# Investigating keystroke dynamics as a two-factor biometric security.

Brian M. Njogholo
*Faculty of Information Technology (FIT)*
*Strathmore University*

# Investigating Keystroke Dynamics as a Two-Factor Biometric Security

Submitted by:

Brian Njogholo Mwandau

069283

A dissertation submitted in partial fulfillment of the requirements for the award of Master of Science in Information System Security at iLabAfrica, Strathmore University

Faculty of Information Technology

Strathmore University

Nairobi, Kenya

April, 2018

# Declaration and Approval

I hereby declare that this research has not been submitted to any other University for the award of a Degree in Master of Science in Information System Security. The work herein is originally done by Brian Mwandau under the guidance and supervision of Dr. Matunda Nyanchama, Faculty of Information Technology in Strathmore University.

Student: Brian Njogholo Mwandau

Student No: 069283

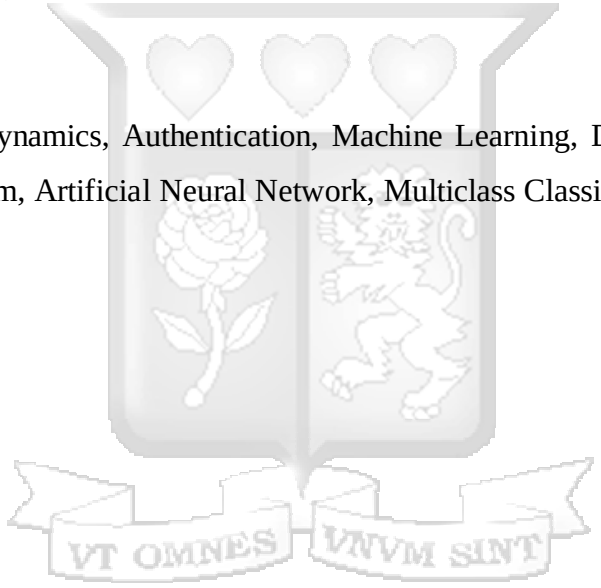
Student Signature: _____ Date: _____

Supervisor: Dr. Matunda Nyanchama

Supervisor Signature: _____ Date: _____

## Abstract

Keystroke dynamics is the study of how people can be distinguished based on their typing rhythms. This proposal aims at investigating user authentication approaches and how keystroke dynamics can be used to enhance user authentication and access control. With more users embracing technologies and using applications without necessarily understanding the security repercussions, a further protection mechanism needs to be employed. It emphasizes on the need of an additional layer of security, through keystroke dynamics, on top of the traditional username-password combination to enhance security during authentication. It also proposes the use of a machine learning classifier for possible application in keystroke dynamics to verify and validate the legitimacy of a user during authentication.



**Keywords**: Keystroke Dynamics, Authentication, Machine Learning, Deep Learning, Security, Biometric, Typing Rhythm, Artificial Neural Network, Multiclass Classification, Validation.

# Table of Contents

# Table of Figures

# List of Tables

# Acknowledgement

I begin by acknowledging God.

I would like to acknowledge my supervisor, Dr. Matunda Nyanchama for motivating me to pursue this research, and for his guidance.

I would also like to show my gratitude to the lecturers, who have shared their knowledge and experience in security throughout the Masters course.

Special thanks to my friends and family for the moral support and motivation to pursue this Masters and complete my dissertation.

# Abbreviations/Acronyms

| | | |
|---|---|---|
| **ANN** | - | Artificial Neural Network |
| **MLP** | - | Multilayer Perceptron |
| **SGD** | - | Stochastic Gradient Descent |

# Chapter 1: Introduction

## 1.1 Background

Most systems and applications nowadays require users to authenticate themselves by logging in using their respective username and password combinations. This enables them to verify their claimed identity and access their data, which might be different from other users' data depending on the application. But even the most sophisticated and advanced systems employed in an organization are susceptible to attacks that arise from flaws in passwords due to the human factor (Awad, Al-Qudah, Idwan, & Jallad, 2016). It is therefore recommended that passwords should not be used as a sole methods of authentication. Users tend to create simple passwords that are susceptible to brute force attacks and social engineering methods. An imposter, be it from within an organization or external, can simply guess a password and gain access into the system.

For a user to be authenticated they have to be identified. Identity is based on something you know, for example a password or pin; something you have, like a token or smartcard; or something you are, some biometric property such as fingerprint or iris.

Biometric authentication maintains its status as the most secure authentication and verification method because it cannot be forgotten, transferred or stolen (Patil & Renke, 2016). Replicating a biometric can also prove to be very difficult. The identity of the user through biometrics is determined either by their physiological (static) or behavioral (dynamic) features. Physiological features are those that are physically related to a person, for example iris, fingerprint and retina. On the other hand, behavioral features are those that people have learnt to do over a period of time, such as gait, signatures, and keystroke dynamics.

A major advantage of keystroke dynamics over other traditional biometric authentication mechanisms such as iris and fingerprint is the low implementation and deployment costs involved (Teh, Teoh, & Yue, 2013). Palm, iris and fingerprint recognition when capturing authentication data rely heavily on dedicated devices and hardware infrastructure, which are expensive to acquire, whereas keystroke dynamics recognition is entirely software implementable. Another major advantage is continuous monitoring and authentication. Keystroke dynamics offers a way to

continuous validate the legitimacy of a user. As long as user interaction with the system through input devices persist, keystroke pattern can be constantly monitored and reevaluated.

## 1.2 Problem Statement

The number of Internet users worldwide is growing at an exponential rate, with the number increasing fourfold in the last 10 years to just over 4 billion as at December 2017 (Miniwatts Marketting Group, 2018). This represents approximately 54 percent of the world population. Furthermore, their activities and significance of these activities online is also increasing. Most online and offline applications and systems that they interact with for daily activities rely on usernames and passwords for authentication and access control. Such applications include social media platforms, ERP systems and proprietary systems used in specific organizations. The number of tools and devices used for hacking into systems and gaining unauthorized access are becoming more and more easily available even to rookie hackers (Chandrika, 2014). The traditional methods of authentication such as username-password combinations are not as reliable in providing security as they used to due to technology advancements that even hackers exploit to gain unauthorized access into systems and sensitive data. The number of security breaches through unauthenticated entry is alarming.

The users might not necessarily be familiar enough with the ever increasing number of security threats and so there is a need to provide an additional layer of security.

## 1.3 Research Objectives

1. To understand user authentication approaches and how keystroke dynamics can be used to enhance user authentication and access control.
2. To understand previous research on keystroke dynamics and related classification methods that have been applied in keystroke dynamics.

3. To develop and train a suitable machine learning classifier to recognize users and validate their authenticity during authentication.
4. To test and validate the accuracy of the proposed machine learning classifier in validating user authenticity through keystroke dynamics.

## 1.4 Research Questions

1. What approaches have been used in user authentication and what role does keystroke dynamics play in enhancing user authentication?
2. What research has been previously done on classification models employed in keystroke dynamics?
3. How can a machine learning classifier be developed and trained to discriminate legitimate users from imposters during authentication?
4. With what accuracy can the machine learning classifier validate users?

## 1.5 Research Hypothesis

One of the hypotheses is that keystroke dynamics is that the machine learning classifier identified and tested will have an acceptable accuracy rate for it to correctly classify users depending on their typing rhythm.

Another hypothesis is that the research will provide a strong basis to support the claim that keystroke dynamics as a behavioral biometric security mechanism can be used to improve authentication and access controls employed in applications and systems.

## 1.6 Scope and Limitations

The scope of the research will focus on mainly the analysis of an already existing dataset to classify users based on their already-captured typing rhythm, as opposed to developing our own keystroke dynamics system in order to store the typing rhythms of users. This dataset must have at least the basic measurement parameters suitable for keystroke dynamics, which we will discuss in detail in the next chapter. The dataset will be limited to measurement parameters for a common password phase for all users as opposed to the use of different passphrases by different users.

The most commonly used programming languages for machine learning and data science in general, according to recent polls, are R and Python (Piatetsky, 2017). This research will be purely based on python programming within a Windows environment. Further details will be provided in Chapter 3.

## 1.7 Justification

Each individual has a particular typing pattern and speed. If we record patterns of persons then we can correctly identify whether they are legitimate or not. In this way we add an extra layer of protection in a sense that even if someone get to know our password then we can identify the fake user when their typing rhythm does not match with that of the actual user.

With more users embracing technologies and using applications without necessarily understanding the security repercussions, a further protection mechanism needs to be employed. Keystroke dynamics can solve this by providing an additional layer of security to the common and traditional systems based on passwords to provide authentication.

This chapter introduced keystroke dynamics as an area that can be used to improve authentication of users to improve their security in this age where advancement of technology also leads to a corresponding increase in cyber threats. It explained the flaws of traditional password-based systems and why there is a need for improvement. It also introduced machine learning as one of the enablers of the analysis of keystroke dynamics. In the next chapter we will review the literature based on the research in order to have a better insight on how well we can improve authentication and access control using keystroke dynamics.

VT OMNES
VNVM SINT

# Chapter 2: Literature Review

## 2.1 User Authentication

Most, if not all systems used in corporate businesses and governments contain sensitive and valuable information. These systems are continuously relied on in day-to-day activities and unauthorized access to such systems due to poor authentication could lead to massive losses for both organizations and stakeholders (Mwagwabi, McGill, & Dixon, 2014). Consequences ranging from data theft and destruction to financial losses and reputational damage could all result due to this unauthorized access. In order to ensure security, one of the most obvious methods is user authentication. Authentication is the process of confirming a user's identity; to determine that the person really is, in fact who they claim to be (Vinayak & Arora, 2015).

## 2.2 Authentication Approaches

Identity confirmation is based on either what we know, what we have or what we are.


*Figure 2 1: Authentication Approaches*

Source (Scotton & Alexander, 2016).

### 2.2.1 Knowledge

This is generally one of the most used authentication methods. The user is authenticated based on something only they know so as to protect the security of the data. It is commonly used in the form of passwords, pin codes, pattern codes or a piece of information. Passwords and pins have been the generally accepted and established methods of knowledge-based authentication over the past few decades (Karnan & Akila, 2009). Though as the years have progressed they have become weaker due to availability of highly sophisticated password cracking tools, poor user password habits as well as limitations of human nature (Rathanavel & Mali, 2017). For example, using a password that can easily be guessed. On the other hand, if a password is hard to guess, then it is often hard to remember. The users might also opt to use the same password for authentication in multiple applications.

### 2.2.2 Token-based

Token based authentication deals with what the user physically possesses, such as a smart card, key, token and RFID-Tag. They are generally more secure but are susceptible to loss or theft and the user can find it difficult to keep safe at all times due to their physical nature (Vinayak & Arora, 2015). In such scenarios mechanisms employed to enhance security through token-based authentication is usage of tokens alongside knowledge-based method.

