DEGREE PROJECT IN COMPUTER SCIENCE AND ENGINEERING, SECOND CYCLE, 30 CREDITS

STOCKHOLM, SWEDEN 2018

Evaluating XGBoost for User Classification by using Behavioral Features Extracted from Smartphone Sensors

ZIAD SALAM PATROUS

KTH ROYAL INSTITUTE OF TECHNOLOGY
SCHOOL OF ELECTRICAL ENGINEERING AND COMPUTER SCIENCE

Evaluating XGBoost for User Classification by using Behavioral Features Extracted from Smartphone Sensors

ZIAD PATROUS

Master in Computer Science
Date: July 3, 2018
Supervisor: Elena Troubitsyna
Examiner: Mads Dam
Swedish title: Utvärdering av XGBoost för användarklassificering med hjälp av beteendebaserade attribut utvunna från sensorer i mobiltelefonen
School of Computer Science and Communication

Abstract

Smartphones have opened the possibility to interact with people anytime and anywhere. A significant amount of individuals rely on their smartphone for work-related and everyday tasks. As a consequence modern smartphones include sensitive, valuable and confidential information, such as e-mails, photos, notes, and messages. The primary concern is to prevent unauthorized access to data stored on the smartphone and applications. Traditional authentication methods are entry-point based and do not support continuous authorization. Therefore, as long as the session is active, there are no mechanisms to assure that it involves the same authorized user. This thesis studies the concept of continuous authentication, an authentication approach used to assure authorization periodically. Furthermore, we discuss behavioral biometric and attributes useful for continuous authentication, and investigates Extreme Gradient Boosting (XGBoost) for user classification by using behavioral features extracted from the mobile sensors Accelerometer, Gyroscope, and Magnetometer.

Experimental results show that using XGBoost, an average Equal Error Rate of 14,7% was received using ninety users. Furthermore, experiments were performed using different sensor combination and testing on specific activities.

Sammanfattning

Mobiltelefoner har gjort det möjligt att interagera med andra människor när som helst och var som helst. En betydande mängd människor är beroende av sin mobiltelefon för arbetsrelaterade och vardagliga uppgifter. Som en konsekvens inkluderar moderna mobiltelefoner känslig, värdefull och konfidentiell information, exempelvis e-postmeddelanden, foton, anteckningar och meddelanden. Det främsta problemet är att hindra utomstående åtkomst till data lagrad på mobiltelefonen och i applikationerna. Traditionella autentiseringsmetoder är entry-point baserade. Det finns inga metoder som används regelbundet för att säkerställa autentisering så länge sessionen är aktiv. Denna avhandling studerar begreppet kontinuerlig autentisering, en autentiseringsmetod som används för att regelbundet säkerställa att rätt användare brukar mobiltelefonen. Vidare diskuteras beteendebaserad biometri och attribut som är användbara för kontinuerlig autentisering, samt en undersökning om användarklassificering med hjälp av Extreme Gradient Boosting (XGBoost) och beteendebaserade attribut framtagna med hjälp av följande mobilsensorer: Accelerometer, Gyroskop, och Magnetometer

Resultaten av utförda experiment visar att XGBoost får en genomsnittlig Equal Error Rate på 14,7 % med nittio användare. Vidare utfördes experiment med användning av olika sensorkombinationer och testning på specifika aktiviteter.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

This chapter introduces the motivation behind this study, presents the problem statement, research questions, and the objective. Furthermore, it defines the delimitations, and finally presents the outline of this thesis.

1.1 Introduction

Modern smartphones are considered to be personal assistants. They are used to both store and access personal and work-related information. Therefore, securing the smartphone from intruders and unauthorized access is of significant concern.

Smartphone users have traditionally been authenticated using PIN-codes, passwords, and patterns. In recent years, biometrics in smartphones have become an accepted method for authentication. In forecast research published by ABI, authors state that 95% of smartphones shipped in 2022 will include a fingerprint sensor. The authors also state that the growing acceptance will lead to other types of biometrics solutions [1]. Recent releases from mobile giants such as Apple, Samsung, and Huawei have included several types of biometric authentication methods, such as fingerprint, iris, and face recognition showing clear indications of this trend. Even though this improves security, there are still some remaining concerns.

Commonly, the authentication is performed once at the entry-point. Once the user is authenticated, there is no mechanism to reassure that the correct user remains throughout the active session, which implies a security risk. Therefore, it is necessary to authenticate the user periodically. Due to the inconvenience of repeatedly typing the password or providing a fingerprint, reliance on behavioral biometric is a possible solution for continuously authenticating the user.

The use of behavioral biometric has a long history. In World War Two the telegraph operators developed a unique rhythm for transmitting Morse code. It was used by the military intelligence to distinguish allies from enemies. Rythm traits have been the underlying concept for keystroke dynamics, a way to identify or authenticate an individual by examining typing behavior on the keyboard. There is substantial research made showing the possibilities of user authentication using only keystroke dynamics [21, 41]. One approach for applying the same underlying concept on smartphones is to utilize the vast amount of sensors that continuously captures data.

Behavioral biometrics make use of behavioral attributes of an individual, mainly monitoring how the individual interacts with the technology, to build a behavioral profile. With the accelerated research and success of machine learning, it is now possible to learn behavioral profiles with good result [2]. However, increasing accuracy and evaluating different methods is still of significant interest.

In this thesis, the focus is primarily on the machine learning aspect of behavioral biometrics, more specifically applying Extreme Gradient Boosting (XGBoost), a gradient boosting approach, to classify users as either genuine or imposters. The impact of XGBoost has been widely recognized in many machine learning and data mining challenges on websites such as Kaggle [22]. Using XGBoost have shown state-of-the-art results for numerous problems such as web text classification, customer behavior prediction, motion detection, and malware classification [7, 28].

1.2 Problem Statement

The traditional authentication methods fail to detect an intruder when the attacker passes through the point of entry. Furthermore, requiring users to reauthenticate using interrupting authentication methods is cumbersome and can lead to users deactivating their security. A solution to this is to add another layer of authentication, running in the background, and continuously reassuring authorization without interrupting the user. This concept is called continuous authentication. Machine learning enables the possibility to utilize existing sensors from smartphones to build a behavioral profile for users. With the goal of providing additional security, it is crucial for the authentication system to improve consistently. With the accelerated advancement of smartphones as well as machine learning, it is of significant interest to study alternative solutions. This thesis aims at contributing to this goal by investigating the research questions:

Can XGBoost be utilized to provide state-of-the-art results for user classification using sensor data captured from smartphones?

How do different sensor combination affect the outcome of the user classification and which sensor combination provides the best performance?

How do different activities performed during device usage affect the user classification?

1.3 Objective

In this thesis, the central focus is on the classification part of a behavioral biometric system. The primary objective is to evaluate the performance of applying XGBoost to classify users. To the best of our knowledge, XGBoot has not been applied to this problem in any study. Moreover, additional objectives of this thesis are also to investigate how different sensor combinations and activities might affect the outcome of the model. The conclusions of this thesis may be useful for anyone interested in building a model for classifying users with the help of mobile sensors.

1.4 Delimitations

This thesis will not examine how to implement an application that captures data and authenticates users. The primary purpose is to evaluate whether XGBoost can be used to classify users with good results. Therefore, a publicly available dataset is used to evaluate how well the model performs. The thesis will only consider the dataset provided by Yang et al. [40]. The dataset includes data from several sensors. This thesis will examine the accelerometer, magnetometer and gyroscope sensors, due to time limitations. Using a public dataset, it is possible to combine several sensors in future work as well as make it comparable to other models. However, the results may differ using different dataset due to noise and environment factors when collecting the data. Authentication sensitivity depending on in-device usage will not be studied, due to limitations in the dataset. The classification will be made similar disregarding of what activity is performed. For evaluation, a neural network and a Support Vector Machine classifier will also be trained to compare the results with XGBoost. It is important to note that the focus of the thesis will primarily be on using XGBoost. The other two models will only be used as a reference during the evaluation.

1.5 Ethics & Sustainability

The underlying factor in continuous authentication described in this project builds on learning human behavior. The information used to learn behavioral profiles for individuals is collected by gathering data that does not necessarily require user permission to collect. Profiling user behavior using multiple modalities is profoundly connected to the motor cortex of humans and is hard to mimic or change. As the behavioral profiles become more advanced and accessible, parties can misuse it for surveillance and monetizing purposes.

The purpose of this thesis is to evaluate the performance of applying XGBoost on behavioral attributes from multiple sensors. The results are heavily depended on the dataset used. It is possible that results achieved in this thesis are not reproducible using another dataset. Also, due to advancements in smartphones newly gathered data using newer smartphone devices might better capture movement with less noise and therefore provide better results. In contrary to the results, the methodology used and decisions taken can be re-used in future research or software. Therefore, it is necessary to be open with the steps taken and give a detailed explanation of how the experiments were prepared and conducted.

1.6 Outline

The disposition is organized as follows: Chapter 2 introduces relevant background about authentication, mobile sensors, behavioral biometrics, and finally, an overview of the related work. Chapter 3 describes the methodology. The chapter includes data analysis followed by a description of how the models are trained. Chapter 4 presents the results achieved by running the experiments. Chapter 5 includes discussion of the results and future work. Chapter 6 concludes the thesis.

Chapter 2

Background

This chapter provides relevant theoretical knowledge essential for understanding the problem and techniques used in the thesis. It includes a description of different authentication methods, mobile sensors embedded in smartphones, behavioral biometrics, and biometric systems. It is followed by a brief explanation of the machine learning models used and related work.

2.1 Authentication

This section contains a description of the differences between identification, verification, and authentication, followed by a description of biometric authentication. Finally, the section introduces the concept of continuous authentication and what problems it solves.

2.1.1 Identification, Verification and Authentication

Authentication: *an act, process, or method of showing something (such as an identity, a piece of art, or a financial transaction) to be real, true, or genuine.* [3]

Identification, verification, and authentication tend to be mixed up. The terms are most easily explained using real-life scenarios. In biometric systems identification is the process of seeking an identity. For example at an airport, a traveler might provide a fingerprint to be identified. To identify the traveler, the fingerprint is compared to a set of fingerprints stored in a database, resulting in a one-to-many comparison. However, because of the comparisons made, this process is very inefficient if the database is extensive. Verification is the process of verifying a single identity. For example, the traveler at the airport first provides his or her passport

to be fetched from the database and then provides the fingerprint to verify that the person indeed gave the correct passport. This results in a one-to-one comparison. As a result, the individual is permitted to pass through the checkpoint. Thus, it is said that the individual is authenticated [10].

To summarize, the main difference between the identification and verification process is that identification is a one-to-many procedure. Meanwhile, verification is a one-to-one procedure [10]. Throughout this thesis, the term authentication will be used when referring to the process of giving a genuine user an access to a system.

Three general approaches for authenticating a user are [21, 11, 32]:

- Knowledge-based: What you know. Cognitive information.
- Possession-based: What you have. Items such as smart cards.
- Inference-based: What you are. Physiological and behavioral attributes.

The traditional approach for authentication requires users to remember, create, and manage long and complex passwords. In smartphones, a user is authenticated several times during the day. Therefore, using long and complex passwords potentially imposes the risk of individuals deactivating their security. In smartphones, the traditional approach is made less cumbersome by using knowledge-based authentication methods such as 4-digit Personal Index Number (PINs) or pattern passwords. However, these methods are subjects to several types of attacks, such as [2]:

- *Smudge attack*: a type of attack utilizing the smudge left by fingers on the touchscreen. In a study made by Aviv et al. [4], the feasibility of smudge attacks was examined on pattern-based passwords. In one of the experiments conducted, the authors showed that by using photographs taken under a variety of lighting sources, they could identify 68% of the pattern passwords.
- *Shoulder surfing attack*: a type of social engineering attack used to obtain passwords, PINs and other personal information by looking over the shoulder of the victim.

To avoid such attacks and make authentication more user-friendly, there is a trend shift towards smartphones using inference-based authentication, more commonly referred to as biometric authentication, to improve the security. [1].

2.1.2 Biometric Authentication

Biometric authentication or biometrics is defined as follows: *The measurement and analysis of unique physical or behavioral characteristics primarily as a means of verifying personal identity* [5]. In many cases, biometric authentication provides a more secure way to authenticate users. One of the main advantages of using biometrics is that it cannot be easily guessed by attackers. It makes the authentication process unique and more user-friendly. However, as with anything, there are also some disadvantages. Biometric data is not replaceable, unlike passwords. Therefore many privacy concerns arise. One of main privacy concern is that no database is secure enough and is always under risk of data leakage. Such data leakage can result in confidential information becoming available to the public. For this reason, a considerable amount of research is made to explore non-intrusive characteristics for biometric authentication [11].

