



DEGREE PROJECT, IN COMPUTER SCIENCE , SECOND LEVEL  
*STOCKHOLM, SWEDEN 2015*

# Behavioral Monitoring on Smartphones for Intrusion Detection in Web Systems

A STUDY OF LIMITATIONS AND  
APPLICATIONS OF TOUCHSCREEN  
BIOMETRICS

ANTON LÖVMAR

KTH ROYAL INSTITUTE OF TECHNOLOGY

SCHOOL OF COMPUTER SCIENCE AND COMMUNICATION (CSC)



**KTH Computer Science  
and Communication**

# **Behavioral Monitoring on Smartphones for Intrusion Detection in Web Systems**

A Study of Limitations and Applications of Touchscreen Biometrics

ANTON LÖVMAR

September 2015

alovmar@kth.se  
Master's Thesis at CSC  
Project Provider: Valtech AB  
Supervisor: Johan Karlander  
Examiner: Johan Håstad

# Abstract

Touchscreen biometrics is the process of measuring user behavior when using a touchscreen, and using this information for authentication. This thesis uses SVM and k-NN classifiers to test the applicability of touchscreen biometrics in a web environment for smartphones. Two new concepts are introduced: model training using the Local Outlier Factor (LOF), as well as building custom models for touch behaviour in the context of individual UI components instead of the whole screen. The lowest error rate achieved was 5.6 % using the k-NN classifier, with a standard deviation of 2.29 %. No real benefit using the LOF algorithm in the way presented in this thesis could be found. It is found that the method of using contextual models yields better performance than looking at the entire screen. Lastly, ideas for using touchscreen biometrics as an intrusion detection system is presented.

# Referat

## Bevakning av användarbeteende på mobila enheter för identifiering av intrång i webbsystem

Pekskärmsbiometri innebär att mäta beteende hos en användare som använder en pekskärm och känna denna baserat på informationen. I detta examensarbete används SVM och k-NN klassifiorare för att testa tillämpligheten av denna typ av biometri i en webbmiljö för smarttelefoner. Två nya koncept introduceras: modellträning med "Local Outlier Factor" samt att bygga modeller för användarinteraktioner med enskilda gränssnittselement istället för skärmen i sin helhet. De bästa resultaten för klassifierarna hade en felfrekvens på 5.6 % med en standardavvikelse på 2.29 %. Ingen fördel med användning av LOF för träning framför slumpmässig träning kunde hittas. Däremot förbättrades resultaten genom att använda kontextuella modeller. Avslutande så presenteras idéer för hur ett system som beskrivet kan användas för att upptäcka intrång i webbsystem.

# Contents

## List of Figures

## List of Tables

<b>1</b>	<b>Introduction</b>	<b>2</b>
1.1	Background . . . . .	2
1.2	This Thesis . . . . .	3
<b>2</b>	<b>Touch Based Identification</b>	<b>6</b>
2.1	Machine Learning and Anomaly Detection . . . . .	6
2.1.1	k-Nearest-Neighbors . . . . .	6
2.1.2	Support Vector Machines . . . . .	8
2.1.3	Backward Elimination . . . . .	14
2.1.4	Feature Scaling . . . . .	15
2.1.5	Local Outlier Factor . . . . .	15
2.2	Behavioral Biometrics . . . . .	17
2.3	JavaScript . . . . .	18
2.4	Personal Touch Patterns . . . . .	21
2.4.1	Strokes . . . . .	21
2.4.2	Screen Taps . . . . .	25
<b>3</b>	<b>Limitations of JavaScript Touch Events</b>	<b>28</b>
3.1	Generation of Touch Events . . . . .	28
3.2	The Android Browser . . . . .	28
3.3	Touchscreen Keyboard . . . . .	29
3.4	Force and Touch Area . . . . .	29
<b>4</b>	<b>Method</b>	<b>30</b>
4.1	Participants . . . . .	30
4.2	Collected Features . . . . .	31
4.3	Collecting Data . . . . .	34
4.3.1	First Phase . . . . .	34
4.3.2	Second Phase . . . . .	35
4.4	Model Training . . . . .	36

4.4.1	Using the Local Outlier Factor . . . . .	36
4.4.2	Using Random Selection . . . . .	37
4.4.3	Contextual Models . . . . .	39
4.5	Evaluation . . . . .	40
4.5.1	Classification . . . . .	41
4.5.2	Evaluation Metric . . . . .	43
<b>5</b>	<b>Results</b>	<b>48</b>
5.1	Selection with LOF . . . . .	48
5.1.1	$k$ -NN . . . . .	48
5.1.2	SVM . . . . .	49
5.2	Contextual models . . . . .	49
5.3	Backward elimination . . . . .	51
5.3.1	LOF Selection . . . . .	51
5.3.2	Random selection . . . . .	52
5.4	Comparison . . . . .	53
<b>6</b>	<b>Discussion</b>	<b>56</b>
6.1	Feature selection . . . . .	56
6.2	Model training . . . . .	57
6.2.1	SVM . . . . .	57
6.2.2	Separate models . . . . .	57
6.3	Results . . . . .	58
6.3.1	Calculation of EER . . . . .	58
6.3.2	Selection with LOF . . . . .	58
6.3.3	Contextual models . . . . .	58
6.3.4	Scaling . . . . .	59
6.3.5	Backward elimination . . . . .	59
6.4	Error sources . . . . .	59
6.4.1	Screen sizes . . . . .	60
6.4.2	Feature selection . . . . .	60
6.4.3	Bug in start-point calculation . . . . .	60
6.4.4	Number of participants . . . . .	61
6.4.5	EER calculation . . . . .	61
6.4.6	Movement . . . . .	62
6.5	Lack of keyboard interaction . . . . .	62
6.6	Security . . . . .	62
6.7	Future work . . . . .	63
6.7.1	Model training . . . . .	63
6.7.2	Backward elimination . . . . .	64
6.7.3	User experience . . . . .	64
6.7.4	SVM . . . . .	64
6.7.5	Contextual models . . . . .	64
6.7.6	Pointer events . . . . .	64

<b>7 Conclusion</b>	<b>66</b>
<b>Bibliography</b>	<b>67</b>
<b>Appendices</b>	<b>70</b>
<b>A Experiment</b>	<b>71</b>
<b>B Experiment participants</b>	<b>75</b>
<b>C Complete results for all configurations</b>	<b>76</b>

## List of Figures

1.1 Change in mobile internet usage in sweden . . . . .	2
2.1 Explanation of $k$ -NN . . . . .	7
2.2 Illustration of hyperplane . . . . .	9
2.3 Data not linearly separable . . . . .	11
2.4 Data for which a linear classifier performs poorly . . . . .	12
2.5 Visualization of $k$ - distance . . . . .	16
2.6 Curvature and first moving direction calculation . . . . .	22
2.7 Touch patterns for different users . . . . .	24
2.8 EER calculation . . . . .	25
4.1 Differentiating Taps and Strokes . . . . .	33
4.2 Client and server architecture for experiment . . . . .	35
4.3 Motivation for LOF usage . . . . .	37
4.4 Contextual Touch Events . . . . .	40
5.1 Contextual models compared to simple models . . . . .	51
5.2 Comparison of results for SVM, $k$ -NN and 1-NN . . . . .	53
5.3 Graph of best EER . . . . .	54
5.4 Graph of best EER with individual user plots . . . . .	55
A.1 Experiment: login screen . . . . .	71
A.2 Experiment: scrolling test . . . . .	72
A.3 Experiment: input and gallery . . . . .	73
A.4 Experiment: final answer . . . . .	74

# List of Tables

4.1	Tuning parameters tested . . . . .	42
5.1	Comparison between results for LOF and random selection for $k$ -NN . .	49
5.2	Comparison between LOF and random selection for SVM . . . . .	49
5.3	Results for using contextual models and simple models . . . . .	50
5.4	Results for backward elimination with LOF selection . . . . .	52
5.5	Results for backward elimination with random selection . . . . .	52
B.1	List of experiment participants . . . . .	75
C.1	Complete list of results . . . . .	76



# Introduction

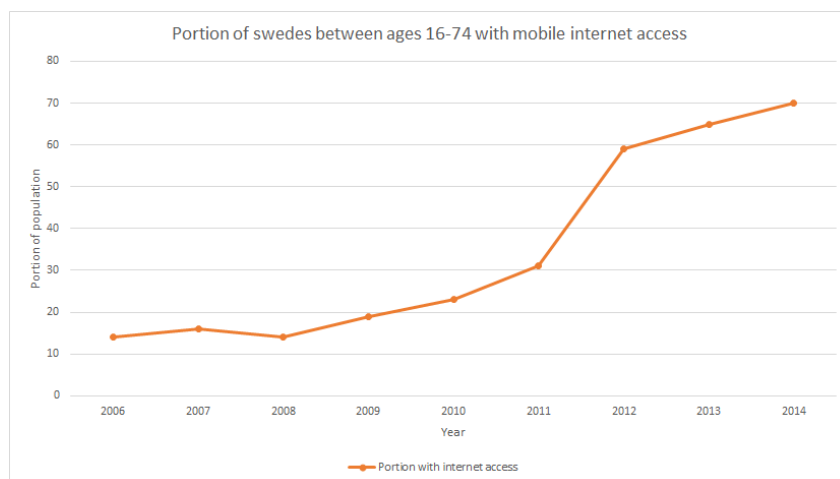
# Chapter 1

## Introduction

In this chapter the motivation behind the thesis is introduced, as well as the goal of the study. Theory, similar studies and related work to that of this thesis are presented in chapter 2.

### 1.1 Background

With the growth of the internet and mobile devices, the ways of accessing banks, social media, news and other common services have changed and moved in part from desktops to the consumers' pockets. Looking at the portion of Swedes with internet access through their mobile phones in the age group 16-74 years, an increase can be seen with 14 % in 2006 to 70 % in 2014 as shown in figure 1.1.



**Figure 1.1.** Change in mobile internet usage between the years 2006-2014 (Data taken from [27])

## 1.2. THIS THESIS

With the rise of the mobile devices comes new ways of authenticating users. Smartphones already incorporate biometric based ways to authenticate users. This includes fingerprint based authentication as well as authentication using facial recognition. On the web today there are several ways of authenticating a user in systems. These range from passwords and electronic identity cards to codes sent through email. However, these methods only prevent unauthorized users from gaining access to a system from the outside. Once inside, identifying an unauthorized user is more difficult. In such a case, an intrusion detection system (IDS) can be used to monitor the system and detect intrusions.

As described by [23, p. 15], intrusion detection is the process of monitoring events in computer systems and applications in order to detect incidents (security breaches). This can be done through analyzing server logs, inspecting packets sent over the network, checking for anomalies in usage, as well as several other methods. [7] presents a way of authenticating a user in the background using keystroke dynamics, that is the rhythm of which a users types on a keyboard. Such an approach can be extended to the virtual on-screen keyboard of smartphones. Building on this concept, biometric data in form of touch gestures, swipes, screen presses (taps) and other events triggered when a user is interacting with the device can be gathered in a non-intrusive and computationally inexpensive way.

## 1.2 This Thesis

Employing touch based IDS in web systems is attractive due to the vast number of services available on the internet, ranging from banks to social networks. Because of the increase in mobile internet usage, it's possible that more of these services will be accessed through mobile touch devices. Therefore new methods of protecting users on the web is crucial. Gesture tracking on websites as opposed to natively on the phone has the benefits of being easy to distribute, as well as users not needing knowledge about the system in the first place (no installation), thus being less intrusive.

In this thesis methods for using touchscreen biometric based intrusion detection in smartphone browsers have been examined. The goal is to identify limitations of using touchscreen biometrics in the browser of a smartphone, as well as proposing new ways of detecting intruders. The browser presents quite a different environment from the one encountered when developing applications for iOS or the Android OS (two of the largest operating systems for mobile devices), since there are several widely spread browsers, all with different implementations of web standards. The focus of this thesis is on how JavaScript can be used to gather touch information in the browser, as well as analytics and classification with machine learning. Two novel concepts are presented: the use of the local outlier factor algorithm ([3]) as a means of selecting observations for training in machine learning, as well as track-

## CHAPTER 1. INTRODUCTION

ing interactions relative individual UI components instead of the entire screen for identifying a user. To evaluate these concept as well the web as an environment for touchscreen biometrics, an experiment was carried out in which users built models portraying their touch behaviour which were used to identify legitimate users as well as detect intruders.

# Theory

## Chapter 2

# Touch Based Identification

In this chapter a basic introduction of machine learning and some related concepts is given in section 2.1, which later tie in to studies presented in section 2.4. Biometrics as well as the difference between behavioral and physiological biometrics is explained in section 2.2. Different metrics for recognizing a user based on touch behaviour are presented in section 2.4, as well as results from related work in touchscreen biometrics. Limitations of JavaScript are highlighted in section 2.3 in the sense of what data can be collected and used for authentication.

### 2.1 Machine Learning and Anomaly Detection

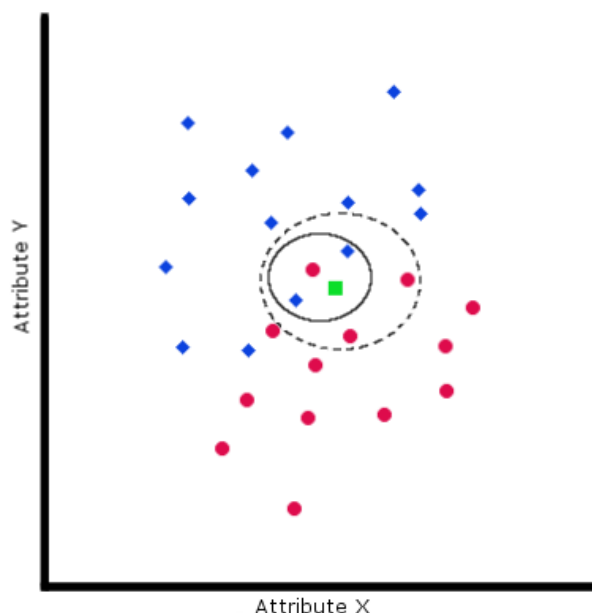
In this section an overview of the  $k$ -Nearest-Neighbors (or  $k$ -NN) and the Support Vector Machine (SVM) algorithms are given as well as the Local outlier factor (LOF). For further reading, see the work of [1], [13] or [3].

#### 2.1.1 $k$ -Nearest-Neighbors

Being one of the simpler machine learning algorithms, the  $k$ -NN algorithm has been used successfully in several other studies of gesture recognition and user identification ([8], [6]). When used for binary classification (two-class), the algorithm classifies an unknown observation based on previous training samples. The classification is simple and intuitive in its nature; for a set of observations, each belonging to some category (or class) and having some features expressed in real numbers, place the observations in a space based on the values of the features. The observations could for example be human beings, and the features be weight and height. Given an unknown observation, the  $k$ -NN algorithm helps determining the class of the observation based on the set of features. Using the same example as before, the classes could be gender (female or male). Classification is done by placing the unknown observation in the same space as the known observations based on its features, and then measuring the distance between the unknown observation and its neighbors. In this thesis, an observation's attributes are assumed to be measured in real numbers.

## 2.1. MACHINE LEARNING AND ANOMALY DETECTION

An unknown observation is classified as the same type as the majority of the  $k$  nearest observations around it. The distance between observations could be the euclidean distance or any other distance function for vectors. This not only makes classification possible but also gives a measure of similarity for the unknown observations, i.e. how similar it is to other observations of the same class. Depending on which value for  $k$  is selected, classification can vary. For example, in figure 2.1 on page 7 the unknown square could be classified as a diamond if a value of 3 for  $k$  is selected. The resulting neighbors are shown by the ellipse with a solid border. A value of 5 for  $k$  would result in a neighborhood marked by the ellipse with a solid border.



**Figure 2.1.** Observations in a space with different attributes as axes. Each observation is placed in the space according to its values for attribute X and Y. The observations' classes are either "circle" or "diamond". The unknown observation discussed in the example is depicted by the square.

The process of collecting the set of known observations is called the training phase (or learning phase). In the  $k$ -NN algorithm, these observations are stored and not processed until a classification is to be done. The learning phase is thus computationally inexpensive compared to the classification, which can consist of comparing an observation to every other observation in the training set ([26]). The training phase could consist of collecting and storing vectors of real numbers representing points in a coordinate system. An implementation of the classification could be to sort these vectors according to the euclidean distance to the observation to be classified, with the vector with shortest distances first. The vectors (observa-

tions) are assumed to be of the form  $x_1 = \begin{pmatrix} x_{11} \\ \vdots \\ x_{1p} \end{pmatrix}, \dots, x_n = \begin{pmatrix} x_{n1} \\ \vdots \\ x_{np} \end{pmatrix}$  where  $x_1, \dots, x_n \in \mathbb{R}^p$ . The number of features (or dimensions) are marked by  $p$ , and each observation also have a class bound to them like  $y_1, \dots, y_n \in \{-1, 1\}$ . Determining the class of the unknown observation is then simply a matter of looking at the first  $k$  observations in the sorted list, classifying the unknown observation as with the same class as the majority of the first  $k$  observations.

### Ensemble Learning $k$ -NN

A problem that arises when using the  $k$ -NN classifier is the selection of the  $k$  parameter. [10] proposes an ensemble learning approach as an alternative to a set value: instead of empirically selecting a value based on earlier experiments or knowledge of the data set, a majority vote from classifications made with several different values of  $k$  are used. The proposed algorithm differs from the standard  $k$ -NN by doing classification for each value between  $1, 3 \dots, \sqrt{k}$ , ignoring even values. Normally using the  $k$ -NN classifier a class would have the same weight as the number of nearest neighbors, that is each observation has the weight 1. In the ensemble classifier the weight is instead calculated using the inverted logarithmic function shown in equation 2.1. In this equation,  $i$  is the order of the neighbor, that is how close it is to the observation to be classified. The third nearest neighbor would have an order of  $i = 3$  and a weight of  $\frac{1}{\log_2(1+3)} = 0.5$ .

$$w(i) = \frac{1}{\log_2(1+i)} \quad (2.1)$$

Setting  $A$  as an array with all observations in the training set sorted with ascending distance to the observation to be classified, and  $n$  the total number of observations in the training set, the weight for each class can be calculated as:

$$WS_c = \sum_{k=1}^{\sqrt{n}} \sum_{i=1}^k \begin{cases} w(i), A_i = c \\ 0, otherwise \end{cases}, k = k + 2 \quad (2.2)$$

In equation 2.2,  $c$  is an integer representing an observations class. The array  $A$  contains integers representing each observations class. The class with the highest weighted sum is the one the observation is classified as ([10]).

### 2.1.2 Support Vector Machines

Just like the  $k$ -NN algorithm, support vector machines can be used to classify an unknown observation based on previous training samples. When presenting the SVM algorithm in this chapter, it will be from the perspective of binary classification, that is to classify observations into either of two classes (or categories). As

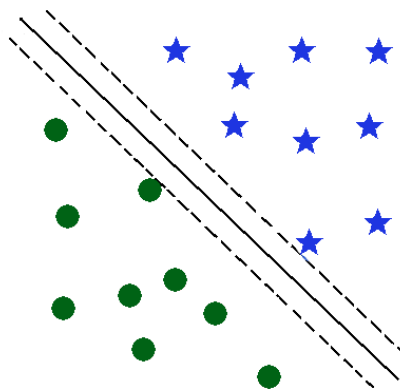


## 2.1. MACHINE LEARNING AND ANOMALY DETECTION

in the case of  $k$ -NN, observations are placed in a space based on some real number features. However, how classification is done differs fundamentally from the  $k$ -NN classifier. The SVM classifier is based on a simpler classifier, the maximal margin classifier, which calculates a hyperplane separating the observations of the different classes.

