CSC8499 Individual Project: Continuous Authentication Using Mobile Technologies

Stella Englezou

Newcastle University,
Department of Computing Science
s.englezou@newcastle.ac.uk

Abstract. Continuous authentication is the process of continuously and implicitly verifying the user's identity the entire time he interacts with a system. In this paper, a continuous authentication system on smartphone devices is proposed and developed based on prior research in existing continuous authentication systems. The system was developed for Android OS using Java and utilizes the accelerometer, magnetometer and Wi-Fi sensor to collect a user's behavioural characteristics and information about his environment, while he performs a set of instructions given to him for three scenarios. The system is evaluated for each scenario using two machine learning algorithms; the Naïve Bayes and Multilayer Perceptron on datasets of six users.

Declaration: I declare that this dissertation represents my own work except where otherwise explicitly stated.

1 Introduction

Authentication is the process of verifying the user's claimed identity before granting him access to sensitive and private information. Currently, most systems verify the user using static methods of authentication such as Knowledge-based authentication, Possession-based authentication and Biometric-based authentication. These methods authenticate the user only at the login stage [1]. In some systems where the cost of security is high these methods of authentication are insufficient.

Passwords, PINs and swipe patterns can be easily guessed or spoofed if they are not strong enough or sometimes the user might deliberately give his credentials to someone else for personal gain. For example, a student could give his credentials to someone else in order for the latter to take an online examination in the student's place [2].

Possession-based authentication sometimes offers greater protection than Knowledge-based, as a token or a smartcard can hold information that is more complex than a password. As a result, it is harder for attackers to successfully attack such a system. However, a token or a smartcard can be easily stolen, lost or skimmed [3]. Moreover, some RFID authentication protocols are vulnerable to disclosure and desynchronization attacks [4].

Nowadays, using Biometric-based authentication on smartphone devices is one of the leading methods of verifying the user, as an increasing number of devices now come with a fingerprint sensor. However, fingerprints [5] and other biometric traits can be easily forged. Other times, a user might not lock his device at all before leaving it unattended, thus giving the opportunity to attackers to access a system and steal or delete sensitive data.

Thus, researchers in security are exploring other methods to verify the user's claimed identity. Such a method is Continuous authentication. It allows the user's identity to be verified continuously and implicitly [6]. Additionally, a system that uses Continuous authentication can react in certain ways when it detects someone else other than the legitimate user using the system. It can either lock the entire system, thus making it inaccessible and unusable by the other people, or provide basic functionality that does not compromise the system's security and integrity. This is essential in situations when a different user, other than the legitimate user, is required to temporarily use the system to perform basic tasks[1].

The main aim of this project is to develop and evaluate a Continuous authentication system for smartphone devices. This dissertation is structured as followed:

- Section 2 introduces the aim of this project and the objectives that must be achieved.
- Section 3 introduces existing methods of authentication; what these are and how they work. Moreover, it introduces the notion of machine learning and how it is usually used in everyday life to improve systems. Additionally, it defines several machine learning algorithms, it states on what their predictions are based on and lists the advantages and disadvantages of these algorithms. There on follows a list of sensors found on Android devices, their functionality, a description of what they measure and their possible use in a continuous authentication system. Lastly, existing Continuous authentication systems are assessed to learn what has been done previously by researchers in this field.
- Section 4 discusses the overall design of the proposed system. It describes what information about the user the proposed system is going to collect in order to verify his identity and briefly describes the tools and language used to develop this system.
- Section 5 presents the overall work carried out to design, implement and evaluate the performance of the system. It describes the experiments performed using the device's accelerometer and the experiments performs with various Weka and Machine Learning, to evaluate the performance of the system.
- Section 6 discusses the implementation of the continuous authentication system.
- Section 7 introduces the testing strategy for collecting sufficient information to evaluate the system and presents the results from the tests carried out.
- Section 8 presents an evaluation of the project, whether the aim and objectives were achieved.
- Section 9 presents possible future work to further improve and evaluate the system's performance.

[1] Introduction based on the Report for CSC8205 - Research Skills.

- Section 10 presents the conclusion of this dissertation.

2 Aim and Objectives

2.1 Aim

The aim of this project is to develop, and evaluate the performance of a continuous authentication system on a smartphone device. The newly created system should use the chosen sensors to gather behavioural characteristics of the user in different modes to verify the user's claimed identity.

2.2 Objectives

1. Research and assess existing continuous authentication systems to gain an insight into the challenges and principles of developing such systems.
2. Research and identify which behavioural characteristics and information about the environment of the users can be used by the authentication system to accurately identify them.
3. Based on the user's behavioural characteristics and information about his environment determined by the previous objective, research and identify which sensors on the smartphone device can be used to authenticate him.
4. Identify the various modes which the smartphone will be used under and define what information about the user and his environment can be used to accurately identify him in each mode.
5. Design and create a continuous authentication system on a smartphone device that uses the chosen sensors, in the defined modes, to gather the behavioural characteristics and environment information of the user and verify the user's identity.
6. Evaluate the performance of the continuous authentication system while being used in the defined modes with the chosen sensors. This will be achieved by training and testing the algorithm with several people.

3 Background

3.1 Methods of Authentication

Knowledge-based Authentication. It is a static method of authentication, which means that it authenticates the user only at the login stage. The user is authenticated based on something he knows. For example, the user might be asked to input his username with a password or some other memorable information such as mother's maiden name, favourite colour or name of first pet [1]. This method of authentication has some advantages and disadvantages. The advantages of a password are that it can be easily remembered and changed. Moreover, knowledge-based authentication can act as a fallback method to other security authentication mechanisms. However, some disadvantages of knowledge-based authentication are that passwords can be easily guessed if they are not strong, and they can be spoofed as an attacker could set up a fake login screen to trick users and get their password. Additionally, some users do not change their passwords often or use the same password for multiple systems, which means that if the attacker acquires the password he can access the user's private data stored in multiple systems [7,8] that use the same password. Another problem with knowledge-based authentication method, as described by Flior et al. in [2], is that a student might deliberately give his password away to someone with the purpose of having that person take an exam in his place. Furthermore, in some systems when the authentication stage fails for a specific number of attempts the system denies access to the legitimate user for a defined amount of time. When an attacker cannot acquire the user's password in any other way, he can exploit this mechanism and deliberately fail the authentication stage in a system multiple times in order to get the legitimate user locked out (denial of service attack). Lastly, swiping patterns are vulnerable to smudge attacks.

Additionally, knowledge-based authentication gives rise to usability issues as many people do not like to enter and re-enter their passwords or PINs multiple times [5].

Possession-based Authentication. It is another static authentication method. The user is authenticated at the login stage (same as Knowledge-based authentication) with something he has, such as a token, a physical key or an ID card [1]. An advantage of using a token to authenticate a user is that it offers protection against guessing attacks (e.g. brute force, dictionary attacks) as the stored passcode can be more complex than a password used in Knowledge-based authentication systems. Moreover, tokens can have tamper-resistant packages and hardware which disables the token if it is been tampered with [1]. However, a token can be stolen or can be used by anyone even if they are not the owner of the token. Additionally, the owner of the token can deliberately give it to someone else to use it in their place. Moreover, compared to knowledge-based method, possession-based systems usually require special hardware to read the information stored on the tokens, which can make them an expensive method of authentication. Lastly, it is recommended that this authentication method is used in combination with either knowledge-based or biometric-based methods to improve security and prevent an attacker from accessing a system if the token is stolen.

Biometric-based Authentication. The user is authenticated based on his physiological and/or behavioural characteristics. Physiological biometrics are the biometrics which provide unique information about someone such as; face, fingerprints, iris and DNA [9]. Moreover, behavioural characteristics are patterns of behaviour that uniquely identify a person such as; keystroke dynamics, gait, speech, blinking pattern [10]. Some advantages of biometric-based authentication are that it is difficult for an attacker to forge someone's biometrics and is impossible to share them compared to knowledge-based method where the user deliberately gives his password to someone else. However, Biometric-based authentication has some important disadvantages. For example, if a biometric is compromised it cannot be replaced, [11] contrary to a password or a token from the previous two methods that can be easily changed. In this case the user will have to use a different biometric, which in time can become problematic because a user has a limited number of biometric traits that he can use. Moreover, biometrics need to be pre-collected, pre-measured and stored in a secure database, which is time consuming and harder to achieve compared to storing passwords, pins and swipe patterns. An attacker could gain access to the database where the biometric traits are stored and replicate them or tamper with them, resulting in getting access to a user's private information. Additionally, an attacker could tamper with the feature extraction method, which is used to extract data from the biometrics, and thus be able to produce pre-defined feature sets, which will allow him to access a critical secure system. Lastly, an attacker could use a previously recorded version of a biometric trait (e.g. audio) to gain access in a system [10].

Biometric systems can be divided in 2 stages: the enrollment module and the identification module. [12]

- The enrollment module is responsible for training the model that identifies a given user. During this stage, biometric data is collected from the user through sensors to create an initial digital profile of the person. Then a feature extractor is applied to this profile, to generate a more concrete and expressive representation of the user's profile, known as the template. This newly created template is then stored in a database.
- The identification module is responsible for verifying the user's claimed identity. During this stage, the system using the sensors collects the biometric characteristics of the person who is being verified, and converts them into a template. The resulting template is then used by the feature matcher, whose job is to compare this newly created template against the stored template to determine whether the user is who he claims to be.

Table 1. Methods of Authentication [11].

Method	Examples	Properties
What you know	User ID Password PIN	Shared Many passwords easy to guess Forgotten
What you have	Cards Badges Keys	Shared Can be duplicated Lost or stolen
What you know and what you have	ATM card + PIN	Shared PIN a weak link
Something unique about the user	Fingerprint Face Iris Voice print	Not possible to share Repudiation unlikely Forging difficult Cannot be lost or stolen

Multi-factor Authentication. It is another static method of authentication. However, this uses multiple factors to verify the user's claimed identity. It can use a combination of knowledge-based authentication (something you know), possession-based authentication (something you have) and biometric-based authentication (something you are). For example, a password and a token can be used together to verify the identity of users, thus making it a two-factor authentication as two factors are being used. Another example is using fingerprint, token and location to authenticate users, thus making it a three-factor authentication. Multi-factor authentication adds a layer of security to systems, as it harder for an adversary to forge all authentication factors to gain access to the system. One can argue that multi-factor authentication decreases the usability of the system because it requires users to carry out additional actions during the authentication process [1].

Continuous Authentication. Continuous authentication is the process of continuously verifying and authenticating the user the whole time he interacts [13] with a system, unlike systems that employ passwords or security tokens and only verify the user's identity at the beginning of the session [13]. Continuous authentication systems usually use the physiological and behavioural characteristics of a user to continuously verify whether the user is who he claims to be. Moreover, continuous authentication not only improves the security of systems but also improves their usability as well, as they are able of implicitly verify the user's identity without interrupting him [9]. Additionally, continuous authentication systems can use other authentication mechanisms as a fall-back. For example, if the system is not confident enough that the user is who he claims to be, then it can ask the user to input their password or fingerprint before allowing them to fully use the system.

3.2 Machine Learning Algorithms

According to Arthur Samuel in 1959, machine learning is "the field of study that gives the ability to learn without being explicitly programmed". Nowadays machine learning is used almost everywhere, from filtering spam emails in email accounts to making targeted recommendations for movies, gadgets, books etc based on what someone liked and disliked previously [14].

When someone attempts to solve a problem with machine learning, at the beginning of this process he cannot be fully certain which machine learning algorithm is the best to use to solve the problem at hand. Thus, it is recommended to try various machine learning algorithms before deciding which one yields the best results [15].

Machine learning algorithms could be categorized under three categories:

1. Linear Machine Learning Algorithms. The classification decision of the algorithms under this category is a linear combination of the given attributes. Examples of machine learning algorithms belonging to this category are the Linear Regression algorithm, the Logistic Regression algorithm and the Fisher's Linear Discriminant algorithm also known as Linear Discriminant Analysis [15].

2. Non-Linear Machine Learning Algorithms. The classification decision is not heavily based on assumptions made on the relationship between the given attributes. Some examples of machine learning algorithms belonging to this category are the Naïve Bayes algorithms, the Decision Tree algorithm, the k-Nearest Neighbours algorithm and the Multilayer Perceptron algorithm [15].
3. Ensemble Machine Learning Algorithms. The classification decision is based on predictions from multiple models. Some examples of machine learning algorithms belonging to this category are the Random Forest, the Bootstrap Aggregation, and the Stacked Aggregation algorithm [15].

Multilayer Perceptron (Artificial Neural Network). It is a supervised machine learning algorithm and consists of at least three layers of neurons; the input layer, one or more hidden layers and the output layer. The input layer consists of a set of neurons which represent the input features. The hidden layer(s)' neurons are responsible for transforming the data from the previous layer(s) using linear combination based on the data's weights, followed by a non-linear activation function (e.g. hyperbolic tan function). The role of the output layer is to receive from the last hidden layer the transformed values and further transform them into output values [16].

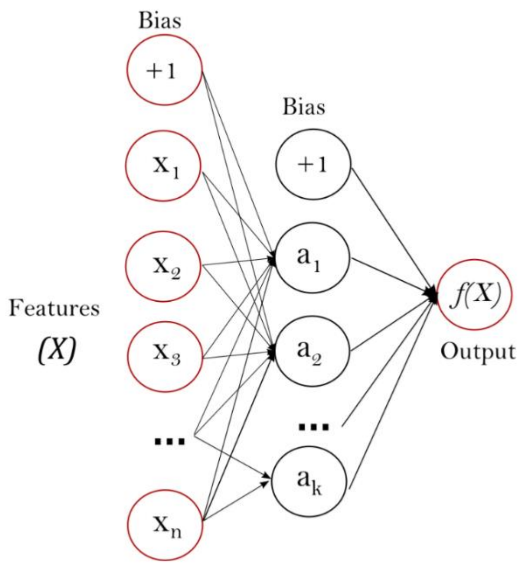


Fig. 1. Multilayer Perceptron with one hidden layer [16].

Naïve Bayes. It is a set of supervised learning algorithms based on applying Bayes' Theorem with the "naïve" assumption of independence between the given features. Some advantages of these algorithms are that they do not require a lot of training data to perform well and they are quick to build and test a model compared to other algorithms, such as the Multilayer Perceptron algorithm. Lastly, they are mostly used for document classification and spam filtering for email accounts [14,17].

Random Forest. The Random Forest algorithm constructs several decision trees while training. Each of those trees gives a classification and from these classifications the final output is calculated. Some of the advantages of this algorithm is that it quickly builds and tests a model, works well on large datasets, and maintains accuracy even when the given dataset has missing values[2]. Moreover, it can be used for classification, regression and feature selection [14].

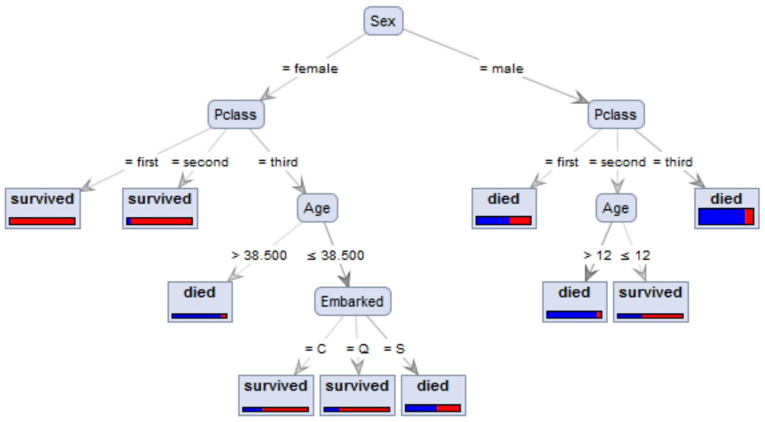


Fig. 2. Visualization of a decision tree created using the Random Forest Algorithm based on the Titanic Dataset provided by the Kaggle [18].

Support Vector Machine. Support Vector Machines are a set of supervised learning algorithms used for classification, regression and outliers' detection. Support Vector Machines take labeled training data and create a model which is capable of assigning each new instance into the appropriate category by creating a hyperplane or a set of hyperplanes in a high or infinite dimensional space. However, there can be several hyperplanes which can offer possible solutions to the problem at hand, but not all them are adequate [19]. Thus, Support Vector Machines calculate an optimal hyperplane which gives the largest minimum distance (known as functional margin) to the nearest training instances of any class, maximising the margin of the training data and producing the best possible solution as the larger the margin is the smaller the generalization error of the classifier is [20].

[2] Based on the Report for Machine Learning Coursework.

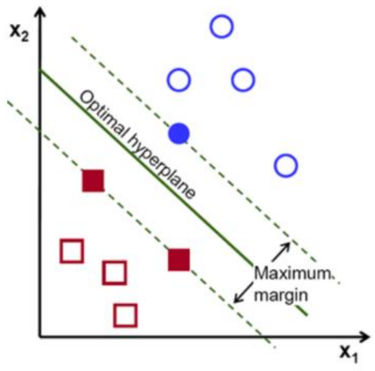


Fig. 3. Optimal Hyperplane of a SVM [19].

3.3 Sensors

Table 2 lists some of the most common sensors found on smartphone devices along with their functionality, a description of what they measure, their possible use in a continuous authentication system and whether Android Permissions are required to access and use each of the listed sensors.

It should be noted that usually the sensors that require permission from the user to be used need more battery to operate and in a continuous authentication system they could drain the device's battery quickly[3].

Table 2. Sensor description, use and user permissions [21].

Sensor	Description	Use	Permission
Accelerometer	Returns the acceleration force data in m/s^2 for the three coordinate axes (x, y and z).	Motion detection	No
Orientation (Gyroscope)	Measures degrees of rotation that a device makes around all three physical axes (x, y, z) in Rad.	Rotation/Motion Detection	No
Gyroscope	Measures the rate of rotation in rad/s around a device's x, y, and z axis.	Motion Detection	No
Uncalibrated Gyroscope	Measures the rate of rotation (without drift compensation) in rad/s around a device's x, y, and z axis.	Motion Detection	No
Magnetometer	Measures the ambient geomagnetic field for all three physical axes (x, y, z) in µT.	Environment Detection	No

[3] Used in the Presentation for CSC 8499

Network (Wi-Fi)	Provides Connection to the Internet.	Connectivity, Location	Yes
Light	Measures the ambient light level (illumination) in lx.	Environment Detection	
Proximity	Measures the proximity of an object in cm relative to the view screen of a device.	Phone Position	No
Pressure	Measures the ambient air pressure in hPa or mbar.	Environment Detection	No
Temperature	Measures the ambient room temperature in degrees Celsius.	Environment Detection	No
GPS	Determines locations and positions.	Tracking	Yes
Microphone	Records Voice.	Speech Recognition	Yes
Camera	Records Video.	Face Recognition	Yes

3.4 Related Work

Shen et al. [22] proposed a framework of multimodal biometric authentication system which continuously and non-intrusively verifies the identity of the current logged-in user. They explored 3 passive biometric modalities; the keystroke biometric, the face biometric and the user's skin colour (soft biometric). The proposed system consists of a fused module and 3 authenticator modules, one for each biometric trait. Each authenticator module calculates a classification score from the collected data. Then the fusion model integrates the calculated scores on the decision level and compares the result with a pre-defined threshold to verify whether the user is legitimate or an imposter.

They discovered that their proposed model performs better when using the face modality alone, compared to using the keystroke biometric modality only. Moreover, they discovered that the system performs better when it uses the combination of skin colour with either one of the two modalities, rather when it uses only one modality. Lastly, they observed that performance of the system is much better when it uses all three modalities together, as it achieves a FAR and FRR of 0% and 0.72%, respectively.

From this research paper, we can clearly see that combining more than two modalities together improves the performance of the system. Moreover, if the correlation between modalities is low then the performance of the system will be higher, as the modalities verify the user's identity from completely different biometric traits. Generally, modalities can be fused together at the score, the decision or the feature level, but the authors of this paper chose to fuse the 3 modalities at the decision level. This choice gives some advantages compared to fusing the modalities at the other two levels. It allows the authenticators to work independently of each other which would not be possible it they were fused at the feature level. Moreover, fusing modalities at the decision level makes it easier to add more modalities to the system later on if deemed necessary.

Lastly, having a small observation window allows the system to authenticate a user faster, however, it also means that it uses less biometric data which can decrease the accuracy of the system.