The biometric characteristics, or modalities, are generally divided into two categories: *physiological biometrics* and *behavioral biometrics*. Physiological biometrics is based on physical attributes of an individual, such as fingerprint, iris, palm print, and facial features. Behavioral biometrics is connected to the way individuals behave, such as gait recognition, keystroke dynamics, hand-typing, and voice recognition [32, 11].

Current smartphones using both traditional and biometric authentication are mainly based on entry-point authentication. As long as the session is kept active, there are no mechanisms to verify that the same authenticated individual is using the smartphone. Thus if an intruder has passed through the entry-point, no security measurements prevent the user from using the smartphone, except if provided by applications [2, 12].

Continuous authentication is a concept seeking to address this problem by re-authenticating the user utilizing non-interrupting methods.

2.1.3 Continuous Authentication

Continuous authentication, also known as, implicit, passive or active authentication is the process of periodically assuring authorization. The goal of continuous authentication systems is to use non-interruptive methods to add an additional security level during phone usage. Majority of the authentication mechanisms used in smartphones requires a user's full attention [23], for example PINs and fingerprint recognition. However, it is not convenient to require a user's full

attention to ensure authorization periodically. Therefore, it is beneficial to use non-interruptive methods. A possible solution is to learn behavioral attributes in how the individual interacts with the smartphone. Using behavioral attributes is shown to be the most suitable approach for continuous authentication [30].

Behavior-based Continuous Authentication

A behavior-based continuous authentication approach for smartphones aims at building a behavioral profile, by examining patterns in data captured from several channels in the mobile device. If significant fluctuations between the captured data and the actual data appear, a potential intrusion can be detected [30]. This can be achieved by monitoring the data captured from mobile sensors over a time-period. In a behavior-based approach, it is not as common to use a single source of data for profiling. Instead, multiple sensors are used to build a stronger user profile [30, 38].

The characteristic of a good continuous authentication system includes [11]:

- Non-intrusive: The information gathered for authentication should not be intrusive. Furthermore, the system must authenticate users in a seamless and non-interruptive manner.
- Behavioral attributes: The authentication system should utilize already existing hardware to collect and learn behavior profiles. This approach results in easily integrated cost-effective authentication systems. Behavioral biometrics is described in section 2.3.
- System independent: The system should be able to function as a security measurement in applications or be integrated into the device itself.
- Fast response and energy-efficient: Continuous authentication requires constant background computations. Therefore, it is crucial to have a system that does not require a significant amount of energy and computational power.

In summary, a continuous authentication solution should be fast, robust, require minimal hardware, and easy to integrate [11, 2].

2.2 Mobile Sensors

A sensor measures physical quantities and converts it into signals which can be read by an observer or by an instrument [16]. The sensitivity of a sensor controls how much the output value alters in relation to the measured quantity value. The sensors are used every day in different systems and scenarios. Some examples are thermometers, radar guns, automatic door openers, cameras, GPS, vehicle systems and traffic lights [16].

Mobile devices have always utilized sensors such as microphones to convert sound to digital signals. However, sensors are becoming smaller, faster cheaper and more accurate. The modern smartphones are getting packed with sensors such as GPS, compass, and proximity. Making the smartphones "smart" [16].

Since the data used in this project is collected using an Android device, sensors available to collect in Android will only be discussed. The sensors supported in the Android platform are position sensors, environmental sensors and motion sensors. Position sensors measure the physical position of a device, such as a Magnetometer. Environmental sensors measure different environmental parameters, such as ambient light. Lastly, motion sensors measure acceleration forces and rotational forces, such as Accelerometer and Gyroscope [18].

For a continuous authentication system, the goal is to capture how an individual interacts with the smartphone, as described in section 2.1.3. The interactions can be extracted from touchscreen usage and sensors that monitor the movement of the smartphone. For this project, we are only interested in sensors that monitor the movement of the device. This can be achieved by the Accelerometer, Gyroscope, and Magnetometer. These sensors are suitable because they are included in the majority of the newer smartphones and have been shown to be sufficient for capturing user behavior [18, 23]. In addition, these sensors do not require user permission for data capturing. Therefore, they are suitable for continuous authentication systems because they are not considered to be intrusive or sensitive [18].

The Accelerometer and the Gyroscope sensors are useful for monitoring device movement. The Accelerometer measures acceleration force in m/s^2 along the x, y, and z-axis, meanwhile Gyroscope measures the device's rotation rate in rad/s along the same three-dimensional axis. Rotation of x, y, and z-axis are referred to as pitch, roll, and yaw respectively. However, to know where the mobile device is in the physical world, it is also necessary to use Magnetometer sensors that

measure the geomagnetic field in microtesla μT along the x, y and z-axis. Using the Magnetometer it is possible to determine a device's position in relative to the magnetic north pole [18].

A challenge in utilizing the sensor data is the noise generated from external factors during data acquisition. Applying filters such as moving average, to smooth and minimize the noise, is a necessary procedure. A moving average filter has been used in other research papers with good result. In papers by Lee and Lee [23] and Ehatisham-Ul-Haq et al. [14] a moving average filter was applied to the data set to reduce the noise. Both papers proposed taking the average of a set of contiguous data points into one data point to reduce the computational complexity as well as the noise.

2.3 Behavioral Biometrics

Behavioral biometric uniquely identifies characteristic that can be acquired from user action. Identifying individuals using behavioral treats dates back as far as the invention of the telegraph in the 1860s. The telegraph messages were sent using Morse code by pressing a key rhythmically. The telegraph operators started to adopt different behaviors on how they send the messages that were recognizable by co-workers. This technique was used in World War II where it allowed allied forces to verify if the message came from allies [35].

A benefit of using behavioral features is that the information is based on the nature of the individual and not on static information. Using several biometric features to build a behavioral template makes it very hard to replicate or steal [35].

2.3.1 Behavioral Features

An essential step for designing a behavioral biometric system and achieving acceptable accuracy is the choice of attributes [11]. In this section, we describe some general behavioral features suitable for smartphones.

Touchscreen Dynamics

Touchscreen dynamics is based on the same concept of Keystroke dynamics. Keystroke dynamics utilizes the rhythm of how an individual types on the keyboard. Similarly, touchscreen dynamics utilizes the rhythm of how an individual interacts with the touchscreen [11, 41]. This type of attributes captures cognitive attributes and subjective preferences of an individual. The gestures captured may include how the user tap, double tap, drag, flick, pinch, rotate, and the duration of each touch [11].

Hand Movement, Orientation, and Grasp

Hand movement, orientation, and grasp capture micro-movements and orientation patterns of how an individual is using the smartphone during different activities. The possibilities of using sensors for behavioral profiling is shown in numerous publications from companies and researchers. In section 2.5 some sensors utilized in research papers are presented. These sensors can be used alone or in combination with touchscreen dynamics to improve the accuracy [11].

The combination of multiple biometric characteristics makes the biometric system more robust and tends to provide better performance and usability [2]. The process of combining multiple biometric features is divided into different fusion strategies, these strategies are [11]:

- *Feature level fusion:* Features extracted from different biometric traits are independent of each other. Thus, the vectors can be combined with each other.
- *Decision level fusion:* Each sensor acquires biometric data and the extracted feature vectors are used individually to classify the claimed identity. A scheme is later applied to make the final decision utilizing the individual classifications.
- *Matching score level fusion:* Every single biometric sub-system provides a score, these scores are later combined for the final assertion.

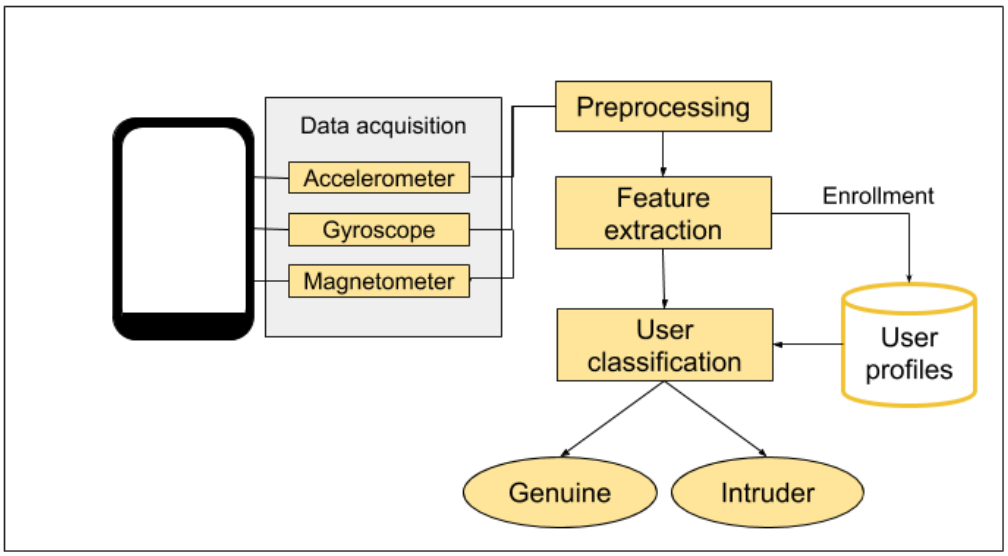

Figure 2.1: A representation of a behavioral biometric system, showing the components included and how they are connected.

2.3.2 Behavioral Biometric Systems

A biometric system is generally composed of four major components, namely, enrollment unit, feature extraction unit, template matching unit and decision-making unit. Biometric systems for behavior-based continuous authentication is similar to a general biometric recognition system. In comparison to physiological-based approaches, behavior-based approaches do in many cases not require additional hardware, such as fingerprint reader or advanced cameras. A representation of the architecture of such system is presented in Fig.2.1 The main components are:

1. *Data acquisition:* In this component, the raw data is gathered from the mobile device. The data can come from many sources such as camera, GPS, microphone and movement sensors. In this project, the three sensors considered are shown in Fig. 2.1, namely Accelerometer, Gyroscope, and Magnetometer. The data captured in this step is often noisy due to factors such as human errors and the environment in which the data is collected [25].
2. *Data preprocessing:* This component aims at improving the data quality, for example by reducing noise. The data is in this step prepared for feature extraction.

3. *Feature extraction:* In this component, a set of features used for capturing behavior are extracted from the preprocessed data.
4. *User profiles:* Is a database or local storage where the features of the behavior profiles are stored. The profile training is done during the enrollment phase, and during the recognition phase, the gathered features are compared to the saved profiles. In this project, the user profiles will be trained using machine learning algorithms.
5. *User classification:* Is also referred to as the decision-making unit [25]. It is only used during recognition phase and compares the extracted features to existing user profiles to identify whether it is from a genuine user or imposter.

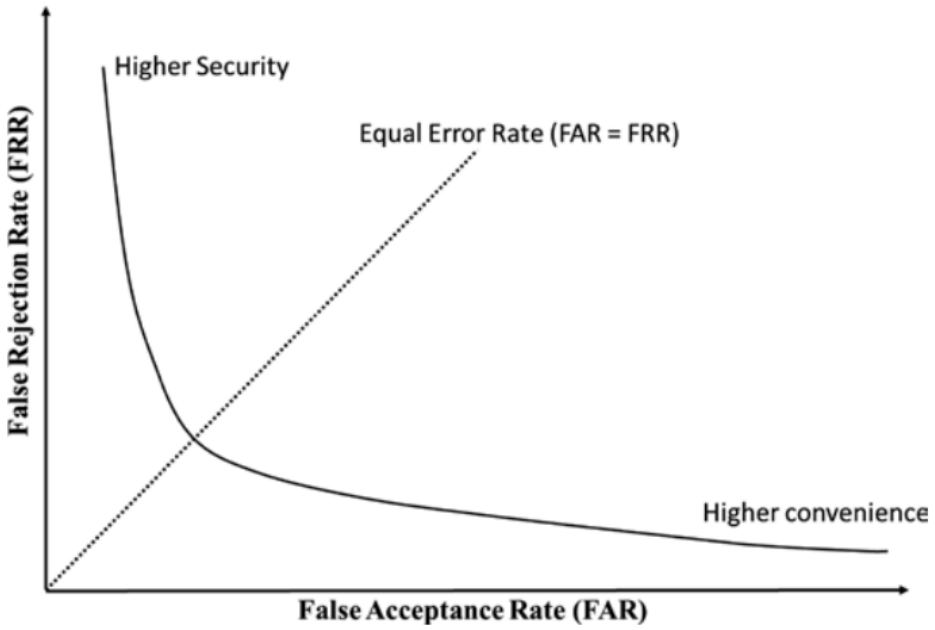

Figure 2.2: A representation of False acceptance rate (FAR), False rejection rate (FRR) and Equal Error Rate (EER), showing the tradeoff as well as where the biometric system receives the lowest EER. *Source: Dasgupta [11], page: 46.*

2.3.3 Performance of Biometric Systems

In this project, performance in biometric systems refers to measurements of the system's accuracy. For biometric systems, the primary goal is only to give authorization to genuine users and only reject imposters. Therefore, the performance measurements are closely tied to how many false acceptance and false rejections the system has. A biometric system makes use of scores, or weights, to express the similarity between a pattern and a saved biometric template. A user is only granted access if the similarity score of the provided data compared to the biometric template is higher then a given threshold. Variating the threshold alters the sensitivity of a biometric system. The sensitivity refers to the trade-off between security and convenience. Using a high threshold will eliminate the chances of having false accepted users, but instead, introduces a lot of false rejected users. On the other hand, using a low threshold will give no false rejected users, but instead, increase the number of false accepted users. Since none of the abovementioned extremes exists in real applications, the threshold introduces a trade-off where both false rejection and false acceptance occurs [39]. The trade-off of a biometric system can be seen in Fig 2.2. The measurements used to evaluate the accuracy are *False Acceptance Rate*, *False Rejection Rate*, and *Equal Error Rate*.