### Maximal Margin Classifier

In the case of binary classification, that is to determine whether an observation belongs to one or the other of two classes of known observations, one can picture a 2-dimensional space as shown in figure 2.2. It can be used to tell the class of an observation with an unknown class; it depends on which side of the hyperplane the observation appears. Of course, there might not be a single hyperplane which can be used to separate the observations; in such an event the hyperplane with the largest margin to each observation of respective class is chosen ([18, p. 4-6]). A hyperplane in a  $p$ -dimensional space is a subspace of  $p - 1$  dimensions.



**Figure 2.2.** The observations of the two classes (circle and star) are separated by a hyperplane as shown by the black line. The dotted lines represent the margin from the hyperplane to each of the nearest observations of the two classes. In order to yield a good hyperplane, the margin should be as large as possible

Thus in the case of figure 2.2, the space is of 2 dimensions and the hyperplane a 1-dimensional subspace (a line). For perfectly separable data as in figure 2.2, there exists an infinite number of hyperplanes which separates the data. In the maximal margin classifier the hyperplane with the largest margin, that is the largest distance to the nearest observation of each of the two sets, is selected. Calculating this hyperplane is an optimization problem: given a set of training observations

$$x_1 = \begin{pmatrix} x_{11} \\ \vdots \\ x_{1p} \end{pmatrix}, \dots, x_n = \begin{pmatrix} x_{n1} \\ \vdots \\ x_{np} \end{pmatrix} \text{ where } x_1, \dots, x_n \in \mathbb{R}^p \text{ and each observation}$$

belong to a class marked by  $y_1, \dots, y_n \in \{-1, 1\}$ , the task is to find a maximal margin hyperplane separating the data of the two classes. In the general case, a  $p$ -dimensional hyperplane can be defined by equation 2.3 [13].

$$\beta_0 + \beta_1 x_1 + \beta_2 x_2 + \dots + \beta_p x_p = 0 \quad (2.3)$$

If a point  $x = (x_1, \dots, x_p)$  satisfies equation 2.3, it lies on the hyperplane. If the resulting value is instead greater than or lesser than 0, it lies on one of the sides of the hyperplane. In the case of the given set of observations and classes, a separating hyperplane would have the properties shown in formulas 2.4-2.6 [13][p. 339-340].

$$\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \dots + \beta_p x_{ip} > 0 \text{ if } y_i = 1 \quad (2.4)$$

$$\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \dots + \beta_p x_{ip} < 0 \text{ if } y_i = -1 \quad (2.5)$$

$$y_i(\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \dots + \beta_p x_{ip}) > 0 \quad (2.6)$$

Given these properties, the task is to maximize the margin under a set of conditions. The resulting optimization problem is described in equation 2.7-2.9 [13][p. 343].

$$\underset{\beta_0, \beta_1, \dots, \beta_p}{\text{maximize}} \quad M \quad (2.7)$$

$$\text{subject to} \quad \sum_{j=1}^p \beta_j^2 = 1, \quad (2.8)$$

$$y_i(\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \dots + \beta_p x_{ip}) \geq M \quad \forall i = 1, \dots, n \quad (2.9)$$

In the optimization problem,  $p$  is the number of dimensions in the space the hyperplane is to be constructed in, and  $n$  is the total number of observations in the space. Equation 2.8 has the effect that under this constraint the perpendicular distance from the hyperplane to an observation is given by 2.9. Together they make sure that each observation is on the correct side of the hyperplane with at least a distance of  $M$  [13][p. 343].

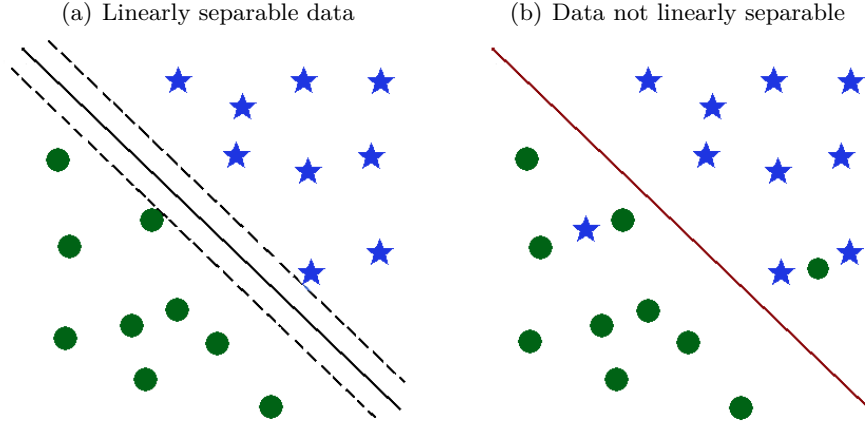
In the previously discussed case, the data points are assumed to be linearly separable. However, this might not be the case for most data sets. In order to apply this method to general data sets without this property, a generalization is needed. This is where the support vector classifier is introduced.

### Support Vector Classifier

The support vector classifier is an extension of the maximum margin classifier. Similarly to the maximum margin classifier, it constructs a hyperplane which separates and classifies data. However, the hyperplane is not required to perfectly separate

## 2.1. MACHINE LEARNING AND ANOMALY DETECTION

the data from the two classes in the training set. The idea is that it might be beneficial to incorrectly classify some observations in order to improve performance, and that a classifier which incorrectly classifies some training observations can still perform well. Figure 2.3 shows that the addition of some observations can cause data to not be linearly separable. However, a hyperplane might still perform reasonably well when used for such a data set as in figure 2.3(a) [13][p. 345]. In order for the



**Figure 2.3.** Figure 2.3(b) shows a space where stars and circles are linearly separable. In figure 2.3(a), two observations have been added to the space which makes it impossible to separate the classes perfectly in the space. The hyperplane used previously can no longer perfectly separate the data. However, if it was to be used for classification, it would still correctly classify most of the training set.

classifier to allow misclassifications, the previous optimization problem is extended with an additional constraint as well as variables  $C$  and  $\epsilon$ . Equation 2.10-2.13 show the new optimization problem [13][p. 346].

$$\underset{\beta_0, \beta_1, \dots, \beta_p, \epsilon_1, \dots, \epsilon_n}{\text{maximize}} \quad M \quad (2.10)$$

$$\text{subject to} \quad \sum_{j=1}^p \beta_j^2 = 1, \quad (2.11)$$

$$y_i(\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \dots + \beta_p x_{ip}) \geq M(1 - \epsilon_i) \forall i = 1, \dots, n, \quad (2.12)$$

$$\epsilon_i \geq 0, \sum_{i=1}^n \epsilon_i \leq C \quad (2.13)$$

The variables  $\epsilon_1, \dots, \epsilon_n$  mark where an observation is located relative to the hyperplane and the margin. If  $\epsilon_i = 0$  observation  $i$  is on the correct side of the margin, if  $\epsilon_i > 0$  the observation is on the wrong side of the margin but still on the correct side

of the hyperplane. If  $\epsilon_i > 1$  the observation is on the wrong side of the hyperplane. Thus from one can see that the parameter  $C$  limits the violations and their severity: as  $C$  increases more and more observations are allowed to be on the incorrect side of the margin or hyperplane. The value of  $C$  affects classification: a small value for  $C$  makes the classifier fit the structure of the data well, but it also becomes sensitive to small fluctuations in data and run the risk of modeling noise in the data (overfitting). On the other hand, if the  $C$  parameter is large the classifier might miss some important relations in the data (underfitting) [13][p. 348].

### Support Vector Machines

For some data sets, separating the observations linearly might yield bad performance even when allowing misclassifications. In figure 2.4, a linear classifier in the 2-dimensional space is will perform poorly when classifying data. To address

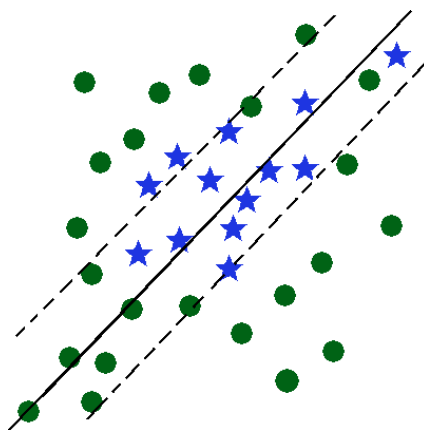


Figure 2.4.

the problem where a linear classifier might not work, features can be mapped to a higher dimension where data might be able to better fit a linear classifier. A support vector classifier using features  $x_1, x_2, \dots, x_n \in \mathbb{R}$  could be mapped to  $x_1, x_1^2, x_2, x_2^2, \dots, x_n^2 \in \mathbb{R}$ . The optimization problem previously described would change as described in equation 2.14-2.17 [13][p. 350].

$$\underset{\beta_0, \beta_{11}, \beta_{12}, \dots, \beta_{p1}, \beta_{p2}, \epsilon_1, \dots, \epsilon_n}{\text{maximize}} \quad M \quad (2.14)$$

$$\text{subject to} \quad \sum_{j=1}^p \sum_{k=1}^2 \beta_{jk}^2 = 1, \quad (2.15)$$

$$y_i(\beta_0 + \sum_{j=1}^p \beta_{j1}x_{ij} + \sum_{j=1}^p \beta_{j2}x_{ij}^2) \geq M(1 - \epsilon_i) \forall i = 1, \dots, n, \quad (2.16)$$

## 2.1. MACHINE LEARNING AND ANOMALY DETECTION

$$\epsilon_i \geq 0, \sum_{i=1}^n \epsilon_i \leq C \quad (2.17)$$

The solution to the modified optimization problem will be linear, however, this only goes for the new space the features are mapped to. In the original space, the solution might be non-linear. As the number of dimensions increase, computation time also increases. This makes it necessary to be careful when doing the feature mapping, or find a way to enlarge the feature space in a way where computations can still be done efficiently [13][p. 350]. The support vector machine is an extension of the support vector classifier for enlarging the feature space and doing calculations in a specific and more efficient way. This is based on the fact that the solution for the optimization problem described in 2.10-2.13 for the support vector classifier only involves inner products of the observations. Thus it can be shown that the support vector classifier can be represented by formula 2.18 where the inner product is  $\langle x, x_i \rangle = \sum_{j=1}^p x_j x_{ij}$ , that is the dot product [13][p. 351].

$$f(x) = \beta_0 + \sum_{i=1}^p \alpha_i \langle x, x_i \rangle \quad (2.18)$$

What makes the support vector machine different from the support vector classifier is that the dot product is replaced with a generalization of the inner product, also called a kernel function. This is shown in formula 2.19 where the kernel is marked by function  $K(x, x_i)$ .

$$f(x) = \beta_0 + \sum_{i=1}^p \alpha_i K(x, x_i) \quad (2.19)$$

If the kernel is the dot product in euclidean space, the classifier will simply be the support vector classifier. However, if another kernel is used like the radial basis function (RBF) kernel in formula 2.20, the space the classifier operates in can be generalized to any number of dimensions. A support vector machine is thus a support vector classifier combined with non-linear kernel [13][p. 352].

$$K(x_i, x_{i'}) = \exp(-\gamma \sum_{j=1}^p (x_{ij} x_{i'j})^2) \quad (2.20)$$

When using kernels like this, one avoids doing computations explicitly in the enlarged feature space and uses the inner products of the observations instead. This is practical since computations explicitly in the enlarged feature space might become intractable or impossible, as in the RBF case. This is due to the feature space of the kernel being an infinite number of dimensions [13][p. 353].

## Selecting Parameters

When using the SVM for classification, some parameters need to be set in order to train a model. The tuning parameter  $C$  is such a parameter, as well as eventual parameters for the kernel. When using the RBF kernel,  $\gamma$  needs to be set as well as  $C$ . One way of selecting parameters is to use a so called grid-search. This search takes some values for  $C$  and  $\gamma$ , for example  $C = 1, 10, 100, 100$  and  $\gamma = 0.001, 0.01, 0.1, 1$  and then tries training the model using every possible pair of  $(C, \gamma)$  ([11][p. 5-8]). The pair which yielded the best performance is selected for final training. This brings the question how performance is evaluated, especially when all available observations are part of the training set. If no large test set is available to estimate performance of the trained model, cross-validation can be used. It is a method for estimating the error when fitting a learning method on a data set of observations, for example when training an SVM model. In cross-validation, the available set to be used for training is partitioned into different smaller sets. Some are used for training, and some used for validation [13][p. 176]. One way of doing this is to use  $k$ -fold cross-validation. In this approach observations are split randomly into  $k$  different sets, or folds. The first set is treated as the validation set, and the rest of the  $k - 1$  sets are used for training. The procedure is repeated  $k$  times where every set is used once as the validation set. This produces  $k$  different estimates of the error, which can be summated by using the average error [13][p. 181]. In binary classification, the error could be calculated as the fraction of incorrect classifications. Another way of doing cross-validation is the leave-one-out cross-validation. This method is similar to  $k$ -fold cross-validation: one observation is used for validation, and all other observations for training. This is repeated until all observations have been used as the training observation. This approach is really a special case of  $k$ -fold cross-validation for when  $k = n$  where  $n$  is the total number of observations [13][p. 179-180].

### 2.1.3 Backward Elimination

When selecting features used for classification for a certain problem, it might be difficult to determine relevant features if one is not familiar with the data set and problem at hand. It might be possible to intuitively select features, however another approach described by [4, p. 11-12] is to use so called wrapper techniques which are guided by the performance of the classifier when selecting features. Backward elimination is one such wrapper where features are deleted one at a time from a pre-existing set of features. If the deletion of a features improves performance of the classifier, this features is kept as deleted, it is otherwise restored to the set of features. This will continue until no feature can be removed in order to increase performance. Performance in this case refers to accuracy of the classifier; higher accuracy means greater performance, and accuracy is measured in the ratio of successful classifications. Backward elimination can be done for the  $k$ -NN classifier after training a model, since model training only consist of collecting observations.

## 2.1. MACHINE LEARNING AND ANOMALY DETECTION

When classifying an observation after training the model, one can skip using a feature when calculating distances to the new observations. For SVM backward elimination cannot be done after the training phase, since training consists of calculating a separating hyperplane. It is not possible to ignore dimensions (features) completely since the hyperplane is calculated for a certain number of features. Feature removal for SVM would have to be done before training.

### 2.1.4 Feature Scaling

Feature scaling is the process of standardizing values of data into a set range. In this thesis work Min-Max scaling as described by [20] is used when scaling is referred to. The reason for using feature scaling is that values of different features can have different metrics or even different magnitude. By using scaling no feature will impact results more than any other, thus preventing results from being skewed by a certain feature. Formula 2.21 shows how a value is scaled. In the formula,  $X$  is the value of the point to be scaled,  $X_{min}$  the minimum value of this feature in the data set and  $X_{max}$  the maximum value.

$$X_{norm} = \frac{X - X_{min}}{X_{max} - X_{min}} \quad (2.21)$$

Using formula 2.21, all features are scaled to a value in the interval  $[0, 1]$ .

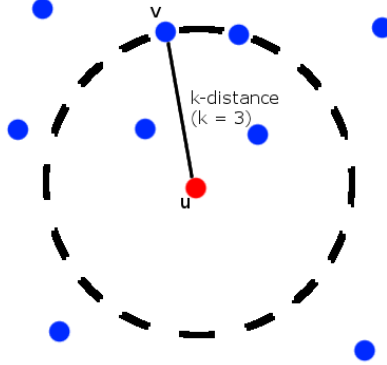
### 2.1.5 Local Outlier Factor

The local outlier factor (LOF) is an algorithm proposed by [3] for identifying data points not located in a cluster of data points in a dataset, i.e. outliers. The algorithm determines the outlierness ([3]) for a point of a multi-dimensional dataset, that is to which degree the point differs from its closest points in the dataset.

*"An outlier is an observation that deviates so much from other observations as to arouse suspicion that it was generated by a different mechanism" - Hawkins outlier ([3])*

What makes the LOF different from other outlier detection methods is that it considers outlier-ness relative to the nearest cluster of points (or neighborhood), as well as the density of that cluster. LOF also views the notion of being an outlier as a non-binary property, i.e. there are different degrees to which a point can be an outlier.

The first step when constructing the local outlier factor algorithm is defining the  $k$ -distance( $u$ ), where  $u$  is a point in a data cluster. This is the distance between  $u$  and its  $k$ -th nearest neighbor. For illustration,  $k$ -distance for a point  $u$  is shown in figure 2.5 on page 16. As seen in the figure, the 3-distance for a point  $u$  is the distance to its third nearest neighbor. In this thesis the distance used when implementing the LOF was the euclidean distance, although another distance function could



**Figure 2.5.** In the figure, the  $k$ -distance for  $k = 3$  is depicted by the dotted circle. That is, the distance to point  $u$ 's third nearest neighbor (point  $v$ ). Note that there are actually four points encompassed by the sphere: this is because of the fact that two points are at the same distance from  $u$

also be used. This means that all observations in the data cluster have attributes with real numbers which are used for calculating distance. Using  $k$ -distance, the  $k$ -distance neighborhood of a point ( $N_k(u)$  for short) can be defined. This set is made up of every point which lies no further than the  $k$ -distance for a certain point, and is defined by formula 2.22. In this formula, the set  $D$  are made up of all data points in the data cluster examined.

$$N_k(u) = \{v' \in D \setminus \{u\} \mid d(v', u) \leq k\text{-distance}(u)\} \quad (2.22)$$

In figure 2.5 the  $k$ -distance neighborhood of point  $u$  consist on all points within or at the edge of the dotted circle. Note that the number of members of the neighborhood can be larger than  $k$  if there are several observations at the line. Another important concept is reachability distance, or  $reach\text{-}dist_k(u, v)$  which further expands on  $k$ -distance. This distance is defined by formula 2.23.

$$reach\text{-}dist(u, v) = \max\{k\text{-distance}(v), d(u, v)\} \quad (2.23)$$

Reachability distance is used to yield more stable results of LOF since it replaces the distance between sufficiently close neighbors with the  $k$ -distance, reducing statistical fluctuations of the the distance ([3]). It is used to define the next formula, local reachability density, which is the final component in calculating the local outlier factor (defined in formula 2.24).

$$lrd(u) = \frac{|N_k(u)|}{\sum_{v \in N_k(u)} reach\text{-}dist_k(u, v)} \quad (2.24)$$



## 2.2. BEHAVIORAL BIOMETRICS

Finally the local reachability density is used for calculating the local outlier factor. This is defined in formula 2.25, and makes use of all the other previous formulas.

$$LOF_k(u) = \frac{\sum_{v \in N_k(u)} \frac{lrd_k(v)}{lrd_k(u)}}{|N_k(u)|} \quad (2.25)$$

An issue that arises for calculating LOF is what value for  $k$  should be used. [3] propose calculating the local outlier factor for a point using each value in a range of values for  $k$ . The highest yielded outlier factor using the values in the range for  $k$  is the one used. The factor calculated by formula 2.25 shows to which degree the point  $u$  is an outlier. With a low value for the local reachability density of  $u$  and a high value the reachability density of its nearest neighbors, the factor will increase. This means that  $u$  will have a high outlier degree if it is located in a low density area of points relative to its nearest neighbors, i.e. "far" out from a local cluster of points.