### 2.2.3 Biometrics

Biometrics can be described as the physiological or behavioural characteristic that is uniquely associated to a person (Teh, Teoh, & Yue, 2013). It is the autonomous recognition of individuals on the basis of the characteristic anatomic features (physiological) and characteristic behavior.

Biometric authentication is the most secure and convenient method of authentication (Patil & Renke, 2016). It is basically who the person is. It cannot be transferred, borrowed, forgotten or

stolen, and forging one is particularly difficult. It is therefore highly unique and can therefore be used to differentiate one individual from another.

Physiological biometrics are those features that describe who the user is depending on their physical attributes (Vinayak & Arora, 2015). These physical attributes include fingerprints, iris, retina, hand or palm geometry and facial recognition characteristics. Physiological biometrics are considered highly robust and secure, with one major caveat: price. Acquisition of specialized hardware and software combinations are needed to detect the physical features.

Behavioral characteristics include signature, voice, typing rhythm (keystroke dynamics) and walking style (gait recognition). In addition to being the cheaper than physiological biometrics, another edge that behavioral biometrics has over physiological biometrics is the ability to work on stealth mode verification (Teh, Teoh, & Yue, 2013). The user might not necessarily know that they are being monitored. As such, minimal interaction is required during authentication process reduces invasiveness and thus promotes user acceptability. One drawback to behavioral biometrics is the nature of its variability over time, and therefore related systems need to be designed to be more dynamic and accept some level of instability (Malinka, 2009).

## 2.3 Keystroke Dynamics

Keystroke dynamics is a behavioral biometric that analyzes the way a user types on a keyboard and identify them based on their habitual typing rhythm. One assumption of keystroke dynamics is that different people type in a different manner (Patil & Renke, 2016). Each user has a particular typing pattern and speed, which are dependent on several factors. If these patterns are recorded then we can be able to correctly identify whether the user is legitimate or not. This adds an extra layer of security such that even when someone knows our password they can be identified as an imposter when their behavioral typing rhythm does not match with that of the legitimate user.

Basic measurement parameters for keystroke dynamics used to describe a user's typing pattern are:

1. Latencies between successive keystrokes (the elapsed time between the release of the first key and the depression of the second).

2. Duration of each keystroke (How long is the key held down).
3. Finger placement.
4. Pressure applied on the keys.
5. Overall typing speed (Barghouthi, 2009).

These measurements provide timing information that vary between users. It is due to this difference in timings that we can authenticate users and differentiate legitimate users from imposters based on their typing characteristics. Analysis of keystroke dynamics can be useful in protecting personal data because an individual is authenticated not only by password, but also by that individual's keystroke patterns. In this way, intrusion becomes more difficult because the username/password pair, as well as the typing rhythm and correct keystroke pattern must both be duplicated, which is very difficult to achieve.

Access control could incorporate keystroke dynamics both by requiring a legitimate user to type a password with the correct rhythm, and by continually authenticating that user while they type on the keyboard.

Studies show that biometric authentication through keystroke dynamics is a very active sub-field of the larger domain of computer security, and has already reached a high level of maturity (Giroux & Wachowiak-Smolikova, 2009). These studies have demonstrated that different users exhibit different typing habits that discriminate them from one another. In systems that implement keystroke patterns as part of their authentication mechanisms, the login process requires not only the correct username and password, but also matching specific keystroke patterns that have been identified and stored for that individual. Such a mechanism could increase the difficulty of intruders being authenticated, as the unique typing pattern would be very difficult to reproduce.

In a somewhat early study of keystroke dynamics, pattern recognition methods were employed to identify users by constructing a database of 42 profiles based on the users' keystroke profiles (Monrose & Rubin, 2000). The users ran a data collection programme in their personal machines for which their keystroke patterns were recorded, and after analysis were classified. Various classification methods were used to identify users, recording success rates between 83.22% and

92.14% depending on the method employed. The users were requested to enter pre-determined text as opposed to free text during authentication – an approach that will be used in this research.

The feasibility of validation through digraphs and trigraphs was also investigated. A digraph is the difference in time latencies between two successive keystrokes, that is, the time difference between the key press event of the first key and the release event of the second key, while a trigraph is the time difference between the key press event of the first key and the release event of the third key. The study tested users who logged in as themselves, and also as someone else to simulate impersonation attempts. The false rejection rate was between 0% and 55%, and the false acceptance rate was 0% for 80% of those acting as intruders. This was considered to be a satisfactory result. False acceptance rate (FAR) is the ratio of number of false matches divided by total number of fraud match attempts (Patil & Renke, 2016). Thus FAR gives the number of frauds or imposters who are inaccurately allowed as genuine users. A smaller FAR indicates less imposter accepted. False rejection rate (FRR) is the ratio of number of false rejections divided by total number of genuine match attempts. Thus FRR gives the number of genuine users who are rejected from using the system. Higher FRR is preferred in high security systems. A lower FRR implies less rejection and easier access by genuine user.

Sim and Janakiraman, in their study on keystroke analysis using free text, they investigated the effectiveness of digraphs and more generally n-graphs for free text keystroke biometrics, and concluded that n-graphs are discriminative only when they are word-specific (Sim & Janakiraman, 2007). As such, the digraph and n-graph features do depend on the word context they are computed in.

## 2.4 Classification/Matching in Keystroke Dynamics

This is the most important phase of most recognition systems. It is the stage where feature data are categorized and discriminated for later use to make decision. Diverse algorithms have been applied by previous researches with a common goal of increasing authentication accuracy. These algorithms include K-means, Nearest Neighbor rule, Bayes classifier, Support Vector Machine and Artificial Neural Networks.

Majority of the pattern recognition algorithms employed in keystroke dynamics for the past three decades can be broadly classified into two main categories, namely, statistical and machine learning approaches (Teh, Teoh, & Yue, 2013).

### 2.4.1 Statistical Matching

Statistical approaches have been employed in the early stages of keystroke dynamics (Senk & Dotzler, 2011). They have also been employed in recent works due to their simplicity and ease of implementation (Stweart, Monaco, Cha, & Tappert, 2011). Some of the basic statistical methods employed are mean, median, standard deviation, k-nearest neighbor and statistical t-test. Distance techniques like Euclidean distance, weighted Euclidean distance, Manhattan distance etc. are used for comparing the training dataset with testing dataset (Banerjee & Woodard, 2012). It is not necessary that the data collected for keystroke authentication and verification is linear, thus sometimes these linear statistical approaches do not provide good results. So, there is a need of some approaches that use probabilistic data rather than deterministic data for pattern recognition.

### 2.4.2 Machine Learning Classifiers

Pattern recognition is defined as an act of taking raw data (patterns, objects) and classifying them into different categories based on algorithms. Pattern recognition includes machine-learning algorithms, various classification techniques like Nearest Neighbor rule, Bayes classifier, and Support Vector Machine, clustering techniques like K-means etc. Various artificial neural networks (ANN) have also been applied to the keystroke dynamics (Deng & Zhong, 2015).

Singh and Thakur developed an artificial neural network classifier for free-text in keystroke dynamics (Singh & Thakur, 2012); and even though it produced good results, it was recommended that the use of a pre-determined text such as the password for the different users would produce more accurate results due to uniformity of the feature data obtained. It is due to this reason that the

data to be involved in this research will involve text set beforehand (such as the same password) for all users as opposed to free text.

# Chapter 3: Methodology

This chapter introduces the methodology that will be implemented in developing the machine learning algorithm for authenticating users based on their typing rhythms. The chapter covers the proposed methodology, the algorithm requirements and design, and the implementation details.

## 3.1 Methodology

This research will employ the Rapid Application Development (RAD) methodology for software development. RAD is a condensed software development process that produces a high quality system with low investment costs aimed at reducing the amount of construction needed to build a product (Korkishko, 2017). In its most basic form, RAD minimizes planning and intensifies prototyping by allowing developers to quickly adjust to shifting requirements in a fast-paced and constantly changing environment (Stiner, 2016).

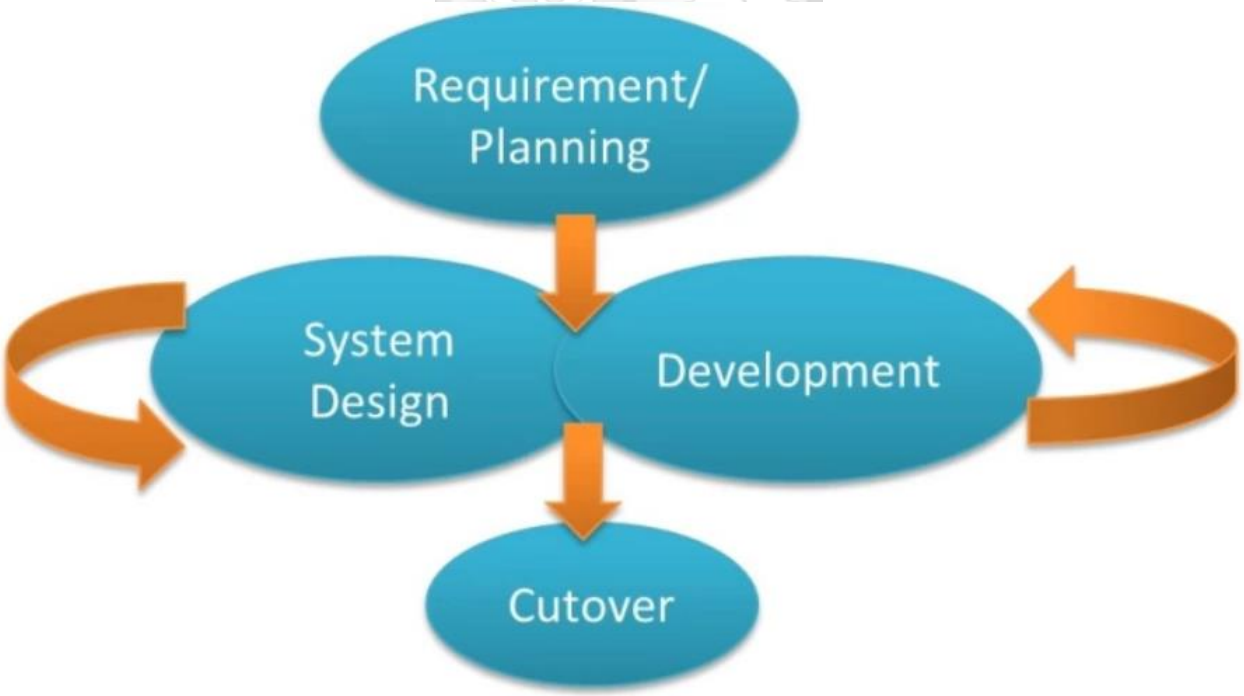



*Figure 3 1: Rapid Application Development Phases*

Source (GormLey, 2017).

The RAD methodology is suitable for this research because the objectives are clearly defined in the requirements planning phase and the condensed development timeline, so that more focus can be emphasized in the next phases, which are system design and development of the machine learning classifier. These two phases will be repeated until all requirements of the algorithm as defined in the objectives are met. The cutover phase is where implementation and testing of the algorithm will be done.