Internet of Things devices often lack some, if not all, the user interface characteristics (e.g. mouse, keyboard) that traditional devices have [23]. This increases the need to explore user authentication methods for such devices. One solution to this problem is the use of continuous authentication. Shahzad et al., proposed a system which is capable of continuously authenticating users for the Internet of Things devices. The proposed system, WifiU, uses Wi-Fi signals to distinguish and recognize users from their gait. It is made of two Wi-Fi devices, one of the devices continuously sends Wi-Fi signals and the other continuously accepts Wi-Fi signals and measures the channel state information (CSI) of each signal.

This research paper, demonstrates that the proposed system is easier to be deployed compared to other authentication systems which also collect gait information to verify the user. This is because WifiU does not need any special hardware such as cameras, floor sensors or any other kind of wearable sensors to collect gait information unlike previously proposed gait recognition systems [24]. However, extracting gait information from the channel state information (CSI) signal is not an easy task, as the signal includes information of all the user's moving body parts. Moreover, more Wi-Fi sender-receivers can be easily added to increase the system's performance and thus more people can concurrently be recognized. However, this system cannot be used in systems with high-risk security as an attacker could easily intercept the Wi-Fi signals to identify users. This poses new threats to the security and privacy of users.

Feng et al., developed a framework which uses the fingerprint, accelerometer and motion sensors of a smartphone device to implicitly and continuously authenticate the owner of the device [25]. They collected the fingerprint from the fingerprint sensor to identify possible user identity changes and then they used the other two sensors to detect device-leaving-hands events. From the data they collected, they produced statistics related to who the legitimate user gave his smartphone to and why he lent it to the other person. Additionally, they used the data to train a preliminary model and presented the results from the model to the participants of the study. They further improved the model based on the feedback of the users. The system they proposed manages to reduce unauthorized application access and unnecessary authentication events by more than 90% and 85% respectively.

A lot of data is needed to be collected for the system to accurately detect device-leaving-hand events in an uncontrolled environment. That is why the authors of this paper, asked the participants to follow a set of instructions to collect the necessary amount of data. The users could perform this instruction in any way they pleased. Additionally, this system not only improves security for smartphones but also improves usability as the user does not have to perform as many unnecessary authentication events compared to other systems.

Wu et al. proposed a model which identifies a user based on the way he interacts with the device's touchscreen. The model records the location where a user touched on the screen, the area of the screen which was touched, the pressure applied, the direction in which the user moved his finger and the velocity of the movement [26]. They deduced that their method verifies the user's identity with approximately 98% of accuracy and if the system is combined with current data leakage prevention systems it can significantly reduce the workload of this DLP systems.

This paper demonstrates that users have a unique way of typing and using a touchscreen and that it is not always necessary for users to complete a set of predefined tasks to collect the necessary data for training the model, as other research authors did. Additionally, other researchers advise that authentication based on touch-motion interaction should be combined with other biometric traits of a user to guarantee security.

Kumar et al., developed a fusion-based authentication mechanism for smartphone devices [27]. They collected the users' swiping patterns, typing patterns and phone movement patterns. From the study, they discovered that phone movement patterns were present during the entire interaction time, but typing and swiping did not occur continuously or simultaneously. They also discovered that combing phone movement patterns and swiping achieves better accuracy, compared to combining phone movement patterns and typing.

This paper comes to agreement with other papers mentioned previously that support that combing biometric modalities improves the system's performance. Additionally, this combination of biometric modalities makes it difficult for an attacker to attack the system, as he will have to produce data for all modalities instead of just one, at the same time. Moreover, they discovered that combining the swiping patterns with the corresponding phone movements at the feature level, the accuracy performance of the system is better than when the modalities are fused at the score level. This adds to the paper from Shen et al. [22] that supports that fusion of modalities at decision-levels improves the performance of a system.

Lee et al., proposed a system which continuously authenticates and distinguishes the smartphone user from unauthorized users after the login stage, without disturbing the normal usage of the device. The proposed system collects and uses data from the accelerometer, orientation and magnetometer sensors and then uses the SVM machine learning technique to build and train the user's profile [28]. The proposed system needs ten seconds to train the user's profile, twenty seconds to detect an unauthorized user and achieves more than 90% of authentication accuracy. Moreover, their proposed method does not need users to complete a predefined set of tasks in order to achieve high accuracy performance as other proposed systems explored in this report.

Some notable observations made from this paper are that when increasing the sampling rate or the size of the training data, we increase the accuracy of the system, as more data is used to train the model. However, the higher the sampling rate is, the more time is needed to train the user's profile. Generally, the time needed to train the user's profile increases exponentially when the sample rate or the size of training data are increased.

In conclusion, this sub-section provides an overview of existing Continuous authentication systems for various applications. Each system provides us with a solid understanding of the challenges in developing a Continuous authentication system. Moreover, we have learnt that combining biometric characteristics which have a low correlation improves the system's performance. However, this increases the size of data that is being collected and processed by the system and this can affect the system's usability. Lastly, there is no universal machine learning technique which works well with all systems. For each system, multiple experiments must be first carried out on the collected data with various machine learning algorithms before deciding which algorithm works best. In other words, this is a trial and error process to find the best model for the problem at hand.

4 Design

4.1 User Characteristics and Sensor Selection

As mentioned in section 3.1, Continuous authentication systems usually use the physiological and behavioural characteristics of a user to continuously verify whether the user is who he claims to be. Additionally, from the research conducted for section 3.4 of this report, one can deduct that combining multiple biometric characteristics to authenticate the user improves the performance of such systems and offers greater protection against attackers, as they will have to spoof all characteristics at once. For these reasons, the system is designed to collect information about how the user interacts with a smartphone device along with information about the user's environment.
To build a model to verify the user's identity, data is going to be collected through three scenarios. The first scenario is when the user is sitting and holding his device, the second scenario is when the phone is flat on a desk while the user uses the phone and the last scenario is while the phone is in the user's pocket and he is walking from point A to B. In each of these scenarios the system collects data about the Wi-Fi network, and particularly the BSSIDs (the MAC address of the wireless access point (WAP)) of the wireless connections available around the user.

The designed system uses the accelerometer sensor to collect coarse-grained motion of a user, the magnetometer sensor to measure the orientation of the device and collect fine-grained motion of a user, and the wireless capabilities of a phone to collect the BSSIDs of the available wireless connections.

The accelerometer sensor measures the force applied to the device, including the gravitational force [21], using Equation (1). The magnetometer is used in combination with the accelerometer to calculate the device's orientation. It provides data about three orientation angles; the Azimuth (degrees of rotation about the -z axis), the Pitch (degrees of rotation about the x axis) and the Roll (degrees of rotation about the y axis) [21]. Lastly, applications for Android devices can use the Wi-Fi API to communicate with the wireless stack. This stack is responsible for providing Wi-Fi Network Access, which will allow the designed system to collect the specified Wi-Fi information.

$$A_D = -g - \left(\frac{1}{mass}\right)\sum F_s \qquad (1)\ [21]$$

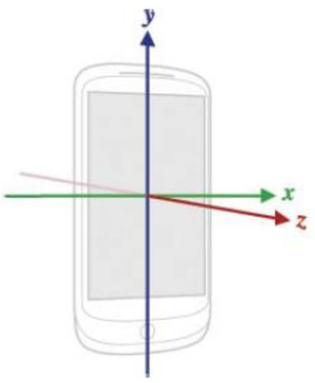


Fig. 4. Android Coordinate System [29].

Figure 1 shows the Android 3-axis coordinate system which is used to express values for the accelerometer and magnetometer sensors. When the phone is in an upright position the x-axis is horizontal and points to the right, the y-axis is vertical and points upwards and the z-axis is horizontal and points outwards [29].

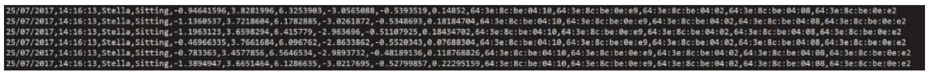
4.2 System Design

As mentioned in section 4.1, data from the accelerometer sensor, magnetometer sensor and the BSSIDs of the available wireless connections around the user are being collected every 0.25s. Moreover, the date, time, user's name and the device's mode are recorded as well. This data is saved in a comma-separated value (CSV) file on the device's memory. At the beginning of this file there is a Header, which describes the columns of the file, that correspond to the data that the system collects every second. Table 3 lists the number of columns and the name of each column in the CSV file and figure 6 provides an example of data collected from the user Stella using the smartphone device while sitting. This file is created each time the user is performing the set of instructions given to him for the three scenarios.

Table 3. Columns of the CSV file.

Column No.	Data Collected
1	Date
2	Time
3	ID (Name of User)
4	Position (Sitting, Desk, Walking)
5	ACCX (Accelerometer X-axis, m/s^2)
6	ACCY (Accelerometer Y-axis, m/s^2)
7	ACCZ (Accelerometer Z-axis, m/s^2)
8	ORX (Orientation X-axis)

9	ORY (Orientation Y-axis)
10	ORZ (Orientation Z-axis)
11	WIFI1
12	WIFI2
13	WIFI3
14	WIFI4
15	WIFI5



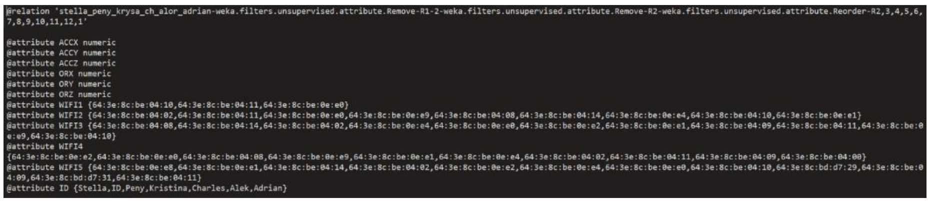
```
25/07/2017,14:16:13,Stella,Sitting,-0.94641596,3.8281996,6.3253903,-3.0565088,-0.5393519,0.14852,64:3e:8c:be:04:10,64:3e:8c:be:0e:e9,64:3e:8c:be:04:02,64:3e:8c:be:04:08,64:3e:8c:be:0e:e2
25/07/2017,14:16:13,Stella,Sitting,-1.1360537,3.7218604,6.1782885,-3.0261872,-0.5348693,0.18184704,64:3e:8c:be:04:10,64:3e:8c:be:0e:e9,64:3e:8c:be:04:02,64:3e:8c:be:04:08,64:3e:8c:be:0e:e2
25/07/2017,14:16:13,Stella,Sitting,-1.1963123,3.6598294,6.415779,-2.963696,-0.51107925,0.18434702,64:3e:8c:be:04:10,64:3e:8c:be:0e:e9,64:3e:8c:be:04:02,64:3e:8c:be:04:08,64:3e:8c:be:0e:e2
25/07/2017,14:16:13,Stella,Sitting,-0.46966335,3.7661684,6.096762,-2.8633862,-0.5520343,0.07688304,64:3e:8c:be:04:10,64:3e:8c:be:0e:e9,64:3e:8c:be:04:02,64:3e:8c:be:04:08,64:3e:8c:be:0e:e2
25/07/2017,14:16:13,Stella,Sitting,-0.783363,3.4577856,6.5646534,-2.9893732,-0.48189136,0.118768826,64:3e:8c:be:04:10,64:3e:8c:be:0e:e9,64:3e:8c:be:04:02,64:3e:8c:be:04:08,64:3e:8c:be:0e:e2
25/07/2017,14:16:13,Stella,Sitting,-1.3894947,3.6651464,6.1286635,-3.0217695,-0.52799857,0.22295159,64:3e:8c:be:04:10,64:3e:8c:be:0e:e9,64:3e:8c:be:04:02,64:3e:8c:be:04:08,64:3e:8c:be:0e:e2
```

Fig. 5. Example data in initial CSV file

After the user has completed the set of instructions, the CSV file is manually imported into Weka for further processing. Once in Weka, the Date, Time and Position columns are removed from the file and the ID column is marked as the class column and becomes the last column of the file. In Weka, there must be a class column as this defines what is being classified.

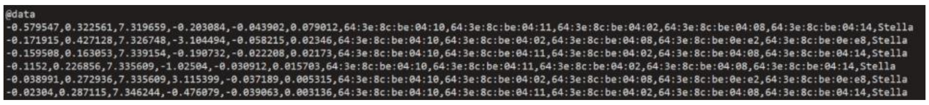
Once the file is processed it is then saved as an Attribute-Relation File Format (ARFF). This is an ASCII text file which describes a list of instances sharing a set of attributes, and is used by Weka when building and testing a model. Moreover, this file is divided in two sections the Header and the Data section [30]. The Header section consists of the relation declaration, a list of attributes (the columns of the CSV file) and their type. Lastly, the Data section consists of the data declaration line and the actual instance lines [30].

Moreover, because the BSSIDs are initially in nominal format, meaning they consist of characters and numbers, they must be translated into numeric values as some machine learning algorithms provided by Weka cannot handle columns being nominal if they are not the class column. Thus, each unique BSSID in the file is converted into a unique number.



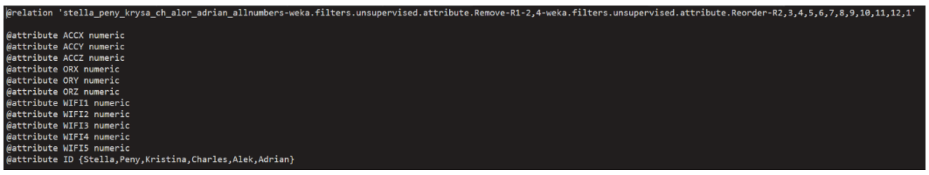
```
@relation 'stella_peny_krysa_ch_alor_adrian-weka.filters.unsupervised.attribute.Remove-R1-2-weka.filters.unsupervised.attribute.Remove-R2-weka.filters.unsupervised.attribute.Reorder-R2,3,4,5,6,
7,8,9,10,11,12,1'

@attribute ACCX numeric
@attribute ACCY numeric
@attribute ACCZ numeric
@attribute ORX numeric
@attribute ORY numeric
@attribute ORZ numeric
@attribute WIFI1 {64:3e:8c:be:04:10,64:3e:8c:be:04:11,64:3e:8c:be:0e:e0}
@attribute WIFI2 {64:3e:8c:be:04:02,64:3e:8c:be:04:11,64:3e:8c:be:0e:e0,64:3e:8c:be:0e:e9,64:3e:8c:be:04:08,64:3e:8c:be:04:14,64:3e:8c:be:0e:e4,64:3e:8c:be:04:10,64:3e:8c:be:0e:e1}
@attribute WIFI3 {64:3e:8c:be:04:08,64:3e:8c:be:04:14,64:3e:8c:be:04:02,64:3e:8c:be:0e:e4,64:3e:8c:be:0e:e0,64:3e:8c:be:0e:e2,64:3e:8c:be:0e:e1,64:3e:8c:be:04:09,64:3e:8c:be:04:11,64:3e:8c:be:0
e:e9,64:3e:8c:be:04:10}
@attribute WIFI4
{64:3e:8c:be:0e:e2,64:3e:8c:be:0e:e0,64:3e:8c:be:04:08,64:3e:8c:be:0e:e9,64:3e:8c:be:0e:e1,64:3e:8c:be:0e:e4,64:3e:8c:be:04:02,64:3e:8c:be:04:11,64:3e:8c:be:04:09,64:3e:8c:be:04:00}
@attribute WIFI5 {64:3e:8c:be:0e:e8,64:3e:8c:be:0e:e1,64:3e:8c:be:04:14,64:3e:8c:be:04:02,64:3e:8c:be:0e:e2,64:3e:8c:be:0e:e4,64:3e:8c:be:0e:e0,64:3e:8c:be:04:10,64:3e:8c:bd:d7:29,64:3e:8c:be:0
4:09,64:3e:8c:bd:d7:31,64:3e:8c:be:04:11}
@attribute ID {Stella,ID,Peny,Kristina,Charles,Alek,Adrian}
```

Fig. 6. The initial Header of the ARFF File (before the BSSID columns are converted into numeric values)



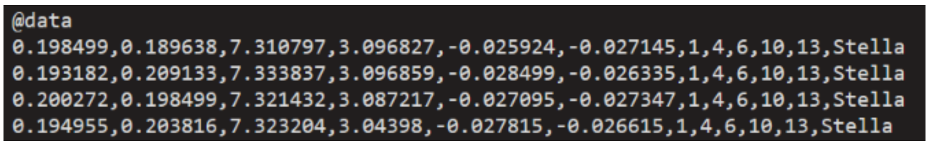
```
@data
-0.579547,0.322561,7.319659,-0.203084,-0.043902,0.079012,64:3e:8c:be:04:10,64:3e:8c:be:04:11,64:3e:8c:be:04:02,64:3e:8c:be:04:08,64:3e:8c:be:04:14,Stella
-0.171915,0.427128,7.326748,-3.104494,-0.058215,0.02346,64:3e:8c:be:04:10,64:3e:8c:be:04:02,64:3e:8c:be:04:08,64:3e:8c:be:0e:e2,64:3e:8c:be:0e:e8,Stella
-0.159508,0.163053,7.339154,-0.190732,-0.022208,0.02173,64:3e:8c:be:04:10,64:3e:8c:be:04:11,64:3e:8c:be:04:02,64:3e:8c:be:04:08,64:3e:8c:be:04:14,Stella
-0.1152,0.226856,7.335609,-1.02504,-0.030912,0.015703,64:3e:8c:be:04:10,64:3e:8c:be:04:11,64:3e:8c:be:04:02,64:3e:8c:be:04:08,64:3e:8c:be:04:14,Stella
-0.038991,0.272936,7.335609,3.115399,-0.037189,0.005315,64:3e:8c:be:04:10,64:3e:8c:be:04:02,64:3e:8c:be:04:08,64:3e:8c:be:0e:e2,64:3e:8c:be:0e:e8,Stella
-0.02304,0.287115,7.346244,-0.476079,-0.039063,0.003136,64:3e:8c:be:04:10,64:3e:8c:be:04:11,64:3e:8c:be:04:02,64:3e:8c:be:04:08,64:3e:8c:be:04:14,Stella
```

Fig. 7. Part of the initial Data section of the ARFF File (before the BSSID instances are converted into numeric values)



```
@relation 'stella_peny_krysa_ch_alor_adrian_allnumbers-weka.filters.unsupervised.attribute.Remove-R1-2,4-weka.filters.unsupervised.attribute.Reorder-R2,3,4,5,6,7,8,9,10,11,12,1'

@attribute ACCX numeric
@attribute ACCY numeric
@attribute ACCZ numeric
@attribute ORX numeric
@attribute ORY numeric
@attribute ORZ numeric
@attribute WIFI1 numeric
@attribute WIFI2 numeric
@attribute WIFI3 numeric
@attribute WIFI4 numeric
@attribute WIFI5 numeric
@attribute ID {Stella,Peny,Kristina,Charles,Alek,Adrian}
```

Fig. 8. The Header section of the ARFF File (after the BSSID columns are converted into numeric values)



```
@data
0.198499,0.189638,7.310797,3.096827,-0.025924,-0.027145,1,4,6,10,13,Stella
0.193182,0.209133,7.333837,3.096859,-0.028499,-0.026335,1,4,6,10,13,Stella
0.200272,0.198499,7.321432,3.087217,-0.027095,-0.027347,1,4,6,10,13,Stella
0.194955,0.203816,7.323204,3.04398,-0.027815,-0.026615,1,4,6,10,13,Stella
```

Fig. 9. Part of the Data section of the ARFF File (after the BSSID instances are converted into numeric values)

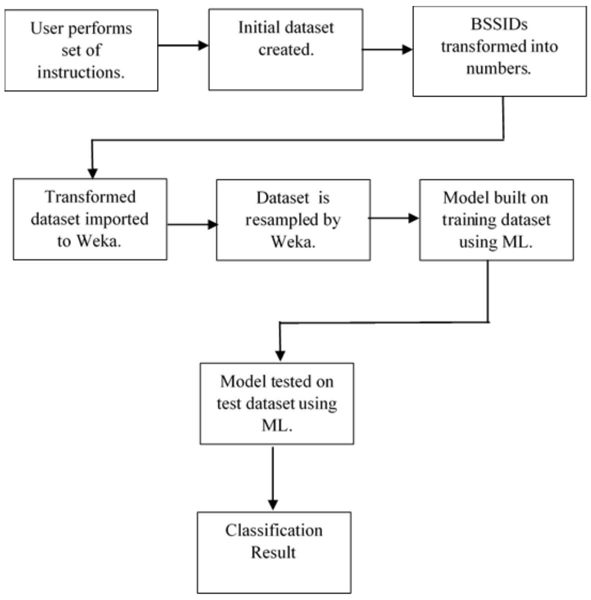

User performs set of instructions.
Initial dataset created.
BSSIDs transformed into numbers.
Transformed dataset imported to Weka.
Dataset is resampled by Weka.
Model built on training dataset using ML.
Model tested on test dataset using ML.
Classification Result

Fig. 10. System Design.