## 2.2 Behavioral Biometrics

Biometrics are numerous measurements of physical or behavioral traits for humans and can vary a lot in their nature. Metrics such as finger prints, face recognition, retinal scanning, etc. are all physiological biometrics. Keystroke dynamics (typing rhythm for a user on a keyboard) fall under the category behavioral biometrics. Other biometrics of this type include handwriting signatures, voice and walking style. One key difference between behavioral and physiological biometrics are how they vary over time ([28]), as well as the length of the time interval with which they change. For example, a system using face recognition might need regular updates since a users face might change with age; these changes are however slow and don't vary as much from day to day as other biometrics such as voice. The face of a user will probably not change much between two authentication attempts with face recognition, but speaking patterns or tone might differ a lot when a user speaks the same phrase two consecutive times. Since the variability is a lot higher for behavioral biometrics, it might impact accuracy of authentication in a negative way; legitimate users run a higher risk of being rejected because of day-to-day variance. On the other hand, it is easier to update and change a compromised behavioral biometric than a physiological one; changing ones face might be more difficult than speaking a different passphrase.

One biometric closely related to touch biometrics in the sense of how it can be used in digital systems is keystroke dynamics, that is the typing rhythm of a user on a keyboard. [9] presents several algorithms and features for analyzing keystroke dynamics and classifying users. The features for different keys on the keyboard are: hold time, keydown-keydown time and keyup-keydown time. Hold time is the total time a key was pressed, while the other two are the time between key presses. Keydown-keydown time is the time between **pressing** a key and **pressing** the next,

and keyup-keydown time is the time between **releasing** a key and **pressing** the next. These features are in the context of a user typing a certain phrase, but are applicable for typing different phrases and sentences.[9, p. 31] manages an accuracy of at most 73.88 % when using all of the above features for user identification, and even higher accuracy is achieved in studies mentioned by [28] in their survey of keystroke dynamics. As will be further presented in chapter 2, the number of metrics which can be extracted from touchscreen taps and strokes greatly exceeds the number of features in keystroke dynamics. This allows for more complex user models and behavioural patterns, and the idea that similar identification as in keystroke dynamics could be done seems feasible.

## 2.3 JavaScript

JavaScript is the scripting programming language of the web and is used for providing more sophisticated user interaction in web pages, computation and in more recently server-side scripting. The capabilities of JavaScript differ from the operating system it runs on. The difference is introduced by the fact that JavaScript follows the ECMAScript standard and is executed within a browser application's provided host environment. This makes the language capable of whatever features of the standard are implemented in the browser, not the OS. So the functionality of the language is in a sense not OS-dependent but rather browser-dependent. The manufacturer of the browser can of course implement their own functionality and features, but this introduces problems since this manufacturer-specific code wouldn't run in any other browser thus taking away from the benefits of having a standard ([5]). Although the standard implemented in most browsers is the ECMAScript standard, when talking about the client side scripting of browsers the name JavaScript will be used in this thesis.

### Touch Events

Typically in the early years of web browsing with touch devices, taps and touch gestures on the screen were interpreted and converted to mouse clicks in order to maintain compatibility with desktop versions of the same web page. This however imposed restrictions on the user experience since touchscreens are fundamentally different from mouse pointers, and are used in different ways ([25]). The World Wide Web Consortium (W3C) introduced a first proposal for touch events in 2011 ([24]), which has later been revised and changed with the latest version released in 2013.

The W3C specification defines an object where touch information is stored, the "TouchEvent" object. The object has the following fields specified:

- **touches** - List of touch points for every pointer currently active on the screen

### 2.3. JAVASCRIPT

- **targetTouches** - List of touch points for active pointers that started on the targeted element
- **changedTouches** - List of touch points for every point that contributed to fire the event
- **altKey** - True if the alt key modifier is active
- **metaKey** - True if the meta key modifier is active
- **ctrlKey** - True if the ctrl key modifier is active
- **shiftKey** - True if the shift key modifier is active

The TouchEvent object is caught and handled by listeners that listen to certain available touch events. These are:

- **touchstart** - This event is triggered when a user places a pointer on the touchscreen
- **touchend** - This event indicates when a user removes a pointer from the screen, or slides a pointer off the screen
- **touchmove** - This event is triggered when a user moves a pointer across the screen
- **touchcancel** - This event is triggered when an event is canceled because of some other event or a pointer leaves the context area on the page and enters some other area capable of handling touch interactions

All of these events keep the target of their TouchEvent object as the element that the event chain first occurred on. So a *touchend* event for a pointer will have the same target as the touchstart element that started the chain of events.

The TouchEvent object contains three lists of touch objects, each containing a number of Touch objects, each representing a pointer's position in the screen ([25]). The Touch object holds the following information;

- **identifier** - Unique identifier for the pointer the touch object belongs to
- **target** - The target (element) the pointer was first placed on when becoming active on the screen
- **screenX** - x-coordinate relative to the screen
- **screenY** - y-coordinate relative to the screen
- **clientX** - x-coordinate relative to the viewport, excluding scrolling offset
- **clientY** - y-coordinate relative to the viewport, excluding scrolling offset

- **pageX** - x-coordinate relative to the viewport, including scrolling offset
- **pageY** - y-coordinate relative to the viewport, including scrolling offset

The difference between the viewport and screen positions in this list is that the screen positions are the actual positions in pixels on the screen for the touch point. The viewport is the part of the web page that the user can see, so for a web page that doesn't fit the screen of the device used, the pageX and pageY fields can be used to read the position of the touch relative to the "start" of the whole page (upper-left corner). clientX and clientY also retrieves positions in the viewport but does not take scrolling into account ([25]).

### Orientation and Device Motion

In a smartphone browser, not only touch events can be observed using JavaScript. Orientation and device motion events are fired when the device is moved or rotated. These readings are provided by the accelerometer and gyroscope of the device. Motion from the accelerometer are a measure the physical force of acceleration a device is receiving along the  $x$ ,  $y$  and  $z$  axes. The axes refer to the Earth coordinate frame 'East, North, Up'. The ground plane is tangent to the spheroid of the World Geodetic System 1984 ([12]) at a user's location. This is a spheroid representing the body of the earth with the center being the earth's center of mass. The specification from W3C ([29]) states that:

- East (X) is in the ground plane, perpendicular to the North axis and positive towards the East.
- North (Y) is in the ground plane and positive towards True North (towards the North Pole).
- Up (Z) is perpendicular to the ground plane and positive upwards.

The orientation is the rotation in degrees along these axes. All these readings are measured in real numbers, positive or negative depending on which direction along the axis in question rotation or motion is taking place. A device lying on a flat surface without moving and the touchscreen upwards would have an acceleration of 9.81 for  $z$ , and 0 for the other axes. The rotation would be 0 for all axes. For the same device, orientation values would be 0 for all axes, assuming the device is pointing north. The axes  $x$ ,  $y$  and  $z$  are called *beta*, *gamma* and *alpha* in the specification from W3C [29] for orientation events, but in this thesis they will be referred to as  $x$ ,  $y$  and  $z$ .

### Event Frequency

The frequency of with which JavaScript can handle fired events is not specified in the standard by [25]. Tested on both chrome for android and the Android browser,

## 2.4. PERSONAL TOUCH PATTERNS

the frequency of which the *touchmove* event was 60Hz. This was tested for a Samsung SM-G800F smartphone running Android 4.4.2 as well as a Samsung GT-I9195 running Android 4.2.2.

## 2.4 Personal Touch Patterns

After presenting some different machine learning algorithms as well as introducing biometrics, this section presents different metrics for touchscreen gestures as well as their applicability for distinguishing users. There have been several studies concerning the applicability of touch biometrics for user authentication, including [15, 2, 31, 8].

### 2.4.1 Strokes

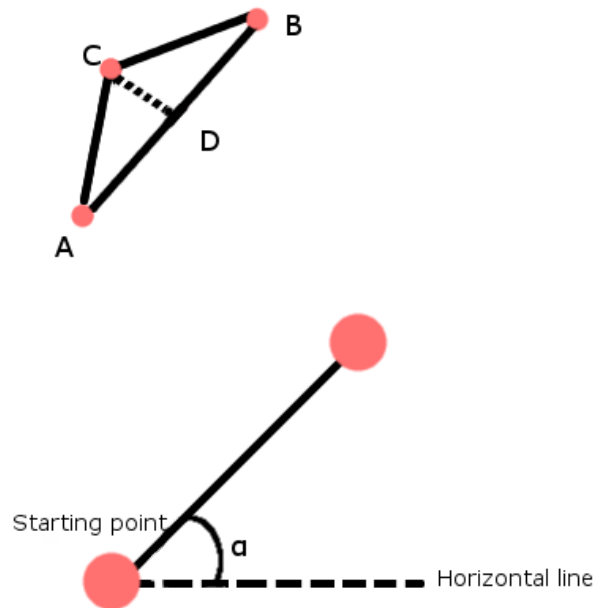
Strokes on the touchscreen are generally more complex than simple taps since they tend to be made up from more touch points on the screen. Since strokes also differ from taps in the sense of movement, features like velocity, angle and curvature are not shared with taps.

#### Features

In their paper *Unobservable Re-authentication for Smartphones*, [16, p. 7-8] proposes features which can be used when looking at strokes on the touchscreen. The features were evaluated and measured on Motorola Droid phones with Android OS version 2.2.2 ([16, p. 2]). The metrics proposed were the following:

- **First touch position** - Coordinates of the starting point for the stroke.
- **First touch pressure** - Screen pressure of the first touch point.
- **First touch area** - Area of the first touch.
- **First moving direction** - The direction of the first stroke. This is calculated by looking at the angle between the first point and the succeeding point of the stroke, as seen in figure 2.6 on page 22.
- **Moving distance** - The total distance of a stroke. This is calculated by summation of the distance between every two continuous points ([16, p. 5]).
- **Duration** - The total stroke duration.
- **Average moving direction** - Average moving direction for the entire stroke, where directions are calculated in the same fashion as the *first moving direction* and done for every two continuous points.

- **Average moving curvature** - As can be seen in figure 2.6 on page 22 (points A, B, C), the curvature is calculated using any three measured points in the stroke and looking at the angle  $\angle ACB$ . This metric refers to the average value of these angles.
- **Average curvature distance** - In the same fashion as the *average moving curvature*, any three points are used but this time the distance of the line AC 2.6 on page 22 is used. The metric is made up of the average distance of all these calculated lines.
- **Average pressure** - The average pressure on the screen of the measured points.
- **Average touch area** - Average area of the points measured
- **Max-area portion** - This metric is calculated by indexing all the measured points between 1 and  $n$  where  $n$  is the total number of points, and then dividing the index of the point with the largest touch area by  $n$ .
- **Min-pressure portion** - Indexed in the same fashion as *max-area portion*, the metric is calculated by dividing the index of the point with the lowest pressure by the total number of points.



**Figure 2.6.** Image depicting how curvature and first moving direction are calculated. The curvature is the angle ACB in the figure, and the first moving direction is the angle  $\alpha$ . If one imagines the figure as the smartphone screen, the horizontal line would be the line drawn horizontally across the screen. All dots in the figure are measured touch points during a stroke gesture.

## 2.4. PERSONAL TOUCH PATTERNS

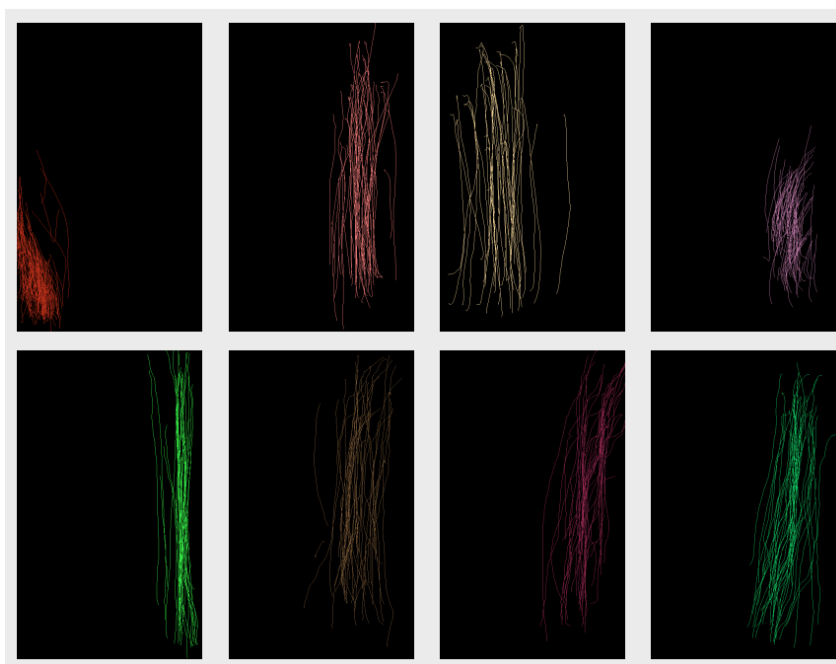
These features are not a complete list of features that can be extracted from strokes. Additional features are proposed in a paper by [8, p. 10], where the applicability of touchscreen input as a behavioral biometric is evaluated. Some additional features proposed in the article are:

- **Mid-stroke finger orientation** - The orientation of the finger at the middle of the stroke (middle measured points).
- **Mid-stroke area covered** - Touch area of the finger at the middle of a stroke. This metric is similar to the *max-area portion* proposed by [16, p. 6], but is measured with the area itself as a metric.
- **Phone orientation** - The orientation of the phone at the time of the gesture. This can either be portrait (phone held upright) or landscape (phone laying on the side).
- **Inter-stroke time** - The time between two consecutive strokes.
- **Deviation from end-to-end line** - How much the line created from the measured points of the stroke deviates from one drawn directly between the first and last points.
- **Last touch position** - The coordinates on the screen at the end point of the stroke.
- **Pairwise velocity** - Average velocity of the stroke.

### Applicability

Although there are several features that can be extracted from strokes, not all of them might be fit for usage in biometric authentication. In their study, [16] evaluated each metric to test whether two data sets from different users were significantly different. For a metric to be classified as good, it had to be usable for distinguishing two users. It was found that all of the proposed gestures having to do with pressure were bad features when used as the single distinguishing feature, as well as the average curvature. By just looking at the orientation, position and direction of strokes it is possible to see quite large variations between users as shown in figure 2.7 on page 24. The figure shows an interesting picture visualizing strokes made from users when reading text (scrolling). When reading a text, the user does not have much else to interact with other than the text area on the device itself (no other interface elements). Thus, the content of the text itself isn't important for distinguishing between users; it is how they scroll down content on a touchscreen device.

[8] found that when looking at a single feature, the most distinct one among the features proposed in the paper was the mid-stroke area covered. This feature was



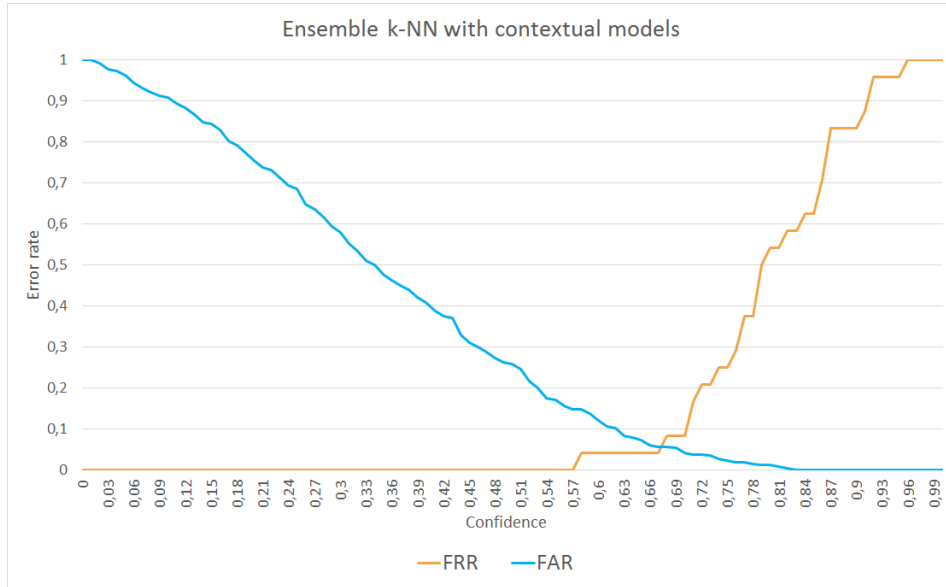
**Figure 2.7.** As demonstrated in their paper, [8] show that strokes can differ quite a lot from user to user. In this picture, strokes made by users reading a text on a touch device are shown. Samples were taken from scrolling through three different pieces of text (image from [8, p. 2], ©2012 IEEE )

tightly followed by the pairwise velocity feature, both which gave the most information about the user. When evaluating the features and classifying users, Support Vector Machines as well as k-Nearest Neighbors algorithms were used. The evaluation features used in the paper were False Acceptance Rate (FAR), False Rejection Rate (FRR) as well as the time to make the first authentication decision in a new user session. In this case, FAR refers to the fraction of impostors that were recognized as a legitimate users, and FRR to legitimate users that were classified as impostors. Final results were a combination of both FAR and FFR, the Equal Error Rate (EER). EER is the error rate where the FAR and FRR are equal. In order to calculate this error rate, some type of confidence threshold is needed for a biometric system. This threshold marks how certain the system has to be of a user's identity in order to classify her as authorized. In figure 2.8 on page 25 one can see how FAR and FRR varies when changing the confidence value. The intersection of the two curves in the figure mark the EER. Confidence can be calculated in several ways depending on the method used for identifying users. For example, when using the  $k$ -NN classifier, confidence could be calculated using the fraction of nearest neighbors of a certain class. This is shown in formula 2.26.

$$confidence_c = \frac{n_c}{k} \quad (2.26)$$



## 2.4. PERSONAL TOUCH PATTERNS



**Figure 2.8.** Graph over how FAR and FRR changes according to the confidence requirement of the system. At confidence 0, FAR is 1 since every user including intruders are accepted, but FRR is 0 since no user is incorrectly rejected. The EER can be found where FAR and FRR are equal, in this case for a confidence of about 0.68

In formula 2.26,  $n_c$  is the number of nearest neighbors of a certain class  $c$ , and  $k$  is the total number of neighbors inspected.

In [8], depending on how many strokes were used in order to classify a user, EER varied. For a single stroke the EER was 13 %, but when using 11 or more strokes the EER converged to between 2 % to 3 %. However, these results were taken from inter-session authentication, which means sessions recorded during the same day. For inter-week authentication, i.e. authentication attempts made a week after the first training phase (for the classifier), results were worse than for inter-session authentication. Looking at the fraction of incorrectly classified strokes, inter-session managed a close to zero fraction while inter-week had an error rate of between 0 % and 4 %.

### 2.4.2 Screen Taps

Taps are in their nature not as complex as strokes, and not as many features can be taken from a tap on the touchscreen. However, one of the earlier studies in the area by 2008 ([15]) achieved an EER of 30 % for detecting anomalies (intruders) using a naive Bayes classifier and only tracking taps. The study was inspired by previous work done for keystroke dynamics and how they could be applied to touchscreen devices instead of keyboards. In this section, different features used for taps in

previous studies are presented as well as their applicability for classification.

### Features

Though simpler in their nature, taps can still have a significant amount of features which can be extracted. They are an important part of navigating on a device using a touchscreen, having a similar function as mouse clicks on desktops, and should not be ignored. In their paper [15, p. 3-4] proposes different metrics that can be used for distinguishing users based on screen taps:

- **Mean hold time** - The average time a user makes contact with the screen (during a tap).
- **Double click frequency** - The frequency of which the user performs double taps.
- **Landing position** - The position of a click performed on a button (position inside the button).
- **Pressure** - The pressure on the screen during a tap.