False Acceptance Rate (FAR)

In a biometric system, the False Acceptance Rate (FAR) is the most important measurement. FAR is the measure of the likelihood that the biometric system will give an access to an unauthorized user. It is calculated according to equation 2.1 [11].

$$FAR(\mu) = \frac{\text{Number of false accepted attempts}}{\text{Total number of attempts made to authenticate}} \tag{2.1}$$

where μ is the threshold indicating security level.

False Rejection Rate (FRR)

The False Rejection Rate is the likelihood that the biometric system will incorrectly reject access for an authorized user. The FRR is defined as follows: [11]:

$$FAR(\mu) = \frac{\text{Number of false rejected attempts}}{\text{Total number of attempts made to authenticate}} \tag{2.2}$$

where μ is the threshold indicating security level.

The lower the FAR and FRR are, the better the biometric system becomes. To compare biometric systems, it is recommended to provide both FAR and FRR scores. However, since FAR and FRR depend on the threshold given, it is hard to know whether one set of FAR and FRR scores are better compared to another set. Therefore, the Equal Error Rate can be used to give a performance measurement independent of the threshold [11].

Equal Error Rate (ERR)

Equal error rate (EER) is the value where the proportion between FAR and FRR are equal. Generally, the lower the EER, the higher the accuracy of the biometric system [39]. The graph Fig. 2.2 shows a representation of EER in relation to FAR and FRR. In this project, the threshold μ will not be fixed. Instead, the performance will be measured by using the Equal Error Rate.

2.4 Machine Learning

In this section, an introduction to machine learning is given, followed by a brief explanation of each algorithm used in this project.

2.4.1 Introduction to Machine learning

Machine learning is a field within computer science that enables computer systems to perform tasks by learning patterns in datasets. The computer system can then use the learned knowledge to perform the same task on new unseen data points [24, 9, 29]. For the majority of machine learning problems, the learning process can be roughly divided into two branches, namely *supervised learning* and *unsupervised learning* [9, 24]. In supervised learning the data samples provided for training come with the correct associated output, also denoted as labels. If the provided labels consist of discrete values, the problem is denoted as a classification problem. Similarly, if the labels instead consist of continuous values, the problem is denoted as a regression problem [9]. In contrary, the machine learning algorithms that use unlabeled data sets are called unsupervised learning methods. In unsupervised learning, the learning process relies entirely on the provided data only, with no external knowledge [9, 24]. Typical problems solved by unsupervised learning methods are clustering, outlier detection, dimensionality

reduction and association [9]. In the context of this study, we will only consider supervised learning because the dataset used in this project includes the ground truth and can be used during training.

Machine learning algorithms are generic and can be adapted to many different problem domains. Therefore, the choice of algorithm depends heavily on the dataset being used. As a consequence, there are several ways to alter an algorithm during the learning process to achieve satisfactory performance. This constitutes the challenges associated with the learning process [9].

Bias-variance tradeoff

Bias-variance tradeoff is the term used to represent a trade-off between the minimization of two prediction errors. A model with high bias and low variance tends to produce a simple model, where both training and test data result in higher prediction error. Thus the model is said to underfit. In contrary, a model with low bias and high variance tend to produce a complex model with low prediction error on the training data, but higher prediction error on the testing data. Thus the model is said to overfit [19].

Overfitting can be detected if the model tends to perform much worse on the testing subset compared to the training subset, which is an indication that the model may overfit [19, 26]. There are several methods for avoiding overfitting. During training, *K-fold cross-validation* (K-fold CV) can be applied to the training set. In K-fold CV, the training set is divided into k equally sized subsets. A subset k is then selected for testing, and the remaining $k-1$ subsets are used for training. This process is repeated for each of the k subsets. Using cross-validation makes sure that all data is taken into consideration for training and testing. Therefore, it minimizes the chances of having a model that does not generalize on new unseen data. [9]

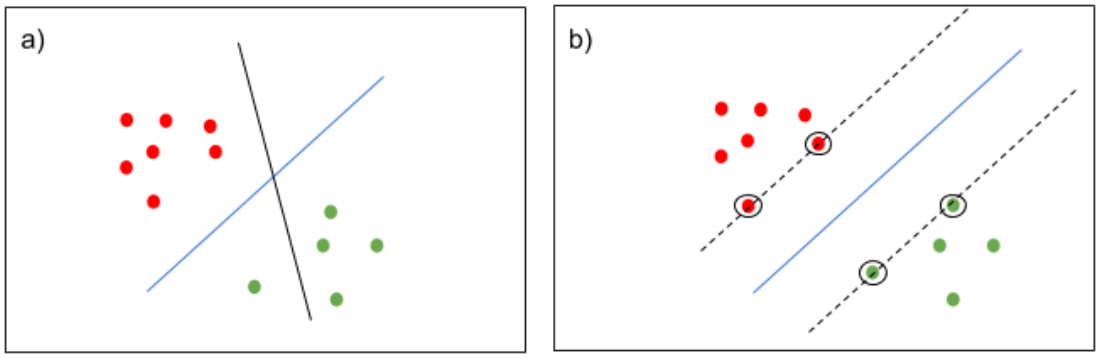

Figure 2.3: Illustration of hyperplanes that separates two classes. Figure a) shows two possible solutions. Figure b) shows the best solution, with the widest margin represented by the dotted lines.

2.4.2 Support Vector Machines

Support Vector Machines (SVM) is a supervised learning approach that is used for both regression and classification problems. For this study, SVM will be explained in the context of binary classification. The goal of an SVM is to find the optimal separating hyperplane which divides a set of data points into their corresponding classes. In the case of a two-dimensional input space, the hyperplane is the line that separates the two classes. An illustration of two possible hyperplanes in the two-dimensional space can be seen in Fig. 2.3 *a)*. However, the optimal solution is found by selecting the hyperplane which maximizes the distance between the nearest data points from either set, also referred to as the margin. Such hyperplane is shown in Fig. 2.3 *b)*, where the circled data points, denoted support vectors, defines the widest margin. The hyperplane is then used as a decision boundary for unknown data points [20].

In many cases, the data might not be linearly separable. However, by transforming the data into an adequate high-dimensional space, it is possible to find a hyperplane that separates the data. As a result, a drawback is that the computations become inefficient. SVMs solves this by applying the *kernel trick*, explained in James et al. [20].

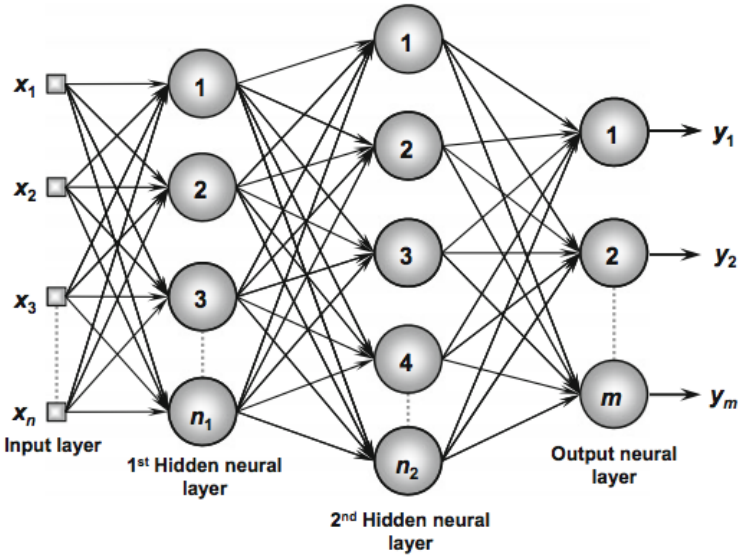

Figure 2.4: Illustration of a multi-layer feedforward network by Silva [36]

2.4.3 Artificial Neural Networks

Artificial neural networks (ANN) are computational models inspired by the nervous system of humans. The architecture of an artificial neural network defines how its *neurons* are arranged in relation to each other. Neurons are computational units in the network that have weighted input signals and produce an output signal using an activation function [36].

The main architectural features of ANN are the *input layer*, *hidden layers*, and the *output layer*. The input layer handles data input from the external environment. The hidden layers are composed of several neurons which are responsible for extracting patterns. Lastly, the output layer is responsible for providing the final output depending on the computations made before [36].

The most simple case of a feed-forward ANN is *Single-Layer Perceptron*, a feed-forward network consisting of only an input layer and an output layer, where the inputs are fed directly to the output by using the sum of the product of each

input weight and a bias. Single-layer perceptrons are linear classifiers, thus only capable of finding patterns that are linearly separable. In order to learn non-linear functions, more hidden layers must be added [36].

Multi-layer Perceptron

Multi-Layer Perceptron (MLP) is a feed-forward network with three or more layers, including the input and output layers. MLP can be used for both classification and function approximation. They are proven to be a universal approximation algorithm, capable of approximating any continuous multivariate function [36]. An illustration of the architecture can be seen in figure 2.4, consisting of $x_1 ..., x_n$ inputs, two hidden layers with n_1 and n_2 neurons respectively and finally, the output layer with m neurons responsible for the outcome $y_1, y_2, ..., y_m$.

2.4.4 Classification and Regression Trees

Classification and regression trees (CART) is a term used for referring to the two approaches for decision trees, namely classification trees and regression trees. The basis of a CART model is a binary tree. The model is trained to fit on a given training set by building a tree where each node represents a question based on the feature space. The node is split depending on the answer. When there are no more splits remaining, the leaves of the tree contains an outcome variable used for prediction [19].

Both regression trees and classification trees are similar in the sense of building a binary tree. Both approaches also use recursive binary splitting, which is a greedy algorithm for building a tree top-down by always choosing the best possible split. The algorithm is greedy in the sense that it does not take into consideration whether the selected split can lead to a better or worse future step. The main difference between the two trees lies in the criteria for deciding the splits. For regression trees, the goal is to choose the split which results in the lowest Residual Sum of Squares (RSS). However, for classification trees, it is more common to use either the Gini index or Entropy which are two methods for computing the information gained by each split [19].

When growing a tree, it is common that the CART model is too complex and overfits. This can be avoided by terminating node splits when the statistical difference is below a given threshold. The main problem with this approach is that such split may lead to a future split which is very beneficial. Therefore, a better

approach is to build a large tree and then apply a method called *tree pruning* for reducing the complexity. Tree pruning involves partitioning the initial grown tree into subtrees. The partition is decided depending on the pruning approach used. If the pruned tree performs at least as good as the initial tree, the pruned tree is chosen. Thus, reducing the complexity [19].

The main problem with a CART model is that they do not perform on the same level of accuracy as other machine learning methods. The trees also tend to overfit and not be as robust, meaning that they are sensitive to data changes [19]. However, a solution to this problem is using tree ensemble [19, 6, 15].

Tree ensemble builds on the idea of building a "strong" model by combining an ensemble of "weak" learners. Two common tree ensemble algorithms are *Random Forest* and *Boosted Trees*.

Random Forest uses *Bootstrap Aggregating* and *Random Subspace*. In Bootstrap Aggregating (bagging) the goal is to reduce the variance. For regression trees, the variance is reduced by averaging the prediction of several decision trees with high variance and low bias. This can be achieved by fitting a set of regression trees to a set of separated training datasets of the same size and then averaging the predictions. However, partitioning the training set can be hard due to data size limitations. Thus, bagging can be used by taking repeated random samples from the initial training dataset. The same principle is applied in bagging for classification trees. However, instead of averaging the prediction, a majority vote is taken for each of the predicted class. Random Subspace, also called feature bagging, aims to reduce the correlation between each tree. Reducing correlation between each tree is achieved by selecting features from a random subset of all features when building the tree instead of considering all features.