4.3 Tools and Language Used

Google Nexus 5x. The system was developed and tested on the Google Nexus 5x smartphone which runs on the Android Operating System v6.0.1 (Marshmallow) API level 23. This device was released in October 2015 and has the unmodified Android stack. Table 4 lists the key specifications of the Nexus 5X.

Table 4. Specifications of Google Nexus 5X [31].

Processor	Qualcomm MSM8992 Snapdragon 808 Hexa-core (4x1.4 GHz Cortex-A53 & 2x1.8 GHz Cortex-A57)
Operating System	Android 6.0.1 (Marshmallow)
Memory	32 GB flash memory, 2 GB RAM
WLAN	Wi-Fi 802.11 a/b/g/n/ac, dual-band, Wi-Fi Direct, DLNA, hotspot
Display	5.2 inches (1080 x 1920)
Battery	2,700mAh
Sensors	Fingerprint (Rear-Mounted) Sensor Accelerometer Sensor Magnetometer Sensor Gyroscope Sensor Proximity Sensor Compass Barometer
Dimensions	147 x 72.6 x 7.9mm

Android. Android is one of the most popular open-source mobile Operating Systems in the world, owned by Google. It is based on the Linux Kernel and most of its applications are developed in the Java language. The system was developed using Android Studio, as is the official Android IDE and provides useful tools to build applications for any type of Android device [32].

Weka. Weka is an open-source software that provides one with a vast collection of machine learning algorithms that can either be applied directly to datasets using Weka's Explorer Graphical User Interface or called directly from any Java code. Furthermore, Weka provides tools for data pre-processing, classification, regression, clustering, association rules and visualization [33].

Using Weka to build and test a model has many advantages because as mentioned in section 3.3, when one starts working with a machine learning problem it is a good practice to try several machine learning algorithms before deciding which one gives the best results for the problem at hand. Thus, using Weka is quite easy and not as time-consuming compared to other ways of applying and testing how various machine learning algorithms perform with the problem at hand.

5 Work Undertaken

At the beginning of this project different Continuous authentication systems were examined and a comprehensive insight into the challenges of creating a strong Continuous authentication system was gained. Moreover, based on this research it was decided that the system should collect the user's behavioural characteristics (how he uses the device) along with information about his environment. As a result, the designed system will perform better as it is verifying the user's claimed identity on multiple modalities and it will be harder for an attacker to successfully attack the system. Lastly, it was decided to give the users a set of instructions to perform, in order to collect sufficient amount of data to build and test the model.

Furthermore, various sensors found on Android devices were examined in order to understand their functionality and to determine their use in a Continuous authentication system. Based on this, the sensors that were selected for this project to collect information were the accelerometer, magnetometer, and the Wi-Fi sensors.

Once all the data was collected from the six participants for each scenario, the BSSIDs were transformed into numbers, as having them in nominal form affects some machine learning algorithms when training and testing a model. Then the transformed data was divided randomly in two datasets, the training dataset which contained 70% of the transformed data. This dataset is then used to build a model with the chosen machine learning algorithms and the rest of the data is put into a test dataset, which is then used to test the performance of the model.

5.1 Experiments performed with the Accelerometer Sensor

Initially, the system was collecting the raw data from the accelerometer sensor, and calculating the change between two consecutive accelerometer sensor events for the x, y and z axes. The system then checked whether these calculated values were lower than a threshold value, which was set to 2. If the calculated values were lower than the defined threshold, they were considered as noise and each was replaced with a zero before being written to the file. If they were above the threshold value they were written into the file unchanged.

It was quickly discovered that this method was not satisfactory for the proposed system when a participant performed the given set of instructions two times and a model was built with the 70% of the collected data and tested on the 30%. Having a look at the CSV file were all the data collected from the three sensors was stored, it could be seen that the system was writing a lot of zeros for the x, y and z axes of the accelerometer sensor, and this was also affecting the x, y and z values of the collected orientation. This is due to the fact that in Android the orientation of the device is calculated using values from both the accelerometer and magnetometer sensors. The overall performance of the system using this method was not satisfactory as a lot of subtle movement by the user was getting lost.

In another experiment, the system was tested using the collected data of the accelerometer sensor unchanged. The same participant once again performed the same set of instructions two times and again the system's performance was not satisfactory. By examining the CSV file this time, it could be seen that there was a lot of jitter in the data. Thus, more research was carried out into filtering accelerometer data for a Continuous authentication system and it was decided to use a low-pass filter.

A low-pass filter is a filter that allows low frequency signals to pass unchanged, reducing signals which are above the cut-off frequency. In order to use a low-pass filter in a digital application such as this one, discrete-time must be used to sample the accelerometer values at discrete time intervals. A value, called alpha, is used to determine how much the data is smoothed. This alpha value is defined in equation (2). Time Constant (dt) is the relative duration of the signal that the filter will be applied on, meaning that signals that are longer than the time constant will pass through unchanged but signals shorter than the time constant will be ignored. The Delivery Rate (RC) is the event delivery rate (the time between two consecutive accelerometer events) [34].

To sum up, having a smaller alpha value means more smoothing of the data, and having a larger alpha value means more data can get pass the filter, thus finding a good alpha value is important.

$$alpha = \frac{dt}{(\mathrm{RC+dt})} \tag{2}$$

Moreover, a digital system can have either a static or a dynamic alpha. For this system, a static alpha was selected. Firstly, the alpha was set to 0.5 and allowed most of the data to pass through. This value did not reduce jitter much and did not improve the system’s performance significantly, especially when compared to the previous method of simply using the raw data unchanged. Thus, the alpha value was set to 0.26 for this system, as it smooths enough the data to reduce noise without reducing the system’s performance.

Lastly, the alpha value can be calculated dynamically depending on the sampling rate of the sensor on the device. Calculating alpha dynamically was attempted. However, it did not work well and thus the static value is used.

5.2 Experiments performed with various Machine Learning Algorithms

Initially, it was decided to use data from the first three times the users performed the set of instructions given to them for each scenario to train the model, and use the data collected the fourth time to test the model.

The model was built and tested using various machine learning algorithms in order to check which machine learning algorithm suits better this system. The machine learning algorithms used are the Random Forest algorithm, the Support Vector Machine algorithm, the Naïve Bayes algorithm, the Multilayer Perceptron algorithm and lastly the Decision Tree Algorithm. Generally, the system’s performance was not satisfactory, especially when compared to the performance of existing continuous authentication systems studied in this report. For the Support Vector Machine, the Naïve Bayes and the Decision Tree algorithms, the accuracy of the model was lower than 75% percent in all scenarios and for the Multilayer Perceptron and Random Forest algorithms the accuracy did not exceed 85%. This clearly showed that there was an error with the built models and thus more research was carried out into Weka’s Wiki and StackOverflow discussions to improve the model’s performance.

One of the solutions was to change some of the default settings of the given algorithms. The models were built and tested again, using the same training and test datasets, and machine learning algorithms as before but with different algorithms settings. There was some improvement in the system’s performance for some of the algorithms, but still it was not satisfactory.

Another solution found, this time in the research papers studied for this project, was to combine the data from all four experiments into a file and then randomly select roughly 70% of the data to be used for training the models and roughly 30% for testing the models, while verifying that the two files did not contain the same or partially the same data to each other. To achieve this the file was imported into Weka and through the Pre-Processing tab in Weka’s Explorer Graphical User Interface, a resampling filter was applied and the data was divided into two datasets which contained unique data from each other.

Furthermore, using the dataset which contained 70% of the initial data, models were built using the same algorithms with their default settings as before. The models were then tested using the other dataset which contained 30% of the initial data.

The system’s performance improved significantly and it was finally decided to use the Naïve Bayes and Multilayer Perceptron algorithms for this project. This is since they are both supervised algorithms and each one bases its classification on different factors. It should be noted that to my best knowledge there no Continuous authentication systems that use these algorithms.

6 Implementation

6.1 Permissions Request

Since Android 6.0, the way that an application requests permission to access resources and information has changed. Permissions are split into two categories; normal and dangerous. Both types of permissions have to be listed into the application’s manifest but they differ in one significant way. Normal permissions do not risk the user’s privacy and the system’s integrity thus, the system automatically grants them. However, dangerous permissions pose a risk to the system’s integrity and user’s privacy as they allow access to the user’s confidential and sensitive data. Thus, the user must explicitly grant specific permissions to the application at run-time [35].

The designed system uses the phone’s wireless capabilities to access and collect information about the available wireless connections. It also writes all the data from the accelerometer sensor, the magnetometer sensor and the wireless connections’ BSSIDs in the device’s external memory. Thus, the application must explicitly request from the user to grant the permissions to access the Location Services API, to access the Wi-Fi state and to write at the external memory at run time. This is achieved by calling the *permissionGrant* method in the *MainActivity* class. Every time the application needs to use a dangerous permission, this method is called and it is responsible of checking whether the application has the specific permission. The *permissionGrant* method performs this check by calling the *ContextCompat.checkSelfPermission* method. If the application has the permission, the *ContextCompat.checkSelfPermission* method returns *PackageManager.PERMISSION_GRANTED* and the application can proceed with using the resource. However, if the application does not have the required permission, it asks the user to grant the permission and the user can either allow or deny the permission.

The *permissionGrant* method also provides the user with an explanation as to why it needs to access a specific permission, in the case where a user has previously denied the permission request and then tried to access the resource. So, the user is not overwhelmed with unnecessary messages. This functionality is handled with the utility method *shouldShowRequestPermissionRationale*, which returns true if the application has requested this permission before and the user denied the request [35].

If the application does not have a required permission, the *permissionGrant* method invokes one of the *requestPermissions* methods to request the permission. To achieve this the *permissionGrant* method passes to the invoked *requestPermissions* method the required permission along with an integer. The integer was specified at the beginning of the *MainActivity* class, in order to identify the permission request. This method can be seen in Appendix A1. When the user responds to the request the system invokes the *onRequestsPermissionResult* method which is found in the *MainActivity* class and passes to the *onRequestsPermissionResult* method, the user's respond and the request code that was passed to the invoked *requestPermissions* method. Finally, the application overrides this method to determine whether or not the permission is granted. This method can be found at Appendix A2.

6.2 User Interface

The proposed system consists of a simple user interface, which allows setting the participant's name and which scenario they are about to complete. Moreover, the user interface has two buttons; start and stop. The start button triggers the background service which is responsible for collecting the data from the chosen sensors while the user is using his phone. The stop button is responsible for terminating the background service after the user has completed the set of instructions he was given. The user interface is needed to collect the data from the sensors in a controlled way.

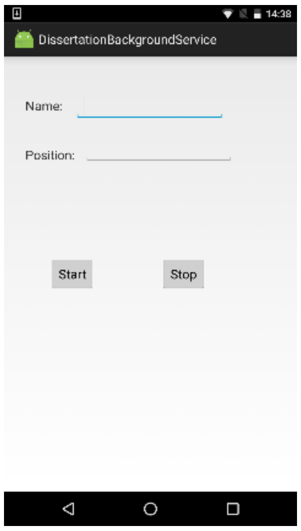


Fig. 11. Application’s User Interface

6.3 Background Service

The designed system relies on a single background service, as other applications on the device will be accessed by the participants to complete the set of instructions they are given. This background service is responsible for invoking the necessary methods to collect data from the participants while they are using the phone.

Firstly, an explicit intent for the Service class in the *onCreate* method of the *MainActivity* class was created and is passed to the *startService* method. The *startService* method is called when the start button is clicked. Once the *startService* is called, the *onStartCommand* method in *MyService* class is invoked [36].

The *onCommandStart* method is responsible for initializing an Arraylist of Strings in which the BSSIDs are stored into before they are written into the CSV file. Then, the method calls the *writeHeader* method which is responsible for creating a folder in the device’s external storage, creating a CSV file in the created folder and then writing in the CSV file the defined header. If the folder and the CSV file already exist, the *writeHeader* method only appends at the end of the CSV file the defined header, without deleting any data that was written previously. This prevents the application from accidently deleting previously written data and also allows easier manipulation of the CSV file. This is the case as a new header is added at the start of an experiment, thus making it easier to distinguish a new user or new experiment in the same file.

Moreover, to use the device’s sensors a reference to the sensor service is created by creating an instance of the *SensorManager* class through the *getSystemService* method and passing to the *getSystemService* method the sensor service.

The *onCommandStart* method checks whether the specific sensor exists on the device that the application is installed on, using the *getDefaultSensor* method. If the sensors are available the method invokes the *registerListener* method for each sensor and specifies the sampling rate of the sensor. In the application, the sampling rate is set to default delay, which uses a delay of 20,000 milliseconds [37].

Furthermore, the *onCommandStart* method starts scanning for the available wireless connections by calling the *scanID* method which uses the inner *WifiReceiver* class to collect the BSSIDs.

Lastly, the *onCommandStart* returns *START_STICKY*, which recreates and restarts the service if it is killed by the operating system after the method has returned. This, verifies that the service is stopped only when the stop button is clicked and the stopService method is invoked.

```
@Override
public int onStartCommand(Intent intent, int flags, int
startId){
    bssidList = new ArrayList<String>();
    MainActivity  m = new MainActivity();
    strID = MainActivity.getName();
    strPosition = m.getPosition();
    Log.v("Name", strID);
    Log.v("Position", strPosition);
    try {
        writeHeader();
    } catch (IOException e) {
        e.printStackTrace();
    }
                                wifiManager = (WifiManager)
getSystemService(Context.WIFI_SERVICE);
    wasEnabled = wifiManager.isWifiEnabled();
    if (wifiManager.isWifiEnabled() == false) {
        Toast.makeText(getApplicationContext(), "wifi is disa-
bled..making it enabled", Toast.LENGTH_LONG).show();
        wifiManager.setWifiEnabled(true);
    }
    sensorManager = (SensorManager) getSystemService(Con-
text.SENSOR_SERVICE);
    accelerometer = sensorManager.getDefaultSensor(Sen-
sor.TYPE_ACCELEROMETER);
    if(accelerometer != null) {
            sensorManager.registerListener(this, accelerometer,
SensorManager.SENSOR_DELAY_NORMAL);

    }
    magnetometer = sensorManager.getDefaultSensor
         (Sensor.TYPE_MAGNETIC_FIELD);
    if(magnetometer!=null) {
        sensorManager.registerListener(this, magnetometer,
         SensorManager.SENSOR_DELAY_NORMAL);
    }
    scanID();
    return  START_STICKY;
}
```

6.4 Available Wireless Connections

The application detects the available wireless connections around the user through the *WifiReceiver* inner class. This class extends the *BroadcastReceiver* class and its *onReceive* method is called every time that the number of available wireless connections changes. The *onReceive* method firstly saves all the available connections in a List of *ScanResults*, it goes through this list and for each saved wireless connection extracts its BSSIDs and adds the extracted BSSIDs into a List of Strings called *bssidList*. Lastly, before this method sets the *wifiFlagList* to true, it checks that the *bssidList* is not empty.

6.5 Data collection

The application overrides and implements the *onSensorChanged* method to collect data about how the user interacts with the device through the accelerometer and magnetometer sensors. This method is called every time there is a new sensor event, and initializes one array of floats and two String variables; the *accelerEntry* and the *orientationEntry*. These strings are used by the method later on to construct a string consisting of the measured values from the two sensors.

Furthermore, every time this method detects a sensor event from the accelerometer sensor it passes the measured values to the lowpass method. This lowpass method filters values from noise before the *onSensorChanged* method can save the newly filtered values into the *accelerometerArr*, and can set the *isFlagAccel* parameter to true. If the sensor event detected by the *onSensorChanged* method is from the magnetometer sensor, the measured values are saved into the *magnetometerArr* and the *isFlagMagn* parameter is set to true.

The collected values from the magnetometer sensor must be further processed to get the orientation of the device. The *onSensorChanged* method checks if the *isFlagAccel* and *isFlagMagn* are set to true, before calling the *updateOrientationAngles* method. This check is important because the *updateOrientationAngles* method calculates the device's orientation by using the values collected from both the accelerometer and magnetometer sensors. As a result, the system must verify that events from both sensors occurred before invoking the *updateOrientationAngles* method so it does not break the execution of the application by throwing *NullPointerException*.

Once the device's orientation is calculated, the *onSensorChanged* method checks if all three flags in the class are true before constructing the *accelerEntry* and *orientetionEntry* strings. Moreover, this method passes these two strings to the *printToFileFinal* method and sets the *isFlagAccel* and *isFlagMagn* to false. It should be noted that the *wifiFlagList* is not set to false here because the functionality of collecting the available wireless connections' BSSIDs is managed by the inner class.

```
@Override
public void onSensorChanged(SensorEvent event) {
    String accelerEntry = "";
    String orientationEntry="";
    float [] result;
    if (event.sensor.getType() == Sensor.TYPE_ACCELEROMETER) {
        result = lowPass(event.values);
        accelerometerArr[0] = result[0];
        accelerometerArr[1] = result[1];
        accelerometerArr[2] = result[2];
        isflagAccel = true;
    }
    if (event.sensor.getType() == Sensor.TYPE_MAGNETIC_FIELD) {
       magnemetomerArr = event.values;
        isFlagMagn = true;
    }
    if (isflagAccel && isFlagMagn) {
        updateOrientationAngles();
    }
    if(isflagAccel && isFlagMagn && wifiFlagList) {
        accelerEntry = accelerometerArr[0] + "," +
         accelerometerArr[1]+ "," + accelerometerArr[2] + ",";
        orientationEntry = mOrientation[0] +"," +
         mOrientation[1] +"," + mOrientation[2];
        printToFileFinal(accelerEntry,orientationEntry);
        isflagAccel=false;
        isFlagMagn=false;
    }
}


public void updateOrientationAngles() {
    // Update rotation matrix, which is needed to update orien-
tation angles.
    sensorManager.getRotationMatrix(mR, null, accelerometerArr,
magnemetomerArr);
    // "mR" now has up-to-date information.
   sensorManager.getOrientation(mR, mOrientation);
    // "mOrientation" now has up-to-date information.
}
```

7 Testing and Results

Testing of the system was performed using two Nexus 5X devices and six participants. Five of the participants are students at Newcastle University and one participant is a lecturer. Moreover, all participants were right handed, four participants were male and two were female. The participants were asked to follow a set of instructions for three scenarios and repeat the three scenarios four times.

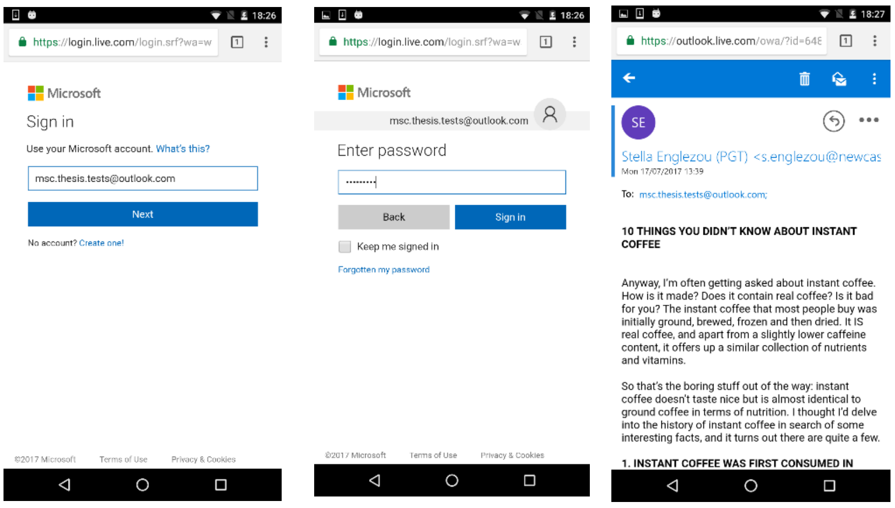
For the first scenario, the participants were asked to be seated and use the phone as they normally would in their everyday life to execute the following instructions after they were given the phone:

1. When ready click the start button in the Application's interface.
2. Go back to the Main Screen of the phone, and open the Google Chrome Application.
3. Once presented with the Outlook Login Page, use the following email address; msc.thesis.tests@outlook.com and password; Msc123test to login into the Outlook account.
4. Once logged in, read the email with subject "Email 2".
5. After reading the email, go back to the Application and press the stop button.

Figure **3** shows the screens for Steps 3 and 4 from the set of instructions.

