversial; in this work, we use them exclusively.

3.7 Model selection

When outlining the parameter-estimation procedure in the previous section, we assumed that a model equation had been given. A separate procedure is needed to determine which of many possible alternative model equations offer the best fit. For example, in our running example, we have presented two model equations: one with a classifier/feature-set interaction (Section 3.5), the other without (Section 3.4). We need a procedure for choosing which of these (and other) model equations is best.

The model selection procedure we use begins by identifying a family of possible models for consideration, and then searching the family for the model that is best supported by the data. In this section, we describe this procedure.

3.7.1 Constructing a family of models

We use a fairly common procedure of crafting two model equations—one simple and one complex—that act like lower and upper bounds defining a family of model equations. Specifically, a model equation m is in the family if and only if (1) all the terms in the simple model equation are also in m, and (2) all the terms in m are also in the complex model equation. In this way, the simple equation is analogous to a lower bound on the models in the family, while the complex equation is analogous to an upper bound. Establishing such boundary equations is a standard practice during *stepwise model selection*, the name

Lower Bound: Simplest Model Equation

$$\begin{aligned}
\text{VST}(\textit{miss rate})_{ijk} &= \mu + (\textit{user})_i + (\textit{impostor})_j + \varepsilon_k \\
(\textit{user})_i &\sim N(0, \sigma^2_{(\textit{user})}) \\
(\textit{impostor})_j &\sim N(0, \sigma^2_{(\textit{impostor})}) \\
\varepsilon_k &\sim N(0, \sigma^2_{\varepsilon})
\end{aligned}$$

Upper Bound: Most Complex Model Equation

$$\begin{aligned}
\text{VST}(\textit{miss rate})_{ijklm} &= \mu + (\textit{Classifier})_i + (\textit{Feature Set})_j \\
&\quad + (\textit{Classifier} : \textit{Feature Set})_{ij} + (\textit{user})_k + (\textit{impostor})_l + \varepsilon_m \\
(\textit{user})_k &\sim N(0, \sigma^2_{(\textit{user})}) \\
(\textit{impostor})_l &\sim N(0, \sigma^2_{(\textit{impostor})}) \\
\varepsilon_m &\sim N(0, \sigma^2_{\varepsilon})
\end{aligned}$$

Figure 3.3: Two model equations that define a family of model-selection candidates. When performing model selection, an analyst must define a family of candidate models to be, in effect, the search space. A family can be defined by two bounding equations. A simple equation acts as a lower bound on the family of candidate models; all models in the family must have all the terms in the lower-bound equation. A complex equation acts as an upper bound on the family; no model in the family can contain terms not in the upper-bound equation.

for the procedure we employ in our analysis (Venables and Ripley, 1997).

For instance, continuing with our running example, we might use the simple and complex model equations in Figure 3.3. The simple equation in the figure describes a model in which the only factors that affect the miss rate are per-user effects $(\textit{user})_k$ and per-impostor effects $(\textit{impostor})_l$. The complex model equation includes not only those random effects but also the fixed effects for the classifier $(\textit{Classifier})_i$, the feature set $(\textit{Feature Set})_j$, and their interaction $(\textit{Classifier} : \textit{Feature Set})_{ij}$.

According to the minimal model equation, the probability of a miss would be the same regardless of the classifier. The only sources of variation are the genuine-user subject on whom the classifier was trained and the impostor subject whose sample was used to test the classifier. One might ask whether an even simpler model might be used as a lower bound. The simplest model would contain only one term, the baseline (μ), so why require all models in the family to include per-user and per-impostor random effects? On some level, the range of models under consideration is the prerogative of the analyst. In this case, we choose to include the per-user and per-impostor random effects in all models because they are present in the experimental design. Not including these random effects would be akin to treating samples from the same subject as independent when in fact they are surely correlated—a statistical fallacy called *sacrificial pseudoreplication* (Hurlbert, 1984).

Specifically, if the per-user and per-impostor terms were not in the model, the parameter estimation and subsequent analysis would treat the evaluation results as equivalent to one in which all 90 impostor attempts (10 genuine-user subjects $\times$ 9 impostor subjects) came from different subjects. Much stronger inferences can be made when data are collected from 90 subjects than from 10 subjects, and we believe it is a mistake to treat the two kinds of experiment as equivalent in the event that our model-selection criteria finds no evidence of per-user and per-impostor effects. While different analysts make different choices with respect to this issue, our position is reasonably common among analysts (Bolker et al., 2009).

According to the maximal model equation, the classifier and feature set have effects, and their effects interact. One might ask whether an even more complex model might be created by creating a classifier/user interaction or a user/impostor interaction as well as the classifier/feature-set interaction. Such higher-order interactions among random effects and interactions between fixed and random effects are certainly possible. However, there are both practical and theoretical issues with models containing such terms. Estimating random effect terms involves estimating a large number of intermediary effects, and estimating interactions involving random effects often involves a combinatorial explosion in the number of these intermediary effects. For instance, a user/impostor interaction requires estimating an intermediary effect for every combination of user and impostor. There would effectively be one such estimate per evaluation result (i.e., per user/impostor miss rate). When the number of effects is of the same magnitude as the amount of data, the parameter-estimation procedure becomes both slow and unreliable. In theory, these estimates will have high variance and are likely to be wildly inaccurate. Because of the practical and theoretical issues, we do not consider models with interacting random effects in this work.

Strictly speaking, we should not refer to these intermediary per-user, per-impostor and per-user/impostor effects as *estimates*, per se. To be more precise, we should call them *best predictions*, since they are predicted values of random variables, not estimates of parameters (Searle et al., 2006). The need for such nuanced terminology rarely arises in this work, but we acknowledge the difference between estimates and predictions for the readers attuned to the subtle distinction.

Like the decision to treat an effect as fixed or random, the analyst has some discretion when choosing the bounds of the model family. For this work, the two random effects are the genuine user and the impostor, and based on the theoretical and practical considerations, we always include these random effects in our models, and we never model interactions involving them. More complicated models may be considered in the future. In the meantime, the proof that our current model families are acceptable will be seen in the success of the validation step (described in Section 3.10).

Note that we adopt a *principle of hierarchy* when working with a family of models. According to this principle, if a higher-order interaction among fixed effects is part of a model, then we also include all the lower-order fixed effects involving the same factors. For instance, we cannot have a classifier/feature-set interaction term in a model without also including individual classifier and feature-set terms. This principle is applied partly for theoretical and partly for pragmatic reasons. For some experimental designs, parameter-estimation procedures break when the principle is violated. Specifically, parameter estimation involves solving systems of linear equations, and when the principle of hierarchy is violated, relevant matrices sometimes become singular. Even when computational issues do not prohibit the principle of hierarchy, we observe it because interaction effects are often more easy to interpret in conjunction with main effects (Faraway, 2006).

3.7.2 Selecting the best model in a family

Having constructed a family of candidate models, we need to identify the best model in the family. Various criteria have been proposed for comparing models. The one we adopt is Schwartz's Bayesian Information Criterion (BIC). As discussed by Yang (2005), this criterion has the desirable property that, in the limit (i.e., as the amount of data increases), the BIC will correctly select the *true* model from a family of models if the true model is in the family.

Intuitively, the BIC offers a trade-off between model expressiveness and simplicity. A more expressive model provides a better fit for the data. With BIC, the fit is measured in terms of the likelihood function. Recall that maximum-likelihood parameter estimates are those which maximize the likelihood of the data. Of two models, the one whose maximum-likelihood parameters produce a higher likelihood offers a better fit for the data. However, a better fit often comes at the expense of simplicity. With BIC, simplicity is measured in terms of the number of parameters. A model with fewer parameters is simpler, while a model with more parameters is more complex.

The specific equation for calculating BIC is as follows (Hastie et al., 2001):

$$BIC = -2 \cdot L + d \ln n$$

where L is the log-likelihood of the model, d is the number of parameters, and n is the number of observations. The first term ($-2 \cdot L$) is a measure of model fit, and the second term ($d \ln n$) is a measure of simplicity. To calculate the BIC, we perform parameter estimation (as described in the last section), record the logarithm of the maximum of the likelihood function (L), count the number of parameters in the parameter-estimate table (d), and note the amount of data (n).

A lower BIC score is preferable. Suppose we had to choose between a model with few parameters and a model with many parameters. The model with more parameters (higher $d \ln n$) would typically offer a better fit (lower $-2 \cdot L$). By comparing BICs, we decide whether the better fit of the more complex model outweighs the additional complexity.

Let us note one procedural issue when performing BIC-based model selection using mixed-effects models. REML estimation is incompatible with the BIC heuristic because the likelihood calculations produced from REML estimates have been transformed in such a way that likelihoods cannot be compared across models. When comparing two models using BIC, the maximum-likelihood estimates are used. Once a model is selected, the parameters are re-estimated using REML. Intuitively, we use the maximum-likelihood estimates because, despite their bias, they allow us to do model selection. Then, once we have chosen a model, we can switch to the better REML estimates. This series of steps is common practice in LMM analysis (Zuur et al., 2009).

Returning to the running example, we presented two models in Sections 3.4 and 3.5; they differed in that one contained a classifier/feature-set interaction while the other did not. We raised the question of which of the two models was best supported by the data. The BIC score for the model from Section 3.4 (without the classifier/feature-set term) is 4962.40; the BIC score for the model from Section 3.5 (with the classifier/feature-set term) is 4965.37. The first score is lower, and so model selection would choose the model with no classifier/feature-set interaction. Based on the analysis in this example, we would conclude that there was insufficient evidence to support a classifier/feature-set interaction. (Note that this example is only for illustrative purposes. The investigation in Chapter 6 revisits this issue using more data, and we do find that a model with a classifier/feature-set interaction term has the lowest BIC.)

The size of the family of candidate models can grow quite large if there are a lot more terms in the maximal equation than the minimal one. The size of the family is exponential in the number of terms, which is itself exponential in the number of factors. For these large families, it is common to employ stepwise model selection. In particular, we start with the maximal model equation, and we calculate the BICs of all models generated by removing one term from the maximal model. Whichever model has the lowest BIC, we use to start the next iteration. We calculate the BICs of all models generated by removing one term from the model chosen in the previous iteration, and choose the one with the lowest BIC. The process iterates until we arrive at the minimal model equation. From the models encountered in this search, we choose the model with the lowest BIC overall. This iterative procedure is called *stepwise model selection* (Venables and Ripley, 1997).

Stepwise approaches such as this one have the undesirable property that they are, in essence, greedy with no assurance that the greedy approach will arrive at the optimal solution. However, for large families of candidate models, a greedy heuristic is the only way to make the search tractable. In this work, we have had the opportunity to compare the full-search and stepwise model selection approaches in some cases where both are tractable (Chapters 4 and 5). In all cases, the stepwise approach found the same model as the full-search approach. That finding gives us some confidence that the stepwise approach finds a good model even without optimality guarantees.

Once the model family is defined using minimal and maximal model equations, our model-selection procedure is largely automated. Many sources advise against an entirely automated model-selection procedure (Burnham and Anderson, 2002). The common concern with a fully automated process is that one blindly trusts the correctness of the selected model, ignoring the other models that may offer nearly the same explanatory power. An analyst often has expertise that can be used to guide model selection in a sensible direction. However, since this research is in its early stages, we are not able to rely much on prior experience. Instead, we rely on the validation step to ensure that the LMM produced through this largely automated process is useful for predicting the results of subsequent evaluations. In the future, researchers might use our models as starting points, and the community will develop the necessary expertise to dispense with automated procedures.

3.8 Testing hypotheses

So far, we have explained how to find an appropriate model to describe the evaluation results (i.e., the effect of factors such as the classifier and feature set on the miss rate). Now, we explain how we use such a model to draw conclusions about classifier behavior. Specifically, we will be interested in answering two kinds of research questions. The first kind of question is whether a particular factor (e.g., the feature set) has a significant effect on miss rates. The second is which values of a factor (e.g., which classifiers) have the lowest miss rate.

We use traditional statistical hypothesis testing to answer both kinds of questions. In both cases, we employ the notion of a *null hypothesis* and *statistical significance* (Dodge, 2003). In the first case, testing whether a factor has an effect, the null hypothesis is that different values of the factor (e.g., different feature sets) do not differently affect the miss rate. In the second case, identifying which values of a factor (e.g., which classifiers) have the lowest miss rate, there are multiple null hypotheses, one for each pair of classifiers. For two classifiers, A and B, the null hypothesis is that their miss rates are the same.

Hypothesis testing is relatively simple in theory but complicated in particular cases. One must find an appropriate test statistic from the data and then derive the distribution of that statistic under the null hypothesis (Casella and Berger, 2002). If the test statistic has a sufficiently small *p-value*—meaning that the statistic is sufficiently far into the tail of its distribution—we reject the null hypothesis (e.g., such extreme values occur with less than 5% probability). In our case, finding an appropriate test statistic is relatively easy; deriving its distribution and corresponding p-value are comparatively difficult.

The relevant test statistics are simple arithmetic combinations of the parameter estimates and their standard errors. For example, to test whether the feature set has a significant effect, an appropriate test statistic would be the estimated effect divided by its standard error. In the first LMM (from Section 3.4), the estimated effect was 24.91. The *standard error* of the estimate—which we have not described in detail, but which is also calculated during parameter estimation—is 1.90. The test statistic is $24.91/1.90 = 13.12$).

With simpler modeling techniques than LMMs (e.g., t-tests and ANOVA), this same test statistic arises during hypothesis testing (Weisberg, 2005). When those simpler modeling techniques are used, the test statistic can be shown to have a t distribution with a degrees-of-freedom parameter that can be calculated from the data. Unfortunately, with LMMs, the test statistic does not have a t distribution with an easily calculated degrees-of-freedom parameter (Bates, 2006).

When the precise distribution of the test statistic cannot be derived analytically, as in this case, the analyst has various options. A typical option, which we employ, is to find its *asymptotic distribution* and use that. Because of the central limit theorem, as the amount of data increases (e.g., the number of subjects), the distribution of many test statistics converge to standard Normal distributions. Statisticians conventionally recommend around 30 subjects to justify asymptotic arguments (Wackerly et al., 2002). The minimum number of subjects in the investigations in this work is 26, within range of the recommended number.

A further complexity arises when we perform many hypothesis tests at once. We will have two reasons to perform multiple tests. We intend to answer multiple questions within each investigation (e.g., one test for each potentially influential factor). Also, some questions require multiple tests to answer. For instance, to identify the top performing classifiers among the three in the example, we must make three comparisons: (1) between A and B, (2) between A and C, and (3) between B and C. In general, when a factor takes more than two values, and we wish to identify which value (or values) result in the lowest miss rates, we must make comparisons between all pairs of values.

Multiple testing is a problem because the standard hypothesis testing framework aims to limit the probability of mistakenly rejecting the null hypothesis (called a *Type I error*). Specifically, if the null hypothesis is true, we intend to limit the probability of a mistaken rejection to 5%. However, if we make three comparisons, and for each one we have limited the probability of a Type I error to 5%, the overall probability of at least one Type I error may be higher than 5%. For instance, if the probability of a mistake is 5% for each of three comparisons, and the three probabilities are independent (for the sake of simplicity), then the overall probability of making at least one mistake is 14.3% (i.e., $1 - (1 - .05)^3$).

Multiple testing issues have been studied extensively; different options are available, depending on the particular statistical model and the kinds of questions being asked (Miller, Jr., 1981). In our work, all of the questions can be framed in terms of testing a null hypothesis that a particular linear combination of parameters is equal to zero. For instance, referring back to Table 3.1, the LMM has 4 fixed-effect parameters: μ, two parameters corresponding to $(Classifier)_i$ (one for B and one for C), and one $(Feature\ Set)_j$ parameter. For the sake of this discussion, let us denote the four parameters as μ, $(B - A)$, $(C - A)$, and $(\mathsf{DD} - \mathsf{Hold})$. This naming scheme follows from the logic that the $(B - A)$ parameter represents the effect of substituting Classifier B for A.

To test the null hypothesis that there is no difference in the miss rates of Classifiers A and B, we would test whether parameter $(B - A) = 0$. Analogously, to test the null hypothesis that there is no difference in the miss rates of Classifiers C and A, we would test whether parameter $(C - A) = 0$. To test the null hypothesis that there is no difference in the miss rates of Classifiers C and B, the situation is more complicated but only slightly. The effect of substituting C for B is just the sum of substituting A for B and then C for A. So, the null hypothesis is $(C - A) - (B - A) = 0$. In all cases, the null hypothesis tests whether a linear combination of the parameters is zero.

When multiple hypothesis tests involve only testing linear combinations of the parameters, the set of tests can be organized into a matrix, each column representing one of the parameters, and each row corresponding to a test (i.e., a linear combination of the parameters). This matrix is called a *contrast matrix*. When the tests can be organized into a contrast

	μ	$(B-A)$	$(C-A)$	$(\mathsf{DD}-\mathsf{Hold})$
classifier: B - A	0	1	0	0
classifier: C - A	0	0	1	0
classifier: C - B	0	-1	1	0
featureset: DD - Hold	0	0	0	1

(a): Contrast matrix

	effect	stderr	t.stat	p.value
classifier: B - A	-6.787	2.325	-2.919	0.0133
classifier: C - A	1.629	2.325	0.700	0.8778
classifier: C - B	8.416	2.325	3.619	0.0012
featureset: DD - Hold	24.907	1.899	13.118	<.0001

(b): Test results

Table 3.3: Hypothesis tests comparing different classifiers and feature sets. Panel (a) presents the contrast matrix that represents the four tests. Note that each test can be represented as a linear combination of the parameters. Panel (b) presents the results of the four tests: p-values, effect size, estimated standard error, and t-statistic. If a p-value $< .05$ is judged significant, this set of tests finds (1) little difference between classifiers A and C, (2) a significant difference between those two classifiers and classifier B, and (3) hold times offer a significant reduction (statistically and practically) in miss rates.

matrix, one can use the multiple-testing solution provided by Hothorn et al. (2008). Given a contrast matrix, this approach uses the parameter estimates, their standard errors, and the correlations between those estimates to correct the p-values for multiple testing. In other words, the p-values of each test are adjusted so that if they are all compared to a threshold significance level of 5%, the probability of making even one Type I error across all the tests is limited to 5%.

There are some standard contrast matrices for testing particularly common sets of hypotheses. In particular, *Tukey contrasts* are often used when a factor takes more than two values (e.g., 3 different classifiers), since they test each pair of values for a significant difference (e.g., Classifiers A vs. B, B vs. C, and A vs. C). We will use Tukey contrasts when identifying the top performing set of classifiers. Not every test corresponds to a set of standard contrasts, and in some cases, we will be forced to write our own linear combination of parameters. The manually-created contrast matrices are described as best possible in the text. They are also provided in the online supplement to this work (described in Chapter 7).

To ground this discussion in our running example, Table 3.3 presents a contrast matrix and the results of four simultaneous hypothesis tests. Three of the tests compare the three classifiers using Tukey contrasts to compare each pair of classifiers. The fourth test compares down-down and hold times. Panel (a) presents the contrast matrix for this set of tests. As described, there is one column for each parameter and one row for each test. Each test is encoded as a linear combination of parameters, and the null hypothesis is that the combination equals zero.

In Panel (b), the test results are tabulated. The estimated value of the parameter combination is given along with its standard error. The t statistic (i.e., the test statistic) is the ratio of the estimate and its standard error. The p-value in the last column has been adjusted for multiple testing (i.e., they are larger than they would be if any one of the tests were conducted in isolation).

Based on this table, we would conclude that Classifier B is better than Classifiers A or C. The p-values when B is compared to each of the other two classifiers are below $.05$. Since the p-value for the comparison of Classifiers A and C is so high, there is little evidence that the two classifiers are different (i.e., we retain the null hypothesis for now). Regarding the feature set, the effect of switching from hold times to down-down times is highly significant (with the p-value calculated as below 0.0001). While the test results in this section are presented for illustrative purposes (not scientific conclusions), the analysis would support our initial observation that using hold times instead of down-down times has more effect on miss rates than the choice of classifier.

3.9 Computational details

All analyses in this work used the following software: the R statistical programming environment, version 2.13.1 (R Development Core Team, 2008); the `lme4` mixed-effects modeling package, version 0.999375-41 (Bates, 2005); and the `multcomp` multiple-comparison package, version 1.2-7 (Bretz et al., 2011).

As noted in Section 3.6, parameter estimation for LMMs requires numerical optimization; the closed forms for the maximum-likelihood solutions cannot be derived analytically. The `lmer` function in the `lme4` package implements the parameter-estimation procedure. Options to the function enable us to switch between maximum-likelihood estimation (during model selection) and REML estimation (after the model equation is chosen). The BIC calculation used in model selection is also part of the `lme4` package. The log-likelihood calculations of different statistical packages often differ by a constant factor (e.g., because they drop different constant terms when optimizing for performance). However, since we only compare BIC scores from LMMs fitted using the `lmer` function in `lme4`, we avoid this pitfall.

Most of the evaluations and all of the analyses were performed on an 8 core Intel i7 CPU (2.8GHz) workstation, running the Linux (Redhat 4.3.2) operating system. Because some investigations required a massive number of evaluation runs, and because these evaluations can be run in parallel, we distributed them across 10–20 nodes of a Condor cluster maintained by Computing Facilities in the CMU School of Computer Science. The Condor cluster included 12 servers each running 10–12 virtual machines. Each virtual machine acted as a Condor node running Linux. When a job is submitted to the cluster, nodes are allocated at random, dependent in part on the running jobs of other cluster users.

3.10 Validation

While LMM model selection and parameter estimation are based on sound statistical theory, the analyst does face many choices in the process: the decision to treat some factors as fixed effects and others as random; the choice of simple and complex boundary model equations during model selection; the use of maximum-likelihood estimates and BIC-based model selection. One could argue that these series of decisions can guide the results in such a way that they confirm the analyst's biases rather than uncovering a model that best supports the data.

To avoid such an argument, we validate the model by demonstrating its usefulness. Specifically, we use the fitted model to make predictions about the results of subsequent evaluations. The subsequent evaluation is then performed, and the results are compared to the predictions. If the model is able to predict the error rates of the classifiers and which factors cause those error rates to increase or decrease, then it has demonstrated its usefulness regardless of the choices the analyst made when creating it.

This validation procedure is already used in standard evaluations of machine-learning algorithms (Hastie et al., 2001); separate data are used to train and test the algorithm. When collecting data, researchers collect a primary data set and withhold a second data set for the validation. The primary data set is used to train the algorithm and produce a classifier; the second data set is labeled using the classifier. The classifier-produced labels are compared to the true classes, and the accuracy of the classifier is established.

Analogously, we perform a primary and a secondary evaluation. In each evaluation, the same classifiers are evaluated under the same conditions. However, the subjects acting as genuine users and impostors in the two evaluations are separate. The primary evaluation is conducted with typing data from 2/3rds to 3/4ths of the total set of subjects, and the secondary evaluation is conducted using typing data from the remainder of the subjects. These proportions are typical of the split between training and testing in machine-learning evaluations (Mitchell, 1997).

One might ask why we opted to use a single training set and test set. More typical in machine learning is to use cross-validation: repeatedly splitting the data into several different training and test sets to produce multiple estimates of accuracy. Under standard assumptions, the average of these estimates vary less than the estimate produced from a single training/test split (Hastie et al., 2001).

A single training/test split was used because the standard assumptions of machine learning may not be acceptable assumptions in computer-security research (Sommer and Paxson, 2010). One oft-cited threat to security technologies is *concept drift*. Intuitively, concept drift is a term used to explain why a technology (e.g., an intrusion-detection system) may appear to work initially, but accuracy degrades because the environment slowly changes. In our work, concept drift is not only a threat for the classifiers (i.e., because typing behavior might evolve over time), it is a threat for the LMMs. Subject recruitment occurred over several years and through various means (e.g., class participation, active recruitment, etc.) One could imagine that some aspect of subject behavior might drift over this time period. Cross-validation would have obscured such drift, whereas as single training/test split will reveal it.

We adopt the convention of splitting the data into primary and secondary sets chronologically. The data for the secondary evaluation are collected *after* the data used for the primary evaluation. As such, our validation step resembles the typical scientific practice of replicating important scientific experiments to confirm or refute the initial results.

For illustration, we conduct a mock validation of the LMM in Table 3.1. We collected a secondary data set by having 10 new subjects type the password .tie5Roanl 200 times. As in the primary evaluation described in Section 3.2, we ran an evaluation trial for each combination of the three classifiers (A, B, and C), two feature sets (hold and down-down times), and 10 subjects. In each trial, the given subject was designated as the genuine user, and the given classifier was trained using 100 repetitions of the genuine-user subject typing the password. The remaining 100 repetitions were used to tune the classifier to obtain a 5% false-alarm rate. Then, each of the other 9 subjects in the secondary data set are designated as impostors, and the first 50 repetitions from each one are presented to the trained classifier. The miss rate is calculated from the classifier's response.

Figure 3.4 shows the results of this secondary evaluation. They are laid out in the same way as the results of the primary evaluation were in Figure 3.1. As in the earlier figure,

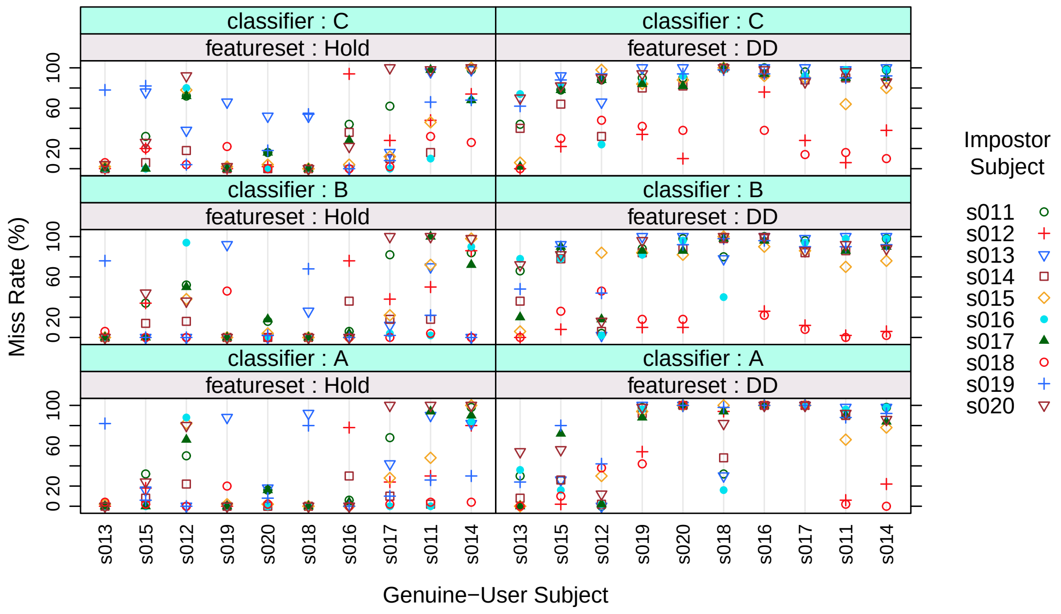


Figure 3.4: Example results from secondary evaluation for model validation. After results of a primary evaluation (shown in Figure 3.1) were used to build an LMM, data were collected from 10 more subjects and used to conduct a secondary evaluation. Genuine-user subjects have been ordered from least to greatest average miss rate. As in the primary evaluation, the wide range of miss rates within each panel and for each user make it difficult to interpret the results. The purpose of the validation is to establish whether the model provides an accurate interpretation.

for each classifier and feature set, some miss rates are low and others are high. For most genuine-user subjects, some per-impostor miss rates are low and others are high. This figure offers a reminder as to why we need statistical modeling to help us make sense of evaluation results. By comparing the models predictions to these results, we check whether the model offers an accurate description of how miss rates are affected by the classifier, feature set, user, and impostor.

Since the model predictions are quantitative (e.g., miss rates), we cannot simply validate the model by checking whether the predicted miss rates equal the actual miss rates. Differences between the predicted and the actual error rates are to be expected. In fact, the model predicts a high level of per-user and per-impostor variability (with the high $\sigma_{(user)}$ and $\sigma_{(impostor)}$ estimates).

To validate the predictions, the following procedure is used. First, we use the model to make predictions about the average miss rates and the per-user deviations. Specifically, we estimate 95th percentile prediction intervals for the per-user miss rates. For instance, in

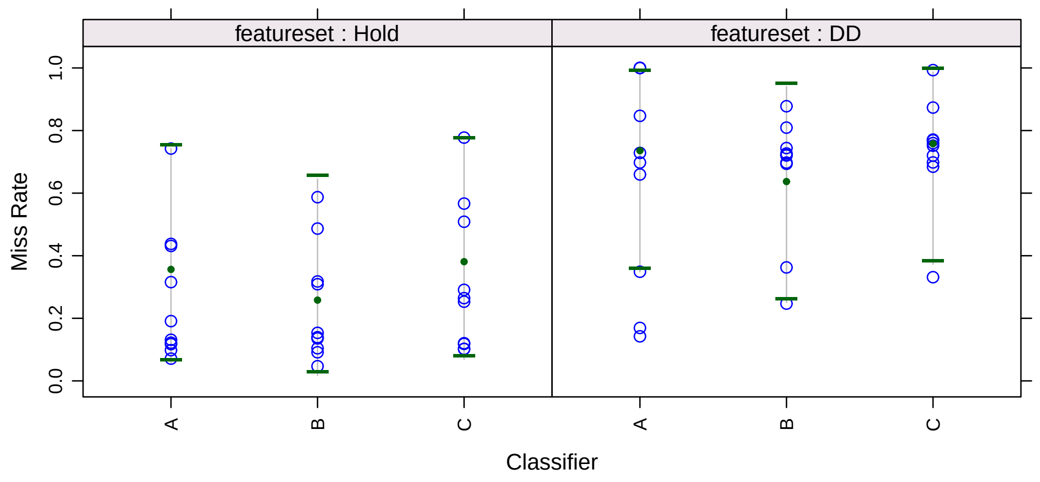


Figure 3.5: Model predictions and validation data for the example evaluation. The panels on the left and right presents results using hold times and down-down times respectively as features. In each panel, results are presented for the three classifiers in separate columns. Horizontal lines delineate the 95% prediction intervals for each classifier from the model. The blue dots indicate the per-user miss rates for the 10 users in the secondary evaluation data set. Nearly all actual results fall within the predicted intervals (with some exceptions to be expected), indicating a successful validation.

Section 3.4, we calculated a 95% prediction interval for the average miss rate for 95% of users using Classifier A and hold times:

$$\mathrm{VST}^{-1}(40.72 \pm 1.96 \cdot 12.98) = [5.6\%, 74.3\%]$$

The numbers come from the estimates of μ and $\sigma_{(user)}$ in Table 3.1. From the secondary-evaluation results, we calculate per-user average miss rates for each combination of fixed effects (e.g., for each combination of classifier and feature set). We graphically plot the prediction intervals and overlay the actual results from the second evaluation. Based on these plots, we assess whether the predictions of the model are accurate.

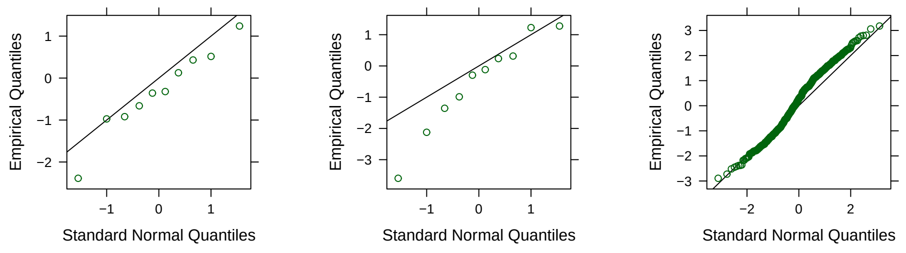
Figure 3.5 presents the results of this first validation step. For each classifier and feature set, we plot the 10 points corresponding to the average per-user miss rates for the 10 genuine-user subjects in the secondary evaluation. Superimposed on each set of points is the prediction interval calculated from the model. Most of the points lie within the predicted interval, indicating that the model is making accurate predictions. A few points do lie outside the intervals, but a few such points are to be expected. Of particular note, one can observe that the predicted intervals and actual results for all three classifiers are much higher for down-down times than for hold times. This observation further confirms our conclusions about the relative importance of the feature set.

As already noted, because of the large per-user variance, the prediction intervals are quite large. One might wonder whether they are overly large, thereby ensuring that the results of the secondary evaluation fall within them. To address this potential issue, after the initial assessment using prediction intervals, we perform a more comprehensive check that the model accurately predicts the results of the secondary evaluation. Based on the model, the per-user, per-impostor, and residual effects should all be Normally distributed with zero mean and a particular variance (given by the standard deviations in the parameter-estimate table).

The more rigorous assessment of the model will check not only the accuracy of the average prediction but the distributional characteristics of the errors in the predictions. For each miss rate in the secondary evaluation, we calculate the difference between the miss rate and what the model predicts based on the classifier and feature set. This produces 90 miss-rate differences for each classifier and feature-set combination (10 genuine-user subjects $\times$ 9 impostor subjects). We decompose these miss-rate differences into per-user effects, per-impostor effects, and residual effects. For each classifier and feature set, we calculate 10 per-user effects by averaging the 9 differences involving the same genuine-user subject. Likewise, for each classifier and feature set, we calculate 10 per-impostor effects by averaging the 9 differences involving the same impostor subject. Finally, we subtract the corresponding per-user and per-impostor effect from each of the 90 miss-rate differences to obtain a residual effect.

Based on the model, the per-user effects should be Normally distributed with zero mean and standard deviation equal to the per-user standard-deviation parameter estimate in the model. Likewise, the per-impostor and residual effects should be Normally distributed with zero mean and their own standard deviations. In each case, if we divide the effects and differences by their standard deviation according to the model, the resulting values should follow standard Normal distributions. We can assess how closely the results meet this assumption by using a QQ-plot (Venables and Ripley, 1997).

Figure 3.6 presents QQ-plots for the per-user, per-impostor, and residual effects. In a QQ-plot, the quantiles of the empirical distribution of the data are compared to the quantiles of a Normal distribution (in this work, though other distributions can be used as well). Different statisticians and software use different conventions when assigning the empirical or theoretical distribution to the x or y axes. In this work, we number the x axis with the quantiles of a standard Normal distribution and the y axis with those of the empirical



(a): per-user (b): per-impostor (c): residual

Figure 3.6: QQ-plots for validating LMM predictions from the example evaluation. The left panel assesses the Normality of per-user effects; the middle panel assesses per-impostor effects; the right panel assesses residual effects. Per-user and per-impostor deviations meet expectations except for one or two low outliers. Residuals show a slightly heavy tail.

distribution. If the data are Normally distributed, the quantiles should match and the points should fall along the diagonal line. Deviations from the line indicate deviations from the standard Normal. Particular properties of the empirical distribution (e.g., bias, heavy tails, and skew) can be read from the deviations.

Reading the deviations in a QQ-plot can be something of an art. In panels (a) and (b), the points below the line on the left side of the panel indicate either a few outliers or a left-skewed distribution. With only 10 points, it is not possible to precisely differentiate two cases. In panel (c), the slight sinusoidal shape is indicative of a slightly heavy tail (i.e., extreme points are observed a bit more often than expected from a Normal distribution). Such small discrepancies are to be expected, but the QQ-plots are largely consistent with the modeling assumptions. In such cases, we will conclude that the validation was successful. The model accurately predicted the results of the secondary evaluation

The careful reader will note that even a successful validation attempt does not prove that the initial findings were correct. There should be multiple, independent replications before we can trust a technology with critical security tasks. Such a reader is right, in that a single, successful validation attempt—especially by the same researcher who conducted the initial investigation—can only be so convincing. More work, by many researchers, will need to be done to truly convince a reader, especially of a surprising result.

3.11 Summary

This chapter introduced linear mixed-effects models (LMMs), providing examples of models and some of the intuition behind them. We introduced the need to perform a variance-stabilizing transformation in order to study classifier miss rates, without which the assumptions of the model would be violated. We describe LMMs through example, and we explain our procedure for using them: apply model selection to find the best model from a family of candidate models, and use maximum likelihood to estimate the effects of factors in the model. We investigate these effects using hypothesis testing. In the end, we validate the model by using it to predict the outcome of a subsequent evaluation. By comparing the predictions to the actual results, we establish that the model is making useful predictions. LMMs offer a means to identify and understand the factors that affect keystroke-dynamics error rates. The investigations of the next three chapters use LMMs to this end.

Chapter 4

Benchmarking Keystroke-Dynamics Classifiers

In this chapter, we conduct a fairly straightforward investigation to find the top-performing classifier among a set of promising classifiers for keystroke dynamics. As should be evident from the review in Chapter 2, many classifiers have been proposed and evaluated. Unfortunately, the results reported in the past are difficult to compare; too many factors vary from one evaluation to another. We collect a benchmark data set and evaluate 10 classifiers under controlled conditions so that their relative error rates can be compared soundly. In the process, we discover substantial per-user and per-impostor effects: some typists are easy to distinguish and other very hard to distinguish, regardless of the classifier.

4.1 Background

While the basic concept of keystroke dynamics is simple, different applications and classification technologies require us to distinguish different kinds of keystroke dynamics. In some scenarios, keystroke-dynamics classifiers are used to identify which typist, from a set of known typists, submitted a test sample. In other scenarios, classifiers are used to verify that the test sample was submitted by a particular known typist. In addition, the typing samples can range from login-time data (e.g., usernames and passwords) or in-session data (e.g., whatever keystrokes the user happens to type).

The different kinds of keystroke dynamics are relevant because they require different kinds of classification algorithm (i.e., with different inputs and outputs). Many machine-learning algorithms require training samples from every class that they learn. Such algorithms cannot be used in applications where the training samples all come from a single class (e.g., from a particular known typist). Algorithms operating on in-session data must accept arbitrary keystrokes as input while those operating on login-time data can assume the same keystrokes will be present in all samples (e.g., because the same password is typed every time).

These differences in the kinds of algorithm that can be used for each application make it difficult to compare classifiers across applications. Differences in accuracy could be attributed to the classifier or the application. While such comparisons can be made, for this benchmark (and the remainder of this work), we focus on a single application. Specifically, we concentrate on login-time authentication, and on classifiers that train on samples from a single class (often called *anomaly detectors*). This application is of particular importance since classifiers which operate well in this application can be put to immediate use. A classifier operating on login-time typing behavior adds a second layer of authentication, in the event that a user's secret password has been compromised. Monrose et al. (2002) likened this application to *hardening* a password.

Even constrained to one application, many factors beyond the classifier itself might make keystroke dynamics work better or worse. In login-time applications, one can analyze the typing rhythms of various typing tasks: the user's ID, full name, password, or some combination thereof. One can use hold times, down-down times, up-down times, or various other features and combinations. One can use a greater or lesser number of typing samples to train the classifier. These and other differences in how keystroke-dynamics classifiers are applied to login authentication might affect classifier results.

Unfortunately, in prior work, these factors have not been suitably controlled in such a way that classifiers can be compared across studies. Table 4.1 presents a summary of seven studies wherein login-time classifiers were evaluated. Each study described one or more classifier, gathered login-time typing data, conducted an evaluation, and reported the results. The key observation from this table is that, despite having focused on the same application of keystroke dynamics, each one used a different evaluation procedure. These differences make it impossible to soundly compare the different classifiers across studies.

The table has been split into two sections for readability. The first column in each section provides a reference to the source study. The remaining columns provide the following information:

Classifier: A descriptive name for the classification algorithm used in the study. Sometimes we use different terminology than the source authors to clarify the type of calculations that underlie the classification strategy.

Feature Sets: Features used to train and test the classifier. A check in the Return-key

			Feature Sets				Password	
	Source Study	Classifier	Return key	Down-down	Up-down	Hold	Length	Reps
1	Joyce and Gupta (1990)	Manhattan	✓	✓			N/A	8
2	Bleha et al. (1990)	Euclidean		✓			11–17	30
		Mahalanobis		✓			11–17	30
3	Cho et al. (2000)	Mahalanobis k-NN	✓		✓	✓	7	75–325
		Auto-Associative Neural Net	✓		✓	✓	7	75–325
4	Haider et al. (2000)	Outlier Count		✓			7	15
5	Yu and Cho (2003)	SVM	✓		✓	✓	6–10	75–325
6	Araújo et al. (2004)	Scaled Manhattan		✓	✓	✓	10+	10
7	Kang et al. (2007)	k-Means			✓	✓	7–10	10

		Filtering		Testing		Results (%)		
	Source Study	Users	Times	#Attempts	Updating	Threshold	Miss	False Alarm
1	Joyce and Gupta (1990)		✓	1		heuristic	0.25	16.36
2	Bleha et al. (1990)			1	✓	heuristic	2.8	8.1[a]
				1	✓	heuristic	2.8	8.1
3	Cho et al. (2000)	✓	✓	1		zero-miss	0.0	19.5
		✓	✓	1		zero-miss	0.0	1.0
4	Haider et al. (2000)			2		heuristic	19.	11.[b]
				2		heuristic	22.	20.
				2		heuristic	13.	2.
5	Yu and Cho (2003)	N/A	N/A	1		zero-miss	0.0	15.78
6	Araújo et al. (2004)			1	✓	heuristic	1.89	1.45
7	Kang et al. (2007)	N/A	N/A	1	✓	equal-error	3.8	3.8

Table 4.1: Different studies use substantially different evaluation procedures. We characterize the evaluation procedures used in seven studies evaluating login-time classifiers trained only on genuine-user samples. Despite working on the same keystroke-dynamics application, procedural differences in the evaluations make it impossible to soundly compare classifier error rates. Every difference offers an alternative explanation for different error rates. (Section 4.1 explains each column of the table.)

column means that the Return key is considered part of the typing task and its timing features are included in the feature set; a check in the down-down column means the feature set includes times between digraph down-down events; checks in the up-down and hold column indicate that the digraph up-down events and key hold times, respectively, are included in the feature set.

Password–Length: Number of characters used in the typing task (e.g., the username or password). An N/A indicates the length was not available from the source study.

Password–Repetitions: Number of typing-task repetitions used to train the classifier (i.e., samples of a user typing a password repeatedly). A range means that different amounts of training data were used for different users within the study.

Filtering–Users: A check indicates that users whose typing times were highly variable or inconsistent were identified during data collection and excluded from the study.

Filtering–Times: A check indicates that the collected timing data were processed with an outlier-handling procedure to remove extreme values.

Testing–#Attempts: Number of attempts that users were given to try to authenticate successfully. For instance, a 2 means a user who was rejected the first time would be given a second chance to repeat the task.

Testing–Updating: A check indicates that the typing profile was updated during testing to accommodate any changes in the typing behavior over time.

Results–Threshold: The procedure for choosing a classifier's decision threshold. These classifiers produce an anomaly score expressing the difference between a typing sample and the genuine user's typing profile. A threshold score is used as a decision boundary between genuine-user and impostor class labels: 'heuristic' means the threshold was chosen using some heuristic described in the study; 'zero-miss' means the threshold was chosen to obtain a miss rate of zero; 'equal-error' means the threshold was chosen to obtain equal miss and false-alarm rates.

Results–Miss/False Alarm: Reported miss and false-alarm rates. Superscript (a) denotes that the detectors were combined into an aggregate detector, and only the results for the aggregate were reported; superscript (b) indicates that only results for two-attempt authentication were reported.

For every one of these evaluation factors, the table shows that at least two studies differ in their treatment of the factor. In some cases, such as Password–Length, the factor is allowed to vary within the study as well. If these factors affect classifier miss and false-alarm rates, then when these error rates differ across studies, one cannot separate the effect of the factor from the effectiveness of the classifier; the two are confounded.

To illustrate the problem we encounter when we try to use the literature to determine which classifier has the best performance, suppose we tried to compare two classifiers: the Auto-Associative Neural Network developed by Cho et al. (2000), and the Outlier Count classifier designed by Haider et al. (2000). The neural net has a reported miss rate of 0.0% and a false-alarm rate of 1.0%. The outlier-counting detector has a reported miss rate of 13% and a false-alarm rate of 2%. Since the neural net has better miss and false-alarm rates, one might be inclined to conclude that it is better than the outlier-counting detector.

However, that conclusion is unsound because the table reveals many differences between the procedures used to evaluate the two classifiers. The neural net (1) trained on a different set of timing features, (2) had more repetitions in the training data, (3) did not have to deal with users whose typing was inconsistent, or with timing outliers in the training data, (4) was given only one attempt (not two) to correctly verify the user, and (5) was assessed using a different type of threshold on the anomaly score. Any of these five factors might explain why the neural net has lower error rates than the outlier-counting detector. If these factors were all controlled, we might discover that the outlier-counting detector outperforms the neural net. (The results of our benchmark will support this very reversal of the classifiers.)