## 3.2 Requirements Analysis

This is the first phase of the RAD methodology in which the algorithm requirements are defined. It is the phase where the objectives are clearly defined, together with the scope and application requirements. These were discussed in chapter 1 when developing the objectives and research questions.

Any machine learning algorithm cannot be modelled without data. Therefore acquiring relevant data is a mandatory requirement. The dataset is discussed further in the next section of this chapter.

It would also be recommended to use a high performance desktop/laptop computer to create a conducive algorithm development environment, since the proposed classifier would require high processing power and adequate RAM. Recommendation would be Core i5 or later and minimum 6 GB of RAM. Tools and techniques to be used in such an environment are discussed later in this chapter.

Generally all major keystroke dynamics evaluations involve (1) recruiting subjects for data collection & presenting them with a typing task, (2) recording keystroke-timing information, (3) feature extraction suitable for training and testing a classifier, (4) training the classifier using one portion of the typing data and (5) testing the classifier performance using another portion of typing data (Patil & Renke, 2016). Researchers make a lot of choices in each of the phase. This research is focused on analyzing keystrokes and training a machine learning classifier to validate the

authenticity of a user. It will therefore use a pre-existing dataset that the owner has made available to the public so that further research can be done using it.

### 3.2.1 Data Selection

This research will employ the dataset collected by Killourhy and Maxion to develop a repeatable evaluation procedure (Killourhy & Maxion, 2009). The data is arranged as a table with 34 columns. Each row of data corresponds to the timing information for a single repetition of the password by a single subject. The first column, subject, is a unique identifier for each subject (e.g., s002 or s057). Even though the data set contains 51 subjects, the identifiers do not range from s001 to 051; subjects have been assigned unique IDs across a range of keystroke experiments, and not every subject participated in every experiment. For instance, Subject 1 did not perform the password typing task correctly and so s001 does not appear in the data set. The second column, sessionIndex, is the session in which the password was typed (ranging from 1 to 8). The third column, rep, is the repetition of the password within the session (ranging from 1 to 50). The remaining 31 columns present the timing information for the password. The name of the column encodes the type of timing information. The keystroke dynamics features employed in this dataset are:

1. **Hold time (H)** – the time between press and release of a key
2. **Keydown-Keydown time (DD)** – the time recorded between the pressings of two consecutive keys.
3. **Keyup-Keydown time (UD)** – the time between the release of one key and the press of the next key.

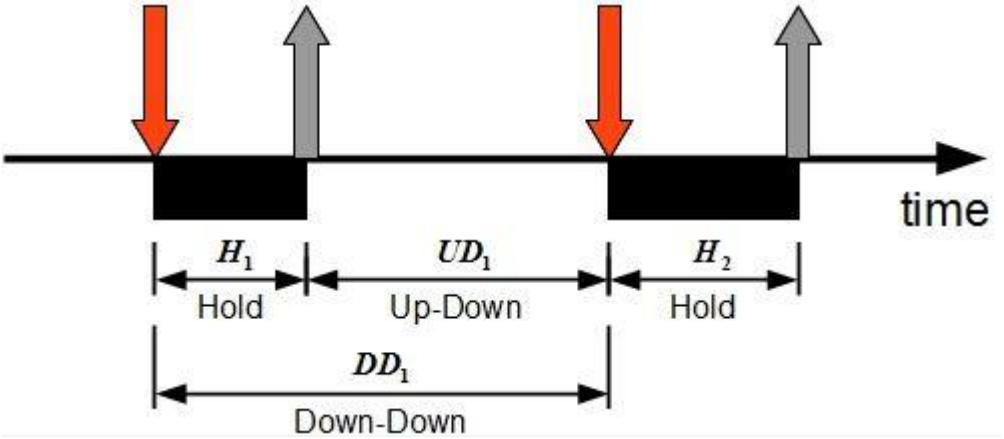



*Figure 3 2: Keystroke timing features*

Source ( ML BOT1, 2017)

The figure below shows how the three time features above are used as features in this dataset.

| | subject | sessionIndex | rep | H.period | DD.period.t | UD.period.t |
|---|---|---|---|---|---|---|
| **0** | s002 | 1 | 1 | 0.1491 | 0.3979 | 0.2488 |
| **1** | s002 | 1 | 2 | 0.1111 | 0.3451 | 0.2340 |

*Figure 3 3: Sample timing features as used in dataset*

In the end any data with errors, blanks and other issues were removed, remaining with data for 51 individuals. I considered this a good data set because each of the 51 users participated 50 times per session, in 8 different sessions that were at least a day's gap between the sessions to ensure that any day-to-day variations between the user's typing can be captured. This produced 400 typing repetitions of the password (**.tie5Roanl**) for each user. The use of a common password among all participants was to ensure the comparability of typing patterns of the same password from different individuals. The dataset was downloaded from Kaggle (Suman, 2017). The data has been continuously reviewed by the Kaggle community of data scientists since 2009.

## 3.3 System Design

System design entails coming up with the conceptual design of the algorithm. In this research a multilayer perceptron (MLP) classifier will be used to model the algorithm to be used for user authentication via keystroke dynamics. A MLP is a feed forward artificial neural network model that maps sets of input data onto a set of appropriate outputs (Wankhede & Verma, 2014). The classifier is used to analyse the features of the user.

A deliverable in this stage will be a flow map showing the logical flow of how the algorithm is modelled and trained to learn keystroke patterns of users up to the point it is evaluated and user verification can be done.

## 3.4 Development

The algorithm shall be developed using the Python programming language. Specific Python libraries that are essential and will be used for the modelling include:

1. *Keras*. This is a high-level neural networks API, written in python that allows for easy and fast prototyping through user friendliness, modularity and extensibility (Keras Team, 2018).
2. *Jupyter Notebook*. The Jupyter Notebook is an open-source web application that allows you to create and share documents that contain live code, equations, visualizations and narrative text (Jupyter, 2018). This will be the environment in which the algorithm development will be done.
3. *Scikit-learn, Numpy, Pandas, Seaborn and Matplotlib* are other python libraries that will be used for algorithm development.

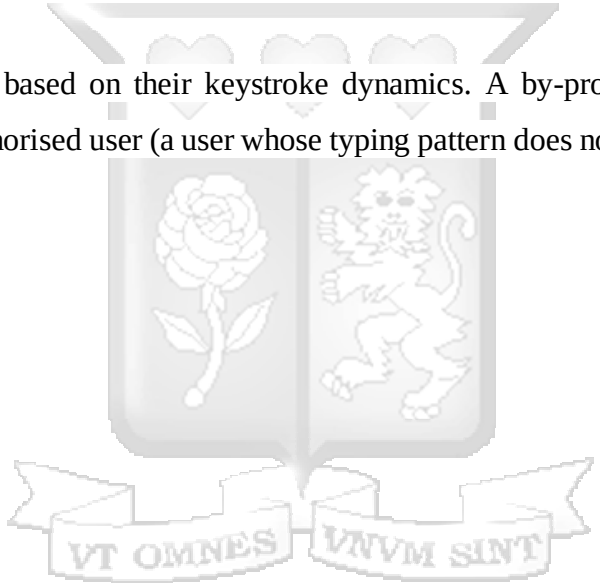
## 3.5 Cutover/Implementation

The aim of this stage is to demonstrate that the proposed algorithm satisfies the specified requirements and objectives.

This is the final phase of the RAD methodology in which both implementation and testing are performed. Several iterations are done for the system design and development phases before the cutover phase. This allows for quick prototypes to be developed, allowing the user to easily envision the final solution.

Once the model is trained, its accuracy will be checked using unseen test data. This will be done through inference on the test data and it will enable us determine two things:

1. The accuracy of the developed algorithm in recognizing a user based on their typing patterns.
2. Predicting a user based on their keystroke dynamics. A by-product of this is correctly flagging an unauthorised user (a user whose typing pattern does not match the tested typing patterns).



# Chapter 4: Algorithm Design and Architecture

## 4.1 Introduction

This chapter describes the procedure in which the algorithm will be designed and optimized for verifying the identity of users based on their keystroke timings. It mainly involves the steps involved in the development of a deep learning artificial neural network model that is able to learn from the timing features that were recorded for each user while typing the unique password (**.tie5Roanl**).

An artificial neural network gets its inspiration from the biological nervous system to process information (Singh & Thakur, 2012). It contains a large number of interconnected neurons (nodes) that work collectively to process the information. These neurons are arranged into interconnected layers to form an artificial neural network, hence increasing the learning capabilities over that of a single neuron.

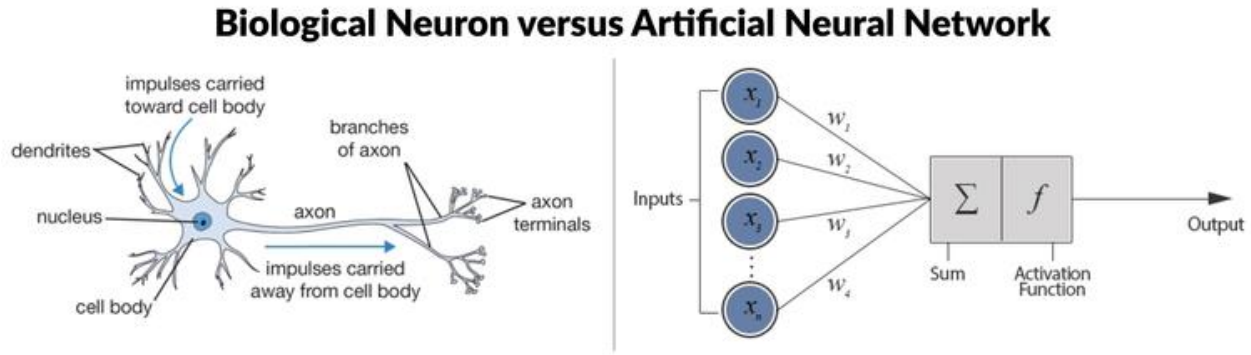



*Figure 4 1: Biological neuron versus an artificial neural network*

Source (Willems, 2017)

A neural network processes information by multiplying an input variable by a pre-determined weight. These inputs in the neurons are the parameters which the neural network has to learn during training. After all inputs ($x_1$, $x_2$, $x_3 \ldots x_n$) in one layer are multiplied by their respective weights ($w_1$,

$w_2$, $w_3 \ldots w_n$), their sum is computed and passed through an activation function to normalize the sum before an output is produced (Gupta, 2017). This output could also be an input of another node in yet another layer. This ability of artificial neural networks to contain more than one layer for information processing gives rise to what is termed as deep learning.