The second scenario is similar to the first one as the participants had to follow the same set of instructions; to click the start button once ready, then go to the Google Chrome Application, login into the Outlook Account using the same credentials and read the same email as in scenario 1 and once they were finished to go back to the application and click the stop button. The only difference from the first scenario is that they had to use the phone while it was flat on the desk and they were not allowed to pick it up at any time.

Lastly, for the third scenario the participants were asked to press the start button on the application and then put the device in their right-front pocket and walk from the Access Lab down one floor, and at the end of this walk to click the stop button before they return the device.



Stella Englezou (PGT) <s.englezou@newcas
Mon 17/07/2017 13:39

To: msc.thesis.tests@outlook.com;

10 THINGS YOU DIDN'T KNOW ABOUT INSTANT COFFEE

Anyway, I'm often getting asked about instant coffee. How is it made? Does it contain real coffee? Is it bad for you? The instant coffee that most people buy was initially ground, brewed, frozen and then dried. It IS real coffee, and apart from a slightly lower caffeine content, it offers up a similar collection of nutrients and vitamins.

So that's the boring stuff out of the way: instant coffee doesn't taste nice but is almost identical to ground coffee in terms of nutrition. I thought I'd delve into the history of instant coffee in search of some interesting facts, and it turns out there are quite a few.

1. INSTANT COFFEE WAS FIRST CONSUMED IN

Fig. 12. Steps 3 and 4 of the set of Instructions.

For all three scenarios accelerometer, orientation and Wi-Fi Network data was collected for each participant from the moment they pressed the Start button until the moment they pressed the Stop button. After the data is collected from all the participants, three datasets are initially created, one for each scenario and the BSSIDs in all datasets are transformed into numeric values from nominal. Furthermore, each dataset is manually imported into Weka, and it is *resampled* to create 2 smaller datasets for each initial dataset. One dataset of the newly created datasets has 70% of the data from the initial dataset and it is used for training the model, and the other dataset contains 30% of the initial dataset and it used for testing the trained model. It should be noted that the two newly created datasets contain unique data from each other.

The training dataset for the first scenario contains a total of 13568 instances for all six participants and the corresponding testing dataset contains 5816 instances. Moreover, the training dataset for the second scenario contains a total of 13194 instances for all six participants and the corresponding testing dataset contains 5655 instances. Lastly, for the last scenario the training dataset for the first scenario contains a total of 3710 instances for all six participants and the corresponding testing dataset contains 2474 instances.

The system's performance is evaluated using the Naïve Bayes Algorithm and the Multilayer Perceptron Algorithm, as the system has the best classification performance using these two algorithms, as mentioned in the Work Undertaken section. Moreover, combinations of the chosen sensors were evaluated to check how these affect the system's performance.

The classification percentage of each algorithm is represented using four metrics; accuracy, average precision, average recall and average F-measure. The accuracy (3) is the fraction of all the correct predictions over all the predictions made by the algorithm. Moreover precision (4), also known as the positive predictive value, is the number of the true positive values divided by the total number of true positive and false positive values. In other words, precision is the number of the positive predictions made by the algorithm that are actually true positives. Furthermore recall (5), also called sensitivity, is the true positive rate. This is the number of positive predictions that were predicted correctly. Lastly, the F-measure (6) is the harmonic mean of the precision and recall [38,39,40].

$$accuracy = \frac{TP + TN}{TP+FN+TP+TN} \tag{3}$$

$$precision = \frac{TP}{TP+FP} \tag{4}$$

$$recall = \frac{TP}{\mathrm{TP+FN}} \tag{5}$$

$$F - measure = 2\frac{precision * recall}{precision + recall} \tag{6}$$

For the first scenario (Sitting), when the system is using all the chosen sensors to verify the user's claimed identity the accuracy is 95.43% for the Naïve Bayes Algorithm and 97.83% for the Multilayer Perceptron Algorithm. For the second scenario (Desk), when the system is using all the chosen sensors to verify the user's claimed identity the accuracy is 95.79% for the Naïve Bayes Algorithm and 99.98% for the Multilayer Perceptron Algorithm. Lastly, for the third scenario (Walking), when the system is using all the chosen sensors to verify the user's claimed identity the accuracy is 87.15% for the Naïve Bayes Algorithm and 94.42% for the Multilayer Perceptron Algorithm.

Furthermore, combinations of the sensors were tested to examine how the system performs in those cases. It was discovered that when the system is using the collected orientation from the magnetometer sensor in combination with the collected BSSIDs only, it has a better accuracy compared to the accuracy when it uses only the data collected from the accelerometer sensor combined with the BSSIDs only, in all the defined scenarios except the third scenario. Lastly, as it can be seen from the results below, the system relies heavily on the collected BSSIDs to verify the users. This was also confirmed through Weka's Feature Selection tab which allows us to check which attributes contribute the most in successfully solving the problem at hand.

The following tables show the accuracy acquired from the model for the Naïve Bayes and Multilayer Perceptron Algorithms in each scenario.

Table 5. The System's performance for the first scenario using the Naive Bayes Algorithm.

	Accuracy	Average Precision	Average Recall	Average F-Measure
All sensors	95.43%	95.4%	95.4%	95.1%
Acceler + Orient	69.22%	69.1%	69.2%	67.7%
Acceler + Wi-Fi	89.51%	89.1%	89.5%	88.7%
Orient + Wi-Fi	96.12%	96.2%	96.1%	95.8%

Table 6. The System's performance for the first scenario using the Multilayer Perceptron Algorithm.

	Accuracy	Average Precision	Average Recall	Average F-Measure
All sensors	97.83%	97.9%	97.8%	95.7%
Acceler + Orient	82.17%	83.8%	82.2%	81.3%
Acceler + Wi-Fi	97.42%	97.7%	97.4%	97.4%
Orient + Wi-Fi	97.62%	97.8%	97.6%	97.5%

Table 7. The System's performance for the second scenario using the Naive Bayes Algorithm.

	Accuracy	Average Precision	Average Recall	Average F-Measure
All sensors	95.33% %	96.2%	95.8%	95.6%
Acceler + Orient	95.97%	96.3%	96.0%	95.8%
Acceler + Wi-Fi	93.94%	94.6%	93.9%	93.4%
Orient + Wi-Fi	95.97%	96.4%	96.0%	95.8%

Table 8. The System's performance for the second scenario using the Multilayer Perceptron Algorithm.

	Accuracy	Average Precision	Average Recall	Average F-Measure
All sensors	99.98%	100%	100%	100%
Acceler + Orient	99.69%	99.7%	99.7%	99.6%
Acceler + Wi-Fi	93.49%	93.6%	93.5%	93.5%
Orient + Wi-Fi	99.96%	100%	100%	100%

Table 9. The System's performance for the third scenario using the Naïve Bayes Algorithm.

	Accuracy	Average Precision	Average Recall	Average F-Measure
All sensors	87.15%	89.5%	87.1%	87.0%
Acceler + Orient	54.69%	49.4%	54.7%	49.2%
Acceler + Wi-Fi	88.23%	89.3%	88.2%	88.5%
Orient + Wi-Fi	88.68%	90.5%	88.7%	88.8%

Table 10. The System's performance for the third scenario using the Multilayer Perceptron Algorithm.

	Accuracy	Average Precision	Average Recall	Average F-Measure
All sensors	94.42%	95.6%	94.4%	94.5%
Acceler + Orient	61.28%	59.8%	61.3%	59.7%
Acceler + Wi-Fi	99.91%	99.9%	99.9%	99.9%
Orient + Wi-Fi	99.35%	99.4%	99.4%	9.44%

8 Evaluation

This section of the paper is dedicated on evaluating whether the proposed Aim and Objectives set out in *Section 2. Aims and Objectives* were achieved and give a clear evaluation of what has been done in order to achieve these objectives and the aim of the project.
The main aim of the project was to develop a continuous authentication system for Android smartphone devices, and a set of objectives were set to guide and aid the design, implementation and testing the system's performance.

The first objective was to research and assess existing continuous authentication systems. In section 3, various systems were examined and observations were made. We mainly focused on six systems; four of which were about authenticating the user using a smartphone device, one was about continuously verifying the presence of the logged-in user, and the other one was about continuously verifying the user from their gait using Wi-Fi signals. From these systems, we learnt that using multiple biometric characteristics to verify the user's identity improves the security of the system, but more data is needed to be collected and processed which can decrease the system's usability as it will take longer to authenticate the user. Moreover, we learnt that if the combined biometrics chosen have low correlation, then they improve the security performance. Lastly, we have an understanding of what to expect with regards to the accuracy of the proposed system.

The second objective was to identify which biometric characteristics could be used for this system to accurately verify the user. This objective was achieved by studying existing continuous authentication systems on smartphone devices, and understanding what each characteristic had to offer to the system. Moreover, we examined how biometric characteristics are also used in other authentication methods, and listed their advantages and disadvantages.

The third objective was to research sensors found on smartphone devices that can be used to collect the user's biometric information. In section 3.1, a table was given listing the most common sensors on Android devices along with their functionality, a description of what they measure and how they can be used in a continuous authentication system.

The fourth objective of this paper was to identify which modes the designed system will be used in and define what information will be collected about the user in each scenario. We decided that the system will be used and tested under three scenarios; for the first scenario the user is using the device while sitting, for the second scenario the user is using the device while it is flat on the desk and for the last scenario the user is walking and the phone is in his right-front pocket. For all the scenarios, the force applied to the device and the orientation of the device are measured, and the BSSIDs of the available wireless connections around the user are collected. We decided to use the device's accelerometer to measure the force applied to the device, to use the accelerometer and magnetometer sensors together to measure the orientation of the device, and lastly to use the Wi-Fi sensor to collect the information about the available wireless connections around the user.

The fifth and one of the most important objectives of this paper, is to design and implement a continuous authentication system on a smartphone device that uses the chosen sensors, in the defined modes to gather behavioural characteristics and information about the user's environment to verify the user's claimed identity. An application was developed which runs as a background service when the start button on its User Interface its clicked. This application utilises the defined sensors to continuously and non-intrusively collect data about how the user interacts with the device, and collect data about the user's environment. The system filters the data from the accelerometer sensor, then using the filtered data along with the data from the magnetometer sensor, it calculates the orientation of the device. Then the system writes all the data in a CSV file for further processing, in order to verify the user.

Furthermore, in section 7, the system's performance was evaluated and in result the sixth objective of this paper was achieved. Firstly, six people were given a set of instructions to perform four times using a smartphone device where the developed application was installed and was running on. At the end of the data collection stage, for each scenario one dataset was created that contained all the data collected from all the people. Then, BSSIDs in these files were manually translated into numeric values from nominal and they were imported into Weka for further processing. After a lot of experimentation with Weka and the provided machine learning algorithms, we decided to randomly divide each dataset into a training dataset that contained 70% of data and a test dataset that contained the other 30% for the specified scenario. For each scenario, the corresponding training and testing datasets were used along with two machine learning algorithms; Naïve Bayes and Multilayer Perceptron to build and test a model.

Even though the proposed system is not complete for use in real life applications, as it only works offline (the collected data is manually processed and imported into Weka for training and testing the models for each scenario), a server can be integrated with the system, to receive the data straight from the sensors, pre-process it, make a classification prediction and then return the result back to the system. The system, compares the returned value with on a threshold value and makes a decision on what applications and information the current user can access on the device without being asked for additional authentication.

9 Future Work

As mentioned in section 8 the system currently works offline and a potential improvement would be to make it work online (figure 13). This could be achieved by integrating a server which will be receiving the data from the chosen sensors immediately, process this data, use Weka to make classification predictions using one of the chosen machine learning algorithms and then send the result back to the application. Once the application receives the result from the sensor, it will compare it to a threshold value, and based on the result from this comparison decide what applications and information the current user can access without additional authentication. If the user requires access to either applications or information that he is not currently allowed to access, then the application can request from the user to further verify his identity with his fingerprint.

Moreover, as the system relies a lot on the collected BSSIDs to verify the users and the evaluation of the system was perform in just one room, another area of future work would be to evaluate the performance of the model when the user tries to use his smartphone device in a place where he does not usually go. Additionally, in the future the system's performance could be further evaluated by building a model based on a dataset for only one user and testing this model against datasets of other users.

Lastly, in the future the system could be implemented using the sliding windows technique instead of the chosen algorithms. The sliding window technique (*w*) converts sequential supervised learning problems into classical supervised learning problems. A window classifier, hw, maps an input of width (window) *w* into an individual output value *y* [34]. In other words, it uses previous data to predict the next value. I believe this will improve the performance of the system in general and especially when there are no available wireless connections and only data from the accelerometer and magnetometer sensor can be collected.

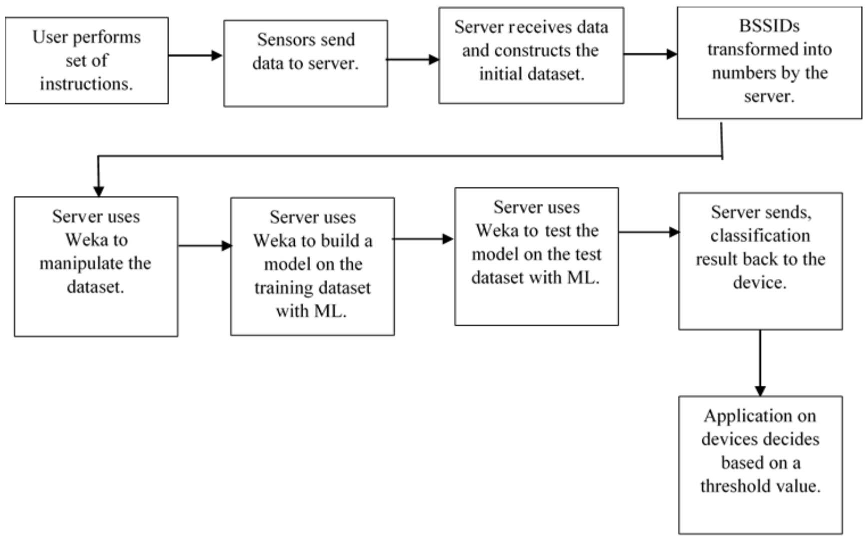

User performs set of instructions.
Sensors send data to server.
Server receives data and constructs the initial dataset.
BSSIDs transformed into numbers by the server.
Server uses Weka to manipulate the dataset.
Server uses Weka to build a model on the training dataset with ML.
Server uses Weka to test the model on the test dataset with ML.
Server sends, classification result back to the device.
Application on devices decides based on a threshold value.

Fig. 13. System Design based on Future work.

10 Conclusion

The main aim of this paper was achieved, as a continuous authentication system on Android smartphone devices was designed, developed, and evaluated. Moreover, existing continuous authentication systems on smartphone devices and other systems were studied in section 3.4 of this report, research into biometric characteristics and how they can be used in such systems was carried out throughout this project. Furthermore, in section 3.3 sensors found on Android devices were examined and selected. In section 4.1, the user's characteristics, what information about his environment and how they are going to be collected through the device were defined. In section 4.2 various machine learning algorithms were reviewed to gain a better understanding about Machine Learning, and lastly in section 4.3 an overview of the system's design was given. Moreover, section 5 describes all the work carried out for designing and implementing the system. This section also includes experiments performed with the data collected from the accelerometer sensor and experiments performed with various machine learning algorithms. Section 6, gives a description of how the system was implemented using Android and Java. In section 7 we present how the system performance was evaluated and finally in section 8, an evaluation of the aim and objectives defined was carried out, to investigate what has been done and achieved.

Acknowledgements. Firstly, I would like to thank my supervisors Dr Martin Emms and Dr Charles Morriset for their support and advice throughout this project. Their continuous guidance helped me complete this project successfully. I would also like to thank my friends for supporting me through another year of studying. Lastly, I would like to thank all the people for participating in the testing phase of the application, without them this project would not have been completed.

11 References

1. L. O'Gorman, "Comparing passwords, tokens, and biometrics for user authentication," in Proceedings of the IEEE, vol. 91, no. 12, pp. 2021-2040, Dec 2003.
2. E. Flior and K. Kowalski, "Continuous Biometric User Authentication in Online Examinations," 2010 Seventh International Conference on Information Technology: New Generations, Las Vegas, NV, 2010, pp. 488-492.
3. M. Barasz, B. Boros, P. Ligeti, K. L6ja, and D. Nagy. Passive attack against the M2AP mutual authentication protocol for RFID tags. The First International EURASIP Workshop on RFID Technology, 2007.
4. E. Taqieddin and J. Sarangapani, "Vulnerability analysis of two ultra-lightweight RFID authentication protocols: RAPP and gossamer," 2012 International Conference for Internet Technology and Secured Transactions, London, 2012, pp. 80-86
5. Hern, A. (2017). Hacker fakes German minister's fingerprints using photos of her hands. [online] the Guardian. Available at: http://www.theguardian.com/technology/2014/dec/30/hacker-fakes-german-ministers-fingerprints-using-photos-of-her-hands [Accessed 24 Aug. 2017].
6. No idea
7. A. Conklin, G. Dietrich and D. Walz, "Password-based authentication: a system perspective," 37th Annual Hawaii International Conference on System Sciences, 2004. Proceedings of the, 2004, pp. 10 pp.-.
8. NuData Security. (2017). 4 Big Problems with Knowledge Based Authentication. [online] Available at: https://nudatasecurity.com/blog/risk-based-authentication/4-big-problems-with-knowledge-based-authentication/
9. Chao Shen, He Zhang, Zhenyu Yang and Xiaohong Guan, "Modeling multimodal bio-metric modalities for continuous user authentication," 2016 IEEE International Confer-ence on Systems, Man, and Cybernetics (SMC), Budapest, 2016, pp. 001894-001899.
10. Hogben, G. (2010). ENISA Briefing: Behavioural Biometrics.
11. N. K. Ratha, J. H. Connell and R. M. Bolle, "Enhancing security and privacy in biometrics-based authentication systems," in IBM Systems Journal, vol. 40, no. 3, pp. 614-634, 2001.
12. S. Pankanti, R. M. Bolle and A. Jain, "Biometrics: The future of identification [Guest Eeditors' Introduction]," in Computer, vol. 33, no. 2, pp. 46-49, Feb. 2000.
13. G. Peng; G. Zhou; D. T. Nguyen; X. Qi; Q. Yang; S. Wang, "Continuous Authentication With Touch Behavioral Biometrics and Voice on Wearable Glasses," in IEEE Transactions on Human-Machine Systems, vol.PP, no.99, pp.1-13
14. DeZyre. (2017). Top 10 Machine Learning Algorithms. [online] Available at: https://www.dezyre.com/article/top-10-machine-learning-algorithms/202 [Accessed 24 Aug. 2017].
15. Brownlee, J. (2017). How to Use Machine Learning Algorithms in Weka - Machine Learning Mastery. [online] Machine Learning Mastery. Available at: http://machinelearningmastery.com/use-machine-learning-algorithms-weka/ [Accessed 24 Aug. 2017].