4.2 Aim and approach

Our aim in this particular investigation is to benchmark promising classification technologies under controlled conditions so their error rates can be compared. The top-performing classifiers will be the focus of further evaluations in subsequent investigations (Chapters 5 and 6). In addition to comparing the classifiers themselves, we establish how much variation is due to the subjects who participate in an evaluation as genuine-user and impostor typists.

Our approach is as follows:

1. Conduct an experiment to evaluate a set of classifiers. We collect a benchmark data set, identify and implement 10 promising keystroke-dynamics classifiers, and evaluate each classifier using the same procedure.
2. Analyze the evaluation results using linear mixed-effects models (LMMs). We build the LMM, interpret the parameter estimates, and perform hypothesis tests. We identify the top performing classifier.
3. Validate the model by conducting a secondary evaluation. With the LMM, we make predictions about the error rates in the secondary evaluation. We compare the empirical results to the predictions, and we assess the accuracy of the model.

As an outcome, we identify which classifiers have the lowest error rates, and we also estimate how much deviation from the average error to expect for different users and impostors.

4.3 Experimental method

The experiment consists of data collection, classifier implementation, and the evaluation procedure. In brief, we collected typing data from 51 subjects, each typing 400 repetitions of a password. The various timing features used by researchers (e.g., down-down times, up-down times, and hold times) were extracted from the raw data. Ten classifiers from the keystroke-dynamics and machine-learning literature were identified and implemented. Each classifier was evaluated on the same data and under the same conditions, so that their error rates can be compared.

4.3.1 Password-data collection

The first step in our evaluation was to collect a sample of keystroke-timing data. We explain how we chose a password to use as a typing sample, designed a data-collection apparatus, recruited subjects to type the password, and extracted a set of password-timing features.

Choosing passwords for a keystroke-dynamics evaluation is tricky. On one hand, it is often more realistic to let users choose their own passwords. On the other hand, data collection becomes more difficult since different impostor samples would be needed for each password. Some researchers have suggested that letting users choose their own passwords makes it easier to distinguish them (Araújo et al., 2004). If the choice of a password is truly a factor that affects classifier error rates, then letting users choose different passwords could introduce a confounding factor. We decided that the same password would be typed by all of our subjects.

To make a password that is representative of typical, strong passwords, we employed a publicly available password generator (PC Tools, 2008) and password-strength checker (Microsoft, 2008). We generated a 10-character password containing letters, numbers, and punctuation, and then modified it slightly, interchanging some punctuation and casing to better conform with the general perception of a strong password. The result of this procedure was the following password:

.tie5Roanl

The password-strength checker rated this password as ‘strong’ because it contained a capital letter, a number, a punctuation character, and more than 7 characters. The top rating of ‘best’ was reserved for passwords longer than 13 characters, but according to the studies that were presented in Table 4.1, 10 characters is typical. Those studies that used longer strings often used names and English phrases that are easier to type.

We set up a laptop with an external keyboard to collect data, and we developed a Windows application that prompts a subject to type the password. As shown in the screenshot in Figure 4.1, the application displays the typing task on a screen with a text-entry field. In order to advance to the next screen, the subject must type the 10 characters of the password correctly, in sequence, and then press Return. If any errors in the sequence are detected, the subject is prompted to retype the password. The subject must type the password correctly 50 times to complete a data-collection session. Whenever the subject presses or releases a key, the application records the event (i.e., key-down or key-up), the name of the key involved, and what time the event occurred. An external reference clock was used to generate highly accurate timestamps. The reference clock was demonstrated to have an accuracy of ± 200 microseconds (by using a function generator to simulate key presses at

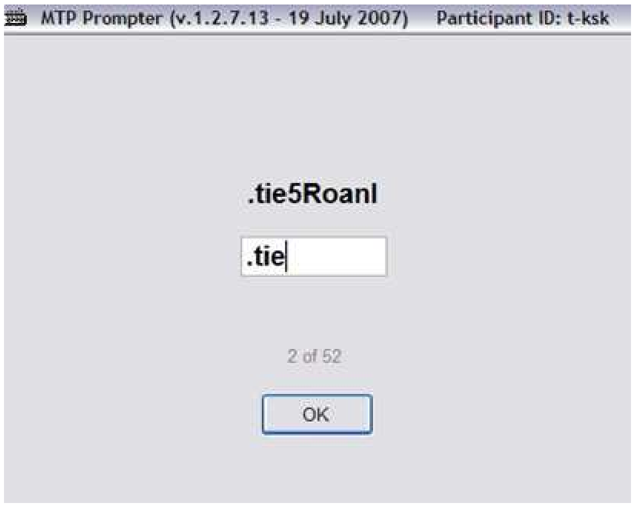

MTP Prompter (v.1.2.7.13 - 19 July 2007) Participant ID: t-ksk
.tie5Roanl
.tie
2 of 52
OK

Figure 4.1: Screenshot of the collection software for typing data. The subject is prompted with a character sequence. The program monitors the typing to ensure that the correct sequence of keystrokes is entered. When typographical errors are detected, the subject is prompted to repeat the sequence, ensuring that the desired number of correct typing sequences is collected. Each key-down and key-up event is recorded with its timestamp.

fixed intervals).

Of course, subjects would not naturally type their password 50 times in a row, and they would type it on their own computers, not our keyboard. We chose to sacrifice some amount of realism so we could use this carefully-controlled data-collection apparatus. We had two reasons for this decision. First, we wanted to ensure the accuracy of the timestamps (as described above). Second, we wanted to make the environment as consistent as possible for all subjects. If some subjects typed the password more frequently than others, or if different subjects used different keyboards, these differences would introduce uncontrolled factors.

We recruited 51 subjects from within the university. Subjects completed 8 sessions of data collection (with 50 password repetitions in each session), for a total of 400 password-typing samples. They waited at least one day between each sessions, to capture some of the day-to-day variation of each subject's typing. Additional demographic information on the subjects will be discussed in Chapter 5, where we investigate whether personal traits like age, gender, dominant hand and typing style affect classifier error rates.

The raw typing data (e.g., key events and timestamps) cannot be used directly by a classifier. Instead, sets of timing features are extracted from the raw data. These features are typically organized into a *timing vector*. Different researchers extract different combinations of features (as shown in the Feature Sets columns of Table 4.1). Since some earlier studies considered the Return key to be part of the password, we included the Return key timing features as well (effectively making the 10-character password 11 keystrokes long). and we extracted down-down times and hold times for all keys in the password. For each password, 21 timing features were extracted and organized into a vector (i.e., 11 hold times and 10 down-down times). The times are stored in seconds (as floating-point numbers).

We chose not to include up-down times among our features, since they are linearly dependent on the other features (i.e., each up-down time can be derived by subtracting a hold time from a down-down time). Such features violate the assumption of linear independence that many classifiers make. In Chapter 6, we investigate how other choices for the feature sets affect classifier results.

4.3.2 Classifier implementation

The second step in our evaluation was to implement ten classification algorithms that analyze password-timing data. As explained at the beginning of this chapter (Section 4.1), different keystroke-dynamics applications require different kinds of classification algorithms. We focused on one kind of algorithm: anomaly detectors, classifiers that train on samples from only a single class (e.g., typing from one genuine user rather than from multiple users and/or impostors).

Unfortunately, the classifiers used in keystroke-dynamics research are rarely shared as working code. Using a classifier proposed in the literature requires re-implementing it on the basis of a description in a research report. Rather than attempting to re-implement each classifier by adhering as faithfully as possible to the description, we attempted to build a classifier that was faithful to the underlying concept. For instance, the auto-associative neural network proposed by Cho et al. (2000) used learning-rate and momentum parameters that, on our data, caused the training procedure to fail or produce poor results. We tuned these parameters and increased the number of epochs used for training, to produce an auto-associative neural network that performed better and yet remained similar in spirit to the originally proposed classifier.

As reported in Chapter 2, hundreds of keystroke-dynamics studies have been conducted and dozens of classifiers have been proposed; many of them are login-time anomaly detectors. It would be impossible to re-implement every classifier previously proposed, so we selected 10 promising algorithms. Six were chosen from classifiers presented in Table 4.1, and four were among "classic" classification methods (e.g., Euclidean, Manhattan, Mahalanobis, and k-NN).

The classifiers were implemented using the R statistical programming environment (R

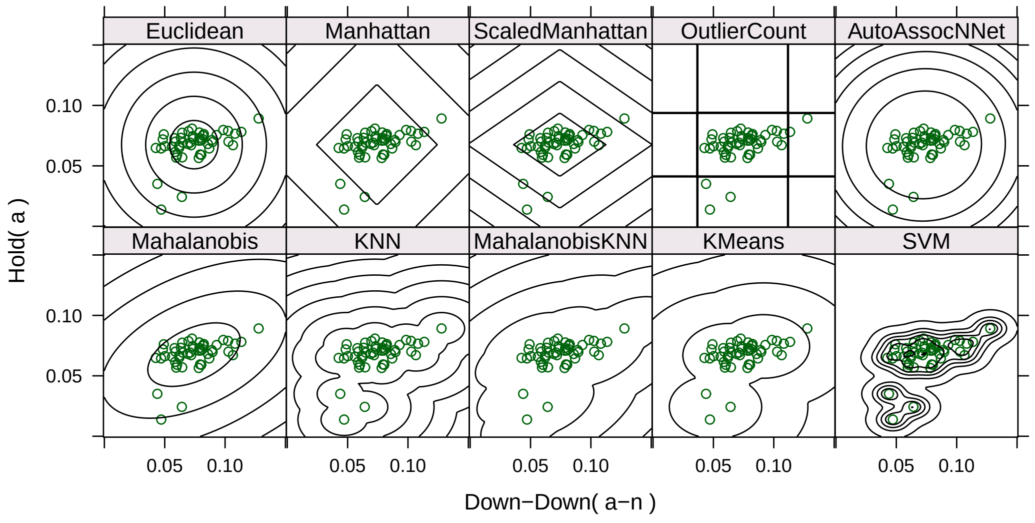


Figure 4.2: Contours of the decision surfaces for the 10 implemented classifiers. To create the plot, a user typed .tie5Roanl 50 times, and we extracted the a hold time and an down-down time from each repetition. The 50 pairs of timing features were used to train each of the 10 classifiers. Each panel illustrates the typing profile learned by one of the classifiers. A contour line defines a set of timing pairs of equal similarity. Differences in two contour plots represent differences in the profiles learned from the typing data.

Development Core Team, 2008). Each classifier has a training phase and a classification phase. During training, a set of timing vectors from the genuine user is used to build a profile of the user's typing behavior. During the classification phase, a new test vector is assigned a class label and a score. That score is a measure of dissimilarity with the user's typing profile. Lower values indicate similarity, and high values express dissimilarity between the test sample and the training profile. As a result, these scores are often called *anomaly scores*. Different classifiers assess similarity differently, and so they assign different scores to a test sample. The class label is determined by comparing the anomaly score to a threshold, and if the score exceeds the threshold, the typist is classified as an impostor; otherwise, the typist is classified as the genuine user.

When describing these classifiers, we must occasionally resort to terminology that is specific to a statistical or machine-learning technique (e.g., the vocabulary of neural networks); references are provided. Before we delve into the details of each of the 10 classifier implementations, Figure 4.2 presents contour plots of each classifier's anomaly scores when trained on the same data set. To create the plot, a user typed .tie5Roanl 50 times, and we extracted the a hold times and the an down-down time from each repetition. The 50 pairs were used to train the 10 classifiers we implemented. Once trained, the anomaly scores produced by these classifiers were used to generate contour plots.

The contours of each classifier are presented along with the points used to train the classifier. Looking at the contours, one can compare the profiles learned by each classifier. Classifiers with similar contours more-or-less learned similar profiles; they can be expected to have similar behavior. Those classifiers with markedly different contours learned different concepts and may have markedly different behavior.

We find contour plots useful for developing our intuition about classifier behavior. For instance, consider three groups of points in each panel: those in the center of the cloud of points, those on the left edge of the cloud of points, and those outside the cloud in the lower left. The Euclidean contours radiate out from the center of the points. Relatively speaking, points in the middle of the cloud will have low anomaly scores; those on the edge of the cloud will have medium scores; those outside the cloud in the lower-left corner will have very high scores. In contrast, the k-NN contours radiate from the nearest point, so every point—be it in the middle of the cloud, on the edge, or outside—will have a low anomaly score. The different shapes of the Euclidean and k-NN contours depict the very different typing profiles learned by these classifiers. With the intuition provided by the contour plots in mind, we present the details of the 10 classifier implementations.

Euclidean. This classic anomaly-detection algorithm (Duda et al., 2001) models each password as a point in p-dimensional space, where p is the number of features in the timing vectors. It treats the training data as a cloud of points, and computes the anomaly score of the test vector based on its proximity to the center of this cloud. Specifically, in the training phase, the mean vector of the set of timing vectors is calculated.

In the classification phase, the Euclidean distance between the test vector and the mean vector is computed. With x_i and y_i representing the i-th timing feature of the mean vector and test vector respectively, the anomaly score is calculated as

$$anomaly\ score = \sqrt{\sum_{i=1}^{p} (x_i - y_i)^2}$$

In Figure 4.2, we see that the contours form concentric circles (or hyperspheres in n dimensions) around the central mean vector. As noted above, the anomaly score of a point depends on its distance from that central mean vector.

Manhattan. This anomaly-detection algorithm (Duda et al., 2001) resembles the Euclidean classifier except that the distance measure is not Euclidean distance, but Manhattan (or city-block) distance. In the training phase, the mean vector of the timing vectors is calculated.

In the classification phase, the Manhattan distance between the mean vector and the test vector is computed. Specifically, the difference between the two vectors is calculated across each of the 21 features, and the absolute values of these differences are summed. Again, using x_i and y_i as the i-th timing features of the mean and test vectors, respectively, the anomaly score is

$$anomaly\ score = \sum_{i=1}^{p} |x_i - y_i|$$

In Figure 4.2, we see that the contours form concentric diamonds around the mean vector. As with the Euclidean classifier, points in the center of the cloud will have low anomaly scores, and points outside the cloud in the lower left will have high scores. Unlike the Euclidean classifier, points on the left edge of the cloud have relatively low anomaly scores. Because of the diamond shape of the Manhattan contours, points that differ from the center in only one dimension (i.e., horizontal *or* vertical) are assigned lower anomaly scores than those that differ in multiple dimensions. Specifically, a 50 millisecond deviation in one feature is equivalent to 25 millisecond deviations in each of 2 features, since the sum total deviation is the same.

Scaled Manhattan. This classifier is similar to that described by Araújo et al. (2004). In the training phase, the mean vector of the timing vectors is calculated, and a vector of standard deviations is calculated as well. This vector has the same length as the mean vector and contains the standard deviations of each feature. In the classifier described by Araújo et al., the mean absolute deviation is calculated instead. (Empirically, we found little difference between the two classifiers, and while the mean absolute deviation is potentially a more robust estimator, we used the standard deviation for consistency with other scaled classifiers).

In the classification phase, the calculation is similar to the Manhattan distance, but with a small change. The distances in each dimension (i.e., for each feature) are scaled by the standard deviation of the training samples in that dimension. Specifically, with x_i and y_i defined as with the previous classifiers, and with s_i as the standard deviation of the i-th feature over the training data, the anomaly score is

$$anomaly\ score = \sum_{i=1}^{p} |x_i - y_i| / s_i$$

In Figure 4.2, we see that the contours form concentric diamonds, but unlike the Manhattan diamonds, these ones are wider than they are tall. The training data varies more widely in the Down-Down(an) dimension than in the Hold(a) dimension. Consequently, points on the right and left edge of the cloud of points have almost the same anomaly score as points on the upper and lower edge of the cloud, despite the points on the right and left being 25 milliseconds from the center of the cloud rather than 10 milliseconds.

Outlier Count. This classifier was described by Haider et al. (2000), who called it the "statistical technique." In the training phase, the classifier calculates the mean and standard deviation of each timing feature.

In the classification phase, the classifier computes the absolute z-score of each feature of the test vector. The z-score for the i-th feature is calculated as

$$z_i = |x_i - y_i| / s_i$$

where x_i, y_i, and s_i are defined as above. Each of the z-scores is compared to the two-sided 95th percentile of the standard Normal distribution. Specifically, any z-score lower than the 2.5th percentile or higher than the 97.5th percentile was considered an outlier. Since these percentiles correspond to ± 1.96, the classifier counts the number of z-scores that exceed these values:

$$anomaly\ score = \sum_{i=1}^{p} \mathbb{I}\left(|z_i| > 1.96\right)$$

where $\mathbb{I}(x)$ is the indicator function: 1 if x is true and 0 otherwise.

In Figure 4.2, we see grid-like contours. In the middle rectangle, the anomaly score is 0 because neither timing features' z-scores exceed the ± 1.96 threshold. In the 4 rectangles directly above, below, to the left, and to the right of the middle rectangle, the anomaly score is 1 because one timing feature's z-score exceeds the threshold, but the other does not. Finally, in the four corners of the panel, the anomaly score is 2 because both features' z-scores exceed the threshold. Observe that one potential limitation of this classifier is the granularity of its anomaly scores; only integer values between 0 and the number of features (p) are possible. This granularity can make it difficult to fine-tune the classifier for a particular false alarm rate.

Auto-Associative Neural Network. This classifier was described by Cho et al. (2000), who called it an "auto-associative multilayer perceptron." Like a traditional neural network, the layout is feed forward and the back-propagation algorithm is used for training. Unlike a typical neural network, the structure of the network is designed for use as an anomaly detector (Hwang and Cho, 1999). Intuitively, the training phase teaches the network to produce output vectors close to the inputs for the training vectors (hence the "auto-associative" descriptor). Then, during the classification phase, input vectors that produce dissimilar outputs are assigned high anomaly scores.

In the training phase, the neural network is constructed with p input nodes and p output nodes (where p is the number of timing-vector features). We used p hidden nodes as well (as described by Cho et al. (2000)). The network is trained to reproduce each input timing vector as the output. We trained for 500 epochs using a learning-rate of 0.01 and a momentum parameter of 0.03. The `AMORE` neural-network package, version 0.2-12 (Castejón Limas et al., 2010), was used to implement the classifier.

In the classification phase, the p-feature test vector is run through the network, producing a p-feature output vector. The anomaly score is calculated as the Euclidean distance between the test vector and the output vector. In Figure 4.2, the Neural Net contours resemble the Euclidean contours: concentric circles. In this case, the neural net typing profile resembles the Euclidean profile. More typically, especially in higher dimensions, the two profiles are very different, and the neural net profile is an improvement over the Euclidean profile.

Mahalanobis. This classic detection algorithm (Duda et al., 2001) resembles the Euclidean and Manhattan classifiers, but the distance measure is more complex. Mahalanobis distance can be viewed as an extension of Euclidean distance to account for correlations between features. In the training phase, both the mean vector and the covariance matrix of the timing vectors are calculated.

In the classification phase, the Mahalanobis distance is calculated between the mean vector and the test vector. Specifically, letting $\boldsymbol{x}$ be the p-dimensional mean vector, $\boldsymbol{y}$ be the p-dimensional test vector, and $\boldsymbol{S}$ be the $p \times p$-covariance matrix, the Mahalanobis distance (and the anomaly score) is

$$\textit{anomaly score} = \sqrt{(\boldsymbol{x} - \boldsymbol{y})^{\top} \boldsymbol{S}^{-1} (\boldsymbol{x} - \boldsymbol{y})}$$

In Figure 4.2, we see that this classifier's contours form tilted ellipses. The elliptical shape reveals that this classifier, like the Scaled Manhattan classifier, accounts for differences in variation across dimensions. The tilt shows that the classifier also accounts for correlations among paired features. In this case, shorter a hold times correlate with shorter an down-down times.

k-Nearest Neighbor. Most of the classifiers described above calculate the anomaly score using distances from the test vector to the center of the training vectors. Nearest-Neighbor algorithms calculate distances from the training vectors nearest to the test vector (Duda et al., 2001). Specifically, in the training phase, the mean vector of the timing vectors and the standard deviations of each feature are calculated. The training data are scaled and re-centered to have zero mean and unit variance (i.e., by subtracting the mean from each feature and dividing by the standard deviation). The classifier saves this list of scaled training vectors.

During the classification phase, the classifier performs the same scaling and re-centering on the test vector. Then, it finds the k training vectors closest to the test vector (using Euclidean distance), where for this investigation $k = 1$ was chosen. The anomaly score is calculated as the average of these k vectors. The ANN approximate-nearest neighbor library, version 0.1 (Mount and Arya, 2010), was used to implement this classifier.

Figure 4.2 shows how the contours across the panel depend on the distance to the closest of the training points. Each point in the training data has an anomaly score of zero, and the contours radiate in concentric circles from the nearest point.

Mahalanobis k-Nearest Neighbor. This classifier was described by Cho et al. (2000). In the training phase, the classifier calculates the mean vector and the covariance matrix of the timing vectors. Each training vector is transformed as follows:

$$\boldsymbol{x}' = (\boldsymbol{x} - \overline{\boldsymbol{x}})^{\mathsf{T}} \boldsymbol{E} \boldsymbol{D}^{-1/2} \boldsymbol{E}^{\mathsf{T}}$$

where $\boldsymbol{x}$ is the training vector (arranged as a column), $\overline{\boldsymbol{x}}$ is the mean vector, $\boldsymbol{E}$ is the set of eigenvectors of the covariance matrix (each in a separate column of the matrix), and $\boldsymbol{D}$ is a diagonal matrix with the eigenvalues along the diagonal. The eigenvectors and eigenvalues in $\boldsymbol{E}$ and $\boldsymbol{D}$ are arranged in corresponding order (e.g. the eigenvalue in the first column of $\boldsymbol{D}$ corresponds to the eigenvector in the first column of $\boldsymbol{E}$).

After this transformation, the new training vectors (denoted $\boldsymbol{x}'$ above) have zero mean, unit variance, and zero covariance between features. As a result, Euclidean distances between points post-transformation are equivalent to Mahalanobis distances between points pre-transformation. In addition to the mean vector and covariance matrix, the classifier saves the list of transformed training vectors.

During the classification phase, the classifier performs the same transformation on the test vector (using $\overline{\boldsymbol{x}}$, $\boldsymbol{E}$, and $\boldsymbol{D}$ from the training phase). Then, it finds the k training vectors closest to the test vector, where again $k = 1$ was chosen. Since the Euclidean distance was used to calculate these distances in the transformed-vector space, the closest training vectors are those with the shortest Mahalanobis distance in the original space. The anomaly score is calculated as the average of these k distances. As with the k-NN classifier implementation, the `ANN` approximate-nearest neighbor library, version 0.1 (Mount and Arya, 2010), was used to implement this classifier.

Figure 4.2 shows how the contours of this classifier bear some resemblance to those of both the Mahalanobis and k-NN classifiers. Like the k-NN classifier, the contours depend on the nearest point, but now the distance is scaled and tilted in the same manner as the Mahalanobis ellipses.

k-means. This classifier is similar to that described by Kang et al. (2007). It uses the k-means clustering algorithm to identify clusters in the training vectors, and then calculates whether the test vector is close to any of the clusters. In the training phase, the classifier simply runs the k-means algorithm on the training data with $k = 3$. The algorithm produces three centroids such that each training vector should be close to at least one of the three centroids.

In the classification phase, the nearest centroid to the test vector (using Euclidean distance) is identified. The anomaly score is calculated as the distance between this centroid and the test vector. The classifier we implemented differs from that described by Kang et al. because they scaled the distances to each cluster by the average distance of points in the cluster. Presumably, in their evaluation, this adjustment produced better results. In our preliminary test, the simpler algorithm (with no such adjustment) performed much better, so we used it.

Figure 4.2 illustrates the behavior of this classifier. Two centroids have been fitted to the main cloud of points in the middle of the panel, and one centroid has been fitted to the three points near the bottom of the panel. The contours throughout the panel emanate from the nearest centroid.

SVM. This classifier was described by Yu and Cho (2003). It incorporates an algorithm called a support-vector machine (SVM) that projects two classes of data into a high-dimensional space and finds a linear separator between the two classes. A "one-class" SVM variant was developed for anomaly detection (Schölkopf et al., 2001); it projects the training data and finds a separator between the projection and the origin. In the training phase, the classifier builds a one-class SVM using the training vectors. In contrast to the classifier described by Yu and Cho, the SVM parameter (ν) was set to 0.05, since it corresponds to a target false-alarm rate and a 5% false-alarm rate seemed reasonable.

In the classification phase, the test vector is projected into the same high-dimensional space and the (signed) distance from the linear separator is calculated. The anomaly score is calculated as this distance, with the sign inverted, so that vectors with negative scores are on the same side of the separator as the training data and those with positive scores are separated from the data. The SVM functionality in the `kernlab` package, version 0.9-13 (Karatzoglou et al., 2004), was used to implement the classifier.

Figure 4.2 captures the somewhat peculiar behavior of this classifier. Effectively, the transformation projects the area around each training data point into one region, and everything else near the origin. As a result, the separating hyperplane manifests in the original space as a series of tight ellipses around each data point. The tight packing of these ellipses is an artifact of the small distances between all the points in the projection space.

4.3.3 Evaluation procedure

Consider a scenario in which a user's long-time password has been compromised by an impostor. The user is assumed to be practiced in typing their password, while the impostor is unfamiliar with it (e.g., typing it for the first time). We evaluate each classifier's ability to discriminate between the impostor's typing and the genuine user's typing in this scenario.

We start by designating one of our 51 subjects as the genuine user, and the rest as impostors. We train a classifier and test its ability to recognize the genuine user and impostors as follows:

1. We run the training phase of the classifier on the timing vectors from the first 200 password repetitions typed by the the genuine user. The classifier builds a profile of the genuine user's typing behavior.
2. Then, we run the classification phase of the classifier on the timing vectors from the remaining 200 repetitions typed by the genuine user. We record the anomaly scores assigned to each timing vector. We find the 95th percentile of these scores and tune the classifier to operate with this threshold. Such tuning ensures that the classifier

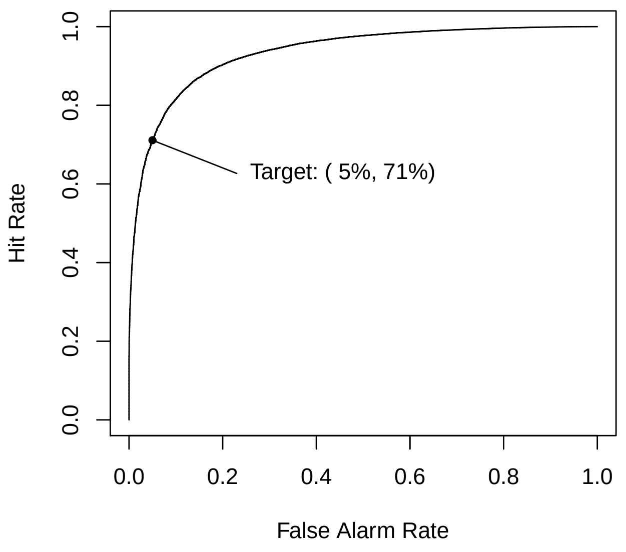


Figure 4.3: Example ROC curve with the 5% false-alarm operating point marked. The curve illustrates the false-alarm/miss trade-off for a classifier. The marker indicates the operating point when the classifier has been tuned to operate with a 5% false-alarm rate. At that false-alarm rate, this particular classifier achieves a 71% hit rate—equivalent to a 29% miss rate.

operates with a 5% false-alarm rate.

3. Finally, we run the classification phase of the classifier on the timing vectors from the first 50 repetitions typed by each of the 50 impostors. We record the anomaly scores assigned to each timing vector. These scores constitute the *impostor anomaly scores* for the particular user. We compare these scores to the operating threshold (calculated in step 2), and we determine whether the classifier would have recognized or missed the impostor. From all 50 samples from a particular impostor subject for a particular genuine-user subject, we calculate the *per-user/impostor* miss rate for that combination of classifier, genuine-user subject, and impostor subject.

This process is then repeated, designating each of the other subjects as the genuine user in turn. After training and testing each of the 10 classifiers, we have a total of 1,275,000 impostor anomaly scores (10 classifiers $\times$ 51 genuine-user subjects $\times$ 50 impostor subjects $\times$ 50 repetitions) and 25,000 per-user/impostor miss rates.

The process of tuning a classifier to have a 5% miss rate is illustrated on the ROC curve in Figure 4.3. The hit rate is the frequency with which impostors are classified as such (i.e., $1 - miss\ rate$), and the false-alarm rate is the frequency with which genuine users are mistakenly classified as impostors. Whether or not a sample is classified as being from an impostor depends on how the threshold anomaly score is chosen. The threshold determines an operating point on the ROC curve. Over the continuum of possible thresholds, the ROC curve illustrates the hit and false-alarm rates that would be attained at each possible operating point.

For this work, the operating point is chosen to achieve a 5% false-alarm rate. Different choices are certainly possible, but we believe that this represents a reasonable choice. A 5% false-alarm rate is a reasonable target for a keystroke-dynamics classifier. If 1 in 20 genuine attempts is rejected, the rate is similar to the rate of password typographical errors (at least in our experience). In either case, the user would have to retype the password to authenticate. While a higher or lower false-alarm rate may be desired under certain circumstances, a 5% target appears reasonable.

Our decision to use this tuning procedure stemmed from the lack of desirable alternatives in the literature. Equal-error rates are often recommended as a means of summarizing classifier performance in a single number (Peacock et al., 2004; Crawford, 2010), but often that number represents a point on the ROC curve far from the region of interest. For instance, if one classifier has a 20% EER and another has a 30% EER, we can infer very little about their relative performance at a 5% false-alarm rate. (About the only thing we can conclude, based on the convexity of ROC curves, is that the first classifier's miss rate is higher than 20% and the second one's is higher than 30%.) A second alternative, used by Cho et al. (2000), is to analyze the test-sample anomaly scores (post-hoc) to find the zero-miss threshold. We believe such a procedure has weaker supporting rationale than our 5%-false-alarm threshold. Estimating a threshold to obtain a target false-alarm rate requires only genuine-user anomaly scores; estimating a threshold to obtain a miss rate requires impostor anomaly scores. The latter are presumably much harder to come by in practice, making it seem less practical to tune a classifier to operate at a fixed miss rate.

Finally, it may seem that 200 repetitions is an unrealistically large amount of training data. We were concerned that fewer passwords might unfairly cause one or more classifiers to under-perform. Table 4.1 on page 69 presented studies in which classifiers required up to 325 passwords for training. Likewise, an unpracticed impostor might seem unrealistic, since impostors might practice if they knew timing mattered. We believe that our choices are fair for an initial benchmark; subsequent investigations (e.g., in Chapter 6) involve fewer training repetitions and more practiced impostors.

Classifier	False-Alarm Rate	Miss Rate
ScaledManhattan	5.0	23.6
KNN	5.0	29.8
SVM	5.0	30.2
OutlierCount	2.9	31.7
MahalanobisKNN	5.0	33.7
KMeans	5.0	35.0
Mahalanobis	5.0	39.1
Manhattan	5.0	41.8
AutoAssocNNet	5.0	56.3
Euclidean	5.0	61.0

Table 4.2: Average error rates for the classifiers on the benchmark data. False-alarm and miss rates are presented as percentages. Classifiers were tuned to have a 5% false-alarm rate (insofar as possible), and results are sorted by miss rate.

4.4 Empirical results

The per-user/impostor miss rates were averaged over every user/impostor combination to find the average miss rates for each classifier. These results and the corresponding false-alarm rates are presented in Table 4.2. Because we tuned each classifier to have a false-alarm rates of 5%, we should explain the 2.9% false-alarm rate for the Outlier Count classifier. As noted earlier, the anomaly scores of this classifier are so coarse that 2.9% is the closest operating point to a 5.0% false-alarm rate.

Scaled Manhattan has the lowest empirical miss rate: 23.6%. A group of six other classifiers have miss rates below 40%. The remaining three classifiers have miss rates between 41% and 61%. The results in this table are typical of those reported from keystroke-dynamics evaluations in the sense that they include only averages: average miss and false-alarm rates.

Tables such as these imply that each classifier has *an* error rate (or *a* miss rate and *a* false-alarm rate). They do not capture or convey any of the uncertainty in these estimates. For the Scaled Manhattan classifier, can a 23%–24% miss rate be expected in a deployment, or a 3%–53% miss rate? Either could correspond to an average miss rate of 23.6%. This question can only be answered by analyzing variability. Per-user and per-impostor miss rates can provide some insight into this variability.

Figures 4.4 and 4.5 plot the per-user miss rates for the 10 classifiers. They were divided among two figures due to layout constraints. Figure 4.4 presents results for the five classifiers with the lowest average miss rates, and Figure 4.5 presents results for the five

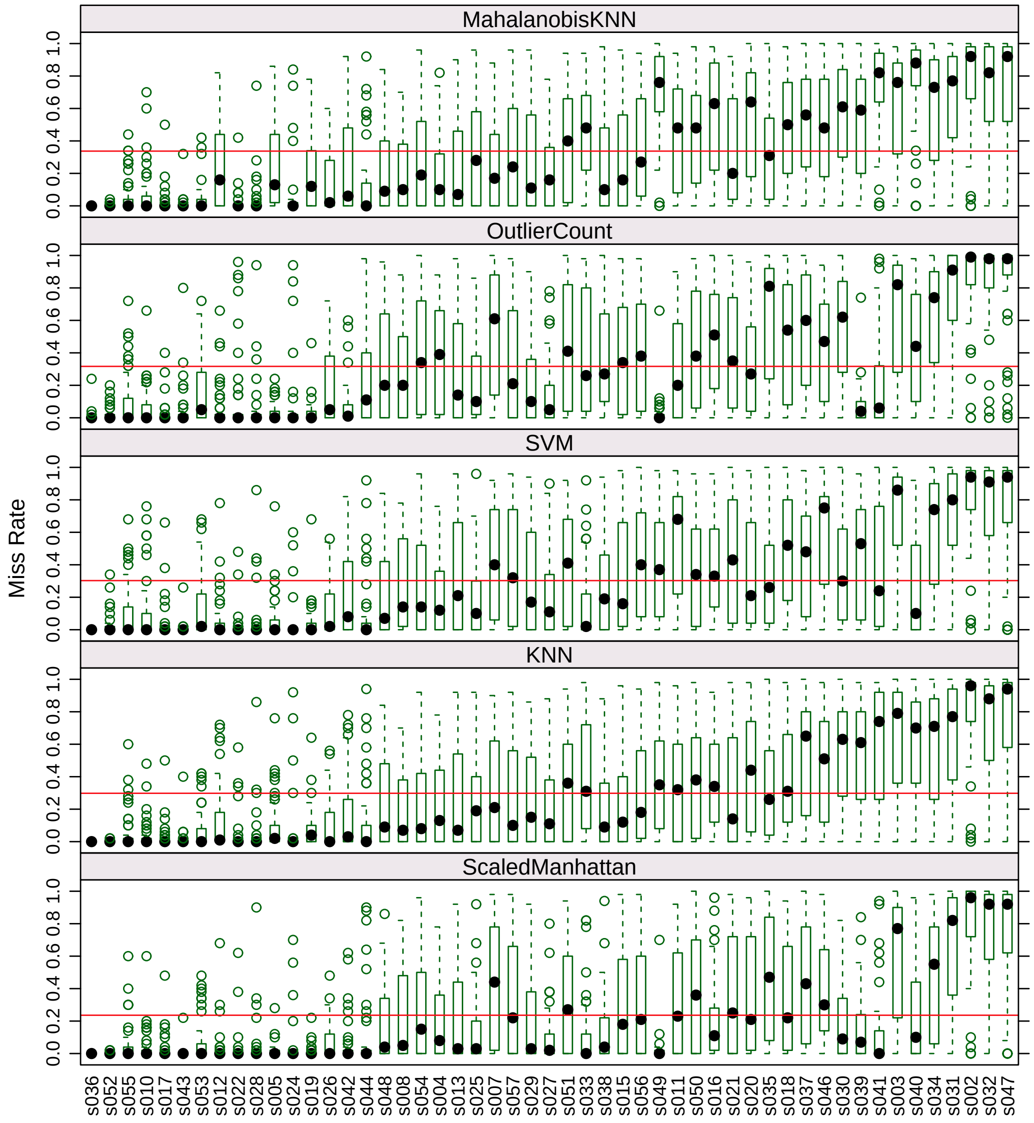


Figure 4.4: Per-user miss rates for the first 5 classifiers in the benchmark evaluation. The miss rates are presented separately for each of the 51 genuine-user subjects, ordered by average miss rate. Each boxplot summarizes the distribution of miss rates for the genuine-user subject over all 50 impostor subjects. The red line denotes the average miss rate for the classifier. Note how some per-user miss rates are low for every classifier and others are high.

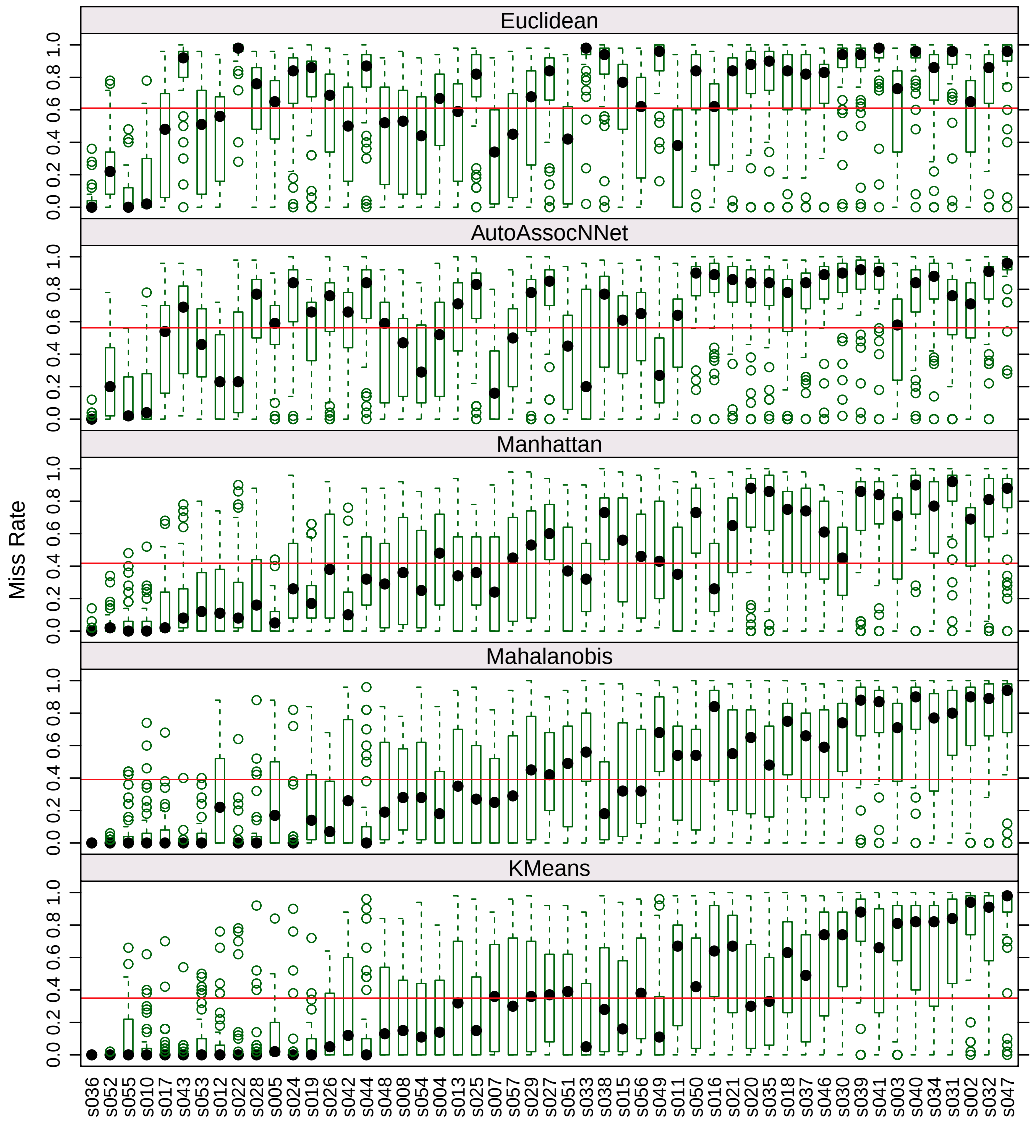


Figure 4.5: Per-user miss rates for the last 5 classifiers in the benchmark evaluation. Each panel is structured as in Figure 4.4. Again, note that some per-user miss rates are low for every classifier and others are high.

classifiers with the highest average miss rates. These figures contain a panel per classifier, and separate columns within each panel for individual users. Each column contains a boxplot to capture the distribution of miss rates for the corresponding user. Boxplots are standard tools for visualizing data distributions (Gonick and Smith, 1993; Moore and McCabe, 1998). In these boxplots, the medians are presented as black dots. The range from the 25th to the 75th percentile is enclosed in the box. The whiskers stretch to the minimum and maximum points, up to a heuristic maximum length (i.e., 1.5 times the inter-quartile range, the distance from the 25th to 75th percentile). Points beyond the end of the whisker are drawn individually, denoting their status as possible outliers.

The red line across each panel indicates the average error rate for that classifier. Every panel contains boxplots that stretch from 0% to 100%, indicating that for at least some genuine-user subjects, some impostor subjects are perfectly detected and others are never detected. As in the example in Chapter 3, high variability continues to be a key feature of keystroke-dynamics miss rates.

It is interesting to note how similar many panels are. The medians for all five of the classifiers in Figure 4.4 and several classifiers in Figure 4.5 have a similarly shaped trends. They are near zero for about a third of the panel and then rise almost linearly to one across the remainder of the panel. The similarity of these panels suggests that these classifiers not only have similar average miss rates (per the results in Table 4.2), but also similar per-user miss rates. The implication is that one cannot dramatically improve error rates by matching each user with the best classifier for that user. It is not the case that one classifier has low error for one group of users, and another classifier has low error for a separate group.

These plots also reveal some users for whom this technology works well, and others for whom it works poorly. The 23.6% average miss rate for the Scaled Manhattan classifier is an average over a wide range. On the one hand, this is a discouraging observation, because it means that there are some users for whom the technology works much worse than the 23.6% average miss rate. On the other hand, if we can understand the users for whom the technology gets very good results (e.g., below 5% miss rate), then for those users we have a viable technology.

4.5 Statistical analysis

Having performed exploratory analysis of the empirical results, we now proceed with a more formal statistical analysis. As described in Chapter 3, we use linear mixed-effects models. The false-alarm rate was held constant at 5%, and so we model the miss rates.

Minimal Model Equation:

$$\begin{aligned}
\mathrm{VST}(\mathit{miss\ rate})_{ijk} &= \mu + (\mathit{user})_i + (\mathit{impostor})_j + \varepsilon_k \\
(\mathit{user})_i &\sim N(0, \sigma^2_{(\mathit{user})}) \\
(\mathit{impostor})_j &\sim N(0, \sigma^2_{(\mathit{impostor})}) \\
\varepsilon_k &\sim N(0, \sigma^2_{\varepsilon})
\end{aligned}$$

Maximal Model Equation:

$$\begin{aligned}
\mathrm{VST}(\mathit{miss\ rate})_{ijkl} &= \mu + (\mathit{Classifier})_i + (\mathit{user})_j + (\mathit{impostor})_k + \varepsilon_l \\
(\mathit{user})_j &\sim N(0, \sigma^2_{(\mathit{user})}) \\
(\mathit{impostor})_k &\sim N(0, \sigma^2_{(\mathit{impostor})}) \\
\varepsilon_l &\sim N(0, \sigma^2_{\varepsilon})
\end{aligned}$$

Figure 4.6: Equations for model selection in the benchmark-evaluation analysis. The minimal equation includes only the random effects, which are part of the structure of the experiment. The maximal equation also includes the classifier. In this case, the two model equations are the only two in the family. Model selection, in this case, reduces to a question of whether the classifier has an effect on the miss rate.

We begin with model selection and parameter estimation followed by statistical hypothesis testing.