## 4.2 Model Architecture

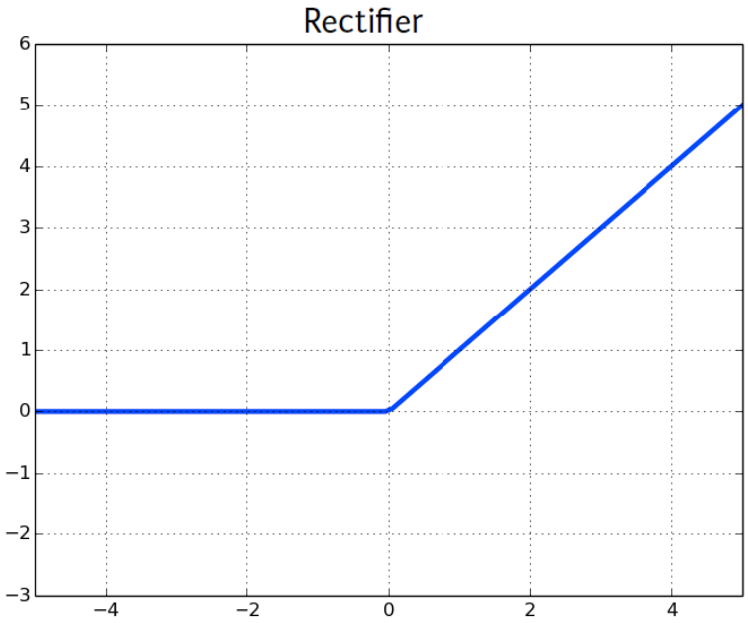
This research will employ a powerful deep learning library called *Keras* to build develop the deep learning neural network. Keras will use TensorFlow as its backend software library for high performance numerical computation (TensorFlow, 2018). Keras delegates low-level operations such as tensor (matrix) manipulation and differentiation to TensorFlow, hence allowing for fast and easy prototyping of models.

As a recap, 400 instances were recorded for each of the 51 users over 8 sessions per user. Sessions had a gap of at least one day. Key-hold times, keyup-keydown times and keydown-keydown times for each keystroke of the password were recorded, leading up to a total of 31 timing features.

The neural network will have 4 layers: an input layer, two hidden layers and an output layer. The input layer is the first layer, and is used to provide the input data to the network. Since the neural network will learn from the timing features, they will be the ones fed to the input layer, hence the input layer will contain 31 nodes.

The model will have two hidden layers, each with 100 nodes. The neurons in the hidden layers apply transformations to the inputs before passing them forward to the next layers. As the network is being trained, the weights get updated in order to remove the overall error, making the model to become more predictive.

The activation function that will be used in these first 3 layers is the Rectified Linear Unit (*ReLU*). *ReLU* allows only positive values to pass through it. The negative values are mapped to zero (Gupta, 2017). *ReLU* performs normalization on the data by introducing non-linearity to the model, helping it to learn non-linear boundaries in order to find better correlation between the feature timings.




$$RELU(x) = \begin{cases} 0 \; if \; x < 0 \\ x \; if \; x >= 0 \end{cases}$$

*Figure 4 2: ReLU activation function*

The output layer is the layer that gives the predictions made by the model during training. It will have 51 nodes, representing the number of users who by their respective feature timings are predicted. The activation function used in the hidden layer is the *softmax* function since the problem being solved is a multiclass classification problem. A multiclass classification problem is one in which there are more than two classes to be predicted (Brownlee, 2016). In this case 51 different classes need to be predicted.

The figure below shows the architecture of the deep learning neural network.

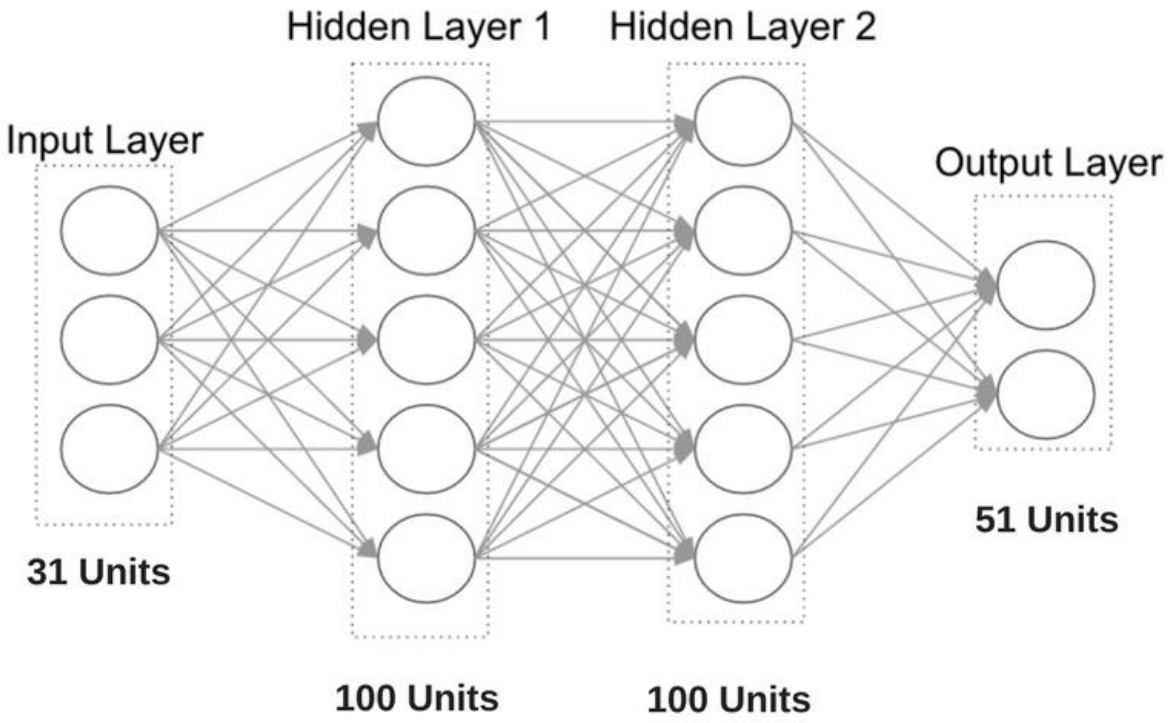



*Figure 4 3: Neural Network Architecture*

## 4.3 Train and Test Sets

Since each of the 51 users has exactly 400 instances of feature timings recorded, the total number of parameters that will be used by the neural network are 20400. If all the data is used for training the neural network, no data will remain for testing purposes. There is the need of keeping some of the data untrained, hence will be split into a training set and testing set. The training set will be used by the model to learn the timing features for each user, and the testing set will be used to evaluate the accuracy of the trained model. Further details on how the model will be evaluated will be explained in the next chapter.

The data will be split into the training and testing sets in the ratio 80:20 respectively. Here the *Scikit-learn* library will come into play, such that the *train_test_split* function will be used to split the data. This will prevent the problem of imbalanced data which involves the classes in the multiclass classification not being represented equally.

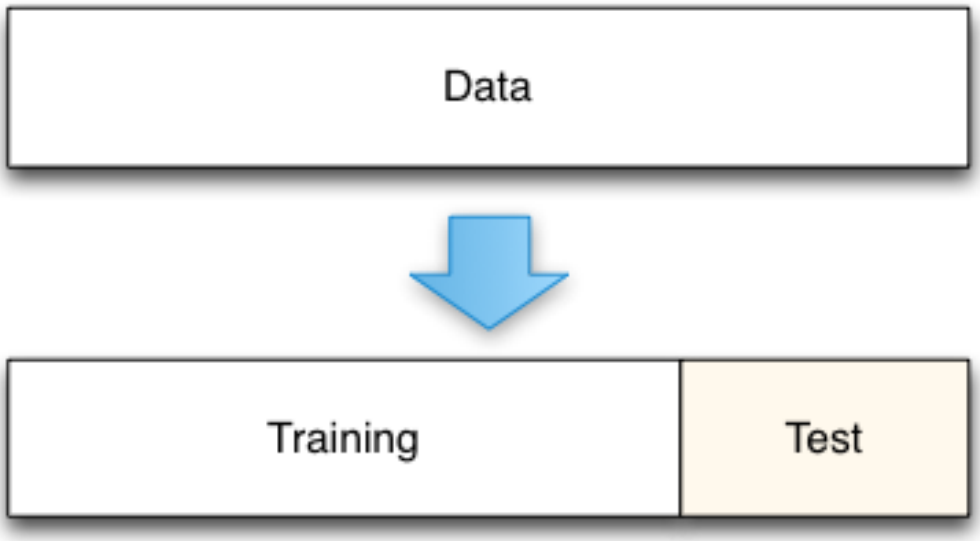



*Figure 4 4: Train-Test Data Split*

## 4.4 Model Development Workflow

There are 5 major steps that will be performed in the modelling of the deep learning network. These steps are as follows:

1. Create Model
2. Configure Model
3. Train Model
4. Evaluate Model
5. Get Predictions

Before developing a neural network, its architecture has to be established beforehand. This includes the number of layers that will define the neural network and the number of neurons in each of the layers. The first step will then now be creating the model. This involves specifying the layers, neurons per layer and the activation functions for each layer. In the configuration step, the model is compiled. This is the step where the loss function, optimizer and metrics are specified. In

the third step, the training data is fit into the model for the learning process to take place. After the model has been trained, its accuracy will be evaluated and then used to make predictions on new data. The new data is the testing dataset that was set aside beforehand and was not involved in the training process of the neural network.

The flow map below shows the sequence of these steps.

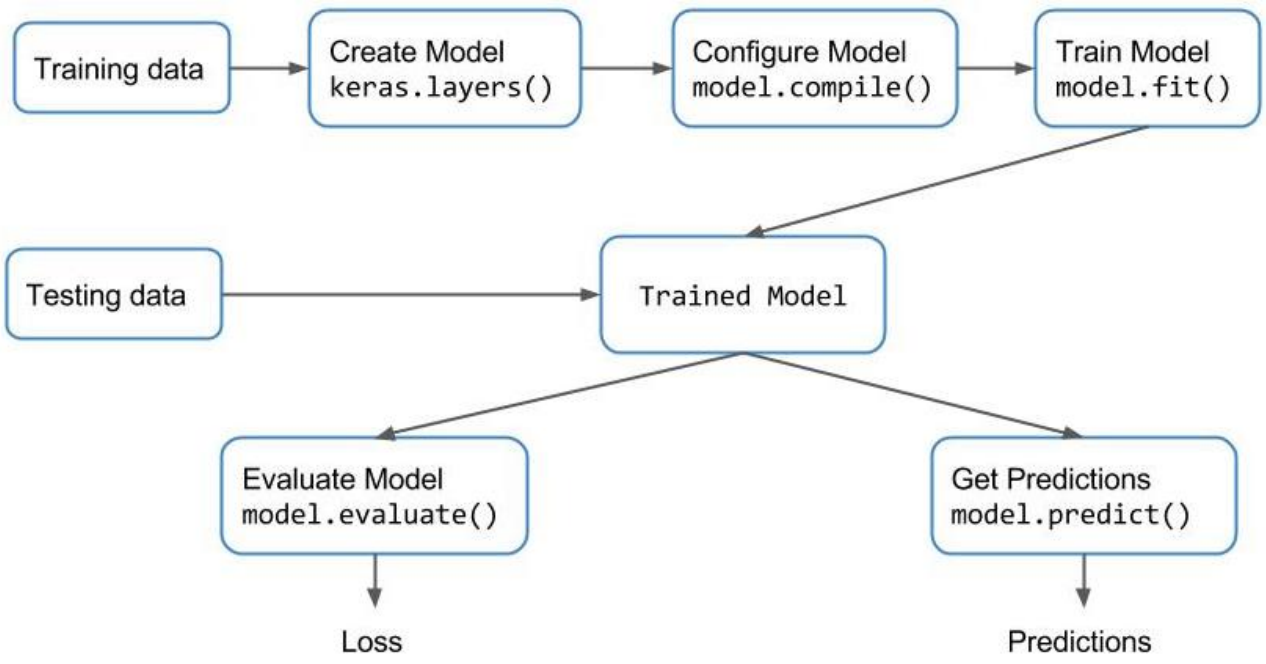



*Figure 4 5: Model Development Workflow*

This chapter demonstrated the architecture of the neural network that will be modelled. It also described the procedure that will be followed in developing the model. In the next chapter, implementation details on how the neural network was developed are explained. The various parameters used are also discussed. Model evaluation and other testing parameters are also presented in detail here.