Another perhaps not so intuitive feature that can be used with taps are acceleration. To be more precise, the accelerometer on the smartphone is read when a tap is performed and the changes are used for classification. As a tap occurs, the phone moves along three axes ( $x,y,z$ ) in the 3D space, which are captured by the accelerometer ([2, p. 5]). These axes refer to ones described in section 2.3.

$$m_{gesture} = \sqrt{m_x^2 + m_y^2 + m_z^2} \quad (2.27)$$

$m_x, m_y, m_z$  is the acceleration (or motion) along the three respective axes. This was also done by [32] for touchscreen keyboards, where the changes on the accelerometer for each key made up a biometric feature. Another feature that was discussed was the area of a tap. This feature is shared between stroke gestures and tap gestures, just like pressure and duration.

Apart from the accelerometer, the gyroscope of the device can also be monitored for changes in rotation caused by a tap. [2, p. 5] uses angular velocity measured with the gyroscope for classification, calculated by formula 2.28.

$$aV_{gesture} = \sqrt{aV_x^2 + aV_y^2 + aV_z^2} \quad (2.28)$$

In formula 2.28,  $aV_x, aV_y, aV_z$  are the rotational velocities along the  $x, y$  and  $z$  axes. These axes are the ones presented in section 2.3. Gyroscope readings only give the current orientation of the device in degrees; the velocity must be calculated from measuring changes in orientation during a touch gesture.

## 2.4. PERSONAL TOUCH PATTERNS

### **Applicability**

[22] studies the applicability of taps as a behavioral biometric for inputting PIN-codes on a touchscreen. Although similar to the works of [15], Saevanee and Bhatarakosol use a  $k$ -NN classifier instead of the naive Bayes classifier and yields a much lower Equal Error Rate (EER). Using pressure, hold-time and inter-key time an EER could be reached of 1%. However, the study also shows that by only using finger pressure an EER of 1% can be reached. This might be an indication that the measured features have a strong correlation since the result is not improved by looking at them in conjunction. It also somewhat contradicts the findings of [16]. However, in their evaluation, pressure was used for strokes as well as taps, and not in the context of only inputting PIN-codes.

The results of the study made by [22] differs significantly from the one presented by [15]. Also, [32] manages an EER of below 5% when only monitoring pressure, size, time and acceleration of taps. Although this study observes a number of taps when inputting specific PIN-codes on a touchscreen, not taps in general. The algorithm used for classification is a not clearly stated, but is presumably a one-class classifier based on the paper. Looking at only tapping behaviour, it would seem it could be used with fair performance as a single biometric for classifying users, if one were to use another classifier.

## Chapter 3

# Limitations of JavaScript Touch Events

In this chapter identified limitations and problems with tracking touch gestures in the browser environment with JavaScript are presented. The impact of these limitations is discussed in chapter 6, and influenced the choice of method in chapter 4.

### 3.1 Generation of Touch Events

As mentioned in section 2.3, the maximum event frequency for JavaScript in android browsers is unspecified. What is clear however is that it drastically changes when dragging over scrollable elements in the web interface using the chrome for android browser. For such an event the frequency of which *touchmove* events are fired drops to around 8 Hz, which is considerably lower than the standard frequency. This also happens when scrolling in the viewport itself, not just individual UI components (lists, text, buttons). This poses a challenge since the number of available touch points for analysis and feature extraction dwindles when performing actions which scroll elements or the viewport, which would make metrics such as the average curvature or deviation from end-to-end line less precise.

A possible explanation for the low number of *touchmove* events fired when scrolling is that the browser, apart from executing the JavaScript code on the page and rendering its contents, needs to recalculate the positions of all elements (which aren't fixed) on the page and move the contents up or down. This might cause a dip in performance for firing the events since the browser needs to make other calculations when scrolling; one can notice that as soon as the page or element has been scrolled to the top or bottom of the page events start firing as normal again.

### 3.2 The Android Browser

Both the stock browser for android version 4.4.2 and 4.2.2 exhibit an unusual behaviour when a user scrolls the content of a page. When a touch event (*touchstart*,

### 3.3. TOUCHSCREEN KEYBOARD

touchmove or touchend) is fired, the *preventDefault* method must be called for the event in order to register it and the next element to be fired. If this method call is not made, the default browser behaviour will override execution and scroll the page or element if possible. However, if the default behaviour is prevented, this will prevent the user from scrolling down the page or any elements thus rendering swiping useless. In other words, in order for the site to function as normal for the android stock browser no tracking can be done for swiping movements. Tapping can still be measured since the touchstart event will be fired before the user agent cancels the event chain and takes over, this however is a significant decrease in the information that can be extracted from the users behaviour. For other smartphone browsers such as chrome for android, the browser will both be able to execute its default behaviour as well as keeping the event chain firing.

## 3.3 Touchscreen Keyboard

Touchscreen smartphones without a hardware keyboard installed features a virtual keyboard which appears as an overlay when a user writes text, types in a passcode or other interactions requiring text input. [32] tries to implement keystroke dynamics using this virtual keyboard instead of traditional hardware keyboards for desktops. This kind of technique becomes problematic when applied to a website and done using JavaScript. After a user has switched context from navigating the web page to typing on the touchscreen keyboard, touch events are no longer being sent in the browser and thus cannot be caught and handled by JavaScript code. This problem doesn't exist for native applications since they use the Android touch API and event listeners.

## 3.4 Force and Touch Area

Although not part of the specification by W3C ([25]), information contained in the JavaScript Touch object for the Android browser as well as chrome for android includes force, radiusX and radiusY. Force in this case is a value between 0.0 and 1.0, where 0.0 is no pressure on the screen and 1.0 is the maximum pressure that can be registered. The area of the touch on the screen is measured by radiusX and radiusY, which are the radii for the x- and y axes of an approximate ellipse representing the area. These values appear to be inconsistent across devices; a Samsung SM-G800F smartphone running Android 4.4.2 returned much higher values (between 0.4 to 0.8 force) for both radius and pressure than a Samsung GT-I9195 running Android 4.2.2 (between 0.1 to 0.4 force).

## Chapter 4

# Method

In order to evaluate JavaScript and the browser environment as a platform for touchscreen biometrics, as well as examine the use of the local outlier factor, a small experiment was carried out in two phases using a system built for classifying users based on touch behaviour. The purpose of the experiment was to see what EER could be achieved for recognizing users based on their touch behaviour under the limitations of JavaScript, as well as how the results differed from previous studies ([6, 31, 22, 2, 16, 8]). Based on the results from previous studies, an EER as low as 10 % was expected from the experiment. Results from the experiment were later used in chapter 7 to determine whether JavaScript and the browser were suitable for touchscreen biometrics. This chapter covers the different phases of the experiment and what data was collected and processed in each one, as well as their purpose. An overview of the system is given as well as a presentation of the different algorithms and technologies used. Section 4.3 describes the gathering of data, and section 4.4 describes how models were trained. Finally section 4.5 presents how the results were produced and evaluated.

### 4.1 Participants

A small group of 18 users were participating in the first phase of the experiment, and was carried out in an open environment where users were able to complete a test on any mobile device of their choice. The participants were aware of their touch data being gathered. However, what wasn't clear to the participants was that the answers to the questions were not important, but the gestures made in order to navigate the page. Exactly what data, when it was collected and how remained unknown. All participants were in ages between 23-28 years old, and their experience with using smartphones was assumed to be good.

## 4.2 Collected Features

This section presents a list of what data was gathered and processed during the data collection phases. In order to train a classifier and use it for classification, a set of observations are needed for which classes are known. These observations are comprised of several features extracted from a data set. Selection of the main features presented in this section was done based on the work of [8, p. 10]. In [8], features were ranked according to their informativeness about a users identity. Analysis was also done on correlation between features. In this thesis work, features were selected based on this ranking of informativeness. Features which had a strong correlation where not selected, since they they might add noise to the classification and not provide more information about user identity. Also, some features like mid-stroke area covered and mid-stroke pressure could not be used when using JavaScript touch events (described in chapter 3) and were thus ignored. First moving direction was selected from the research of [16], and is similar to "average direction" from the work of [8].

In this thesis work the following features were extracted from raw touch data:

- **Duration** - The duration in milliseconds of a stroke or tap.
- **Start position** - The starting position of a stroke or tap. For tracked elements, this is the position including the elements' offset to the viewport. The start position is made up of an  $x$  and  $y$  coordinate on the screen.
- **Starting angle** - This is the angle between the first and fourth measured touch points. If four points are not available, the first and last points are used. The angle is calculated in the same fashion as described in Figure 2.6 on page 22.
- **Total distance** - This is the summated distance between every two consecutive touch points of the gesture.
- **Displacement** - The distance between the start and end positions of a gesture. This calculation is shown in formula 4.1.

$$displacement = \sqrt{(sX - eX)^2 + (sY - eY)^2} \quad (4.1)$$

This is the euclidean distance between the start and end points, where  $sX$  and  $sY$  represent the starting coordinates on the screen. The ending position is represented by  $eX$  and  $eY$ .

- **Device motion** - Readings from the accelerometer of the device. This is calculated as described in algorithm 1 on page 32.
- **Angular motion** - Calculated in the same fashion as the device motion, but using readings from the gyroscope to retrieve angular motion across the three axes.

- **Average velocity** - The average velocity of the gesture, measured in pixel-s/ms and calculated using the total distance and duration of the gesture.
- **End position** - The ending position of a stroke or tap. For tracked elements, this feature is not used due to the fact that the ending position might be outside of the element's boundaries. The end position is made up of an  $x$  and  $y$  coordinate on the screen.

**Algorithm:** Device Motion

**Data:** List of motion objects

**Result:** Summated device motion

startAcceleration = motionObjects.first;

meanDeviationX := 0;

meanDeviationY := 0;

meanDeviationZ := 0;

**for** all motion objects **do**

    meanDeviationX += abs(object.x - startAcceleration.x);

    meanDeviationY += abs(object.y - startAcceleration.y);

    meanDeviationZ += abs(object.z - startAcceleration.z);

**end**

meanDeviationX /= motionObjects.length;

meanDeviationY /= motionObjects.length;

meanDeviationZ /= motionObjects.length;

summatedMotion :=

$\sqrt{\text{meanDeviationX}^2 + \text{meanDeviationY}^2 + \text{meanDeviationZ}^2}$ ;

**return** summatedMotion;

**Algorithm 1:** Summated device motion is calculated by taking the average deviation in acceleration from the first measured motion point. This is done by comparing the difference between the starting motion and the motion for every point which make up the gesture. The values for the  $x$ ,  $y$  and  $z$  axes are then summated using formula 2.27 on page 26

The gesture was also classified as a tap or stroke. Classification of the gestures are based on the article and analysis of [30], where classification is done based on speed and displacement. In figure 4.1, gestures with a displacement and speed (or velocity) that fall under the threshold are classified as taps, and the ones above the threshold as strokes. For strokes, the direction was calculated (upwards or downwards) according to formula 4.2.

$$direction = \begin{cases} 1, & sY - eY > 0 \\ -1, & otherwise \end{cases} \quad (4.2)$$



## 4.2. COLLECTED FEATURES

**Algorithm:** Total distance

**Data:** List of touch objects

**Result:** Total distance

distance := 0;

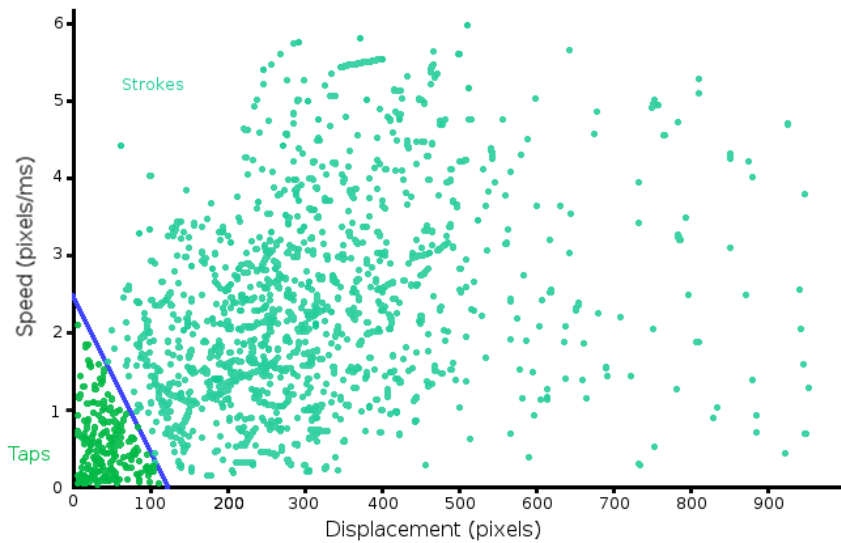
**for**  $i = 1$  **to**  $i = touchObjects.length-1$  **do**

  | distance += euclideanDistance(touchObject[i-1], touchObject[i]);

**end**

**return** distance;

**Algorithm 2:** The total distance of a gesture is the summated distance between every two continuous points that make up the gesture. The distance between two touch points is measured by the euclidean distance, and all touch objects contain information about their screen position in  $x$  and  $y$  coordinates as described in section 2.3



**Figure 4.1.** By plotting the displacement and speed of gestures, it is possible to intuitively separate taps from strokes. [30] show a similar picture of data from taps and strokes, and propose a differentiating line to be drawn between velocity  $2.5px/ms$  and a displacement of  $120px$ . Any gestures falling into the resulting triangle is classified as a tap, and otherwise as a stroke. The data in this figure is randomly generated for illustration purpose.

In formula 4.2,  $sY$  is the  $y$ -coordinate on the screen of the first touch point of the stroke, and  $eY$  the  $y$ -coordinate of the last point. A positive value for direction means the stroke was upwards, and a negative value that the stroke was downwards.

### 4.3 Collecting Data

To collect data through JavaScript events in a browser, a small server was created on which the test site was hosted. Apart from hosting the website used to collect data, the server was also used for analyzing user data and implementing the different algorithms for classifying users. Figure 4.2 gives an overview of the architecture. The server was written in JavaScript running in the Node.js runtime environment, a platform for server-side development. All data collected during both test phases were sent to this server, after it had been processed in the browser. Data was sent as a feature vector containing values for all the features listed in section 4.2. When referring to observations later in this chapter, feature vectors as presented in vector 4.3 are referred to.

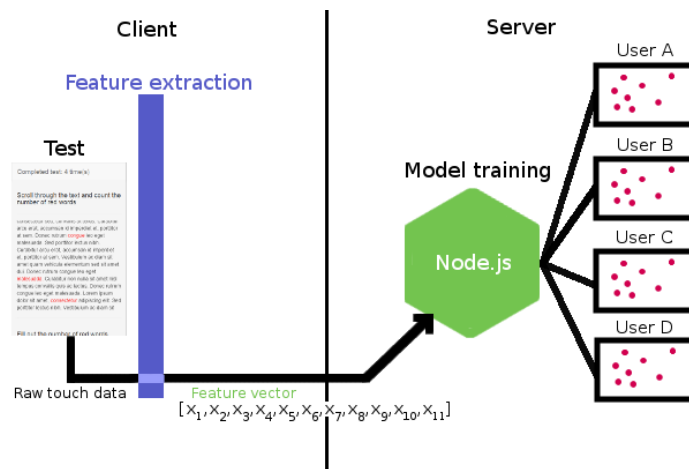
$$\begin{aligned}
 \text{observation} &= [x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8, x_9, x_{10}, x_{11}] \\
 &\text{where} \\
 x_1 &= \text{Duration} \\
 x_2 &= \text{Start x} \\
 x_3 &= \text{Start y} \\
 x_4 &= \text{Starting angle} \\
 x_5 &= \text{End x} \\
 x_6 &= \text{End y} \\
 x_7 &= \text{Total distance} \\
 x_8 &= \text{Displacement} \\
 x_9 &= \text{Device motion} \\
 x_{10} &= \text{Average velocity} \\
 x_{11} &= \text{Angular motion}
 \end{aligned} \tag{4.3}$$

All the values for the different features were real numbers extracted by client side scripts. The architecture of the server is given in figure 4.2. The server contained functions for registering and logging in to a password protected user account, and then building and training a model for the users touch behaviour. This was done in order to separate user profiles and make sure no user accidentally or intentionally sent data to the wrong profile. The accounts also filled a critical role in the second phase of the test, since knowledge of whether the current user was the legitimate user or not was necessary in order to calculate the FRR of the system.

#### 4.3.1 First Phase

The first phase of the experiment to evaluate the system was carried out over the course of three weeks. It consisted of reaching out to a group of testers which performed a test online in their mobile browsers. The test was short and was made up

### 4.3. COLLECTING DATA



**Figure 4.2.** Figure giving an overview of how test data were sent from the test to the sever, and what computation was done at the server/client. Data is collected and processed at the client (website) using the constructed test, features extracted from the raw data and then sent to the server for model training. Data was also sent indicating whether the gesture made was a stroke or tap, as well as the direction of the stroke. The data was sent as a feature vector with real numbers which was stored on the server as part of the model training.

of answering a few questions and scrolling through a web page; a complete walk-through of the test can be found in appendix A.

The purpose of the first phase of the test was to collect a large set of data from different users and train a model with this data for a k-NN classifier. All users were encouraged to complete the test 7 times, with preferably a day between each session. This was to ensure that enough data had been collected, as well as taking day-to-day differences in touch behaviour into account. The profiles built for the different users were used in the second phase of the experiment. Another reason why the test was carried out over time with several sessions is because it was so short, not much data could be collected from a single session. The length was due to the assumption that finding volunteers for a longer test would be harder, as well as the idea that tasks carried out on a real web page might not always consist of much user interaction.

#### 4.3.2 Second Phase

The second phase of the experiment was shorter and lasted for 10 days. During this phase, anyone with access to the server where the test was hosted could participate in the test without signing up for an account or completing the initial model training from the first phase. The purpose of this phase was to collect data from users the system didn't recognize, and use this data to test if the system was capable of classifying these users as intruders when matched against an existing model. During

this phase, the participants of the first phase also participated, although they were required to first sign in to the system. These excessive sessions, not used for model training by the registered users, were saved and used for evaluation. This was done in order for the system to know that the user was a known user, and gather data to test that the system could also recognize legitimate users and correctly accept them.

## 4.4 Model Training

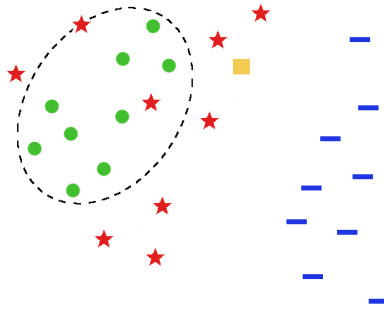
After data collection, the different models only contained data from the respective legitimate users. In order to perform classification, additional data needed to be added in order to have data sets from two different known classes: intruders and legitimate users. When training a model for a user, the "intruder" data was taken from other registered users' data. The data was kept balanced, meaning that the number of intruder observations selected were equal to the number of legitimate user observation already in the model. This prevents the classifier from being overwhelmed by a relatively large number of observations from intruders compared to the legitimate user. The idea behind using other users data instead of generating random dummy intruder data is to try to build a classifier which more closely reflects the behaviour of real potential intruders. However, this poses a problem with which observations to select from other users to act as intruder data. This selection was done in two different ways, as presented in section 4.4.1 and section 4.4.2.

### 4.4.1 Using the Local Outlier Factor

One way of selecting which observations to use as intruder data is the using the local outlier factor algorithm. This selection is done using the local outlier factor in the following manner: In Figure 4.3, the circles marked by the dotted ellipse represents the user whose model is being built. The stars as well as lines represents data sets from other users, and the square a new observation which is to be classified. By ordering the other users' data in a list according to each points local outlier factor value relative to the circle cluster, a selection is made from the middle values of the sorted list. This results in extreme outliers *or* inliers not being used for building the model. By not using points inside the cluster, the classifier won't get confused by intruder values too similar to the legitimate user. When classifying the square, while clearly being an outlier relative to the circles and thus a potential intruder, the classification is highly dependent on the number of points selected from the stars and lines data sets. If a majority of points are selected from the set of lines, the square might be classified as legitimate even though it is made by an intruder.