4.5.1 Model selection

During model selection, we construct a family of candidate models by defining a minimal and a maximal member of the family. These two model equations effectively define the boundaries of the model family. For this analysis, the minimal and maximal models are presented in Figure 4.6. The minimal equation contains the user and impostor random effects, which we include in every model of miss rates. The maximal equation contains those random effects and also a classifier fixed effect. Since the two equations differ in the presence/absence of a single term, they are the only two models in the family they define.

With this set of model equations, the model selection process effectively degenerates to the question of whether there is sufficient evidence that the classifier has an effect on miss rates. If not, then model selection will choose the minimal equation (without the term); if so, then model selection will choose the maximal equation (with the term). Under other circumstances, an analyst might choose to use a different statistical technique (e.g., likelihood-ratio testing) to choose between these models. However, for consistency with the other investigations in this work, we use model selection based on the BIC criteria.

We use maximum-likelihood parameter estimation to estimate the parameters of both the minimal and maximal model. For each model, we compute the BIC and compare. Not surprisingly given the different empirical average miss rates for the different classifiers, the BIC for the maximal model (226212.24) is lower than the BIC for the minimal model (231623.44). We select the model containing a classifier effect.

4.5.2 Parameter estimation

Having selected the model, we repeat the parameter-estimation procedure using REML. Table 4.3 presents the selected model equation and the REML parameter estimates. Note that these estimates are in the variance-transformed space (i.e., $\mathrm{VST}(y/n)$ for miss rate y/n), as explained in Chapter 3.

In the parameter-estimation table, the baseline is the Euclidean classifier. The choice of baseline was made arbitrarily, but it happened to be the classifier with the highest miss rate. The estimated baseline is 57.68, corresponding to an estimated miss rate of $\mathrm{VST}^{-1}(57.68) = 61.9$%. Note that this estimate is very close to but not exactly the same as the empirical average miss rate for the Euclidean classifier (61.0%) from Table 4.2. The small difference is due to the non-linearity of the variance-stabilizing transformation. The average of the transformed values is not the same as the transformation of the averaged values. Nevertheless, the estimate is still quite close to the observed error rate.

The nine classifier-effect parameters estimate the change in the variance-transformed miss rate when the Euclidean classifier is replaced with each of the other nine classifiers. All these estimates are negative, indicating that the miss rate is lower. For some classifiers, like the Auto-Associative Neural Net, the change is comparatively small (-4.63); for others, like Scaled Manhattan, the change is quite large (-33.14). The estimated miss rate for the Scaled Manhattan classifier is 14.1% ($\mathrm{VST}^{-1}(57.68 + -33.14)$).

This estimate from the LMM is substantially lower than the empirical average miss rate for the Scaled Manhattan classifier (23.6%) from Table 4.2. Again, the reason for a discrepancy between the estimate and the empirical average is the non-linearity of the variance-stabilizing transformation. The final three rows of the parameter-estimation table present estimates of the per-user, per-impostor, and residual standard deviations (i.e., square roots of the variances). These standard deviations—17.31, 15.74, and 20.14—are only slightly smaller than most of the classifier fixed-effects. The high variability magnifies the non-linearity in the variance-stabilizing transformation. The size of the discrepancy would seem to be a problem, but the validation (in Section 4.6) will show that the predictions remain quite accurate despite this discrepancy.

Using the standard deviations and Normality assumptions, we can calculate per-user

Selected Model Equation:

$$\begin{aligned} VST(miss\,rate) &= \mu + (Classifier)_i + (user)_j + (impostor)_k + \varepsilon_l \\ (user)_j &\sim N(0, \sigma^2_{(user)}) \\ (impostor)_k &\sim N(0, \sigma^2_{(impostor)}) \\ \varepsilon_l &\sim N(0, \sigma^2_{\varepsilon}) \end{aligned}$$

Parameter Estimates:

Parameters	classifier	estimate
(μ) baseline	Euclidean	57.68
classifier	Manhattan	-16.35
	ScaledManhattan	-33.14
	OutlierCount	-25.31
	AutoAssocNNet	-4.63
	Mahalanobis	-19.34
	KNN	-27.36
	MahalanobisKNN	-23.85
	KMeans	-23.02
	SVM	-26.92
$\sigma_{(user)}$		17.31
$\sigma_{(impostor)}$		15.74
σ_{ε}		20.14

Table 4.3: LMM for the results of the benchmark evaluation. The model equation is at the top. The classifier term was kept during the model-selection process. The maximum-likelihood parameter estimates for the baseline classifier (Euclidean) and the adjustments for the other classifiers are tabulated. The estimates of per-user, per-impostor, and residual standard deviations are at the bottom of the table.

and per-impostor miss-rate prediction intervals. For example, for the Scaled Manhattan classifier, the estimated variance-stabilized miss rate is 24.54, calculated by adding the baseline (57.68) and the Scaled-Manhattan classifier effect (-33.14) from the model. The per-user standard deviation is (17.31), and so the 95% prediction interval (in the variance-stabilized space) is

$$24.54 \pm 1.96 \times 17.31.$$

When we apply the inverse of the variance-stabilizing transformation, the interval is 0.0% to 63.1%.

We interpret a 95% per-user prediction interval as follows. In the long term, a particular user will have some *average* miss rate. While the miss rate may be higher or lower on a given day, against particular impostors, we focus on the long-term average miss rate. The 95% per-user prediction interval is a region in which we predict that 95% of users long-term average miss rates will lie. Such intervals are useful in establishing how a classifier will perform for the large majority of users. The 95% per-user prediction interval for the Scaled Manhattan classifier spans nearly 2/3rds of the range of viable miss rates. This estimate quantifies our observation that the particular user has a major effect on whether a classifier's performance is quite good or quite bad.

Similar calculations can be made to estimate 95% per-user prediction intervals for other classifiers. The only difference in the range of these intervals comes from the non-linearity of the variance-stabilizing transformation. In all cases, the intervals span more than half of the range of viable miss rates; the substantial per-user effect is observed for all classifiers.

Similar calculations can also be used to estimate 95% per-impostor prediction intervals (i.e., describing the long-term average miss rates for the large majority of impostors). We simply substitute the estimated per-impostor standard deviation instead of the per-user standard deviation. For the Scaled Manhattan classifier, the prediction interval is 0.0% to 58.4% (i.e., $\mathrm{VST}^{-1}(24.54 \pm 1.96 \times 15.74)$). The per-impostor interval also covers almost 2/3rds of the viable space. Like the user, the impostor has a substantial effect on whether a classifier's miss rate will be high or low, regardless of the classifier.

These estimates support our observation from the empirical results that, even for the best classifier, there is a lot of uncertainty in the miss rate because some users have much lower miss rates than others, and some impostors have much lower miss rates than others. Something about the typing characteristics of different users and impostors seems to have as much if not more effect on the miss rate than the classifier. We will investigate some aspects of user and impostor typing that might explain these differences in Chapter 5. The high residual standard deviation establishes that, even after the user and impostor effects have been taken into account, there remains a lot of uncertainty when predicting miss rates. This residual variation suggests that other factors, beyond the user and impostor, are also exerting unknown influence on classifier miss rates. We will investigate some of these factors and measure their influence in Chapter 6.

4.5.3 Statistical hypothesis testing

The model-selection and parameter estimation phases of the analysis provide strong evidence that the classifier is an important factor (but not the only one) in determining the miss rate. Such a result is not surprising as it only confirms that some classifiers work better than others. That result alone does not establish which classifiers are better than which other classifiers. For instance, based on Table 4.2 with empirical miss rates and Table 4.3 with parameter estimates, the Scaled Manhattan classifier has one of the lowest miss rates.

The k-NN and SVM classifiers also have low error rates. Are these three classifiers equally good in the long term, or is the Scaled Manhattan classifier uniquely the best? To answer such a question, we must use statistical hypothesis testing to compare individual classifiers.

As discussed in Chapter 3, we use contrasts to compare pairs of classifiers. In this analysis, we use all-pairs contrasts (i.e., Tukey contrasts) to compare the miss rates of each pair of classifiers (Bretz et al., 2011). With 10 classifiers, there are 45 distinct pairings (i.e., *10 choose 2*). Table 4.4 presents the results of these 45 tests. The pairing is presented in the first column. The estimated difference in effect is in the second column (effect), with the sign indicating whether the miss rate for the first classifier is higher (positive) or lower (negative) than the second. The standard error of the estimate and t-statistic are presented in the third and fourth columns. The standard error is calculated from the residual variability and the sample size. The final column lists the p-values for each test; they have been adjusted using the method of Bretz et al. (2011) to accommodate the simultaneous testing of 45 hypotheses.

Just as a procedural note, although our primary interest is in identifying the top performing classifiers, we must perform 45 tests, one for each pair of classifiers. It might seem as though one would only need to perform 9 tests, comparing the top-performing classifier, Scaled Manhattan, to each of the other 9 classifiers. However, to perform such a series of tests, one must already have calculated that the Scaled Manhattan classifier is the top-performing classifier. Implicit in such a calculation is a comparison of each classifier's effect with each other classifier's effect. In other words, to compare the best classifier to the other 9 classifiers, one is actually performing 45 comparisons.

Nearly every pairwise comparison is highly significant ($<.0001$). In particular, at any reasonable level of significance, the Scaled Manhattan classifier has a lower error rate than each of the other 9 classifiers. Based on this analysis, we would conclude that the Scaled Manhattan classifier has the best miss rate.

Given the high uncertainty, due to per-user, per-impostor, and residual variance, one might be surprised by such low p-values. The explanation is that per-user and per-impostor miss rates are highly correlated across classifiers. Some users have miss rates much lower than average, and others have miss rates much higher than average. Nevertheless, users with a low miss rate have low miss rates across all classifiers, and users with a high miss rate have high miss rates across all classifiers. As a result, even though the per-user variability is quite large, one can still show differences between the classifiers by looking at how each user's error rate changes across classifiers. In statistical terms, the classifier effect is being tested *within each subject* rather than *between subjects*, enabling us to identify the Scaled

	effect	stderr	t.stat	p.value
Manhattan - Euclidean	-16.352	0.564	-28.993	<.0001
ScaledManhattan - Euclidean	-33.138	0.564	-58.755	<.0001
OutlierCount - Euclidean	-25.310	0.564	-44.874	<.0001
AutoAssocNNet - Euclidean	-4.627	0.564	-8.203	<.0001
Mahalanobis - Euclidean	-19.342	0.564	-34.294	<.0001
KNN - Euclidean	-27.360	0.564	-48.510	<.0001
MahalanobisKNN - Euclidean	-23.855	0.564	-42.295	<.0001
KMeans - Euclidean	-23.025	0.564	-40.823	<.0001
SVM - Euclidean	-26.916	0.564	-47.722	<.0001
ScaledManhattan - Manhattan	-16.786	0.564	-29.762	<.0001
OutlierCount - Manhattan	-8.957	0.564	-15.881	<.0001
AutoAssocNNet - Manhattan	11.726	0.564	20.790	<.0001
Mahalanobis - Manhattan	-2.990	0.564	-5.301	<.0001
KNN - Manhattan	-11.008	0.564	-19.517	<.0001
MahalanobisKNN - Manhattan	-7.502	0.564	-13.301	<.0001
KMeans - Manhattan	-6.672	0.564	-11.830	<.0001
SVM - Manhattan	-10.563	0.564	-18.729	<.0001
OutlierCount - ScaledManhattan	7.829	0.564	13.881	<.0001
AutoAssocNNet - ScaledManhattan	28.512	0.564	50.552	<.0001
Mahalanobis - ScaledManhattan	13.796	0.564	24.461	<.0001
KNN - ScaledManhattan	5.778	0.564	10.245	<.0001
MahalanobisKNN - ScaledManhattan	9.284	0.564	16.460	<.0001
KMeans - ScaledManhattan	10.114	0.564	17.932	<.0001
SVM - ScaledManhattan	6.223	0.564	11.033	<.0001
AutoAssocNNet - OutlierCount	20.683	0.564	36.671	<.0001
Mahalanobis - OutlierCount	5.967	0.564	10.580	<.0001
KNN - OutlierCount	-2.051	0.564	-3.636	0.0106
MahalanobisKNN - OutlierCount	1.455	0.564	2.580	0.2265
KMeans - OutlierCount	2.285	0.564	4.051	0.0019
SVM - OutlierCount	-1.606	0.564	-2.848	0.1210
Mahalanobis - AutoAssocNNet	-14.715	0.564	-26.091	<.0001
KNN - AutoAssocNNet	-22.733	0.564	-40.306	<.0001
MahalanobisKNN - AutoAssocNNet	-19.228	0.564	-34.091	<.0001
KMeans - AutoAssocNNet	-18.398	0.564	-32.620	<.0001
SVM - AutoAssocNNet	-22.289	0.564	-39.518	<.0001
KNN - Mahalanobis	-8.018	0.564	-14.216	<.0001
MahalanobisKNN - Mahalanobis	-4.512	0.564	-8.000	<.0001
KMeans - Mahalanobis	-3.683	0.564	-6.529	<.0001
SVM - Mahalanobis	-7.573	0.564	-13.428	<.0001
MahalanobisKNN - KNN	3.506	0.564	6.215	<.0001
KMeans - KNN	4.335	0.564	7.687	<.0001
SVM - KNN	0.444	0.564	0.788	0.9988
KMeans - MahalanobisKNN	0.830	0.564	1.471	0.9037
SVM - MahalanobisKNN	-3.061	0.564	-5.427	<.0001
SVM - KMeans	-3.891	0.564	-6.899	<.0001

Table 4.4: Hypothesis tests comparing classifiers in the benchmark evaluation. Between all 10 classifiers, there are 45 pairs of classifiers, and test results for each each pairing are presented. The p-values (adjusted for multiple testing) are accompanied by the effect size, standard error estimate, and t-statistic. For most pairs of classifiers, the p-values show highly significant differences in the miss rates.

Manhattan as the unique classifier (among the 10) with the lowest average miss rate.

Only 4 comparisons produce p-values above $.05$. These 4 comparisons form a sequence of 5 classifiers: k-NN, SVM, Outlier Count, Mahalanobis k-NN, k-Means. Each consecutive pair of classifiers in the sequence (e.g., k-NN–SVM) has a p-value above $.05$, which we interpret to mean there is little evidence that the classifier miss rates differ substantially. These five classifiers all have similar empirical miss rates in Table 4.2. This result suggests that, even though the classifiers would seem to learn very different concepts, they exhibit similar overall behavior on keystroke-dynamics data.

4.6 Validation

While model-selection and statistical hypothesis-testing procedures are designed to reduce or limit the possibility of drawing incorrect conclusions, they are necessarily based on various modeling assumptions. In the end, we believe that the value of any model lies in its ability to predict the future behavior of a classifier. Consequently, to validate the model, we make predictions and then check those predictions in a second evaluation.

We collected a second data set using the same procedure. For this secondary data collection, we had 14 subjects type the same strong password (.tie5Roanl) 400 times over 8 sessions. The same software and timing apparatus were used. With this second, smaller data set, we perform the same evaluation procedure. Each of the 10 classifiers was trained, tuned, and tested in a series of trials. The evaluation procedure is the same one described in Section 4.3.3, substituting the 14-subject typing data from the secondary evaluation for the 51-subject typing data used in the primary evaluation. With this smaller data set, we obtain 1820 per-user/impostor miss rates (10 classifiers $\times$ 14 genuine-user subjects $\times$ 13 impostor subjects).

Table 4.5 presents the average miss and false-alarm rates for all the classifiers in this validation data set. As in the primary evaluation, the classifiers have been tuned to have a 5% false-alarm rate. Again, the Outlier Count classifier has coarse anomaly scores, and the operating threshold was tuned as close as possible to a 5% false-alarm rate. Each classifier's average miss rate is within a few percentage points of the corresponding average miss rates from the original evaluation (Table 4.2).

Our first step in validating the model is to check the per-user predictions. Earlier, in Section 4.5, we explained how to calculate 95% per-user prediction intervals for each classifier. These intervals enclose a region in which we expect that 95% of users' long-term miss rates will lie. Using the validation data, we can estimate long-term miss rates for each

Classifier	False-Alarm Rate	Miss Rate
ScaledManhattan	5.0	20.6
KNN	5.0	27.1
SVM	5.0	29.4
OutlierCount	2.6	31.7
MahalanobisKNN	5.0	30.3
KMeans	5.0	33.0
Mahalanobis	5.0	35.9
Manhattan	5.0	47.2
AutoAssocNNet	5.0	50.5
Euclidean	5.0	57.9

Table 4.5: Average error rates for classifiers on the validation set. On the 14-subject validation data, the 10 classifiers were trained and tuned to have 5% miss rates (or as close as technically possible). The classifiers are sorted in order of their miss rates on the evaluation data set (Table 4.2 on page 85). The average miss rates are similar on this secondary data set.

of the users. For each classifier and each genuine-user subject, there are 13 miss rates, corresponding to each of the 13 impostor subjects. The average of these 13 miss rates is a reasonable estimate of the long-term miss rate for that genuine-user subject. We calculate 14 per-user average miss rates for each classifier. By examining whether these 14 points fall within the predicted region, we can demonstrate whether the model is making accurate predictions about classifier performance.

Figure 4.7 presents the per-user 95% prediction intervals for each of the classifiers. Along with the intervals, we have plotted the 14 per-user average miss rates for each classifier from the validation data set. Nearly all of these points fall within the corresponding prediction interval, demonstrating that the predictions are accurately bounding the range of observed behavior. That only one point falls outside the interval suggests that the intervals might be slightly conservative (i.e., overly long). With 95% prediction intervals, we would expect 1 point out of 20 to exceed the interval, and the figure contains 140 points. Nevertheless, for most classifiers, the points are distributed fairly consistently around the entire range of the interval, suggesting that the intervals are not very conservative.

Checking that the per-user miss rates are within the prediction intervals is only part of validating the model. The per-user, per-impostor, and residual distributions are assumed to be Normally distributed with variance estimated from the data. These prediction intervals only concern the validity of the per-user variability.

Our second step in validating the model is to check these modeling assumptions more rigorously. As explained in the explanation of the validation step in Chapter 3 (Section

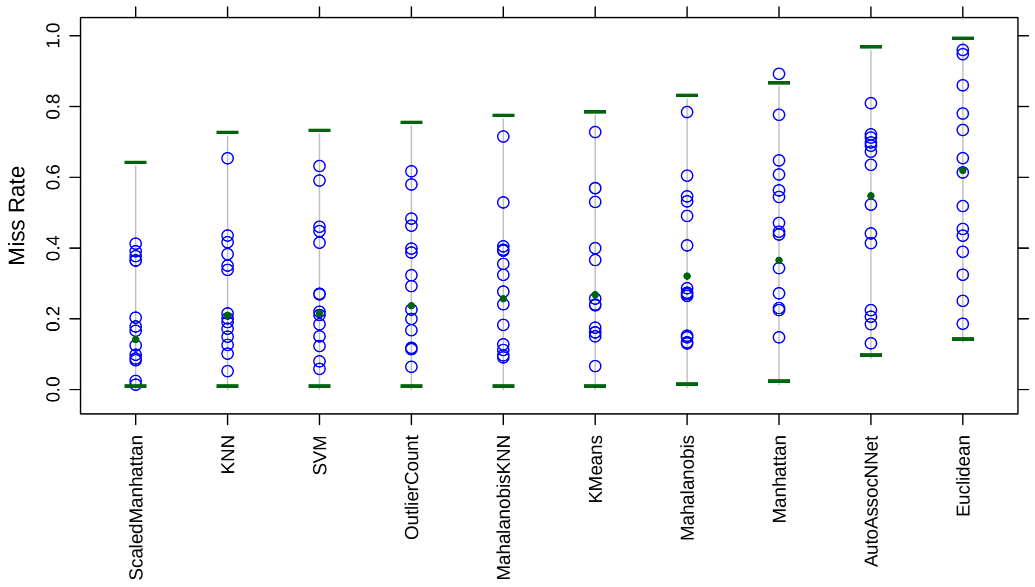


Figure 4.7: Model predictions and validation data for the benchmark evaluation. The horizontal lines delineate the 95% prediction intervals for each classifier. The blue dots indicate the per-user miss rates for the 14 users in the validation data set. Nearly all results fall within the predicted intervals.

3.10), we calculate per-user effects, per-impostor effects, and residual effects by comparing the results of the secondary evaluation with the model predictions. Through this process, we find 14 per-user effects (one for each subject), 14 per-impostor effects (one for each subject), and 1820 residual miss rates (one for each classifier, genuine-user, and impostor subject). All three sets of points are assumed to be Normally distributed with zero mean and variance predicted by the model. Validating the model involves checking these assumptions.

The variances of these Normal distributions were estimated as part of the model. If one divides a set of Normally distributed values with zero mean by their standard deviation, the resulting values should have a standard Normal distribution (i.e., zero mean and unit variance). We perform this scaling calculation on the per-user effects, per-impostor effects, and residuals using the standard deviations from the model to obtain *standardized* per-user effects, per-impostor effects, and residuals.

To assess whether the Normality assumptions are met for these standardized effects and residuals, we use QQ-plots. Figure 4.8 shows the QQ-plots for the 14 standardized per-user effects (on the left), the 14 standardized per-impostor effects (in the middle), and

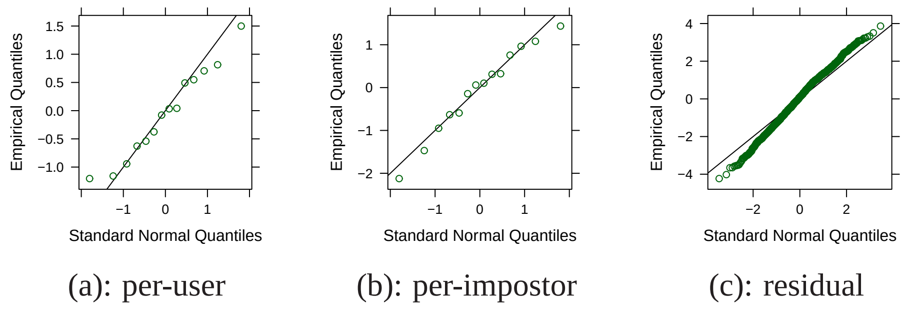


Figure 4.8: QQ-plots for validating LMM predictions from the benchmark evaluation. The left panel has per-user deviations from Normality; the middle panel has per-impostor deviations; the right panel has residual deviations. Per-user and per-impostor deviations meet expectations. Residuals are somewhat heavy tailed.

the 1820 standardized residuals (on the right). In the panels on the left and in the middle, assessing the Normality of the per-user and per-impostor effects respectively, the points closely adhere to the diagonal line. These panels support the Normality assumption for these effects. In the panel on the right, assessing the Normality of the residuals, the points have a slightly sinusoidal shape. This shape is indicative of slightly heavy tails.

In general, some violation of the modeling assumptions is acceptable and to be expected. Heavy-tailed residuals suggest that while our model can accurately predict average effects for users and impostors, it falls a little short when predicting how much variation to expect for a particular user or impostor. For a single user, there can be day-to-day variation in the miss rate for unknown reasons. Even with slightly heavy-tailed residuals, QQ-plots demonstrate that the model constructed from the benchmark evaluation makes accurate predictions of error rates in a secondary evaluation. The model has captured not only the average error rates of each classifier, but also how much per-user, per-impostor, and residual variability to expect.

4.7 Discussion

Recall that the goal of this work is to understand the factors that affect classifier error rates. Our analysis shows how much more can be learned by modeling the effects of individual users and impostors on error rate, rather than simply tabulating the average miss rates of each classifier. The average miss rates did reveal which classifier did best overall in the evaluation (i.e., Scaled Manhattan), but they told us nothing about whether we could expect individual user miss rates to be close to the overall average. In fact, analysis of those individual miss rates revealed that regardless of the classifier's average miss rate, some user's have extremely low miss rates and others have very high miss rates. Likewise, some impostors are substantially more difficult to detect than others. Reports of only the average error rate may potentially mislead a reader into thinking that individual results will not vary substantially.

For instance, the average miss rate for the Scaled Manhattan classifier is 23.6%; the average miss rate for the Euclidean classifier is 61.0%, a seemingly large difference. However, when we estimate prediction intervals based on the per-user standard deviation, we find that with the Scaled Manhattan classifier, 95% of per-user average miss rates will be between 0.0% and 63.1%, while with the Euclidean classifier, 95% of per-user average miss rates will be between 13.3% and 98.3%. One might get pretty good results with a bad classifier (e.g., 13.3% with the Euclidean classifier) or pretty bad results with a good classifier (e.g., 63.1% with the Scaled Manhattan classifier).

Statistical hypothesis testing confirmed that the Scaled Manhattan classifier had the lowest error rate, but the actual difference between that classifier and other good classifiers (e.g., k-NN and SVM) was small compared to the per-user and per-impostor standard deviation. This observation suggests that we may have reached a point of diminishing returns for using new classification technology to improve keystroke-dynamics error rates. Most classifiers' average miss rates are within a few percentage points of the others. The classifier effect is comparatively small. Different users and impostors sway the miss rates by tens of percentage points. We might see greater improvement in keystroke-dynamics error rates by understanding these larger effects.

This discovery of high per-user and per-impostor variability also presents a problem for keystroke-dynamics evaluations. With such high variability, one needs many subjects to accurately estimate error rates. Specifically, suppose we conduct an evaluation like that described in this chapter, and we determine that a classifier has a 30% miss rate (or 36.9 in the variance-transformed space). Assuming that the per-user standard deviation is roughly what we estimated earlier in this chapter ($\sigma_{(user)} = 17.31$ in the variance-transformed space), we can estimate the 95% confidence interval with the following equation:

$$CI_{95\%} = \mathrm{VST}^{-1}(36.9 \pm 1.96 \times 17.31/\sqrt{n})$$

where n is the sample size. In this case, a single sample corresponds to a subject. Note that the confidence interval is not the same as a prediction interval. The confidence interval estimates the range in which the true miss rate will fall 95% of the time. If n is greater than 1, the confidence interval will be narrower than the prediction interval. As n gets larger, the confidence interval gets narrower.

By translating the interval back from the variance-transformed space to the space of error rates, we can calculate how many subjects are needed for the confidence interval to narrow to under ± 1 percentage point. With 50 subjects, the 95% confidence interval is 23.3% to 37.1%. With 100 subjects, the confidence interval only narrows to 25.2% to 35.0%. To narrow the interval to ± 1 percentage point (i.e., 29.0%–31.0%), one needs 2401 subjects. Few studies have that many genuine-user subjects, and so the uncertainty in many error-rate estimates (i.e., the "margin of error") is likely to be several percentage points wide.

As noted earlier in this chapter, many different classifiers have been proposed, and some studies which proposed a new classifier did compare its performance to that of one or more existing classifiers. However, as we saw in the literature review from Chapter 2, very few studies in keystroke dynamics used any inferential statistics to generalize from their empirical results.

When a study uses no inferential statistics, it is difficult to compare the results to ours. For instance, Harun et al. (2010) evaluated many of the same classifiers as this work (e.g., Euclidean, Manhattan, Mahalanobis, and Scaled Manhattan). They calculated equal-error rates (EERs) for each of the classifiers and presented them in a histogram. The EERs ranged from about 5% to 19%, but with no error bars or other statistical analysis, one cannot tell whether the results suggest a significant difference between the classifiers, or whether they are all about the same.

The few studies that used inferential statistics in comparing classifiers raise some interesting concerns in relation to our work. Specifically, Cho et al. (2000) evaluated a Mahalanobis k-NN and an Auto-Associative Neural Network; they used a paired t-test and established that the Neural Network was significantly better than the Mahalanobis k-NN ($p \approx 0.00079$). In this work, we find that the Neural Network does substantially worse than the Mahalanobis k-NN. Explaining discrepancies in the results of different evaluations is of fundamental importance to the practical use of keystroke dynamics.

In search of explanations, one might ask whether classifier implementation differences are the cause of these discrepancies. Because we used our own implementations of the classifiers evaluated in this work, small implementation differences may change performance. However, our classifier implementations were tuned to obtain good performance on our typing data. We expect that other researchers likewise tuned their classifiers. Consequently, we believe that implementation differences cannot explain the wild differences in classifier error rates across study; some other factors must be responsible for the differences. In the study by Cho et al., subjects chose their own passwords, those with inconsistent typing were removed from the study, and 75–325 training samples were used for each genuine-user subject (depending on the amount of data collected from the subject). In the current work, all subjects used the same password, no subjects were filtered from the study, and 200 training samples were used for all genuine-user subjects. These or any other differences may explain why different classifiers perform best in different evaluations. Identifying and understanding these differences spawned the remaining investigations in this work.

During the model-validation step, we effectively collected a second data set and used it to test whether the model built from the first data set was accurate. This process might be described as a replication since the first evaluation was an experiment and the second evaluation was conducted as an attempt to reproduce the first experiment's results. Replication is one of the hallmarks of the scientific process, but to our knowledge this work contains one of the few successful replications in keystroke dynamics. Replication is important because it demonstrates a reliable understanding of the phenomena under study (e.g., classifier error rates). By repeating the evaluation with new genuine-user and impostor subjects, we have effectively shown an accurate understanding of how these classifiers behave, at least in the evaluation environment.

4.8 Summary

This chapter presented our first investigation using LMMs to understand classifier behavior in keystroke-dynamics. A benchmark evaluation of 10 classifiers was conducted to identify the top-performing classifiers. LMMs were fit to the per-user/impostor miss rates for each classifier. The models were used to accurately predict the results of a subsequent evaluation thereby validating the model.

From this first investigation, we draw the following conclusions:

1. *The Scaled Manhattan classifier has an estimated miss rate of* 14.1*%, significantly lower than those of the other 9 benchmarked classifiers.* This conclusion depends, obviously, on the details of the evaluation (e.g., the use of a strong password, tuning classifiers to a 5% false-alarm rate, etc.). Nevertheless, if one were to use this evaluation to choose the best classifier, Scaled Manhattan would be it.
2. *However, the estimated 95% prediction interval for per-user long-term average miss rates is from* 0.0*% to* 63.1*% for Scaled Manhattan, and spans a similarly large range for the other classifiers.* Consequently, while Scaled Manhattan may have the best average miss rate, the actual miss rate seems to depend more on the user than on the

classifier.

3. *Likewise, the estimated 95% prediction interval for per-impostor long term average miss rates is from* 0.0*% to* 58.4*% for Scaled Manhattan, and similarly large for the other classifiers.* Some impostors are substantially harder to detect than others, regardless of the user.
4. *Because of the high per-user effect on miss rates, an accurate estimate of a classifier's error rate (i.e., to within* ± 1 *percentage point) may require thousands of subjects.* Few studies use so many subjects, so one explanation for why the same classifier is reported to have wildly different error rates across studies may be the sample size. Fortunately, because classifier error rates seem to be highly correlated across subjects, hypothesis tests require substantially fewer subjects (e.g., 30–50) to find a significant difference between two classifiers' error rates.

These results support our claim that there are many factors beyond the classifier itself that affect keystroke-dynamics error rates. For keystroke dynamics in the short term, these results may seem somewhat discouraging. In the long term, the results suggest we should focus on identifying and understanding these factors as a way to make progress in keystroke dynamics.

In the thesis statement from Chapter 1, we claimed in part that LMMs offer better understanding of classifier behavior than current practices. In this investigation, we reviewed earlier work and showed that—because most classifiers were evaluated in different evaluation environments—few conclusions could be drawn about which classifier was the best. Then, by conducting a benchmark and using LMMs to analyse the result, we established which classifier performs best (i.e., Scaled Manhattan) in a consistent evaluation environment. We also discovered other factors, namely the user and impostor, that have at least as much effect on miss rates as the classifier. Consequently, the investigation has furthered our understanding of keystroke-dynamics classifier behavior.

Chapter 5

Personal Traits and Keystroke Dynamics

The investigation in this chapter attempts to identify some of the sources of per-user and per-impostor variation in classifier error rates. In particular, could personal traits such as age, gender, dominant hand and typing style explain why some typists (or pairs of typists) are more difficult to distinguish than others?

5.1 Background

In the last chapter, we discovered that the user and impostor have a substantial effect on the miss rate of a classifier. To better understand how and why the particular user and impostor affect classifier error rates, we consider various traits that might make a typist easier or harder to distinguish from others.

There is some surprising evidence that even traits such as age and gender might affect typing characteristics (and consequently affect classifier behavior). The Halstead-Reitan Battery is a sequence of neuropsychological tests which include a finger-tapping test (Spreen and Strauss, 1998). This test measures the rate at which a subject repeatedly pushes a button. Test results have been shown to correlate with a subject's age, gender, dominant hand, overall health, and even whether he or she is putting maximal effort into the task (Carlier et al., 1993; Okuno et al., 2006; Wall and Millis, 1998). Since using a keyboard is a lot like pushing a button repeatedly, perhaps these same traits manifest in different typing rhythms.

Anything that affects typing times and rhythms could plausibly affect the error rates of keystroke-dynamics classifiers. For instance, right-handed typists might type digraphs on the right side of the keyboard more quickly and consistently, while left-handed typists type digraphs on the left side more quickly and consistently. Consequently, miss rates might be lower for right-handed users and left-handed impostors. Touch typists might have lower miss rates because their more consistent typing rhythms are more easily recognized by a classifier than the inconsistent rhythms of a hunt-and-peck typist. Alternatively, touch typists may have higher miss rates because they all type with the same or very similar rhythms. Regardless of the direction of the effect (i.e., higher or lower), it is plausible that these traits might affect classifier error rates.

Very little work has been done to establish what effect personal traits have on keystroke-dynamics error rates. Potential effects have certainly been acknowledged. Some earlier studies have provided breakdowns of subjects in terms of age, gender, and other personal traits. For instance, Hosseinzadeh and Krishnan (2008) and El-Abed et al. (2010) both tabulated the number of subjects by age and gender. Kotani and Horii (2005) reported that all the subjects in their evaluation data set were right-handed. Dozono and Nakakuni (2008) report that their subjects ranged in skill from "standard computer user to almost blind typist." By reporting these subject demographics, the researchers imply that the reported personal traits may be important.

A few studies went a step further in attempting to compare classifier error rates for different levels of these factors. Tran et al. (2007) reported per-gender average EERs. Their results suggest that the EERs for their female subjects were higher, but no inferential statistics were used to draw that conclusion. Likewise, Hosseinzadeh et al. (2006) found that classification results were particularly poor for "two finger" typists, but again, no inferential statistics were used to test the claim.

Hwang et al. (2009a) did perform a statistical test involving the effects of personal traits on classifier error rates. They tested whether typists who used both hands to type had higher or lower equal-error rates. Based on the reported p-value, they found no significant effect, but their experiment suggested that further investigation of this issue is warranted. The details of the Hwang et al. experiment differ substantially from the present investigation. In particular, they were interested in whether artificial rhythms and cues improve keystroke-dynamics error rates, and they considered the effect of different typing styles in this context.

5.2 Aim and approach

Our aim in this investigation is to understand which (if any) personal traits of genuine users and impostors affect the error rates of keystroke-dynamics classifiers. Specifically, we focus on four traits: age, gender, dominant hand, and typing style. We investigate whether any traits—individually or shared between the user and the impostor—affect miss rates.

We supplement the results from the benchmark evaluation in Chapter 4 with information collected from a demographic survey filled out by each of our subjects. Then, we conduct a statistical analysis of the demographic data using LMMs to identify traits that might explain the per-user and per-impostor variability in classifier error rates. Finally, we validate these models as before using data from a secondary evaluation.

5.3 Experimental method

We administered a lengthy demographic survey to the subjects who provided typing data for our benchmark evaluation in Chapter 4. We collected information about age, gender, dominant hand, and typing style, and we merged this information with the benchmark evaluation results.

5.3.1 Demographic survey

The 51 subjects who provided typing times for the benchmark evaluation and the 14 subjects who provided typing times for the validation data set were asked to complete a demographic survey. The survey was extensive, comprising 9 pages and covering a variety of topics in computer usage, keyboard preferences, and other activities that affect manual dexterity. Our focus in this work is on four questions in the survey pertaining to age, gender, dominant hand, and typing style.

Age: The survey used the following question to assess age.

Write your age here (or check one age group): ____________

☐ 18–20	☐ 21–25	☐ 26–30	☐ 31–35	☐ 36–40
☐ 41–45	☐ 46–50	☐ 51–60	☐ 61–70	☐ 71–80

In retrospect, we might have structured this question differently, since subjects responded in unanticipated ways. Some checked a box; others wrote in their age; still others wrote in a range that sometimes did and sometimes did not correspond to one of the ranges explicitly offered. Nevertheless, for this work, we were able to accommodate the different kinds of answers by defining two age groups: 18–30, and 31+. With these two groups, we can observe whether or not there are any major differences in typing times of older and younger subjects.

Gender: In the demographic survey, subjects were asked their gender. All subjects responded to the question, identifying as Male or Female.

	Age Range		Gender		Dom. Hand		Typ. Style	
	18-30	31+	Male	Female	Right	Left	Touch	Other
Evaluation Pool	31	20	30	21	43	8	28	23
Validation Pool	7	7	8	6	13	1	9	5

Table 5.1: Summary of the subjects' age, gender, dominant hand, and typing style. Results in the first row are for the 51 subjects in the primary evaluation data set. Results in the second row are for the 14 in the secondary validation data set. All traits were reported by subjects via a questionnaire. In most categories, the subjects are balanced fairly evenly among the groups.

Dominant Hand: The survey asked subjects to identify as either left-handed, right-handed, or ambidextrous. All subjects indicated that they were either left- or right-handed.

Typing Style: The demographic survey asked subjects to assess their typing style with the following question:

What is your level of typing skill in English? (Circle one)

1	2	3	4	5
Hunt and peck		Average		Touch type (without looking at keyboard)

Again, in retrospect, we might have structured this question differently. Subjects were given no real guidance on interpreting levels 1–4, but level 5 was defined in straightforward operational terms (i.e., typing without looking at the keyboard). Since we are looking for clear effects at this point, we labeled subjects who circled option 5 as *touch*, and those who circled any of the other 4 options as *other*.

Each of the four traits has been made to take binary values: 18–30/31+, male/female, left-handed/right-handed, and touch-typist/other. Table 5.1 presents the demographics of our subjects according to this survey. The first row presents results for the 51 subjects in the primary evaluation, used to build the statistical model. The second row presents results for the 14 subjects in the secondary evaluation, used for validating the model.

When building a statistical model, a typical heuristic is that one should have at least five participants in each group, and we do have at least five subjects in each group in the primary evaluation pool. For three of the four traits (i.e., age, gender, and typing style), the numbers are fairly or very balanced. For the fourth trait (dominant hand), the numbers are skewed, but the proportions are in keeping with the prevalence of left-handedness in the population (about 10%).

We have enough data to investigate the main effects of each trait and whether there is an interaction between a genuine user's traits and an impostor's traits (e.g., whether a user and impostor who share the same traits are harder to distinguish). We will not be able to look for interactions between the four traits with the data available. In the primary evaluation pool, we have only 4 left-handed subjects in the 18–30 year age range, and 4 left-handed subjects in the 31+ age range; none in the latter group are female.

5.3.2 Merged demographic and evaluation results

Recall that in Chapter 4 we performed a benchmark evaluation of 10 classifiers using 51 subjects. Each subject typed the password .tie5Roanl 400 times. For each classifier and each subject, we designated the given subject as the genuine user and trained the classifier using 200 of his or her typing samples. Then, we tuned the classifier to have a 5% false-alarm rate using the remaining 200 samples from the genuine-user subject.

For each trained and tuned classifier, we designated the other 50 subjects as impostors, and we classified the first 50 samples from each impostor. We recorded the miss rate for each impostor. This process resulted in 25,500 miss rates, one for each of the 10 classifiers, 51 genuine-user subjects, and 50 impostor subjects.

Based on the results of that benchmark, we narrow our focus from ten classifiers to seven: Scaled Manhattan, Outlier Count, Mahalanobis, k-NN, Mahalanobis k-NN, k-Means, and SVM. While that earlier investigation concluded that Scaled Manhattan was the best of the classifiers, all seven had comparatively low miss rates (below 40%). Since user and impostor traits may affect different classifiers differently (e.g., some are better at distinguishing right-handed typists than others), we decided to include these seven classifiers in this investigation.

Using the results of the demographic survey, we matched each of the 25,500 miss rates with the four demographic characteristics (i.e., age, gender, dominant hand, and typing style) for the corresponding genuine-user subject and impostor subject. With this supplemental information, the evaluation results now comprise 25,500 miss rates along with (1) the classifier, (2) the genuine-user subject's ID, (3) the impostor subject's ID, (4) the user's age, (5) the user's gender, (6) the user's dominant hand, (7) the user's typing style, (8) the impostor's age, (9) the impostor's gender, (10) the impostor's dominant hand, and (11) the impostor's typing style.

5.4 Empirical results

Figure 5.1 presents the classifier miss rates for each of the various pairings of user's and impostor's age and gender. The meaning of the dots (medians), boxes (quartiles), whiskers (min and max), and points (outliers) are the same as in Chapter 4. Recall that in the benchmark evaluation we tuned each classifier so that the false-alarm rate for each genuine-user subject was fixed at 5%. Then, we compared classifiers based on their miss rates. In this section we continue with this approach, analyzing whether different user and impostor personal traits have an effect on miss rates.

In part (a), the four panels present results from the four age matchups: 18–30 and 31+ genuine-user and impostor subjects. Looking only at the medians in each panel, we see a gradual increase in the median miss rate from the Scaled Manhattan classifier on the left to the Mahalanobis classifier on the right. Across panels, each classifier's boxplots look very similar. The boxes and whiskers span the whole space of possible miss rates, suggesting that variability remains high even after age is taken into account. The medians appear to be a few percentage points higher for the panels on the right (i.e., for 31+ users), but only a more formal analysis will show whether this visual impression is significant.

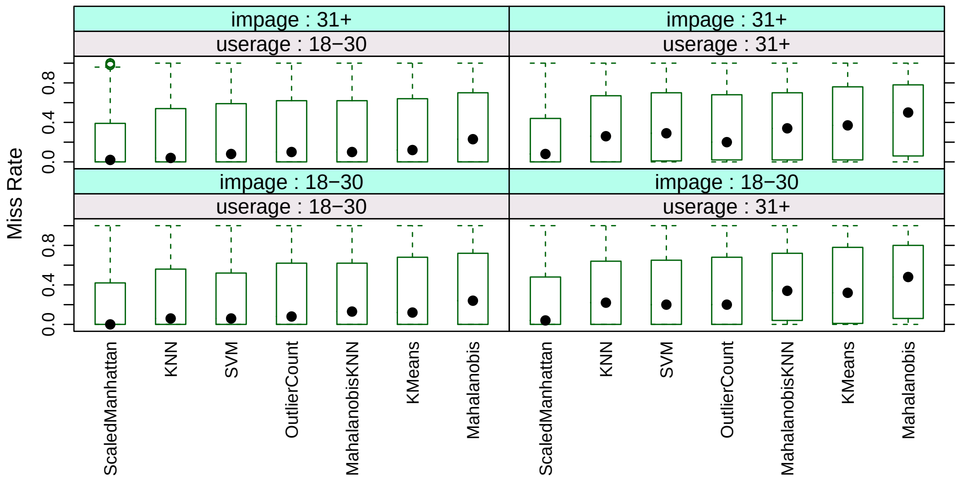
In part (b), the four panels present results from the four matchups of male/female users and male/female impostors. The most noticeable difference between panels is the smaller boxplot for the Scaled Manhattan classifier for female users. The SVM and Outlier Count classifiers also seem to have lower median miss rates for these users. Median miss rates on the bottom row of panels may also be slightly higher than those in the top row (i.e., higher for male impostors).

Figure 5.2 presents the miss rates separated by the remaining two traits: dominant hand and typing style. In part (a), many of the boxplot features (i.e., the maximum, 3rd-quartile marker, and medians) appear to be lower for left-handed users. This observation suggests perhaps that left-handed typists are easier to distinguish from other typists. The dominant hand of the impostor does not appear to have much effect.

In part (b), the four panels present results from the various matchups of touch-typing and non-touch-typing users and impostors. The most striking contrast is between the top-right panel (both touch typists) and the lower-left panel (neither touch typists). The medians and boxplot quartiles are all substantially higher when the user and impostor are both touch typists.