Boosted trees, or boosting, differs from Random Forest by sequentially growing each tree with respect to the error of all the previously grown trees [19, 15]. Growing the model is done by choosing the tree that best optimizes the loss function representing how the model performs. This procedure is repeated additively for each new weak learned added to the model.

2.4.5 Extreme Gradient Boosting

Extreme Gradient Boosting (XGBoost) is a variant of the gradient tree boosting proposed by Friedman [17]. Gradient tree boosting is a tree ensemble boosting method that combines a set of weak classifiers to create a strong classifier. The strong learner is trained iteratively starting with a base learner [7]. Both gradient boosting and XGBoost follows the same principal. The key differences between them lie in implementation details. XGBoost achieves better performance by controlling the complexity of the trees using different regularization techniques [7].

Let $(x_1, y_1), (x_2, y_2), ..., (x_n, y_n)$ be a set of inputs and corresponding outputs. The tree ensemble algorithm uses K addative functions, each representing a CART, to predict the output. The predicted output is given by the sum of each individual function prediction, see equation below [6, 7]:

$$\hat{y_i} = \sum_{k}^{K} f_k(x_i), f_k \in F \tag{2.3}$$

where $f \in F$ is the space of CARTs.

Thus, the objective is to approximate the functions by minizming the following *regularized* objective function [7, 6] given a set of parameters θ:

$$obj(\theta) = \sum_{i}^{n} l(\hat{y_i}, y_i) + \sum_{k}^{K} \Omega(f_k) \tag{2.4}$$

where the first term $l(\hat{y_i}, y_i)$ represents the training loss function that measures the difference between the predicted output and the actual output. The training loss function can be measured using different types of error, such as Mean Squared Error (MSE), given by [6]:

$$MSE = \sum_{i}^{n} (y_i - \hat{y_i})^2 \tag{2.5}$$

and *Logistic Loss*, given by the following equation [6]:

$$\textit{Logistic Loss} = \sum_{i}^{n} [y_i ln(1 + e^{-\hat{y_i}}) + (1 - y_i ln(1 + e^{\hat{y_i}}))] \tag{2.6}$$

The second term $\Omega(f_k)$ is the regularization term, which penalizes the complexity of the model to avoid overfitting [7]. In XGBoost the regularization term is given by [6, 7]:

$$\Omega(f) = \gamma T + \frac{1}{2}\lambda \|w\|^2 \tag{2.7}$$

where T is the number of leaves and the second term is the L2 norm of leaf scores.

During training, the model is trained additively, by optimizing for one tree at a time. Let $\hat{y_i}^t$ be the prediction value at iteration t, the additive procedure is [6]:

$$\begin{aligned}
&\hat{y_i}^0 = 0 \\
&\hat{y_i}^1 = f_1(x_i) = \hat{y_i}^0 + f_1(x_i) \\
&\hat{y_i}^1 = f_1(x_i) + f_2(x_i) = \hat{y_i}^1 + f_2(x_i) \\
&\dots \\
&\hat{y_i}^t = \sum_{k=1}^{t} f_k(x_i) = \hat{y_i}^{t-1} + f_t(x_i)
\end{aligned}$$

The tree added at each step is the tree that optimizes the objective function. The objective function can be rewritten as [7]:

$$\begin{aligned}
obj^{(t)} &= \sum_i l(\hat{y}_i, y_i) + \sum_k^K \Omega(f_k) \\
&= \sum_{i=1}^{n} l(y_i, \hat{y_i}^{t-1} + f_t(x_i)) + \Omega(f_t)
\end{aligned}$$

The objective function can be further simplified using a second-order approximation, explained in detail in Chen and Guestrin [7], into a function that can be used as a scorer. The score function is then used to determine how good the tree structure is [7, 13].

To further prevent overfitting. XGBoost implements the shrinkage introduced by Friedman [17]. The shrinkage variable scales the feature weights by a factor of η, also called learning rate. Furthermore, XGBoost also supports row subsampling and column subsampling, two techniques used to control bias and variance in Random Forest [7].

2.5 Related Work

Due to the advances in smartphones capabilities, many researchers have utilized different types of sensor data from the devices for a broad spectrum of purposes. The studies show that the sensory data can indeed be used for identifying unique traits of users. In a conference proceeding by Rybnicek, Lang-Muhr, and Haslinger [34], a roadmap to continuous biometric authentication for smartphones is presented. The authors refer to multiple papers using Accelerometer and Gyroscope individually as well as hybrid approaches. Their research concludes that combination of biometric traits such as gait, keystroke, gesture, and hand movement appears to be suitable for continuous verification. Rybnicek, Lang-Muhr, and Haslinger [34] also mention that the most challenging aspects are feature selection and the combination of such features.

In the paper by Sitova et al. [37], the authors introduced a set of behavioral features for continuous authentication. The features included how a user grasps holds and taps on the smartphone. The authors achieved EER as low as 7.16% for walking and 10.05% for sitting by combining motion sensors with tap and keystroke dynamics. The authors showed that using only Accelerometer and Gyroscope, they acquired 13.62% EER for walking and 19,68% EER for sitting, using Scaled Manhattan as the verifier. Similiar scores were received using SVM and Scaled Euclidian. The same dataset gathered by the authors is used in this project. However, in this project tap gestures are not included. Furthermore, we investigate how the different activities performed in the dataset affect the outcome of the model, as well as look at the different sensor combinations.

In the paper by Lee and Lee [23] the authors proposed a multi-sensor-based system to achieve continuous and implicit authentication for smartphone users. The sensors used in the research paper were the Accelerometer, Gyroscope, and Magnetometer. The data was re-sampled by averaging the data with sampling rates ranging from 1 second to 20 minutes. The authors used SVM to train a model on data structured as 9-dimensional vectors, three values for each sensor representing the x, y, and z-axis. For evaluation, one user's data was labeled as positive and the other users' data as negative. The experiment included testing different sampling rates as well as different combinations of the sensors. They used two data sets to evaluate their model. The combination of three sensors provided the best result of 93,9% and 97,4% accuracy. The authors also stated that the data collected from the Gyroscope sensor is not as relevant as the data from the Accelerometer and Magnetometer, which tends to measure more stable, longer-term

characteristics of the user.

In the research made by Ehatisham-Ul-Haq et al. [14], the authors proposed a framework for multi-class smart user authentication, utilizing sensors such as the Accelerometer, Gyroscope, and Magnetometer. The authors claimed that using the Magnetometer in combination with Accelerometer and Gyroscope contributed to better accuracy. The authors used data collected from 10 participants performing six different physical activities; walking, sitting, standing, running, walking upstairs and walking downstairs. An average smoothing filter was applied to handle the noise in the data. The most suited classifier was selected by comparing the machine learning classifiers decision trees, k-NN, Support Vector Machines, and Bayesian network. An evaluation was made for both user and activity classification. The authors concluded that Bayesian Network is the best use for user authentication based on physical activity recognition. The dataset used by Ehatisham-Ul-Haq et al. [14] is small and includes many activities compared to the one used in this thesis. The same sensors will be used in this thesis as in [14] but on a larger dataset with fewer activities.

Roy, Halevi, and Memon [33] proposed a Hidden Markov Model-based (HMM) multi-sensor approach for continuous mobile authentication. The authors studied continuous authentication for touch interface based smartphones. The gesture patterns of the users were modeled from the touch, Accelerometer and Gyroscope sensors using a continuous left-right HMM. In comparison, this degree project will instead only focus on sensor data from the Accelerometer, Magnetometer, and Gyroscope. We will also investigate classification rather than outlier detection.

Chapter 3

Methodology

This chapter describes the dataset we worked with, the methodology used to extract features and how the models were trained and evaluated. An overview of the outline is shown in Fig. 3.1

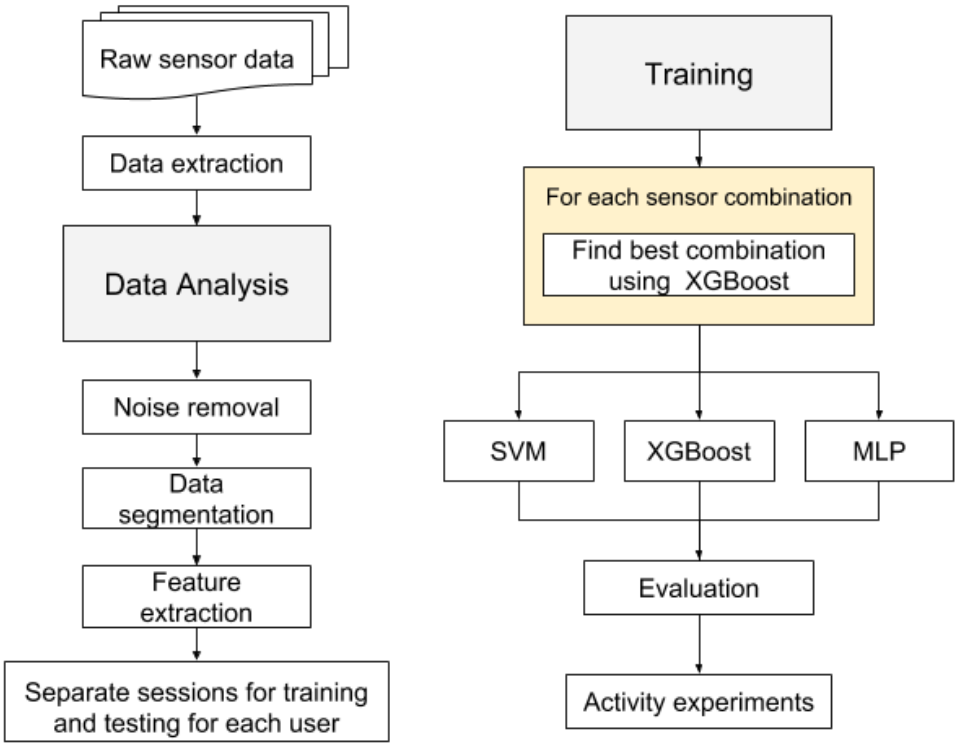

Figure 3.1: Outline of the methodology. The two major parts were data analysis and training, as represented in larger colored boxes. The intermediate steps used in each of the larger parts are represented as the small boxes.

The summary presented in Fig. 3.1 shows an overview of the methodology. It includes the steps and decisions taken to fulfill the objective and problem of this thesis. The actual implementation of the model differs slightly from the figure. A short description of the implementation is included in this chapter. The methodology used in this thesis is adapted to be suitable for biometric systems. As described in section 2.3.2 a biometric system includes components to preprocess raw sensor data, extract features from it, and send to a classifier for verification or identification. Therefore, the outline shown in Fig. 3.1 is structured to include all the steps necessary for building a model which is suitable for biometric systems.

3.1 Environment Setup

Python was used to implement and execute the methodology described. Python was used due to the availability of several libraries which simplifies data preparation, preprocessing and analysis. Furthermore, three libraries named Scikit-learn, Keras, and the DMLC XGBoost were used. Scikit-learn was used for grid-search, cross-validation, and built-in machine learning metrics [31]. Keras was used for building and training the neural network model [8]. DMLC is a community which creates open-source machine learning projects, XGBoost being the one used in this thesis [13]. By using these libraries, it is possible to focus on building and training the models instead of implementing the models.

3.2 Dataset

The data used in this project was gathered and made public in research by Yang et al. [40]. The work was supported in part by several grants such as *Defence Advanced Research Project Agency (DARPA)* and *New York Institute of Technology*. The authors gathered data from 100 volunteers using a Samsung Galaxy S4, expected to perform 24 sessions. The captured data during the sessions were: *Accelerometer, Gyroscope, Magnetometer, Raw touch events, Tap gestures, Scale gestures, Scroll gestures, Fling gestures, and Key-press on virtual keyboard*. The sensor data was captured using a sampling rate of 100 Hz and gathered while performing the tasks reading, typing, and navigating a map while walking and sitting, respectively [37].

The data is structured with 100 folders, each representing a user, containing 24 session folders. Within each session folder, the data is represented as csv files. The data samples from the Accelerometer, Gyroscope, and Magnetometer sensors includes the following information:

- Systime: Absolute time-stamp.
- EventTime: Sensor event relative time-stamp.
- ActivityID: The activity performed during the data acquisition
- X, Y, and Z: The three values captured by the sensor.
- Phone orientation: The orientation of the phone, whether it is in landscape or portrait.