The  $k$  value for local outlier factor is not set and can be chosen freely, although a recommendation by [3] states that it should be set to a maximum and minimum value. One could see the minimum value of  $k$  as the smallest size of what should be considered a cluster in the data set, and the maximum as the largest. When

#### 4.4. MODEL TRAINING



**Figure 4.3.** Showing hypothetical data from a legitimate user as well as intruders, in this case being other users whose touch data has been recorded. The circle set is from a legitimate user, and the stars and lines from different intruders. Depending on which data points are chosen among the intruders, the square might be classified as an intruder or legitimate user. To make a selection for model training, LOF was used.

building models using the local outlier factor in this thesis work, a value of  $k = \frac{N}{2}$  was chosen, where  $N$  is the total number of points. The reason why a range for  $k$  in the local outlier factor algorithm wasn't used as described in section 2.1.5 on page 15 was because of performance issues. For building a single model for a user using a set value for  $k$ , the local outlier factor would have to be calculated for every observation for every other user in the system (of the same type). If a range for  $k$  was used, this would also have had to be done for every value in the range. A value of  $k = \frac{N}{2}$  meant that the cluster size an intruder observation would be compared against was half of the total observations for the legitimate user. This was based on the assumption that the legitimate users observations would not be perfectly aligned in a uniform cluster but a bit more scattered. Algorithm 3 shows how selection was done and how the LOF score was used for selecting observations from other users to train a certain user model.

#### 4.4.2 Using Random Selection

Perhaps the one of the simplest ways of selecting data from other users to pose as intruder data for a user model is to randomly select data from the other users. This way of selecting observations was done for comparison to selecting observations using LOF values, and is simply done through randomly selecting the same number of intruder observations as there are legitimate user observations in the model. Algorithm 4 shows how this was done.

**Algorithm:** LOF data selection

**Data:** Authorized user model, list of observations from the other users

**Result:** Selection of observations

$n := \text{authorizedModel.numberOfObservations};$

**for** *all observations from other users* **do**

$\text{lof} := \text{LOF}_{\frac{n}{2}}(\text{authorizedModel}, \text{observation});$

$\text{observation.lofScore} = \text{lof};$

**end**

$\text{otherObservations.sortAscendingByLOF}();$

$\text{startIndex} := (\text{otherObservations.length} - n)/2;$

$\text{endIndex} := \text{start} + n;$

$\text{selection} = \text{otherObservations.between}(\text{startIndex}, \text{endIndex});$

**return** *selection*;

**Algorithm 3:** Algorithm for LOF selection. Input for the algorithm is the model with legitimate user observations as well as a list of all the observations made from other users. For each observation made by other users, the LOF score is calculated. These observations are then sorted, and an equal amount of observations to the ones in the authorized user models are selected from the middle of the list. The LOF calculated is the observations outlier factor relative to the authorized user model.

**Algorithm:** Random data selection

**Data:** Authorized user model, list of observations from the other users

**Result:** Selection of observations

$n := \text{authorizedModel.numberOfObservations};$

$\text{selection} := [];$

**while**  $\text{selection.length} < n$  **do**

$\text{observation} := \text{otherObservations.getRandomElement}();$

$\text{selection.add}(\text{observation});$

$\text{otherObservations.remove}(\text{observation})$

**end**

**return** *selection*;

**Algorithm 4:** Algorithm for random selection. An equal amount of observations to the amount of observations in the authorized user model are selected. Each observation is selected randomly from the list of other user observations.

## 4.4. MODEL TRAINING

### 4.4.3 Contextual Models

As seen in chapter 2, there are many biometric features which can be extracted from touch gestures. While [6] does look at the context of gesture tracking in the sense of which application is used, there is another way of viewing context. [15] looks at touch position on buttons; expanding on the concept, the system built and evaluated in this thesis trains models for certain UI elements as well as the screen itself. This means that not only biometric features from the entire screen’s perspective are examined, but interaction with each individual UI component as well. In the framework developed, each element can be marked for tracking to enable a developer to choose which elements should be included in the tracking and model training. The idea is that a model is built for every tracked component, and classified according to the two algorithms previously mentioned in this chapter. A majority vote is made from the classifications of the observation in the different the models.

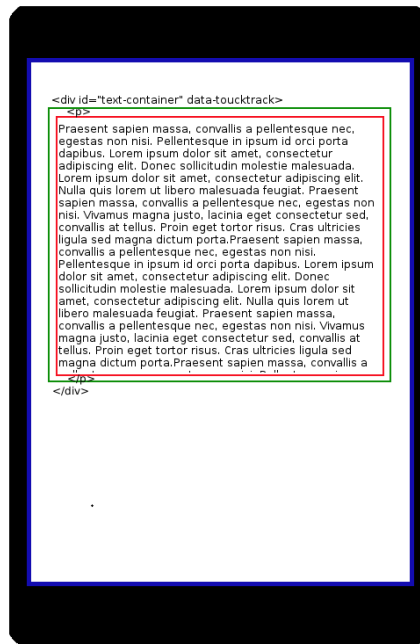
There are two differences in the features extracted when tracking individual UI components. The first one is that end position for a tap or swipe is not used. This is because of how JavaScript handles touch events; all events will be bound to the element which the first event was fired on in the event chain. Preferably when a user moves her finger outside a tracked element, the event chain would be able to restart and tracking done for the new element. Since this is not possible the end point is instead skipped for contextual tracking. Another difference is start position; instead of being the position on the screen, the position is calculated by formula 4.4.

$$\begin{aligned} relativeX &= x - ox \\ relativeY &= y - oy \end{aligned} \tag{4.4}$$

In formula 4.4,  $x$  and  $y$  are the start position’s coordinates relative to the web page, that is the  $pageX$  and  $pageY$  variables described in section 2.3. The element in question’s offset position from the start of the web page is represented by  $ox$  and  $oy$ .

Tracking based on UI component context is relevant due to the fact that there might be different components on different pages of the website, as well as these components having different positions on the screen based on how much the page is scrolled down. The aspect ratio of the screen being used might also influence the starting position of for example taps on elements. Portrait versus landscape mode also plays a part in this, since it might cause repositioning or stretching of certain elements. Figure 4.4 on page 40 shows in more detail how the tracking for an individual component is done. The *div* element (green rectangle) containing a paragraph tag with text has the *data-touchtrack* attribute set, which means that all gestures performed inside of the element will be bound to it. When a touch event it caught, it will not necessarily be bound to its HTML target element; the DOM is traversed

upwards until a parent element with the *data-touchtrack* attribute is found. The gesture will be bound to this element and an appropriate model is trained on the server. If no parent element with the *data-touchtrack* attribute could be found, only the model representing interactions in the context of the entire screen is trained. This means that if the paragraph element in Figure 4.4 had the *data-touchtrack* attribute, gestures made on this element would not "reach" the *div* element.



**Figure 4.4.** Image of a "web page" on a smartphone showing how touch events for individual components are caught. The red rectangle denotes the boundaries of the text element (p-element in HTML), and the green rectangle the containing div-element. In this example, events generated in the p-element will bubble up to the nearest parent having the *data-touchtrack* attribute, which is the div-element. The event will thus be connected to this element. Without the *data-touchtrack* attribute being used, all events belong to the "screen" of the phone used.

## 4.5 Evaluation

After data collection and model training, the final step was to classify sessions from intruders and legitimate users in order to evaluate the system. Classification is described in section 4.5.1, and motivation for the choice of metric and how it was calculated is given in section 4.5.2.



## 4.5. EVALUATION

### 4.5.1 Classification

Classification was done using ensemble  $k$ -NN, SVM and 1-NN classifiers. These classifiers have been used successfully in previous works examining touchscreen biometrics, and were deemed appropriate to test in this thesis work ([6, 31, 22, 2, 16, 8]). For  $k$ -NN, the ensemble learning version of the algorithm described in section 2.1.1 was implemented. The pseudo code for the implementation used is shown in algorithm 5.

**Algorithm:** Ensemble  $k$ -NN

**Data:** Unknown observation, Model

**Result:** Authorized score, Intruder score

intruder := 0;

authorized := 0;

n := model.numberOfObservations;

sortedSet = sort(model, UnknownObservation);

**for**  $k := 0, k \leq \sqrt{n}$ , *step 1* **do**

**if**  $k$  is even **then**

        | continue;

**end**

    i := 1;

**for** all observations *in* model **do**

**if** observation.class == 'authorized' **then**

            | authorized +=  $\frac{1}{\log_2 1+i}$ ;

**else**

            | intruder +=  $\frac{1}{\log_2 1+i}$ ;

**end**

        i++;

**end**

**end**

**return** (authorized, intruder);

**Algorithm 5:** Ensemble  $k$ -NN used for classification. In the algorithm, the model is a list of previously collected observations. When sorting this list, it is done ascendingly based on the distance to the unknown observation. The euclidean distance was used in this implementation.

Instead of a class, the implementation returns each classes weight. The classes are assumed to be two, "authorized" and "intruder". These weights were used in order to summate results from several different models, since each user didn't only have a single model. In turn these summated weights were used to calculate the confidence of the classification. For comparison, the 1-NN classifier was also used. This classifier as previously stated works in the same way as the  $k$ -NN classifier, but

only a value of 1 for  $k$  will be used. This means that with the same implementation as in algorithm 5, weights will either be 1 for intruders and 0 for authorized, or the opposite. Classification using a support vector machine was also used. This was done using an implementation from an installed package for Node.js, created by [14]. When producing the results in chapter 5, a RBF (radial basis function) kernel was used with tuning parameters along with a grid-search for parameters  $C$  and  $\gamma$  shown in table 4.1.

**Table 4.1.** This table shows the different values for  $C$  and  $\gamma$  when tuning the parameters for the SVM models. Note that the final pairs used are not represented by a row in the table: it simply lists all the different values used

$C$	$\gamma$
0.0625	0.00012207031
0.125	0.00048828125
0.5	0.001953125
2	0.0078125
4	0.03125
8	0.125
16	0.5
32	1
128	2
256	4

A 5-fold cross-validation was implemented and used to select the parameters in the grid-search, as per recommendation of [13][p. 184]. Since a user had more than one model, several pairs of the tuning parameters had to be calculated to get the best fit, one pair for each model.

Apart from evaluating the classifier using all of the features, backward elimination as described in section 2.1.3 on page 14 were also used for comparison. This was done in order to reduce the number of dimensions the classifier operated in, as well as try to remove irrelevant features. Backward elimination was only done for the 1-NN and  $k$ -NN classifiers, since they did not need retraining each time a feature was eliminated. This was because of how the training sets were stored: the training sets were only a list of vectors, so when ignoring a new feature one would simply not use it when calculating the distances between an unknown observations and the ones in the training set. Since the SVM classifier needs to construct a hyperplane for classification, a feature cannot be ignored once this has been done. A new hyperplane would have to be constructed ignoring a certain feature. Due to training being slow, this was not done for SVM.

## 4.5. EVALUATION

### 4.5.2 Evaluation Metric

EER was chosen as the metric for evaluating the system due to accuracy not being representative enough of the systems performance. EER has also been used in other works such as [8, 32, 22]. Accuracy is the fraction of correct classifications. A system rejecting all users would achieve an accuracy of 50 %, assuming that there are an equal amount of legitimate users and intruders when calculating accuracy. In the event of accepting all users under the same condition, an accuracy of 50 % would also be achieved. A system using completely random classification would reach an accuracy of about 50 %. In other words, the accuracy can be greatly skewed by either FAR or FRR being very high or low. EER shows the performance at its most stable and also weights FAR and FRR equally. It conveys the rate of incorrect classifications for random users, regardless of them being intruders or legitimate users; the classification error is the same.

As mentioned in section 2.4 in chapter 2, a confidence value is needed for calculating EER. When using confidence in this section, it refers to how confident the system is that the current user is an authorized user. The value was calculated for each session, taking every observation in the sessions into account. The confidence was calculated using algorithm 6.

**Algorithm:** Confidence Fraction

**Data:** Session, User models

**Result:** Confidence value

authorized := 0;

intruders := 0;

**for** *all models* **do**

**for** *all observations in session* **do**

        intruders += getIntruderWeight(model, observation);

        authorized += getAuthorizedWeight(model, observation);

**end**

**end**

confidence = authorized/(authorized + intruders);

**return** *confidence*;

**Algorithm 6:** Confidence calculation based on results from classification. The weights for the intruder and authorized classes are number of nearest neighbors for 1-NN, summated weights for ensemble k-NN and respective classifications for SVM.

In algorithm 6, the weights refer to the number of nearest neighbors of a certain class in a model for an observation in the  $k$ -NN case. In the SVM case, the implementation used only allowed for a result in the form of a class. The confidence was calculated according to algorithm 6, but instead of using the number of neighbors (as in  $k$ -NN) for weight, the number of classifications were used. So if a session contained 10 observations, where 8 of them were classified as authorized and two of them as intruders, the "intruder weight" for an observation classified as authorized would be 0, but the authorized weight would be 1. Assuming 5 models where all observations were classified in the same way for every model, the confidence would be  $\frac{5*8}{(5*2+5*8)} = 0.8$ . The same went for the 1-NN classifier, since this classifier will return either 1 for intruder weight an 0 for authorized, or the opposite.

False acceptance rate (FAR) and false rejection rate (FRR) was calculated by classifying every session recorded in phase two from the intruders, as well as the ones from the legitimate users. Intruder sessions were used for FAR calculation, and legitimate sessions for FRR calculation. Every observation made in a session were taken into account when classifying a user. FAR was calculated using algorithm 7 on page 45, and FRR using algorithm 8 on page 45. In both of these algorithms, an intruder session is a list of feature vectors, each vector representing an observation made during the session. A user profile is a set of all models belonging to a user. Each model is a list of feature vectors from the user the model belongs to, as well as intruder vectors added in the training phase.

EER was calculated by trying every confidence value for the system between 0 and 1 with 0.01 increments in value, and find the confidence for which the FAR and FRR were equal. FAR and FRR were calculated based on results from classifications using machine learning with the collected data set. Ensemble  $k$ -NN, 1-NN and SVM were used, and their models trained in the way described in section 4.4.

## 4.5. EVALUATION

**Algorithm:** False Acceptance Rate

**Data:** User profiles, intruder sessions, confidence threshold

**Result:** Error rate

incorrect := 0;

total := 0;

**for** *all intruder sessions* **do**

**for** *all users* **do**

        confidence := confidenceFraction(session, userModels);

        if(confidence > confidenceThreshold) incorrect += 1;

        total += 1;

**end**

**end**

errorRate := incorrect/total;

**return** *errorRate*;

**Algorithm 7:** Calculation of false acceptance rate. This algorithm classifies all intruder sessions and counts the number of sessions which were accepted, that is had a confidence above the threshold. An error rate of 0 would mean that no intruder session was accepted. User profiles are a set of every users set of trained models.

**Algorithm:** False Rejection Rate

**Data:** User profiles, legitimate sessions, confidence threshold

**Result:** Error rate

incorrect := 0;

total := 0;

**for** *all legitimate sessions* **do**

**for** *all users* **do**

        if(session.user != userModels.user) continue ;

        confidence := confidenceFraction(session, userModels);

        if(confidence < confidenceThreshold) incorrect += 1;

        total += 1;

**end**

**end**

errorRate := incorrect/total;

**return** *errorRate*;

**Algorithm 8:** Calculation of false rejection rate. This algorithm classifies all sessions from legitimate users and counts the number of sessions which were rejected, that is had a confidence below the threshold. An error rate of 0 would mean that no legitimate session was rejected. User profiles are a set of every users set of trained models.

**Algorithm:** Equal error rate

**Data:** User profiles, legitimate sessions, intruder sessions

**Result:** Error rate

minErrorRate :=  $\infty$ ;

minDifference :=  $\infty$ ;

**for**  $c := 0, c \leq 1$ , **step** 0.01 **do**

    far := FAR( $c$ , userProfiles, intruderSessions);

    frr := FRR( $c$ , userProfiles, legitimateSessions);

**if**  $\text{abs}(\text{far}-\text{frr}) < \text{minDifference}$  **then**

        minDifference =  $\text{abs}(\text{far}-\text{frr})$ ;

        minErrorRate :=  $\min(\text{minErrorRate}, (\text{far}+\text{frr})/2)$ ;

**end**

**end**

errorRate := minErrorRate;

**return** errorRate;

**Algorithm 9:** Algorithm for calculating EER. This algorithm does not make the assumption that there will always be a confidence value for which FAR and FRR are equal. Instead the point where the difference between FAR and FRR is the smallest is examined. If there are several points where this holds true, the point with the smallest average error rate is chosen. If at this point the difference between FRR and FAR is 0, the error rate will be equal to the FRR (or FAR). Otherwise, it is the average of the two error rates.

# Analysis

# Chapter 5

## Results

In this chapter the results of the experiment are presented. The results presented in this chapter are the ones deemed most relevant for evaluating the different choices in methodology discussed in chapter 4 on page 30, however a full list of results for all the different configurations of the system used in the experiment can be found in appendix C. All results in this chapter have been produced from the same data set. The difference between them are what algorithms and configurations were used when analyzing results, as well as how models were trained.

Results were calculated by using the models of 11 legitimate users which had trained their models as well as completed additional test sessions. The number of intruder sessions (from users with no registered profile) used were 25, and 24 sessions from registered users were used.

### 5.1 Selection with LOF

In this section results from selection observations for selection using the LOF as described in section 4.4.1 compared to random selection are presented. Evaluation and EER calculation was done as described in section 4.5.

#### 5.1.1 $k$ -NN

As seen in table 5.1 classification using the  $k$ -NN classifier, the lowest EER was achieved by using 1-NN with contextual models with an EER of 8.3 %. This however was only when LOF was chosen as the method for selecting observations; when using random selection, the average EER was 13.3 %. The average was calculated by doing random selection 5 times and evaluating each of the model sets separately.



## 5.2. CONTEXTUAL MODELS

**Table 5.1.** Comparison of LOF and random selection. EER is presented for both LOF selection (LOF EER) and the average of 5 instances of random selection (Rnd. EER). The standard deviation is shown in parenthesis for the random selection EER.

Algorithm	Scaling	Contextual models	LOF EER	Rnd. EER
Esm. $k$ -NN	No	Yes	11.20 %	9.00 % (2.88 %)
Esm. $k$ -NN	Yes	Yes	12.10 %	22.54 % (4.05 %)
Esm. $k$ -NN	No	No	12.50 %	11.42 % (1.99 %)
Esm. $k$ -NN	Yes	No	22.30 %	21.04 % (0.46 %)
1-NN	No	Yes	8.30 %	13.30 % (3.29 %)
1-NN	Yes	Yes	16.50 %	27.08 % (5.38 %)
1-NN	No	No	18.00 %	15.04 % (2.03 %)
1-NN	Yes	No	25.20 %	26.06 % (1.82 %)

### 5.1.2 SVM

The lowest EER when using SVM was achieved when using random selection, scaling and simple models (seen in table 5.2). The difference in performance between contextual and simple models was only 1.1 percentage points when using LOF selection, and 0.7 percentage points when using random selection.

**Table 5.2.** Comparison between LOF selection (LOF EER) and random selection (Rnd. EER) for SVM. The EER for random selection is an average EER for the different configurations from 5 different instances of trained models using random selection. Standard deviation for random selection EER is shown in parenthesis.