In summary, the empirical results do suggest that the personal traits of the user and the impostor may make distinguishing the two typists easier or harder. Of course, one of

Part (a): Age as a factor



Part (b): Gender as a factor

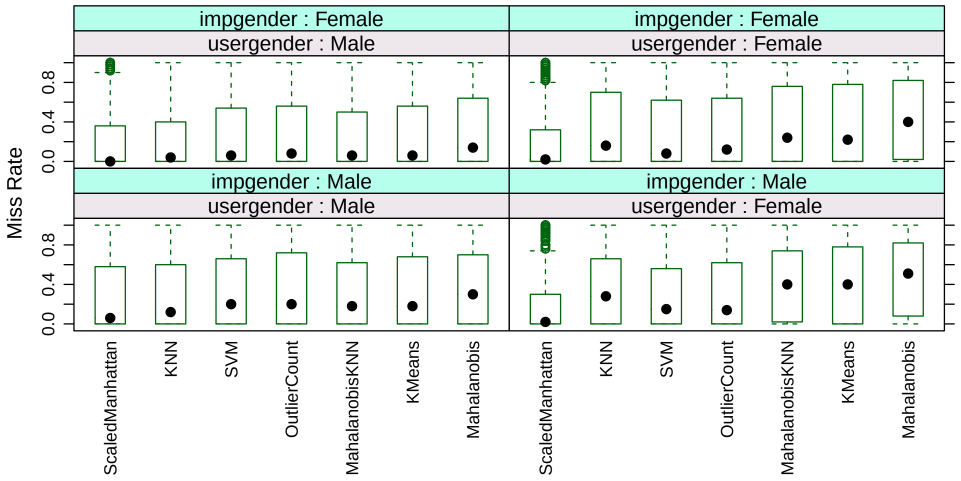
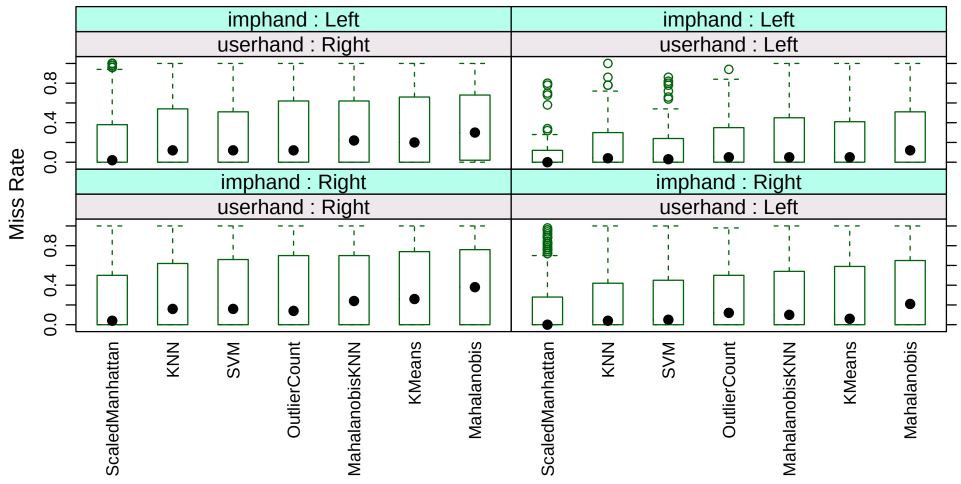


Figure 5.1: Miss rates by user and impostor age and gender. Each of the panels presents one pairing of user and impostor traits. In part (a), separation is by user and impostor age range. Medians may be somewhat higher for 31+ users. In part (b), separation is by user and impostor gender. Medians are somewhat higher for male impostors; SVM and Outlier Count classifiers perform somewhat better for female users.

Part (a): Dominant hand as a factor



Part (b): Typing style as a factor

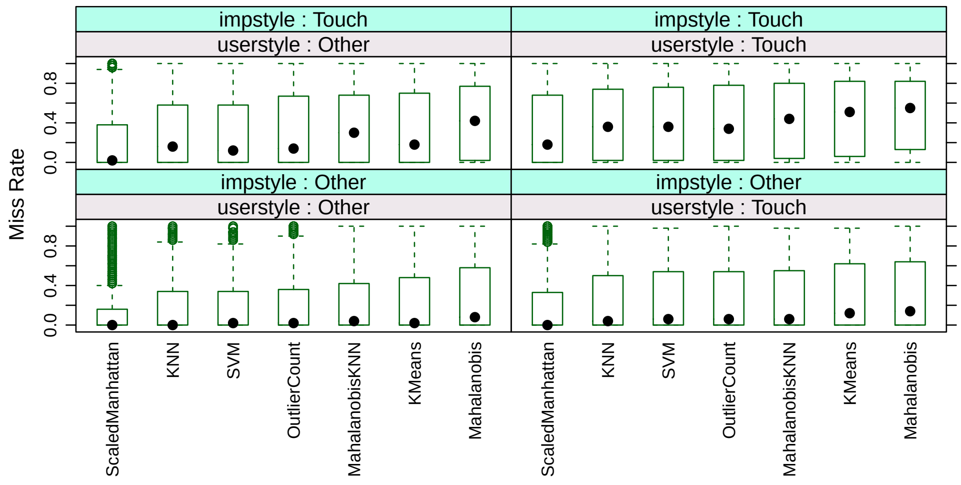


Figure 5.2: Miss rates by user and impostor dominant hand and typing style. Each of the panels presents one pairing of user and impostor traits. In part (a), separation is by user and impostor dominant hand. Medians and boxplot quartiles seem somewhat lower for left-handed users. In part (b), separation is by user and impostor typing style. Medians and boxplot quartiles appear to be substantially higher for touch-typing impostors, particularly when the user is also a touch typist.

Minimal Model Equation:

$$\begin{aligned}
\text{VST}(\textit{miss rate})_{ijk} &= \mu + (\textit{user})_i + (\textit{impostor})_j + \varepsilon_k \\
(\textit{user})_i &\sim N(0, \sigma^2_{(\textit{user})}) \\
(\textit{impostor})_j &\sim N(0, \sigma^2_{(\textit{impostor})}) \\
\varepsilon_k &\sim N(0, \sigma^2_{\varepsilon})
\end{aligned}$$

Maximal Model Equation:

$$\begin{aligned}
\text{VST}(\textit{miss rate})_{ijklmnopqrst} &= \mu + (\textit{Classifier})_i \times (\textit{User Age})_j \times (\textit{Imp. Age})_k \\
&\quad + (\textit{Classifier})_i \times (\textit{User Gender})_l \times (\textit{Imp. Gender})_m \\
&\quad + (\textit{Classifier})_i \times (\textit{User Hand})_n \times (\textit{Imp. Hand})_o \\
&\quad + (\textit{Classifier})_i \times (\textit{User Style})_p \times (\textit{Imp. Style})_q \\
&\quad + (\textit{user})_r + (\textit{impostor})_s + \varepsilon_t \\
(\textit{user})_r &\sim N(0, \sigma^2_{(\textit{user})}) \\
(\textit{impostor})_s &\sim N(0, \sigma^2_{(\textit{impostor})}) \\
\varepsilon_t &\sim N(0, \sigma^2_{\varepsilon})
\end{aligned}$$

Figure 5.3: Equations for model selection in the personal-trait investigation. The minimal equation includes only the random effects which are part of the structure of the experiment. The maximal equation also includes fixed effects for the classifier, the 4 personal traits of the genuine user, and the 4 personal traits of the impostor. All interactions between the classifier and corresponding user and impostor traits are also included. The $A \times B \times C$ notation is shorthand for including the three-way interaction term and all lower-order terms involving factors A, B, and C.

the running themes of this work is that inferential statistics are required to draw general observations from empirical results, and so a more formal analysis follows.

5.5 Statistical analysis

As with the original benchmark data, we analyze the results using linear mixed-effects models, proceeding through the model selection, parameter estimation, and hypothesis-testing stages.

5.5.1 Model selection

Figure 5.3 presents the minimal and maximal model equations used to define the search space for model selection. The minimal equation only contains the user and impostor random effects, which we include in all models of miss rates. The maximal model includes fixed-effect terms for the classifier, the four personal traits of the user, and the four personal traits of the impostor. Interaction terms between the classifier, the user traits and the impostor traits are also included in the maximal model. Missing are interactions between different traits; as noted earlier, investigation of such interactions cannot be supported by our data set. For instance, an interaction between the user's typing style and the impostor's typing style is included, but no interactions between the user's typing style and user's age, or between the user's typing style and the impostor's age are included.

We used stepwise model selection starting from the maximal model equation. We removed terms based on the term whose removal resulted in the smallest BIC score. After iteratively applying this procedure until we arrived at the minimal model equation, we selected the model with the lowest BIC score of all those seen during the search.

5.5.2 Parameter estimation

Once the model equation was selected, we repeated the parameter-estimation procedure using REML rather than maximum-likelihood, as explained in Chapter 3. Figure 5.2 presents the model equation and the parameter-estimates from this procedure. Note, again, that the estimates are in the variance-transformed space; they are not directly interpretable as miss rates.

Looking first at the model equation, we see some terms that we expected to see, and some terms that we found somewhat surprising. The presence of user and impostor typing style and an interaction between them was expected from looking at the empirical results in Figure 5.2, part (b). The presence of user and impostor gender are somewhat more unexpected. Of the various possible effects we observed in the boxplots of the empirical results, we did not anticipate the inclusion of these effects. In retrospect, revisiting Figure 5.1, part (b), the interaction term between the classifiers and the genuine-user gender is understandable. Some classifiers did have markedly lower median miss rates for female genuine-user subjects.

Turning to the parameter-estimate table, note that the baseline situation includes the Scaled Manhattan classifier and male/non-touch-typist users and impostors. The remaining fixed-effect estimates are adjustments from changing the baseline conditions. The baseline estimate of 19.28 corresponds to a miss rate of 8.9%. Note that this is substantially lower than the estimate for the Scaled Manhattan classifier in the last chapter. In Chapter 4, the per-classifier estimates averaged over all users, whereas here the estimate is only for a particular subset of genuine-user and impostor typists who are easiest to distinguish. As the rest of the analysis will show, other sets of typists increase the estimated miss rates.

Looking down the parameter-estimate table, we note that the Scaled Manhattan classi-

Selected Model Equation:

$$\begin{aligned} VST(miss\ rate)_{ijklmnop} &= \mu + (Classifier)_i \times (User\ Gender)_j \\ &\quad + (User\ Gender)_j \times (Imp.\ Gender)_k \\ &\quad + (User\ Style)_l \times (Imp.\ Style)_m \\ &\quad + (user)_n + (impostor)_o + \varepsilon_p \\ (user)_n &\sim N(0, \sigma^2_{(user)}) \\ (impostor)_o &\sim N(0, \sigma^2_{(impostor)}) \\ \varepsilon_p &\sim N(0, \sigma^2_{\varepsilon}) \end{aligned}$$

Parameter Estimates:

Parameters	classifier	usergender	impgender	userstyle	impstyle	estimate
(μ) baseline	ScaledManhattan	Male	Male	Other	Other	19.28
classifier	OutlierCount					6.81
	Mahalanobis					8.31
	KNN					1.42
	MahalanobisKNN					4.29
	KMeans					5.57
	SVM					5.17
usergender		Female				-5.19
impgender			Female			-7.31
userstyle				Touch		4.34
impstyle					Touch	11.30
classifier:usergender	OutlierCount	Female				2.48
	Mahalanobis	Female				13.31
	KNN	Female				10.58
	MahalanobisKNN	Female				12.13
	KMeans	Female				11.02
	SVM	Female				2.56
usergender:impgender		Female	Female			3.57
userstyle:impstyle				Touch	Touch	4.15
$\sigma_{(user)}$						19.40
$\sigma_{(impostor)}$						13.84
σ_{ε}						19.01

Table 5.2: LMM for the results of the personal-traits study. The model equation is at the top. Terms for the classifier and the genders and typing styles of the genuine users and impostors were kept during model selection. The estimated effects of these factors and the per-user, per-impostor, and residual standard deviations are in the parameter-estimates table at the bottom.

fier is still estimated to have the lowest error rate under all conditions. The estimates in the classifier section of the table are all positive, so the other classifiers are expected to have higher miss rates for male genuine users. The estimates in the classifier/user-gender section are also all positive, so the other classifiers are also expected to have higher miss rates for female genuine users.

Based on the negative user-gender and impostor-gender estimates, we would expect that miss rates are somewhat lower for female users and impostors, but the positive user-gender/impostor-gender interaction estimate somewhat offsets those reductions when both user and impostor are female. Note, however, that all these effects are fairly small, and we postpone drawing conclusions about the effect of gender until we test the hypothesis formally.

One of the biggest effects in the table is the impostor's typing style (11.30), indicating a large increase in the miss rate when a touch-typing impostor is substituted for a non-touch-typist. For the Scaled Manhattan classifier, substituting a touch-typing impostor is estimated to cause the miss rate to increase from 8.9% to 21.4%. Substituting a touch-typing user for a non-touch-typing user is estimated to increase the miss rate to 13.1%. When both the user and impostor are touch typists, the interaction effect is also positive; for the Scaled Manhattan classifier, with touch-typing users and impostors estimated miss rate increases to 33.2%. Since the user and impostor typing-style effects are independent of the classifier, the estimated increase for the other classifiers would be similar.

Finally, note that the user, impostor, and residual standard deviations are quite high, as they were in the model from the last chapter. One might have expected the standard deviations to decrease since we have included personal traits in the model that help to explain some of the per-user and per-impostor effects. Looking at these numbers, the per-user standard deviation has increased from the previous chapter, the per-impostor standard deviation has decreased, and the residual remains about the same. We should note that only seven of the ten classifiers from the last chapter are evaluated in this chapter, so it is not wholly proper to compare the estimates of the models across these chapters. Nevertheless, the overall variance—calculated as the sum of the per-user, per-impostor, and residual variance—has decreased somewhat with the new model. This decrease suggests that user and impostor personal traits such as touch typing do offer a partial explanation for why some users and impostors have much higher miss rates than others.

5.5.3 Statistical hypothesis testing

With so many factors and interactions, there are literally thousands of possible tests we could perform. Consequently, unlike in the previous chapter, where we compared every pair of classifiers during hypothesis testing, here we must be more selective. Based on our observations and the analysis so far, we believe that the most important effects to test concern (1) the effect of touch-typing users and impostors, and (2) whether the classifiers/user-gender term in the model means that some classifiers really work better for male typists than for female typists. The first of these topics concerns the possibility that the effectiveness of keystroke-dynamics is greatly reduced by touch-typing impostors. The second topic concerns whether we can improve keystroke-dynamics by selecting the best classifier based on a typist's personal traits.

Pair #1		Pair #2					
user style	imp. style	user style	imp. style	effect	stderr	t.stat	p.value
touch	other	other	other	4.340	5.513	0.787	0.8794
other	touch	other	other	11.303	3.943	2.867	0.0212
touch	touch	other	other	19.790	6.764	2.926	0.0176
touch	other	other	touch	-6.963	6.763	-1.030	0.7491
touch	touch	touch	other	15.449	3.938	3.923	0.0006
touch	touch	other	touch	8.487	5.510	1.540	0.4170

Table 5.3: Hypothesis tests comparing miss rates for touch-typists and others. Each row compares one pairing of users' and impostors' typing skills against another pairing. Highly significant positive effects (corresponding to increased miss rates) are found when we compare pairings in which the impostor is a touch typist to those where the impostor is not.

Regarding the touch-typing effects, we identified 4 relevant terms in the model: the baseline, the user typing-style effect, the impostor typing-style effect, and the user/impostor typing-style interaction effect. To investigate whether the user's and impostor's typing styles significantly affect miss rates, we tested for differences among all 6 combinations of these factors. Each combination corresponds to testing whether the miss rate is the same for one pair of users and impostors and another pair of users and impostors. There are 4 ways to pair a touch-typing/non-touch-typing user and a touch-typing/non-touch-typing impostor (2×2), and there are 6 ways to choose two pairs for comparison out of 4 pairs.

Table 5.3 lists all possible comparisons among pairs and presents the results of the hypothesis tests: the estimated effect, the standard error, the test statistic, and the p-value (adjusted for multiple testing as explained in Chapter 3). Significant effects are those with p-values below .05, and we find three significant effects. The significant effects correspond to the 2nd, 3rd, and 5th rows of the table. In each of these rows, we compare pairings with a touch-typing impostor to pairings with a non-touch-typing (other) impostor. In all three cases, we find a significant increase in the miss rate corresponding to the pairing with the touch-typing impostor. Based on this analysis, we conclude that touch-typing impostors are more difficult to detect than non-touch-typing impostors.

Regarding the question of whether the user's gender affects a classifier's error rate, we conducted a series of tests, one for each classifier. In each test, we compared the classifier effect for female users to the effect for male users. The results of these tests are in Table 5.4. None of the comparisons show a significant difference (at any reasonable level). Consequently, while having the classifier/user-gender term in the model was preferred by the model-selection criterion, the hypothesis test downplays its effect.

This result highlights the different goals and properties of model selection and hypoth-

	effect	stderr	t.stat	p.value
ScaledManhattan	-5.192	5.615	-0.925	0.8103
OutCount	-2.715	5.615	-0.483	0.9755
Mahalanobis	8.120	5.615	1.446	0.4766
KNN	5.386	5.615	0.959	0.7909
MahalanobisKNN	6.935	5.615	1.235	0.6164
KMeans	5.832	5.615	1.039	0.7434
SVM	-2.629	5.615	-0.468	0.9782

Table 5.4: Hypothesis tests comparing error rates across gender. Each row tests whether gender has an effect on one of the seven classifiers. None of the results are significant; there is little evidence that gender plays a substantial role in classifier accuracy.

esis testing. BIC included these terms in the model because that criterion aims for large-sample fidelity to the true model. In the context of model selection, factors and terms can be admitted to models more leniently than in the context of a significance test. In this case, while we keep the gender terms in the model, the lack of a significant effect on classifier error rate and the relatively small size of the effects leads us to consider this factor of little practical importance.

5.6 Validation

We validate the model by using it to make predictions about a secondary evaluation. In this case, the predictions from the model concern classifier miss rates for different combinations of users and impostors, characterized by traits such as age, gender, dominant hand, and typing style. In particular, since the model contains terms for the user and impostor gender and typing-style, we aim to predict error rates based in part on these traits. The validation proceeds in two stages. First, we check whether the average miss rates of the users in the secondary evaluation fall within the prediction intervals of the model. Then, we use QQ-plots to more rigorously test the various modeling assumptions.

We use the same data for the secondary evaluation as in the previous chapter. The demographic survey was administered to the 14 subjects in the secondary evaluation data set, and we characterized them according to age, gender, dominant hand, and typing style. The demographic breakdown of these subjects was presented in the second row of Table 5.1 on page 106. We match the evaluation results from the validation data set with the personal traits of the genuine-user subject and the impostor subject.

Recall that the model includes terms for the classifier, user age, user gender, impostor age, and impostor gender. For each combination of these factors, we use the model to predict the average miss rate and use the per-user standard deviation to place 95% prediction intervals around that average. In the secondary evaluation, we have at least two data points with which to test each of these prediction intervals.

The prediction intervals and the per-user miss rates are presented in Figures 5.4 and 5.5. The results are split across figures—with results for male impostors in Figure 5.4 and female impostors in 5.5—because of layout constraints. In each panel, the intervals represent the 95% prediction interval for the per-user miss rates of each classifier. The points represent the actual per-user miss rates calculated from the second evaluation data set.

Most of the points are within the prediction intervals. When points do exceed the intervals (e.g., the panel in Figure 5.5 for female non-touch-typing users and impostors), a correspondingly extreme point exists for all classifiers. Such sets of points actually correspond to miss rates for a single subject who, because classifier error rates are so highly correlated with one another, appears as multiple extreme points, once for each classifier. Because points by-and-large fall within the prediction intervals, we believe that the results validate the model.

In some panels, the points appear to be clustered in a region smaller than the full length of the predicted interval. In other words, the intervals might be too conservative, predicting that points will fall across a wider range than they actually do in the validation data set. We must be cautious not to read too much into a small number of points. In many panels, we only have two or three data points. When a small number of points cluster together by chance, we should not assume that a larger number of points would exhibit the same clustering. In fact, in panels with many per-user points (e.g., in Figure 5.4, with male touch-typing user and impostor subjects), the spread of the points roughly matches the intervals.

QQ-plots offer a more rigorous test of the model predictions and assumptions than the prediction intervals above. Per-user, per-impostor, and residual effects were calculated as described in Chapter 3. These effects were scaled by the corresponding standard-deviation estimate from the model, resulting in three sets of standardized effects. Figure 5.6 presents the QQ-plots comparing each set of standardized effects against a standard Normal distribution: per-user effects on the left, per-impostor effects in the middle; residuals on the right. The data are largely consistent with the standard Normal distribution. Overall, this validation step confirms that our model is making accurate predictions.

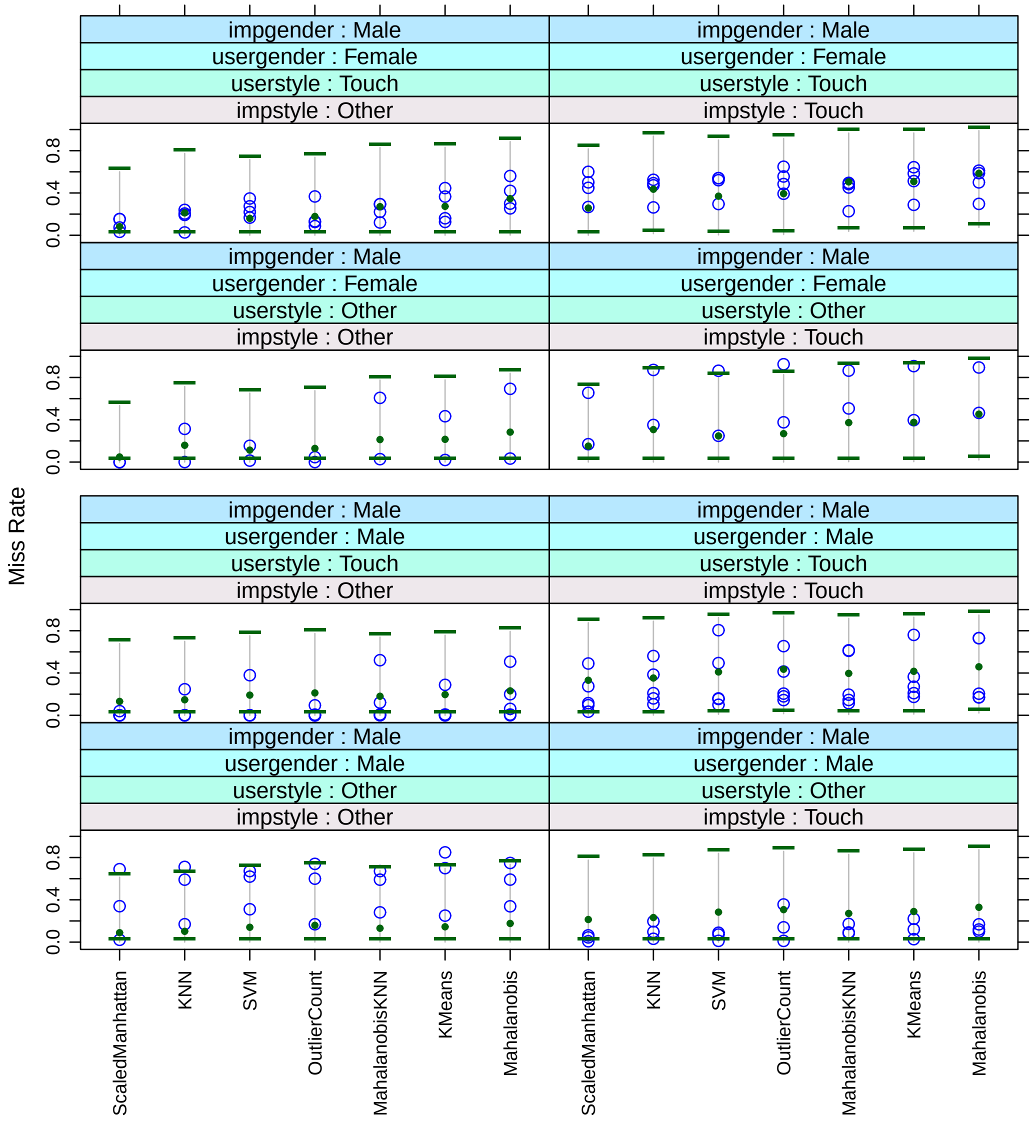


Figure 5.4: Model predictions and validation data for personal-traits study (part 1). 95% prediction intervals were estimated for the cases where the impostor is male. (Results when the impostor is female are in Figure 5.5.) In all but two panels, the data points—per-user average miss rates—fall within the predicted intervals.

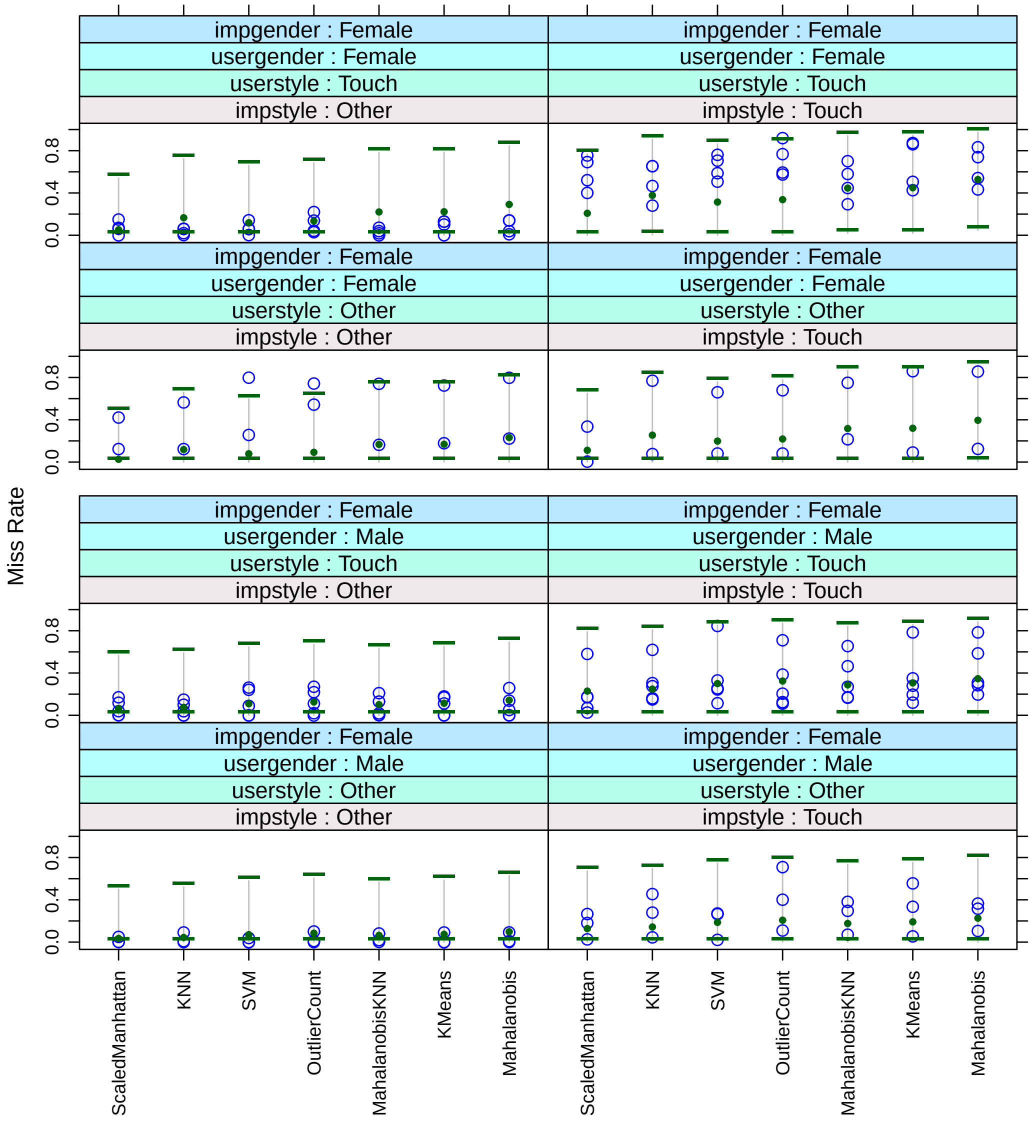
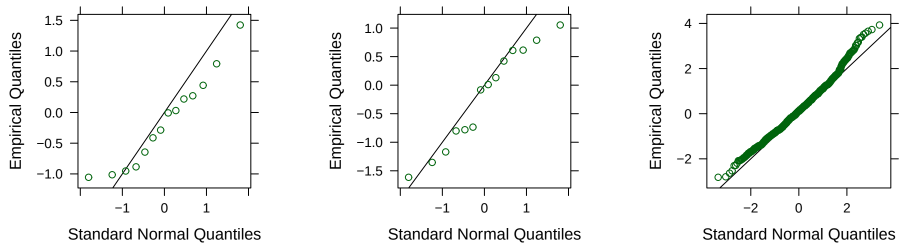


Figure 5.5: Model predictions and validation data for personal-traits study (part 2). 95% prediction intervals were estimated for the cases where the impostor is female. (Results when the impostor is male are in Figure 5.4.) In most cases, the data points fall within the predicted intervals.



(a): per-user (b): per-impostor (c): residual

Figure 5.6: QQ-plots for validating LMM predictions from the personal-traits study. Per-user effects (left), per-impostor effects (middle), and the residuals (right) are all shown to be largely consistent with the Normal distribution assumed by the model.

5.7 Discussion

This investigation discovered fixed-effect factors other the classifier that affect keystroke-dynamics error rates, namely the user and impostor typing styles. The effect was seen across all seven classifiers that we evaluated. In fact, swapping touch-typing users and impostors for non-touch-typing ones is estimated to have a greater effect on the miss rate than changing from any one of the seven classifiers to any other one. Based on the model, the biggest difference in classifier variance-stabilized miss rates is between Scaled Manhattan and Mahalanobis: 8.31 for male users and 13.31 for female users. In comparison, the biggest difference due to typing styles is 19.79, between non-touch-typist and touch-typist users and impostors. This finding further supports our claim that keystroke-dynamics error rates depend on many factors in addition to the classifier.

As noted at the beginning of the chapter, other studies have considered the effect of personal traits on keystroke-dynamics error rates. In particular, others have considered typing style or skill in various forms and their effects on error rates. Mészáros et al. (2007) provide a histogram showing that in their evaluation equal-error rates are lower for genuine-user subjects who spend more "hours in front of computer." It is unclear whether the effect is statistically significant, or whether the results contradict our own findings. Touch typists may not spend any more time on computers than other kinds of typists.

One study by Hosseinzadeh et al. (2006) compared false-alarm and miss rates across "*two finger* typers" and more expert typists. They observed that false-alarm and miss rates were higher for the two-finger typists. Insofar as their expert and two-finger groups correspond to our touch-typing and non-touch-typing groups, respectively, that result is in contrast to ours. However, they did not perform a statistical analysis to see whether there was evidence that the difference between the two groups generalized beyond the experiment. Further, and perhaps more importantly, Hosseinzadeh et al. (2006) allowed their impostor subjects to observe the genuine-user subjects' typing (e.g., either directly or by examining plots of their hold and down-down times). Their impostors watched the genuine-user subjects type their passwords. Their conjecture is that two-fingered typists, who are known to be such, are easier to imitate.

In similar work, Lin (1997) reported classifier errors by user and impostor typing skill. That study identified proficient and non-proficient typists. The proficiency criteria were based on an overall measure of words-per-minute in a typing task. Proficient typists scored at or above 90 words per minute. Just as we have, that work found a higher number of errors when users and impostors were both proficient. However, no statistical analysis was performed to support the observation, and the sample size was comparatively small.

We split typists into those who touch type and those who do not based on a survey. Subjects in the touch-typing group self identified as touch typists. Naturally, this procedure raises the question of whether these subjects really are touch typists who type without looking at the keyboard. In some cases, we were able to check our lab notebook to confirm that we observed the subjects in this group touch typing during data collection. More generally, because the question of typing style was assessed on a 5 point scale, participants were able to answer any value between 2–4 in the event that they were reluctant to admit to being hunt-and-peck typists. We split our groups between those who answered 1–4 and those who answered 5. To be incorrectly included in the touch-typing group, subjects would have had to circle the extreme value (5) in the survey. Consequently, we believe that subjects had little incentive to falsely claim touch-typing skill.

We should also provide a word of caution about interpreting our results concerning personal traits and classifier error rates. In our discussion, we have informally stated, for instance, that gender *may affect* miss rates. We adopt this expression because it is much more tortuous to say that gender *may be correlated with differences in the miss rate*. However, we cannot forget the oft-repeated mantra of statistics teachers: correlation is not causation. In this case, we recruited a convenience sample of subjects, some of whom were male and some female. Even if a gender effect survived statistical hypothesis testing (which it did not in this investigation), it would be incorrect to infer from our results that something intrinsic to gender makes one's typing more or less amenable to keystroke dynamics. An effectively infinite number of alternative explanations cannot be ruled out as hidden variables. Investigating human physiology through convenience samples is perilous, and we caution readers against drawing conclusions without a great deal of further research.

5.8 Summary

Prior work in and outside keystroke dynamics suggested that typists' personal traits—such as their age, gender, dominant hand, and typing style—may affect classifier error rates. Personal-trait effects would help to explain the large per-user and per-impostor effects on miss rates. To investigate these effects, we administered a demographic survey to subjects whose data were used to evaluate keystroke-dynamics classifiers. Based on the survey results, we used LMMs to analyze whether the personal traits of the user and impostor affected the classifier miss rates (for a fixed false-alarm rate).

From this second investigation, we draw the following conclusions:

1. *The Scaled Manhattan classifier continues to have the lowest estimated miss rate, regardless of the user and impostor age, gender, dominant hand, or typing style.* The previous investigation established Scaled Manhattan as the best overall classifier; the current evaluation finds that we cannot do better than Scaled Manhattan by choosing a classifier based on the user's age, gender, dominant hand, or typing style.
2. *When the impostor is a touch typist rather than another kind of typist, the Scaled Manhattan miss rate increases from* 13.1% *to* 33.2% *if the genuine user is also a touch typist, and from* 8.9% *to* 21.4% *if the genuine user is not.* The effects of user and impostor typing skill are independent of the classifier, so similar increases in the miss rate are expected for all classifiers, not just Scaled Manhattan. This result suggests a possible vulnerability in keystroke dynamics, since an impostor who wished to evade detection would do well to learn to touch type.
3. *Other than the user and impostor typing style, none of the other tested traits (i.e., age, gender, or dominant hand) were found to have a significant effect on classifier miss rates.* With this conclusion, we do not reject the possibility that these other traits might have effects; we only accept that such effects may be smaller and require more work to find.

These findings reveal a new factor that must be considered when explaining the behavior of keystroke-dynamics classifiers. They also expose a potential vulnerability that impostors might use to increase their chances of evading detection.

Returning to the thesis statement from Chapter 1, we claimed in part that LMMs offer better understanding of classifier behavior than current practices. In this investigation, we examined earlier efforts to understand whether user and impostor personal traits affected classifier behavior. We found the results inconclusive, in part because of small sample size and in part because of a lack of inferential statistics. Then, by merging our benchmark evaluation results (from Chapter 4) with demographic-survey results, and by analyzing the merged data set using LMMs, we discovered that typing skill strongly influences classifier error rates. We were also able to gather evidence that other traits such as age, gender, and dominant hand have less (if any) influence; ruling out factors as having a substantial effect is a contribution in its own right. Consequently, the investigation has furthered our understanding of keystroke-dynamics classifier behavior and the factors that affect it.

Chapter 6

Screening Keystroke-Dynamics Factors

Many factors have varied across prior classifier evaluations, and in this chapter we investigate what effect those factors have. Specifically, in addition to the classifier we consider the following factors: the typing task, the feature set, the amount of training, the updating strategy, and impostor familiarity with the typing task. Different researchers have made different choices with respect to these factors in earlier evaluations; if those choices do affect evaluation results effect, it would help explain why results vary wildly across evaluations. By investigating these factors, we might also discover potential ways to optimize performance (e.g. the best feature set and updating strategy) and potential vulnerabilities (e.g., impostor familiarity).

6.1 Background

One can easily list dozens of factors that might affect whether a classifier can distinguish two people's typing. Many of the papers we surveyed in Chapter 2 included a discussion of how the results might not generalize to other users, typing tasks, environments, and so on. With so many factors that *might* affect keystroke-dynamics error rates, how do we find those that do, and how do we decide which factors can be safely ignored? For keystroke dynamics to become a dependable computer-security technology, we must be able to explain what factors influence their error rates, and how.

Some earlier work has gone beyond listing the possible factors that might affect error rates. We have identified six factors that have been investigated in the literature by comparing error rates across different levels or values of the factor:

1. Classifier: Obviously, the classifier itself is a factor; some classifiers may have higher error rates than others. In an earlier survey, we found that about 54% of keystroke-dynamics studies compared multiple classifiers (Killourhy and Maxion, 2011). This is one factor whose effects are investigated fairly frequently.

2. Typing task: Even if we restrict our attention to login-time authentication, keystroke dynamics might be applied to a variety of tasks: different words, passwords, usernames, and even numeric codes. Bleha et al. (1990) had subjects type both their names and a fixed English phrase. Their findings that error rates were lowest with names establishes the typing task itself as a possibly influential factor.

3. Feature set: A variety of timing features, including hold times, keydown-keydown times, and keyup-keydown times have been tried. Different researchers use different combinations of these features in their evaluations. One study compared many combinations (Araújo et al., 2005); they recommended using all three at once for the lowest error rate.

4. Training amount: Some researchers have trained their classifiers with as few as 5 repetitions of a password, while others have used over 200. Increasing the number of training repetitions even from 5 to 10 can reduce error (Joyce and Gupta, 1990), so the amount of training is expected to be highly influential.

5. Updating strategy: Most research has focused on classifiers that train once on typing samples and then the profile is fixed. More recently, researchers have suggested that regularly updating the profile, retraining on the most recent typing samples, can improve error rates (Kang et al., 2007).

6. Impostor familiarity: An impostor is an intelligent adversary and will presumably try to evade detection. Some researchers have given their impostor subjects the opportunity to become familiar with a password by typing it themselves before the evaluation; they found that impostor familiarity raises miss rates (Lee and Cho, 2007).

Any of these six factors might explain different keystroke-dynamics error rates across evaluations. However, earlier work on the effects of these factors is inconclusive. As detailed in Chapter 2, evaluation results in keystroke dynamics are rarely analyzed with formal inferential statistics; when they are used, the inferential technique almost never takes into account possible interactions among the factors.

6.2 Aim and approach

Our aim in this investigation is to understand which of the six factors—classifier, typing task, feature set, training amount, updating, and impostor familiarity—affect the error rate of keystroke-dynamic classifiers. We also hope to identify which of these factors interact with each other so that future work can take such complexities into account.

As in the earlier investigations, we collect typing data and conduct an evaluation. The evaluation is conducted such that we systematically vary the six factors of interest across a sequence of evaluation runs. We analyze the results of the evaluation using LMMs to identify which factors and interactions between factors have a substantial effect on classifier error rates. Finally, we collect typing samples and conduct a secondary evaluation to validate the model. In a slight departure from our earlier analyses, the size of the evaluation-results data set necessitates splitting the analysis in two: (1) finding the best feature set, and (2) understanding the effects of the many other factors.

6.3 Experimental method

The experimental method is comprised of three steps: (1) select values of interest for each of the factors, (2) collect typing data with which to investigate those values of the factors, and (3) repeatedly run an evaluation, while systematically varying the factors over a sequence of runs.

6.3.1 Selecting factor values.

The factors of interest in this investigation—classifier, typing task, feature set, training amount, and impostor familiarity—can take many different values. For this study, we need to choose a subset of values to test.

1. Classifier: We selected three classifiers for the investigation: Scaled Manhattan, Mahalanobis k-NN, and SVM. Scaled Manhattan had the lowest miss rate among all the classifiers evaluated in Chapters 4 and 5. The SVM and Mahalanobis k-NN had the 3rd lowest and 5th lowest miss rates, respectively. These three were chosen because the typing profiles they learn seem markedly different (as illustrated in the contour plots of Figure 4.2 on page 75). By using them in this investigation, we hope to discover cases where their miss rates diverge, showing the relative strengths and weaknesses of each classifier.

2. Typing Task: In our earlier investigations, subjects typed a strong password, but login-time keystroke dynamics has also been applied to words, passphrases, names, user IDs, and numeric passcodes. We chose two additional typing tasks for comparison with the strong password (.tie5Roanl): a simple-to-type user ID (hester), and a numeric passcode (412 193 7761). Together, these Strong, Simple, and Numeric tasks enable us to understand how much effect the typing task has.

3. **Amount of training:** Prior researchers have trained login-time anomaly detectors with as few as 5 repetitions and as many as 200 or more repetitions. Our values were chosen to broadly map the contours of the range of possible values: 5, 25, 50, and 100 repetitions.

4. **Feature sets:** In the earlier investigations in this dissertation, we provided classifiers with hold and down-down, times (not up-down times). We also provided timing features for the Return key at the end of the password. In this investigation, we look each of these four feature sets as separate binary factors (sub-factors of the feature-set factor, if you will): (1) presence/absence of hold-time features; (2) presence/absence of down-down features; (3) presence/absence of up-down features; (4) presence/absence of Return key features. Four binary variables lead to 16 combinations, but 2 combinations are impossible (i.e., one of hold, down-down, or up-down timing features must be in the feature set).

5. **Updating strategy:** In the earlier evaluations in this work, the typing data for the designated genuine-user subject was split so that the first samples (e.g., 1–200) were used during a training phase to train a classifier, at which point the genuine-user subject's typing profile was fixed for the remainder of the evaluation. An updating classifier effectively has multiple training phases. After the initial training phase, the remaining typing samples must be presented to the classifier in batches (e.g., of 5 samples each). The samples in a batch would be scored and then used to retrain the classifier during a subsequent training phase. Then, the next batch would be presented to the newly retrained classifier. For this investigation, we chose to compare a *sliding window* updating strategy with *none* (i.e., no updating).

6. **Impostor familiarity:** While prior work has considered the issue of impostors trying to evade detection by becoming familiar with a password or watching the genuine-user type it, the concept has not been operationalized formally. For this work, we measure familiarity in terms of the number of practice repetitions an impostor used. We define two levels of familiarity: *none* and *high*. With no familiarity, the impostor test samples provided to the classifier are from impostors who never typed the password before. Specifically, they are the first 50 repetitions from an impostor subject. One might argue that by the 50th repetition, the impostor is somewhat familiar with the password. Regardless, we use 50 samples to have enough data to accurately estimate

the miss rate for the impostor. With high familiarity, the samples are taken after the impostor subject has practiced typing the password 150 times. Specifically, they are the 151st to 200th repetitions typed by the impostor (i.e., 100 additional repetitions after the first 50).

To test whether a factor has an effect, one must identify the range of values that the factor can take in an investigation. For each of the factors under study in this investigation, we have established two or more relevant values that will be studied and compared. While other choices of values are certainly possible, the range of values chosen are realistic, and will help us understand keystroke-dynamics under comparatively practical conditions.

6.3.2 Typing-data collection

We begin with the extant Strong data from Chapter 4, and we collect new typing data for the Simple and Numeric typing tasks. In each task, as in the Strong task, subjects were prompted to repeatedly type a given sequence of characters in multiple data-collection sessions. The same data-collection apparatus was used to collect data for all three tasks. Previously described in Chapter 4, the apparatus consists of a laptop with an external keyboard running a software application that prompts subjects to type key sequences and monitors their compliance. In a given session, the subject must type the sequence correctly 50 times. If any errors in the sequence are detected, the subject is prompted to retype the sequence. The external keyboard is connected to an external reference timer calibrated to generate timestamps with an accuracy of ± 200 microseconds. Whenever a subject presses or releases a key, a timestamp for the keystroke event is logged.

We chose to have subjects perform the following three typing tasks (with the already-described Strong task included for completeness):

Strong: To make a password that is representative of typical, strong passwords, we employed a publicly available password generator (PC Tools, 2008) and password-strength checker (Microsoft, 2008). We generated a 10-character password containing letters, numbers, and punctuation, and then modified it slightly, interchanging some punctuation and casing to better conform to the general perception of a strong password. The result of this procedure was the password: .tie5Roanl.

Simple: To construct a simple keystroke sequence that is representative of user IDs or English words, we considered the common letters and digraphs in English words. Letter and digraph frequency charts are compiled from corpora of English texts; a common usage is in cryptanalysis. Menezes et al. (1997) presents a frequency table for the top 15 digraphs in a standard English corpus. In consultation with a dictionary, we build two six-letter words comprised solely of these digraphs: hester and rested. Pilot testing showed that both could be typed quite quickly by several typists, and the former seemed like a suitably representative user ID. For our simple typing task, subjects typed the user ID: hester.