16. Scikit-learn.org. (2017). 1.17. Neural network models (supervised) — scikit-learn 0.19.0 documentation. [online] Available at: http://scikit-learn.org/stable/modules/neural_networks_supervised.html [Accessed 24 Aug. 2017].
17. Scikit-learn.org. (2017). 1.9. Naive Bayes — scikit-learn 0.19.0 documentation. [online] Available at: http://scikit-learn.org/stable/modules/naive_bayes.html [Accessed 24 Aug. 2017].
18. Tfbarker.wordpress.com. (2017). random forests | Data mining. [online] Available at: https://tfbarker.wordpress.com/tag/random-forests/ [Accessed 24 Aug. 2017].
19. Docs.opencv.org. (2017). Introduction to Support Vector Machines — OpenCV 2.4.13.3 documentation. [online] Available at: http://docs.opencv.org/2.4/doc/tutorials/ml/introduction_to_svm/introduction_to_svm.html [Accessed 24 Aug. 2017].
20. Scikit-learn.org. (2017). 1.4. Support Vector Machines — scikit-learn 0.19.0 documentation. [online] Available at: http://scikit-learn.org/stable/modules/svm.html [Accessed 24 Aug. 2017].
21. Sensors Overview | Android Developers. [online] Developer.android.com. Available at: https://developer.android.com/guide/topics/sensors/sensors_overview.html [Accessed 24 Aug. 2017].
22. Chao Shen, He Zhang, Zhenyu Yang and Xiaohong Guan, "Modeling multimodal bio-metric modalities for continuous user authentication," 2016 IEEE International Confer-ence on Systems, Man, and Cybernetics (SMC), Budapest, 2016, pp. 001894-001899.
23. M. Shahzad and M. P. Singh, "Continuous Authentication and Authorization for the Internet of Things," in IEEE Internet Computing, vol. 21, no. 2, pp. 86-90, Mar.-Apr. 2017.
24. D. Gafurov, E. Snekkenes and P. Bours, "Gait Authentication and Identification Using Wearable Accelerometer Sensor," 2007 IEEE Workshop on Automatic Identification Advanced Technologies, Alghero, 2007, pp. 220-225.
25. T. Feng, X. Zhao, N. DeSalvo, Z. Gao, X. Wang and W. Shi, "Security after login: Iden-tity change detection on smartphones using sensor fusion," 2015 IEEE International Sym-posium on Technologies for Homeland Security (HST), Waltham, MA, 2015, pp. 1-6.
26. J. S. Wu, W. C. Lin, C. T. Lin and T. E. Wei, "Smartphone continuous authentication based on keystroke and gesture profiling," 2015 International Carnahan Conference on Security Technology (ICCST), Taipei, 2015, pp. 191-197.
27. R. Kumar, V. V. Phoha and A. Serwadda, "Continuous authentication of smartphone users by fusing typing, swiping, and phone movement patterns," 2016 IEEE 8th Inter-national Conference on Biometrics Theory, Applications and Systems (BTAS), Niagara Falls, NY, 2016, pp. 1-8.
28. W. H. Lee and R. B. Lee, "Multi-sensor authentication to improve smartphone security," 2015 International Conference on Information Systems Security and Privacy (ICISSP), Angers, 2015, pp. 1-11.
29. F. T. Garcia, K. Krombholz, R. Mayer and E. Weippl, "Hand Dynamics for Behavioral User Authentication," 2016 11th International Conference on Availability, Reliability and Security (ARES), Salzburg, 2016, pp. 389-398.
30. Cs.waikato.ac.nz. (2017). Attribute-Relation File Format (ARFF). [online] Available at: http://www.cs.waikato.ac.nz/ml/weka/arff.html [Accessed 24 Aug. 2017].
31. Gsmarena.com. (2017). LG Nexus 5X - Full phone specifications. [online] Available at: http://www.gsmarena.com/lg_nexus_5x-7556.php [Accessed 24 Aug. 2017].
32. Developer Android. (2017). Download Android Studio and SDK Tools | Android Studio. [online] Available at: https://developer.android.com/studio/index.html [Accessed 24 Aug. 2017].

33. Cs.waikato.ac.nz. (2017). Weka 3 - Data Mining with Open Source Machine Learning Software in Java. [online] Available at: http://www.cs.waikato.ac.nz/ml/weka/ [Accessed 24 Aug. 2017].
34. (2017). Low-Pass Filter: The Basics. [online] Kircherelectronics.com. Available at: http://www.kircherelectronics.com/blog/index.php/11-android/sensors/8-low-pass-filter-the-basics [Accessed 24 Aug. 2017].
35. Developer Android. (2017). Requesting Permissions at Run Time | Android Developers. [online] Developer.android.com. Available at: https://developer.android.com/training/permissions/requesting.html [Accessed 24 Aug. 2017].
36. Developer Android. (2017). Services | Android Developers. [online] Available at: https://developer.android.com/guide/components/services.html#Basics [Accessed 24 Aug. 2017].
37. Developer Android. (2017). Sensors Overview | Android Developers. [online] Developer.android.com. Available at: https://developer.android.com/guide/topics/sensors/sensors_overview.html [Accessed 24 Aug. 2017].
38. Brownlee, J. (2017). Classification Accuracy is Not Enough: More Performance Measures You Can Use - Machine Learning Mastery. [online] Machine Learning Mastery. Available at: https://machinelearningmastery.com/classification-accuracy-is-not-enough-more-performance-measures-you-can-use/ [Accessed 24 Aug. 2017].
39. Scikit-learn.org. (2017). 3.3. Model evaluation: quantifying the quality of predictions — scikit-learn 0.19.0 documentation. [online] Available at: http://scikit-learn.org/stable/modules/model_evaluation.html [Accessed 24 Aug. 2017].
40. Scikit-learn.org. (2017). 3.3. Model evaluation: quantifying the quality of predictions — scikit-learn 0.19.0 documentation. [online] Available at: http://scikit-learn.org/stable/modules/model_evaluation.html [Accessed 24 Aug. 2017].

12 Appendix A: Implementation Methods

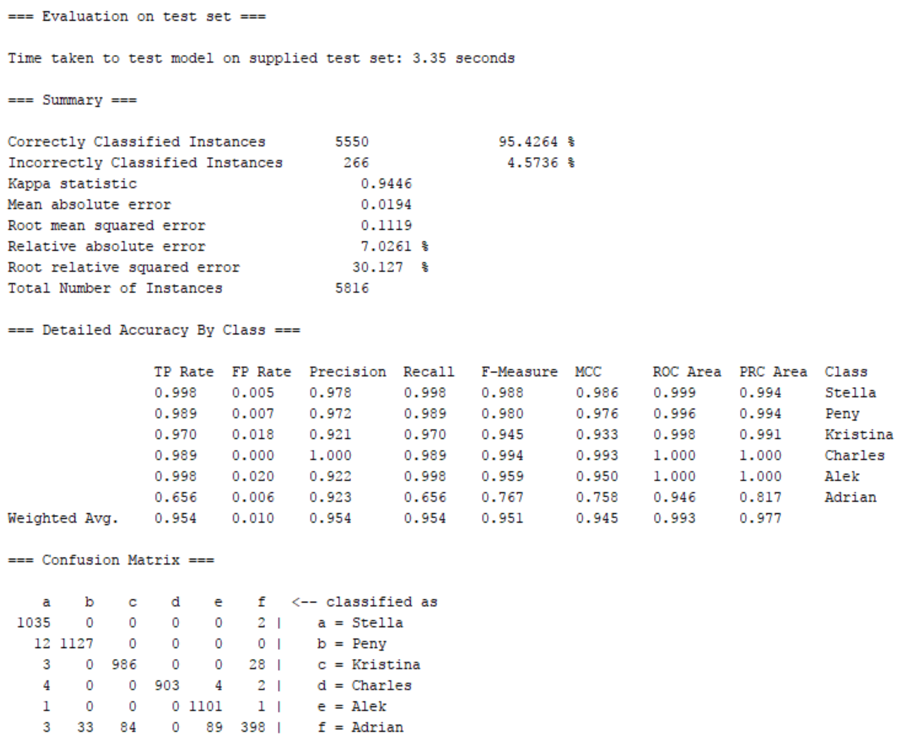
A1. The permissionGrant method

```
public void permissionGrant() {
    if (ContextCompat.checkSelfPermission(MainActivity.this,
Manifest.permission.ACCESS_FINE_LOCATION) +
        ContextCompat.checkSelfPermission(MainActiv-
ity.this,Manifest.permission.ACCESS_COARSE_LOCATION) +
            ContextCompat.checkSelfPermission(MainActiv-
ity.this,Manifest.permission.WRITE_EXTERNAL_STORAGE) +
            ContextCompat.checkSelfPermission(MainActiv-
ity.this,Manifest.permission.ACCESS_WIFI_STATE)
            != PackageManager.PERMISSION_GRANTED) {
        if (ActivityCompat.shouldShowRequestPermissionRa-
tionale(MainActivity.this,
                Manifest.permission.ACCESS_FINE_LOCATION)|| Ac-
tivityCompat.shouldShowRequestPermissionRationale(MainActiv-
ity.this,
                Manifest.permission.ACCESS_COARSE_LOCATION) ||
ActivityCompat.shouldShowRequestPermissionRationale(MainActiv-
ity.this,
                Manifest.permission.ACCESS_WIFI_STATE) || Ac-
tivityCompat.shouldShowRequestPermissionRationale(MainActiv-
ity.this,
                Manifest.permission.WRITE_EXTERNAL_STORAGE) ) {

        } else {
            ActivityCompat.requestPermissions(MainActivity.this,
                    new String[]{Manifest.permis-
sion.ACCESS_FINE_LOCATION,Manifest.permission.ACCESS_COARSE_LOCA
TION,
                            Manifest.permis-
sion.ACCESS_WIFI_STATE,Manifest.permission.WRITE_EXTERNAL_STORAG
E},MY_PERMISSIONS_REQUEST_LOCATION);
        }
    }
}
```

A2. The onRequestPermissionsResult method

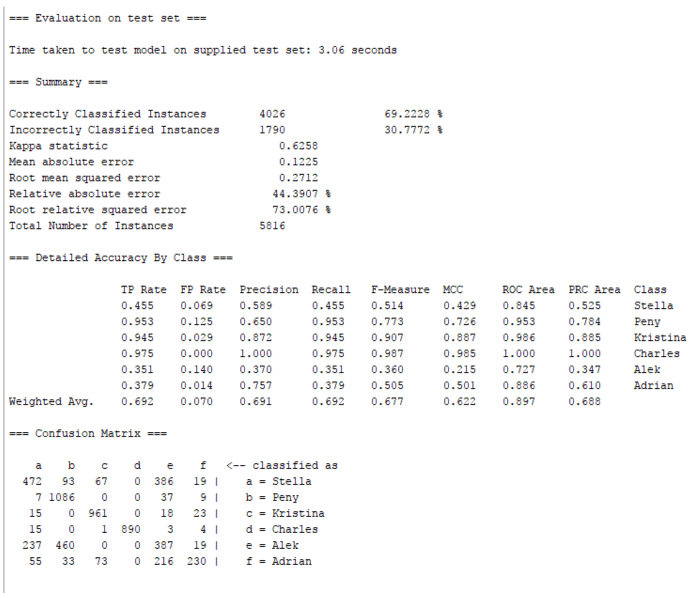
```
@Override
public void onRequestPermissionsResult(int requestCode,
                                       String permissions[],
int[] grantResults) {
    switch (requestCode) {
        case MY_PERMISSIONS_REQUEST_LOCATION: {
            // If request is cancelled, the result arrays are
empty.
            if (grantResults.length > 0
                    && grantResults[0] ==
        PackageManager.PERMISSION_GRANTED) {
                // permission was granted, yay! Do the
                // location-related task you need to do.
                if (ContextCompat.checkSelfPermission(this,
                        Manifest.permission.
ACCESS_FINE_LOCATION)
                        == PackageManager.PERMISSION_GRANTED) {
                      }
            } else {
                // permission denied, boo!
            }
            return;
        }
    }
}
```

13 Appendix B: Weka Results



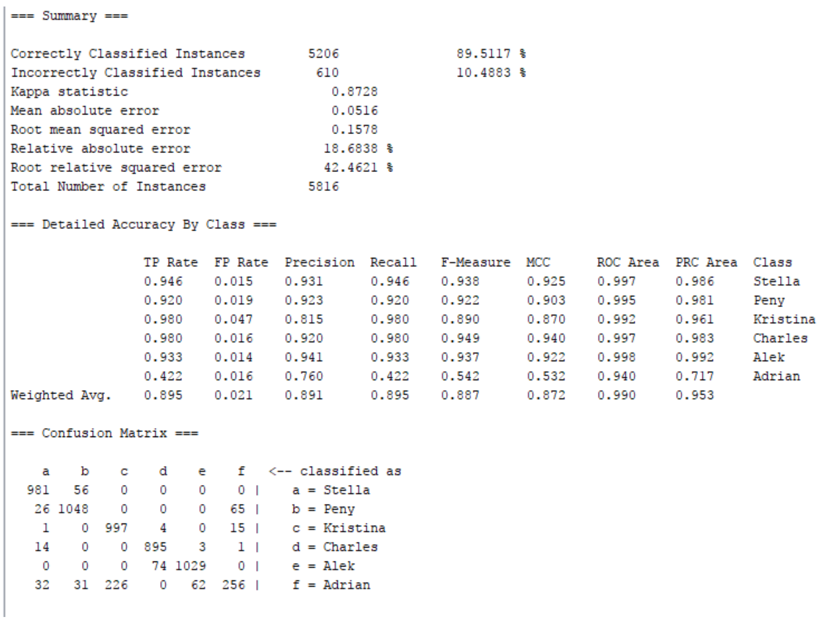
```
=== Evaluation on test set ===

Time taken to test model on supplied test set: 3.35 seconds

=== Summary ===

Correctly Classified Instances        5550               95.4264 %
Incorrectly Classified Instances       266                4.5736 %
Kappa statistic                          0.9446
Mean absolute error                      0.0194
Root mean squared error                  0.1119
Relative absolute error                  7.0261 %
Root relative squared error             30.127  %
Total Number of Instances             5816

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0.998    0.005    0.978      0.998    0.988      0.986    0.999     0.994     Stella
                 0.989    0.007    0.972      0.989    0.980      0.976    0.996     0.994     Peny
                 0.970    0.018    0.921      0.970    0.945      0.933    0.998     0.991     Kristina
                 0.989    0.000    1.000      0.989    0.994      0.993    1.000     1.000     Charles
                 0.998    0.020    0.922      0.998    0.959      0.950    1.000     1.000     Alek
                 0.656    0.006    0.923      0.656    0.767      0.758    0.946     0.817     Adrian
Weighted Avg.    0.954    0.010    0.954      0.954    0.951      0.945    0.993     0.977

=== Confusion Matrix ===

    a    b    c    d    e    f   <-- classified as
 1035    0    0    0    0    2 |    a = Stella
   12 1127    0    0    0    0 |    b = Peny
    3    0  986    0    0   28 |    c = Kristina
    4    0    0  903    4    2 |    d = Charles
    1    0    0    0 1101    1 |    e = Alek
    3   33   84    0   89  398 |    f = Adrian
```

Fig. 14. Sitting Scenario - Naïve Bayes. All Sensors.



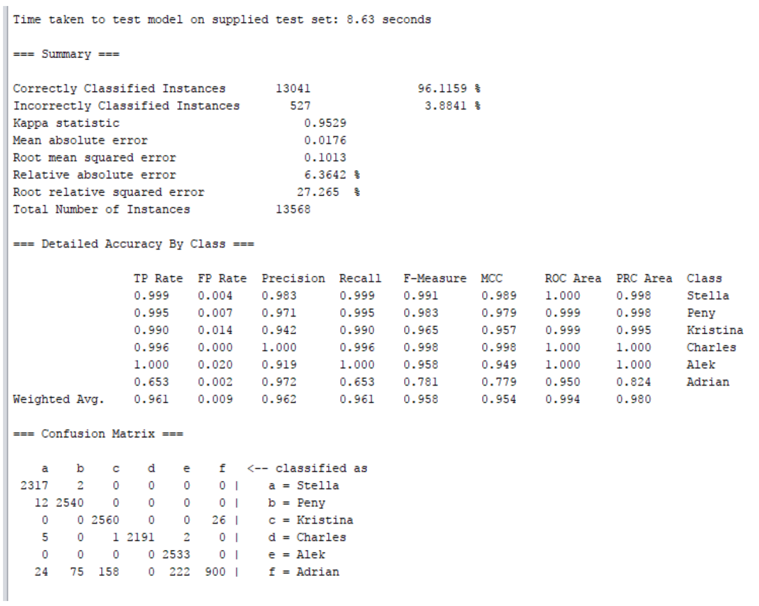
```
=== Evaluation on test set ===

Time taken to test model on supplied test set: 3.06 seconds

=== Summary ===

Correctly Classified Instances        4026               69.2228 %
Incorrectly Classified Instances      1790               30.7772 %
Kappa statistic                          0.6258
Mean absolute error                      0.1225
Root mean squared error                  0.2712
Relative absolute error                 44.3907 %
Root relative squared error             73.0076 %
Total Number of Instances             5816

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0.455    0.069    0.589      0.455    0.514      0.429    0.845     0.525     Stella
                 0.953    0.125    0.650      0.953    0.773      0.726    0.953     0.784     Peny
                 0.945    0.029    0.872      0.945    0.907      0.887    0.986     0.885     Kristina
                 0.975    0.000    1.000      0.975    0.987      0.985    1.000     1.000     Charles
                 0.351    0.140    0.370      0.351    0.360      0.215    0.727     0.347     Alek
                 0.379    0.014    0.757      0.379    0.505      0.501    0.886     0.610     Adrian
Weighted Avg.    0.692    0.070    0.691      0.692    0.677      0.622    0.897     0.688

=== Confusion Matrix ===

    a    b    c    d    e    f   <-- classified as
  472   93   67    0  386   19 |    a = Stella
    7 1086    0    0   37    9 |    b = Peny
   15    0  961    0   18   23 |    c = Kristina
   15    0    1  890    3    4 |    d = Charles
  237  460    0    0  387   19 |    e = Alek
   55   33   73    0  216  230 |    f = Adrian
```

Fig. 15. Sitting Scenario - Naïve Bayes. Accelerometer and Magnetometer Sensors.



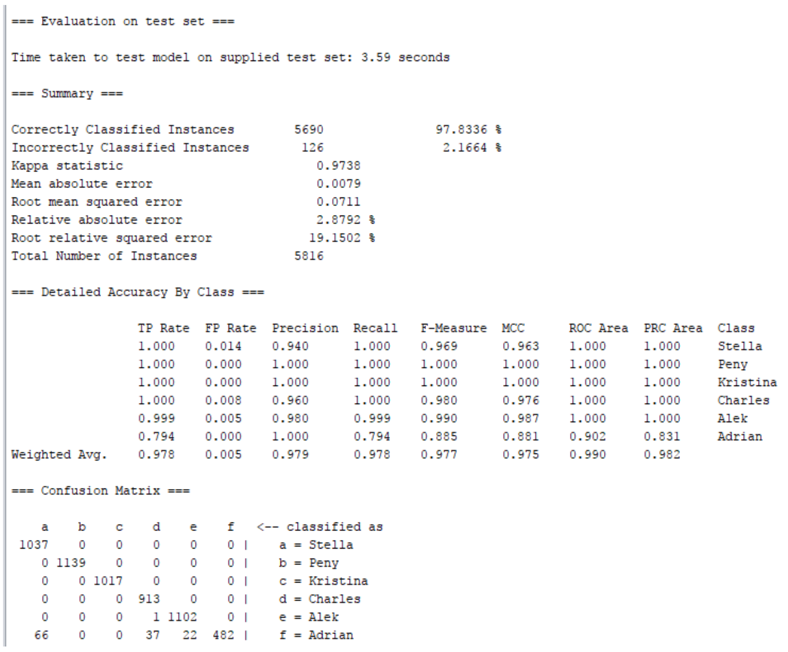
```
=== Summary ===

Correctly Classified Instances        5206               89.5117 %
Incorrectly Classified Instances       610               10.4883 %
Kappa statistic                          0.8728
Mean absolute error                      0.0516
Root mean squared error                  0.1578
Relative absolute error                 18.6838 %
Root relative squared error             42.4621 %
Total Number of Instances             5816

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0.946    0.015    0.931      0.946    0.938      0.925    0.997     0.986     Stella
                 0.920    0.019    0.923      0.920    0.922      0.903    0.995     0.981     Peny
                 0.980    0.047    0.815      0.980    0.890      0.870    0.992     0.961     Kristina
                 0.980    0.016    0.920      0.980    0.949      0.940    0.997     0.983     Charles
                 0.933    0.014    0.941      0.933    0.937      0.922    0.998     0.992     Alek
                 0.422    0.016    0.760      0.422    0.542      0.532    0.940     0.717     Adrian
Weighted Avg.    0.895    0.021    0.891      0.895    0.887      0.872    0.990     0.953

=== Confusion Matrix ===

    a    b    c    d    e    f   <-- classified as
  981   56    0    0    0    0 |    a = Stella
   26 1048    0    0    0   65 |    b = Peny
    1    0  997    4    0   15 |    c = Kristina
   14    0    0  895    3    1 |    d = Charles
    0    0    0   74 1029    0 |    e = Alek
   32   31  226    0   62  256 |    f = Adrian
```

Fig. 16. Sitting Scenario - Naïve Bayes. Accelerometer and Wi-Fi Sensors.