Algorithm	Scaling	Contextual models	LOF EER	Rnd. EER
SVM	Yes	Yes	11.40 %	6.72 % (2.89 %)
SVM	Yes	No	12.50 %	6.02 % (2.18 %)

The EER for random selection was taken from 5 different instances, as in the  $k$ -NN evaluation. Unlike  $k$ -NN, random selection for SVM performed more consistently with a lower EER for 10 out of 10 instances for the different configurations.

## 5.2 Contextual models

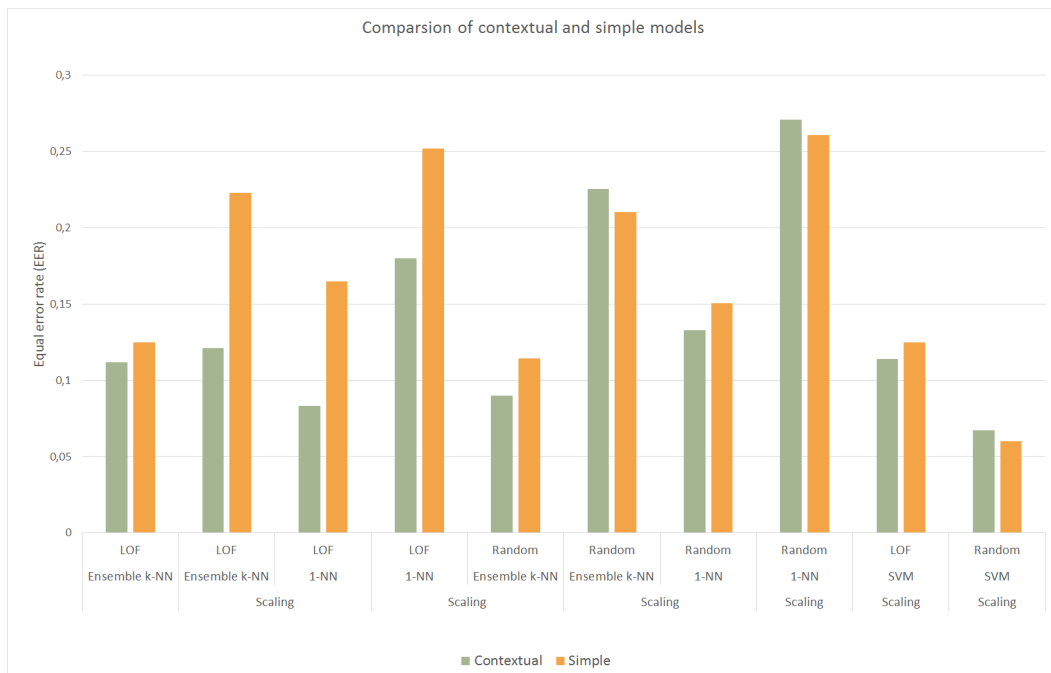
In this section a comparison between using contextual models and simple models is presented. The lowest EER was achieved when using simple models as opposed to contextual, with an EER of 6.02 %.

**Table 5.3.** This table shows all the different algorithms and their EER when using contextual or simple models. The difference in EER between contextual and advanced models seem to be smaller when using random selection as opposed to LOF. The standard deviation is shown in parenthesis.

Algorithm	Scaling	Selection	Contextual models EER	Simple EER
Esm. $k$ -NN	No	LOF	11.20 %	12.50 %
Esm. $k$ -NN	Yes	LOF	12.10 %	22.30 %
1-NN	No	LOF	8.30 %	16.50 %
1-NN	Yes	LOF	18.00 %	25.20 %
Esm. $k$ -NN	No	Random	9.00 % (2.88 %)	11.42 % (1.99 %)
Esm. $k$ -NN	Yes	Random	22.54 % (4.05 %)	21.04 % (0.46 %)
1-NN	No	Random	13.30 % (3.29 %)	15.04 % (2.03 %)
1-NN	Yes	Random	27.08 % (5.38 %)	26.06 % (1.82 %)
SVM	Yes	LOF	11.40 %	12.50 %
SVM	Yes	Random	6.72 % (2.89 %)	6.02 % (2.18 %)

Contextual models yielded a lower error rate for most configurations, in 7 out of 10 cases, even though the lowest EER was achieved with simple models. This is further highlighted in figure 5.1.

### 5.3. BACKWARD ELIMINATION



**Figure 5.1.** Visual representation of the data in table 5.3. In 7 out of the 10 different cases contextual models had a lower EER. For the cases where simple models had lower EER, the difference was not as great as when contextual models had lower EER.

## 5.3 Backward elimination

In this section results for backward elimination are presented. These results are compared to the results when not using backward elimination, and separated into a section for LOF selection and random selection where results are compared separately.

### 5.3.1 LOF Selection

Backward elimination yielded improved results for 5 out of 8 configurations with LOF selection. Unlike previous results the Ensemble  $k$ -NN performed better than 1-NN. The lowest EER achieved was 10.6 % for the ensemble  $k$ -NN with contextual models, shown in table 5.4.

**Table 5.4.** This table shows the different configurations and their improvement with backward elimination (BWE). Lowest EER was achieved with ensemble  $k$ -NN with contextual models and the greatest improvement in performance can be seen with the 1-NN classifier without scaling and with simple models

Algorithm	Scaling	Contextual models	EER	EER (BWE)
Esm. $k$ -NN	No	Yes	11.20 %	10.60 %
Esm. $k$ -NN	Yes	Yes	12.10 %	12.70 %
Esm. $k$ -NN	No	No	12.50 %	11.90 %
Esm. $k$ -NN	Yes	No	22.30 %	21.20 %
1-NN	No	Yes	8.30 %	20.60 %
1-NN	Yes	Yes	16.50 %	13.10 %
1-NN	No	No	18.00 %	16.50 %
1-NN	Yes	No	25.20 %	32.80 %

### 5.3.2 Random selection

As in LOF selection, results improved when using backward elimination together with random selection. Random selection performed better than LOF selection with a lowest average EER of 5.6 %, shown in table 5.5. All configurations and instances improved when using backward elimination except for 1-NN classification with Scaling and simple models; the error rate for this configuration was 0.24 % higher.

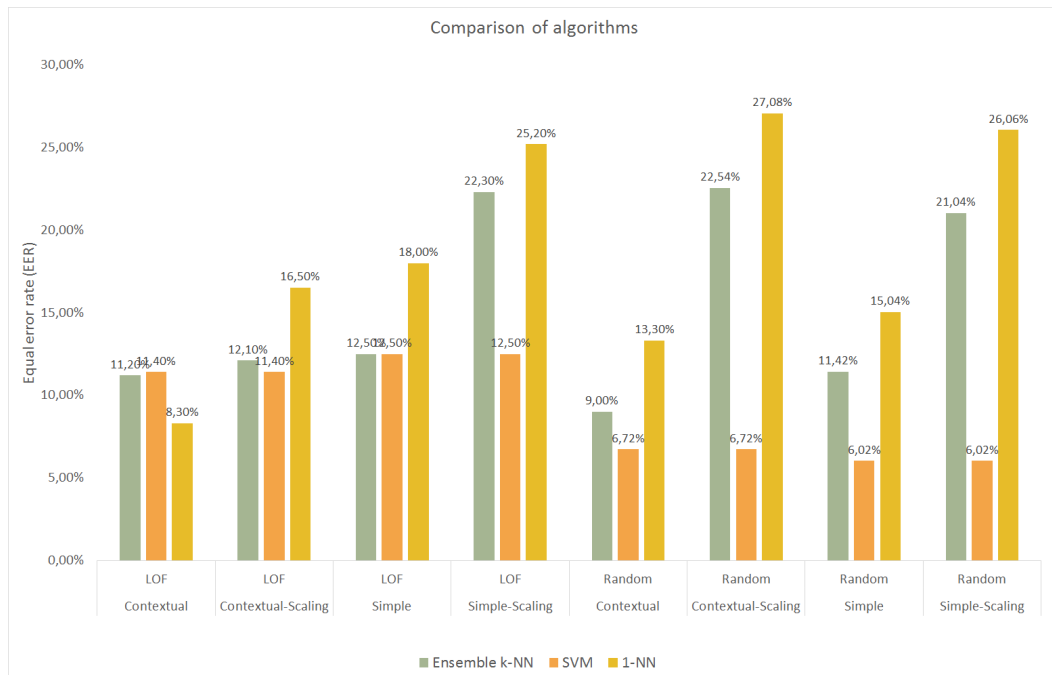
**Table 5.5.** This table shows the different configurations and their improvement with backward elimination (BWE). Lowest EER was achieved with ensemble  $k$ -NN with contextual models and the greatest improvement in performance can be seen with the 1-NN classifier with scaling and contextual models. The standard deviation for the average values are shown in parenthesis.

Algorithm	Scaling	Contextual models	EER	EER (BWE)
Esm. $k$ -NN	No	Yes	9.00 % (2.88 %)	5.60 % (2.29 %)
Esm. $k$ -NN	Yes	Yes	22.54 % (4.05 %)	20.02 % (5.74 %)
Esm. $k$ -NN	No	No	11.42 % (1.99 %)	7.62 % (3.12 %)
Esm. $k$ -NN	Yes	No	21.04 % (4.6 %)	18.88 % (3.27 %)
1-NN	No	Yes	13.30 % (3.29 %)	6.22 % (3.3 %)
1-NN	Yes	Yes	27.08 % (5.38 %)	19.94 % (4.14 %)
1-NN	No	No	15.04 % (2.03 %)	11.08 % (2.62 %)
1-NN	Yes	No	26.06 % (1.82 %)	26.30 % (4.51 %)

## 5.4. COMPARSION

### 5.4 Comparison

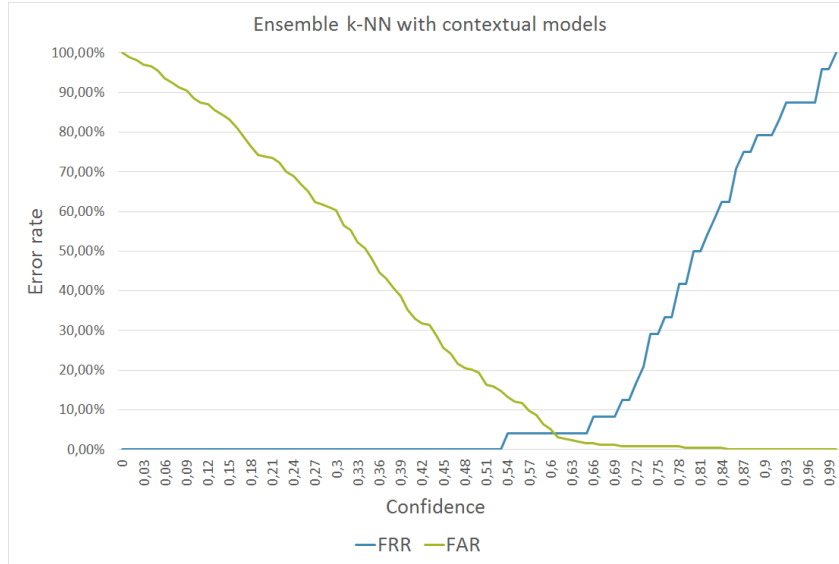
In this section a comparison of all the different algorithms is given in the same graph. Looking at figure 5.2, random selection performance for SVM is almost the same regardless of the usage of contextual or simple models; this appears to be consistent for every algorithm. However, when using LOF as selection method results vary greatly between using contextual models and simple models.



**Figure 5.2.** Comparison of performance for 1-NN, ensemble  $k$ -NN and SVM. Performance is measured in equal error rate (Y-axis), where lower error rate is better, and results are shown for different configurations. Since SVM selection and classification was always scaled, the results are the same for SVM for scaled and un-scaled contextual and simple models

The lowest EER achieved among all configurations was 3.6 % using ensemble  $k$ -NN and contextual models. This result was achieved using random selection and backward elimination, and the average EER for all random instances using this configuration was 5.6 %, which is also the lowest average EER for all configurations. The average FAR and FRR and their values when changing confidence for this configuration can be seen in figure 5.3. Only 0.42 percentage points behind this however is the configuration using SVM and simple models with scaling. These results were well in line with what was expected, and even lower than the expected error rate.

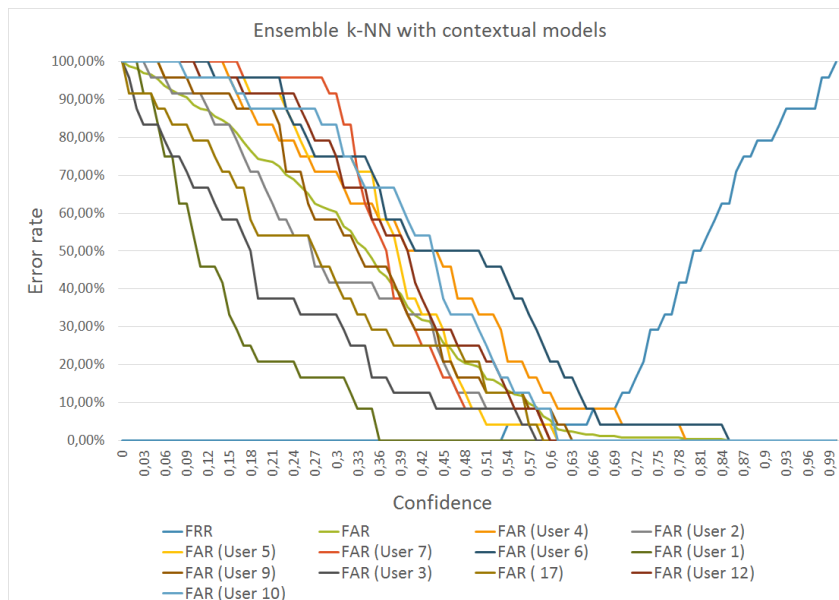
Some previous studies achieved an even lower error rate than 5.6 % ([31, 2, 8]), however in these works features like pressure and touch area were available.



**Figure 5.3.** FAR and FRR curves for the configuration which yielded the lowest EER. The EER can be observed at a confidence value of 0.61; at this confidence FAR is at 3 % and FRR at 4.2 %. Since an exact value of EER can not be reached with an interval of 0.01 for confidence, the EER is calculated as the average of the values at this point which is 3.6 %

Although performance was measured for average FAR, the FAR for individual users were not as uniform and differed from user to user. Figure 5.4 shows an individual curve for each user and their FAR. This FAR is the rate with which intruders are incorrectly classified as legitimate for a certain user profile. Though the EER when looking at a specific user might not be very different from the average, the confidence required for achieving this result might differ. User 1 achieves an FAR of 0 % for a confidence as low as 0.36, while user 6 does not reach this error rate before a confidence of 0.85.

## 5.4. COMPARSION



**Figure 5.4.** This graph shows individual FAR values for the different users used in calculating the average FAR. The error rate differed a lot between users, especially which confidence was needed to yield a low FAR.

## Chapter 6

# Discussion

In this chapter the methodology is discussed in sections 6.1 and 6.2, and the results discussed in section 6.3. Error sources are presented in section 6.4, as well as what can be done to mitigate their impact. Since the motivation behind this thesis was usage of touchscreen biometrics as an IDS, section 6.6 presents some security issues with touchscreen biometrics on the web and how they can be solved. Section 6.5 explains why keystroke dynamics for touchscreens are hard to implement in the browser, and lastly section 6.7 presents areas warranting more research and ideas for future work.

### 6.1 Feature selection

When selecting features on the touchscreen to be used, both multi-touch gestures and horizontal strokes were left out. Although it is quite possible to detect more advanced gestures on touchscreens with support for multi-touch, since the touch event objects in JavaScript contains data for all active pointers on the screen. However, more advanced gestures such as pinches and rotations are less common when building websites; this is due to websites needing to function on both mobile browsers as well as desktop browsers. Most desktops using keyboard and mouse cannot perform gestures similar to pinches or rotations, however clicking and scrolling are substitutes for swiping and tapping.

Regarding horizontal scrolling, it has long been considered bad practice for websites. Of course, this view is not completely un-biased, and many designers and usability experts might disagree about its use on the modern web. [19] discusses and tests where the attention of users are directed on websites, which seems to lean towards the left. In light of this, as well as that the same study found that the majority of users do not look at the right-most part of a page even if it is horizontally scrollable, only upwards and downwards stroke gestures are inspected in this thesis work.



### 6.2 Model training

In this section model training and its impact on results is discussed. Tuning parameters for SVM as well as separating models for taps and strokes are discussed and motivated.

#### 6.2.1 SVM

For the SVM, the scaling was done by taking every model and every observation from the intruder sessions into consideration. In other words, all data from the sessions and observations were traversed in order to find the maximum and minimum values for the different features. The reason scaling was done cumulatively for all sessions instead of individually for each session was performance; the library used would have required re-training of the model in order to scale the values. Due to training being computationally expensive, all model training was done prior to evaluating the system. This meant that at the point of model training the maximum and minimum values had to be known, thus the pre-computation of these values. In a practical situation, intruder sessions would not be known prior to model training, which could result in features with values much greater or smaller than the ones of the trained model skewing results. This problem did not impact performance for k-NN since no re-training of the model needed to be done, scaling could be done at the time of classification.

When using the SVM classifier, a 5-fold cross-validation was used with a grid-search in order to select tuning parameters. There are other methods for selecting tuning parameters, as well as using another value for  $k$  in the  $k$ -fold cross-validation. The reason a value of  $k = 5$  was used is because of long computation times when using a higher value. Since every user had multiple models, tuning had to be done for over 100 different models, which made computation times long. This means the SVM classifier might not have had the best performance in this thesis work, but should not be disregarded as a good classifier before more research has been done on tuning these parameters. Different kernels as well as methods for cross-validation could be used.

#### 6.2.2 Separate models

The reason behind separating taps, upwards and downwards strokes into their own model is difficulty of representing direction (or lack of) for a swipe or tap as a metric. If 1, 0 and -1 were used, these would be normalized to 0, 0.5 and 1, and not having more impact than other features. This could result in upwards swipes being compared to downwards swipes; a user who swipes upwards in the similar fashion as another user swipes downwards should not be considered similar if they do not also swipe similarly in the opposite directions. Also, models are trained both in relation to the screen itself as well as individual UI components (described in section 4.4.3).

## 6.3 Results

In this section results from chapter 5 are discussed. Not only the results with the lowest achieved EER are discussed, but also the different configurations used and their performance.

### 6.3.1 Calculation of EER

Due to a finite amount of test cases for calculating the FRR, and usage of an average value for the FAR, it was not always possible to yield an exact value of the EER. This was caused by the FAR and FRR not having the exact same values for any confidence thresholds in a set interval, i.e the point where the curves intersect in the graphs presented in chapter 5. In such cases the value for the EER was set to an average of the FAR and FRR at the point where their values were the closest. This is not an exact measurement and should not be treated as such, though it gives an estimate of the system's performance.

### 6.3.2 Selection with LOF

Regarding usage of the LOF for observation selection in the way described in section 4.4.1 on page 36, it is hard to conclude that selecting observations for training with LOF in this way yields better performance than random selection. To the contrary, the lowest EER was achieved with random selection. Considering the average EER came from evaluation of 5 different sets of randomly trained models, there is little indication that the results would be too unstable to disregard. Even the highest EER achieved for most random selection sets was still lower than the one with LOF. Considering the difference in computation time with LOF as opposed to random selection, LOF cannot be recommended as a means of selecting observations as described in this thesis work.