Numeric: To construct a numeric passcode, we chose a mock phone number. Such a number could be used for authentication on a mobile phone; it could also act as an access control code typed on a numeric keypad. A US phone number is composed of an area code (first 3 digits), and NXX code (next 3 digits), and line code (last 4 digits). The 412 area-code would be familiar to our subjects so we chose it. To avoid publicizing a real phone number, we used an invalid NXX code. Ours started with a 1 since real NXX codes cannot start with a 0 or 1. The remainder of the code was chosen to ensure a range of finger movements (Maxion and Killourhy, 2010). For realism in entering an access-control code, and for a highly controlled experiment, we instructed subjects to use the numeric keypad on the keyboard and to type the number using only their right index finger. The following number was chosen for the Numeric typing task: 412 193 7761 (spacing included for readability).

In our first investigation with the Strong typing task, 65 subjects typed 8 sessions with 50 repetitions in each session. Of the 65 subjects, 51 were used in the evaluation pool and the remaining 14 for the validation pool. For the Simple typing task, we recruited 38 subjects to type 4 sessions with 50 repetitions in each session (i.e., 200 repetitions for 38 subjects). For the Numeric typing task, we recruited 40 subjects to type 4 repetitions with 50 repetitions in each session (i.e., 200 repetitions for 40 subjects). Note that, because we only collected 4 sessions of data for the Simple and Numeric tasks, for this investigation, all of our analyses will concern 200 repetitions of each task. In particular, even though we have 400 repetitions for each subject who typed the Strong password, we only use the first 200 repetitions in evaluating classifiers in this chapter. For each subject who participated, we extracted hold times, down-down times, and up-down times for all the keystrokes including the Return key.

Across the Strong, Simple, and Numeric tasks, 26 subjects performed all three tasks, and the remaining subjects performed only one or two of the tasks. We selected the data from the 26 subjects for inclusion in the primary evaluation pool because having the same subjects perform all three tasks will make the statistical analysis of the typing-task effect more powerful. The data from the remaining subjects will be used in the secondary evaluation to validate the model. Table 6.1 compiles the number of subjects per typing task and

	Strong	Simple	Numeric
Evaluation Pool	26	26	26
Validation Pool	14	12	14

Table 6.1: Summary of the number of subjects who completed each typing task. In the Strong, Simple, and Numeric tasks, subjects typed .tie5Roanl, hester, and 412 193 7761 respectively. The 26 subjects who typed all three were chosen for the primary evaluation pool (i.e., those whose evaluation data are used to build the LMM). Those who did not type all three were reserved for the secondary validation pool (i.e., those whose evaluation data are used to validate the model).

how many subjects are used in the primary evaluation vs. the secondary evaluation. While we have data from 65 subjects for the Strong task, we chose 14 for the validation set. The particular subset of subjects were selected arbitrarily, but the size of the subset (i.e., 14) was chosen so that the relative sizes of the three typing tasks' validation pools would be approximately the same.

6.3.3 Evaluation procedure

We have described how values were chosen for each of the factors under study, and we collected data with which to study their effects. Our evaluation is *full factorial* in the six factors of interest (Box et al., 2005). Every combination of the factor values is run in the course of the experiment. We have 3 classifiers, 3 typing tasks, 14 feature sets, 4 amounts of training data, 2 updating strategies, and 2 levels of impostor familiarity, leading to 2016 combinations ($3 \times 3 \times 14 \times 4 \times 2 \times 2$).

For each of the 2016 combinations of factors listed above, we conduct a series of evaluation trials. In each trial, one of the subjects is designated as the genuine-user subject, and the remainder as impostors. A trial can be thought of as a procedure that takes inputs, performs a series of steps, and produces outputs. In this case, the inputs are values for each of the 6 factors—classifier, typing task, feature set, amount of training, updating strategy, and impostor familiarity—as well as the designated genuine-user subject.

To accommodate the different combinations of factor values, the evaluation procedure is fairly complicated. We present it here as an enumerated set of steps, including sub-steps and branches when necessary (like a computer program).

1. Depending on the typing task, the appropriate evaluation data set is loaded (i.e., Strong, Simple, or Numeric). The evaluation data set contains 200 repetitions of 26 subjects typing the key sequence for the task. Each repetition contains hold times, down-down times, and up-down times.

2. Depending on the feature set, unnecessary typing times are removed. For instance, when the feature-set value is hold and up-down times without the Return key, then all down-down times are discarded along with timing features involving Return.
3. Depending on the amount of training and the genuine-user subject, a training-data set is constructed. The training data are formed from the first of that subject's samples, up to the given amount of training (i.e., 5, 25, 50, or 100 repetitions).
4. The classifier—Scaled Manhattan, Mahalanobis k-NN, or SVM—is trained using the training sample.
5. The next steps depend on whether the updating strategy is *none* or *sliding window*. If the updating strategy is *none*, the following steps are performed:
   (a) The 100 samples following the genuine-user subject's training data are presented to the classifier and scored. These anomaly scores are used to tune the classifier to have a 5% false-alarm rate.
   (b) The 25 subjects not designated as the genuine-user subject are designated as impostor subjects. If the impostor-familiarity level is *none*, then the first 50 samples from each of the impostor subjects are presented to the tuned classifier. If the impostor-familiarity level is *high*, then the last 50 samples from each impostor subject (i.e., repetitions 151–200) are presented to the tuned classifier. In either case, we record whether each sample was detected or missed.

   If the updating strategy is *sliding window*, the following steps are performed.
   (a) The 5 samples following the genuine-user subject's training data are presented to the classifier and scored. The classifier is saved (to be used in the following steps), and a new training data set is created by removing the first 5 samples from the earlier training set and replacing them with the 5 samples just scored. The classifier is retrained using the new training set.
   (b) The previous step is repeated until 100 samples have been scored, and 20 classifiers have been trained and saved. Each of these 20 classifiers were trained on different (but overlapping) sets of genuine-user samples, and each classifier is used to score the five samples after its training set.
   (c) A single anomaly-score threshold is found for all 20 trained classifiers by pooling the 100 anomaly scores and tuning the classifiers to have an overall false-alarm rate of 5%. Specifically, we pool the scores produced by the 20 classifiers, and we find the 95th percentile of the scores.
   (d) The 25 subjects not designated as the genuine-user subject are designated as impostor subjects. If the impostor-familiarity level is *none*, then the first 50

samples from each of the impostor subjects are presented to each of the 20 tuned classifiers. For each sample and each classifier, we record whether the sample was detected or missed. If the impostor-familiarity level is *high*, the last 50 samples from each impostor subject are presented to each of the 20 tuned classifiers, and we record whether they were detected or missed by each classifier.

The portion of this procedure concerning *sliding window* updating is somewhat complex, and a few aspects of it should be explained. We make the simplifying assumption that the classifier will only retrain on the genuine user's data. Impostor poisoning of the training is not considered in this work. We classify each repetition of impostor test data multiple times, once with each of the 20 trained classifiers. A sample's anomaly score may change whenever the classifier is retrained, and so a sample may be detected by some instances of the classifier and not others. By testing each sample with each classifier and then calculating the miss rate using all the results, we effectively aggregate over the variability due to updating, and we find the average miss rate.

For each of the 2016 combinations, and within each combination for each pair of genuine-user and impostor subjects, we calculate the proportion of samples that were missed. There are 1,310,400 such miss rates for our analysis (2016 combinations $\times$ 26 genuine-user subjects $\times$ 25 impostor subjects).

6.4 Empirical results

While the full analysis will consider the role that per-user and per-impostor effects have on classifier miss rates, our initial exploration of the results will necessarily focus on averages. Specifically, for each of the 2016 combinations of factor values, we calculate the average miss rate over all genuine-user and impostor subjects. Note that if we did not take the averages and instead tried to understand this experiment through latticed boxplots of per-user/impostor miss rates as in the previous investigations, we would have hundreds of panels no matter which factors we chose for the boxplots. That overwhelming number is further reason why inferential statistics are needed to understand environments with many factors and many potential interactions. Our exploration will simply try to develop an intuition for each factor's effect by averaging out the effects of other factors.

Typing task. Figure 6.1 shows the average miss rates for each classifier on each typing task. Within each panel, the classifier's results are shown for increasing amounts of training

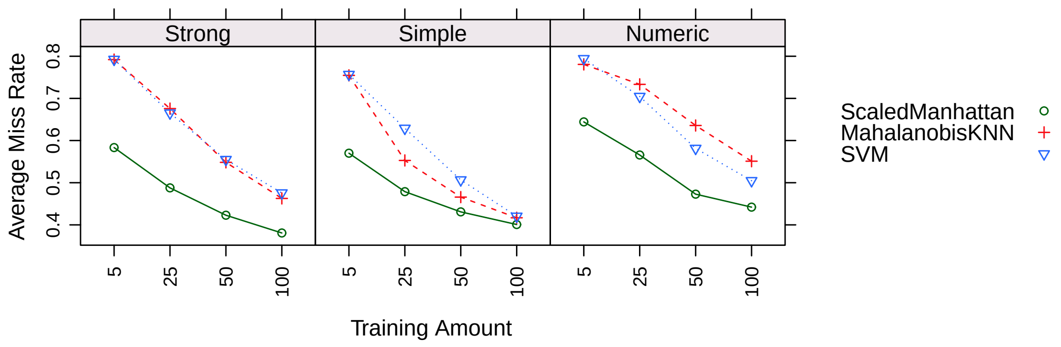


Figure 6.1: Average miss rates for different typing tasks. Each panel presents miss rates for the three classifiers for different amounts of training. The results for each typing task are presented in separate panels. Training appears to improves performance, but the amount of improvement may depend on the classifier and the task. Miss rates are lowest for the Strong task.

data. The miss rates in the figure are averaged over the other factors (feature set and impostor familiarity). Each panel shows clear improvement resulting from additional training. Reductions in the miss rate are observed all the way up to 100 repetitions, though in some cases with diminishing returns.

The Scaled Manhattan classifier performs better than the other classifiers. It is unclear whether the Mahalanobis k-NN and the SVM perform the same under all circumstances or whether there are slight differences. In the Simple task, the Mahalanobis k-NN appears to respond somewhat more quickly to increased training. Not only do the typing task and amount of training affect classifier performance, but that there might be an interaction between the two. Going from 5 to 25 typing samples reduces miss rates in the Simple task, more than in the Numeric task, for instance.

Feature sets. Figure 6.2 presents average miss rates for each classifier with each combination of timing features. Above each panel is a code containing four pluses and minuses (+/-), indicating which features are present or absent for the results in that panel. For instance, the `-:-:+:+` code in the top left panel means that hold and down-down times are not included, but up-down times and Return key features are included.

Note that the column of panels on the right and the column on the left are very similar. The difference between these two columns is the presence/absence of Return key features. There appears to be little effect to having the Return as part of the password when extracting timing features. The bottom six panels are also quite similar. In these six panels,

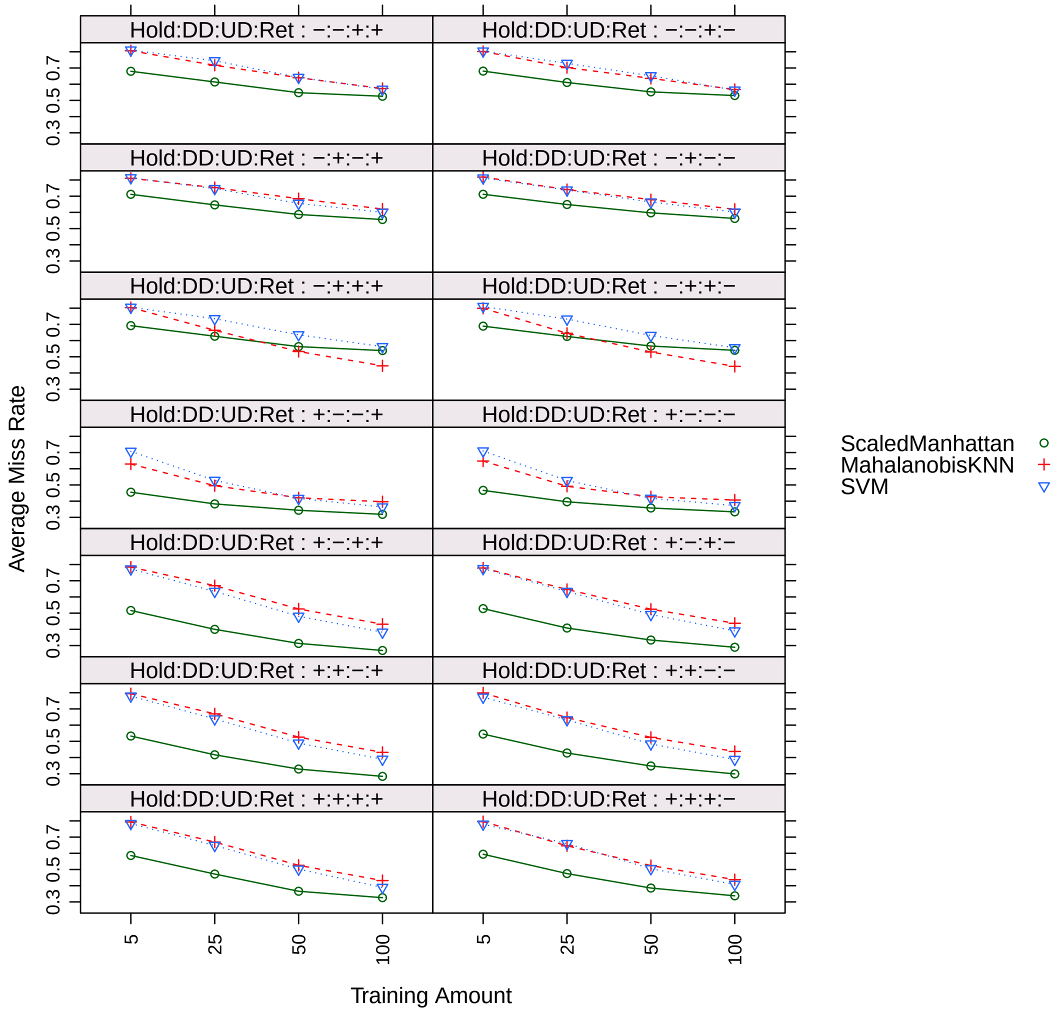


Figure 6.2: Average miss rates for different timing features. Each of the 14 panels presents classifier miss rates for a different combination of timing features. There is little difference between panels on the left and panels on the right (with/without Return features). The panels with hold-time features and either digraph feature (bottom six) appear better than those without hold-time features (top six) or with just the hold-time feature (middle two).

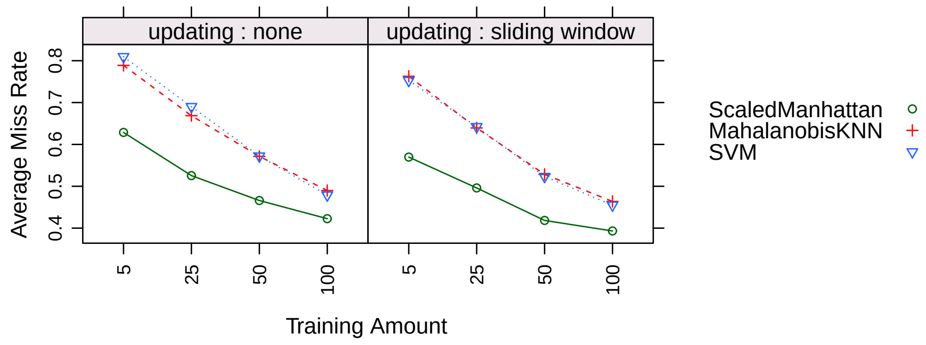


Figure 6.3: Average miss rates for different updating strategies. The panel on the left shows average miss rates for classifiers that train only once (i.e., an updating strategy of *none*). The panel on the right shows average miss rates for classifiers that repeatedly update the typing profile (i.e., a *sliding-window* updating strategy). Across classifiers and training amounts, sliding-window average miss rates are lower.

the hold-time features are paired with either down-down or up-down features. All such combinations appear roughly equivalent.

In the top six panels, the miss rates are higher, at least for high amounts of training (e.g., 50–100 repetitions). In these panels, the hold-time features are absent. Contrasting the top six panels with the bottom six, hold times appear to be important features for reducing miss rates. In the middle two panels, where only hold-time features are used, the classifiers perform better than when only down-down or up-down times are used. For the Mahalanobis k-NN and SVM classifiers, these are among the best results. For the Scaled Manhattan classifier, they are not as good as when digraph timings are used as well.

Updating strategy. Figure 6.3 presents the average miss rates for each classifier with and without a sliding-window updating strategy. The panel on the left shows miss rates when classifiers do not use updating, and the panel on the right shows miss rates with a sliding-window updating strategy. Every line representing classifier miss rates is lower in the sliding-window panel, suggesting that updating improves every classifier, no matter how much training data was used in the initial training phase.

In the sliding-window panel, the Scaled Manhattan classifier's miss rate starts to level off at 50 samples of training. Perhaps with updating, this classifier can be trained just as effectively using only 50 samples of training as using 100 samples. Such a result would be encouraging since 100 samples of training would be onerous for many applications of keystroke dynamics.

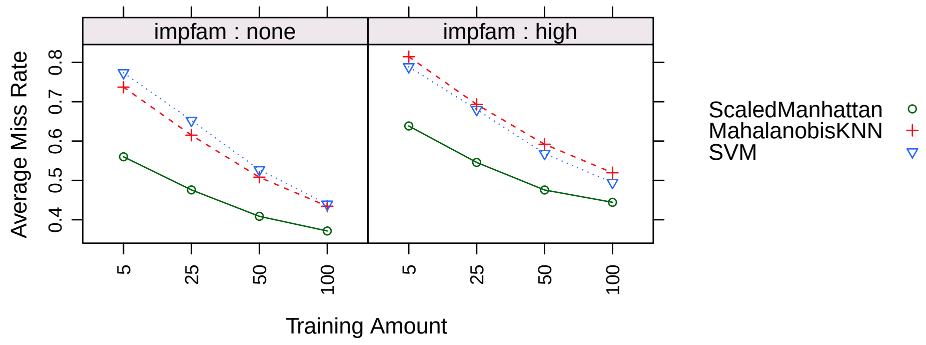


Figure 6.4: Average miss rates for different impostor-familiarity levels. The panel on the left shows average miss rates for all classifiers when the impostor has not practiced the typing task and become familiar with it. The panel on the right shows results for impostors who practiced 150 repetitions of the typing task and became familiar. The familiar-impostor miss rates are noticeably higher.

Impostor familiarity. Figure 6.4 presents average miss rates when impostors are unfamiliar with the typing task to those when impostors have practiced to become familiar with the task. The miss rates do appear to be higher—for all classifiers and all amounts of training—when the impostors are familiar with the task. This observation suggests that impostors might increase their chances of evading detection by practicing a compromised password a few hundred times before trying to use it to access a system.

Overall observations. Our observations suggest that many of the factors studied in this investigation will turn out to have a significant effect on miss rates. Of course, formal statistical analysis and validation of the model are necessary. We now proceed with that analysis. Because of the complexity of the data and the number of factors and interactions possible, we conduct two analyses. The first focuses on the feature sets, and the second focuses on the amount of training, updating, and impostor familiarity. In each case, we examine the effect across classifiers and typing tasks.

6.5 Statistical analysis #1: Feature sets

For the first analysis, we restrict our attention to a subset of evaluation results. Specifically, we analyze only the miss rates for 100 samples of training data, with no updating, and with unfamiliar impostors. With this subset of data, we investigate the effect of different feature-set combinations on classifier miss rates. As in previous analyses, we begin with

Minimal Model Equation:

$$\begin{aligned}
\text{VST}(\textit{miss rate})_{ijk} &= \mu + (\textit{user})_i + (\textit{impostor})_j + \varepsilon_k \\
(\textit{user})_i &\sim N(0, \sigma^2_{(\textit{user})}) \\
(\textit{impostor})_j &\sim N(0, \sigma^2_{(\textit{impostor})}) \\
\varepsilon_k &\sim N(0, \sigma^2_{\varepsilon})
\end{aligned}$$

Maximal Model Equation:

$$\begin{aligned}
\text{VST}(\textit{miss rate})_{ijklmnopq} &= \mu + (\textit{Classifier})_i \times (\textit{Typing Task})_j \\
&\quad \times ((\textit{Hold})_k + (\textit{Down-Down})_l + (\textit{Up-Down})_m)^2 \\
&\quad \times (\textit{Ret})_n + (\textit{user})_o + (\textit{impostor})_p + \varepsilon_q \\
(\textit{user})_o &\sim N(0, \sigma^2_{(\textit{user})}) \\
(\textit{impostor})_p &\sim N(0, \sigma^2_{(\textit{impostor})}) \\
\varepsilon_q &\sim N(0, \sigma^2_{\varepsilon})
\end{aligned}$$

Figure 6.5: Equations for model selection in the feature-set analysis. The minimal equation includes only the random effects. The maximal equation includes fixed effects for the classifier, the typing task, and all combinations of feature sets except the combination of hold, down-down, and up-down effects. The $(A + B + C)^2$ notation indicates all pairwise interactions between factors A, B, and C, but no three-way interaction.

model selection and parameter estimation followed by hypothesis testing.

6.5.1 Model selection

The minimal and maximal models for model selection are presented in Figure 6.5. The minimal equation only contains the user and impostor random effects, which we include in all models because of the structure of the experiment. The maximal model includes fixed effects for the classifier, the typing task, and the various combinations of timing features: hold times, down-down times, up-down times, and Return features. We use stepwise model selection to search the family of models for the model equation with the lowest BIC score.

Note that we structure the data and the model so that the feature-set terms represent the effect of *removing* the corresponding timing features from the baseline model (e.g., $(\textit{Hold})_k$ represents removing all hold times). The baseline model itself contains hold times, down-down times, up-down times, and Return features. This choice of baseline may seem contrary to intuition, but with it, we avoid a problem. Specifically, if features were added to the baseline rather than removed, the baseline would contain no features: no hold, down-down, or up-down times. Because evaluating classifiers using no features is impossible, we have no data for that case; building an LMM with such a baseline would fail. For the

Selected Model Equation (Feature-Set Analysis):

$$\begin{aligned}
VST(\mathit{miss\ rate})_{ijklmnop} &= \mu + (\mathit{Classifier})_i + (\mathit{Typing\ Task})_j + (\mathit{Hold})_k + (\mathit{Down\text{-}Down})_l \\
&\quad + (\mathit{Up\text{-}Down})_m + (\mathit{Classifier} : \mathit{Typing\ Task})_{ij} + \\
&\quad + (\mathit{Hold} : \mathit{Down\text{-}Down})_{kl} + (\mathit{Hold} : \mathit{Up\text{-}Down})_{km} \\
&\quad + (\mathit{Down\text{-}Down} : \mathit{Up\text{-}Down})_{lm} + (\mathit{Classifier} : \mathit{Hold})_{ik} + \\
&\quad + (\mathit{Classifier} : \mathit{Down\text{-}Down})_{il} + (\mathit{Classifier} : \mathit{Up\text{-}Down})_{im} \\
&\quad + (\mathit{Typing\ Task} : \mathit{Hold})_{jk} + (\mathit{Typing\ Task} : \mathit{Down\text{-}Down})_{jl} \\
&\quad + (\mathit{Typing\ Task} : \mathit{Up\text{-}Down})_{jm} \\
&\quad + (\mathit{Classifier} : \mathit{Hold} : \mathit{Down\text{-}Down})_{ikl} \\
&\quad + (\mathit{Classifier} : \mathit{Hold} : \mathit{Up\text{-}Down})_{ikm} \\
&\quad + (\mathit{Classifier} : \mathit{Down\text{-}Down} : \mathit{Up\text{-}Down})_{ilm} \\
&\quad + (\mathit{Typing\ Task} : \mathit{Hold} : \mathit{Down\text{-}Down})_{jkl} \\
&\quad + (\mathit{Typing\ Task} : \mathit{Hold} : \mathit{Up\text{-}Down})_{jkm} \\
&\quad + (\mathit{Typing\ Task} : \mathit{Down\text{-}Down} : \mathit{Up\text{-}Down})_{jlm} \\
&\quad + (\mathit{Classifier} : \mathit{Typing\ Task} : \mathit{Hold})_{ijk} \\
&\quad + (\mathit{user})_n + (\mathit{impostor})_o + \varepsilon_p \\
(\mathit{user})_n &\sim N(0, \sigma^2_{(\mathit{user})}) \\
(\mathit{impostor})_o &\sim N(0, \sigma^2_{(\mathit{impostor})}) \\
\varepsilon_p &\sim N(0, \sigma^2_{\varepsilon})
\end{aligned}$$

Table 6.2: LMM model equation for the results of the feature-set analysis. Based on the model-selection criteria, three-way interactions between the classifier, typing task, and feature sets are supported by the evaluation results.

same reason, we also do not consider a 3-way interaction between hold times, down-down times, and up-down times, since that interaction would correspond to the removal of all three feature sets.

6.5.2 Parameter estimation

Tables 6.2 and 6.3 present the model equation and parameter-estimation table respectively. The LMM has been split across the two tables because of its size. The model equation in Table 6.2 is quite complex, involving two- and three-way interactions among the classifier, typing task, and feature sets. For instance, the $(\mathit{Classifier} : \mathit{Typing\ Task} : \mathit{Hold})_{ijk}$ term is present, meaning that the effect of removing hold-time features on the miss rate depends on the particular combination of classifier (e.g., Scaled Manhattan, Mahalanobis k-NN, SVM) and typing task (e.g., Strong, Simple, Numeric).

Perhaps the most interesting aspect of the model equation is what is missing from it. The Return-key factor is not part of the selected model. Any term that is not in the model equation effectively has zero effect on miss rates. By its absence, we can infer that the presence or absence of the Return timing features have negligible effect on any of the three classifiers.

The actual estimates in Table 6.3 show a baseline estimate of 30.21, corresponding to a

Parameter Estimates (Feature-Set Analysis):

Parameters	classifier	typingtask	Hold	DD	UD	estimate
(μ) baseline	ScaledManhattan	Strong	+	+	+	30.21
classifier	MahalanobisKNN					12.03
	SVM					9.44
typingtask		Simple				4.10
		Numeric				2.16
Hold			-			21.48
DD				-		-5.35
UD					-	-5.16
classifier:typingtask	MahalanobisKNN	Simple				-7.66
	SVM	Simple				-3.39
	MahalanobisKNN	Numeric				-3.30
	SVM	Numeric				-4.14
Hold:DD			-	-		6.92
Hold:UD			-		-	7.36
DD:UD				-	-	7.01
classifier:Hold	MahalanobisKNN		-			-19.11
	SVM		-			-3.76
classifier:DD	MahalanobisKNN			-		3.97
	SVM			-		3.63
classifier:UD	MahalanobisKNN				-	3.18
	SVM				-	3.18
typingtask:Hold		Simple	-			-5.77
		Numeric	-			-8.89
typingtask:DD		Simple		-		1.33
		Numeric		-		2.82
typingtask:UD		Simple			-	3.25
		Numeric			-	2.69
classifier:Hold:DD	MahalanobisKNN		-	-		5.80
	SVM		-	-		-1.98
classifier:Hold:UD	MahalanobisKNN		-		-	8.23
	SVM		-		-	-1.19
classifier:DD:UD	MahalanobisKNN			-	-	-8.18
	SVM			-	-	-8.68
typingtask:Hold:DD		Simple	-	-		-5.76
		Numeric	-	-		-4.62
typingtask:Hold:UD		Simple	-		-	-1.72
		Numeric	-		-	-7.48
typingtask:DD:UD		Simple		-	-	1.21
		Numeric		-	-	8.22
classifier:typingtask:Hold	MahalanobisKNN	Simple	-			3.07
	SVM	Simple	-			-7.03
	MahalanobisKNN	Numeric	-			6.55
	SVM	Numeric	-			2.37
$\sigma_{(user)}$						12.78
$\sigma_{(impostor)}$						9.20
σ_{ε}						23.35

Table 6.3: LMM parameter-estimate table for the feature-set analysis. In the baseline, all timing features are present. The adjustments represent the effect of removing the corresponding features. Many of the largest effects involve hold times, indicating that hold times are important for achieving low miss rates.

miss rate of 20.9%. The other fixed-effects estimates are adjustments to this baseline miss rate. The largest such estimate is for the hold-time factor: 21.48. Based on this estimate, we would expect the Scaled Manhattan miss rate to increase to 52.7% if hold-time features are *not* included as timing features.

If we look at the typing-task/hold rows of the table, we see negative adjustments for removing hold times from the Simple and Numeric typing tasks rather than the Strong task. Removing hold times increases the miss rate *less* for Simple and Numeric than for Strong. Since these typing-task/hold-time effects (e.g., -5.77 and -8.89) are much smaller (in absolute terms) than the hold-time main effect (21.48), the hold time features remain important for all three typing tasks.

If we look instead at the classifier/typing-task rows of the table, we see a small negative number for the SVM/hold-time effect (-3.76) and the second largest adjustment (in absolute terms) in the table for the Mahalanobis k-NN effect (-19.11). These estimates suggest that hold times are as important to the low miss rate of the SVM classifier as for the Scaled Manhattan classifier, but not very important for the Mahalanobis k-NN classifier. Such a result would make sense since hold times are a linear combination of the down-down and up-down times. By the nature of the Mahalanobis distance, linearly dependent features are redundant, so the classifier can operate just as well with only down-down and up-down times.

The estimated per-user, per-impostor, and residual standard deviations are quite large compared to the main effects (i.e., 12.78, 9.20, and 23.35). Even after we have taken into account the effects of different classifiers, typing tasks, and feature sets, there remains a good deal of variance. The residual variation must be explained by factors yet to be identified.

6.5.3 Statistical hypothesis testing

The key question in this statistical analysis is which combinations of features produce the lowest miss rates. The presence of classifier/feature-set and typing-task/feature-set effects means that the same feature set may not be best for all combinations of classifier and task. Because we would like to use features that are generally useful across tasks and for multiple classifiers, we ask which feature set is best when the results are averaged across classifiers and typing tasks.

Constructing a contrast matrix for this set of tests is tricky. There are 7 combinations of features: three atomic feature sets (i.e., hold, down-down, and up-down times), all three pairs of feature sets, and the triple containing all three feature sets ($3 + 3 + 1$). For each feature set, we can estimate the average miss rate across all classifiers and typing tasks by appropriately weighting the estimates in the parameter-estimate table. For instance, for the hold-time feature set, we give a weight of 1 to the baseline, the down-down removal, the up-down removal, and the down-down/up-down interaction (i.e., so the only features left are hold times). We assign weights of $1/3$ to the classifier main effects, and classifier interactions involving the removal of down-down or up-down features. This fractional weight effectively averages the estimate over the three classifiers. Likewise, we assign weights of $1/3$ to the typing-task main effects and interactions involving the removal of down-down and up-down features. Interactions involving both the classifier and the typing task are assigned weights of $1/6$, averaging over the 6 combinations of classifier and typing task. The remaining effects are assigned a weight of zero.

Having constructed these weight vectors for each of the seven combinations of feature sets, we construct a contrast matrix by pairing each of the seven vectors with each of the other seven vectors. The difference between the hold-time vector and the down-down-time vector is, in effect, a contrast of the average effect of hold times vs. the average effect of down-down times. There are 21 such pairings (i.e., *7 choose 2*), and together these contrast vectors compare each feature-set combination to each other feature-set combination.

These contrasts were all tested simultaneously, using the multiple-comparisons correction from Chapter 3. Table 6.4 presents the estimated contrasts, standard errors, t-statistics, and adjusted p-values. The p-values alone confirm that there are statistically significant differences between most of the feature-set combinations. Looking at the effect sizes, we see that the biggest contrast (in absolute terms) is between (a) the combination of hold and up-down times and (b) down-down times alone (i.e., H+UD vs. DD). Empirically, these two feature sets have the lowest and highest average miss rate across all classifiers and typing tasks. Since the sign of the estimate is negative, we conclude that hold and up-down times have the lowest miss rate, and down-down times have the highest.

Scanning the p-values related to the hold and up-down feature set (i.e., H+UD), we find that this feature set performs significantly better than every other combination of features except hold and down-down features (i.e., H+UD vs. H+DD). Both of these combinations of features are commonly used in keystroke-dynamics research, and based on this analysis, they appear to be similarly good feature sets to use. Note that the only other non-significant test compares hold and down-down features against hold, down-down, and up-down features (i.e., H+DD+UD vs. H+DD). One might ask whether this third feature set is just as good as the other two. Unfortunately, despite our intuition, statistical test results are not necessarily transitive. Even though we find no significant difference between H+UD and

	effect	stderr	t.stat	p.value
DD vs. H	13.513	0.355	38.057	<.0001
UD vs. H	11.236	0.355	31.643	<.0001
H+DD vs. H	-3.079	0.305	-10.086	<.0001
H+UD vs. H	-3.455	0.305	-11.318	<.0001
DD+UD vs. H	7.961	0.355	22.420	<.0001
H+DD+UD vs. H	-1.980	0.305	-6.484	<.0001
UD vs. DD	-2.277	0.305	-7.460	<.0001
H+DD vs. DD	-16.592	0.355	-46.729	<.0001
H+UD vs. DD	-16.968	0.355	-47.788	<.0001
DD+UD vs. DD	-5.552	0.305	-18.187	<.0001
H+DD+UD vs. DD	-15.493	0.355	-43.632	<.0001
H+DD vs. UD	-14.315	0.355	-40.315	<.0001
H+UD vs. UD	-14.691	0.355	-41.374	<.0001
DD+UD vs. UD	-3.275	0.305	-10.727	<.0001
H+DD+UD vs. UD	-13.215	0.355	-37.218	<.0001
H+UD vs. H+DD	-0.376	0.305	-1.232	0.8784
DD+UD vs. H+DD	11.040	0.355	31.092	<.0001
H+DD+UD vs. H+DD	1.100	0.305	3.602	0.0054
DD+UD vs. H+UD	11.416	0.355	32.151	<.0001
H+DD+UD vs. H+UD	1.476	0.305	4.834	<.0001
H+DD+UD vs. DD+UD	-9.940	0.355	-27.995	<.0001

Table 6.4: Hypothesis tests comparing miss rates on different feature sets. The average (variance-stabilized) miss-rates for each feature-set combination are compared. Effects are averaged over all three classifiers and all three typing tasks. The feature-set with the lowest estimated miss rate is H+UD (hold and up-down times), and the only other feature set without a significantly worse miss rate is H+DD (hold and down-down times).

H+DD, and we find no significant difference between H+DD and H+DD+UD, we *do* find a significant difference between H+UD and H+DD+UD. With this set of test results, we simply conclude that either H+UD or H+DD provide the lowest miss rates.

6.6 Validation #1: Feature sets

We validate the model by using it to make predictions. As in the earlier investigations, we compare those predictions to the results of a secondary evaluation, first with per-user prediction intervals and then with QQ-plots. The data for the validation was collected using the same procedure as for the initial evaluation. Table 6.1 on 131 reported the number of subjects who typed the Strong, Simple, and Numeric tasks for this secondary data set.

Using the validation data set, we repeat the evaluation procedure described in Section 6.3.3. As with the first analysis, in this first validation step, we restrict our attention to

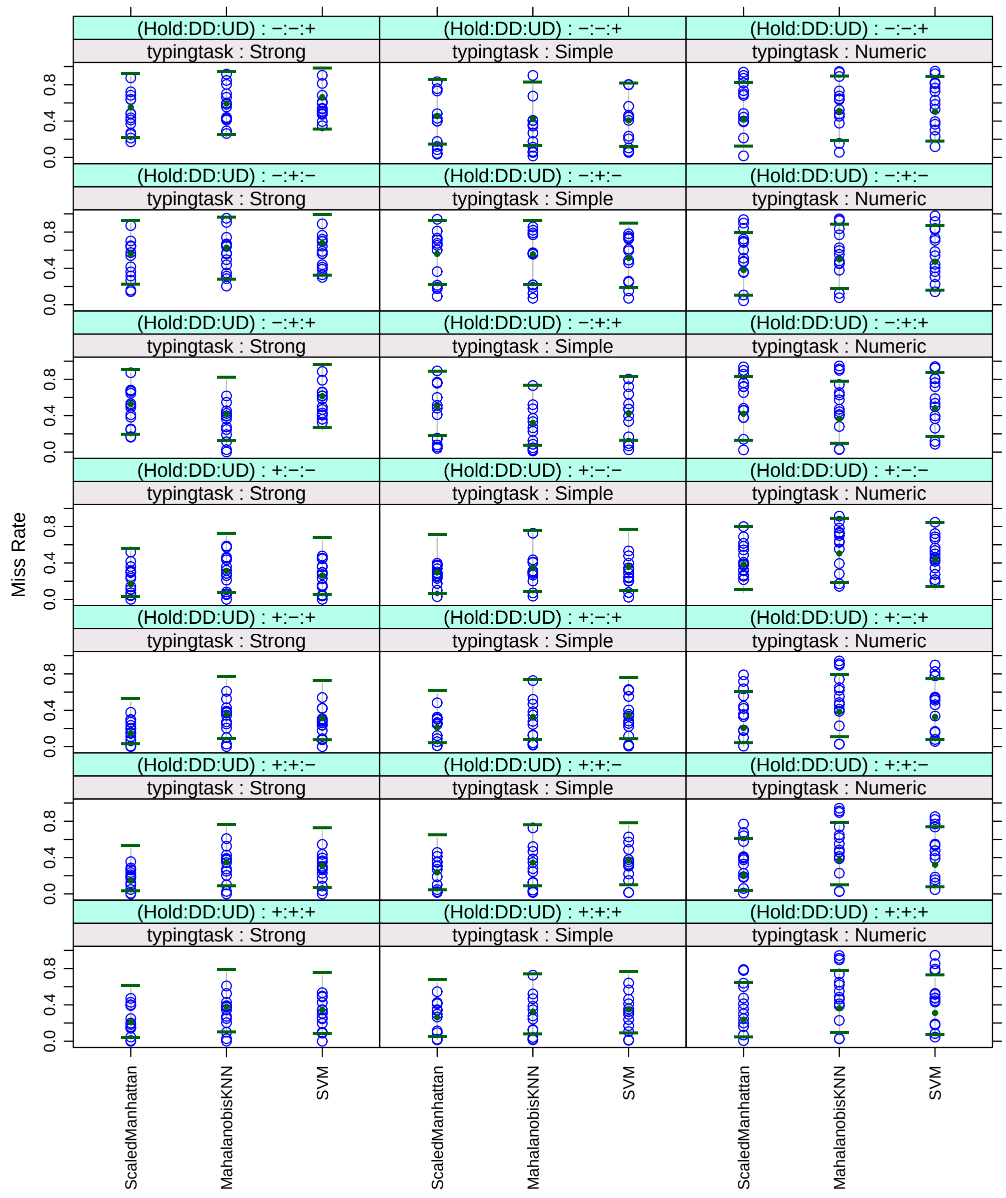


Figure 6.6: Model predictions and validation data for feature-set analysis. 95th percentile prediction intervals are compared to evaluation data for each classifier, typing task, and feature set. The data largely fall within the expected intervals.

the subset of results with 100 samples of training, no updating, and unfamiliar impostors. With 3 classifiers, 14 feature sets, 14 subjects in the Strong and Numeric data sets and 12 subjects in the Simple data set, we have 20832 miss rates (3 classifiers $\times$ 14 feature sets $\times$ 14 genuine-user subjects $\times$ 13 impostor subjects for each of the Strong and Numeric typing tasks and 12 genuine-user subjects $\times$ 11 impostor subjects for the Simple data set). From these per-user/impostor miss rates, we can calculate the average per-user miss rate for each user.

Figure 6.6 compares the model predictions to the per-user miss rates in the validation data set. As in the previous validation steps, each panel contains the results for the classifiers under one set of conditions. The particular conditions are expressed above each panel. The presence or absence of each of the three timing features—hold times, down-down times, and up-down times—is denoted by a triplet of $+/-$ symbols. Within each panel, the prediction intervals for each classifier are represented by horizontal lines. The per-user effects in the validation data set are represented as points.

By and large, the points fall within the expected interval within each panel. In the top panels, where hold times have been removed, the miss rates are higher, as predicted. In the bottom panels, where hold times are included, the miss rates are lower, also as predicted. Note that, of all the panels, the ones where the most points exceed the prediction interval are in the right column. This column corresponds to the Numeric typing task. It would appear that the miss rates for this task are more variable than the miss rates for the other two typing tasks, a possibility not considered among the family of models we considered during model selection. (Future work might consider more expressive models that capture different variance parameters for different typing tasks.)

We complete the validation step by checking modeling assumptions with QQ-plots. Figure 6.7 presents the three QQ-plots for this analysis. The left panel assesses the distribution of the per-user standardized effects. These effects largely fall along the line predicted for Normally distributed effects. The right panel assesses the distribution of the residuals. The quantiles of the residuals are also largely in keeping with the Normality assumptions of the model.

The middle panel assesses the distribution of the per-impostor standardized effects. Aside from one outlier (in the lower left corner), these effects are also in line with what would be expected from Normally distributed effects. Examining the results responsible for that outlier in more detail, we find one subject who is apparently very easy to detect as an impostor. When designated as an impostor, this subject's typing is perfectly classified as that of an impostor (i.e., a 0% miss rate) for 9 of the 13 other subjects in the validation

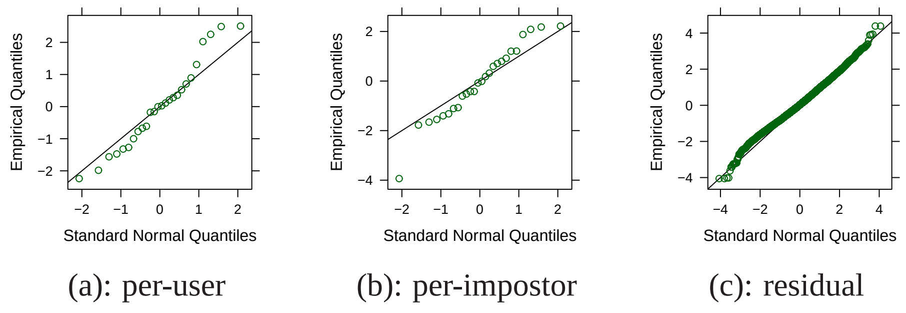


Figure 6.7: QQ-plots for validating LMM predictions from the feature-set analysis. The panels present per-user effects (left), per-impostor effects (middle), and residuals (right). In each panel, the line represents where one would expect Normally distributed points to fall. The per-user and residual plots largely conform to expectation. The per-impostor plot has a single outlier corresponding to one impostor who was exceptionally easy to detect.

data set. A single outlier is not surprising, but it serves as a reminder that the reality of keystroke-dynamics behavior may involve a heavier tail and more extreme values than predicted by current models.

In summary, the validation has provided evidence that the models are fairly accurate at predicting both average miss rates and the variability around those averages. Extreme values occur somewhat more frequently than would be expected from truly Normal distributions, but the general effect of different typing tasks and feature sets on classifiers has been captured.

6.7 Statistical analysis #2: Many-factor analysis

The previous analysis and validation aimed to understand the effect of different combinations of features. Due to the tractability issues that arise when analyzing and fitting models to large data sets, we chose to conduct that part of the analysis first, and to use it to inform the remainder of the analysis. We found that hold times and either down-down or up-down times produce the lowest miss rates across classifiers and typing tasks. In the current analysis, we restrict our attention only to those evaluation results when classifiers were trained and tested using hold times and up-down times. The previous analysis found that removing the Return-key features had negligible effect on miss rates, and so we keep them as features in the data used for this analysis.