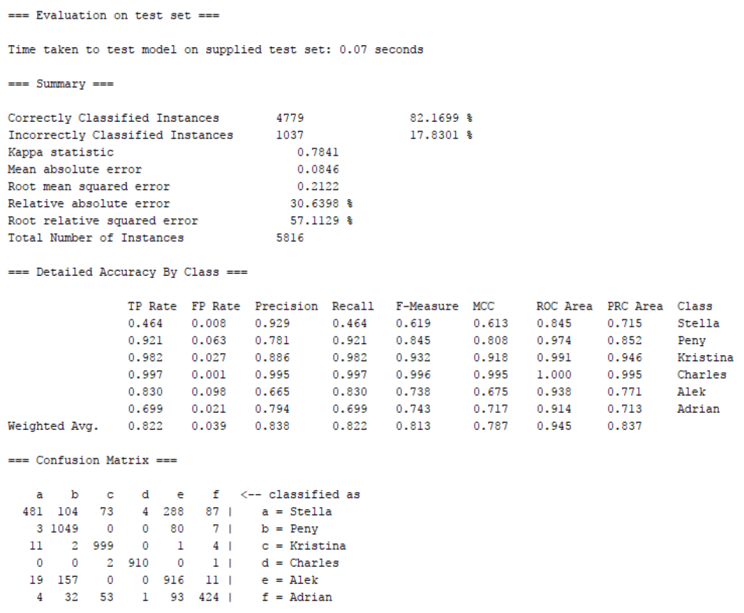
```
Time taken to test model on supplied test set: 8.63 seconds

=== Summary ===

Correctly Classified Instances       13041               96.1159 %
Incorrectly Classified Instances       527                3.8841 %
Kappa statistic                          0.9529
Mean absolute error                      0.0176
Root mean squared error                  0.1013
Relative absolute error                  6.3642 %
Root relative squared error             27.265  %
Total Number of Instances            13568

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0.999    0.004    0.983      0.999    0.991      0.989    1.000     0.998     Stella
                 0.995    0.007    0.971      0.995    0.983      0.979    0.999     0.998     Peny
                 0.990    0.014    0.942      0.990    0.965      0.957    0.999     0.995     Kristina
                 0.996    0.000    1.000      0.996    0.998      0.998    1.000     1.000     Charles
                 1.000    0.020    0.919      1.000    0.958      0.949    1.000     1.000     Alek
                 0.653    0.002    0.972      0.653    0.781      0.779    0.950     0.824     Adrian
Weighted Avg.    0.961    0.009    0.962      0.961    0.958      0.954    0.994     0.980

=== Confusion Matrix ===

    a    b    c    d    e    f   <-- classified as
 2317    2    0    0    0    0 |    a = Stella
   12 2540    0    0    0    0 |    b = Peny
    0    0 2560    0    0   26 |    c = Kristina
    5    0    1 2191    2    0 |    d = Charles
    0    0    0    0 2533    0 |    e = Alek
   24   75  158    0  222  900 |    f = Adrian
```

Fig. 17. Sitting Scenario - Naïve Bayes. Orientation and Wi-Fi Sensors.



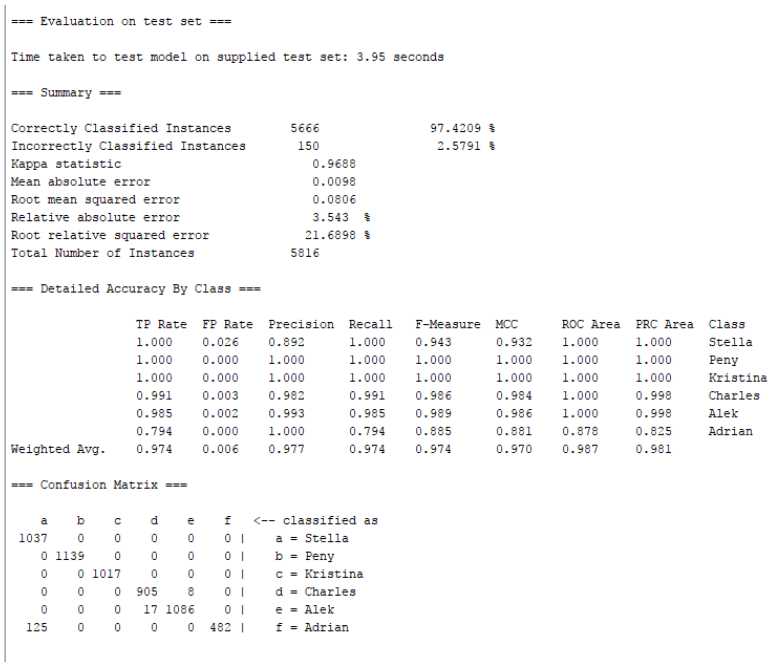
```
=== Evaluation on test set ===

Time taken to test model on supplied test set: 3.59 seconds

=== Summary ===

Correctly Classified Instances        5690               97.8336 %
Incorrectly Classified Instances       126                2.1664 %
Kappa statistic                          0.9738
Mean absolute error                      0.0079
Root mean squared error                  0.0711
Relative absolute error                  2.8792 %
Root relative squared error             19.1502 %
Total Number of Instances             5816

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 1.000    0.014    0.940      1.000    0.969      0.963    1.000     1.000     Stella
                 1.000    0.000    1.000      1.000    1.000      1.000    1.000     1.000     Peny
                 1.000    0.000    1.000      1.000    1.000      1.000    1.000     1.000     Kristina
                 1.000    0.008    0.960      1.000    0.980      0.976    1.000     1.000     Charles
                 0.999    0.005    0.980      0.999    0.990      0.987    1.000     1.000     Alek
                 0.794    0.000    1.000      0.794    0.885      0.881    0.902     0.831     Adrian
Weighted Avg.    0.978    0.005    0.979      0.978    0.977      0.975    0.990     0.982

=== Confusion Matrix ===

    a    b    c    d    e    f   <-- classified as
 1037    0    0    0    0    0 |    a = Stella
    0 1139    0    0    0    0 |    b = Peny
    0    0 1017    0    0    0 |    c = Kristina
    0    0    0  913    0    0 |    d = Charles
    0    0    0    1 1102    0 |    e = Alek
   66    0    0   37   22  482 |    f = Adrian
```

Fig. 18. Sitting Scenario – Multilayer Perceptron. All Sensors.



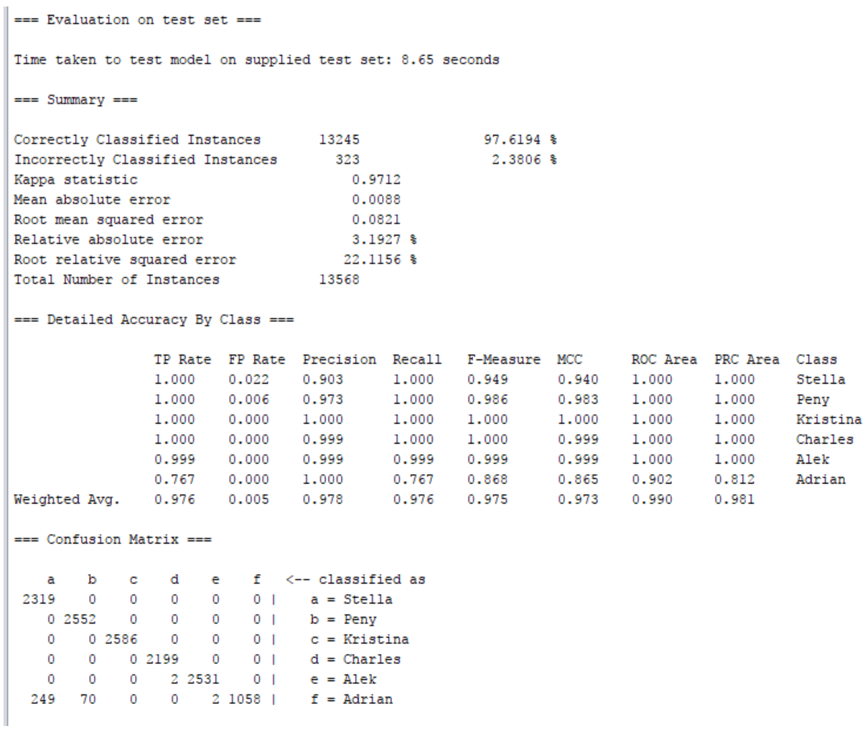
```
=== Evaluation on test set ===

Time taken to test model on supplied test set: 0.07 seconds

=== Summary ===

Correctly Classified Instances        4779               82.1699 %
Incorrectly Classified Instances      1037               17.8301 %
Kappa statistic                          0.7841
Mean absolute error                      0.0846
Root mean squared error                  0.2122
Relative absolute error                 30.6398 %
Root relative squared error             57.1129 %
Total Number of Instances             5816

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0.464    0.008    0.929      0.464    0.619      0.613    0.845     0.715     Stella
                 0.921    0.063    0.781      0.921    0.845      0.808    0.974     0.852     Peny
                 0.982    0.027    0.886      0.982    0.932      0.918    0.991     0.946     Kristina
                 0.997    0.001    0.995      0.997    0.996      0.995    1.000     0.995     Charles
                 0.830    0.098    0.665      0.830    0.738      0.675    0.938     0.771     Alek
                 0.699    0.021    0.794      0.699    0.743      0.717    0.914     0.713     Adrian
Weighted Avg.    0.822    0.039    0.838      0.822    0.813      0.787    0.945     0.837

=== Confusion Matrix ===

    a    b    c    d    e    f   <-- classified as
  481  104   73    4  288   87 |    a = Stella
    3 1049    0    0   80    7 |    b = Peny
   11    2  999    0    1    4 |    c = Kristina
    0    0    2  910    0    1 |    d = Charles
   19  157    0    0  916   11 |    e = Alek
    4   32   53    1   93  424 |    f = Adrian
```

Fig. 19. Sitting Scenario – Multilayer Perceptron. Accelerometer and Orientation Sensors.



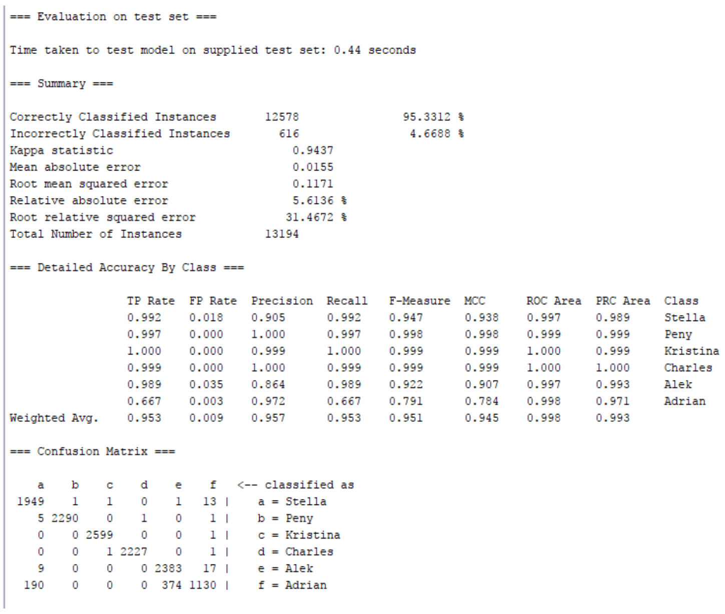
```
=== Evaluation on test set ===

Time taken to test model on supplied test set: 3.95 seconds

=== Summary ===

Correctly Classified Instances        5666               97.4209 %
Incorrectly Classified Instances       150                2.5791 %
Kappa statistic                          0.9688
Mean absolute error                      0.0098
Root mean squared error                  0.0806
Relative absolute error                  3.543  %
Root relative squared error             21.6898 %
Total Number of Instances             5816

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 1.000    0.026    0.892      1.000    0.943      0.932    1.000     1.000     Stella
                 1.000    0.000    1.000      1.000    1.000      1.000    1.000     1.000     Peny
                 1.000    0.000    1.000      1.000    1.000      1.000    1.000     1.000     Kristina
                 0.991    0.003    0.982      0.991    0.986      0.984    1.000     0.998     Charles
                 0.985    0.002    0.993      0.985    0.989      0.986    1.000     0.998     Alek
                 0.794    0.000    1.000      0.794    0.885      0.881    0.878     0.825     Adrian
Weighted Avg.    0.974    0.006    0.977      0.974    0.974      0.970    0.987     0.981

=== Confusion Matrix ===

    a    b    c    d    e    f   <-- classified as
 1037    0    0    0    0    0 |    a = Stella
    0 1139    0    0    0    0 |    b = Peny
    0    0 1017    0    0    0 |    c = Kristina
    0    0    0  905    8    0 |    d = Charles
    0    0    0   17 1086    0 |    e = Alek
  125    0    0    0    0  482 |    f = Adrian
```

Fig. 20. Sitting Scenario – Multilayer Perceptron. Accelerometer and Wi-Fi.



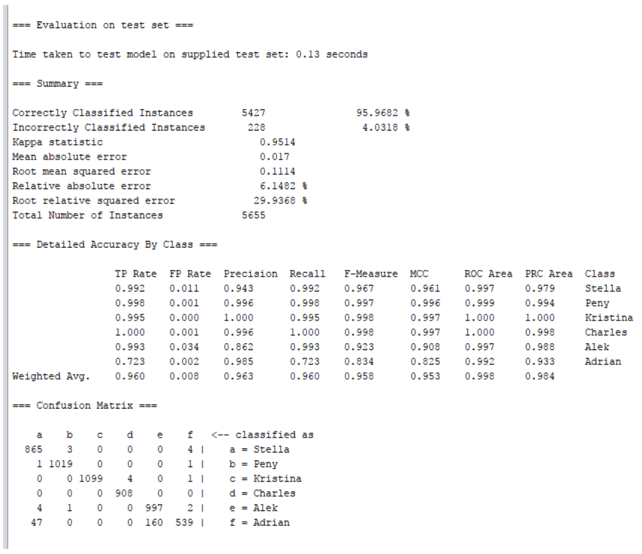
```
=== Evaluation on test set ===

Time taken to test model on supplied test set: 8.65 seconds

=== Summary ===

Correctly Classified Instances       13245               97.6194 %
Incorrectly Classified Instances       323                2.3806 %
Kappa statistic                          0.9712
Mean absolute error                      0.0088
Root mean squared error                  0.0821
Relative absolute error                  3.1927 %
Root relative squared error             22.1156 %
Total Number of Instances            13568

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 1.000    0.022    0.903      1.000    0.949      0.940    1.000     1.000     Stella
                 1.000    0.006    0.973      1.000    0.986      0.983    1.000     1.000     Peny
                 1.000    0.000    1.000      1.000    1.000      1.000    1.000     1.000     Kristina
                 1.000    0.000    0.999      1.000    1.000      0.999    1.000     1.000     Charles
                 0.999    0.000    0.999      0.999    0.999      0.999    1.000     1.000     Alek
                 0.767    0.000    1.000      0.767    0.868      0.865    0.902     0.812     Adrian
Weighted Avg.    0.976    0.005    0.978      0.976    0.975      0.973    0.990     0.981

=== Confusion Matrix ===

    a    b    c    d    e    f   <-- classified as
 2319    0    0    0    0    0 |    a = Stella
    0 2552    0    0    0    0 |    b = Peny
    0    0 2586    0    0    0 |    c = Kristina
    0    0    0 2199    0    0 |    d = Charles
    0    0    0    2 2531    0 |    e = Alek
  249   70    0    0    2 1058 |    f = Adrian
```

Fig. 21. Sitting Scenario – Multilayer Perceptron. Orientation and Wi-Fi.



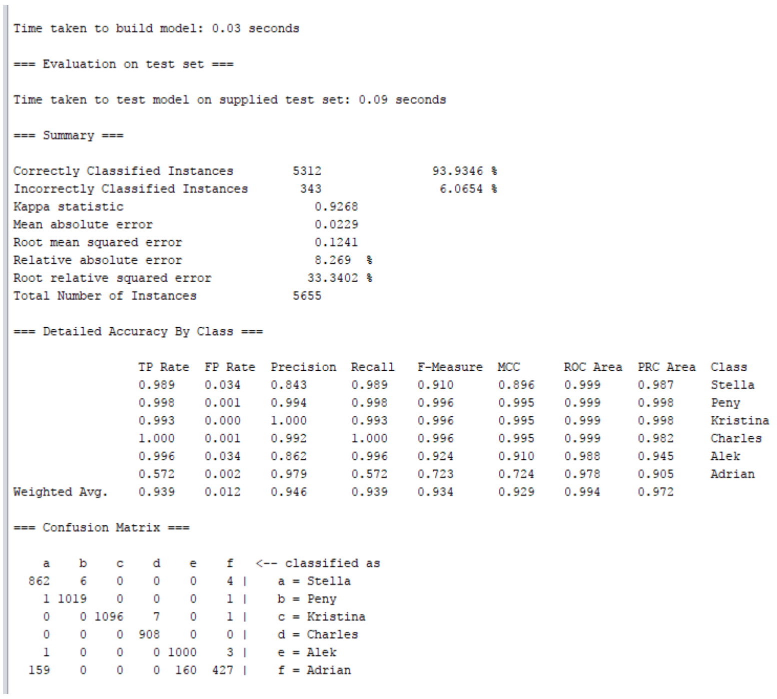
```
=== Evaluation on test set ===

Time taken to test model on supplied test set: 0.44 seconds

=== Summary ===

Correctly Classified Instances       12578               95.3312 %
Incorrectly Classified Instances       616                4.6688 %
Kappa statistic                          0.9437
Mean absolute error                      0.0155
Root mean squared error                  0.1171
Relative absolute error                  5.6136 %
Root relative squared error             31.4672 %
Total Number of Instances            13194

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0.992    0.018    0.905      0.992    0.947      0.938    0.997     0.989     Stella
                 0.997    0.000    1.000      0.997    0.998      0.998    0.999     0.999     Peny
                 1.000    0.000    0.999      1.000    0.999      0.999    1.000     0.999     Kristina
                 0.999    0.000    1.000      0.999    0.999      0.999    1.000     1.000     Charles
                 0.989    0.035    0.864      0.989    0.922      0.907    0.997     0.993     Alek
                 0.667    0.003    0.972      0.667    0.791      0.784    0.998     0.971     Adrian
Weighted Avg.    0.953    0.009    0.957      0.953    0.951      0.945    0.998     0.993

=== Confusion Matrix ===

    a    b    c    d    e    f   <-- classified as
 1949    1    1    0    1   13 |    a = Stella
    5 2290    0    1    0    1 |    b = Peny
    0    0 2599    0    0    1 |    c = Kristina
    0    0    1 2227    0    1 |    d = Charles
    9    0    0    0 2383   17 |    e = Alek
  190    0    0    0  374 1130 |    f = Adrian
```

Fig. 22. Desk Scenario - Naïve Bayes. All Sensors



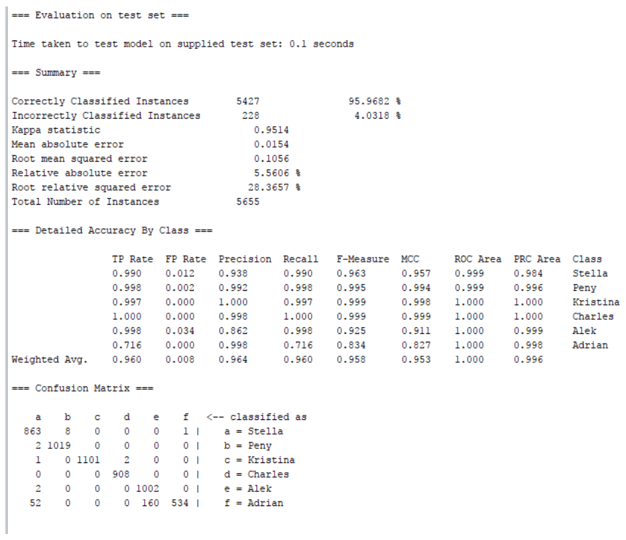
```
=== Evaluation on test set ===

Time taken to test model on supplied test set: 0.13 seconds

=== Summary ===

Correctly Classified Instances        5427               95.9682 %
Incorrectly Classified Instances       228                4.0318 %
Kappa statistic                          0.9514
Mean absolute error                      0.017
Root mean squared error                  0.1114
Relative absolute error                  6.1482 %
Root relative squared error             29.9368 %
Total Number of Instances             5655

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0.992    0.011    0.943      0.992    0.967      0.961    0.997     0.979     Stella
                 0.998    0.001    0.996      0.998    0.997      0.996    0.999     0.994     Peny
                 0.995    0.000    1.000      0.995    0.998      0.997    1.000     1.000     Kristina
                 1.000    0.001    0.996      1.000    0.998      0.997    1.000     0.998     Charles
                 0.993    0.034    0.862      0.993    0.923      0.908    0.997     0.988     Alek
                 0.723    0.002    0.985      0.723    0.834      0.825    0.992     0.933     Adrian
Weighted Avg.    0.960    0.008    0.963      0.960    0.958      0.953    0.998     0.984

=== Confusion Matrix ===

    a    b    c    d    e    f   <-- classified as
  865    3    0    0    0    4 |    a = Stella
    1 1019    0    0    0    1 |    b = Peny
    0    0 1099    4    0    1 |    c = Kristina
    0    0    0  908    0    0 |    d = Charles
    4    1    0    0  997    2 |    e = Alek
   47    0    0    0  160  539 |    f = Adrian
```

Fig. 23. Desk Scenario - Naïve Bayes. Accelerometer and Magnetometer Sensors.