Using a pre-gathered dataset enables the combinations of several sensors, analysis of activities, as well as making the proposed model comparable to future research in active authentication. Besides, gathering data of this extent while performing different activities requires a substantial amount of time. In addition, using a dataset gathered in support by well-known research agencies such as DARPA elevates the credibility of the results achieved in this project.

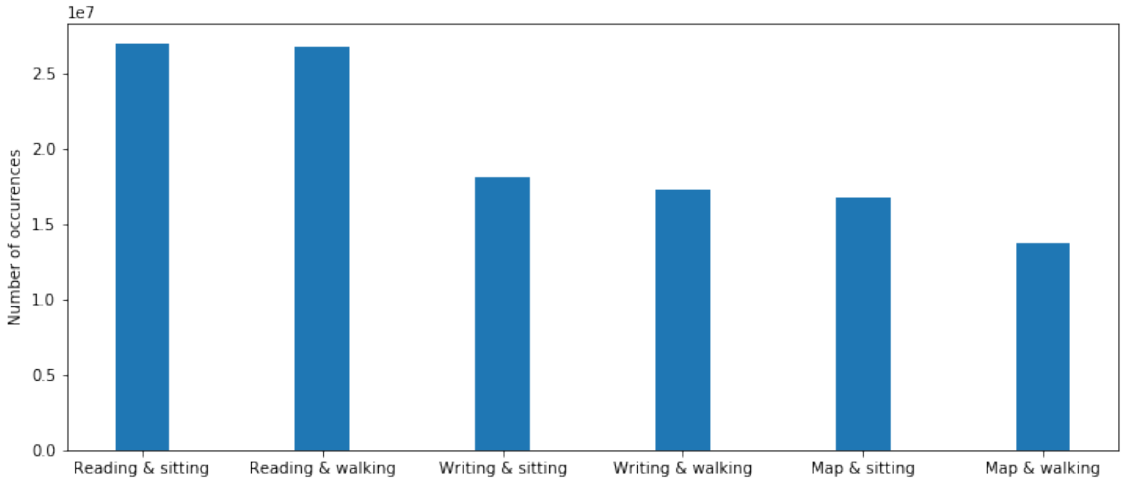

Figure 3.2: Shows the balance of the data set between each activity performed. The x-axis is each activity and the y-axis shows how many times each activity appears in the data set.

3.3 Data Analysis

The sensor data includes a significant amount of noise caused by the environment and conditions during the data gathering. Therefore, it is necessary to do data preprocessing before using it as an input to the machine learning models.

Furthermore, to not have a biased evaluation during testing it is necessary to examine the distribution of activities in the dataset. It can be seen that the distribution of the activities is not heavily imbalanced, as seen in Fig. 3.2. Therefore, there is no need for synthetic data.

In continuous authentication systems, it is beneficial to use non-intrusive sensor data that can capture user behavior. This project studies data acquired from the sensors Accelerometer, Gyroscope, and Magnetometer because they are considered non-intrusive and also included in the majority of newer smartphones. Utilizing these sensors to observe human behavior is shown to be feasible by several research papers [23, 37, 34, 14].

Generally, when building a behavior profile, it is beneficial to include many channels of data that capture different types of behavioral traits to make robust models. Using Accelerometer, Gyroscope, and Magnetometer captures hand movements only. Therefore, it is of interest to study which combination that performs the best. The combinations considered are:

- Accelerometer.
- Accelerometer and Magnetometer.
- Gyroscope.
- Gyroscope and Magnetometer.
- Accelerometer and Gyroscope.
- Accelerometer, Gyroscope, and Magnetometer.

The Magnetometer is not tested individually due to it not capturing any behavior. As described in section 2.2 the Magnetometer places the smartphone in the physical world.

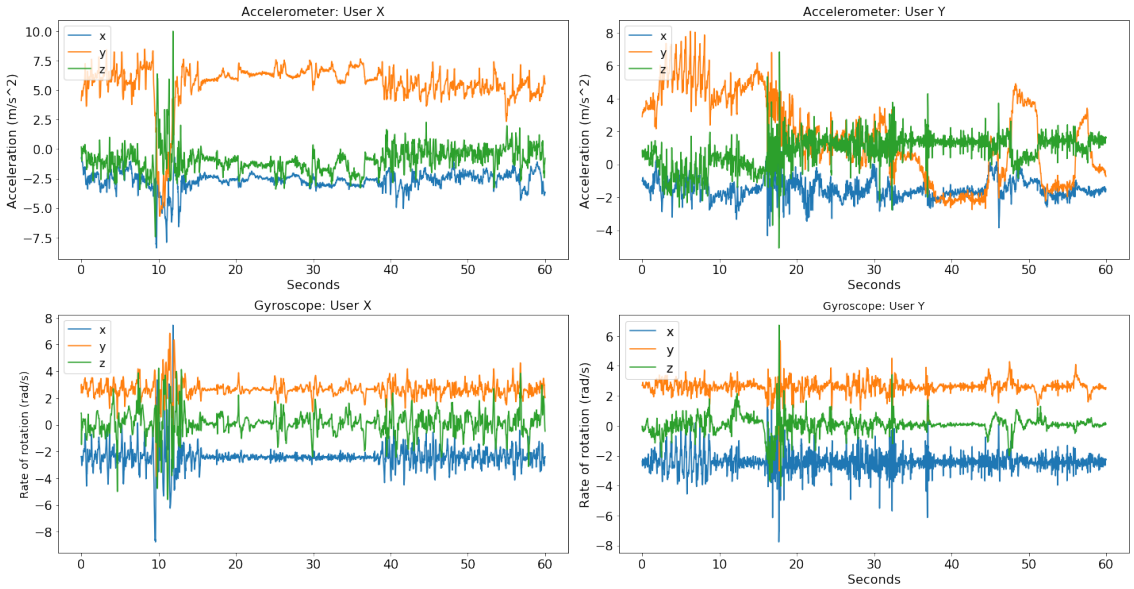

Figure 3.3: Scaled raw data captured during walking from the Accelerometer and Gyroscope. The figure shows the variation between the two users X and Y, performing the same task, typing, during one minute

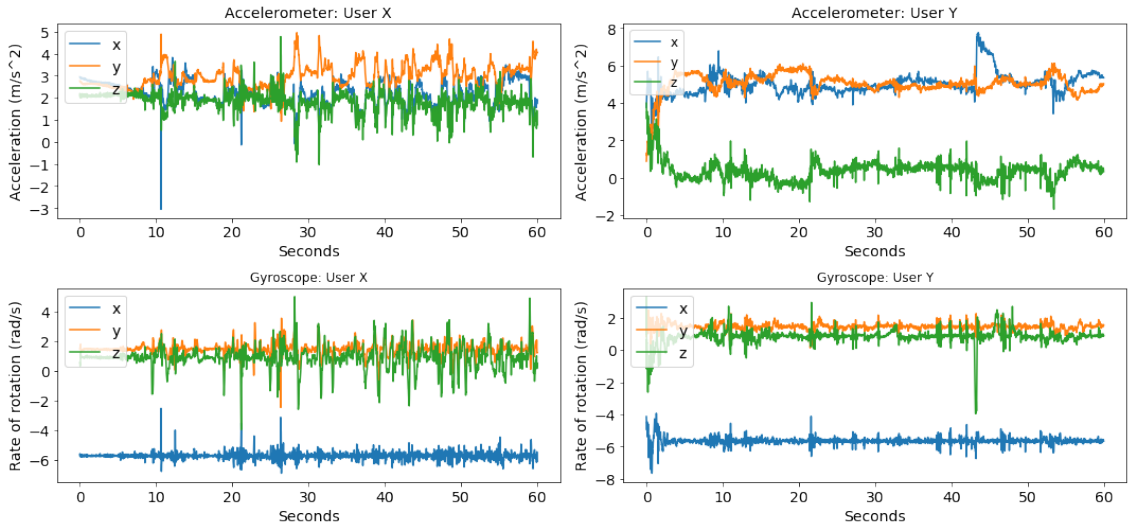

Figure 3.4: Scaled raw data captured during sitting from the Accelerometer and Gyroscope. The figure shows the variation between the two users X and Y, performing the same task, typing, during one minute

3.3.1 Sensor data

It is only necessary to use the x, y, and z-axis values to capture how the smartphone is moving. Since tap gestures on the touchscreen are not included in this project, it is not necessary to include any additional data. Each sensor includes an x, y, and z-axis value. Thus, resulting in a 9-dimensional vector for each data sample.

The sensor data studied in this project variates substantially depending on the activity performed and various external factors. The variations can come from numerous effects. Slight variations can emerge due to taps on the screen while the user is typing. Furthermore, unwanted variations can come from users being stressed, in rush or doing abrupt actions where the phone is moved in unusual manners. The data used in this study was gathered in a controlled environment performing specific activities which minimizes the chances of unwanted variations in the dataset. A representation of the variation in Accelerometer and Gyroscope data for two users during walking and sitting can be seen in Fig. 3.3 and Fig. 3.4, respectively. Both representations are from the users performing the task *typing*. Therefore, there is some slight variation due to the "taps" during typing. The representations include scaled data captured during 60 seconds. As expected it is possible to see that the sensor data from Gyroscope during sitting does not alter as much as when walking, because it is easier to hold the smartphone stable during sitting. Similarly, by examining the y-axis in Fig. 3.3 there are larger alterations in the acceleration during walking compared to sitting in Fig. 3.4.

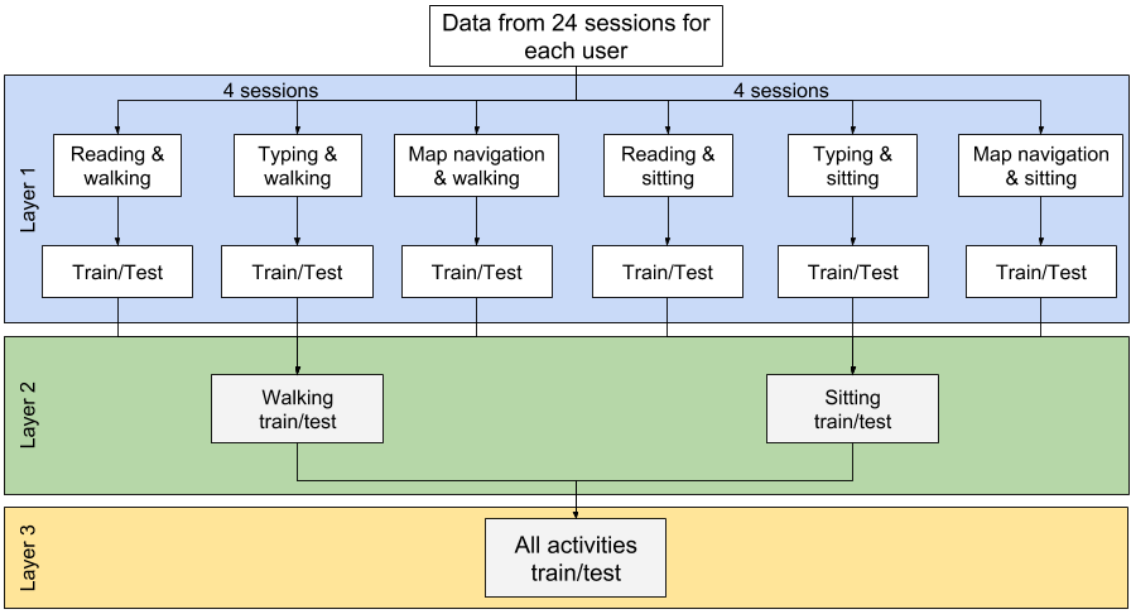

Figure 3.5: Represents how the data was prepared for each user. Layer 1 shows how the users were divided into different activities. Layer 2 shows the data used for models trained specifically on walking and sitting. Finally, layer 3 shows the data used for the general model trained on all activities.

3.3.2 Data Preparation

In this section, the steps taken during data preparation are described. First, we describe how the data was cleaned and set up for experiments. Secondly, noise reduction is described. Finally, the data segmentation and feature extraction are described.

To minimize the chances of incorrect results, all data used had to pass specific checks. First, all users that did not include raw data from Accelerometer, Gyroscope, Magnetometer, and activity in one of their sessions were removed. The data was also checked not to include any non-number values. Furthermore, each session raw data from all three sensors were examined to be gathered doing the same activity. Lastly, during feature extraction described in this section, all users with less than 80 feature vectors were removed due to few training samples.

To prepare the data for training and testing, the data from the three sensors were combined into a single file. Each data sample includes the x, y, and z-axis values captured from the Accelerometer, Gyroscope, and Magnetometer sensor, the ac-

tivity performed as well as a unique UserID for each corresponding user. Thus, making it possible to consider the problem as a classification problem.