For the subset of evaluation results described above, we investigate the effect of other factors in the evaluation environment: (1) decreased amounts of training, (2) use of an

Minimal Model Equation:

$$\begin{aligned}
\mathrm{VST}(\mathit{miss\ rate})_{ijk} &= \mu + (\mathit{user})_i + (\mathit{impostor})_j + \varepsilon_k \\
(\mathit{user})_i &\sim N(0, \sigma^2_{(\mathit{user})}) \\
(\mathit{impostor})_j &\sim N(0, \sigma^2_{(\mathit{impostor})}) \\
\varepsilon_k &\sim N(0, \sigma^2_{\varepsilon})
\end{aligned}$$

Maximal Model Equation:

$$\begin{aligned}
\mathrm{VST}(\mathit{miss\ rate})_{ijklmnop} &= \mu + (\mathit{Classifier})_i \times (\mathit{Typing\ Task})_j \\
&\quad \times (\mathit{Training\ Amount})_k \times (\mathit{Updating})_l \\
&\quad \times (\mathit{Imp.\ Familiarity})_m \\
&\quad + (\mathit{user})_n + (\mathit{impostor})_o + \varepsilon_p \\
(\mathit{user})_n &\sim N(0, \sigma^2_{(\mathit{user})}) \\
(\mathit{impostor})_o &\sim N(0, \sigma^2_{(\mathit{impostor})}) \\
\varepsilon_p &\sim N(0, \sigma^2_{\varepsilon})
\end{aligned}$$

Figure 6.8: Equations for model selection in the many-factor analysis. The minimal equation includes only the random effects, which are part of the structure of the experiment. The maximal equation includes fixed effects for the classifier, the typing task, the amount of training data, the updating strategy (yes/no), and impostor familiarity (none/high). The $A \times B \times C$ notation indicates that all higher-order interactions among the factors are also included (up to the 5-way interaction involving all the factors).

updating strategy, and (3) impostor familiarity. We consider the effects of these factors for all three classifiers and all three typing tasks. As with the other statistical analyses, we employ model selection, parameter estimation, and statistical hypothesis testing.

6.7.1 Model selection

The minimal and maximal model equations used for this model-selection step are presented in Figure 6.8. The minimal model equation is the same one used in the previous analysis, containing only random-effect terms. The maximal model equation adds fixed-effect terms for the classifier, typing task, amount of training, updating strategy, and impostor familiarity; it also includes all two-, three-, four-, and five-way interactions among these factors. To find the best fitting model from this space, we use stepwise model selection with each model's BIC score.

Selected Model Equation (Many-Factor Analysis):

$$\begin{aligned}
VST(\mathit{miss\,rate})_{ijklmnop} &= \mu + (\mathit{Classifier})_i + (\mathit{Typing\ Task})_j + (\mathit{Training\ Amount})_k \\
&\quad + (\mathit{Updating})_l + (\mathit{Imp.\,Familiarity})_m + \\
&\quad + (\mathit{Classifier} : \mathit{Typing\ Task})_{ij} \\
&\quad + (\mathit{Classifier} : \mathit{Training\ Amount})_{ik} \\
&\quad + (\mathit{Typing\ Task} : \mathit{Training\ Amount})_{jk} \\
&\quad + (\mathit{Classifier} : \mathit{Updating})_{il} + (\mathit{Typing\ Task} : \mathit{Updating})_{jl} \\
&\quad + (\mathit{Classifier} : \mathit{Imp.\,Familiarity})_{im} \\
&\quad + (\mathit{Typing\ Task} : \mathit{Imp.\,Familiarity})_{jm} \\
&\quad + (\mathit{Classifier} : \mathit{Typing\ Task} : \mathit{Training\ Amount})_{ijk} \\
&\quad + (\mathit{Classifier} : \mathit{Typing\ Task} : \mathit{Imp.\,Familiarity})_{ijm} \\
&\quad + (\mathit{user})_n + (\mathit{impostor})_o + \varepsilon_p \\
(\mathit{user})_n &\sim N(0, \sigma^2_{(\mathit{user})}) \\
(\mathit{impostor})_o &\sim N(0, \sigma^2_{(\mathit{impostor})}) \\
\varepsilon_p &\sim N(0, \sigma^2_{\varepsilon})
\end{aligned}$$

Table 6.5: LMM model equations for the results of the many-factor analyses. Based on the model-selection criteria, each of these factors may have an effect on miss rates, and the effect may depend on the classifier and typing task. To understand what effects these factors have, we must look at the parameter-estimate table (Table 6.6).

6.7.2 Parameter estimation

Tables 6.5 and 6.6 present the model equation and parameter-estimation table respectively for this analysis. Again, the LMM has been split across two tables due to its size and complexity. The model equation does not contain any four-way terms, but it does contain multiple three-way interactions. Based on this equation, the effects of training amount, updating, and impostor familiarity depend on both the classifier and the typing task. For training amount and impostor familiarity, the dependency involves a three-way interaction. Overall, the model equation is further evidence that keystroke-dynamics classifiers exhibit complicated behavior; error rates cannot be predicted without knowing the particular conditions of deployment configuration and environment.

Drawing formal conclusions from the model will occur when we perform hypothesis tests. Before doing so, we can develop some intuition as to what factors are likely to affect miss rates by examining the rows in the parameter-estimation table with the largest values in absolute magnitude. The largest fixed-effect estimate in absolute terms is -25.39, representing the change in the baseline miss rate when 5 training samples are increased to 100. In general, based on the magnitude of the estimated effects, the amount of training data appears to be one of the most influential factors. Consider all three parameters that estimate how the baseline miss rate changes when the amount of training increases (i.e., the three `trainamt` rows). The first estimate (-12.15) is a large negative number, indicating that as the amount of training increases from 5 to 25 repetitions the miss rate improves substan-

Parameter Estimates (Many-Factor Analysis):

Parameters	classifier	typingtask	trainamt	updating	impfam	estimate
(μ) baseline	ScaledManhattan	Strong	5	None	Low	49.32
classifier	MahalanobisKNN					25.20
	SVM					23.31
typingtask		Simple				-0.81
		Numeric				3.14
trainamt			25			-12.15
			50			-19.29
			100			-25.39
updating				Sliding		-5.06
impfam					High	1.68
classifier:typingtask	MahalanobisKNN	Simple				-6.43
	SVM	Simple				-3.78
	MahalanobisKNN	Numeric				-11.87
	SVM	Numeric				-6.87
classifier:trainamt	MahalanobisKNN		25			3.50
	SVM		25			1.78
	MahalanobisKNN		50			-5.39
	SVM		50			-3.59
	MahalanobisKNN		100			-9.17
	SVM		100			-9.23
typingtask:trainamt		Simple	25			3.02
		Numeric	25			2.93
		Simple	50			4.21
		Numeric	50			0.02
		Simple	100			6.40
		Numeric	100			3.20
classifier:updating	MahalanobisKNN			Sliding		2.21
	SVM			Sliding		0.22
typingtask:updating		Simple		Sliding		-0.05
		Numeric		Sliding		3.00
classifier:impfam	MahalanobisKNN				High	1.54
	SVM				High	-0.95
typingtask:impfam		Simple			High	-0.18
		Numeric			High	11.40
classifier:typingtask:trainamt	MahalanobisKNN	Simple	25			-8.51
	SVM	Simple	25			-3.62
	MahalanobisKNN	Numeric	25			0.83
	SVM	Numeric	25			-0.22
	MahalanobisKNN	Simple	50			-3.13
	SVM	Simple	50			-1.09
	MahalanobisKNN	Numeric	50			10.61
	SVM	Numeric	50			2.53
	MahalanobisKNN	Simple	100			-0.67
	SVM	Simple	100			0.02
	MahalanobisKNN	Numeric	100			7.60
	SVM	Numeric	100			3.35
classifier:typingtask:impfam	MahalanobisKNN	Simple			High	-0.45
	SVM	Simple			High	0.38
	MahalanobisKNN	Numeric			High	1.46
	SVM	Numeric			High	-4.84
$\sigma_{(user)}$						10.65
$\sigma_{(impostor)}$						7.19
σ_{ε}						20.06

Table 6.6: LMM parameter-estimate table for the many-factors analysis. The miss rates depend on the particular combination of classifier, typing task, amount of training, updating, and impostor familiarity.

tially (i.e., from 48.9% to 30.4%). The second estimate (-19.29) is an even larger negative number, indicating that increasing training from 5 to 50 offers even more improvement in the miss rate (i.e., to 20.7%). The even-more-negative third estimate (-25.39) suggests further improvements when 100 samples are used (i.e., to 13.5%).

The second-largest fixed-effect estimate is the classifier, reminding us perhaps that despite the other factors, the classifier does matter. In this case, the estimates are 25.20 and 23.31 for swapping the baseline Scaled Manhattan classifier for the Mahalanobis k-NN and SVM classifiers, respectively. Compared to the 48.9% miss rate estimated for the Scaled Manhattan, the SVM miss rate is estimated to be 82.6%, and the Mahalanobis k-NN miss rate is estimated to be 84.8%. These estimates are so much larger because, in the baseline, the classifier is trained with only 5 training samples; apparently these two classifiers require many more training samples to provide miss rates competitive with those of the Scaled Manhattan classifier. This observation about the effect of a small training sample on SVMs and Mahalanobis k-NN is supported by the theory behind both classifiers. For instance, accurately estimating a large covariance matrix, as needed by the Mahalanobis k-NN is basically impossible with only 5 training samples (Hastie et al., 2001).

Three other fixed-effect estimates have absolute magnitude greater than 10.0, an arbitrary but reasonable threshold between small and large effects. Interestingly, all three additional large effects concern the Numeric task. The first effect is the interaction between the Mahalanobis k-NN classifier and the Numeric task is -11.87. A large negative number suggests that for this particular task, the miss rate of the Mahalanobis k-NN classifier is less vulnerable to having only 5 training samples. Perhaps the controlled nature of typing a phone-number-like code with a single finger makes users type more consistently, so that the Mahalanobis k-NN builds a comparatively accurate typing profile.

The second effect is the three-way interaction between the Mahalanobis k-NN classifier, the Numeric task, and 50 training repetitions: 10.61. The large positive number suggests that the above interaction between Mahalanobis k-NN and the Numeric task at 5 repetitions is effectively cancelled out by this effect at 50 repetitions. If nothing else, the prevalence of large estimates involving the Mahalanobis k-NN classifier and the amount of training data suggests the classifier is very sensitive to the particular training data set.

The final large effect involving the Numeric task concerns the change in the miss rate for impostors who become very familiar with the typing task: 11.40. This large positive number suggests that the Numeric typing task is particularly vulnerable to impostors who intend to evade detection by practicing the task and becoming familiar with it.

While our investigation is intended to understand the factors other than the classifier that affect miss rates, one cannot ignore the practicalities of choosing the best classifier for each combination of factors. Looking over the parameter estimates in the table, we find that the Scaled Manhattan classifier continues to have the lowest estimated miss rate across all combinations of typing task, training amount, typing task, updating strategy, and level of impostor familiarity. Having made these general observations about the model, we turn our attention to drawing more formal conclusions using statistical hypothesis testing.

6.7.3 Statistical hypothesis testing

In an investigation as complicated as this one, with so many factors and interactions, one can ask many different research questions. For this work, based on practicalities of keystroke dynamics, we consider a few questions in particular. First, does each increment in training, from 5 to 25, 25 to 50, and 50 to 100 samples, lower the miss rate? Second, does impostor familiarity increase the miss rate across all classifiers and typing tasks? Third, does updating reduce the miss rate across classifiers and typing tasks?

As in the other investigations, we use multiple testing procedures to adjust our p-values. Note that, even though we have divided our tests into different questions, we correct the p-values for all the questions simultaneously. It is only in presenting the test results that we have separated them into different sets of contrasts.

To investigate the effects of training, we consider each combination of classifier and typing task separately. With 3 classifiers and 3 typing tasks, there are 9 such combinations. For each one, we construct a contrast matrix to make three comparisons: (1) are 25-training-sample miss rates different from 5-sample miss rates; (2) are 50-sample miss rates different from 25-sample miss rates; (3) are 100-sample miss rates different from 50-sample miss rates?

Based on the model equation, the effect of training depends on the classifier and the typing task, but not on the use of updating or impostor familiarity. With 9 combinations of classifier and typing task, and with 3 tests for each combination, we have 27 tests in total. We construct a contrast matrix for these 27 tests.

Table 6.7 presents the results of these tests. First, note that all the tests are highly significant, meaning that in every case, the miss rate is significantly lower with more training. Looking at the effects column, we observe that most of the big effects are seen when going from 5–25 samples and 25–50 samples. On some level, this observation suggests that we start to see diminishing returns in miss-rate reduction as we continue to add training samples. Nevertheless, the significance of all the 50 vs. 100 sample comparisons shows that we have not yet reached that point with 100 samples. All three classifiers for all three typing

typingtask	classifier	trainamt	effect	stderr	t-stat	p-value
Strong	ScaledManhattan	5–25	12.151	0.556	21.843	<.0001
		25–50	7.137	0.556	12.830	<.0001
		50–100	6.098	0.556	10.961	<.0001
	MahalanobisKNN	5–25	8.655	0.556	15.558	<.0001
		25–50	16.019	0.556	28.796	<.0001
		50–100	9.886	0.556	17.771	<.0001
	SVM	5–25	10.369	0.556	18.640	<.0001
		25–50	12.512	0.556	22.491	<.0001
		50–100	11.734	0.556	21.093	<.0001
Simple	ScaledManhattan	5–25	9.130	0.556	16.412	<.0001
		25–50	5.951	0.556	10.697	<.0001
		50–100	3.901	0.556	7.013	<.0001
	MahalanobisKNN	5–25	14.141	0.556	25.420	<.0001
		25–50	9.455	0.556	16.996	<.0001
		50–100	5.233	0.556	9.406	<.0001
	SVM	5–25	10.964	0.556	19.709	<.0001
		25–50	8.802	0.556	15.823	<.0001
		50–100	8.427	0.556	15.149	<.0001
Numeric	ScaledManhattan	5–25	9.223	0.556	16.579	<.0001
		25–50	10.048	0.556	18.063	<.0001
		50–100	2.918	0.556	5.245	<.0001
	MahalanobisKNN	5–25	4.898	0.556	8.805	<.0001
		25–50	9.152	0.556	16.451	<.0001
		50–100	9.711	0.556	17.457	<.0001
	SVM	5–25	7.661	0.556	13.771	<.0001
		25–50	12.670	0.556	22.776	<.0001
		50–100	7.742	0.556	13.916	<.0001

Table 6.7: Hypothesis tests comparing error rates as amount of training increases. Each increment in training-set size is tested for each combination of classifier and typing task. In every case, the effect of increased training is highly significant, with higher effect sizes for increases from 5–25 and 25–50 samples.

tasks benefit from very large amounts of training data.

To investigate the effect of impostor familiarity, we again consider each combination of classifier and typing task separately. Based on the model equation, the effect of impostor familiarity depends on the classifier and typing task, but not on amount of training or updating. We construct a contrast matrix with which, for each combination of classifier and typing task, we test whether impostor familiarity has an effect on the miss rate.

Table 6.8 presents the results of these tests. In this table, it is most interesting to note which p-values are not highly significant. At the 5% level, the only two cases where we cannot find a significant effect are with the SVM classifier for the Strong and Simple typing tasks. It would appear that impostor familiarity is a threat to keystroke-dynamics accuracy, raising the miss rates in all but these two cases. Even for the Numeric typing task, where the effect of a familiar impostor is much larger for all three classifiers, the effect for the SVM is relatively smaller. We are aware of no theoretical justification for why the SVM would be comparatively robust to impostor familiarity, but the possibility is intriguing.

To investigate the effect of updating, we perform a similar set of comparisons. In the

typingtask	classifier	effect	stderr	t-stat	p-value
Strong	ScaledManhattan	1.682	0.393	4.276	0.0008
	MahalanobisKNN	3.225	0.393	8.200	<.0001
	SVM	0.729	0.393	1.854	0.9160
Simple	ScaledManhattan	1.497	0.393	3.806	0.0057
	MahalanobisKNN	2.587	0.393	6.578	<.0001
	SVM	0.921	0.393	2.342	0.5267
Number	ScaledManhattan	13.084	0.393	33.263	<.0001
	MahalanobisKNN	16.083	0.393	40.886	<.0001
	SVM	7.287	0.393	18.525	<.0001

Table 6.8: Hypothesis tests comparing miss rates for familiar and unfamiliar impostors. For many combinations of classifier and typing task, impostors familiar with the task have significantly higher miss rates than those that are unfamiliar. For some typing tasks, the SVM classifier appears robust against this potential vulnerability.

classifier	effect	stderr	t-stat	p-value
ScaledManhattan	-5.063	0.293	-17.267	<.0001
MahalanobisKNN	-2.855	0.293	-9.737	<.0001
SVM	-4.844	0.293	-16.521	<.0001
typingtask				
Strong	-5.063	0.293	-17.267	<.0001
Simple	-5.108	0.293	-17.421	<.0001
Number	-2.062	0.293	-7.034	<.0001

Table 6.9: Hypothesis tests comparing miss rates with and without updating. If typing styles evolve over time, a classifier that updates its profile will be more accurate. These test results show that updating significantly reduces miss rates for three different classifiers on three different typing tasks.

model equation, updating interacts with the classifier and the typing task, but there is no three-way interaction. The absence of a three way interaction means that the effect of updating may change with the classifier and with the typing task, but the effect of changing both is additive. As a result, we can test whether updating has an effect for each of the three classifiers and each of the three typing tasks, without considering all nine combinations of classifier and typing task.

Table 6.9 presents the results of these tests. Note that all tests have highly significant p-values, meaning that for each classifier and typing task, updating has an effect. All the effect estimates are negative, reassuringly showing that updating lowers the miss rates. The improvement appears to be somewhat less for the Mahalanobis k-NN and the Numeric typing task, but the overall finding from these tests is that updating helps.

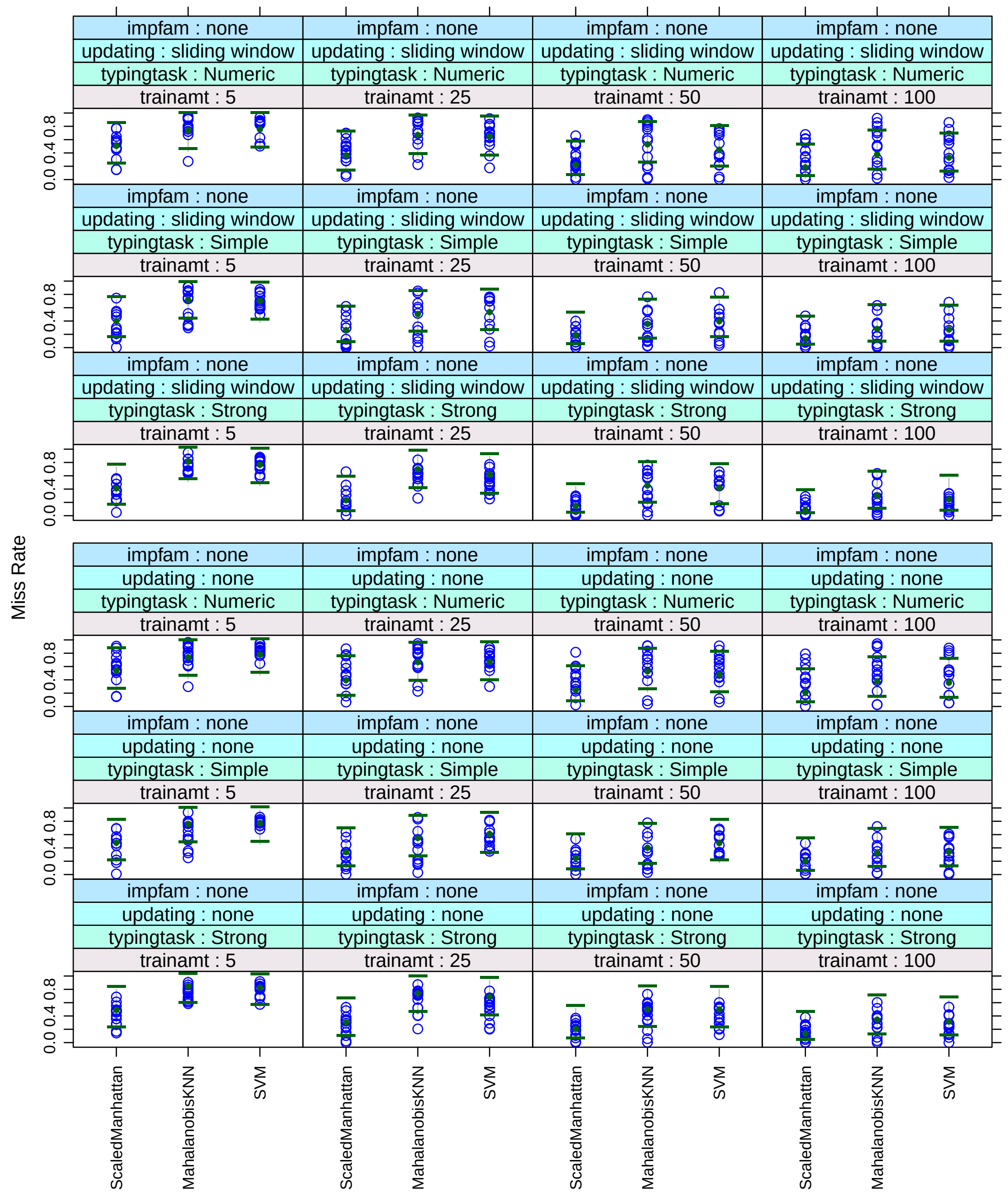


Figure 6.9: Model predictions and validation data for training analysis (part 1). 95th percentile prediction intervals are compared to evaluation data for each classifier, typing task, training amount, and updating strategy for unfamiliar impostors. The data largely fall within the expected intervals.

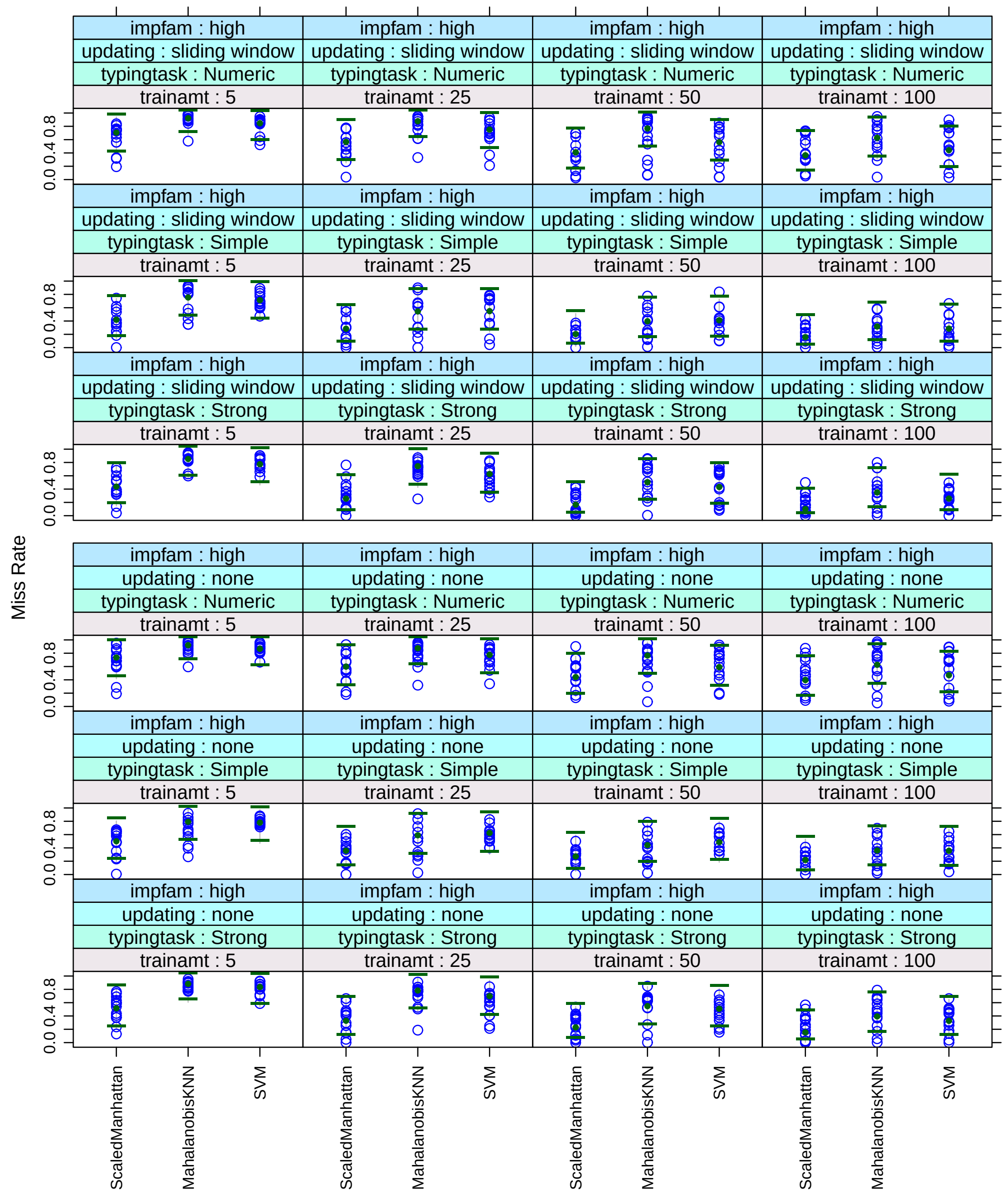


Figure 6.10: Model predictions and validation data for training analysis (part 2). 95th percentile prediction intervals are compared to evaluation data for each classifier, typing task, training amount, and updating strategy for highly familiar impostors. The data largely fall within the expected intervals.

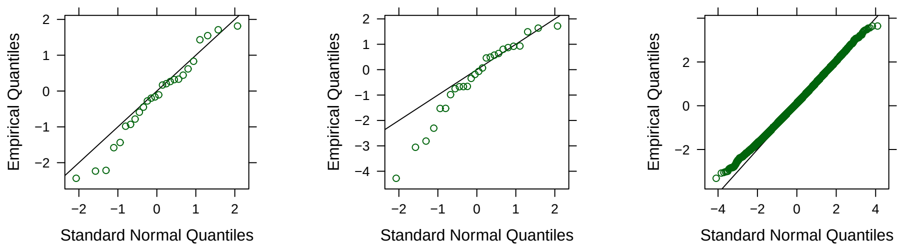


Figure 6.11: QQ-plots for validating LMM predictions from the many-factor analysis. The panels present per-user effects (left), per-impostor effects (middle), and residuals (right). In each panel, the line represents where one would expect Normally distributed points to fall. The per-user effects and residuals largely conform to expectation. The per-impostor effects show a somewhat heavy left tail.

6.8 Validation #2: Many-factor analysis

To validate this model, we again use prediction intervals and QQ-plots. We run the evaluation procedure described in Section 6.3.3 for each combination of the 3 classifiers, 4 amounts of training, 2 updating strategies, 2 impostor-familiarity levels, and 12–14 genuine user subjects for each of the typing tasks (i.e., 12 for the Simple task, and 14 for the Strong and Numeric tasks). The evaluations result in 23,808 miss rates.

Figures 6.9 and 6.10 compare the per-user model predictions to the per-user average miss rates on the validation data. The first figure contains predictions for unfamiliar impostors. The second figure contains predictions for highly familiar impostors. In each figure, the panels in the top half concern classifiers that use updating; the panels in the bottom half concern classifiers that do not use updating. Each row corresponds to results for a different typing task, and from left to right, each column corresponds to increasing amounts of training data.

Across the many panels in the two figures, the predictions largely appear to be accurate. The 95th percentile prediction intervals enclose most of the empirical results. A few points outside the intervals are to be expected, and the number of such points appears to be in the expected proportion (i.e., about 1 in 20). It appears that points outside the prediction intervals are typically below the interval, which corresponds to a lower-than-predicted miss rate.

In addition to validating the per-user error-rate predictions, we validate the modeling assumptions using QQ-plots. Figure 6.11 presents the QQ-plots for this validation. The per-user effects in the left panel appear to be in line with the Normality assumption. The per-impostor effects in the middle show evidence of a heavy left tail. This deviation from

Normality is consistent with our observation that miss rates outside the prediction interval tend to be lower than the interval rather than higher. While it would be best for the Normality assumption to be completely satisfied, of the possible violations, it is probably better to over-estimate the miss rates than to under-estimate them. The residuals in the QQ-plot on the right are largely consistent with the quantiles of a Normal distribution. Overall, while the assumptions are not perfectly met, the data are mostly consistent with the modeling assumptions, and the model is making accurate predictions.

6.9 Discussion

More clearly than any of the earlier investigations, this chapter establishes that many factors affect keystroke-dynamics error rates. We have estimated the effect of different typing tasks, different classifiers, different classifier configurations (e.g., feature set, amount of training, and updating), and different evasion strategies that impostors might use. All of these factors affect the error rate, and the specifics of the effect depend on complex interactions between factors.

As noted at the beginning of the chapter, other studies have considered various factors that might affect keystroke-dynamics error rates. Joyce and Gupta (1990) used only 8 password repetitions to train their classifier, but they found that the same accuracy was observed with as few as 6 repetitions. Araújo et al. (2005) considered training sets ranging from 6 to 10 repetitions. Bartmann et al. (2007) considered training repetitions over as large a range as in our investigation (i.e., 5–100), and they found that error rates stabilized with as few as 30 repetitions. Note that their classifier used both genuine-user and impostor samples during training, and so their findings are for a different family of classifiers. To our knowledge, all earlier research on the amount of training has focused on a single classifier. Unlike the earlier work, our work shows that the amount of training affects different classifiers differently; this interaction should be taken into account.

Araújo et al. (2005) evaluated a Scaled Manhattan classifier with the seven different feature sets we used in this investigation. They found that using all three types of feature (e.g., hold times, down-down times, and up-down times) produced the lowest error rates. In contrast, we found that, so long as hold times and either down-down or up-down times are included, the particular combination has little effect. Our findings benefit from the hypothesis testing we used to test whether small differences in the error rate are significant. The earlier work assumed that any difference in the empirical error rates was significant.

Regarding updating as a factor, Araújo et al. (2005) compared a Scaled Manhattan classifier with and without updating. Kang et al. (2007) compared a k-means classifier trained with no updating to ones trained with growing and sliding windows. Both sets of researchers found that an updating strategy lowered error rates. Their results for individual classifiers, coupled with our results for three classifiers (backed by a validated statistical model), strongly support the claim that a sliding-window updating scheme reduces classifier error rates.

Regarding impostor familiarity with the typing task, Lee and Cho (2007) gave their impostors the opportunity to practice, but they did not describe how many repetitions of practice were taken by each impostor. Araújo et al. (2005) split their impostor subjects into two groups. One group observed the genuine user typing the password, and one group did not. The observers seemed to be more successful at mimicking the typing style of the genuine user, but no statistical test was performed. In contrast, our work operationalized practice in terms of the number of repetitions, and then quantified the effect of practice on error rates.

In general, while each of the factors considered in this investigation has been investigated previously, the earlier work has looked at the factors largely in isolation. For instance, amount of training is investigated keeping all other factors constant, using only one classifier. This investigation has shown that different factors matter for different classifiers. Factors do not exert their effects in isolation; they exert their effects in concert with the effects of other factors. One-factor-at-a-time investigations ignore that complexity.

6.10 Summary

This chapter presented a third use of LMMs to understand keystroke-dynamics error rates. A series of evaluations was conducted in which 3 classifiers, 3 typing tasks, 14 feature sets, 4 different amounts of training data, 2 updating strategies, and 2 levels of impostor familiarity with the typing task were all involved in the evaluation. The results were split between two analyses. In the first analysis, LMMs were used to identify the best timing features to use. In the second analysis, LMMs were used to understand the myriad other factors and their effects on classifier miss rates.

From this investigation, we draw the following conclusions:

1. *The Scaled Manhattan classifier continues to have the lowest miss rates, across most feature sets, typing tasks, amounts of training, updating strategies, and impostor-familiarity levels.* The actual miss rates depend on the setting, but in most cases, the Scaled Manhattan miss rates were estimated to be the lowest. (The only exceptions

involved sub-optimal feature sets, with which other classifiers had lower estimated miss rates.)

2. *The miss rates across classifiers are significantly lower when hold times and either up-down times or down-down times are used as features than with any other combination of features tried.* Hold times are particularly useful; for the Scaled Manhattan classifier and one typing task (i.e., strong passwords), removing hold times as features increases the estimated miss rate from 20.9% to 52.7%.
3. *Increasing the number of typing samples used to train a classifier significantly reduces classifier miss rates even when the initial training test is large (e.g., 50 samples).* This conclusion was tested for each of three classifiers and three typing tasks, and it held in every case. When increasing from 50 to 100 samples, the smallest improvement in miss rates was from 24.8% to 21.0% (for the Scaled Manhattan classifier and the Numeric task); this difference was still statistically significant.
4. *Impostors who become familiar with a typing task often significantly increase the miss rate (i.e., the chance of successfully evading detection).* For the Scaled Manhattan classifier, the most extreme example is on the Numeric typing task, where impostor familiarity increased the miss rate from 53.9% to 73.5% (under already difficult conditions such as few training samples). Interestingly, there is some evidence that the SVM classifier might be robust to impostor familiarity, but that conjecture requires further research.
5. *Employing an updating strategy significantly reduces miss rates across classifiers and typing tasks.* In particular, updating reduced the miss rate for the Scaled Manhattan classifier from 48.9% to 41.0% (again under difficult initial conditions).

These results offer further evidence that to understand keystroke-dynamics error rates, one must understand the multitude of factors in the evaluation environment (beyond the classifier itself). In fact, the particular classifier may have less of an effect on error rates than factors like the amount of training or the feature-set used. Additionally, this chapter offered another example in which analysis using LMMs enabled us to make sense of the complexities of keystroke-dynamics error rates.

Recall the thesis statement from Chapter 1 in which we claimed in part that LMMs offer better understanding of classifier behavior than current practices. In this investigation, we reviewed other efforts to understand the effects of different features, different amounts of training, different updating strategies, and different levels of impostor familiarity on classifier behavior. Those efforts typically considered only one classifier and offered no definitive conclusions supported by inferential statistics. In contrast, this investigation used

LMMs to draw conclusions about which feature sets were best, how much training data is recommended, whether updating strategies are useful, and whether impostor familiarity poses a danger. Our conclusions were supported by both inferential statistics and validation using a secondary data set. Consequently, we believe that this investigation has further shown the benefit of using LMMs to understand classifier behavior.

Chapter 7

Summary, Contribution, and Future Work

In this section we review the discoveries that have been made through the investigations in this work, we enumerate the contributions of the work, and we consider implications and opportunities for future work. We believe that the methods used in this work would be useful beyond keystroke dynamics, and we offer some final thoughts about the need for inferential statistics in computer-security research.

7.1 Summary of findings

Recall that this work was motivated by wildly different error rates for the same classifier, across different evaluations. That background led us to frame the problem as follows:

> **In keystroke dynamics, a classifier does not have *an* error rate; it has *many* error rates, depending on a multitude of factors in the evaluation environment. Without identifying and understanding the effects of these factors, we cannot understand classifier behavior.**

This problem is serious because, without knowing the conditions under which a classifier will have a low error rate versus a high error rate, we cannot trust classifiers in critical security applications.

To address this problem, we proposed a methodology involving a series of evaluations, inferential statistics to draw conclusions from the evaluation results, and validation of the statistical models. We introduced linear mixed-effects models (LMMs) as an appropriate technique for the necessary statistical analysis, and we offered the following thesis statement:

Using linear mixed-effects modeling (LMM) to explain classifier behavior, (a) is novel for keystroke dynamics, and (b) offers better understanding of the behavior (e.g., changing error rates) than current practices.

In Chapter 2, we performed a literature review that demonstrated the different choices researchers make when conducting an evaluation, and suggested that these different choices may produce wildly different results. We established that inferential statistics might help us make sense of how these choices affect the results, but that they are rarely used in keystroke dynamics; in particular, LMMs are never used. In Chapter 3, we described LMMs and explained how they might give us needed understanding of the multitude of factors that affect classifier error rates. Through these chapters, we support part (a) of our thesis statement. Using LMMs to explain classifier behavior is novel for keystroke dynamics.

Then, we conducted three investigations to identify various factors that might affect classifier error rates. In Chapter 4, we investigated per-user and per-impostor effects on the error rates (i.e., the influence that individual typists have on raising or lowering the miss rate). In Chapter 5, we extended our investigation to explore whether personal traits—age, gender, dominant hand, and typing style—of the users and impostors explained the per-user and per-impostor effects. In Chapter 6, we screened several other factors—from the amount of training to the impostor's familiarity with the typing task—to understand their influence on classifier error rates.

From the first investigation, we drew the following conclusions.

1. *The Scaled Manhattan classifier has an estimated miss rate of 14.1%, significantly lower than those of the other 9 benchmarked classifiers.*
2. *However, the estimated 95% prediction interval for per-user long-term average miss rates is from 0.0% to 63.1% for Scaled Manhattan, and spans a similarly large range for the other classifiers.*
3. *Likewise, the estimated 95% prediction interval for per-impostor long term average miss rates is from 0.0% to 58.4% for Scaled Manhattan, and similarly large for the other classifiers.*
4. *Because of the high per-user effect on miss rates, an accurate estimate of a classifier's error rate (i.e., to within ± 1 percentage point) may require thousands of subjects.*

From the second investigation, we drew the following conclusions:

1. *The Scaled Manhattan classifier continues to have the lowest estimated miss rate, regardless of the user and impostor age, gender, dominant hand, or typing style.*
2. *When the impostor is a touch typist rather than another kind of typist, the Scaled*

Manhattan miss rate increases from 13.1*% to* 33.2*% if the genuine user is also a touch typist, and from* 8.9*% to* 21.4*% if the genuine user is not.*

3. *Other than the user and impostor typing style, none of the other tested traits (i.e., age, gender, or dominant hand) had a were found to have a significant effect on classifier miss rates.*

From the third investigation, we drew the following conclusions:

1. *The Scaled Manhattan classifier continues to have the lowest miss rates, across most feature sets, typing tasks, amounts of training, updating strategies, and impostor-familiarity levels.*
2. *The miss rates across classifiers are significantly lower when hold times and either up-down times or down-down times are used as features than with any other combination of features tried.*
3. *Increasing the number of typing samples used to train a classifier significantly reduces classifier miss rates even when the initial training test is large (e.g., 50 samples).*
4. *Impostors who become familiar with a typing task often significantly increase the miss rate (i.e., the chance of successfully evading detection).*
5. *Employing an updating strategy significantly reduces miss rates across classifiers and typing tasks.*

In each investigation, we drew these conclusions by evaluating classifiers under systematically varied conditions and analyzing the evaluation results using LMMs. We compared our findings to those of earlier work. In each investigation, by drawing inferences using LMMs and statistical principles, we were able to make discoveries and understand phenomena in ways that would not have been possible without this work. Through these investigations, we support part (b) of our thesis statement. LMMs offer better understanding of classifier behavior than current practices.

7.2 Impact for keystroke dynamics

The findings listed in the previous section are not simply of abstract scientific interest. They help researchers and practitioners who believe in the promise of keystroke dynamics to realize that promise. To someone who wants to use this work to make keystroke-dynamics better in the future, we present the following possibilities:

1. The figures and models of Chapter 4 (Benchmarking Keystroke-Dynamics Classifiers) show that for about a third of typists, the top classifiers (e.g., Scaled Manhattan

and k-NN) have nearly perfect accuracy. The average error rate of the Scaled Manhattan classifier might be 23.6%, but Figure 4.4 shows that for 16 subjects, this classifier's miss rate is nearly 0%. If researchers can identify what makes these subjects' typing so easy to distinguish, we would have a very promising technology for a large subset of the typing population.

2. The models and hypothesis-testing results from Chapter 5 (Personal Traits and Keystroke Dynamics) show that when the genuine user and impostor are both touch typists, miss rates are much higher. Perhaps by using the same typing technique, touch typists' rhythms are more similar than those users who have other typing styles. Different techniques may be needed to distinguish touch typists. Future researchers would do well to divide and conquer. Evaluating classifiers separately for users who touch type and those who use other typing techniques would allow us to find the (possibly distinct) classifiers that work best for each group. At the very least, future researchers should pay attention to the proportion of touch typists in their evaluation sample, and report it. Like the Hawthorne Effect described in Chapter 2, the touch-typing effect should be taken into account in future evaluations.
3. The models of Chapter 6 showed that the typing task and amount of training affect classifier accuracy. Further, both factors interact with each other and with other factors in their effects. For some typing tasks (e.g., Simple), fewer training repetitions are necessary to get low error rates. Based on these findings, future researchers might search more systematically for typing tasks that improve miss rates (e.g., by require fewer repetitions to provide high accuracy). One possibility based on the literature is full names. As we have already noted several times, it is dangerous to draw conclusions by comparing results across studies because of the many evaluation differences, but we have observed that many studies reporting low error rates had subjects type their own and each other's names. Typing one's own name is such a familiar task to many typists, that we would not be surprised if this task boosts keystroke-dynamics accuracy. (We did not include this task in our study because we always intended to share our data and including subjects' names as a typing task would preclude sharing the data.)
4. The models and tests of Chapter 6 also show that updating classifiers have better error rates than those that train only once. Future researchers might adopt an updating strategy in all their evaluations, but doing so may introduce run-time issues. Updating a typing profile is more efficient for some classification algorithms than others. In this work, we did not evaluate classifiers with respect to run-time, but future researchers

proposing new classifiers may want to consider whether their classifier can update its typing profile efficiently.

5. Finally, the models and tests in Chapter 6 reveal a systemic weakness in all classifiers evaluated. An impostor who practices a typing task and becomes familiar with it can have a much higher chance of evading detection. At the very least, future researchers should report whether their impostor subjects had the opportunity to practice. Researchers should probably ensure that impostor subjects have enough opportunity to practice that the evaluation results are valid for real-world impostors who would be willing to expend some effort to increase their chances of evading detection. In addition, these otherwise alarming findings included one piece of good news. The analysis revealed one classifier (SVM) that, for two typing tasks (Strong and Simple), did not exhibit the otherwise systemic increase in the miss rate as a result of impostor familiarity. Future research should investigate whether this seeming robustness is real, and whether it might offer a solution to the impostor-familiarity vulnerability for other classifiers and typing tasks.

We have listed five options that researchers have, as a result of this work, for improving keystroke dynamics. None of these options would be known without this work. Certainly, some of the options might have been explored anyway. A researcher might work to make a classifier update more efficiently (option 4) because it seems like a good idea. Even though good ideas might be explored on their own, the current work establishes the importance of each good idea, and the relative importance of different good ideas. Without this work, we would have no scientific evidence or methodology with which to evaluate which good ideas are most promising, and to make informed decisions about the future direction of keystroke-dynamics research.

We feel that it is worth noting an omission from these future-work options: another new classifier. This omission may seem surprising since most of the research in keystroke dynamics involves describing and evaluating a new classifier. However, as we showed in Chapter 4, most classifiers behave similarly. Even if, in theory, three classifiers should learn very different concepts (e.g., Scaled Manhattan, Mahalanobis k-NN, and SVM), in practice, they mostly make the same decisions for the same typing samples. A lot of prior research can be characterized as proposing yet another classifier, but the current work finds that the particular classifier plays a surprisingly small role in evaluation results. The evaluation conditions—and by extension, the operating conditions in which a classifier is deployed—play a much larger role. If we want to improve keystroke dynamics, we need to understand the effects of these evaluation conditions more than we need another classifier.

7.3 Supplemental materials

To help future researchers build upon the current work, we have created a webpage that acts as a supplement to this dissertation:

`http://www.cs.cmu.edu/~keystroke/ksk-thesis`

This webpage shares the data, classifier implementations, evaluation procedures, and analysis scripts used in our investigations. It has been argued by proponents of reproducible research that at least some of the research scholarship is in the data and software that generated the results, tables, and figures included in the report, not the report itself (Buckheit and Donoho, 1995). In this report, we have tried to provide as much relevant information as possible in the document itself. As an example, for completeness sake, we provided the model equations and parameter-estimation tables for the LMMs in each of our investigations. Nevertheless, despite our best efforts, some parts of the analysis may remain unclear or ambiguous.

On the webpage, we have shared the typing data collected from the subjects who completed the Strong, Simple, and Numeric typing tasks. We have also provided implementations of each of the 10 classifiers used in these investigations. Each classifier was implemented in the R programming language as a set of training and classification functions. We also provide scripts used to train and test each classifier, on each data set, for each investigation. These resources provide other researchers with an unprecedented opportunity to compare their own classifiers and data sets on a consistent basis with ours. We hope that these materials prove useful to future keystroke-dynamics researchers.

7.4 Contributions

As a result of this research effort, we have made the following contributions to keystroke-dynamics and computer-security research.

1. *A solution to the multitude-of-factors problem:* This work demonstrates that LMMs are a useful tool for understanding why a classifier's error rate varies. A classifier still has many different error rates, depending on evaluation conditions, but with LMM analysis, the factors responsible for the differences can be identified and understood.
2. *A series of benchmark data sets:* Several data sets were collected so that we might understand the factors that affect classifier error rates. These data sets were used to benchmark many existing classifiers, and they are being shared so that new classifiers might be benchmarked and soundly compared as well.