# Chapter 5: Implementation and Testing

## 5.1 Introduction

The main objective of this dissertation was to develop a suitable machine learning classifier that will be used to validate the legitimacy of a user based on their keystroke timing features. The classifier chosen to solve this problem is a deep learning neural network.

The neural network was developed in using Python 3.6.4 in the Jupyter Notebook environment. Jupyter Notebook is a web application that allow users to create and share documents (notebooks) that contain live code, equations, visualizations as well as text. It was suited for this project because of the mix of code, figures, links and text.

## 5.2 Implementation

### 5.2.1 Development Environment

Algorithm development was done on a Windows 10 operating system, running on a core i7-7700 HQ processor, with 8 GB of RAM.

The Anaconda Distribution version 5.1 (Anaconda, 2018) was used to install Python 3.6.4. Anaconda is a high performance distribution that comes with not only Python, but several relevant Python packages pre-installed. It also allows for easy installation and management of packages, dependencies and environments. The Jupyter Notebook environment was installed in the Anaconda Prompt to facilitate coding of the neural network.

Some of the important packages that were used in the algorithm development are:

1. ***Keras*** – for building the neural network.
2. ***TensorFlow*** – a deep learning library for which the complex computations from Keras are done.

3. ***Scikit-learn*** – a package that easily helps in developing machine learning algorithms.
4. ***Matplotlib*** and ***Seaborn*** – for visualization in order to diagrammatically present the performance of the neural network.
5. ***Numpy*** – for numerical computations in the algorithm.
6. ***Pandas*** – the data manipulation library that was used to load our data, which is in CSV format into Python.
7. ***Time*** – for determining the training duration of our model.

It is important to note that not all the above packages were imported in their entirety. For example, only specific functions in the Keras package were imported, such as ***Sequential*** from ***keras.models*** for creating the model. The reason behind this is that importing the whole package imports everything contained in it, resulting in decrease in execution performance.

### 5.2.2 Loading the dataset

After all the necessary packages were installed, the next step was using the Pandas library to upload the keystroke dataset CSV file into the python environment. Here the data is stored in a two-dimensional tabular data structure called a dataframe. The figure below shows a sample of the first 10 rows of the data as stored in the dataframe.

| | subject | sessionIndex | rep | H.period | DD.period.t | UD.period.t | H.t | DD.t.i | UD.t.i | H.i | ... | H.a | DD.a.n | UD.a.n | H.n | DD.n.l | UD.n.l | H.l | DD.l. |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | s002 | 1 | 1 | 0.1491 | 0.3979 | 0.2488 | 0.1069 | 0.1674 | 0.0605 | 0.1169 | ... | 0.1349 | 0.1484 | 0.0135 | 0.0932 | 0.3515 | 0.2583 | 0.1338 | |
| **1** | s002 | 1 | 2 | 0.1111 | 0.3451 | 0.2340 | 0.0694 | 0.1283 | 0.0589 | 0.0908 | ... | 0.1412 | 0.2558 | 0.1146 | 0.1146 | 0.2642 | 0.1496 | 0.0839 | |
| **2** | s002 | 1 | 3 | 0.1328 | 0.2072 | 0.0744 | 0.0731 | 0.1291 | 0.0560 | 0.0821 | ... | 0.1621 | 0.2332 | 0.0711 | 0.1172 | 0.2705 | 0.1533 | 0.1085 | |
| **3** | s002 | 1 | 4 | 0.1291 | 0.2515 | 0.1224 | 0.1059 | 0.2495 | 0.1436 | 0.1040 | ... | 0.1457 | 0.1629 | 0.0172 | 0.0866 | 0.2341 | 0.1475 | 0.0845 | |
| **4** | s002 | 1 | 5 | 0.1249 | 0.2317 | 0.1068 | 0.0895 | 0.1676 | 0.0781 | 0.0903 | ... | 0.1312 | 0.1582 | 0.0270 | 0.0884 | 0.2517 | 0.1633 | 0.0903 | |
| **5** | s002 | 1 | 6 | 0.1394 | 0.2343 | 0.0949 | 0.0813 | 0.1299 | 0.0486 | 0.0744 | ... | 0.1272 | 0.1534 | 0.0262 | 0.0858 | 0.2528 | 0.1670 | 0.0792 | |
| **6** | s002 | 1 | 7 | 0.1064 | 0.2069 | 0.1005 | 0.0866 | 0.1368 | 0.0502 | 0.0800 | ... | 0.1318 | 0.1204 | -0.0114 | 0.0782 | 0.1999 | 0.1217 | 0.0879 | |
| **7** | s002 | 1 | 8 | 0.0929 | 0.1810 | 0.0881 | 0.0818 | 0.1378 | 0.0560 | 0.0747 | ... | 0.1322 | 0.1040 | -0.0282 | 0.0821 | 0.2127 | 0.1306 | 0.1006 | |
| **8** | s002 | 1 | 9 | 0.0966 | 0.1797 | 0.0831 | 0.0771 | 0.1296 | 0.0525 | 0.0839 | ... | 0.1262 | 0.1403 | 0.0141 | 0.0787 | 0.2138 | 0.1351 | 0.0882 | |
| **9** | s002 | 1 | 10 | 0.1093 | 0.1807 | 0.0714 | 0.0731 | 0.1457 | 0.0726 | 0.0766 | ... | 0.1463 | 0.1162 | -0.0301 | 0.1207 | 0.2281 | 0.1074 | 0.1204 | |

10 rows × 34 columns

*Figure 5 1: Sample Data*

### 5.2.3 Data Exploration

Basic data exploration was done in order to understand the data. The following observations were made:

1. The shape of the dataframe has the dimensions 20400 by 34, meaning 34 columns containing 20400 rows of keystroke timing features for the users.
2. All the rows of data are numerical data types (float64) containing no null values. This is important because the neural network being built to solve the problem is not suited for non-numerical values. Null values in the data will also have an effect in the training performance of the model. See Appendix 1.
3. All the data is stored in memory, with the memory usage around 5.3 MB.
4. Although the subject column also has 20400 instances of values, the unique number of subjects are 51, labeled s002 to s054. See Appendix 2.

### 5.2.4 Feature selection

In this next step, the subset of relevant features that was used for developing the model was selected. The importance of this step is to feed the model only the data that is relevant for it, hence we isolated only those variables that were necessary for analysis. The drop function was used to remove the *sessionIndex* and *rep* columns since the session in which the user typed the password and the number of repetitions do not have an effect of determining the user. Neural networks are intelligent enough to perform feature selection in data (Kordos, 2017) but the effect would be an increase in training time.

### 5.2.5 Encoding the target class

Here the target class, which is the *subject* column was converted into a format with separate columns for each output for classification to take place. So with the target class containing 51

unique users, encoding resulted into an array containing 51 columns for each feature. This is a prerequisite for any multiclass classification problem for predictions to be accurately made.

### 5.2.6 Train-test split

Here the *Scikit-learn* library was used to split the data and separate the training set from the testing set. This was done by specifying the size of the test data to be 20%. The remaining 80% was used as the training data.

After the above data preparation stages, model creation was the next step.

### 5.2.7 Create Model

In Keras, neural networks are defined as a sequence of layers. The first step was creating an instance of the Sequential class, to which the layers were stacked up in the exact order in which they would be connected. As mentioned above, the first layer was be defined as the number of inputs expected to be fed into the neural network. This led to 31 neurons being added to the first layer. The next two layers to be added were the hidden layers. The input and hidden layers each used the *ReLU* activation function. The hidden layers contain 100 neurons each. The output layer used the *softmax* activation function since this is a multiclass classification problem, and the expected output will be the classes representing the users/subjects, who are 51. In summary:

| **Layer** | **Number of Neurons/Units** | **Activation Function** |
|---|---|---|
| Input Layer | 31 | ReLU |
| Hidden Layer 1 | 100 | ReLU |
| Hidden Layer 2 | 100 | ReLU |
| Output Layer | 51 | Softmax |

*Table 5 1 Neural Network Layers Parameters*

### 5.2.8 Configure Model

After creating the layers of the model the next step was to configure it. This was done using the *compile()* function. In this step the optimizer, loss type and metrics were configured. These configurations affect the overall performance of the neural network.

There are several good optimizers that can be used when compiling a neural network model. The most commonly used optimizers are stochastic gradient descent (SGD), RMSProp and Adam (Keras Team, 2018).The optimizer controls the learning rate of the model and therefore can determine the difference between good results in minutes or hours. I opted for the Adam optimizer because it is very effective in achieving good results fast (Kingma & Ba, 2015).

The type of predictive problem determines the type of loss function that will be used. The loss type selected in the configuration step for out model was *categorical_crossentropy* because this is a classification problem. Loss is the difference between the output of the model (that is, the model prediction) and the expected output, whereas accuracy of the model is defined as the number of correct predictions divided by the total number of predictions (Hellström, 2017).

### 5.2.9 Train Model

After configurations have been made the network is ready to be trained. This is done using the *fit()* function. Here the data that was set for training the model was fed into the network by specifying it, both as a matrix of input patterns *x* and an array of matching output patterns *y*. The testing data was also specified here so as to differentiate it from the training data, but was used for validation later in the next step. A parameter will be specified as well to monitor the progress of the model as it is being trained. The time in which the model is trained was also be recorded. I set up a *checkpointer* to monitor the training progress of the model up until it accuracy could not increase further. The best weights after the training were captured and saved.

After the model has been fully trained, the *summary()* function was used to view the summary of the model, shown below.