The data was prepared in three layers represented in Fig. 3.5. In *Layer 1* the users' sessions were grouped into corresponding activity. It became apparent that each user's activity included data from four sessions. Each user's data was then divided into training and testing. To reduce the chances of having data leakage, a problem in machine learning where the model is trained on the information that it is trying to predict, we decided to not use the same session data in both training and testing since it can result in invalid results. Each session included data of different time lengths. Therefore, we decided to take the split closest to $80/20\%$ distribution with a tolerance of $\pm 0.5\%$, for training and testing. Structuring the data like this made it possible to combine the tasks to train and test on walking and sitting explicitly, as seen in *Layer 2*. Additionally, both walking and sitting were combined to prepare data for a model trained on all activities, as seen in *Layer 3*.

Noise Removal

In general, sensor data includes a considerable amount of noise. To reduce the noise, we applied a weighted moving average filter as done in previous work [23, 14]. The filter is applied to a given set of data points. Therefore, the size of the set is a variable that can affect the result of the model. By including too many data samples when computing the moving average we may decrease the noise a lot. However, it might also introduce a side effect where wanted alterations in the data are removed. If we use a pre-gathered dataset, which acquired data during controlled environment settings, it is not as necessary to apply more advanced filtering methods. In the case when data is collected in an uncontrolled setting, more filtering might be required, such as *Kalman Filtering* and *Low-pass Filter*. After examining related work, we decided to apply a weighted moving average filter to two contiguous data points, using the function below:

$$WMA_n = \sum_{i=1}^{n} W_i D_i \tag{3.1}$$

where,
n = number of points
W_i = the weight for period *i*
$\sum W_i$ = 1.0

Data Segmentation

Since the model trained is used to distinguish users, it is not sufficient to classify a user using only one data sample. For this reason, the data must be segmented into multiple windows, where the data is divided into streams with either a fixed or dynamic time window. By segmenting the raw data, we can extract features from the sequence to train the models. In this project, a fixed time window was used, similar to related work. Selecting the size of the segmentation window is a challenge in continuous authentication systems. In the paper by Ehatisham-Ul-Haq et al. [14] it was presented that different researchers have shown that simple physical activity patterns can be recognized within a five-second duration using motion sensors. Trying different window sizes is not included in the scope of this project. Therefore, a window size of five-seconds with 50% overlapping was used during training. Overlapping is used to also include the features extracted from data which connects the segments.

Feature Extraction

After the data was segmented into streams, we computed a set of features from each stream. The features derived from each raw sensor stream were decided by examining related work [23, 37, 14]. To handle orientation sensitivity of the smartphone sensors, the magnitude of each sensor was calculated [14]:

$$Mag_{sensor} = \sqrt{x^2 + y^2 + z^2} \tag{3.2}$$

From all three sensors, the following time domain features were extracted from the magnitude and x, y, and z-axis values:

- Mean: Average value of the sensor stream
- Std: Standard deviation of the sensor stream
- Max: Maximum of the sensor stream
- Min: Minimum of the sensor stream

resulting in a total of 48 features.

Besides time domain features it is possible also to include frequency domain features using *Discrete Fourier Transform*. However, due to time limitations, frequency domain features were not included in this project.

3.4 Training

As described in section 3.3.2, we prepared one training file and one testing file for each user. Training the model can be made using two different approaches, either as a one-class classifier, where only genuine data is shown during training or including imposters for binary classification. Both approaches have shown similar results in related work. In this project, imposter data where included when training. Mainly because there are a lot more machine learning algorithms that solve binary classification exceptionally well. Furthermore, using the system in a real application, imposter data can still be injected during training. The files were prepared by using the genuine user as the label 1 and including imposters data from ten random users as the label 0. For each genuine user, the imposter data included amounted to making the data balanced between the two classes. Including imposter data from several users makes sure that the model does not only learn the relation between two users. During testing, another set of ten random imposters were included to make sure that the model can generalize.

Including all users in the experiment is cumbersome. Therefore, five users were included in the experiments. For all the models 10-fold cross-validation was used. Thus, we used approximately 60% for training, 20% for validating and finally 20% for testing.

The parameter tuning was made similar for all models using a grid searching technique. A way to train and evaluate several parameters ranging from a defined array of values. The use of grid search results in more overhead and substantially increases training time depending on the number of values tested. Therefore, we decided to tune the parameters with the most significant impact on the model outcome first, to gradually make the model outcome better. Grid search was first used to test over a more comprehensive set of ranges, and then we performed another iteration for smaller ranges.

3.4.1 XGBoost

An essential factor for obtaining good results when training an XGBoost model is parameter tuning. The model was trained in two steps. First, we trained a baseline model to observe how well XGBoost performed in general. A second model was trained using parameter tuning by comparing the results to the baseline model. The approach taken during parameter tuning is similar to that of a general gradient boosting machine. First, a higher learning rate is used to make training faster.

The tree-specific parameters are then tuned. Finally, the learning rate is lowered and then the model is trained to get the optimal amount of estimators.

Parameter Tuning

A brief summary of the tuned parameters is given below [13]:

- **Learning_rate:** represents shrinkage at every step (round) during boosting. The shrinkage parameter controls the penalization of each newly added tree to the model. Using a low learning rate a closer optimum can be found. However, using a low learning rate requires more rounds to find the optimal model.
- **Num_rounds:** represents the number of rounds the model is boosted. In other words, how many estimators to use.
- **Max_depth:** represents the maximum depth of a tree. Increasing the value of the tree depth makes the model more complex.
- **Min_child_weight:** represents the minimum sum of instance weight in a child. If the tree partition step results in a leaf node with the sum of instance weight less than the given min_child_weight. No further partitioning is made.
- **Gamma:** controls regularization. The minimum loss reduction required to make further partition on a leaf node of the tree. Used to control pruning.
- **Subsample and Column Subsample:** controls the subsample ratios of the training instance. The ratio decides how much of the data XGboost can select randomly.

3.4.2 Multi-layer Perceptron Neural Network

The architecture of the MLP network used in this project includes one input-layer, consisting of the number of neurons corresponding to the dimension of the input vector. Two hidden-layers each containing five neurons with a simple non-linear activation function called *Relu*. Finally, the output-layer consists of one neuron with sigmoid activation function, which outputs a probability between 0 and 1.

The number of hidden layers and neurons used were decided by experimenting with different sizes. Dropout was used to the input of each hidden layer to generalize better and prevent overfitting.

During training of MLP, the goal is to find the optimal weights for each of the neurons. Learning a neural network is complex and requires a substantial amount of time. Therefore, a simple model was used. The MLP network was trained using the loss function *Binary crossentropy* and the optimizer *Adam*. Furthermore, the number of epochs and the batch size were decided using grid search.

3.4.3 Evaluation

Table 3.1: Shows the relation between predicted output and actual for the denotations True positive, Fasle positive, True negative, and False negative

	Prediction	**Real output**
True positive	1	1
False positive	1	0
True negative	0	0
False negative	0	1

Evaluating performance of a machine learning algorithm differs depending on the predicted outcome domain. In the context of this thesis, the objective is to evaluate the machine learning models for biometric systems. Therefore, it is essential to consider the correct metrics. As presented in section 2.3.3, the performance of a biometric system is dependant on FAR, FRR, and a given threshold. To calculate these metrics, it is necessary to study the relation between the predicted outcome and the actual outcome by counting the amount of true positive (TP), false positive (FP), true negative (TN), and false negative (FN). However, since the machine learning models output a probability for two classes (genuine and imposter), it is required to select a decision threshold. By applying the decision threshold, it is possible to determine to which class the predictions belongs [27]. However, selecting the correct threshold is a challenge and can result in misleading performance.

Reciever Operating Characteristics Curve, more commonly known as ROC curve, enables evaluation of the model without choosing a specific threshold. The ROC curve is a plot of the true positive rate (TPR) in relation to false positive rate (FPR).

Instead of plotting TPR and FPR for a fixed threshold, the ROC curve plots the relation as a function of the threshold. Therefore, it is possible to show how the model behaves using different thresholds [27].

The TPR and FPR from the ROC curve can be used to plot a trade-off between FRR and FAR shown in Fig. 2.2. The FAR was calculated using the equation 3.3.

$$FAR = FPR = FP/(FN + TN) \tag{3.3}$$

and the FRR using the equation 3.4.

$$FRR = FNR = 1 - TPR \tag{3.4}$$

where $TPR = TP/(TP + TN)$.

Furthermore, the EER is when $FRR = FAR$.

The overall performance of the model is represented with EER by taking the average of the EER for five users. The goal is to have a low EER as possible, which is the case where no false acceptance and false rejections occur.

3.5 Implementation

The implementation of the model was made to handle raw input data. The model takes raw data, pre-process it, extract features and trains a model. The model is then saved. Similarly, during testing, the models take raw data as input and performs the same procedures. Structuring the implementation to handle raw input data makes it easier to test how the model performs for different activities and tasks represented in the dataset. Also, the model could be evaluated using other datasets or own collected data with small modifications.

Chapter 4

Results

In this chapter, the results of the performed experiments are presented. The results are based on EER calculated by evaluating the model for five users as described in section 3.4.3. The results from sensor combinations are presented first. The sensor combination of the highest yielding EER is then used to present the performance of the machine learning models XGBoost, SVM and MLP. Then, we compute EER to evaluate how the model performs on the tasks: reading, typing, and map navigation for walking and sitting respectively. Finally, the overall performance of running XGBoost on 90 users is presented.

4.1 Biometric Performance

In this section, we first present the results from running experiments with different sensor combination and then the results showing how well XGBoost performed before and after tuning using the sensor combination yielding the lowest EER.

Table 4.1: The EER given for each user and sensor combination.

Sensor combination	User 1	User 2	User 3	User 4	User 5
Accelerometer	22,73%	8,34%	5,71%	8,45%	8,79%
Accelerometer & Magnetometer	29,41%	10,63%	4,45%	7,83%	16,46%
Gyroscope	29,41%	18,58%	16,46%	29,11%	22,47%
Gyroscope & Magnetometer	34,32%	20,93%	10,70%	14,36%	15,46%
Accelerometer & Gyroscope	22,04%	7,44%	5,76%	7,45%	8,34%
All	30,73%	10,07%	5,40%	8,14%	15,80%

Table 4.2: Average EER for each sensor combination over all users.

Sensor combination	Average
Accelerometer	10,80%
Accelerometer & Magnetometer	13,76%
Gyroscope	23,20%
Gyroscope & Magnetometer	19,15%
Accelerometer & Gyroscope	10,19%
All	14,03%

4.1.1 Sensor Combination

Thirty models were trained using XGBoost, one for each user and sensor combination to choose the best performing sensor combination. Table 4.1 shows the EER received by all users for each sensor combination. The results indicate that the users who resulted in higher EER also tend to receive higher EER regardless of sensor combination. The highest negative impact was given from *User 1* resulting in average EER of 28,7%, while User 3 gave highest positive impact resulting in average EER of 8,1%, it is also shown that it was the case for each sensor combination.

Table 4.2 shows the average EER for all five users. It can be seen from the table that combinations including Accelerometer tend to yield lower EER. The combination of Accelerometer and Gyroscope gives the lowest average EER of 10,2%. The highest average EER of 23.2% is given by using only Gyroscope. Including magnetometer resulted in higher EER for all cases except Gyroscope.

Table 4.3: The EER before and after tuning parameters for XGBoost using Accelerometer and Gyroscope.

User	Base model	Tuned model	Improvement
User 1	22,67%	22,04%	0,63%
User 2	11,42%	7,44%	3,97%
User 3	6,52%	5,76%	0,76%
User 4	10,91%	7,37%	3,53%
User 5	26,86%	8,34%	18,52%

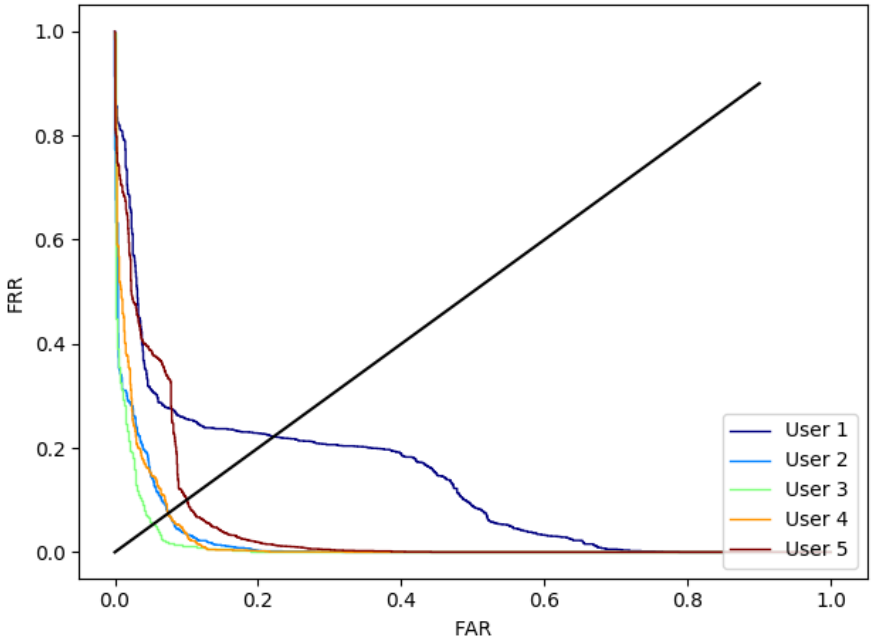

Figure 4.1: Shows the trade-off between convenience and security. Here top right means higher security and lower right higher convenience. The diagonal line represents the EER where FAR = FRR.