### 6.3.3 Contextual models

In most cases using contextual models yielded lower EER than simple models. However, this might have been due to the fact that  $x$ - and  $y$ -coordinates for the end position were not used as a feature for contextual models. This decreases the number of dimensions in which classification is made; it is therefore unclear whether the better results were due to less dimensions or that context was actually important. As can be seen in Table 5.3, the difference in EER when using contextual models as opposed to simple can vary greatly, and is even clearer in figure 5.1 on page 51. Although simple models in some cases yield better EER for certain configurations, the EER of contextual models seem to be overall greater. The difference in EER is not as great when simple models perform better as when contextual models perform better. This difference is further highlighted in figure 5.1.

## 6.4. ERROR SOURCES

### 6.3.4 Scaling

It is curious that the lowest EER were achieved without scaling. Even without backward elimination, the the lowest EER still came from ensemble k-NN with contextual models and no scaling. This might have been caused by the magnitude of certain features being larger than others by their nature, thus having a larger impact on results. If these features were to be particularly good for distinguishing users, scaling might impact results negatively by not allowing these results to have as much influence. This however only holds true for random selection. For LOF selection, results were almost equal or better when using scaling for ensemble k-NN with contextual models. This indicates that the impact of scaling might be related to observation selection.

### 6.3.5 Backward elimination

Performance was better for backward elimination in general, however the increase in performance was not great for k-NN with LOF selection. In some cases, performance even got much worse after backward elimination. This was likely due to how backward elimination was evaluated, that is, what metric was used for determining whether feature removal yielded better results. Accuracy was used instead of EER, which could have resulted in an increase in only FRR or FAR. If the difference is very large between the two and their curves differ greatly, the intersection point might come at a higher error rate. Due to the computation time for calculating EER, accuracy was used in this thesis work.

Backward elimination could have been done for each confidence value for each user to explore the relation between skipping attributes and confidence value, however due to time restraint and long run times for this kind of analysis, backward elimination was only done for a fixed confidence value. Using backward elimination might improve result for a specific set of models and observations, it is however difficult to assess the performance backward elimination would yield in a real-world application of touchscreen biometrics. Since backward elimination is biased towards the data set and model it is performed upon, a larger data set should be used to reliably discard certain attributes.

## 6.4 Error sources

In this section various error sources are presented, and their impact on results are discussed. Both errors from the experiment are presented as well as technical errors or bugs in the classification.

### 6.4.1 Screen sizes

In other studies ([8]), smartphones with similar screen sizes were used, but when freely distributing the experiment to reach a large number of subjects it is hard to control what devices are used. One problem that arises when doing classification for devices with many different screen sizes is that the shape of a stroke might vary for the same user depending on the physical size of the device. For example, if a user with small hands were to use a device with a much larger screen size than she usually does, the coordinates and length of the gestures could be displaced. A similar problem also appears when using devices with different resolutions; a Samsung SM-G800F smartphone has a resolution of 720x1280 pixels and a physical screen size of 5.6x9.8 cm, while a Samsung GT-I9195 has a resolution of 540x960 pixels and a physical screen size of 5.3x9.4 cm. The resolution in physical pixels of the screen aren't proportional to the physical size of the device, however it is partially mitigated by the fact that browsers use reference pixels when handling coordinates on the screen ([17]). This means that point on the physical screen of both devices will have roughly the same coordinates in relative pixels. However, for screens with largely different physical screen sizes the gestures might have the same length in relative pixels (swiping across the screen horizontally), but because of different physical sizes swipes on the larger screen takes longer. This affects velocity since it is easier to get a higher velocity on a smaller device due to reference pixels being the same, but the physical size being smaller.

### 6.4.2 Feature selection

Not all features presented by [8, p. 10] were used in order to avoid the curse of dimensionality. Thus there is a risk that features which might have improved classification was left out. Since results improved by using backward elimination, not all features seemed to be relevant for every user. This means that features which [8] deemed contained little information might have been useful for certain users. It is also possible that too many features were selected; the improved results with backward elimination indicate that this was in fact the case.

### 6.4.3 Bug in start-point calculation

When calculating the relative start point, that is the start point inside an UI element for the contextual models, the element's offset from the top-left corner was used. The position in the viewport was subtracted by the offset to yield a position relative the element. This resulted in relative y-values being negative for certain components. The cause is simple; when retrieving the position inside the viewport, `clientY` was used instead of `pageY` in the touch object (see section 2.3 on page 2.3). This is incorrect since `clientY` does not take scrolling offset into account. Although the value is incorrect, it was calculated in the same fashion for all sessions. What is important for the different features is that they are measured in the same way and reflect difference between users. It is thus unlikely that this bug had negative

## 6.4. ERROR SOURCES

impact on results. The reason why this bug was not fixed even though it is known is that it was discovered after several users had already started their model training. In order to not have to discard test data, the bug was left in place and the feature was extracted in the same fashion for all users.

### 6.4.4 Number of participants

During the data collection in the first phase of the experiment carried out in this thesis (section 4.3), not every participant completed the test 7 times. This might have impacted the results of the experiment, it is however difficult to determine since the total amount of data collected from a single session might vary greatly from user to user. Also, the amount of sessions were decreased from 7 to 5 half-way through the first phase. This was due to many users not completing the test enough times, or showing unwillingness to continue with the test sessions. In total only 11 users completed the test more than 5 times, although not all of these results could be used. A user who completed the test 5 times only built a model; to evaluate the system sessions were needed which were not part of the model building. These sessions would be used to calculate the FRR of the system, since by only having built a model doesn't say anything about how well the system can recognize authorized users. The excessive sessions not part of the model building were few for certain users; even if an average FRR was calculated and presented in chapter 5 on page 48, this does not say much about the FRR for individual users. If a user only completed the test once after building her model, the FRR can only have values 0 or 1. That is, this single session can either be rejected or accepted. This means that although the average FRR seemed promising, it could potentially be worse for individual users. More data should be collected for every user that is not part of the model building to get more reliable results for FRR. It is difficult to estimate the impact from a larger number of sessions, however looking at figure 5.3 on page 54 the FRR curve is rougher than the FAR curve and has more fluctuations. It is possible that an increased number of sessions would have yielded a smoother curve, but the point where it intersects with the FAR curve might not necessarily have differed much.

### 6.4.5 EER calculation

When calculating the EER, the average FAR was used as well as the FRR for all users in the system. However, when looking at the FAR in this way (the average), it show the error rate for the entire system, not individual users. In a natural scenario, an attacker has already gained access to a user account, and the system's task is to identify the attacker as an intruder. This will be done by comparing the observations made in the session against the stored profile for only the user in question. So when looking at the EER in the results, this does not show whether a particular user is easy to impersonate or not, just the all around performance. The reason for why evaluation has been done in this way is the lack of data collected for

calculating the FAR; as mentioned in section 6.4.4, the number of sessions for each user that could be used for FRR calculation varied. Thus the average was used for FRR, that is the FRR for all users and the system as a whole. This led to the choice of also calculating FAR as an average. The relevance of this decision is that although the system might reach a good EER of 5 %, this does not reflect the EER for a specific user, which could be much better or worse.

#### **6.4.6 Movement**

In this study walking patterns and user movement was not taken into consideration when performing tests, as opposed to the works by [2]. Since the experiment was done remotely, no control over the participants were exercised or instructions given. Both accelerometer readings as well as gyroscope was used for classification, both which are influenced by user movement. This poses a problem since a user might complete the training sessions while walking, and then use the system sitting down. The model will thus be trained with different values for motion and angular velocity, which might impact classification.

### **6.5 Lack of keyboard interaction**

As mentioned in section 3.3 in chapter 3, JavaScript cannot catch and listen to touch events fired on the virtual touchscreen keyboard. A possible solution might be to redesign mobile browsers to support these kinds of touch events, or build the web system inside a hybrid application, that is a normal application running web code in a closed environment where native functionality is available. However, both of these solutions have serious drawbacks in the form of compatibility and distribution. If a hybrid application is used, this application must be installed and launched as a normal application even though its running with web technologies, thus eliminating the benefits of distribution through normal web pages and the HTTP protocol. The same goes for native applications, although in this case native code would be run instead of JavaScript. Compatibility is also an issue; a monitoring system using JavaScript only capable of running in certain browsers which enable touch events for the virtual keyboard would not be as useful, since users would be forced to switch browsers in order to use a service.

### **6.6 Security**

One weakness of client side code such as JavaScript is that code can be easily manipulated by an attacker. Since the code for gathering touch input is available to whomever retrieves the web page, an attacker can access and use the code in order to set up a false web page and steal touch data from users. This data could in turn be used for impersonation. This makes contextual tracking even more important, since it might mitigate the risk of impersonation; it would be more difficult for

## 6.7. FUTURE WORK

an attacker to gain information about interaction with certain UI components than interaction with the screen itself, since it poses the requirement that the elements in question are present when data is stolen. This means that not only touch data must be gathered for impersonation, the site where data is gathered must also be identical to the one where the impersonation is later to take place. For an attacker, this poses a greater challenge since an entire site might have to be copied, and the user whose data is being stolen must be fooled that the site is the real site she wanted to access.

Permanence might be a problem with intrusion detection for touchscreen biometrics. [31] found permanence to be lacking for touchscreen interactions, this might result in not enough information being retrieved for reliable classifications. Also, since the JavaScript code can be directly accessed, an attacker could potentially shut down the tracking so that no data is sent. A solution to both of these problems could be to only make decisions when certain defined critical operations are carried out, for example transferring money to a different bank account. Such operations could also be set to only execute if the user passes authentication by the classifier. This allows for data to be gathered when navigating the page and not doing anything "important". In the event of touch tracking being disabled by the attacker, the classifier will have no available data when a critical operation is carried out and will reject it.

## 6.7 Future work

In this section, different areas for future work are presented. These are both completely new areas not explored in this thesis, as well as variations and extensions of the work done in this thesis.

### 6.7.1 Model training

Because of the difference in EER when using random training as opposed to training using the local outlier factor, model training plays an important part when using machine learning for touchscreen biometrics. Model training warrants more research, among these areas different usage of the local outlier factor could be explored, as well as using a complete set of values for  $k$  (using a more optimized implementation). Another way of creating models could be to artificially create user data based on previous sessions, thus minimizing the number of training sessions needed to complete model training.

Being density based, the LOF algorithm should also be affected by the curse of dimensionality. It would be interesting to eliminate features prior to model training thus decreasing the number of dimensions, instead of using it afterwards. LOF training could then be used in the reduced data set with a process same as backward elimination, with accuracy measured as the accuracy for classification using the resulting model.

### 6.7.2 Backward elimination

As discussed in section 6.4.4, more data should be collected from legitimate users with trained profiles in order to better determine FRR. This could be done in a larger study of similar nature: the technique could be implemented in an existing product in order to evaluate the method in a real-world scenario over a longer time. In such project, the applicability of backward elimination could also be evaluated in order to test how the biased nature of the wrapper method affects results when having access to a larger set of data.

### 6.7.3 User experience

Of course, the future work of this subject is not limited to classification and machine learning; user experience plays an important role when implementing this system in a real product. What EER is acceptable for users to not perceive the system as intrusive? Looking at the problem from a computer security perspective, one could ask what other mechanisms should be combined with the touchscreen biometrics, since it is doubtful that an EER of 5.6 % or even 3.6 % is enough as a single deciding factor.

### 6.7.4 SVM

Before disregarding SVM as a good classifier, more research should be put into selecting a suitable kernel function as well as other parameters relevant for classification. Feature selection for SVM could also be done using cross-validation in similar fashion as the backward elimination. More advanced methods for selecting tuning parameters than grid-search exist, which also warrants more research.

### 6.7.5 Contextual models

Looking at the performance when using contextual models as opposed to simple models (entire screen), it is clear that there are differences in performance and even benefits with using contextual models. Perhaps more features could be extracted in a contextual sense, or even completely new features unique to certain kinds of UI components. Other studies such as the one by [6] looks at the context in the form of behaviour for different applications on the phone; this might be extended to different pages on the website.

### 6.7.6 Pointer events

A new specification not widely adopted by browser vendors is the pointer events specification in JavaScript ([21]). This specification might allow for usage of the same code for many different devices with different inputs, since the goal is to standardize input events to make it easier to create cross-device websites, but still be able to device-specific behaviour when necessary. If this specification becomes



## 6.7. FUTURE WORK

adopted in the future, research could be put into examining how it affects touch-screen biometrics on the web and what new challenges are posed by this standard.

## Chapter 7

# Conclusion

Despite not being able to use pressure and touch area as biometric features, an EER of 5.6 % with ensemble  $k$ -NN is roughly in line with earlier research done for the android OS ([6, 31, 22, 2, 16, 8]), and well below the expected error rate. Based on the results in this thesis, the recommended algorithm for machine learning when doing touchscreen biometrics is ensemble  $k$ -NN with contextual models. This is because of this configuration yielding the lowest EER. However, SVM should not be disregarded and is a valid option which can be further researched. The drawback in accuracy from using the web as a platform is deemed to not be severe enough to disregard touchscreen biometrics on the web. Because of the the ease of distributing using the web as a platform, it should be regarded as a viable option for usage in real systems, and warrants more research.

It is determined that the methods presented in this thesis for doing touchscreen biometrics are accurate enough to be used in a real-world scenario, however not as a single means of authentication. With an EER of 5.6 %, the system run the risk of impacting usability since legitimate users will sometimes be classified as intruders. Since it's hard to tell what EER would be suited, the best approach is deemed to evaluate the system in a real application and perform studies of user behaviour and collect feedback to find an appropriate error rate. The usage of critical operations are recommended to prevent intrusions where the attacker shuts down tracking by disabling JavaScript in the browser, as well as giving the system time to gather data for a classification. Critical operations in this context are operations which are defined as having severe impact to the user herself or system.

Although it warrants more research, observation selection using the local outlier factor algorithm as described in this thesis work is not recommended for usage. This is due to the long computation time for using the local outlier factor for large data sets, and it did not yield conclusively lower EER than random selection. Contextual models yielded a lower EER than using simple models, and can as opposed to LOF selection be recommended for usage as well as future work.

# Bibliography

- [1] C. M. Bishop. *Pattern Recognition and Machine Learning [pdf]*. Springer Science+Business Media, LLC, <http://www.rmki.kfki.hu/~banmi/elte/Bishop%20-%20Pattern%20Recognition%20and%20Machine%20Learning.pdf>, 2006.
- [2] C. Bo\*, L. Zhang†, and X.-Y. Li\*. *SilentSense: Silent User Identification via Dynamics of Touch and Movement Behavioral Biometrics [pdf]*. \*Department of Computer Science, Illinois Institute of Technology, USA and †Department of Software Engineering, Tsinghua University, China PR, <http://arxiv.org/pdf/1309.0073.pdf>, 2013.
- [3] M. M. Breunig†, H.-P. Kriegel†, R. T. Ng\*, and J. Sander†. *LOF: Identifying Density-Based Local Outliers [pdf]*. †Institute for Computer Science, \*Department of Computer Science, <http://www.dbs.ifi.lmu.de/Publikationen/Papers/LOF.pdf>, 2000.
- [4] P. Cunningham and S. J. Delany. *k-Nearest Neighbour Classifiers*. \*University College Dublin and †Dublin Institute of Technology, <https://www.csi.ucd.ie/files/UCD-CSI-2007-4.pdf>, 2007.
- [5] E. Ecma International. *Ecmascript language specification*. Website, 2011. <http://www.ecma-international.org/publications/files/ECMA-ST/Ecma-262.pdf>.
- [6] T. Feng, J. Yang, Z. Yan, E. M. Tapia, and W. Shi. *TIPS: Context-Aware Implicit User Identification using Touch Screen in Uncontrolled Environments [pdf]*. Association for Computer Machinery, [http://www2.cs.uh.edu/~tfeng3/context\\_aware\\_touch.pdf](http://www2.cs.uh.edu/~tfeng3/context_aware_touch.pdf), 2014.
- [7] J. Ferreira, H. Santos, and B. Patrão. *Intrusion Detection Through Keystroke Dynamics [pdf]*. University of Minho, Braga, Portugal, [http://www.researchgate.net/publication/230743789\\_Intrusion\\_Detection\\_Through\\_Keystroke\\_Dynamics](http://www.researchgate.net/publication/230743789_Intrusion_Detection_Through_Keystroke_Dynamics), 2014.
- [8] M. Frank, R. Biedert†, E. Ma†, I. Martinovic\*, and D. Song. *Touchalytics: On the Applicability of Touchscreen Input as a Behavioral Biometric*

## BIBLIOGRAPHY

- for *Continuous Authentication [pdf]*. UC Berkeley, †German Research Center for Artificial Intelligence (DFKI) GmbH, \*University of Oxford, <http://www.mariofrank.net/paper/touchalytics.pdf>, ©2012 IEEE, 2012.
- [9] F. Halakou. *Feature Selection in Keystroke Dynamics Authentication Systems [pdf]*. Department of Computer and Electronics, Islamic Azad University, Kalale Branch, Kalale, Iran, [http://www.academia.edu/5343490/Feature\\_Selection\\_in\\_Keystroke\\_Dynamics\\_Authentication\\_Systems](http://www.academia.edu/5343490/Feature_Selection_in_Keystroke_Dynamics_Authentication_Systems), 2013.
- [10] A. B. Hassanat\*, M. A. Abbadi\*, G. A. Altarawneh\*, and A. A. Alhasanat†. *Solving the Problem of the K Parameter in the KNN Classifier Using an Ensemble Learning Approach [pdf]*. \*Mu'tah University, †Al-Hussein Bin Talal University, <http://arxiv.org/ftp/arxiv/papers/1409/1409.0919.pdf>, 2014.
- [11] C.-W. Hsu, C.-C. Chang, and C.-J. Lin. *A Practical Guide to Support Vector Classification*, 2003, Last updated 2010. <http://www.csie.ntu.edu.tw/~cjlin/papers/guide/guide.pdf>.
- [12] N. Imagery and N. p. Mapping Agency. *National Imagery and Mapping Agency Technical Report 8350.2, Third Edition*. National Imagery And Mapping Agency, <http://earth-info.nga.mil/GandG/publications/tr8350.2/wgs84fin.pdf>, 2000.
- [13] G. James, D. Witten, T. Hsastie, and R. Tibshirani. *An Introduction to Statistical Learning [pdf]*. Springer, Springer New York Heidelberg Dordrecht London, 2013. <http://www-bcf.usc.edu/~garth/ISL/ISLR%20Fourth%20Printing.pdf>.
- [14] J.-K. Kang. machine learning. Website, 2014. [https://www.npmjs.com/package/machine\\_learning](https://www.npmjs.com/package/machine_learning).
- [15] S. M. Kolly, R. Wattenhofer, and S. Welten. *A Personal Touch - Recognizing Users Based on Touch Screen Behavior [pdf]*. ETH Zurich, <http://www.tik.ee.ethz.ch/file/a2d39bdadad07bf0a2a7f0af2dac7377/paper.pdf>, 2008.
- [16] L. Li, X. Zhao, and G. Xue. *Unobservable Re-authentication for Smartphones [pdf]*. Arizona State University, Tempe, USA, [http://www.internetsociety.org/sites/default/files/02\\_1\\_0.pdf](http://www.internetsociety.org/sites/default/files/02_1_0.pdf), 2013.
- [17] H. W. Lie, T. Atkins, and E. J. Etemad. *Css values and units module level 3*. Website, 2013. <http://www.w3.org/TR/css3-values/#reference-pixel>.
- [18] A. Ng. *Support Vector Machines [pdf]*. Stanford University, n.d. <http://cs229.stanford.edu/notes/cs229-notes3.pdf>.
- [19] J. Nielsen. Horizontal attention leans left. Webiste, 2013. <http://www.nngroup.com/articles/horizontal-attention-leans-left/>.