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_1 (Dense)              (None, 100)               3200
_________________________________________________________________
dense_2 (Dense)              (None, 100)               10100
_________________________________________________________________
dense_3 (Dense)              (None, 51)                5151
=================================================================
Total params: 18,451
Trainable params: 18,451
Non-trainable params: 0
_________________________________________________________________
```

*Figure 5 2: Model Accuracy*

## 5.3 Testing

### 5.3.1 Evaluate Model

After a neural network has been trained, it can be evaluated on the training data, but this will not provide a useful indication of its performance as a predictive model, since it has already seen all the data before. The performance of the model is therefore evaluated by using a separate dataset that was not seen by the model during training. This will provide an estimate of the predictive performance of the network for unseen data in future.

Generally the lower loss means that the model is performing very well, and the higher the accuracy the better.

The neural network model was set up to evaluate the loss and classification accuracy across the testing dataset that had been initially set aside for this reason.

The graphs below show the performance of the developed neural network model, both during training and validation.

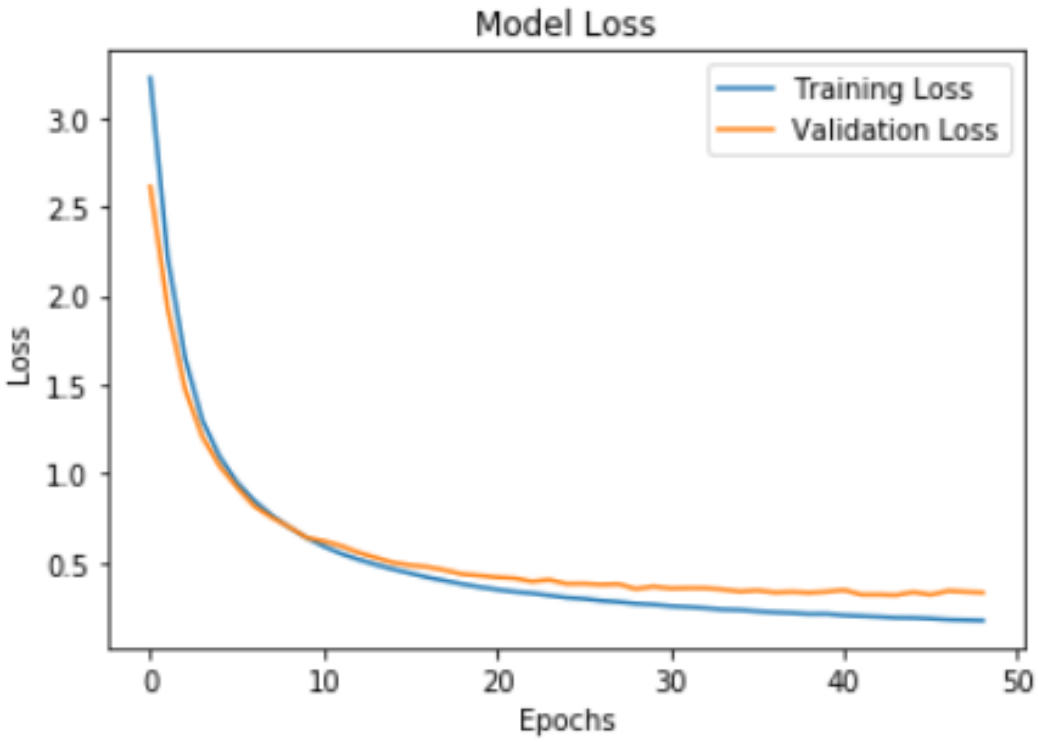



*Figure 5 3: Model Loss*



The above graph shows the gradual performance of the model loss as it was being trained and validated. Both training and validation loss decrease as the number of epochs increases. It also shows that more information is learned in each epoch, up until the checkpoint monitor observes that no further improvement can be made. A similar graph on model accuracy is shown below.

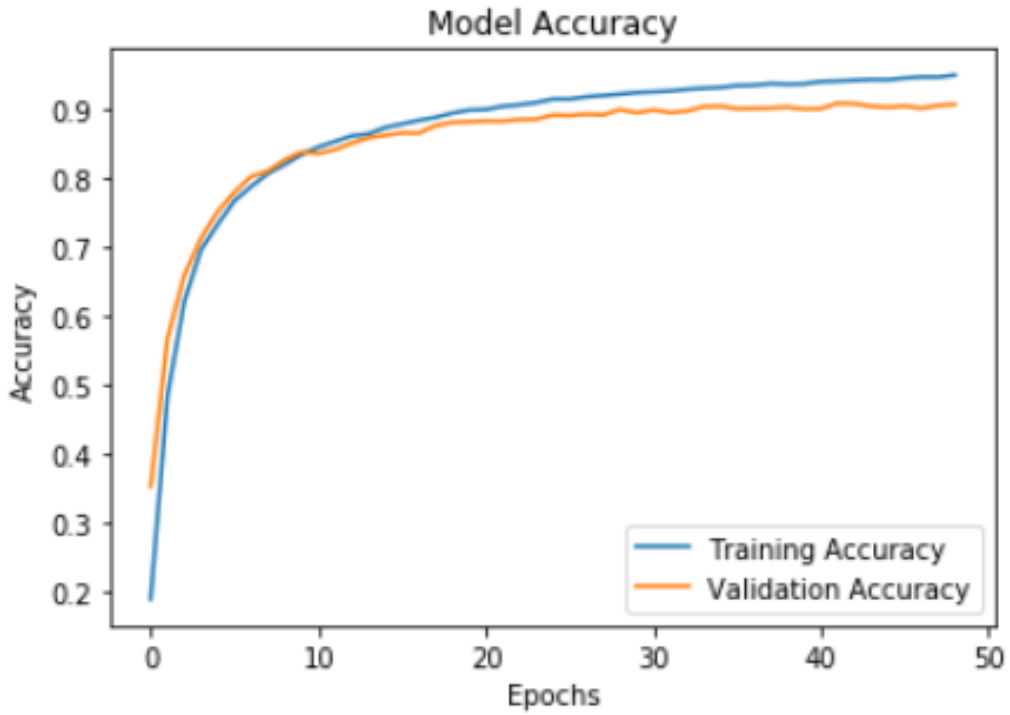



*Figure 5 4: Model Accuracy*

The graph accuracy of the model in training and validation increases gradually with an increase in the number of epochs.

Our trained neural network managed to register a model loss of 0.32 and an accuracy of 90.34%. It also took 45.11 seconds to train.

### 5.3.2 Get Predictions

After evaluating how accurate the model is, it was then used to make predictions for the labels of the test sets. This was done by using the *predict()* function. The testing data set containing timing features was passed to the trained model and the model predicted the respective users to which the timing features belong to. The accuracy of the predictions made were according to the metrics in the evaluation step.

This chapter described the implementation stages that were followed in the development of the deep learning neural network. It also described the testing that was performed to validate the accuracy of the model. Results and other findings will be discussed in the next chapter.

# Chapter 6: Discussion of Results

## 6.1 Visualizing Predictive Performance

In order to view how well our model was in predicting new data, a confusion matrix was used. A confusion matrix is a matrix/table that can be used to measure the performance of a supervised machine learning algorithm. A row of the confusion matrix represents an instance of the actual class, whereas the columns represent instances of the predicted classes. As the name suggests, a confusion matrix makes it easy to understand the kind of confusions that occur in the classification algorithms during prediction. The figure below shows a confusion matrix.

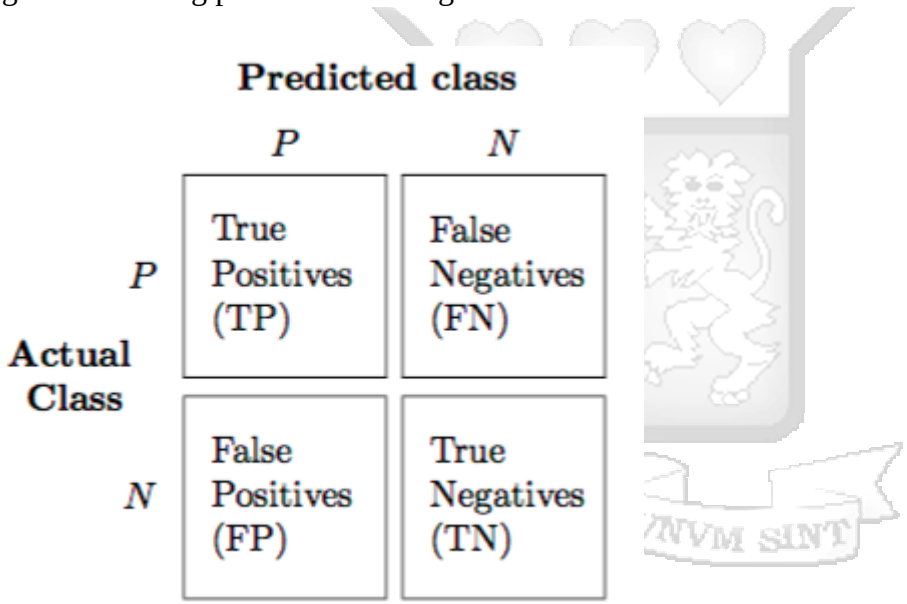



*Figure 5 5: Confusion Matrix*

The following terms are used in the confusion matrix:

1. **Positive (P)** – the observation is positive.
2. **Negative (N)** – the observation is negative.
3. **True Positives (TP)** – these are the cases in which the actual class is positive, and is predicted to be positive

4. **False Negative (FN)** – these are the cases where the observation is positive but is predicted as negative instead.
5. **False Positives (FP)** – in these cases, the observation is negative, but is predicted to be positive.
6. **True Negatives (TN)** – these are the cases where the observation is negative, and is predicted to be negative (GeeeksforGeeks, 2018).

The classification accuracy of the model is calculated as follows:

*Accuracy = (TP + TN) / (TP + TN + FP + FN)*

Confusion matrices are ideally used in binary classification and multiclass classification tasks of machine learning algorithms.

A confusion matrix was generated in order to visually illustrate the predictive capability of the developed neural network model. Two confusion matrices were computed and generated: a normalized confusion matrix and an un-normalized confusion matrix (See Appendix 3). The figure below shows the normalized confusion matrix.

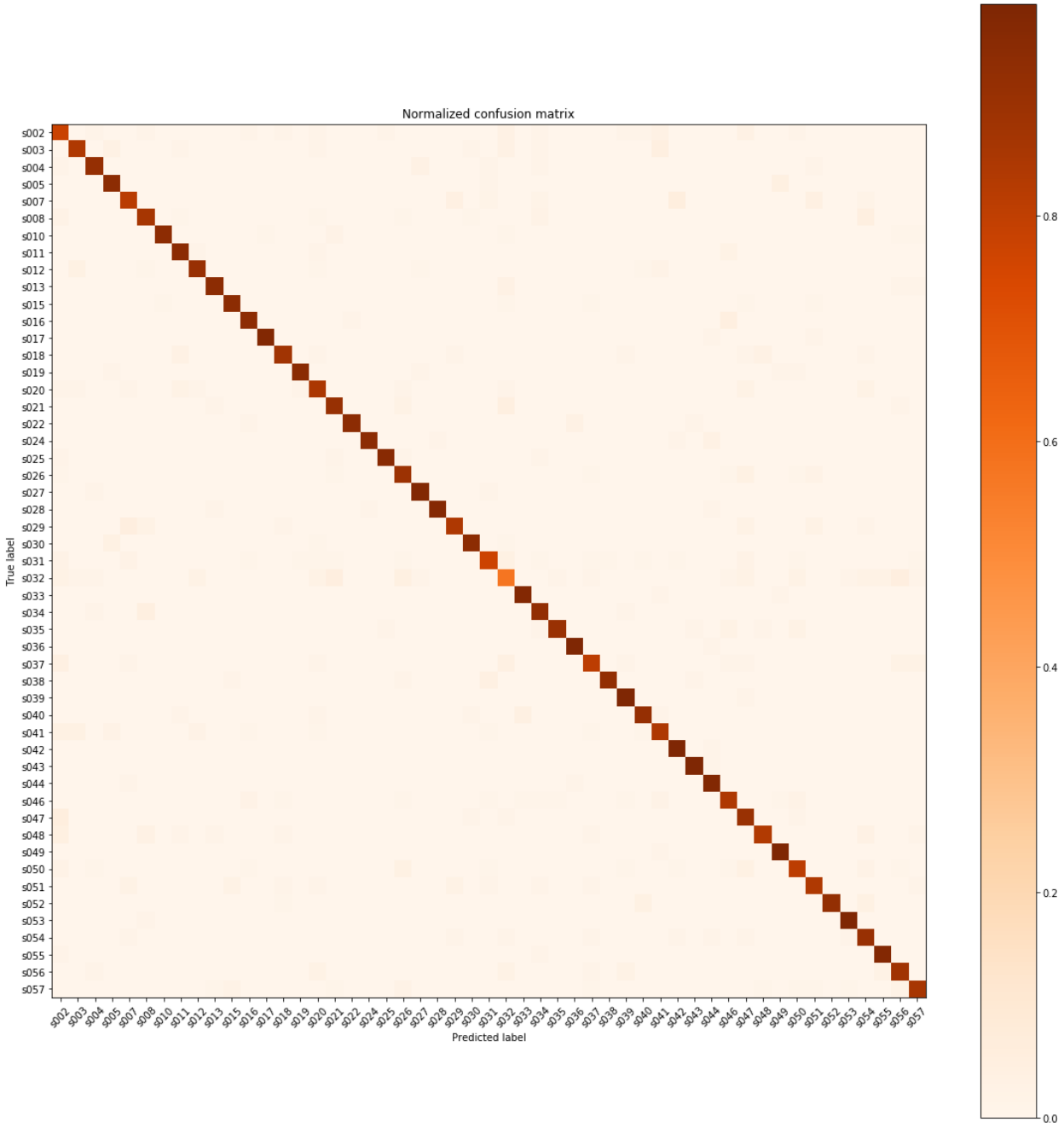



*Figure 5 6: Normalized Confusion Matrix*

The above illustrates the accuracy in which the trained model performed on the test data. All correct predictions are located in the diagonal of the confusion matrix. The darker the oranges the better the model was at predicting a user based on the keystroke timing features from the test data. The results show that the model did a great job in predicting all the respective users based on the

testing dataset that was unseen by the model during training. This also proves that the 90.34% accuracy of the model is a great performance by the model.

As a recap, the objectives of this dissertation were as follows:

1. To understand user authentication approaches and how keystroke dynamics can be used to enhance user authentication and access control.
2. To understand previous research on keystroke dynamics and related classification methods that have been applied in keystroke dynamics.
3. To develop and train a suitable machine learning classifier to recognize users and validate their authenticity during authentication.
4. To test and validate the accuracy of the proposed machine learning classifier in validating user authenticity through keystroke dynamics.

It is therefore fair to conclude that the deep learning network model actually meets the main objective of this dissertation, which was to validate a user based on their keystroke dynamics. The model has solved the multiclass classification problem because it has been able to accurately predict the actual classes of users based on their timing features from the unseen data.

## 6.2 Other Findings

During the implementation phase, several versions of the model were trained. Tweaks with the parameters, such as the changing the number of hidden layers, changing the number of units in each layer, using a different optimizer and increasing the size of the training set. The following observations were made. In each of the following scenarios, the first model prototype A is the final model.

**Scenario 1: Changing the number of hidden layers.**

An assumption was made that increasing the number of hidden layers would improve performance.

A prototype of the model was made with 5 hidden layers, each containing 100 units, resulting in a total of 7 layers for our neural network. The model loss was 0.47 and model accuracy was 89.07%. The model took 70.31 seconds to train. On increasing the number of hidden layers to 10, each still with 100 units, the model took an even longer time to train, 93.98 seconds. The model loss increased to 0.59 whereas its validation accuracy decreased to 85.39%. The table below summarizes the results of the different prototypes.

| **Model Prototype** | **Hidden Layers (100 each)** | **Model Loss** | **Model Accuracy (Percentage)** | **Training Time (Seconds)** |
|---|---|---|---|---|
| **A** | **2** | **0.32** | **90.34** | **45.11** |
| **B** | 5 | 0.47 | 89.07 | 70.31 |
| **C** | 10 | 0.64 | 79.58 | 79.58 |

*Table 6 1: Models with different number of hidden layers*

This proved that increasing the number of hidden layers does not necessarily lead to a corresponding increase in performance, hence settled for 2 hidden layers.