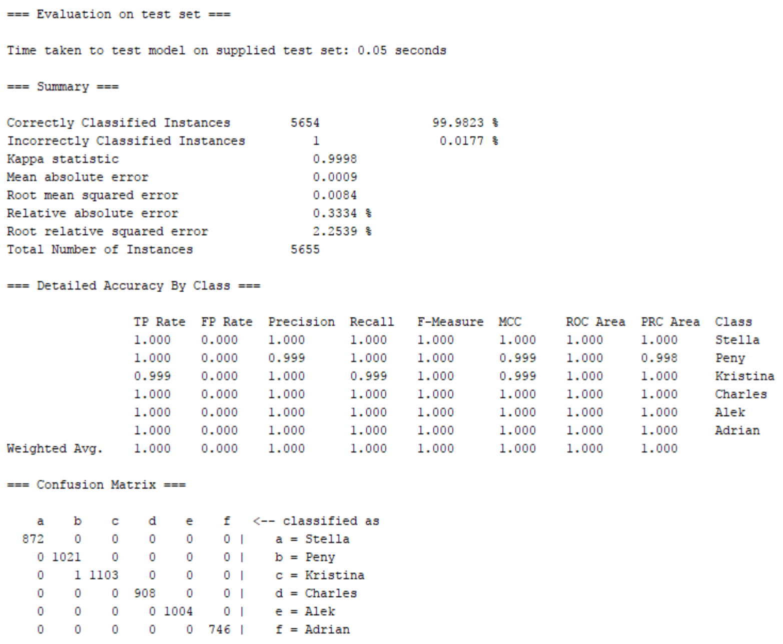
```
Time taken to build model: 0.03 seconds

=== Evaluation on test set ===

Time taken to test model on supplied test set: 0.09 seconds

=== Summary ===

Correctly Classified Instances        5312               93.9346 %
Incorrectly Classified Instances       343                6.0654 %
Kappa statistic                          0.9268
Mean absolute error                      0.0229
Root mean squared error                  0.1241
Relative absolute error                  8.269  %
Root relative squared error             33.3402 %
Total Number of Instances             5655

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0.989    0.034    0.843      0.989    0.910      0.896    0.999     0.987     Stella
                 0.998    0.001    0.994      0.998    0.996      0.995    0.999     0.998     Peny
                 0.993    0.000    1.000      0.993    0.996      0.995    0.999     0.998     Kristina
                 1.000    0.001    0.992      1.000    0.996      0.995    0.999     0.982     Charles
                 0.996    0.034    0.862      0.996    0.924      0.910    0.988     0.945     Alek
                 0.572    0.002    0.979      0.572    0.723      0.724    0.978     0.905     Adrian
Weighted Avg.    0.939    0.012    0.946      0.939    0.934      0.929    0.994     0.972

=== Confusion Matrix ===

    a    b    c    d    e    f   <-- classified as
  862    6    0    0    0    4 |    a = Stella
    1 1019    0    0    0    1 |    b = Peny
    0    0 1096    7    0    1 |    c = Kristina
    0    0    0  908    0    0 |    d = Charles
    1    0    0    0 1000    3 |    e = Alek
  159    0    0    0  160  427 |    f = Adrian
```

Fig. 24. Desk Scenario - Naïve Bayes. Accelerometer and Wi-Fi Sensors.



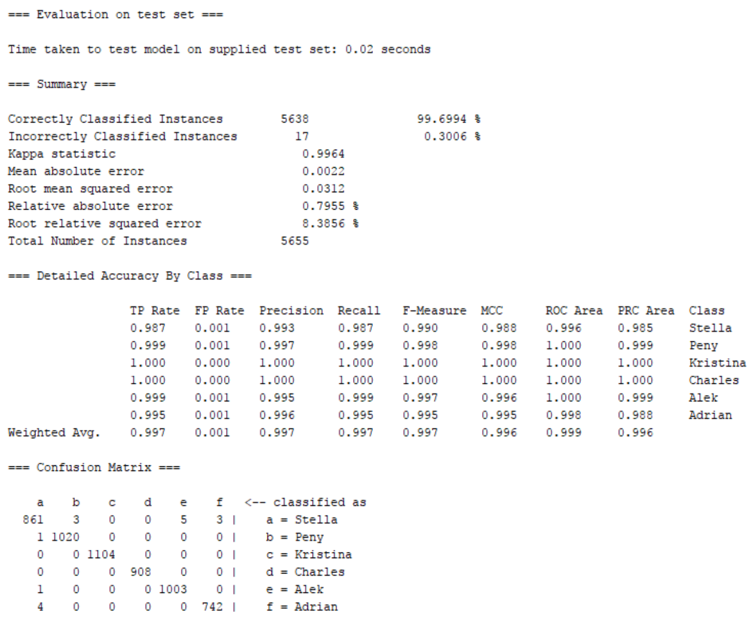
```
=== Evaluation on test set ===

Time taken to test model on supplied test set: 0.1 seconds

=== Summary ===

Correctly Classified Instances        5427               95.9682 %
Incorrectly Classified Instances       228                4.0318 %
Kappa statistic                          0.9514
Mean absolute error                      0.0154
Root mean squared error                  0.1056
Relative absolute error                  5.5606 %
Root relative squared error             28.3657 %
Total Number of Instances             5655

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0.990    0.012    0.938      0.990    0.963      0.957    0.999     0.984     Stella
                 0.998    0.002    0.992      0.998    0.995      0.994    0.999     0.996     Peny
                 0.997    0.000    1.000      0.997    0.999      0.998    1.000     1.000     Kristina
                 1.000    0.000    0.998      1.000    0.999      0.999    1.000     1.000     Charles
                 0.998    0.034    0.862      0.998    0.925      0.911    1.000     0.999     Alek
                 0.716    0.000    0.998      0.716    0.834      0.827    1.000     0.998     Adrian
Weighted Avg.    0.960    0.008    0.964      0.960    0.958      0.953    1.000     0.996

=== Confusion Matrix ===

    a    b    c    d    e    f   <-- classified as
  863    8    0    0    0    1 |    a = Stella
    2 1019    0    0    0    0 |    b = Peny
    1    0 1101    2    0    0 |    c = Kristina
    0    0    0  908    0    0 |    d = Charles
    2    0    0    0 1002    0 |    e = Alek
   52    0    0    0  160  534 |    f = Adrian
```

Fig. 25. Desk Scenario - Naïve Bayes. Orientation and Wi-Fi Sensors.



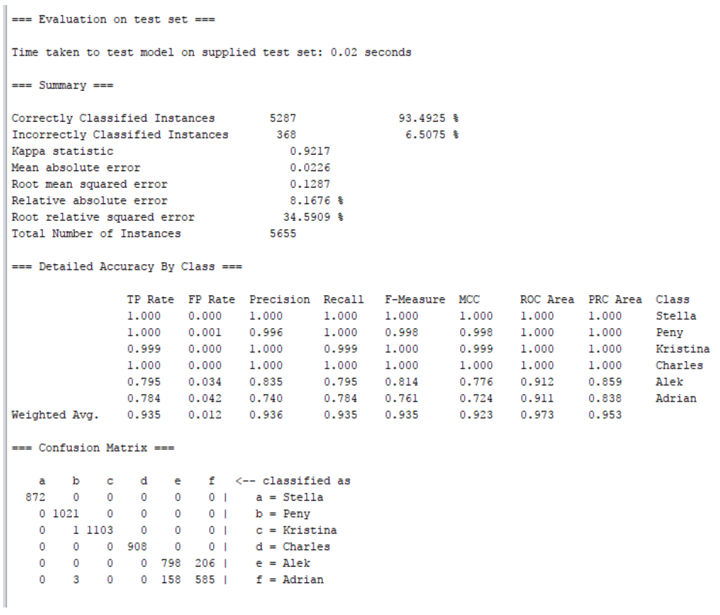
```
=== Evaluation on test set ===

Time taken to test model on supplied test set: 0.05 seconds

=== Summary ===

Correctly Classified Instances        5654               99.9823 %
Incorrectly Classified Instances         1                0.0177 %
Kappa statistic                          0.9998
Mean absolute error                      0.0009
Root mean squared error                  0.0084
Relative absolute error                  0.3334 %
Root relative squared error              2.2539 %
Total Number of Instances             5655

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 1.000    0.000    1.000      1.000    1.000      1.000    1.000     1.000     Stella
                 1.000    0.000    0.999      1.000    1.000      0.999    1.000     0.998     Peny
                 0.999    0.000    1.000      0.999    1.000      0.999    1.000     1.000     Kristina
                 1.000    0.000    1.000      1.000    1.000      1.000    1.000     1.000     Charles
                 1.000    0.000    1.000      1.000    1.000      1.000    1.000     1.000     Alek
                 1.000    0.000    1.000      1.000    1.000      1.000    1.000     1.000     Adrian
Weighted Avg.    1.000    0.000    1.000      1.000    1.000      1.000    1.000     1.000

=== Confusion Matrix ===

    a    b    c    d    e    f   <-- classified as
  872    0    0    0    0    0 |    a = Stella
    0 1021    0    0    0    0 |    b = Peny
    0    1 1103    0    0    0 |    c = Kristina
    0    0    0  908    0    0 |    d = Charles
    0    0    0    0 1004    0 |    e = Alek
    0    0    0    0    0  746 |    f = Adrian
```

Fig. 26. Desk Scenario – Multilayer Perceptron. All Sensors



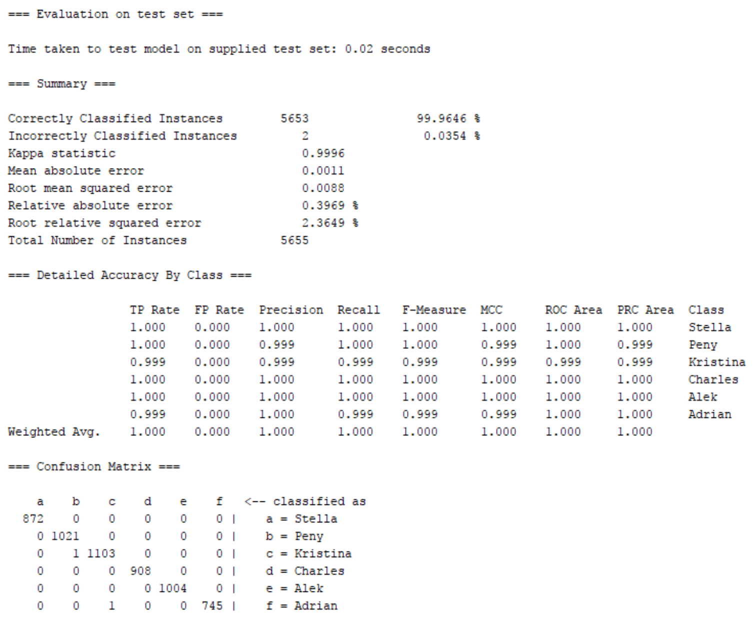
```
=== Evaluation on test set ===

Time taken to test model on supplied test set: 0.02 seconds

=== Summary ===

Correctly Classified Instances        5638               99.6994 %
Incorrectly Classified Instances        17                0.3006 %
Kappa statistic                          0.9964
Mean absolute error                      0.0022
Root mean squared error                  0.0312
Relative absolute error                  0.7955 %
Root relative squared error              8.3856 %
Total Number of Instances             5655

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0.987    0.001    0.993      0.987    0.990      0.988    0.996     0.985     Stella
                 0.999    0.001    0.997      0.999    0.998      0.998    1.000     0.999     Peny
                 1.000    0.000    1.000      1.000    1.000      1.000    1.000     1.000     Kristina
                 1.000    0.000    1.000      1.000    1.000      1.000    1.000     1.000     Charles
                 0.999    0.001    0.995      0.999    0.997      0.996    1.000     0.999     Alek
                 0.995    0.001    0.996      0.995    0.995      0.995    0.998     0.988     Adrian
Weighted Avg.    0.997    0.001    0.997      0.997    0.997      0.996    0.999     0.996

=== Confusion Matrix ===

    a    b    c    d    e    f   <-- classified as
  861    3    0    0    5    3 |    a = Stella
    1 1020    0    0    0    0 |    b = Peny
    0    0 1104    0    0    0 |    c = Kristina
    0    0    0  908    0    0 |    d = Charles
    1    0    0    0 1003    0 |    e = Alek
    4    0    0    0    0  742 |    f = Adrian
```

Fig. 27. Desk Scenario – Multilayer Perceptron. Accelerometer and Magnetometer Sensors



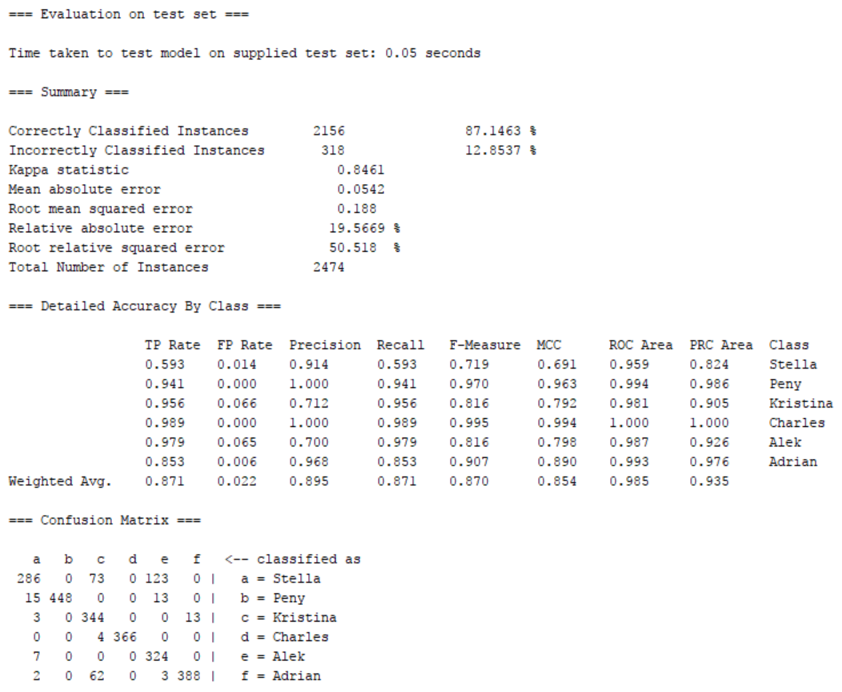
```
=== Evaluation on test set ===

Time taken to test model on supplied test set: 0.02 seconds

=== Summary ===

Correctly Classified Instances        5287               93.4925 %
Incorrectly Classified Instances       368                6.5075 %
Kappa statistic                          0.9217
Mean absolute error                      0.0226
Root mean squared error                  0.1287
Relative absolute error                  8.1676 %
Root relative squared error             34.5909 %
Total Number of Instances             5655

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 1.000    0.000    1.000      1.000    1.000      1.000    1.000     1.000     Stella
                 1.000    0.001    0.996      1.000    0.998      0.998    1.000     1.000     Peny
                 0.999    0.000    1.000      0.999    1.000      0.999    1.000     1.000     Kristina
                 1.000    0.000    1.000      1.000    1.000      1.000    1.000     1.000     Charles
                 0.795    0.034    0.835      0.795    0.814      0.776    0.912     0.859     Alek
                 0.784    0.042    0.740      0.784    0.761      0.724    0.911     0.838     Adrian
Weighted Avg.    0.935    0.012    0.936      0.935    0.935      0.923    0.973     0.953

=== Confusion Matrix ===

    a    b    c    d    e    f   <-- classified as
  872    0    0    0    0    0 |    a = Stella
    0 1021    0    0    0    0 |    b = Peny
    0    1 1103    0    0    0 |    c = Kristina
    0    0    0  908    0    0 |    d = Charles
    0    0    0    0  798  206 |    e = Alek
    0    3    0    0  158  585 |    f = Adrian
```

Fig. 28. Desk Scenario – Multilayer Perceptron. Accelerometer and Wi-Fi Sensors



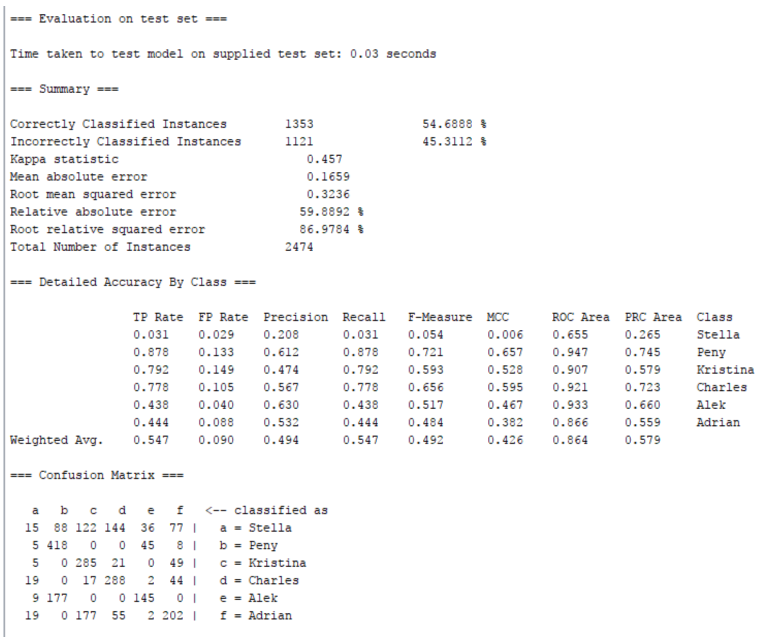
```
=== Evaluation on test set ===

Time taken to test model on supplied test set: 0.02 seconds

=== Summary ===

Correctly Classified Instances        5653               99.9646 %
Incorrectly Classified Instances         2                0.0354 %
Kappa statistic                          0.9996
Mean absolute error                      0.0011
Root mean squared error                  0.0088
Relative absolute error                  0.3969 %
Root relative squared error              2.3649 %
Total Number of Instances             5655

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 1.000    0.000    1.000      1.000    1.000      1.000    1.000     1.000     Stella
                 1.000    0.000    0.999      1.000    1.000      0.999    1.000     0.999     Peny
                 0.999    0.000    0.999      0.999    0.999      0.999    0.999     0.999     Kristina
                 1.000    0.000    1.000      1.000    1.000      1.000    1.000     1.000     Charles
                 1.000    0.000    1.000      1.000    1.000      1.000    1.000     1.000     Alek
                 0.999    0.000    1.000      0.999    0.999      0.999    1.000     1.000     Adrian
Weighted Avg.    1.000    0.000    1.000      1.000    1.000      1.000    1.000     1.000

=== Confusion Matrix ===

    a    b    c    d    e    f   <-- classified as
  872    0    0    0    0    0 |    a = Stella
    0 1021    0    0    0    0 |    b = Peny
    0    1 1103    0    0    0 |    c = Kristina
    0    0    0  908    0    0 |    d = Charles
    0    0    0    0 1004    0 |    e = Alek
    0    0    1    0    0  745 |    f = Adrian
```