3. *A set of reference classifiers:* Researchers can use our reference implementations of classifiers with benchmark data sets that they gather, enabling sound comparisons between data sets as well as between classifiers.
4. *Recommendations of the best classifier under different conditions:* The LMM analysis revealed that the Scaled Manhattan classifier works best under many general conditions, but that other classifiers have particular strengths. The Mahalanobis k-NN is better equipped to handle unusual combinations of timing features. The SVM is least affected by impostors becoming familiar with the typing task.
5. *Establish personal traits that do (and do not) affect keystroke dynamics:* We discovered that impostors who touch type are more challenging to distinguish than those who do not. Many researchers have noted that other traits such as age, gender, and dominant hand might make typists harder to distinguish, but the hypothesis had never formally been tested. We found that none of these traits have a significant effect.

7.5 Limitations

Having enumerated the contributions of this work, we feel an obligation to be candid about its limitations, specifically with respect to the multitude-of-factors problem in keystroke dynamics and linear mixed-effects models as the solution. This work has only begun to understand the factors that affect classifiers across the wide range of keystroke-dynamics applications. Linear mixed-effects models offer a clear step forward, but LMMs are unlikely to be the final step.

The multitude-of-factors problem in keystroke dynamics. One aspect of this work that surprised us was the relatively high miss rates and high amount of per-user and per-impostor variability in these miss rates. In Chapter 4, we presented boxplots showing the range of miss rates for each genuine-user subject and each classifier (Figures 4.4 and 4.5). According to these figures, even when the average miss rate for a user is low, there are usually particular impostors who are nearly impossible to detect. Likewise, for classifiers, even when the average miss rate is low, there are particular users for whom the classifier fails.

Such findings are somewhat discouraging, not with respect to this particular research, but with respect to the promise of keystroke-dynamics research more generally. This particular research is aimed at understanding classifier error rates, whatever they are, rather than obtaining the lowest and least-variable error rates. Nevertheless, for keystroke dynamics in general, one might look at the tables and figures showing the high miss rates and high variability and wonder whether keystroke dynamics could ever be a practical technology, worth additional research effort.

We address this concern as follows. The classifiers evaluated in this work represent only a small area of keystroke dynamics: login-type authentication using only training samples from the genuine user. Among keystroke-dynamics problems, this area is probably one of the hardest. Because authentication is done at login time, the typing sample is necessarily very short. In this work, the samples were all 10 characters or less. With longer samples, a classifier has more information on which to make its decision, and such decisions are likely to be more accurate. As such, the discouraging miss rates presented in this work do not affect the promise of other keystroke-dynamics applications (e.g., for continual in-session authentication).

Further, even among login-type authentication tasks, classifiers which use only genuine-user training samples are likely to have worse error rates than other classifiers. For example, another family of classifier used in keystroke dynamics trains using samples from both the genuine user and also other users called *known impostors*. These known impostors provide typing samples which the classifier uses in contrast to the genuine-user samples. The hope is that by distinguishing the genuine user's typing from other users' typing, a classifier will be able to distinguish between the genuine user and a previously unseen (or *unknown*) impostors. Such classifiers may have lower error rates than those which train only on genuine-user samples. In probabilistic terms, it is harder to accurately estimate a probability density function than to decide which of two alternatives is most probable. The former is akin to training only on genuine-user samples; the latter is akin to training on genuine-user and known-impostor samples. These other kinds of login-type classifiers remain promising despite our somewhat discouraging results.

Our findings of a relatively high error rate and high variability may not carry over to these other classifiers and applications. What will carry over is our analytical methodology. In those other applications, it will remain true that classifiers do not have *an* error rate; they have *many* error rates, depending on a multitude of factors in the evaluation environment. We used LMMs to identify and understand these factors for one type of classifier in one application, but nothing precludes their use with other kinds of classifiers, and for other keystroke-dynamics applications.

Linear mixed-effects models as a solution. This work developed a methodology for using LMMs to understand keystroke-dynamics classifier evaluation data. We established that these models are useful for understanding classifier behavior, but we acknowledge that LMMs are unlikely to be the final word in understanding classifier behavior. There are limitations to the understanding and predictive capabilities that these models provide. One avenue for future work would be more elaborate or more refined statistical models.

In the current work, we used LMMs to express miss rates as a stochastic function of the classifier, the user, the impostor, and various other factors. The miss rate itself is an aggregation of individual hits and misses. For instance, the analysis in Chapter 4 depended on 25,500 miss rates, one for each of the 10 classifiers, 51 genuine-user subjects, and 50 impostor subjects. Each miss rate is actually the proportion of 50 typing samples from the impostor subject that were mislabeled by the classifier.

As naturally happens when data are aggregated, we lose some details. In particular, for this example, of the 50 typing samples included in the miss-rate calculation, one corresponds to the 1st repetition and one corresponds to the 50th. In the investigation from Chapter 6, we saw that as the impostor repeats the typing task and becomes familiar with it, the miss rate increases. As such, there may be a change in the miss rate between the 1st and 50th repetitions. When the results for all these repetitions are aggregated into a single miss rate, we lose the ability to find such fine-scale effects. There are factors that we cannot investigate using LMMs.

A possible refinement is *generalized linear mixed-effects models* (GLMMs). With LMMs, the response variable is expected to be continuous. Miss rates are bounded between 0.0 and 1.0, but within that range any proportion is a valid miss rate. Consequently, one can use LMMs to model the relationship between explanatory factors and the miss rate. However, if we were to analyze the individual hits and misses, our response variable is binary, not continuous. A typing sample is either missed (1) or not (0), and no intermediate value is reasonable. For such an analysis, LMMs would not be appropriate; GLMMs would be.

GLMMs offer the same expressiveness as LMMs in terms of fixed and random effects, but they also support non-continuous response variables (McCulloch et al., 2008). In particular, a logistic mixed-effects model (one kind of GLMM) would be appropriate for binary responses. In such a model, the explanatory variables are modeled as having a linear relationship with the *log odds* of the response. In other words, if p is the probability of a miss, the explanatory variables are linearly related to

$$\log(p/(1-p)).$$

The explanatory variables can still include the classifier, typing task, amount of training, and the per-user and per-impostor effects. They can also include fine-scale factors (e.g., the repetition number in a sequence of repetitions) that cannot be studied using LMMs.

As with LMMs, there are procedures for model-selection, parameter estimation, and hypothesis testing using GLMMs. In preparation for the current work, we explored the use of GLMMs for our analysis. In theory, these models would have been more powerful and possibly more appropriate, as we have just described. In practice, we discovered many computational issues in analyzing such large tables of evaluation results with GLMMs. Parameter estimation for individual models took weeks, at the end of which time, the estimation process often had not converged to the true maximum-likelihood estimates. These models are under active development, and we believe that additional algorithmic improvements and increased computing capabilities may soon make them a viable alternative.

We also must admit that there may be some appearance of arbitrariness to the analytical procedures we use throughout this work. For example, as explained in Chapter 3, when performing model selection, we use maximum-likelihood to estimate parameters, but when fitting the final model we use REML. REML estimates are generally believed to be better (i.e., unbiased), but they change the likelihood calculations for each model such that the model-selection criteria (BIC) cannot be used to choose between models. While this procedure, using maximum-likelihood for some stages and REML for others, is the current best practice (Pinheiro and Bates, 2000; Zuur et al., 2009), we admit to finding it dissatisfyingly ad-hoc.

This dissatisfaction was one of our reasons for including a validation step after each statistical analysis. Whatever procedure was used to construct the model, the validation ensured that the resulting model provided a useful capability. We demonstrated that the model could be used to predict classifier miss rates in subsequent evaluations. Such predictions convince us that the model is accurately describing classifier behavior on a very concrete level. Nevertheless, alternative modeling strategies, including Bayesian approaches to statistical inference, might be explored as a way to reduce dissatisfaction and increase our level of comfort with the modeling procedures.

7.6 A science of security: Beyond keystroke dynamics

Recently, the computer-security research community has faced increasing calls for a *Science of Security*, whereby researchers' goal would be "to develop foundational science to guide system design and understand the safety, security, and robustness of the complex systems on which we depend" (Evans and Stolfo, 2011, p.16). We believe that experiments and analysis, like those performed for this work, are the right methods for this new science.

The differences between security research and other sciences have led some researchers to challenge whether a science of security is possible. In other sciences, the topic of interest is usually studied under carefully controlled conditions dictated by the researcher. Where the laws of other sciences typically deal with average or expected behavior of the system under study, the concerns of security research usually involve the worst-case behavior of a system: what is the worst thing an adversary might do to this system, and how should a defender respond? The following quote succinctly captures the tension between scientific methods of experimentation and security research:

> *For certain areas of computer security, experiments seem useful, and the community will benefit from better experimental infrastructure, datasets, and methods. For other areas, it seems difficult to do meaningful experiments without developing a way to model a sophisticated, creative adversary.*
>
> (Stolfo et al., 2011)

We interpret this statement to mean that the role of experiments may be less than in other sciences, at least until the adversary's capabilities are satisfactorily modeled. In the meantime, the role of experiments is still greater than nil. There are areas where experiments seem useful, and we should do them.

We bring up this debate over the science of security in the current work because the multiplicity-of-factors problem in keystroke dynamics exists throughout security research. In intrusion detection, anomaly detectors have been used to monitor system behavior to find evidence of attacks (Forrest et al., 1996). Just as in keystroke dynamics, different classifiers are proposed based on a variety of pattern-recognition and machine-learning algorithms. These classifiers are evaluated in different evaluation environments by different researchers. During the evaluation, researchers choose the programs to run, the networks to monitor, and the attacks to unleash. Classifiers have different error rates in different evaluations, so the error rate must depend on the program, network, or attack.

In insider-threat detection, algorithms monitor user behavior as observed through the commands they run, the time they log in, the files they download, and the documents they print (Schonlau et al., 2001). Once again, different classifiers are proposed for these algorithms, and they are evaluated by collecting user-behavior data and injecting attacks. Once again, the same classifier has different error rates in different evaluations (Schonlau et al., 2001; Maxion and Townsend, 2004). To understand a classifier's behavior, one needs to understand the factors in the evaluation conditions that affect its error rate.

In worm detection, the top-performing detector depends on the data set (Stafford and

Li, 2010). Some factors in the data, such as the network topology or the normal traffic patterns, may interact with the detector, so that both must be understood in order to predict which detector will have the lowest error rate. In malware analysis, different detectors are evaluated with data sets of different size, using different features, and different proportions of malware and legitimate software (Walenstein et al., 2010). Once again, researchers have trouble distinguishing the effectiveness of the detector from the effects of evaluation decisions.

In all of these cases, it is likely that detection algorithms do not have *an* error rate, they have *many* error rates, depending on a multitude of factors in the evaluation conditions. Just as we have used LMMs to understand the multitude of factors in keystroke-dynamics evaluations, LMMs could be a powerful tool throughout computer-security research for understanding the behaviors of security technologies.

LMMs and inferential statistics are admittedly complicated, and conducting rigorous controlled experiments can be challenging in a domain such as computer security. Whatever the difficulty, researchers have a duty to draw inferences and offer their own explanation of their findings, going beyond the mere reporting of empirical results. By not making any inferences, they offer no explanations of the results, and the meaning of the evaluation is vague at best.

We believe that our work has identified an area in computer-security research where an experimental science of security would be useful. In keystroke dynamics, we have demonstrated its utility. As a result of this work, future researchers can make much more informed decisions about how best to improve keystroke-dynamics error rates. Beyond keystroke dynamics, our methods could help to solve long-standing problems throughout a cross-section of security research, and assist researchers in producing the fundamental principles required of a science of security.

Bibliography

ACM Digital Library. ACM Digital Library, 2011. `http://dl.acm.org`, accessed Aug. 2011.

A. A. E. Ahmed and I. Traore. Anomaly intrusion detection based on biometrics. In *6th Annual IEEE Systems, Man and Cybernetics (SMC) Information Assurance Workshop (IAW 2005)*, pages 452–453, June 15–17, 2005, West Point, NY, 2005. IEEE, Piscataway, NJ.

K. Alghathbar and H. Mahmoud. Noisy password security technique. In *International Conference for Internet Technology and Secured Transactions (ICITST 2009)*, pages 1–5, November 9–12, 2009, London, UK, 2009. IEEE, Piscataway, NJ.

A. Ali, Wahyudi, and M. J. E. Salami. Keystroke pressure based typing biometrics authentication system by combining ANN and ANFIS-based classifiers. In *5th International Colloquium on Signal Processing and Its Applications (CSPA 2009)*, pages 198–203, March 6–8, 2009, Kuala Lumpur, Malaysia, 2009. IEEE, Piscataway, NJ.

L. C. F. Araújo, L. H. R. Sucupira Jr., M. G. Lizárraga, L. L. Ling, and J. B. T. Yabu-uti. User authentication through typing biometrics features. In *Biometric Authentication (ICBA)*, volume 3071 of *Lecture Notes in Computer Science (LNCS)*, pages 694–700, July 15–17, 2004, Hong Kong, China, 2004. Springer-Verlag, Berlin.

L. C. F. Araújo, L. H. R. Sucupira Jr., M. G. Lizárraga, L. L. Ling, and J. B. T. Yabu-uti. User authentication through typing biometrics features. *IEEE Transactions on Signal Processing*, 53(2):851–855, 2005.

G. L. F. B. G. Azevedo, G. D. C. Cavalcanti, and E. C. B. C. Filho. An approach to feature selection for keystroke dynamics systems based on PSO and feature weighting. In *IEEE Congress on Evolutionary Computation (CEC 2007)*, pages 3577–3584, September 25–28, 2007, 2007a. IEEE, Piscataway, NJ.

G. L. F. B. G. Azevedo, G. D. C. Cavalcanti, and E. C. B. C. Filho. Hybrid solutions for the feature selection in personal identification problems through keystroke dynamics. In *International Joint Conference on Neural Networks (IJCNN 2007)*, pages 1947–1952, August 12–17, 2007, Orlando, FL, 2007b. IEEE, Piscataway, NJ.

R. H. Baayen, D. J. Davidson, and D. M. Bates. Mixed-effects modeling with crossed random effects for subjects and items. *Journal of Memory and Language*, 59(4):390–412, 2008.

N. Bartlow. Username and password verification through keystroke dynamics. Master's thesis, Lane Department of Computer Science and Electrical Engineering, West Virginia University, Morgantown, WV, 2005.

N. Bartlow and B. Cukic. Evaluating the reliability of credential hardening through keystroke dynamics. In *17th IEEE International Symposium on Software Reliability Engineering (ISSRE 2006)*, pages 117–126, November 6–11, 2006, Raleigh, NC, 2006. IEEE Computer Society, Los Alamitos, CA.

D. Bartmann, I. Bakdi, and M. Achatz. On the design of an authentication system based on keystroke dynamics using a predefined input text. *International Journal of Information Security and Privacy*, 1(2):1–12, April–June 2007.

D. Bates. Fitting linear mixed models in R. *R News*, 5(1):27–30, May 2005.

D. Bates. [R] lmer, p-values and all that, 2006. `http://tolstoy.newcastle.edu.au/R/help/06/05/27666.html`, accessed Apr. 2011.

F. Bergadano, D. Gunetti, and C. Picardi. User authentication through keystroke dynamics. *ACM Transactions on Information and System Security (ACM TISSEC)*, 5(4):367–397, November 2002.

F. Bergadano, D. Gunetti, and C. Picardi. Identity verification through dynamic keystroke analysis. *Intelligent Data Analysis*, 7(5):469–496, October 2003.

S. Bleha, C. Slivinsky, and B. Hussien. Computer-access security systems using keystroke dynamics. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(12): 1217–1222, December 1990.

B. M. Bolker, M. E. Brooks, C. J. Clark, S. W. Geange, J. R. Poulsen, M. H. H. Stevens, and J.-S. S. White. Generalized linear mixed models: a practical guide for ecology and evolution. *Trends in Ecology & Evolution*, 24(3):127–135, 2009.

R. M. Bolle, J. H. Connel, S. Pankanti, N. K. Ratha, and A. W. Senior. *Guide to Biometrics*. Springer-Verlag, New York, 2004.

G. E. P. Box, J. S. Hunter, and W. G. Hunter. *Statistics for Experimenters: Design, Innovation, and Discovery*. Wiley, New York, 2nd edition, 2005.

F. Bretz, T. Hothorn, and P. Westfall. *Multiple Comparisons Using R*. CRC Press, Boca Raton, FL, 2011.

M. Brown and S. J. Rogers. User identification via keystroke characteristics of typed names using neural networks. *International Journal of Man-Machine Studies*, 39(6):999–1014, 1993.

M. Brown and S. J. Rogers. A practical approach to user authentication. In *10th Annual Computer Security Applications Conference*, pages 108–116, December 5–9, 1994, Orlando, FL, 1994. IEEE Computer Society, Los Alamitos, CA.

W. L. Bryan and N. Harter. Studies in the physiology and psychology of the telegraphic language. *Psychological Review*, 4(1):27–53, 1897.

W. L. Bryan and N. Harter. Studies on the telegraphic language: The acquisition of a hierarchy of habits. *Psychological Review*, 6(4):345–375, 1899.

J. B. Buckheit and D. L. Donoho. WaveLab and reproducible research. Technical Report 474, Department of Statistics, Stanford University, 1995.

K. P. Burnham and D. R. Anderson. *Model Selection and Multimodel Inference: A Practical Information-Theoretic Approach*. Springer, New York, NY, second edition, 2002.

M. Carlier, A. M. Dumont, and J. Beau. Hand performance of French children on a finger-tapping test in relation to handedness, sex, and age. *Perceptual and Motor Skills*, 76: 931–940, June 1993.

G. Casella and R. L. Berger. *Statistical Inference*. Duxbury, Thomson Learning, 2002.

M. Castejón Limas, J. B. Ordieres Meré, A. González Marcos, F. J. Martínez de Pisón Ascacibar, A. V. Pernía Espinoza, and F. Alba Elías. AMORE: A MORE flexible neural network package, 2010. `http://cran.r-project.org/web/packages/AMORE/index.html`, accessed Oct. 2011.

W. Chang. Improving hidden markov models with a similarity histogram for typing pattern biometrics. In *IEEE International Conference on Information Reuse and Integration (IRI 2005)*, pages 487–493, August 15–17, 2005, Las Vegas, NV, 2005. IEEE, Piscataway, NJ.

Z. Changshui and S. Yanhua. AR model for keystroker verification. In *IEEE International Conference on Systems, Man, and Cybernetics*, volume 4, pages 2887–2890, October 8–11, 2000, Nashville, TN, 2000. IEEE, Piscataway, NJ.

W. Chen and W. Chang. Applying hidden markov models to keystroke pattern analysis for password verification. In *IEEE International Conference on Information Reuse and Integration (IRI 2004)*, pages 467–474, November 8–10, 2004, Las Vegas, NV, 2004. IEEE, Piscataway, NJ.

S. Cho and S. Hwang. Artificial rhythms and cues for keystroke dynamics based authentication. In *Advances in Biometrics (ICB-2006)*, volume 3832 of *Lecture Notes in Computer Science (LNCS)*, pages 626–632, January 5–7, 2006, Hong Kong, China, 2005. Springer-Verlag, Berlin.

S. Cho, C. Han, D. H. Han, and H.-I. Kim. Web-based keystroke dynamics identity verification using neural network. *Journal of Organizational Computing and Electronic Commerce*, 10(4):295–307, 2000.

H. Crawford. Keystroke dynamics: Characteristics and opportunities. In *8th Annual International Conference on Privacy Security and Trust (PST 2010)*, pages 205–212, August 17–19, 2010, Ottawa, Canada, 2010. IEEE, Piscataway, NJ.

M. Curtin, C. Tappert, M. Villani, G. Ngo, J. Simone, H. St. Fort, and S.-H. Cha. Keystroke biometric recognition on long-text input: A feasibility study. In *IAENG International Workshop on Scientific Computing and Computational Statistics (IWSCCS 2006)*, June 20–22, Hong Kong, 2006.

M. Davidian and A. R. Gallant. The nonlinear mixed effects model with a smooth random effects density. *Biometrika*, 80(3):475–488, 1993.

W. G. de Ru and J. H. P. Eloff. Enhanced password authentication through fuzzy logic.

IEEE Expert, 12(6):38–45, November/December 1997.

G. R. Doddington, M. A. Przybocki, A. F. Martin, and D. A. Reynolds. The NIST speaker recognition evaluation – overview, methodology, systems, results, perspective. *Speech Communication*, 31(2–3):225–254, 2000.

Y. Dodge, editor. *Oxford Dictionary of Statistical Terms*. Oxford University, New York, NY, 2003.

P. S. Dowland and S. M. Furnell. A long-term trial of keystroke profiling using digraph, trigraph and keyword latencies. In *Security and Protection in Information Processing Systems: Proceedings of the 19th Annual IFIP Security Conference (IFIP/SEC-2004)*, pages 275–289, August 22–27, 2004, Toulouse, France, 2004. Kluwer Academic Publisher.

H. Dozono and M. Nakakuni. An integration method of multi-modal biometrics using supervised pareto learning self organizing maps. In *International Joint Conference on Neural Networks (IJCNN 2008), part of the IEEE World Congress on Computational Intelligence (WCCI 2008)*, pages 602–606, June 1–8, 2008, Hong Kong, 2008. IEEE, Piscataway, NJ.

R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern Classification*. John Wiley & Sons, Inc., second edition, 2001.

M. El-Abed, R. Giot, B. Hemery, and C. Rosenberger. A study of users' acceptance and satisfaction of biometric systems. In *44th IEEE International Carnahan Conference on Security Technology (ICCST 2010)*, pages 170–178, October 5–8, 2010, San Jose, CA, October 2010. IEEE, Piscataway, NJ.

C. Epp, M. Lippold, and R. L. Mandryk. Identifying emotional states using keystroke dynamics. In *Annual Conference on Human Factors in Computing Systems (CHI 2011)*, pages 715–724, May 7–12, 2011, Vancouver, BC, Canada, 2011. ACM, New York, NY.

C. C. Epp. Identifying emotional states through keystroke dynamics. Master's thesis, University of Saskatchewan, 2010.

D. Evans and S. Stolfo. The science of security. *IEEE Security and Privacy*, 9(3):16–17, May/June 2011.

R. A. J. Everitt and P. W. McOwan. Java-based internet biometric authentication system. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(9):1166–1172, 2003.

J. J. Faraway. *Extending Linear Models with R: Generalized Linear, Mixed Effects and Nonparametric Regression Models*. Chapman & Hall/CRC, Boca Raton, FL, 2006.

R. A. Fisher. *The Design of Experiments*. Hafner Publishing, New York, NY, 5th edition, 1949.

S. Forrest, S. A. Hofmeyr, A. Somayaji, and T. A. Longstaff. A sense of self for unix processes. In *IEEE Symposium on Security and Privacy*, pages 120–128. IEEE Computer Society, Los Alamitos, CA, 1996.

G. Forsen, M. Nelson, and J. Raymond Staron. Personal attributes authentication techniques. Technical Report RADC-TR-77-333, Rome Air Development Center, October 1977.

S. M. Furnell, J. P. Morrissey, P. W. Sanders, and C. T. Stockel. Applications of keystroke analysis for improved login security and continuous user authentication. In S. K. Katsikas and D. Gritzalis, editors, *Information Systems Security: Facing the Information Society of the 21st Century*, pages 283–294. Chapman & Hall, on behalf of the International Federation for Information Processing (IFIP), 1996.

R. S. Gaines, W. Lisowski, S. J. Press, and N. Shapiro. Authentication by keystroke timing: Some preliminary results. Technical Report R-2526-NSF, RAND Corporation, May 1980.

A. Gelman. *Bayesian Data Analysis*. Chapman & Hall/CRC, Boca Raton, FL, second edition, 2004.

R. Giot, M. El-Abed, and C. Rosenberger. Keystroke dynamics with low constraints SVM based passphrase enrollment. In *IEEE Third International Conference on Biometrics: Theory, Applications and Systems (BTAS 2009)*, pages 425–430, September 28–30, 2009, Washington, DC, 2009a. IEEE, Piscataway, NJ.

R. Giot, M. El-Abed, and C. Rosenberger. GREYC keystroke: a benchmark for keystroke dynamics biometric systems. In *IEEE Third International Conference on Biometrics: Theory, Applications and Systems (BTAS 2009)*, pages 419–424, September 28–30, 2009, Washington, DC, 2009b. IEEE, Piscataway, NJ.

R. Giot, M. El-Abed, B. Hemery, and C. Rosenberger. Unconstrained keystroke dynamics authentication with shared secret. *Computers & Security*, 2011. (in press).

M. Gladwell. *Blink: The Power of Thinking without Thinking*. Little, Brown and Company, New York, NY, 2005.

L. Gonick and W. Smith. *The Cartoon Guide to Statistics*. Harper Collins, 1993.

D. Gunetti and C. Picardi. Keystroke analysis of free text. *ACM Transactions on Information and System Security (TISSEC)*, 8(3):312–347, August 2005.

D. Gunetti, C. Picardi, and G. Ruffo. Keystroke analysis of different languages: A case study. In *Advances in Intelligent Data Analysis VI (IDA 2005)*, volume 3646 of *Lecture Notes in Computer Science (LNCS)*, pages 133–144, September 8–10, 2005, Madrid, Spain, 2005a. Springer-Verlag, Berlin.

D. Gunetti, C. Picardi, and G. Ruffo. Dealing with different languages and old profiles in keystroke analysis of free text. In *AI*IA 2005: Advances in Artificial Intelligence*, volume 3673 of *Lecture Notes in Artificial Intelligence (LNAI)*, pages 347–358, September 21–23, 2005, Milan, Italy, 2005b. Springer-Verlag, Berlin.

S. Haider, A. Abbas, and A. K. Zaidi. A multi-technique approach for user identification through keystroke dynamics (SMC 2000). In *IEEE International Conference on Systems, Man and Cybernetics*, volume 2, pages 1336–1341, October 8–11, 2000, Nashville, TN, 2000. IEEE, Piscataway, NJ.

N. Harun, W. L. Woo, and S. S. Dlay. Performance of keystroke biometrics authentication system using artificial neural network (ANN) and distance classifier method. In *International Conference on Computer and Communication Engineering (ICCCE 2010)*, pages 1–6, May 11–13, 2010, Kuala Lumpur, Malaysia, 2010. IEEE, Piscataway, NJ.

T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer Series in Statistics. Springer, New York, NY, 2001.

D. Hosseinzadeh and S. Krishnan. Gaussian mixture modeling of keystroke patterns for biometric applications. *IEEE Transactions on Systems, Man, and Cybernetics—Part C: Applications and Reviews*, 38(6):816–826, November 2008.

D. Hosseinzadeh, S. Krishnan, and A. Khademi. Keystroke identification based on Gaussian mixture models. In *IEEE International Conference on Acoustics, Speech, and Signal Processing*, volume 3, pages 1144–1147, May 14–19, 2006, Toulouse, France, 2006. IEEE, Piscataway, NJ.

T. Hothorn, F. Bretz, and P. Westfall. Simultaneous inference in general parametric models. *Biometrical Journal*, 50(3):346–363, 2008.

S. H. Hurlbert. Pseudoreplication and the design of ecological field experiments. *Ecological Monographs*, 54(2):187–211, June 1984.

B. Hussien, R. McLaren, and S. Bleha. An application of fuzzy algorithms in a computer access security system. *Pattern Recognition Letters*, 9(1):39–43, 1989.

B. Hwang and S. Cho. Characteristics of auto-associative MLP as a novelty detector. In *Proceedings of the IEEE International Joint Conference on Neural Networks*, volume 5, pages 3086–3091, 10–16 July, 1999, Washington, DC, 1999.

S.-s. Hwang, S. Cho, and S. Park. Keystroke dynamics-based authentication for mobile devices. *Computers & Security*, 28(1–2):85–93, February–March 2009a.

S.-s. Hwang, H.-j. Lee, and S. Cho. Improving authentication accuracy using artificial rhythms and cues for keystroke dynamics-based authentication. *Expert Systems with Applications*, 36(7):10649–10656, September 2009b.

IEEE Xplore. IEEE Xplore Digital Library, 2011. `http://ieeexplore.ieee.org/Xplore/guesthome.jsp`, accessed Aug. 2011.

R. Janakiraman and T. Sim. Keystroke dynamics in a general setting. In *Advances in Biometrics (ICB 2007)*, volume 4642 of *Lecture Notes in Computer Science*, pages 584–593, August 27–29, 2007, Seoul, Korea, 2007. Springer-Verlag, Berlin.

C.-H. Jiang, S. Shieh, and J.-C. Liu. Keystroke statistical learning model for web authentication. In *2nd ACM Symposium on Information, Computer and Communications Security (ASIACCS 2007)*, pages 359–361, March 20–22, 2007, Singapore, 2007. ACM, New York, NY.

R. Joyce and G. Gupta. Identity authentication based on keystroke latencies. *Communications of the ACM*, 33(2):168–176, February 1990.

P. Kang and S. Cho. A hybrid novelty score and its use in keystroke dynamics-based user authentication. *Pattern Recognition*, 42(11):3115–3127, November 2009.

P. Kang, S.-s. Hwang, and S. Cho. Continual retraining of keystroke dynamics based authenticator. In *Advances in Biometrics (ICB 2007)*, volume 4642 of *Lecture Notes in Computer Science (LNCS)*, pages 1203–1211, August 27–29, 2007, Seoul, Korea, 2007. Springer-Verlag, Berlin.

A. Karatzoglou, A. Smola, K. Hornik, and A. Zeileis. kernlab – an S4 package for kernel methods in R. *Journal of Statistical Software*, 11(9):1–20, 2004.

M. Karnan, M. Akila, and N. Krishnaraj. Biometric personal authentication using keystroke dynamics: A review. *Applied Soft Computing*, 11(2):1565–1573, March 2011.

K. Killourhy and R. Maxion. Why did my detector do *that?!* Predicting keystroke-dynamics error rates. In *Recent Advances in Intrusion Detection (RAID 2010)*, volume 6307 of *Lecture Notes in Computer Science*, pages 256–276, September 15–17, 2010, Ottawa, Ontario, 2010. Springer-Verlag, Berlin.

K. Killourhy and R. Maxion. Should security researchers experiment more and draw more inferences? In *4th Workshop on Cyber Security Experimentation and Test (CSET-2011)*, August 8, 2011, San Francisco, CA, 2011. The USENIX Association, Berkeley, CA.

K. S. Killourhy and R. A. Maxion. Comparing anomaly-detection algorithms for keystroke dynamics. In *IEEE/IFIP International Conference on Dependable Systems and Networks (DSN-2009)*, pages 125–134, June 29–July 2, 2009, Estoril, Lisbon, Portugal, 2009. IEEE Computer Society, Los Alamitos, CA.

K. Kotani and K. Horii. Evaluation on a keystroke authentication system by keying force incorporated with temporal characteristics of keystroke dynamics. *Behavior & Information Technology*, 24:289–302, 2005.

H.-j. Lee and S. Cho. Retraining a novelty detector with impostor patterns for keystroke dynamics-based authentication. In *Advances in Biometrics (ICB 2006)*, volume 3832 of *Lecture Notes in Computer Science (LNCS)*, pages 633–639, January 5–7, 2006, Hong Kong, China, 2006. Springer-Verlag, Berlin.

H.-j. Lee and S. Cho. Retraining a keystroke dynamics-based authenticator with impostor patterns. *Computers & Security*, 26(4):300–310, June 2007.

D.-T. Lin. Computer-access authentication with neural network based keystroke identity verification. In *International Conference on Neural Networks (ICNN 1997)*, volume 1, pages 174–178, June 9–12, 1997, Houston, TX, 1997. IEEE, Piscataway, NJ.

C. C. Loy, W. K. Lai, and C. P. Lim. Keystroke patterns classification using the ARTMAP-FD neural network. In *3rd International Conference on Intelligent Information Hiding and Multimedia Signal Processing*, volume 1, pages 61–64, November 26–28, 2007, Kaohsiung, Taiwan, 2007. IEEE, Piscataway, NJ.

A. Madansky. *Prescriptions for Working Statisticians (Springer Texts in Statistics)*. Springer-Verlag, NY, 1988.

D. Mahar, R. Napier, M. Wagner, W. Laverty, R. D. Henderson, and M. Hiron. Optimizing digraph-latency based biometric typist verification systems: Inter and intra typist differences in digraph latency distributions. *International Journal of Human-Computer*

Studies, 43(4):579–592, 1995.

S. Mandujano and R. Soto. Deterring password sharing: User authentication via fuzzy c-means clustering applied to keystroke biometric data. In *5th Mexican International Conference on Computer Science*, pages 181–187, September 20–24, Colima, Mexico, 2004. IEEE Computer Society, Los Alamitos, CA.

A. J. Mansfield and J. L. Wayman. Best practices in testing and reporting performance of biometric devices, version 2.01. Technical Report NPL Report CMSC 14/02, Centre for Mathematics and Scientific Computing, National Physical Laboratory, UK, 2002.

R. A. Maxion and K. S. Killourhy. Keystroke biometrics with number-pad input. In *IEEE/IFIP International Conference on Dependable Systems and Networks (DSN-2010)*, pages 201–210, June 28–July 1, 2010, Chicago, IL, 2010. IEEE, Los Alamitos, CA.

R. A. Maxion and T. N. Townsend. Masquerade detection augmented with error analysis. *IEEE Transactions on Reliability, Special Section on Quality/Reliability of Engineering of Information Systems*, 53(1):124–147, 2004.

D. Mayer. *Essential Evidence-Based Medicine*. Cambridge University, UK, 2nd edition, 2010.

C. E. McCulloch, S. R. Searle, and J. M. Neuhaus. *Generalized, Linear, and Mixed Models*. Wiley, Hoboken, NJ, second edition, 2008.

A. J. Menezes, P. C. van Oorschot, and S. A. Vanstone. *Handbook of Applied Cryptography*. CRC Press, Boca Raton, 1997.

A. Mészáros, Z. Bankó, and L. Czúni. Strengthening passwords by keystroke dynamics. In *IEEE International Workshop on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications*, pages 574–577, September 6–8, 2007, Dortmund, Germany, 2007. IEEE, Piscataway, NJ.

Microsoft. Password checker, 2008. `http://www.microsoft.com/protect/yourself/password/checker.mspx`, accessed July 2008; replaced by `https://www.microsoft.com/security/pc-security/password-checker.aspx`, accessed November 2011.

R. G. Miller, Jr. *Simultaneous Statistical Inference*. Springer-Verlag, New York, second edition, 1981.

T. M. Mitchell. *Machine Learning*. McGraw-Hill, Boston, MA, 1997.

S. Modi and S. Elliott. Keystroke dynamics verification using a spontaneously generated password. In *40th IEEE International Carnahan Conference on Security Technology*, pages 116–121, October 16–19, 2006, Lexington, KY, 2006. IEEE, Piscataway, NJ.

F. Monrose, M. K. Reiter, and S. Wetzel. Password hardening based on keystroke dynamics. *International Journal on Information Security*, 1(2):69–83, 2002.

D. S. Moore and G. P. McCabe. *Introduction to the Practice of Statistics*. W. H. Freeman & Co., New York, NY, 3rd edition, 1998.

D. M. Mount and S. Arya. ANN: A library for approximate nearest neighbor searching,

2010. `http://www.cs.umd.edu/~mount/ANN/`, accessed Oct. 2011.

R. Napier, W. Laverty, D. Mahar, R. Henderson, M. Hiron, and M. Wagner. Keyboard user verification: Toward an accurate, efficient, and ecologically valid algorithm. *International Journal on Human-Computer Studies*, 43(2):213–222, 1995.

B. Ngugi, M. Tremaine, and P. Tarasewich. Biometric keypads: Improving accuracy through optimal PIN selection. *Decision Support Systems*, 50(4):769–776, March 2011.

R. E. Nisbett, editor. *Rules for Reasoning*. Lawrence Erlbaum Associates, Hillsdale, NJ, 1993.

M. S. Obaidat and B. Sadoun. Verification of computer users using keystroke dynamics. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, 27(2):261–269, 1997.

R. Okuno, M. Yokoe, K. Akazawa, K. Abe, and S. Sakoda. Finger taps movement acceleration measurement system for quantitative diagnosis of parkinson's disease. In *Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBS 2006)*, pages 6623–6626, August 30–September 3, 2006, New York, NY, 2006. IEEE, Piscataway, NJ.

PC Tools. Security guide for windows—random password generator, 2008. `http://www.pctools.com/guides/password/`, accessed Jul. 2008.

A. Peacock, X. Ke, and M. Wilkerson. Typing patterns: A key to user identification. *IEEE Security and Privacy*, 2(5):40–47, 2004.

J. C. Pinheiro and D. M. Bates. *Mixed-Effects Models in S and S-Plus*. Statistics and Computing Series. Springer-Verlag, New York, NY, 2000.

R. R. Plant and G. Turner. Millisecond precision psychological research in a world of commodity computers: New hardware, new problems? *Behavior Research Methods*, 41 (3):598–614, 2009.

R Development Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2008. `http://www.R-project.org`.

B. Schölkopf, J. C. Platt, J. Shawe-Taylor, A. J. Smola, and R. C. Williamson. Estimating the support of a high-dimensional distribution. *Neural Computation*, 13(7):1443–1471, 2001.

M. Schonlau, W. DuMouchel, W.-H. Ju, A. F. Karr, M. Theus, and Y. Vardi. Computer intrusion: Detecting masquerades. *Statistical Science*, 16(1):58–74, 2001.

ScienceDirect. ScienceDirect, 2011. `http://www.sciencedirect.com`, accessed Aug. 2011.

S. R. Searle, G. Casella, and C. E. McCulloch. *Variance Components*. John Wiley & Sons, Inc., Hoboken, NJ, 2006.

W. R. Shadish, T. D. Cook, and D. T. Campbell. *Experimental and Quasi-Experimental Designs for Generalized Causal Inference*. Houghton Mifflin, Boston, 2002.

R. Sommer and V. Paxson. Outside the closed world: On using machine learning for network intrusion detection. In *2010 IEEE Symposium on Security and Privacy*, pages 305–316, May 16–19, 2010, Berkeley, CA, 2010. IEEE Computer Society, Los Alamitos, CA.

R. J. Spillane. Keyboard apparatus for personal identification. *IBM Technical Disclosure Bulletin*, 17(11):3346, April 1975.

O. Spreen and E. Strauss. *A Compendium of Neuropsychological Tests: Administration, Norms, and Commentary*. Oxford University Press, New York, 2nd edition, 1998.

SpringerLink. SpringerLink, 2011. `http://www.springerlink.com`, accessed Aug. 2011.

S. Stafford and J. Li. Behavior-based worm detectors compared. In *Recent Advances in Intrusion Detection (RAID 2010)*, pages 38–57, September 15–17, 2010, Ottawa, Ontario, Canada, 2010. Springer-Verlag, Berlin.

S. Stolfo, S. M. Bellovin, and D. Evans. Measuring security. *IEEE Security and Privacy*, 9 (3):60–65, May/June 2011.

K.-s. Sung and S. Cho. GA SVM wrapper ensemble for keystroke dynamics authentication. In *Advances in Biometrics (ICB 2006)*, volume 3832 of *Lecture Notes in Computer Science (LNCS)*, pages 654–660, January 5–7, 2006, Hong Kong, China, 2005. Springer-Verlag, Berlin.

C. C. Tappert, S.-H. Cha, M. Villani, and R. S. Zack. A keystroke biometric system for long-text input. *International Journal of Information Security and Privacy*, 4(1):32–60, 2010.

D. Tran, W. Ma, G. Chetty, and D. Sharma. Fuzzy and markov models for keystroke biometrics authentication. In *Proceedings of the 7th WSEAS International Conference on Simulation, Modelling and Optimization*, pages 89–94, Sep 15–17, Beijing, China, 2007. WSEAS Press.

D. Umphress and G. Williams. Identity verification through keyboard characteristics. *International Journal of Man-Machine Studies*, 23(3):263–273, 1985.

W. N. Venables and B. D. Ripley. *Modern Applied Statistics with S-PLUS*. Springer, New York, 2nd edition, 1997.

M. Villani, C. Tappert, G. Ngo, J. Simone, H. St. Fort, and S.-H. Cha. Keystroke biometric recognition studies on long-text input under ideal and application-oriented conditions. In *Conference on Computer Vision and Pattern Recognition Workshop (CVPRW 2006)*, pages 39–46, Jun 17–22, 2006, New York, NY, 2006. IEEE Computer Society Press.

D. D. Wackerly, W. M. III, and R. L. Scheaffer. *Mathematical Statistics with Applications*. Duxbury, Thomson Learning, sixth edition, 2002.

H. M. Wadsworth Jr., K. S. Stephens, and A. B. Godfrey. *Modern Methods for Quality Control and Improvement*. John Wiley & Sons, NY, 1986.

A. Walenstein, D. J. Hefner, and J. Wichers. Header information in malware families and impact on automated classifiers. In *5th International Conference on Malicious and Un-*

wanted Software (MALWARE 2010), pages 15–22, October 19–20, 2010, Nancy, France, 2010. IEEE Computer Society, Los Alamitos, CA.

J. R. Wall and S. R. Millis. Can motor measures tell us if someone is trying? an assessment of sincerity of effort in simulated malingering. *International Journal of Rehabilitation and Health*, 4(1):51–57, January 1998.

L. Wasserman. *All of Statistics: A Concise Course in Statistical Inference*. Springer, 2003.

S. Weisberg. *Applied Linear Regression*. John Wiley & Sons, Hoboken, NJ, 3rd edition, 2005.

F. W. M. H. Wong, A. S. M. Supian, and A. F. Ismail. Enhanced user authentication through typing biometrics with artificial neural networks and k-nearest neighbor algorithm. In *Conference Record of the 35th Asilomar Conference on Signals, Systems, and Computers*, volume 2, pages 911–915. IEEE Computer Society Press, 2001.

N. Yager and T. Dunstone. The biometric menagerie. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(2):220–230, February 2010.

Y. Yang. Can the strengths of AIC and BIC be shared? a conflict between model identification and regression estimation. *Biometrika*, 92(4):937–950, December 2005.

E. Yu and S. Cho. GA-SVM wrapper approach for feature subset selection in keystroke dynamics identity verification. In *Proceedings of the International Joint Conference on Neural Networks (IJCNN)*, volume 3, pages 2253–2257, July 20–24, 2003, Portland, OR, 2003. IEEE, Piscataway, NJ.

R. S. Zack, C. C. Tappert, and S.-H. Cha. Performance of a long-text-input keystroke biometric authentication system using an improved k-nearest-neighbor classification method. In *4th International Conference on Biometrics: Theory, Applications and Systems (BTAS 2010)*, pages 1–6, September 27–29, 2010, Washington, DC, 2010. IEEE, Piscataway, NJ.

A. F. Zuur, E. N. Ieno, N. Walker, A. A. Saveliev, and G. M. Smith. *Mixed Effects Models and Extensions in Ecology with R*. Statistics for Biology and Health. Springer, 2009.

Keystroke-Dynamics Bibliography

A. M. Ahmad and N. N. Abdullah. User authentication via neural network. In *Artificial Intelligence: Methodology, Systems, and Applications (AIMSA 2000)*, volume 1904 of *Lecture Notes in Artificial Intelligence (LNAI)*, pages 310–320, September 20–23, 2000, Varna, Bulgaria, 2000. Springer-Verlag, Berlin.

A. A. E. Ahmed and I. Traore. Anomaly intrusion detection based on biometrics. In *6th Annual IEEE Systems, Man and Cybernetics (SMC) Information Assurance Workshop (IAW 2005)*, pages 452–453, June 15–17, 2005, West Point, NY, 2005. IEEE, Piscataway, NJ.

K. Alghathbar and H. Mahmoud. Noisy password security technique. In *International Conference for Internet Technology and Secured Transactions (ICITST 2009)*, pages 1–5, November 9–12, 2009, London, UK, 2009. IEEE, Piscataway, NJ.

A. Ali, Wahyudi, and M. J. E. Salami. Keystroke pressure based typing biometrics authentication system by combining ANN and ANFIS-based classifiers. In *5th International Colloquium on Signal Processing and Its Applications (CSPA 2009)*, pages 198–203, March 6–8, 2009, Kuala Lumpur, Malaysia, 2009. IEEE, Piscataway, NJ.