4.1.2 XGBoost

The results presented below are from evaluation using the sensor combination giving the lowest EER, namely Accelerometer and Gyroscope.

Hyperparameter tuning from XGBoost is presented in table 4.3. It is shown that tuning the hyperparameters for XGBoost does result in lower EER. Compared to

the base model, the tuned model received on average 5,48% lower EER for all users.

The trade-off between FAR and FRR can be seen in Fig. 4.1. The top right corner gives higher security, and the lower right corner gives higher convenience. The diagonal line represents EER, the point where the diagonal line intersects each FAR/FRR is the value that results in the best performance. From the graph, it can be seen that the model gives much higher EER for *User 1*, while the remaining users show similar behavior for each threshold.

Table 4.4: Shows the average EER of median FAR and FRR for five users for XGBoost, SVM, and MLP.

Model	FAR	FRR
XGBoost	10,30%	13,60%
SVM	12,76%	12,77%
MLP	13,09%	13,52%

Table 4.5: The EER given by each user for XGBoost, SVM, and MLP.

User	XGBoost	SVM	MLP
User 1	22,04%	19,84%	22,04%
User 2	7,44%	9,90%	10,46%
User 3	5,76%	5,08%	5,44%
User 4	7,37%	6,22%	8,68%
User 5	8,34%	8,95%	7,01%

Table 4.6: Average EER for each model when using data from all activities to train and test.

Model	Average EER
XGBoost	10,19%
SVM	10,00%
MLP	10,73%

4.1.3 Model evaluation

Table 4.4 shows the FAR and FRR for each model. Since FAR and FRR are threshold dependent, these values are average of five users median FAR and FRR. Table 4.5 shows the EER for each user. The results show that the model EER is heavily related to how well each user, given the provided data, can be distinguished from imposters. All three models tend to show similar changes in EER for the different users. Similarly, for each user, the EER does not alter considerably between the models. Table 4.6 shows the overall EER of each model, where lowest EER of 10% was received by SVM, resulting in 0,19% lower than XGBoost and 0,73% lower than MLP.

4.2 Activity

In this section, we analyze how the models classify the users for the activities walking and sitting, as well as the tasks: reading, typing, and map navigation performed during walking and sitting. First, we present the performance of the activities walking and sitting. Two experiments *A* and *B* were made. Experiment A is made training a model on data from all activities. Experiment B is made training two models, one on training data from walking and one on training data from sitting. Finally, we include results from running experiments on the specific tasks performed during data gathering.

4.2.1 Walking and Sitting

In this section, we present how well the model classifies the two activities walking and sitting. We start by presenting how well the model trained on all activities performed (experiment A), followed by the results from training a specific model for each activity (experiment B).

Table 4.7: Average EER of classifying test data from walking and sitting separately.

Model	Walking	Sitting
XGBoost	13,33%	9,00%
SVM	12,33%	10,23%
MLP	13,08%	9,75%

Experiment A: Model trained on data from all activites

The results from testing the model trained on all activities on test data from walking and sitting separately are presented in table 4.7. The table includes EER for how the model performs on the activities walking and sitting separately. In the table, it can be seen that sitting results in lower average EER of 9,7%, while walking received average EER of 12,9%.

Table 4.8: EER of models trained on walking and sitting specifically

Model	Walking	Sitting
XGBoost	6,53%	9,02%
SVM	7,71%	11,12%
MLP	6,29%	11,22%

Experiment B: Two separate models, one trained on walking and one on sitting

The results from training two models, one for walking and sitting are presented in table 4.8. The results show EER when training two specific models one for walking and one for sitting. It can be seen that the EER becomes significantly lower comparing to the results of table 4.7.

Table 4.9: Average EER of all users for tasks performed during walking and sitting for the models trained on all data

	Walking			Sitting		
Model	Reading	Typing	Map Navigation	Reading	Typing	Map Navigation
XGBoost	14,13%	18,68%	3,25%	21,67%	11,96%	9,97%
SVM	10,95%	20,11%	14,54%	13,88%	10,48%	9,24%
MLP	11,17%	18,53%	5,96%	15,36%	8,65%	8,59%

Table 4.10: Average EER of all users on tasks using models trained on walking and sitting seperatly.

	Walking			Sitting		
Model	Reading	Typing	Map navigation	Reading	Typing	Map navigation
XGBoost	6,35%	9,07%	7,75%	7,98%	3,73%	12,70%
SVM	5,12%	8,82%	3,06%	10,01%	11,95%	10,03%
MLP	5,31%	6,79%	3,00%	12,31%	8,73%	11,09%

4.2.2 Reading, Typing, and Map navigation

The results from testing the model trained in experiment A on specific tasks can be seen in table 4.9. The result shows the average EER from all five users.

The results from testing the models trained in experiment B on the tasks performed can be seen in table 4.10. The results show the EER for each model when trained on two activities walking and sitting. When training on two activities separately there is a decrease in EER. Using separate models specially trained on two activities results in lower overall EER compared to using one model for all activities.

4.3 Overall XGBoost Performance

Table 4.11: Parameters used in XGBoost for testing overall EER.

Parameter	Value
Learning_rate	0.01
Num_rounds	80
Max_depth	8
Min_child_weight	5
Gamma	0.1
Colsample_bytree	0.6
Row_subsample	1

The overall performance of XGBoost testing the model by training and testing one model for each of the 90 users resulted in an average EER of 14,653%. The models were trained using the parameters presented in table 4.11. The parameters were chosen by examining the parameters trained for the five users during previous experiments.

Chapter 5

Discussion

This chapter will discuss and summarize the most significant findings and results from the experiments. Followed by a discussion of future work.

5.1 Summary of findings

The primary objective of this thesis was to examine and evaluate the performance of using XGBoost to classify users by utilizing mobile sensors. Furthermore, additional objectives were also to investigate how different sensor combinations and activities affects the outcome of the model. Overall, the results showed that all three models performed well on distinguishing genuine users from imposters. The results show a trend in how the models perform user-wise. The model performance tends to increase and decrease similarly depending on what user data is used for training and testing. This shows that the models are very data and feature depended. Providing a different set of users would most likely result in different performance. The average EER achieved by using five users resulted in 10,2% while including ninety users resulted in an EER of 14,7%. To make overall results more significant, much more data must be used and gathered for evaluation.

5.1.1 Sensor combination

The research question connected to sensor combinations was to examine how different sensor combinations affect the outcome of the models, and if it is possible to determine the sensor combination that provides the best performance. Indeed, the results from examining different sensor combination show that using Accelerometer and Gyroscope yielded the best performance. It was not expected as

previous studies showed a better result by combining Accelerometer, Gyroscope, and Magnetometer. Combining all sensors resulted in 1,4% higher EER. The difference can come from using different datasets, data preparation as well as feature extraction. The small difference is not significant when using different datasets. In a live authentication system, utilizing all sensors would most likely result in better generalization since data captured during controlled environments differentiate significantly from data captured during a live system.

It was possible to notice by examining table 4.1 that combinations using Accelerometer performed the best and combinations including Gyroscope alone performed the worst. These findings indicate that gyroscope data include more noise compared to the accelerometer and therefore harder to model alone. A similar finding is also shown in related work [23]. To correctly examine the gyroscope sensor, a more thorough data pre-processing and analysis is necessary.

5.1.2 Model evaluation

In this thesis, we also wanted to answer whether XGBoost can be utilized to provide state-of-the-art results for user classification using sensors captured from smartphones. First, it is necessary to discuss the evaluation metrics briefly before trying to answer the question. This thesis focus on providing results that are relevant to a biometric system, as described in section 3.4.3. In any biometric system, security and convenience are two essential requirements. The choice of threshold is depended on the type of biometric system being implemented. Therefore, it is necessary to provide both FAR and FRR during evaluation. In table 4.4 it is shown that on average XGBoost provides the lowest FAR compared to SVM and MLP, but highest FRR. Furthermore, FAR and FRR are threshold depended. By also presenting the performance of the models in EER, we choose the threshold that provides the lowest EER, as seen in Fig. 4.1.

The results in table 4.6 shows that XGBoost indeed does provide comparable results compared to SVM and MLP. A challenge when using XGBoost is that the model requires substantial parameter tuning. In this project, we focused on training a specific model for each user, which made parameter tuning harder but essential for the model to provide good results. The significance of parameter tuning is visible in table 4.3, where the users combined received an average improvement of 5,5% EER after parameter tuning. These results were expected because XGBoost can be tuned in many ways to make it perform well on many problems. In cases where a significant amount of data is provided, and a single model is

needed, XGBoost will most likely provide good results as seen in many Kaggle machine learning contests [28].

In comparison to SVM, XGBoost can be used to get even better results if parameters are tweaked more, the same can be said for MLP. SVM does not include many parameters to tune and from the experiments, we can notice that further tuning of the model would most likely not improve performance significantly. Ultimately, if features used for training were perfect for distinguishing users, SVM would most likely provide comparable results. If more complex sensor data from several channels are included, using a Neural Network would most likely be more suitable compared to XGBoost and SVM due to the neural network having the ability to learn features automatically. Using few modalities such as in this case, choosing a simpler algorithm would be beneficial over a more complex algorithm.

5.1.3 Activites

The research question *How do different activities performed during device usage affect the user classification?* can be answered examining the results in section 4.2. It is clear that the models trained on walking and sitting specifically outperform the model trained on all data from examining table 4.7 and table 4.8. The difference in performance comes mainly from *User 1* which impacted the average EER negatively. The results of *User 1* is most likely due to how the data was collected. It can be seen from studying appendix A.1 that the model trained on *User 1* does exceptionally well on test data from the task *Map navigation*. Thus, the model is not generalizing well when other activities are performed.

It can also be seen in table 4.8 that the model trained on walking performs better than the model trained on sitting. The difference can come from the model also differentiating on how the users walk compared to each other, also called gait recognition.

Studying the task-specific results from table 4.9 and table 4.10 it can be seen that the test runs using two models trained on walking and sitting does outperform the model trained on all data. The model trained on all data seems to overfit on the task *Map navigation*, which performs surprisingly well compared to the other tasks. When using two models, the results generalize much better where EER for each task are closer to each other. This finding indicates that including the data explicitly gathered during the task *Map navigation* is making the model generalize much worse.

Having a model that generalizes well for many types of tasks and activities is very hard. One approach is to use a two-layer classifier, where the first layer classifies the activity, whether the user is walking or sitting and then redirecting to the models explicitly trained on walking and training. A similar approach is used in previous research by examining sensors placed in different places on the body [14].

5.2 Future work

The results of this thesis are made in offline experiments on a computer using a single dataset. Since data gathering differs a lot depending on environment settings, it would be beneficial to evaluate more datasets. However, there are not many datasets available, and the majority of the datasets are collected in controlled environments. Therefore, future research would be to examine performance on data collected in settings which are not controlled and more similar to real life scenarios.

The results showed that only using time domain features extracted from the sliding window segments managed to achieve comparable results to other research papers presented in related work, were both time and frequency domain features were used. Frequency domain features were excluded from this project due to time limitations. The literature study and related work, both point towards the inclusion of both time and frequency domain feature for improving the performance. Including the frequency domain features in this thesis would most likely result in improvement of the overall performance.

Using a fixed window size is a limitation in the model's performance. The main reason is due to the selection of the optimal window size. Optimally, the window size should not include activity transitions which might result in faulty features. Therefore, deciding upon the fixed window size is a challenge. Instead, an improvement would be to adopt a dynamic window segmentation technique.

Feature extraction requires substantial domain expertise. Furthermore, the behavior of each individual is different. Therefore, using pre-determined features to find the behavior of every single human might not result in good models in all cases. Future research can be done in examining feature extraction using neural networks, more specifically Convolutional Neural Network, where "raw" data can be used as input. Also, studying how we can build a general neural network that can learn the behavior of any user would eliminate the hassle of tuning pa-

rameters.