Fig. 29. Desk Scenario – Multilayer Perceptron. Accelerometer and Wi-Fi Sensors



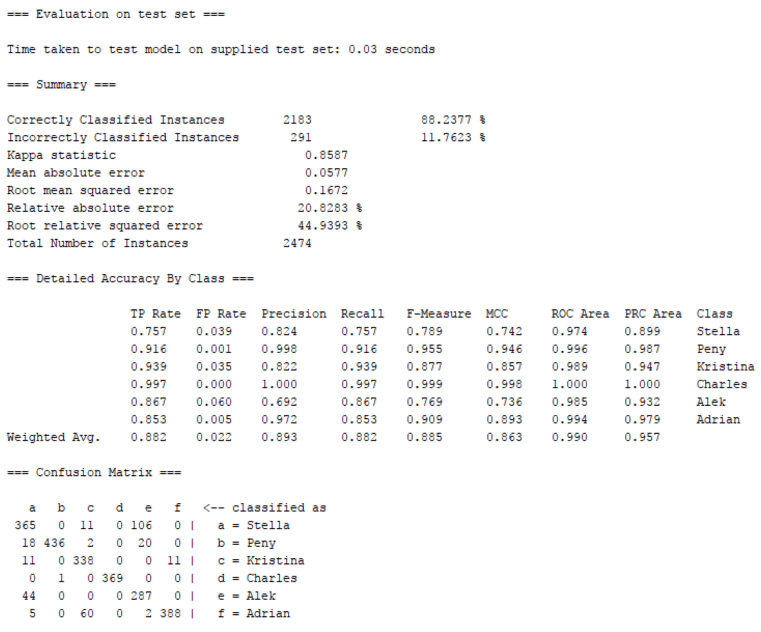
```
=== Evaluation on test set ===

Time taken to test model on supplied test set: 0.05 seconds

=== Summary ===

Correctly Classified Instances        2156               87.1463 %
Incorrectly Classified Instances       318               12.8537 %
Kappa statistic                          0.8461
Mean absolute error                      0.0542
Root mean squared error                  0.188
Relative absolute error                 19.5669 %
Root relative squared error             50.518  %
Total Number of Instances             2474

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0.593    0.014    0.914      0.593    0.719      0.691    0.959     0.824     Stella
                 0.941    0.000    1.000      0.941    0.970      0.963    0.994     0.986     Peny
                 0.956    0.066    0.712      0.956    0.816      0.792    0.981     0.905     Kristina
                 0.989    0.000    1.000      0.989    0.995      0.994    1.000     1.000     Charles
                 0.979    0.065    0.700      0.979    0.816      0.798    0.987     0.926     Alek
                 0.853    0.006    0.968      0.853    0.907      0.890    0.993     0.976     Adrian
Weighted Avg.    0.871    0.022    0.895      0.871    0.870      0.854    0.985     0.935

=== Confusion Matrix ===

   a   b   c   d   e   f   <-- classified as
 286   0  73   0 123   0 |   a = Stella
  15 448   0   0  13   0 |   b = Peny
   3   0 344   0   0  13 |   c = Kristina
   0   0   4 366   0   0 |   d = Charles
   7   0   0   0 324   0 |   e = Alek
   2   0  62   0   3 388 |   f = Adrian
```

Fig. 30. Walking Scenario - Naïve Bayes. All Sensors



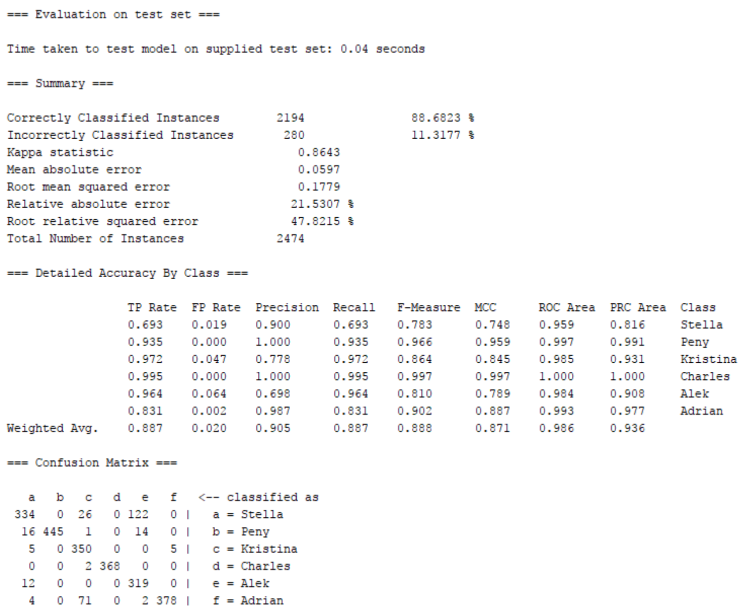
```
=== Evaluation on test set ===

Time taken to test model on supplied test set: 0.03 seconds

=== Summary ===

Correctly Classified Instances        1353               54.6888 %
Incorrectly Classified Instances      1121               45.3112 %
Kappa statistic                          0.457
Mean absolute error                      0.1659
Root mean squared error                  0.3236
Relative absolute error                 59.8892 %
Root relative squared error             86.9784 %
Total Number of Instances             2474

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0.031    0.029    0.208      0.031    0.054      0.006    0.655     0.265     Stella
                 0.878    0.133    0.612      0.878    0.721      0.657    0.947     0.745     Peny
                 0.792    0.149    0.474      0.792    0.593      0.528    0.907     0.579     Kristina
                 0.778    0.105    0.567      0.778    0.656      0.595    0.921     0.723     Charles
                 0.438    0.040    0.630      0.438    0.517      0.467    0.933     0.660     Alek
                 0.444    0.088    0.532      0.444    0.484      0.382    0.866     0.559     Adrian
Weighted Avg.    0.547    0.090    0.494      0.547    0.492      0.426    0.864     0.579

=== Confusion Matrix ===

   a   b   c   d   e   f   <-- classified as
  15  88 122 144  36  77 |   a = Stella
   5 418   0   0  45   8 |   b = Peny
   5   0 285  21   0  49 |   c = Kristina
  19   0  17 288   2  44 |   d = Charles
   9 177   0   0 145   0 |   e = Alek
  19   0 177  55   2 202 |   f = Adrian
```

Fig. 31. Walking Scenario - Naïve Bayes. Accelerometer and Magnetometer Sensors.



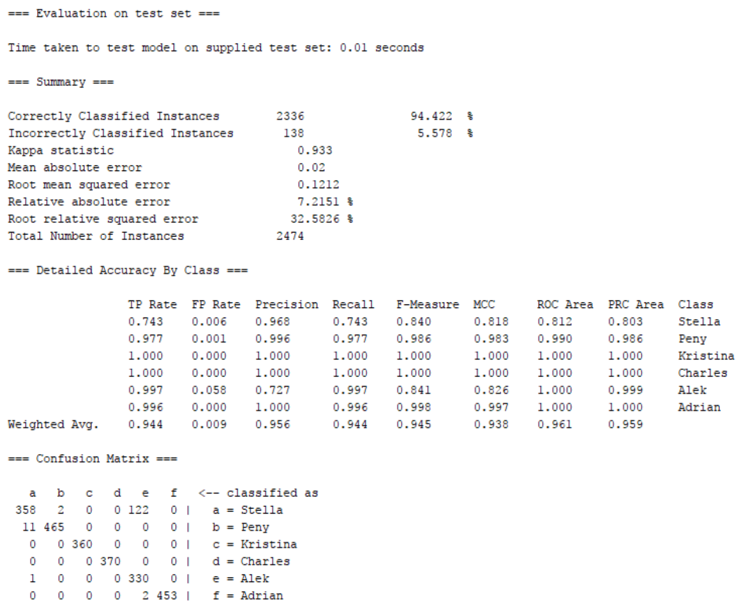
```
=== Evaluation on test set ===

Time taken to test model on supplied test set: 0.03 seconds

=== Summary ===

Correctly Classified Instances        2183               88.2377 %
Incorrectly Classified Instances       291               11.7623 %
Kappa statistic                          0.8587
Mean absolute error                      0.0577
Root mean squared error                  0.1672
Relative absolute error                 20.8283 %
Root relative squared error             44.9393 %
Total Number of Instances             2474

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0.757    0.039    0.824      0.757    0.789      0.742    0.974     0.899     Stella
                 0.916    0.001    0.998      0.916    0.955      0.946    0.996     0.987     Peny
                 0.939    0.035    0.822      0.939    0.877      0.857    0.989     0.947     Kristina
                 0.997    0.000    1.000      0.997    0.999      0.998    1.000     1.000     Charles
                 0.867    0.060    0.692      0.867    0.769      0.736    0.985     0.932     Alek
                 0.853    0.005    0.972      0.853    0.909      0.893    0.994     0.979     Adrian
Weighted Avg.    0.882    0.022    0.893      0.882    0.885      0.863    0.990     0.957

=== Confusion Matrix ===

   a   b   c   d   e   f   <-- classified as
 365   0  11   0 106   0 |   a = Stella
  18 436   2   0  20   0 |   b = Peny
  11   0 338   0   0  11 |   c = Kristina
   0   1   0 369   0   0 |   d = Charles
  44   0   0   0 287   0 |   e = Alek
   5   0  60   0   2 388 |   f = Adrian
```

Fig. 32. Walking Scenario - Naïve Bayes. Accelerometer and Wi-Fi Sensors.



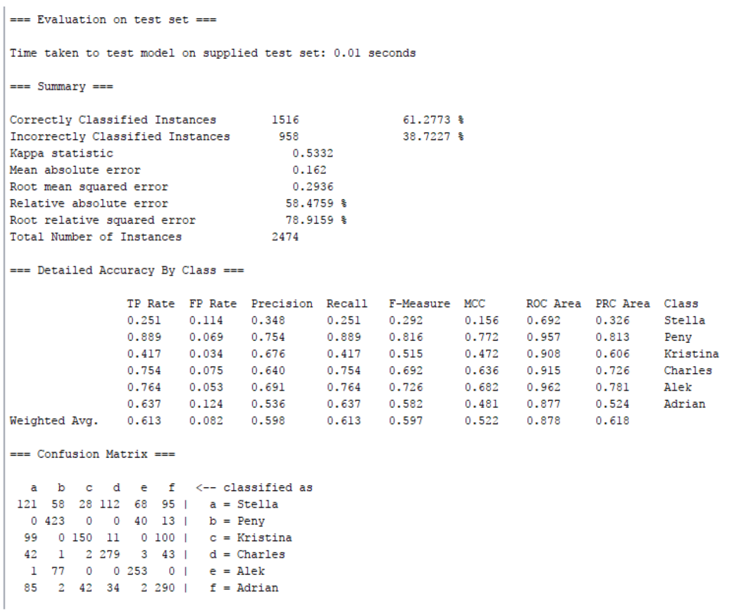
```
=== Evaluation on test set ===

Time taken to test model on supplied test set: 0.04 seconds

=== Summary ===

Correctly Classified Instances        2194               88.6823 %
Incorrectly Classified Instances       280               11.3177 %
Kappa statistic                          0.8643
Mean absolute error                      0.0597
Root mean squared error                  0.1779
Relative absolute error                 21.5307 %
Root relative squared error             47.8215 %
Total Number of Instances             2474

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0.693    0.019    0.900      0.693    0.783      0.748    0.959     0.816     Stella
                 0.935    0.000    1.000      0.935    0.966      0.959    0.997     0.991     Peny
                 0.972    0.047    0.778      0.972    0.864      0.845    0.985     0.931     Kristina
                 0.995    0.000    1.000      0.995    0.997      0.997    1.000     1.000     Charles
                 0.964    0.064    0.698      0.964    0.810      0.789    0.984     0.908     Alek
                 0.831    0.002    0.987      0.831    0.902      0.887    0.993     0.977     Adrian
Weighted Avg.    0.887    0.020    0.905      0.887    0.888      0.871    0.986     0.936

=== Confusion Matrix ===

   a   b   c   d   e   f   <-- classified as
 334   0  26   0 122   0 |   a = Stella
  16 445   1   0  14   0 |   b = Peny
   5   0 350   0   0   5 |   c = Kristina
   0   0   2 368   0   0 |   d = Charles
  12   0   0   0 319   0 |   e = Alek
   4   0  71   0   2 378 |   f = Adrian
```

Fig. 33. Walking Scenario - Naïve Bayes. Magnetometer and Wi-Fi Sensors.



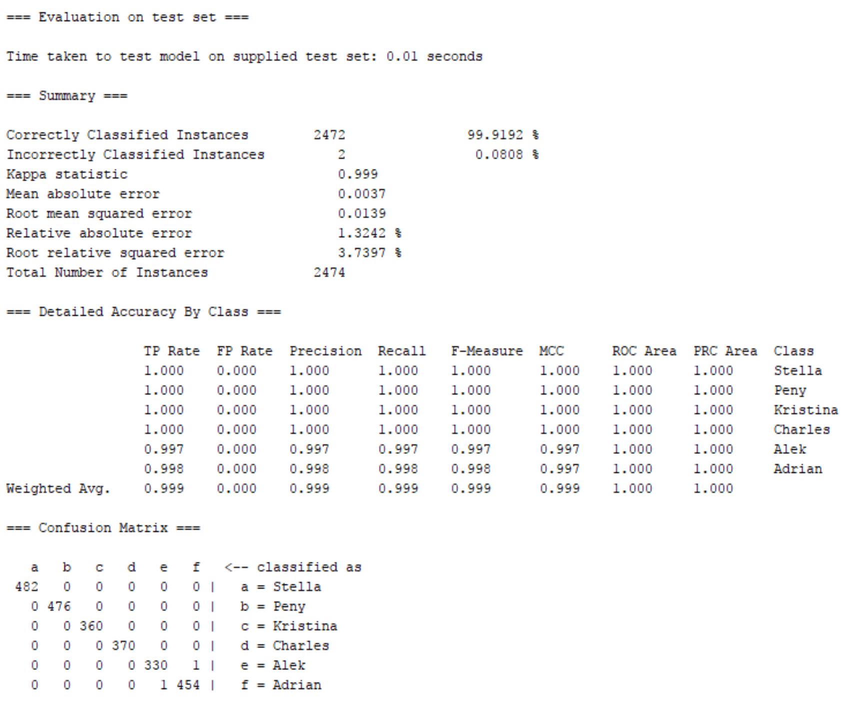
```
=== Evaluation on test set ===

Time taken to test model on supplied test set: 0.01 seconds

=== Summary ===

Correctly Classified Instances        2336               94.422  %
Incorrectly Classified Instances       138                5.578  %
Kappa statistic                          0.933
Mean absolute error                      0.02
Root mean squared error                  0.1212
Relative absolute error                  7.2151 %
Root relative squared error             32.5826 %
Total Number of Instances             2474

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0.743    0.006    0.968      0.743    0.840      0.818    0.812     0.803     Stella
                 0.977    0.001    0.996      0.977    0.986      0.983    0.990     0.986     Peny
                 1.000    0.000    1.000      1.000    1.000      1.000    1.000     1.000     Kristina
                 1.000    0.000    1.000      1.000    1.000      1.000    1.000     1.000     Charles
                 0.997    0.058    0.727      0.997    0.841      0.826    1.000     0.999     Alek
                 0.996    0.000    1.000      0.996    0.998      0.997    1.000     1.000     Adrian
Weighted Avg.    0.944    0.009    0.956      0.944    0.945      0.938    0.961     0.959

=== Confusion Matrix ===

   a   b   c   d   e   f   <-- classified as
 358   2   0   0 122   0 |   a = Stella
  11 465   0   0   0   0 |   b = Peny
   0   0 360   0   0   0 |   c = Kristina
   0   0   0 370   0   0 |   d = Charles
   1   0   0   0 330   0 |   e = Alek
   0   0   0   0   2 453 |   f = Adrian
```

Fig. 34. Walking Scenario – Multilayer Perceptron. All Sensors.



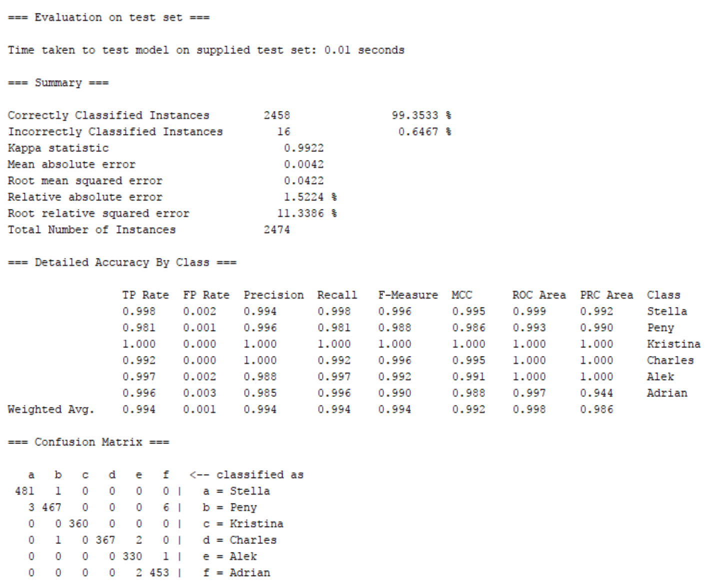
```
=== Evaluation on test set ===

Time taken to test model on supplied test set: 0.01 seconds

=== Summary ===

Correctly Classified Instances        1516               61.2773 %
Incorrectly Classified Instances       958               38.7227 %
Kappa statistic                          0.5332
Mean absolute error                      0.162
Root mean squared error                  0.2936
Relative absolute error                 58.4759 %
Root relative squared error             78.9159 %
Total Number of Instances             2474

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0.251    0.114    0.348      0.251    0.292      0.156    0.692     0.326     Stella
                 0.889    0.069    0.754      0.889    0.816      0.772    0.957     0.813     Peny
                 0.417    0.034    0.676      0.417    0.515      0.472    0.908     0.606     Kristina
                 0.754    0.075    0.640      0.754    0.692      0.636    0.915     0.726     Charles
                 0.764    0.053    0.691      0.764    0.726      0.682    0.962     0.781     Alek
                 0.637    0.124    0.536      0.637    0.582      0.481    0.877     0.524     Adrian
Weighted Avg.    0.613    0.082    0.598      0.613    0.597      0.522    0.878     0.618

=== Confusion Matrix ===

   a   b   c   d   e   f   <-- classified as
 121  58  28 112  68  95 |   a = Stella
   0 423   0   0  40  13 |   b = Peny
  99   0 150  11   0 100 |   c = Kristina
  42   1   2 279   3  43 |   d = Charles
   1  77   0   0 253   0 |   e = Alek
  85   2  42  34   2 290 |   f = Adrian
```

Fig. 35. Walking Scenario - Multilayer Perceptron. Accelerometer and Magnetometer Sensors.



```
=== Evaluation on test set ===

Time taken to test model on supplied test set: 0.01 seconds

=== Summary ===

Correctly Classified Instances        2472               99.9192 %
Incorrectly Classified Instances         2                0.0808 %
Kappa statistic                          0.999
Mean absolute error                      0.0037
Root mean squared error                  0.0139
Relative absolute error                  1.3242 %
Root relative squared error              3.7397 %
Total Number of Instances             2474

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 1.000    0.000    1.000      1.000    1.000      1.000    1.000     1.000     Stella
                 1.000    0.000    1.000      1.000    1.000      1.000    1.000     1.000     Peny
                 1.000    0.000    1.000      1.000    1.000      1.000    1.000     1.000     Kristina
                 1.000    0.000    1.000      1.000    1.000      1.000    1.000     1.000     Charles
                 0.997    0.000    0.997      0.997    0.997      0.997    1.000     1.000     Alek
                 0.998    0.000    0.998      0.998    0.998      0.997    1.000     1.000     Adrian
Weighted Avg.    0.999    0.000    0.999      0.999    0.999      0.999    1.000     1.000

=== Confusion Matrix ===

   a   b   c   d   e   f   <-- classified as
 482   0   0   0   0   0 |   a = Stella
   0 476   0   0   0   0 |   b = Peny
   0   0 360   0   0   0 |   c = Kristina
   0   0   0 370   0   0 |   d = Charles
   0   0   0   0 330   1 |   e = Alek
   0   0   0   0   1 454 |   f = Adrian
```

Fig. 36. Walking Scenario - Multilayer Perceptron. Accelerometer and Wi-Fi Sensors.



```
=== Evaluation on test set ===

Time taken to test model on supplied test set: 0.01 seconds

=== Summary ===

Correctly Classified Instances        2458               99.3533 %
Incorrectly Classified Instances        16                0.6467 %
Kappa statistic                          0.9922
Mean absolute error                      0.0042
Root mean squared error                  0.0422
Relative absolute error                  1.5224 %
Root relative squared error             11.3386 %
Total Number of Instances             2474

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0.998    0.002    0.994      0.998    0.996      0.995    0.999     0.992     Stella
                 0.981    0.001    0.996      0.981    0.988      0.986    0.993     0.990     Peny
                 1.000    0.000    1.000      1.000    1.000      1.000    1.000     1.000     Kristina
                 0.992    0.000    1.000      0.992    0.996      0.995    1.000     1.000     Charles
                 0.997    0.002    0.988      0.997    0.992      0.991    1.000     1.000     Alek
                 0.996    0.003    0.985      0.996    0.990      0.988    0.997     0.944     Adrian
Weighted Avg.    0.994    0.001    0.994      0.994    0.994      0.992    0.998     0.986

=== Confusion Matrix ===

   a   b   c   d   e   f   <-- classified as
 481   1   0   0   0   0 |   a = Stella
   3 467   0   0   0   6 |   b = Peny
   0   0 360   0   0   0 |   c = Kristina
   0   1   0 367   2   0 |   d = Charles
   0   0   0   0 330   1 |   e = Alek
   0   0   0   0   2 453 |   f = Adrian
```

Fig. 37. Walking Scenario - Multilayer Perceptron. Magnetometer and Wi-Fi Sensors.