A. S. Anagun and I. Cin. A neural network based computer access security system for multiple users. In *Computers & Industrial Engineering*, volume 35, issues 1–2, pages 351–354, March 29–April 1, 1997, Chicago, IL, 1998. Elsevier Science Ltd., Great Britain.

L. C. F. Araújo, L. H. R. Sucupira Jr., M. G. Lizárraga, L. L. Ling, and J. B. T. Yabu-uti. User authentication through typing biometrics features. In *Biometric Authentication (ICBA)*, volume 3071 of *Lecture Notes in Computer Science (LNCS)*, pages 694–700, July 15–17, 2004, Hong Kong, China, 2004. Springer-Verlag, Berlin.

L. C. F. Araújo, L. H. R. Sucupira Jr., M. G. Lizárraga, L. L. Ling, and J. B. T. Yabu-uti. User authentication through typing biometrics features. *IEEE Transactions on Signal Processing*, 53(2):851–855, 2005.

G. L. F. B. G. Azevedo, G. D. C. Cavalcanti, and E. C. B. C. Filho. An approach to feature selection for keystroke dynamics systems based on PSO and feature weighting. In *IEEE Congress on Evolutionary Computation (CEC 2007)*, pages 3577–3584, September 25–28, 2007, 2007a. IEEE, Piscataway, NJ.

G. L. F. B. G. Azevedo, G. D. C. Cavalcanti, and E. C. B. C. Filho. Hybrid solutions for the feature selection in personal identification problems through keystroke dynamics. In *International Joint Conference on Neural Networks (IJCNN 2007)*, pages 1947–1952, August 12–17, 2007, Orlando, FL, 2007b. IEEE, Piscataway, NJ.

N. Bartlow. Username and password verification through keystroke dynamics. Master's thesis, Lane Department of Computer Science and Electrical Engineering, West Virginia University, Morgantown, WV, 2005.

N. Bartlow. *Establishing the Digital Chain of Evidence in Biometric Systems*. PhD thesis, West Virginia University, 2009.

N. Bartlow and B. Cukic. Evaluating the reliability of credential hardening through keystroke dynamics. In *17th IEEE International Symposium on Software Reliability Engineering (ISSRE 2006)*, pages 117–126, November 6–11, 2006, Raleigh, NC, 2006. IEEE Computer Society, Los Alamitos, CA.

N. Bartlow and B. Cukic. Keystroke dynamics-based credential hardening systems. In *Handbook of Remote Biometrics for Surveillance and Security*, Advances in Pattern Recognition, chapter 14, pages 329–347. Springer-Verlag, London, 2009.

D. Bartmann, I. Bakdi, and M. Achatz. On the design of an authentication system based on keystroke dynamics using a predefined input text. *International Journal of Information Security and Privacy*, 1(2):1–12, April–June 2007.

J. Bechtel, G. Serpen, and M. Brown. Passphrase authentication based on typing style through an ART 2 neural network. *International Journal of Computational Intelligence and Applications*, 2(2):131–152, June 2002.

F. Bergadano, D. Gunetti, and C. Picardi. User authentication through keystroke dynamics. *ACM Transactions on Information and System Security (ACM TISSEC)*, 5(4):367–397, November 2002.

S. Bleha, C. Slivinsky, and B. Hussien. Computer-access security systems using keystroke dynamics. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(12): 1217–1222, December 1990.

S. A. Bleha. *Recognition Systems Based on Time Durations Between Terminal Keystrokes*. PhD thesis, Department of Electrical and Computer Engineering, University of Missouri-Columbia, Columbia, MO, 1988.

S. A. Bleha and M. S. Obaidat. Dimensionality reduction and feature extraction applications in identifying computer users. *IEEE Transactions on Systems, Man, and Cybernetics*, 21(2):452–456, 1991.

S. A. Bleha, J. Knopp, and M. S. Obaidat. Performance of the perceptron algorithm for the classification of computer users. In *Proceedings of the ACM/SIGAPP Symposium on Applied Computing: Technological Challenges of the 1990's*, pages 863–866, Mar 1–3, 1992, Kansas City, MO, 1992. ACM, New York, NY.

G. C. Boechat, J. C. Ferreira, and E. C. B. C. Filho. Authentication personal. In *International Conference on Intelligent and Advanced Systems (ICAIS 2007)*, pages 254–256, November 25–28, 2007, Kuala Lumpur, Malaysia, 2007. IEEE, Piscataway, NJ.

M. Brown and S. J. Rogers. User identification via keystroke characteristics of typed names using neural networks. *International Journal of Man-Machine Studies*, 39(6):999–1014, 1993.

M. Brown and S. J. Rogers. A practical approach to user authentication. In *10th Annual Computer Security Applications Conference*, pages 108–116, December 5–9, 1994, Orlando, FL, 1994. IEEE Computer Society, Los Alamitos, CA.

P. Campisi, E. Maiorana, M. L. Bosco, and A. Neri. User authentication using keystroke dynamics for cellular phones. *IET Signal Processing*, 3(4):333–341, July 2009.

W. Chang. Improving hidden markov models with a similarity histogram for typing pattern biometrics. In *IEEE International Conference on Information Reuse and Integration (IRI 2005)*, pages 487–493, August 15–17, 2005, Las Vegas, NV, 2005. IEEE, Piscataway, NJ.

Z. Changshui and S. Yanhua. AR model for keystroker verification. In *IEEE International Conference on Systems, Man, and Cybernetics*, volume 4, pages 2887–2890, October 8–11, 2000, Nashville, TN, 2000. IEEE, Piscataway, NJ.

W. Chen and W. Chang. Applying hidden markov models to keystroke pattern analysis for password verification. In *IEEE International Conference on Information Reuse and Integration (IRI 2004)*, pages 467–474, November 8–10, 2004, Las Vegas, NV, 2004. IEEE, Piscataway, NJ.

Q. Cheng. User habituation in keystroke dynamics based authentication. Master's thesis, West Virginia University, Morgantown, WV, 2007.

S. Cho, C. Han, D. H. Han, and H.-I. Kim. Web-based keystroke dynamics identity verification using neural network. *Journal of Organizational Computing and Electronic Commerce*, 10(4):295–307, 2000.

T.-H. Cho. Pattern classification methods for keystroke analysis. In *SICE-ICASE International Joint Conference*, pages 3812–3816, October 18–21, 2006, Busan, Korea, 2006. ICASE, Bucheon-City, Gyeonggi-Do, Korea.

M. Choras and P. Mroczkowski. Keystroke dynamics for biometrics identification. In *Adaptive and Natural Computing Algorithms (ICANNGA 2007)*, volume 4432 of *Lecture Notes in Computer Science (LNCS)*, pages 424–431, April 11–14, 2007, Warsaw, Poland, 2007. Springer-Verlag, Berlin.

D. Chudá and M. Ďurfina. Multifactor authentication based on keystroke dynamics. In *International Conference on Computer Systems and Technologies and Workshop for PhD Students in Computing (CompSysTech 2009)*, volume 2, pages 1–6, June 18–19, 2009, Ruse, Bulgaria, 2009. ACM, New York, NY.

N. L. Clarke and S. M. Furnell. Advanced user authentication for mobile devices. *Computers & Security*, 26(2):109–119, March 2007.

N. L. Clarke, S. M. Furnell, P. L. Reynolds, and P. M. Rodwell. Advanced subscriber authentication approaches for third generation mobile systems. In *3rd International Conference on 3G Mobile Communication Technologies*, pages 319–323, May 8–10, 2002, London, UK, 2002. Institution of Electrical Engineers, London, UK.

M. Curtin, C. Tappert, M. Villani, G. Ngo, J. Simone, H. St. Fort, and S.-H. Cha. Keystroke biometric recognition on long-text input: A feasibility study. In *IAENG International Workshop on Scientific Computing and Computational Statistics (IWSCCS 2006)*, June 20–22, Hong Kong, 2006.

H. Davoudi and E. Kabir. A new distance measure for free text keystroke authentication. In *14th International CSI Computer Conference (CSICC 2009)*, pages 570–575, October 20–21, 2009, Tehran, Iran, 2009. IEEE, Piscataway, NJ.

H. Davoudi and E. Kabir. Modification of the relative distance for free text keystroke authentication. In *5th International Symposium on Telecommunications (IST 2010)*, pages 547–551, December 4–6, 2010, Tehran, Iran, 2010. IEEE, Piscataway, NJ.

S. T. de Magalhães, K. Revett, and H. M. D. Santos. Password secured sites—stepping forward with keystroke dynamics. In *International Conference on Next Generation Web Services Practices (NWeSP 2005)*, pages 293–298, August 22–26, 2005, Seoul, Korea, 2005. IEEE Computer Society, Los Alamitos, CA.

W. G. de Ru and J. H. P. Eloff. Enhanced password authentication through fuzzy logic. *IEEE Expert*, 12(6):38–45, November/December 1997.

M. K. Dehnavi and N. P. Fard. Presenting a multimodal biometric model for tracking the students in virtual classes. In *3rd World Conference on Educational Sciences*, volume 15 of *Procedia—Social and Behavioral Sciences*. Elsevier, 2011.

S. Douhou and J. R. Magnus. The reliability of user authentication through keystroke dynamics. *Statistica Neerlandica*, 63(4):432–449, November 2009.

H. Dozono and M. Nakakuni. An integration method of multi-modal biometrics using supervised pareto learning self organizing maps. In *International Joint Conference on Neural Networks (IJCNN 2008), part of the IEEE World Congress on Computational Intelligence (WCCI 2008)*, pages 602–606, June 1–8, 2008, Hong Kong, 2008. IEEE, Piscataway, NJ.

H. Dozono, S. Ito, and M. Nakakuni. The authentication system for multi-modal behavior biometrics using concurrent pareto learning SOM. In *Artificial Neural Networks and Machine Learning (ICANN 2011)*, volume 6792 of *Lecture Notes in Computer Science (LNCS)*, pages 197–204, June 14–17, 2011, Espoo, Finland, 2011. Springer-Verlag, Berlin.

M. El-Abed, R. Giot, B. Hemery, and C. Rosenberger. A study of users' acceptance and satisfaction of biometric systems. In *44th IEEE International Carnahan Conference on Security Technology (ICCST 2010)*, pages 170–178, October 5–8, 2010, San Jose, CA, October 2010. IEEE, Piscataway, NJ.

W. E. Eltahir, M. J. E. Salami, A. F. Ismail, and W. K. Lai. Dynamic keystroke analysis using AR model. In *IEEE International Conference on Industrial Technology (ICIT 2004)*, volume 3, pages 1555–1560, December 8–10, 2004, Hammamet, Tunisia, 2004. IEEE, Piscataway, NJ.

C. Epp, M. Lippold, and R. L. Mandryk. Identifying emotional states using keystroke dynamics. In *Annual Conference on Human Factors in Computing Systems (CHI 2011)*, pages 715–724, May 7–12, 2011, Vancouver, BC, Canada, 2011. ACM, New York, NY.

C. C. Epp. Identifying emotional states through keystroke dynamics. Master's thesis, University of Saskatchewan, 2010.

R. A. J. Everitt and P. W. McOwan. Java-based internet biometric authentication system. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(9):1166–1172, 2003.

G. Forsen, M. Nelson, and J. Raymond Staron. Personal attributes authentication techniques. Technical Report RADC-TR-77-333, Rome Air Development Center, October 1977.

R. S. Gaines, W. Lisowski, S. J. Press, and N. Shapiro. Authentication by keystroke timing: Some preliminary results. Technical Report R-2526-NSF, RAND Corporation, May 1980.

H. A. Garcia-Baleon, V. Alarcon-Aquino, and O. Starostenko. K-medoids-based random biometric pattern for cryptographic key generation. In *14th Iberoamerican Conference on Pattern Recognition (CIARP 2009)*, volume 5856 of *Lecture Notes in Computer Science (LNCS)*, pages 85–94, November 15–18, 2009, Guadalajara, Mexico, 2009. Springer-Verlag, Berlin.

R. Giot, M. El-Abed, and C. Rosenberger. Keystroke dynamics with low constraints SVM based passphrase enrollment. In *IEEE Third International Conference on Biometrics: Theory, Applications and Systems (BTAS 2009)*, pages 425–430, September 28–30, 2009, Washington, DC, 2009a. IEEE, Piscataway, NJ.

R. Giot, M. El-Abed, and C. Rosenberger. Keystroke dynamics authentication for collaborative systems. In *International Symposium on Collaborative Technologies and Systems (CTS 2009)*, pages 172–179, May 18–22, 2009, Baltimore, MD, 2009b. IEEE, Piscataway, NJ.

R. Giot, H. Baptiste, and C. Rosenberger. Low cost and usable multimodal biometric system based on keystroke dynamics and 2D face recognition. In *20th International Conference on Pattern Recognition (ICPR 2010)*, pages 1128–1131, August 23–26, 2010, Istanbul, Turkey, 2010. IEEE Computer Society, Los Alamitos, CA.

R. Giot, M. El-Abed, B. Hemery, and C. Rosenberger. Unconstrained keystroke dynamics authentication with shared secret. *Computers & Security*, 2011. (in press).

S. Giroux, R. Wachowiak-Smolikova, and M. P. Wachowiak. Keystroke-based authentication by key press intervals as a complementary behavioral biometric. In *IEEE International Conference on Systems, Man and Cybernetics (SMC 2009)*, pages 80–85, October 11–14, 2009, San Antonio, TX, 2009a. IEEE, Piscataway, NJ.

S. Giroux, R. Wachowiak-Smolikova, and M. P. Wachowiak. Keypress interval timing ratios as behavioral biometrics for authentication in computer security. In *1st International Conference on Networked Digital Technologies (NDT 2009)*, pages 195–200, July 28–31, 2009, Ostrava, The Czech Republic, 2009b. IEEE, Piscataway, NJ.

N. J. Grabham and N. M. White. Use of a novel keypad biometric for enhanced user identity verification. In *International Instrumentation and Measurement Technology Conference (I ^{2}MTC 2008*, pages 12–16, May 12–15, 2008, Victoria, British Columbia, Canada, 2008. IEEE, Piscataway, NJ.

D. Gunetti and C. Picardi. Keystroke analysis of free text. *ACM Transactions on Information and System Security (TISSEC)*, 8(3):312–347, August 2005.

D. Gunetti, C. Picardi, and G. Ruffo. Keystroke analysis of different languages: A case study. In *Advances in Intelligent Data Analysis VI (IDA 2005)*, volume 3646 of *Lecture Notes in Computer Science (LNCS)*, pages 133–144, September 8–10, 2005, Madrid, Spain, 2005a. Springer-Verlag, Berlin.

D. Gunetti, C. Picardi, and G. Ruffo. Dealing with different languages and old profiles in keystroke analysis of free text. In *AI*IA 2005: Advances in Artificial Intelligence*, volume 3673 of *Lecture Notes in Artificial Intelligence (LNAI)*, pages 347–358, September 21–23, 2005, Milan, Italy, 2005b. Springer-Verlag, Berlin.

A. Guven and I. Sogukpinar. Understanding users' keystroke patterns for computer access security. *Computers & Security*, 22(8):695–706, December 2003.

S. Haider, A. Abbas, and A. K. Zaidi. A multi-technique approach for user identification through keystroke dynamics (SMC 2000). In *IEEE International Conference on Systems, Man and Cybernetics*, volume 2, pages 1336–1341, October 8–11, 2000, Nashville, TN, 2000. IEEE, Piscataway, NJ.

N. Harun, S. S. Dlay, and W. L. Woo. Performance of keystroke biometrics authentication system using multilayer perceptron neural network (MLP NN). In *7th International Symposium on Communication Systems, Networks & Digital Signal Processing*, pages 711–714, July 21–23, 2010, Newcastle upon Tyne, UK, 2010a. IEEE, Piscataway, NJ.

N. Harun, W. L. Woo, and S. S. Dlay. Performance of keystroke biometrics authentication system using artificial neural network (ANN) and distance classifier method. In *International Conference on Computer and Communication Engineering (ICCCE 2010)*, pages 1–6, May 11–13, 2010, Kuala Lumpur, Malaysia, 2010b. IEEE, Piscataway, NJ.

S. Hocquet, J.-Y. Ramel, and H. Cardot. Fusion of methods for keystroke dynamics authentication. In *4th IEEE Workshop on Automatic Identification Advanced Technologies (AutoID-2007)*, pages 224–229, Oct 17–18, 2005, Buffalo, NY, 2005. IEEE Computer Society, Los Alamitos, CA.

S. Hocquet, J.-Y. Ramel, and H. Cardot. Estimation of user specific parameters in one-class problems. In *Proceedings of the 18th International Conference on Pattern Recognition (ICPR 2006)*, volume 4, pages 449–452, August 20-24, 2006, Hong Kong China, 2006. IEEE Computer Society, Los Alamitos, CA.

S. Hocquet, J.-Y. Ramel, and H. Cardot. User classification for keystroke dynamics authentication. In *Advances in Biometrics (ICB 2007)*, volume 4642 of *Lecture Notes in Computer Science (LNCS)*, pages 531–539, August 27–29, 2007, Seoul, Korea, 2007. Springer-Verlag, Berlin.

D. Hosseinzadeh and S. Krishnan. Gaussian mixture modeling of keystroke patterns for biometric applications. *IEEE Transactions on Systems, Man, and Cybernetics—Part C: Applications and Reviews*, 38(6):816–826, November 2008.

D. Hosseinzadeh, S. Krishnan, and A. Khademi. Keystroke identification based on Gaussian mixture models. In *IEEE International Conference on Acoustics, Speech, and Signal Processing*, volume 3, pages 1144–1147, May 14–19, 2006, Toulouse, France, 2006. IEEE, Piscataway, NJ.

J. Hu, D. Gingrich, and A. Sentosa. A k-nearest neighbor approach for user authentication through biometric keystroke dynamics. In *IEEE International Conference on Communications (ICC 2008)*, pages 1556–1560, May 19–23, 2008, Beijing, China, 2008. IEEE, Piscataway, NJ.

B. Hussien, R. McLaren, and S. Bleha. An application of fuzzy algorithms in a computer access security system. *Pattern Recognition Letters*, 9(1):39–43, 1989.

S.-s. Hwang, H.-j. Lee, and S. Cho. Improving authentication accuracy of unfamiliar passwords with pauses and cues for keystroke dynamics-based authentication. In *Intelligence and Security Informatics (WISI 2006)*, volume 3917 of *Lecture Notes in Computer Science (LNCS)*, pages 73–78, April 9, 2006, Singapore, 2006. Springer-Verlag, Berlin.

S.-s. Hwang, S. Cho, and S. Park. Keystroke dynamics-based authentication for mobile devices. *Computers & Security*, 28(1–2):85–93, February–March 2009a.

S.-s. Hwang, H.-j. Lee, and S. Cho. Improving authentication accuracy using artificial rhythms and cues for keystroke dynamics-based authentication. *Expert Systems with Applications*, 36(7):10649–10656, September 2009b.

E. S. Imsand and J. A. Hamilton Jr. Impact of daily computer usage on GUI usage analysis. In *4th Annual Conference on Information Security Curriculum Development (InfoSecCD 2007)*, September 28–29, 2007, Kennesaw, GA, 2007. ACM, New York, NY.

R. Janakiraman and T. Sim. Keystroke dynamics in a general setting. In *Advances in Biometrics (ICB 2007)*, volume 4642 of *Lecture Notes in Computer Science*, pages 584–593, August 27–29, 2007, Seoul, Korea, 2007. Springer-Verlag, Berlin.

C.-H. Jiang, S. Shieh, and J.-C. Liu. Keystroke statistical learning model for web authentication. In *2nd ACM Symposium on Information, Computer and Communications Security (ASIACCS 2007)*, pages 359–361, March 20–22, 2007, Singapore, 2007. ACM, New York, NY.

S. S. Joshi and V. V. Phoha. Competition between SOM clusters to model user authentication system in computer networks. In *2nd International Conference on Communication System Software and Middleware (COMSWARE 2007)*, pages 284–291, January 7–12, 2007, Bangalore, India, 2007. IEEE, Piscataway, NJ.

R. Joyce and G. Gupta. Identity authentication based on keystroke latencies. *Communications of the ACM*, 33(2):168–176, February 1990.

P. Kang and S. Cho. A hybrid novelty score and its use in keystroke dynamics-based user authentication. *Pattern Recognition*, 42(11):3115–3127, November 2009.

P. Kang, S.-s. Hwang, and S. Cho. Continual retraining of keystroke dynamics based authenticator. In *Advances in Biometrics (ICB 2007)*, volume 4642 of *Lecture Notes in Computer Science (LNCS)*, pages 1203–1211, August 27–29, 2007, Seoul, Korea, 2007. Springer-Verlag, Berlin.

S. Karatzouni and N. Clarke. Keystroke analysis for thumb-based keyboards on mobile devices. In *New Approaches for Security, Privacy, and Trust in Complex Environments*, volume 232, pages 253–263, May 14–16, 2007, Sandton, South Africa, 2007. Springer, New York, NY.

M. Karnan and M. Akila. Identity authentication based on keystroke dynamics using genetic algorithm and partical swarm optimization. In *2nd IEEE International Conference on Computer Science and Information Technology (ICCSIT 2009)*, pages 203–207, August 8–11, 2009, Beijing, China, 2009. IEEE, Piscataway, NJ.

M. Karnan and M. Akila. Personal authentication based on keystroke dynamics using soft computing techniques. In *2nd International Conference on Communication Software and Networks*, pages 334–338, February 26–28, 2010, 2010. IEEE Computer Society, Los Alamitos, CA.

M. Karnan and N. Krishnaraj. Bio password—keystroke dynamic approach to secure mobile devices. In *IEEE International Conference on Computational Intelligence and Computing Research*, pages 1–4, December 28–29, 2010, Tamilnadu, India, 2010. IEEE, Piscataway, NJ.

H. B. Kekre, V. A. Bharadi, P. Shaktia, V. Shah, and A. A. Ambardekar. Keystroke dynamic analysis using relative entropy & timing sequence euclidean distance. In *International Conference & Workshop on Emerging Trends in Technology (ICWET 2011)*, pages 220–223, February 25–26, 2011, Mumbai, India, 2011. ACM, New York, NY.

K. Killourhy and R. Maxion. The effect of clock resolution on keystroke dynamics. In *Recent Advances in Intrusion Detection (RAID-2008)*, volume 5230 of *Lecture Notes in Computer Science*, pages 331–350, September 15–17, 2008, Boston, MA, 2008. Springer-Verlag, Berlin.

K. Killourhy and R. Maxion. Why did my detector do *that?!* Predicting keystroke-dynamics error rates. In *Recent Advances in Intrusion Detection (RAID 2010)*, volume 6307 of *Lecture Notes in Computer Science*, pages 256–276, September 15–17, 2010, Ottawa, Ontario, 2010. Springer-Verlag, Berlin.

K. S. Killourhy and R. A. Maxion. Comparing anomaly-detection algorithms for keystroke dynamics. In *IEEE/IFIP International Conference on Dependable Systems and Networks (DSN-2009)*, pages 125–134, June 29–July 2, 2009, Estoril, Lisbon, Portugal, 2009. IEEE Computer Society, Los Alamitos, CA.

R. Koch and G. D. Rodosek. User identification in encrypted network communications. In *International Conference on Network and Service Management*, pages 246–249, October 25–29, 2010, Niagara Falls, ON, Canada, 2010. IEEE, Piscataway, NJ.

A. Kolakowska. User authentication based on keystroke dynamics analysis. In *Computer Recognition Systems 4*, volume 95 of *Advances in Intelligent and Soft Computing*, pages 667–675. Springer-Verlag, Berlin, 2011.

K. Kotani and K. Horii. Evaluation on a keystroke authentication system by keying force incorporated with temporal characteristics of keystroke dynamics. *Behavior & Information Technology*, 24:289–302, 2005.

H.-j. Lee and S. Cho. Retraining a novelty detector with impostor patterns for keystroke dynamics-based authentication. In *Advances in Biometrics (ICB 2006)*, volume 3832 of *Lecture Notes in Computer Science (LNCS)*, pages 633–639, January 5–7, 2006, Hong Kong, China, 2006. Springer-Verlag, Berlin.

H.-j. Lee and S. Cho. Retraining a keystroke dynamics-based authenticator with impostor patterns. *Computers & Security*, 26(4):300–310, June 2007.

J.-W. Lee, S.-S. Choi, and B.-R. Moon. An evolutionary keystroke authentication based on ellipsoidal hypothesis space. In *9th Annual Conference on Genetic and Evolutionary Computation (GECCO 2007)*, pages 2090–2097, July 7–11, 2007, London, UK, 2007. ACM, New York, NY.

J. Leggett and G. Williams. Verifying identity via keystroke characteristics. *International Journal of Man-Machine Studies*, 28(1):67–76, January 1988.

J. Leggett, G. Williams, M. Usnick, and M. Longnecker. Dynamic identity verification via keystroke characteristics. *International Journal of Man-Machine Studies*, 35(6):859–870, December 1991.

D.-T. Lin. Computer-access authentication with neural network based keystroke identity verification. In *International Conference on Neural Networks (ICNN 1997)*, volume 1, pages 174–178, June 9–12, 1997, Houston, TX, 1997. IEEE, Piscataway, NJ.

C. C. Loy, W. K. Lai, and C. P. Lim. Keystroke patterns classification using the ARTMAP-FD neural network. In *3rd International Conference on Intelligent Information Hiding and Multimedia Signal Processing*, volume 1, pages 61–64, November 26–28, 2007, Kaohsiung, Taiwan, 2007. IEEE, Piscataway, NJ.

H.-R. Lv and W.-Y. Wang. Biologic verification based on pressure sensor keyboards and classifier fusion techniques. *IEEE Transactions on Consumer Electronics*, 52(3):1059–1063, August 2006.

H.-R. Lv, Z.-L. Lin, W.-J. Yin, and J. Dong. Emotion recognition based on pressure sensor keyboards. In *IEEE International Conference on Multimedia and Expo (ICME 2008)*, pages 1089–1092, June 23–26, 2008, Hannover, Germany, 2008. IEEE, Piscataway, NJ.

D. Mahar, R. Napier, M. Wagner, W. Laverty, R. D. Henderson, and M. Hiron. Optimizing digraph-latency based biometric typist verification systems: Inter and intra typist differences in digraph latency distributions. *International Journal of Human-Computer Studies*, 43(4):579–592, 1995.

E. Maiorana, P. Campisi, N. Gonzáles-Carballo, and A. Neri. Keystroke dynamics authentication for mobile phones. In *ACM Symposium on Applied Computing (SAC 2011)*, pages 21–26, March 21–25, 2011, TaiChung, Taiwan, 2011. ACM, New York, NY.

S. Mandujano and R. Soto. Deterring password sharing: User authentication via fuzzy c-means clustering applied to keystroke biometric data. In *5th Mexican International*

Conference on Computer Science, pages 181–187, September 20–24, Colima, Mexico, 2004. IEEE Computer Society, Los Alamitos, CA.

W. Martono, H. Ali, and M. J. E. Salami. Keystroke pressure-based typing biometrics authentication system using support vector machines. In *Computational Science and Its Applications (ICCSA 2007)*, volume 4706 of *Lecture Notes in Computer Science (LNCS)*, pages 85–93, August 26–29, 2007, Kuala Lumpur, Malaysia, 2007. Springer-Verlag, Berlin.

R. A. Maxion and K. S. Killourhy. Keystroke biometrics with number-pad input. In *IEEE/IFIP International Conference on Dependable Systems and Networks (DSN-2010)*, pages 201–210, June 28–July 1, 2010, Chicago, IL, 2010. IEEE, Los Alamitos, CA.

A. Mészáros, Z. Bankó, and L. Czúni. Strengthening passwords by keystroke dynamics. In *IEEE International Workshop on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications*, pages 574–577, September 6–8, 2007, Dortmund, Germany, 2007. IEEE, Piscataway, NJ.

S. Modi and S. Elliott. Keystroke dynamics verification using a spontaneously generated password. In *40th IEEE International Carnahan Conference on Security Technology*, pages 116–121, October 16–19, 2006, Lexington, KY, 2006. IEEE, Piscataway, NJ.

S. K. Modi. Keystroke dynamics verification using spontaneous password. Master's thesis, Purdue University, 2005.

F. Monrose and A. Rubin. Authentication via keystroke dynamics. In *4th ACM Conference on Computer and Communications Security (CCS 1997)*, pages 48–56, Apr 1–4, 1997, Zurich, Switzerland, 1997. ACM, New York, NY.

F. Monrose and A. D. Rubin. Keystroke dynamics as a biometric for authentication. *Future Generation Computer Systems*, 16(4):351–359, February 2000.

F. Monrose, M. K. Reiter, and S. Wetzel. Password hardening based on keystroke dynamics. In *6th ACM Conference on Computer and Communications Security*, pages 73–82, Nov 1–4, 1999, Singapore, 1999. ACM, New York, NY.

J. Montalvão, C. A. S. Almeida, and E. O. Freire. Equalization of keystroke timing histograms for improved identification performance. In *International Telecommunications Symposium*, pages 560–565, September 3–6, 2006, Fortaleza, Brazil, 2006. IEEE, Piscataway, NJ.

J. R. Montalvão Filho and E. O. Freire. Multimodal biometric fusion—joint typist (keystroke) and speaker verification. In *International Telecommunications Symposium*, pages 609–614, September 3–6, 2006, 2006. IEEE, Piscataway, NJ.

P. Mroczkowski. Identity verification using keyboard statistics. Master's thesis, Linköping University, 2004.

R. Napier, W. Laverty, D. Mahar, R. Henderson, M. Hiron, and M. Wagner. Keyboard user verification: Toward an accurate, efficient, and ecologically valid algorithm. *International Journal on Human-Computer Studies*, 43(2):213–222, 1995.

B. Ngugi, M. Tremaine, and P. Tarasewich. Biometric keypads: Improving accuracy through optimal PIN selection. *Decision Support Systems*, 50(4):769–776, March 2011.

M. S. Obaidat and B. Sadoun. Verification of computer users using keystroke dynamics. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, 27(2):261–269, 1997.

M. S. Obaidat and B. Sadoun. Keystroke dynamics based authentication. In A. Jain, R. Bolle, and S. Pankanti, editors, *Biometrics: Personal Identification in Networked Society*, chapter 10, pages 213–229. Springer, New York, 2002.

A. Ogihara, H. Matsumura, and A. Shiozaki. Biometric verification using keystroke motion and key press timing for ATM user verification. In *Proceedings of the International Symposium on Intelligent Signal Processing and Communication (ISPACS'06)*, pages 223–226, December 12–15, 2006, Tottori, Japan, 2006. IEEE, Piscataway, NJ.

K. Omote and E. Okamoto. User identification system based on biometrics for keystroke. In *Information and Communication Security (ICICS 1999)*, volume 1726 of *Lecture Notes in Computer Science (LNCS)*, pages 216–229, November 9–11, 1999, Sydney, Australia, 1999. Springer-Verlag, Berlin.

S. Park, J. Park, and S. Cho. User authentication based on keystroke analysis of long free texts with a reduced number of features. In *2nd International Conference on Communication Systems, Networks and Applications (ICCSNA 2010)*, pages 433–435, June 29–July 1, 2010, Hong Kong, 2010. IEEE, Piscataway, NJ.

N. Pavaday and K. M. S. Soyjaudah. Investigating performance of neural networks in authentication using keystroke dynamics. In *IEEE AFRICON 2007*, pages 1–8, September 26–28, 2007, Windhoek, South Africa, 2007. IEEE, Piscataway, NJ.

N. Pavaday and K. M. S. Soyjaudah. Comparative study of secret code variants in terms of keystroke dynamics. In *3rd International Conference on Risks and Security of Internet Systems (CRiSIS 2008)*, pages 133–140, October 28–30, 2008, Tozeur, Tunisia, 2008. IEEE, Piscataway, NJ.

G. Z. Pedernera, S. Sznur, G. S. Ovando, S. García, and G. Meschino. Revisiting clustering methods to their application on keystroke dynamics for intruder classification. In *IEEE Workshom on Biometric Measurements and Systems for Security and Medical Applications (BioMS 2010)*, pages 36–40, September 9, 2010, Taranto, Italy, 2010. IEEE, Piscataway, NJ.

K. Revett. A bioinformatics based approach to behavioural biometrics. In *Frontiers in the Convergence of Bioscience and Information Technology*, pages 665–670, October 11–13, 2007, Jeju Island, Korea, 2007. IEEE Computer Society, Los Alamitos, CA.

K. Revett. A bioinformatics based approach to user authentication via keystroke dynamics. *International Journal of Control Automation and Systems*, 7(1):7–15, February 2009.

K. Revett, S. T. de Magalhães, and H. Santos. Data mining a keystroke dynamics based biometrics database using rough sets. In *12th Portuguese Conference on Artificial Intelligence (EPIA 2005)*, pages 188–191, December 5–8, 2005, 2005. IEEE, Piscataway, NJ.

K. Revett, S. T. de Magalhães, and H. M. D. Santos. Enhancing login security through the use of keystroke input dynamics. In *Advances in Biometrics (ICB 2006)*, volume 3832 of *Lecture Notes in Computer Science (LNCS)*, pages 661–667, January 5–7, 2006, Hong Kong, China, 2006. Springer-Verlag, Berlin.

K. Revett, S. T. de Magalhães, and H. M. D. Santos. On the use of rough sets for user authentication via keystroke dynamics. In *Progress in Artificial Intelligence*, volume 4874 of *Lecture Notes in Artificial Intelligence (LNAI)*, pages 145–159, December 3–7, 2007, Guimarães, Portugal, 2007a. Springer-Verlag, Berlin.

K. Revett, F. Gorunescu, M. Gorunescu, M. Ene, S. T. de Magalhães, and H. M. D. Santos. A machine learning approach to keystroke dynamics based user authentication. *International Journal of Electronic Security and Digital Forensics*, 1(1):55–70, 2007b.

J. A. Robinson, V. M. Liang, J. A. M. Chambers, and C. L. MacKenzie. Computer user verification using login string keystroke dynamics. *IEEE Transactions on Systems, Man, and Cybernetics—Part A: Systems and Humans*, 28(2):236–241, 1998.

R. N. Rodrigues, G. F. G. Yared, C. R. d. N. Costa, J. B. T. Yabu-Uti, F. Violaro, and L. L. Ling. Biometric access control through numerical keyboards based on keystroke dynamics. In *Advances in Biometrics (ICB 2006)*, volume 3832 of *Lecture Notes in Computer Science (LNCS)*, pages 640–646, January 5–7, 2006, Hong Kong, China, 2005. Springer-Verlag, Berlin.

M. Rybnik, M. Tabedzki, and K. Saeed. A keystroke dynamics based system for user identification. In *7th Computer Information Systems and Industrial Management Applications (CISIM 2008)*, pages 225–230, June 26–28, 2008, Ostrava, The Czech Republic, 2008. IEEE Computer Society, Los Alamitos, CA.

M. Rybnik, P. Panasiuk, and K. Saeed. User authentication with keystroke dynamics using fixed text. In *International Conference on Biometrics and Kansei Engineering (ICBAKE 2009)*, pages 70–75, June 25–28, 2009, Cieszyn, Poland, 2009. IEEE Computer Society, Los Alamitos, CA.

H. Saevanee and P. Bhatarakosol. User authentication using combination of behavioral biometrics over the touchpad acting like touch screen of mobile device. In *International Conference on Computer and Electrical Engineering (ICCEE 2008)*, pages 82–86, December 20–22, 2008, Phuket, Thailand, 2008. IEEE Computer Society, Los Alamitos, CA.

H. Saevanee and P. Bhattarakosol. Authenticating user using keystroke dynamics and finger pressure. In *6th IEEE Consumer Communications and Networking Conference (CCN 2009)*, pages 1–2, January 10–13, 2009, Las Vegas, Nevada, 2009. IEEE, Piscataway, NJ.

T. Samura and H. Nishimura. Keystroke timing analysis for individual identification in japanees free text typing. In *ICROS-SICE International Joint Conference*, pages 3166–3170, August 18–21, 2009, Fukuoka City, Japan, 2009. Society of Instrument and Control Engineers (SICE), Tokyo, Japan.

Y. Sang, H. Shen, and P. Fan. Novel impostors detection in keystroke dynamics by support vector machine. In *Parallel and Distributed Computing: Applications and Technologies (PDCAT 2004)*, volume 3320 of *Lecture Notes in Computer Science (LNCS)*, pages 37–38, December 8–10, 2004, Singapore, 2005. Springer-Verlag, Berlin.

M. Shahzad, S. Zahid, and M. Farooq. A hybrid GA-PSO fuzzy system for user identification on smart phones. In *Proceedings of the 11th Annual Conference on Genetic and Evolutionary Computing (GECCO-2009)*, pages 1617–1624, Jul 8–12, 2009, Montreal, Quebec, Canada, 2009. ACM Press.

V. Shanmugapriya and G. Padmavathi. Keystroke dynamics authentication using neural network approaches. In *Information and Communication Technologies (ICT 2010)*, volume 101 of *Communications in Computer and Information Science*, pages 686–690, September 7–9, 2010, Kochi, India, 2010. Springer-Verlag, Berlin.

M. Sharif, T. Faiz, and M. Raza. Time signatures—an implementation of keystroke and click patterns for practical and secure authentication. In *3rd International Conference on Digital Information Management (ICDIM 2008)*, pages 559–562, November 13–16, 2008, London, UK, 2008. IEEE, Piscataway, NJ.

Y. Sheng, V. Phoha, and S. Rovnyak. A parallel decision tree-based method for user authentication based on keystroke patterns. *IEEE Transactions on Systems, Man, and Cybernetics – Part B: Cybernetics*, 35(4):826–833, August 2005.

S. J. Shepherd. Continuous authentication by analysis of keyboard typing characteristics. In *European Convention on Security and Detection*, pages 111–114, May 16–18, 1995, 1995. IEE, London.

T. Shimshon, R. Moskovitch, L. Rokach, and Y. Elovici. Continuous verification using keystroke dynamics. In *6th International Conference on Computational Intelligence and Security (CIS 2010)*, pages 411–415, Dec 11–14, 2010, Nanning, China, 2010a. IEEE Computer Society, Los Alamitos, CA.

T. Shimshon, R. Moskovitch, L. Rokach, and Y. Elovici. Clustering di-graphs for continuously verifying users according to their typing patterns. In *IEEE 26th Convention of Electrical and Electronics Engineers in Israel*, pages 445–449, November 17–20, 2010, Eilat, Israel, 2010b. IEEE, Piscataway, NJ.

T. Sim and R. Janakiraman. Are digraphs good for free-text keystroke dynamics? In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1–6, June 17–22, 2007, Minneapolis, MN, 2007. IEEE Computer Society, Los Alamitos, CA.

S. Sinthupinyo, W. Roadrungwasinkul, and C. Chantan. User recognition via keystroke latencies using SOM and backpropagation neural network. In *ICROS-SICE International Joint Conference*, pages 3160–3165, August 18–21, 2009, Fukuoka City, Japan, 2009. SICE, Tokyo, Japan.

D. Stefan and D. Yao. Keystroke-dynamics authentication against synthetic forgeries. In *6th International Conference on Collaborative Computing: Networking, Applications, and Worksharing*, pages 1–8, October 9–12, 2010, Chicago, IL, 2010.

A. Sulong, Wahyudi, and M. U. Siddiqi. Intelligent keystroke pressure-based typing biometrics authentication system using radial basis function network. In *5th International Colloquium on Signal Processing & Its Applications (CSPA 2009)*, pages 151–155, March 6–8, 2009, Kuala Lumpur, Malaysia, 2009. IEEE, Piscataway, NJ.

K.-s. Sung and S. Cho. GA SVM wrapper ensemble for keystroke dynamics authentication. In *Advances in Biometrics (ICB 2006)*, volume 3832 of *Lecture Notes in Computer Science (LNCS)*, pages 654–660, January 5–7, 2006, Hong Kong, China, 2005. Springer-Verlag, Berlin.

M. Tapiador and J. A. Sigüenza. Fuzzy keystroke biometrics on web security. In *IEEE Workshop on Automatic Identification Advanced Technologies (AutoID 1999)*, October 28–29, 1999, Summit, NJ, 1999.

C. C. Tappert, S.-H. Cha, M. Villani, and R. S. Zack. A keystroke biometric system for long-text input. *International Journal of Information Security and Privacy*, 4(1):32–60, 2010.

P. S. Teh, A. B. J. Teoh, T. S. Ong, and H. F. Neo. Statistical fusion approach on keystroke dynamics. In *3rd International IEEE Conference on Signal-Image Technologies and Internet-Based Systems (SITIS 2007)*, pages 918–923, December 16–19, 2007, Jiangong Jinjiang, Shanghai, China, 2007. IEEE, Piscataway, NJ.

P. S. Teh, A. B. J. Teoh, C. Tee, and T. S. Ong. Keystroke dynamics in password authentication enhancement. *Expert Systems with Applications*, 37(12):8618–8627, December 2010.

P. S. Teh, A. B. J. Teoh, C. Tee, and T. S. Ong. A multiple layer fusion approach on keystroke dynamics. *Pattern Analysis & Applications*, 14(1):23–36, February 2011.

D. Tran, W. Ma, G. Chetty, and D. Sharma. Fuzzy and markov models for keystroke biometrics authentication. In *Proceedings of the 7th WSEAS International Conference on Simulation, Modelling and Optimization*, pages 89–94, Sep 15–17, Beijing, China, 2007. WSEAS Press.

C.-w. Tseng, F.-j. Liu, and T.-y. Lin. Design and implementation of a RFID-based authentication system by using keystroke dynamics. In *IEEE International Conference on Systems, Man and Cybernetics (SMC 2010)*, pages 3926–3929, October 10–13, 2010, Istanbul, Turkey, 2010. IEEE, Piscataway, NJ.

D. Umphress and G. Williams. Identity verification through keyboard characteristics. *International Journal of Man-Machine Studies*, 23(3):263–273, 1985.

M. Villani, C. Tappert, G. Ngo, J. Simone, H. St. Fort, and S.-H. Cha. Keystroke biometric recognition studies on long-text input under ideal and application-oriented conditions. In *Conference on Computer Vision and Pattern Recognition Workshop (CVPRW 2006)*, pages 39–46, Jun 17–22, 2006, New York, NY, 2006. IEEE Computer Society Press.

Y. Want, G.-Y. Du, and F.-X. Sun. A model for user authentication based on manner of keystroke and principal component analysis. In *5th International Conference on Machine Learning and Cybernetics*, pages 2788–2792, August 13–16, 2006, Dalian, China, 2006. IEEE, Piscataway, NJ.

N. Yager and T. Dunstone. The biometric menagerie. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(2):220–230, February 2010.

E. Yu and S. Cho. GA-SVM wrapper approach for feature subset selection in keystroke dynamics identity verification. In *Proceedings of the International Joint Conference on Neural Networks (IJCNN)*, volume 3, pages 2253–2257, July 20–24, 2003, Portland, OR, 2003a. IEEE, Piscataway, NJ.

E. Yu and S. Cho. Novelty detection approach for keystroke dynamics identity verification. In *Intelligent Data Engineering and Automated Learning (IDEAL 2003)*, volume 2690 of *Lecture Notes in Computer Science (LNCS)*, pages 1016–1023, March 21–23, 2003, Hong Kong, China, 2003b. Springer-Verlag, Berlin.

E. Yu and S. Cho. Keystroke dynamics identity verification—its problems and practical solutions. *Computers & Security*, 23(5):428–440, January 2004.

R. S. Zack, C. C. Tappert, and S.-H. Cha. Performance of a long-text-input keystroke biometric authentication system using an improved k-nearest-neighbor classification method. In *4th International Conference on Biometrics: Theory, Applications and Systems (BTAS 2010)*, pages 1–6, September 27–29, 2010, Washington, DC, 2010. IEEE, Piscataway, NJ.

S. Zahid, M. Shahzad, S. A. Khayam, and M. Farooq. Keystroke-based user identification on smart phones. In *Recent Advances in Intrusion Detection (RAID 2009)*, volume 5758 of *Lecture Notes in Computer Science (LNCS)*, pages 224–243, September 23–25, 2009, Saint-Malo, France, 2009. Springer-Verlag, Berlin.

Y. Zhang, G. Chang, L. Liu, and J. Jia. Authenticating user's keystroke based on statistical models. In *4th International Conference on Genetic and Evolutionary Computing*, pages 578–581, December 13–15, 2010, Shenzhen, China, 2010. IEEE Computer Society, Los Alamitos, CA.

Y. Zhao. Learning user keystroke patterns for authentication. *Transactions on Engineering, Computing, and Technology*, 14:65–70, 2006.