**Scenario 2: Changing the number of units per hidden layer.**

Each of the trained model prototypes had 2 layers. On increasing the number of neurons per hidden layer to 400 each, a loss of 0.33 and an accuracy of 90.93% was recorded. The model trained in 73.38 seconds. On using 20 neurons per hidden layer, the loss was 0.47 and accuracy 86.69%. The model also took 69.97 seconds to train. Hidden layers with 1000 units were also used. The model took an astonishing 250.18 seconds to produce a loss of 0.33 and accuracy of 90.64%.

| Model Prototype | Units per hidden layer (2 layers) | Model Loss | Model Accuracy (Percentage) | Training Time (Seconds) |
|---|---|---|---|---|
| **A** | **100** | **0.32** | **90.34** | **45.11** |
| **E** | 20 | 0.47 | 86.69 | 69.97 |
| **F** | 400 | 0.33 | 90.93 | 73.38 |
| **G** | 1000 | 0.33 | 90.63 | 250.18 |

*Table 6 2: Models with different number of units per hidden layer*

Hence the number of units settled for per hidden layer was 100.

**Scenario 3: Changing the size of the training dataset**

Choosing a suitable size of the testing size is important in any supervised machine learning problem. A large percentage of testing size means the trained model will have more data for which evaluation and testing can be done. However, the less data will be available for training the model. Machine learning algorithms require as much data as possible for training them for them to find better correlations in data in order to make better predictions. It is therefore important to find the right balance between the training and testing datasets.

The table below shows the different performances of models depending on the amount of test data set aside for model validation.

| Model Prototype | Percentage size of testing data set | Model Loss | Model Accuracy (Percentage) | Training Time (Seconds) |
|---|---|---|---|---|
| **A** | **20** | **0.32** | **90.34** | **45.11** |

| | | | | |
|---|---|---|---|---|
| **H** | 25 | 0.36 | 90.01 | 56.34 |
| **I** | 30 | 0.40 | 88.95 | 41.54 |
| **J** | 40 | 0.42 | 88.33 | 44.81 |

*Table 6 3: Models with different test set sizes*

The final model that was used had the training and testing data split in the ratio 80:20.

**Scenario 4: Using Different Optimizers**

The choice of optimizer used also has an overall effect on the performance of the model. The optimizers that were tested for model development in this dissertation were stochastic gradient descent (SGD), RMSProp and Adam. The table below shows performance of prototype models with the different optimizers.

| **Model Prototype** | **Optimizer** | **Model Loss** | **Model Accuracy (Percentage)** | **Training Time (Seconds)** |
|---|---|---|---|---|
| **A** | **Adam** | **0.32** | **90.34** | **45.11** |
| **K** | RMSProp | 0.41 | 89.13 | 42.95 |
| **L** | SGD | 0.72 | 80.76 | 95.81 |

*Table 6 4: Model with different optimizers*

The optimizer that was opted for in the final model was Adam, because it had the best predictive performance, in terms of both loss and accuracy.

In summary, the following parameters were settled for in the algorithm development because they produced the best results after various experimentations:

Train-Test Split ratio: **80:20**

Number of Hidden Layers: **2**

Number of units per hidden layer: **100**

Optimizer: **Adam**

One of the hypotheses made for this dissertation was that the proposed machine learning classifier would have an acceptable accuracy rate for which it will correctly classify users depending on their typing rhythm. This was validated, with the developed deep learning neural network performing at a high rate of 90.34 percent.

The other hypothesis was that this dissertation would provide a strong basis to support the claim that keystroke dynamics as a behavioral biometric security mechanism can be used to improve user authentication by validating a user's legitimacy. This was also validated, because the neural network model developed was able to predict a user's identity based on their keystroke dynamics. This proves that keystroke dynamics, in addition to the traditional username-password combination, can be used as a two-factor biometric security mechanism to enhance security.

# Chapter 7: Conclusions, Recommendations and Future Work

## 7.1 Conclusions and Summary

Keystroke dynamics is the study of how users can be distinguished based on their typing rhythm. When the timing features of their typing rhythms are captured, they can be analysed in order to validate the users' identities.

The main objective of this dissertation was to identify and develop a suitable machine learning classifier that will be able to validate a user's identity based on their keystroke dynamics. A deep learning neural network was modelled to do this. The dataset used was collected by Killourhy and Maxion. The dataset contains keystroke timing features of 51 users, who typed the password (**.tie5Roanl**) 400 times each. The deep learning model was trained with 80% of the data to validate a user's identity based on their unseen keystroke data, which is the remaining 20% used as testing data. The trained model produced a respectable prediction performance of 90.34% during model evaluation. These results prove that that keystroke dynamics can be used as a two-factor biometric security to enhance security during authentication, hence ensuring that the objective of the dissertation was met.

## 7.2 Recommendations

The developed deep learning model, in addition to a keystroke logger, can be implemented in various software applications and systems in which users are required to authenticate themselves through a username-password combination. Software developers can incorporate this model as they develop their applications in order to enhance user security during authentication. When an application user types their username and password, the keystroke logger will capture and record their keystroke timing features. The deep learning model will then capture the keystroke dynamics during the authentication process to validate the user's legitimacy and therefore provide additional security.

## 7.3 Future Work

This dissertation focused on keystroke timings recorded for all users typing the same password several times over different sessions. Future work would involve developing deep learning models that learn from different passwords by different users, because in reality different users do not share the same passwords. This would add complexity to the problem, since different timing features can be recorded for different users, and therefore more research can be done in that area.

# References


ML BOT1. (2017, July 27). *User Verification based on Keystroke Dynamics: Python code*. Retrieved from Machine Learning in Action: https://appliedmachinelearning.wordpress.com/2017/07/26/user-verification-based-on-keystroke-dynamics-python-code/

Anaconda. (2018, February 15). *Download Anaconda Distribution*. Retrieved from Anaconda: https://www.anaconda.com/download/

Awad, M., Al-Qudah, Z., Idwan, S., & Jallad, A. H. (2016). Password security: Password behavior analysis at a small university. *Electronic Devices, Systems and Applications (ICEDSA), 2016 5th International Conference*. IEEE.

Banerjee, S. P., & Woodard, D. L. (2012). Biometric Authentication and Identification using Keystroke Dynamics: A Survey. *Journal of Pattern Recognition Research 7*, 116 - 139.

Barghouthi, H. (2009). *Keystroke Dynamics: How typing characteristics differ from one application to another*. Norway: Gjøvik University College.

Brownlee, J. (2016, June 2). *Multi-Class Classification Tutorial with the Keras Deep Learning Library*. Retrieved from Machine Learning Mastery: https://machinelearningmastery.com/multi-class-classification-tutorial-keras-deep-learning-library/

Chandrika, V. (2014). Ethical Hacking: Types of Ethical Hackers. *International Journal of Emerging Technology in Computer Science & Electronics (IJETCSE)*.

Deng, Y., & Zhong, Y. (2015). *Keystroke Dynamics User Authentication Using Advanced Machine Learning Methods*. Burlington: Science Gate Publishing.