Future research can also be done to study if a model can be improved by applying continuous learning, where data extracted from subjects over a more extended period can be used to make the model more robust over time.

The proposed model can also be applied to other problems. Future research can be made on how the technique can be applied for applications in smartphones, for example during authentication to know whether it is a bot or a real user who is trying to authenticate without interrupting the user with mechanisms such as Captchas.

Chapter 6

Conclusion

In this thesis, we implemented and evaluated XGBoost for user classification utilizing accelerometer, gyroscope, and magnetometer data collected from smartphones. The used approach included training a model for each user by providing data samples from a genuine user and a set of imposters. The models were trained using time domain features extracted from the sensor data with a fixed size sliding window and 50% overlapping. The results of XGBoost were compared to SVM and a simple MLP. The work in this thesis demonstrated that it is possible to use XGBoost to achieve results comparable to state-of-the-art, where XGBoost received an average EER of 10,19% during evaluation using five users. A more extensive test using fixed parameters on ninety users resulted in an average EER of 14,7%.

Experiments were also done to study how different sensor combination and activities affected the models. Two experiments were done, one where the model was trained on data from all activities, and one where two models were trained explicitly for walking and sitting. It was shown that it is possible to use a general model, but better results were achieved by training on the specific activities.

The primary limitation of this thesis is that all assumptions and results are deduced using a single dataset and features extracted from the time domain. Therefore, using another dataset and including frequency domain features can change the results significantly. However, the results of this thesis and related work show that machine learning models are suitable for learning behavior profiles for biometric systems.

Bibliography

[1] *ABI Research forecasts 95% of smartphones to feature fingerprint sensors by 2022 | BiometricUpdate*. Accessed on 04/23/2018. May 2017. URL: https://www.biometricupdate.com/201705/abi-research-forecasts-95-of-smartphones-to-feature-fingerprint-sensors-by-2022.

[2] Abdulaziz Alzubaidi and Jugal Kalita. "Authentication of Smartphone Users Using Behavioral Biometrics". In: *Communications Surveys & Tutorials, IEEE* 18.3 (2016), pp. 1998–2026. ISSN: 1553-877X.

[3] *Authentication Definition by Merriam-Webster*. June 2018. URL: https://www.merriam-webster.com/dictionary/authentication.

[4] Adam J. Aviv et al. "Smudge Attacks on Smartphone Touch Screens". In: *Proceedings of the 4th USENIX Conference on Offensive Technologies*. WOOT'10. Washington, DC: USENIX Association, 2010, pp. 1–7. URL: http://dl.acm.org/citation.cfm?id=1925004.1925009.

[5] *Biometrics Definition by Merriam-Webster*. Accessed on 04/23/2018. URL: https://www.merriam-webster.com/dictionary/biometrics.

[6] Tianqi Chen. *Introduction to Boosted Trees*. Oct. 2014. URL: https://homes.cs.washington.edu/~tqchen/pdf/BoostedTree.pdf.

[7] Tianqi Chen and Carlos Guestrin. "XGBoost: A Scalable Tree Boosting System". In: *Proceedings of the 22nd ACM SIGKDD International Conference on knowledge discovery and data mining*. KDD '16. ACM, Aug. 2016, pp. 785–794. ISBN: 9781450342322.

[8] François Chollet et al. *Keras*. https://keras.io. 2015.

[9] Thiago Christiano Silva. *Machine Learning in Complex Networks*. 1st edition. 2016. ISBN: 3-319-17290-5.

[10] Thakkar Danny. *Identification vs. Verification vs. Segmented Identification*. (Accessed on 04/26/2018). URL: https://www.bayometric.com/identification-verification-segmented-identification/.

[11] Dipankar Dasgupta. *Advances in User Authentication*. Infosys Science Foundation Series. 2017. ISBN: 3-319-58808-7.

[12] Alexander De Luca and Janne Lindqvist. "Is Secure and Usable Smartphone Authentication Asking Too Much?" eng. In: *Computer* 48.5 (May 2015), pp. 64–68. ISSN: 0018-9162.

[13] *dmlc: XGBoost eXtreme Gradient Boosting*. URL: https://github.com/dmlc/xgboost.

[14] Muhammad Ehatisham-Ul-Haq et al. "Authentication of Smartphone Users Based on Activity Recognition and Mobile Sensing". In: *Sensors* 17.9 (Jan. 2017). ISSN: 14248220. URL: http://search.proquest.com/docview/1952107742/?pq-origsite=primo.

[15] Alexey Enatekin and Alois Eknoll. "Gradient Boosting Machines, A Tutorial". In: *Frontiers in Neurorobotics* 7 (Dec. 2013). ISSN: 1662-5218. URL: https://doaj.org/article/1bd6c97fe50b4d1c8e0b3d41f9424222.

[16] Ron Fedwik. *Stanford: Sensors and Cellphones*. URL: https://web.stanford.edu/class/cs75n/Sensors.pdf.

[17] Jerome H. Friedman. "Greedy Function Approximation: A Gradient Boosting Machine". In: *The Annals of Statistics* 29.5 (Oct. 2001), pp. 1189–1232. ISSN: 00905364.

[18] Google. *Sensor overview*. -. URL: https://developer.android.com/guide/topics/sensors/sensors_overview.html.

[19] Gareth James. *An Introduction to Statistical Learning with Applications in R*. 2013. ISBN: 1-4614-7138-9.

[20] Gareth James et al. "Support Vector Machines". In: *An Introduction to Statistical Learning: with Applications in R*. New York, NY: Springer New York, 2013, pp. 337–372. ISBN: 978-1-4614-7138-7. DOI: 10.1007/978-1-4614-7138-7_9. URL: https://doi.org/10.1007/978-1-4614-7138-7_9.

[21] Nattapong Jeanjaitrong and Pattarasinee Bhattarakosol. "Feasibility study on authentication based keystroke dynamic over touch-screen devices". In: IEEE, Sept. 2013, pp. 238–242. ISBN: 978-1-4673-5578-0.

[22] Kaggle. *Kaggle*. -. URL: https://www.kaggle.com/competitions.

[23] Wei-Han Lee and Ruby B. Lee. "Sensor-Based Implicit Authentication of Smartphone Users". In: IEEE, June 2017, pp. 309–320. ISBN: 978-1-5386-0542-4.

[24] Panos Louridas and Christof Ebert. "Machine Learning". In: *Software, IEEE* 33.5 (Sept. 2016), pp. 110–115. ISSN: 0740-7459.

[25] Ahmed Mahfouz, Tarek M. Mahmoud, and Ahmed Sharaf Eldin. "A survey on behavioral biometric authentication on smartphones". In: *Journal of Information Security and Applications* 37 (Dec. 2017), pp. 28–37. ISSN: 2214-2126.

[26] Kevin P. Murphy. *Machine Learning: a probabilistic perspective*. 2012. ISBN: 978-0-262-01802-9.

[27] Kevin P. Murphy. "Machine Learning: a probabilistic perspective". In: 2012, pp. 182–191. ISBN: 978-0-262-01802-9.

[28] Didrik Nielsen. *Tree Boosting With XGBoost - Why Does XGBoost Win "Every" Machine Learning Competition?* Nielsen, Didrik, 2016.

[29] J. Palmer and A. Chakravarty. "Supervised machine learning". In: *An Introduction To High Content Screening: Imaging Technology, Assay Development, and Data Analysis in Biology and Drug Discovery*. wiley, Jan. 2015, pp. 231–245. ISBN: 9780470624562.

[30] Vishal M. Patel et al. "Continuous User Authentication on Mobile Devices: Recent progress and remaining challenges". In: *Signal Processing Magazine, IEEE* 33.4 (July 2016), pp. 49–61. ISSN: 1053-5888.

[31] F. Pedregosa et al. "Scikit-learn: Machine Learning in Python". In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.

[32] "Performance Evaluation of Biometric Systems". In: *Signal and Image Processing for Biometrics*. Hoboken, NJ, USA: John Wiley and Sons, Inc., Jan. 2013, pp. 207–230. ISBN: 9781848213852.

[33] A. Roy, T. Halevi, and N. Memon. "An HMM-based multi-sensor approach for continuous mobile authentication". In: *Proceedings - IEEE Military Communications Conference MILCOM*. Vol. 2015-. Institute of Electrical and Electronics Engineers Inc., Dec. 2015, pp. 1311–1316. ISBN: 9781509000739.

[34] Marlies Rybnicek, Christoph Lang-Muhr, and Daniel Haslinger. "A roadmap to continuous biometric authentication on mobile devices". In: IEEE, Aug. 2014, pp. 122–127. ISBN: 978-1-4799-7324-8.

[35] Khalid Saeed et al. "Introduction to Behavioral Biometrics". In: *New Directions in Behavioral Biometrics*. 2017. ISBN: 978-1-4987-8462-7.

[36] Ivan Nunes da Silva. *Artificial Neural Networks A Practical Course*. 2017. ISBN: 3-319-43162-5.

[37] Zdenka Sitova et al. “HMOG: New Behavioral Biometric Features for Continuous Authentication of Smartphone Users”. In: *Information Forensics and Security, IEEE Transactions on* 11.5 (May 2016), pp. 877–892. ISSN: 1556-6013.

[38] Y. Sui et al. “Active user authentication for mobile devices”. In: vol. 7405. 2012, pp. 540–548. ISBN: 9783642318689.

[39] *Technical Document About FAR FRR and EER*. Accessed on 04/23/2018. URL: syris.com/SYRIS_ACS_DVD-ROM/UserGuideManual/Reader/SYRDF5/About%20FAR_FRR_EER.pdf.

[40] Qing Yang et al. “A multimodal data set for evaluating continuous authentication performance in smartphones”. In: *Proceedings of the 12th ACM Conference on embedded network sensor systems*. SenSys ’14. ACM, Nov. 2014, pp. 358–359. ISBN: 9781450331432.

[41] Y. Yu Zhong, A. K. Deng, and A. K. Jain. “Keystroke dynamics for user authentication”. In: IEEE, June 2012, pp. 117–123. ISBN: 978-1-4673-1611-8.

Appendix A

Appendix

A.1 EER from classification using data from activities separately

User 1			Walking			Sitting		
Model	Walking	Sitting	Reading	Typing	Map navigation	Reading	Typing	Map navigation
XGBoost	40,03%	11,51%	43,79%	62,33%	2,00%	45,68%	22,65%	10,00%
SVM	37,06%	14,12%	23,10%	67,33%	58,67%	18,11%	25,29%	14,62%
MLP	39,22%	13,17%	25,52%	62,00%	14,00%	26,49%	14,41%	10,77%

User 2			Walking			Sitting		
Model	Walking	Sitting	Reading	Typing	Map navigation	Reading	Typing	Map navigation
XGBoost	5,30%	12,50%	2,78%	9,76%	2,08%	33,53%	9,64%	17,27%
SVM	8,07%	12,92%	7,22%	12,93%	4,17%	15,29%	12,50%	8,18%
MLP	6,99%	11,97%	4,44%	8,78%	5,83%	15,29%	12,68%	7,27%

User 3			Walking			Sitting		
Model	Walking	Sitting	Reading	Typing	Map navigation	Reading	Typing	Map navigation
XGBoost	3,83%	7,25%	1,43%	1,85%	8,39%	12,90%	2,08%	9,57%
SVM	4,83%	7,75%	2,00%	2,22%	7,74%	11,94%	1,46%	9,57%
MLP	4,75%	8,04%	1,43%	2,22%	7,42%	11,61%	2,08%	10,00%

User 4			Walking			Sitting		
Model	Walking	Sitting	Reading	Typing	Map navigation	Reading	Typing	Map navigation
XGBoost	8,81%	7,42%	17,27%	7,31%	2,30%	7,14%	18,75%	7,50%
SVM	5,03%	9,39%	16,36%	8,85%	1,15%	13,57%	5,00%	5,36%
MLP	7,70%	7,73%	19,09%	8,08%	1,53%	11,43%	5,42%	3,93%

User 5			Walking			Sitting		
Model	Walking	Sitting	Reading	Typing	Map navigation	Reading	Typing	Map navigation
XGBoost	8,69%	6,34%	5,36%	12,16%	1,50%	9,10%	6,67%	5,50%
SVM	6,67%	6,96%	6,07%	9,22%	1,00%	10,50%	8,15%	8,50%
MLP	6,77%	7,83%	5,36%	11,57%	1,00%	12,00%	8,64%	11,00%

Table A.1: Shows EER for each activity on model trained on all activites.