## BIBLIOGRAPHY

- [20] S. Raschka. About feature scaling and normalization. Website, 2014. [http://sebastianraschka.com/Articles/2014\\_about\\_feature\\_scaling.html](http://sebastianraschka.com/Articles/2014_about_feature_scaling.html).
- [21] J. Rossi and M. Brubeck. Pointer events. Website, 2015. <http://www.w3.org/TR/pointerevents/>.
- [22] H. Saevanee and P. Bhatarakosol. *User Authentication using Combination of Behavioral Biometrics over the Touchpad acting like Touch screen of Mobile Device [pdf]*. Chulalongkorn University, Thailand, <http://lsia.fi.uba.ar/papers/saevanee08.pdf>, 2008.
- [23] K. Scarfone and P. Mell. *Guide to Intrusion Detection and Prevention Systems [pdf]*. National Institute of Standards and Technology, U.S. Department of Commerce, <http://csrc.nist.gov/publications/nistpubs/800-94/SP800-94.pdf>, 2007. [Accessed 4 April 2015].
- [24] D. Schepers, S. Moon, and M. Brubeck. Touch events specification. Website, 2011. <http://www.w3.org/TR/2011/WD-touch-events-20110505/>.
- [25] D. Schepers, S. Moon, M. Brubeck, and A. Barstow. Touch events. Website, 2013. <http://www.w3.org/TR/touch-events/>.
- [26] H. Shimodaira. *Classification and K-nearest neighbours [pdf]*. The University of Edinburgh, <http://www.inf.ed.ac.uk/teaching/courses/inf2b/learnnotes/inf2b-learn-note04-2up.pdf>, 2015.
- [27] S. Statistiska centralbyrån. Internetanslutning via mobiltelefon. Website, 2014. [http://www.statistikdatabasen.scb.se/pxweb/en/ssd/START\\_\\_LE\\_\\_LE0108\\_\\_LE0108C/LE0108T09/?rxid=6887f0ae-0a9d-4da1-8415-cbaad59f8c88](http://www.statistikdatabasen.scb.se/pxweb/en/ssd/START__LE__LE0108__LE0108C/LE0108T09/?rxid=6887f0ae-0a9d-4da1-8415-cbaad59f8c88).
- [28] P. S. Tah, A. B. J. Teoh, and S. Yue. A survey of keystroke dynamics biometrics. Website, 2013. <http://www.ncbi.nlm.nih.gov/pmc/articles/PMC3835878/>.
- [29] R. Tibbett, T. Volodine, S. Block, and A. Popescu. Deviceorientation event specification. Website, 2014. <http://w3c.github.io/deviceorientation/spec-source-orientation.html>.
- [30] W. Walmsely. Taps and swipes: Intuition vs. machine learning in ux design. Website, 2014. <http://minuum.com/taps-and-swipes/>.
- [31] H. Xu, Y. Zhou, and M. R. Lyu. *Towards Continuous and Passive Authentication via Touch Biometrics: An Experimental Study on Smartphones [pdf]*. The Chinese University of Hong Kong, <https://www.usenix.org/system/files/conference/soups2014/soups14-paper-xu.pdf>, 2014.

## BIBLIOGRAPHY

- [32] N. Zheng\*, H. H. Kun Bai†, and H. Wang\*. *You Are How You Touch: User Verification on Smartphones via Tapping Behaviors [pdf]*. \*College of William and Mary, †IBM T.J. Watson Research Center, <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.419.2490&rep=rep1&type=pdf>, 2012.

## Appendix A

# Experiment

**Training stage**

To build your profile, you should complete the small set of tasks once every day for one week. It is important that you use **the same device** for the all of the test instances. Remember to keep your device upright when doing the test!

Wobbuffet

Password

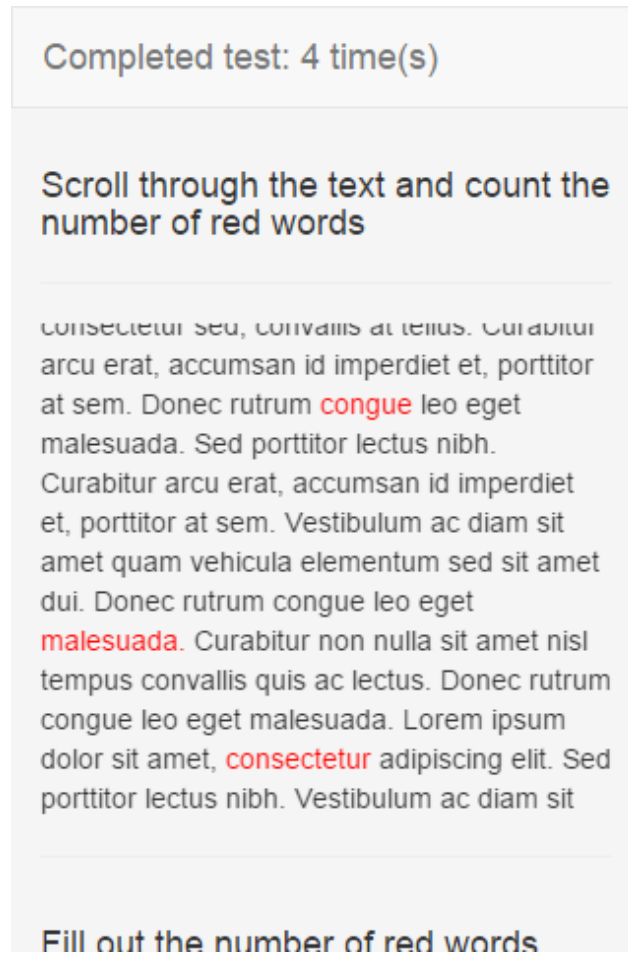
Submit

*Haven't got a profile? [Register here.](#)*

Keep scrolling down! :)

Have a look at some pretty pictures

**Figure A.1.** The participants in the first phase of the experiment were greeted with this screen, where one could login to an existing account or create a new one.



**Figure A.2.** The first part of the experiment consisted of scrolling through a text in order to collect stroke gestures. A small task of counting red words were given to trigger the scrolling behaviour and attempt to make users not ignore the text and skip it.


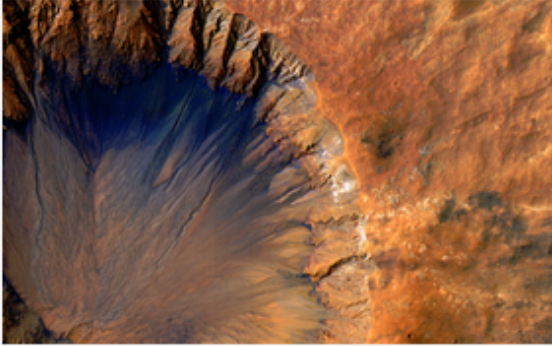


Fill out the number of red words.

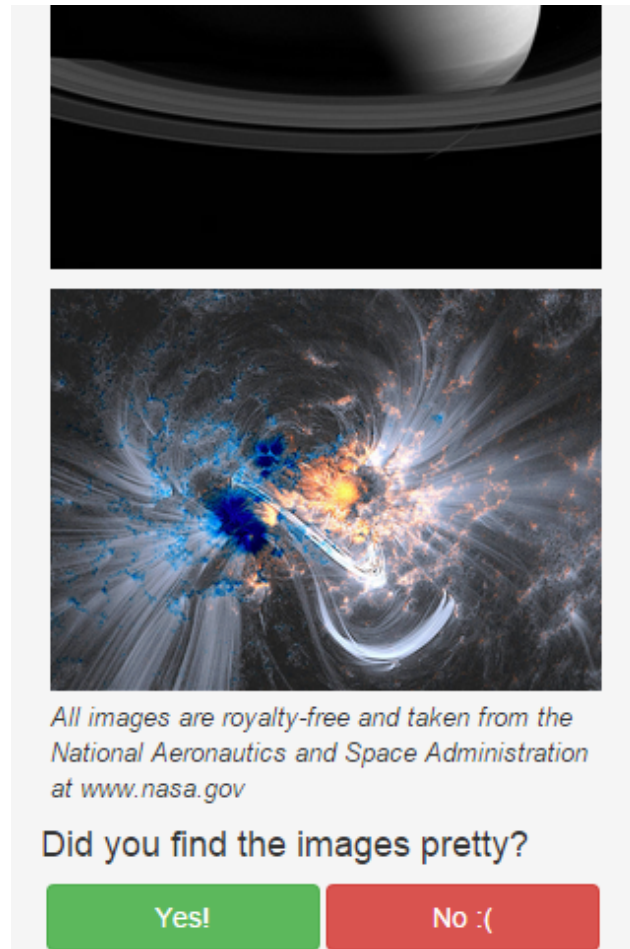
Enter the following code: 9-0-4-1-7

Keep scrolling down! :)

Have a look at some pretty pictures



**Figure A.3.** Input fields after the word count serving as elements for which taps could be recorded, as well as giving the user a sense of importance since the first field connects to the previous exercise. This was followed by a small image gallery.



**Figure A.4.** The purpose of the image gallery was to provide a more lean scrolling experience where the user didn't have to perform a task. The test ended with a simple question designed to put an end to the test as well as yet another element which triggers user interaction.

## Appendix B

### Experiment participants

User	Sessions for training	Excessive sessions
User 1	7	2
User 2	7	4
User 3	7	2
User 4	7	8
User 5	7	1
User 6	7	1
User 7	7	1
User 8	4	0
User 9	5	1
User 10	7	1
User 11	5	1
User 12	4	0
User 13	5	0
User 14	5	0
User 15	1	0
User 16	7	0
User 17	5	3
User 18	5	0

**Table B.1.** This table shows a list of the different experiment participants of the first phase and their number of sessions. Only 11 users had any excessive sessions which could be used for FRR calculation

## Appendix C

# Complete results for all configurations

**Table C.1.** A complete list of results showing all test runs with all configurations from all training sets. BWE is short for backwards elimination, and CM is short for contextual models. Norm. stands for normalization

Algorithm	Selection	Norm.	CM	BWE	FAR	FRR	EER
Esm. k-NN	LOF	No	Yes	No	12.5 %	9.8 %	11.2 %
Esm. k-NN	LOF	Yes	Yes	No	12.5 %	11.7 %	12.1 %
Esm. k-NN	LOF	No	No	No	12.5 %	12.5 %	12.5 %
Esm. k-NN	LOF	Yes	No	No	20.8 %	23.9 %	22.3 %
1-NN	LOF	No	Yes	No	8.3 %	8.3 %	8.3 %
1-NN	LOF	Yes	Yes	No	16.7 %	16.3 %	16.5 %
1-NN	LOF	No	No	No	16.7 %	19.3 %	18 %
1-NN	LOF	Yes	No	No	25 %	25.4 %	25.2 %
Esm. k-NN	LOF	No	Yes	Yes	8.3 %	12.9 %	10.6 %
Esm. k-NN	LOF	Yes	Yes	Yes	12.5 %	12.9 %	12.7 %
Esm. k-NN	LOF	No	No	Yes	12.5 %	11.4 %	11.9 %
Esm. k-NN	LOF	Yes	No	Yes	20.8 %	21.6 %	21.2 %
1-NN	LOF	No	Yes	Yes	20.8 %	20.5 %	20.6 %
1-NN	LOF	Yes	Yes	Yes	12.5 %	13.6 %	13.1 %
1-NN	LOF	No	No	Yes	16.7 %	16.3 %	16.5 %
1-NN	LOF	Yes	No	Yes	33.3 %	32.2%	32.8 %
Esm. k-NN	Random	No	Yes	No	4.2 %	5.7 %	4.9 %
Esm. k-NN	Random	Yes	Yes	No	16.7 %	18.9 %	17.8 %
Esm. k-NN	Random	No	No	No	12.5 %	14 %	13.3 %
Esm. k-NN	Random	Yes	No	No	20.8 %	20.5 %	20.6 %
1-NN	Random	No	Yes	No	8.3 %	09.1 %	8.7 %
1-NN	Random	Yes	Yes	No	20.8 %	17.4 %	19.1 %
1-NN	Random	No	No	No	12.5 %	14 %	13.3 %
1-NN	Random	Yes	No	No	29.2 %	29.2 %	29.2 %
Esm. k-NN	Random	No	Yes	No	12.5 %	10.2 %	11.4 %
Esm. k-NN	Random	Yes	Yes	No	29.2 %	27.7 %	28.4 %

Esm. k-NN	Random	No	No	No	8.3 %	10.6 %	09.5 %
Esm. k-NN	Random	Yes	No	No	20.8 %	21.2 %	21 %
1-NN	Random	No	Yes	No	16.7 %	16.3 %	16.5 %
1-NN	Random	Yes	Yes	No	33.3 %	33.3 %	33.3 %
1-NN	Random	No	No	No	16.7 %	17.4 %	17 %
1-NN	Random	Yes	No	No	25 %	24.2 %	24.6 %
Esm. k-NN	Random	No	Yes	No	8.3 %	8.3 %	8.3 %
Esm. k-NN	Random	Yes	Yes	No	25 %	23.5 %	24.2 %
Esm. k-NN	Random	No	No	No	12.5 %	13.6 %	13.1 %
Esm. k-NN	Random	Yes	No	No	20.8 %	21.2 %	21 %
1-NN	Random	No	Yes	No	12.5 %	9.8 %	11.2 %
1-NN	Random	Yes	Yes	No	29.2 %	29.5 %	29.4 %
1-NN	Random	No	No	No	16.7 %	16.7 %	16.7 %
1-NN	Random	Yes	No	No	25. %	26.9 %	25.9 %
Esm. k-NN	Random	No	Yes	No	12.5 %	11.7 %	12.1 %
Esm. k-NN	Random	Yes	Yes	No	20.8 %	19.3 %	20.1 %
Esm. k-NN	Random	No	No	No	12.5 %	11.7 %	12.1 %
Esm. k-NN	Random	Yes	No	No	20.8 %	22.7 %	21.8 %
1-NN	Random	No	Yes	No	16.7 %	15.2 %	15.9 %
1-NN	Random	Yes	Yes	No	29.2 %	28.4 %	28.8 %
1-NN	Random	No	No	No	12.5 %	12.5 %	12.5 %
1-NN	Random	Yes	No	No	25. %	25.4 %	25.2 %
Esm. k-NN	Random	No	Yes	No	8.3 %	8.3 %	8.3 %
Esm. k-NN	Random	Yes	Yes	No	20.8 %	23.5 %	22.2 %
Esm. k-NN	Random	No	No	No	8.3 %	9.8 %	9.1 %
Esm. k-NN	Random	Yes	No	No	20.8 %	20.8 %	20.8 %
1-NN	Random	No	Yes	No	16.7 %	11.7 %	14.2 %
1-NN	Random	Yes	Yes	No	25. %	24.6 %	24.8 %
1-NN	Random	No	No	No	16.7 %	14.8 %	15.7 %
1-NN	Random	Yes	No	No	25. %	25.8 %	25.4 %
SVM	LOF	No	Yes	No	12.5 %	10.2 %	11.4 %
SVM	LOF	No	No	No	12.5 %	12.5 %	12.5 %
Esm. k-NN	Random	No	Yes	Yes	4.2 %	4.2 %	4.2 %
Esm. k-NN	Random	Yes	Yes	Yes	20.8 %	21.6 %	21.2 %
Esm. k-NN	Random	No	No	Yes	12.5 %	12.9 %	12.7 %
Esm. k-NN	Random	Yes	No	Yes	16.7 %	15.9 %	16.3 %
1-NN	Random	No	Yes	Yes	4.2 %	4.9 %	4.5 %
1-NN	Random	Yes	Yes	Yes	20.8 %	16.7 %	18.8 %
1-NN	Random	No	No	Yes	8.3 %	8.3 %	8.3 %
1-NN	Random	Yes	No	Yes	16.7 %	20.8 %	18.8 %
Esm. k-NN	Random	No	Yes	Yes	4.2 %	3 %	3.6 %
Esm. k-NN	Random	Yes	Yes	Yes	29.2 %	27.7 %	28.4 %
Esm. k-NN	Random	No	No	Yes	8.3 %	7.6 %	8 %
Esm. k-NN	Random	Yes	No	Yes	25 %	22.7 %	23.9 %

APPENDIX C. COMPLETE RESULTS FOR ALL CONFIGURATIONS

1-NN	Random	No	Yes	Yes	4.2 %	5.3 %	4.7 %
1-NN	Random	Yes	Yes	Yes	25 %	27.7 %	26.3 %
1-NN	Random	No	No	Yes	12.5 %	15.5 %	14 %
1-NN	Random	Yes	No	Yes	25 %	25.8 %	25.4 %
Esm. k-NN	Random	No	Yes	Yes	4.2 %	3.8 %	4 %
Esm. k-NN	Random	Yes	Yes	Yes	16.7 %	17.4 %	17 %
Esm. k-NN	Random	No	No	Yes	4.2 %	6.1 %	5.1 %
Esm. k-NN	Random	Yes	No	Yes	20.8 %	19.7 %	20.3 %
1-NN	Random	No	Yes	Yes	4.2 %	4.9 %	4.5 %
1-NN	Random	Yes	Yes	Yes	16.7 %	15.5 %	16.1 %
1-NN	Random	No	No	Yes	12.5 %	12.1 %	12.3 %
1-NN	Random	Yes	No	Yes	29.2 %	27.7 %	28.4 %
Esm. k-NN	Random	No	Yes	Yes	8.3 %	8 %	8.1 %
Esm. k-NN	Random	Yes	Yes	Yes	12.5 %	13.3 %	12.9 %
Esm. k-NN	Random	No	No	Yes	8.3 %	6.1 %	7.2 %
Esm. k-NN	Random	Yes	No	Yes	16.7 %	15.5 %	16.1 %
1-NN	Random	No	Yes	Yes	12.5 %	11.7 %	12.1 %
1-NN	Random	Yes	Yes	Yes	16.7 %	17 %	16.9 %
1-NN	Random	No	No	Yes	12.5 %	12.5 %	12.5 %
1-NN	Random	Yes	No	Yes	29.2 %	30.3 %	29.7 %
Esm. k-NN	Random	No	Yes	Yes	8.3 %	8. %	8.1 %
Esm. k-NN	Random	Yes	Yes	Yes	20.8 %	20.5 %	20.6 %
Esm. k-NN	Random	No	No	Yes	4.2 %	6.1 %	5.1 %
Esm. k-NN	Random	Yes	No	Yes	16.7 %	18.9 %	17.8 %
1-NN	Random	No	Yes	Yes	4.2 %	6.4 %	5.3 %
1-NN	Random	Yes	Yes	Yes	20.8 %	22.3 %	21.6 %
1-NN	Random	No	No	Yes	8.3 %	8.3 %	8.3 %
1-NN	Random	Yes	No	Yes	29.2 %	29.2 %	29.2 %
SVM	Random	No	Yes	No	0.083 %	0.076 %	0.080 %
SVM	Random	No	No	No	0.042 %	0.049 %	0.045 %
SVM	Random	No	Yes	No	0.125 %	0.098 %	0.112 %
SVM	Random	No	No	No	0.083 %	0.080 %	0.081 %
SVM	Random	No	Yes	No	0.042 %	0.042 %	0.042 %
SVM	Random	No	No	No	0.042 %	0.045 %	0.044 %
SVM	Random	No	Yes	No	0.042 %	0.064 %	0.053 %
SVM	Random	No	No	No	0.083 %	0.091 %	0.087 %
SVM	Random	No	Yes	No	0.042 %	0.057 %	0.049 %
SVM	Random	No	No	No	0.042 %	0.045 %	0.044 %