GeeeksforGeeks. (2018). *Confusion Matrix in Machine Learning*. Retrieved from GeeksforGeeks: A computer science portal for geeks: https://www.geeksforgeeks.org/confusion-matrix-machine-learning/

Giroux, S., & Wachowiak-Smolikova, R. (2009). *Keystroke-Based Authentication by Key Press Intervals as a Complementary Behavioral Biometric*. North Bay: IEEE.

GormLey, M. M. (2017, April 26). *How Rapid Application Development Makes Building Apps Easy And Fast*. Retrieved from BuzzFeed: https://www.buzzfeed.com/michaelmg/how-rapid-application-development-makes-building-a-2xd2p?utm_term=.bbxvQmxPz#.hlbDn1Lj4

Gupta, V. (2017, October 9). *Understanding Feedforward Neural Networks*. Retrieved from Learn OpenCV: https://www.learnopencv.com/understanding-feedforward-neural-networks/

Hellström, E. (2017). *Feature learning with deep neural networks for keystroke biometrics: A study of supervised pre-training and autoencoders.* Luleå, Sweden: Luleå University of Technology.

Hubbell, T. (2017, March 28). *Top 4 Software Development Methodologies*. Retrieved from Black Duck Softeare: https://blog.blackducksoftware.com/top-4-software-development-methodologies

Jupyter. (2018, March 16). *Project Jupyter*. Retrieved from Jupyter: http://jupyter.org/

Karnan, M., & Akila, M. (2009). Identity authentication based on keystroke dynamics using genetic algorithm and particle Swarm Optimization. *2nd IEEE International Conference on Computer Science and Information Technology*. Beijing: IEEE.

Keras. (2018). *The Sequential model API*. Retrieved from Keras: https://keras.io/models/sequential/

Keras Team. (2018). *Keras: The Python Deep Learning library*. Retrieved from Keras: https://keras.io/

Killourhy, K. S., & Maxion, R. A. (2009). Comparing Anomaly-Detection Algorithms for Keystroke Dynamics. *Dependable Systems & Networks, 2009. DSN '09. IEEE/IFIP International Conference*. Lisbon: IEEE.

Kingma, D. P., & Ba, J. L. (2015). Adam: A Method For Stochastic Optimization. *3rd International Conference for Learning Representations.* San Diego: Cornell University.

Kordos, M. (2017). *Data Selection for Neural Networks.* Bielsko-Biala, Poland: Department of Computer Science and Automatics, University of Bielsko-Biala.

Korkishko, I. (2017, October 11). *Top 6 software development methodologies*. Retrieved from DEV.TO(GETHER): https://dev.to/iriskatastic/top-6-software-development-methodologies-9b

Malinka, K. (2009). Usability of Visual Evoked Potentials as Behavioral Characteristics for Biometric Authentication. *Internet Monitoring and Protection, 2009. ICIMP '09. Fourth International Conference.* Venice, Italy: IEEE.

Miniwatts Marketting Group. (2018, February 21). *Usage and Population Statistics*. Retrieved from Internet World Stats: https://www.internetworldstats.com/emarketing.htm

Monrose, F., & Rubin, A. (2000). Authentication via Keystroke Dynamics. *Future Generation Computer Systems, vol. 16(4)*, 351-359.

Mwagwabi, F., McGill, T., & Dixon, M. (2014). Improving Compliance with Password Guidelines: How User Perceptions of Passwords and Security Threats Affect Compliance with Guidelines. *System Sciences (HICSS), 2014 47th Hawaii International Conference.* Waikoloa, HI, USA: IEEE.

Patil, R. A., & Renke, A. L. (2016). Keystroke Dynamics for User Authentication and Identification by using Typing Rhythm. *International Journal of Computer Applications (0975 – 8887),Volume 144 – No.9*.

Piatetsky, G. (2017, September). *Python vs R – Who Is Really Ahead in Data Science, Machine Learning?* Retrieved from KDnuggets: https://www.kdnuggets.com/2017/09/python-vs-r-data-science-machine-learning.html

Portilla, J. (2017, March 21). *A Beginner's Guide to Neural Networks in Python and SciKit Learn 0.18*. Retrieved from Springboard: https://www.springboard.com/blog/beginners-guide-neural-network-in-python-scikit-learn-0-18/

Rathanavel, V., & Mali, S. (2017). Graphical Password as an OTP. *International Journal Of Engineering And Computer Science ISSN: 2319-7242*.

Scotton, M., & Alexander, J. (2016, March 17). *Secure Your Salesforce Org with Two-Factor Authentication*. Retrieved from LinkedIn SlideShare: https://www.slideshare.net/awesomeadmin/secure-your-salesforce-org-with-twofactor-authentication

Senk, C., & Dotzler, F. (2011). Biometric authentication as a service for enterprise identity management deployment: a data protection perspective. *Availability, Reliability and Security (ARES), 2011 Sixth International Conference.* Vienna: IEEE.

Sim, T., & Janakiraman, R. (2007). Are Digraphs Good for Free-Text Keystroke Dynamics? *Computer Vision and Pattern Recognition, 2007. CVPR '07.* Minneapolis: IEEE.

Singh, P. I., & Thakur, G. S. (2012). Enhanced Password Based Security System Based on User Behavior using Neural Networks. *I.J. Information Engineering and Electronic Business*, 29-35.

Stiner, S. (2016, August 24). *Rapid Application Development (RAD): A Smart, Quick And Valuable Process For Software Developers*. Retrieved from Forbes: https://www.forbes.com/sites/forbestechcouncil/2016/08/24/rapid-application-development-rad-a-smart-quick-and-valuable-process-for-software-developers/#32a58e6e19e8

Stweart, J., Monaco, J., Cha, S.-H., & Tappert, C. (2011). An investigation of keystroke and stylometry traits for authenticating online test takers. *Biometrics (IJCB), 2011 International Joint Conference* . Washington DC: IEEE.

Suman, K. N. (2017). *Keystroke Dynamics - Typing patterns for keystroke authentication*. Retrieved from Kaggle: https://www.kaggle.com/knityansuman/keystroke-dynamics

Teh, P. S., Teoh, A. B., & Yue, S. (2013). A Survey of Keystroke Dynamics Biometrics. *The Scientific World Journal, Volume 2013, Article ID 408280*, 24.

TensorFlow. (2018). *An open source machine learning framework for everyone*. Retrieved from TensorFlow: https://www.tensorflow.org/

Vinayak, R., & Arora, K. (2015). A Survey of User Authentication using Keystroke Dynamics. *International Journal of Scientific Research Engineering & Technology (IJSRET)*, 378 - 384.

Wankhede, S. B., & Verma, S. (2014). Keystroke Dynamics Authentication System Using Neural Network. *International Journal of Innovative Research & Development*.

Willems, K. (2017, May 2). *Keras Tutorial: Deep Learning in Python*. Retrieved from DataCamp: https://www.datacamp.com/community/tutorials/deep-learning-python

VT OMNES
VNVM SINT

# Appendices

**Appendix 1: Information about the dataframe**

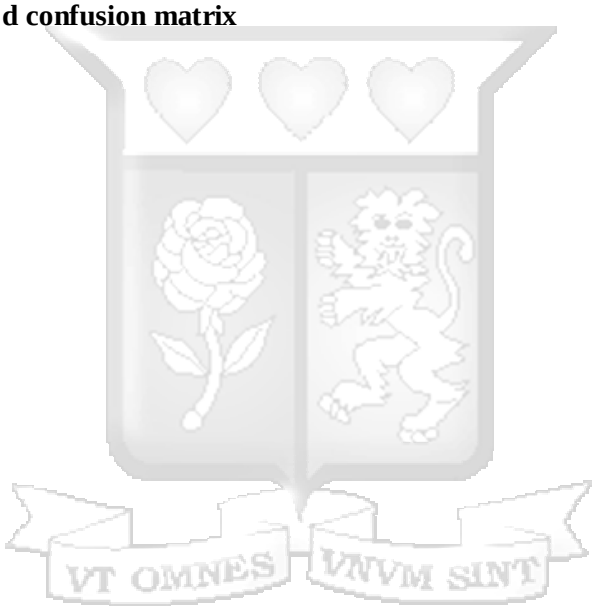
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 20400 entries, 0 to 20399
Data columns (total 34 columns):
subject             20400 non-null object
sessionIndex        20400 non-null int64
rep                 20400 non-null int64
H.period            20400 non-null float64
DD.period.t         20400 non-null float64
UD.period.t         20400 non-null float64
H.t                 20400 non-null float64
DD.t.i              20400 non-null float64
UD.t.i              20400 non-null float64
H.i                 20400 non-null float64
DD.i.e              20400 non-null float64
UD.i.e              20400 non-null float64
H.e                 20400 non-null float64
DD.e.five           20400 non-null float64
UD.e.five           20400 non-null float64
H.five              20400 non-null float64
DD.five.Shift.r     20400 non-null float64
UD.five.Shift.r     20400 non-null float64
H.Shift.r           20400 non-null float64
DD.Shift.r.o        20400 non-null float64
UD.Shift.r.o        20400 non-null float64
H.o                 20400 non-null float64
DD.o.a              20400 non-null float64
UD.o.a              20400 non-null float64
H.a                 20400 non-null float64
DD.a.n              20400 non-null float64
UD.a.n              20400 non-null float64
H.n                 20400 non-null float64
DD.n.l              20400 non-null float64
UD.n.l              20400 non-null float64
H.l                 20400 non-null float64
DD.l.Return         20400 non-null float64
UD.l.Return         20400 non-null float64
H.Return            20400 non-null float64
dtypes: float64(31), int64(2), object(1)
memory usage: 5.3+ MB
```

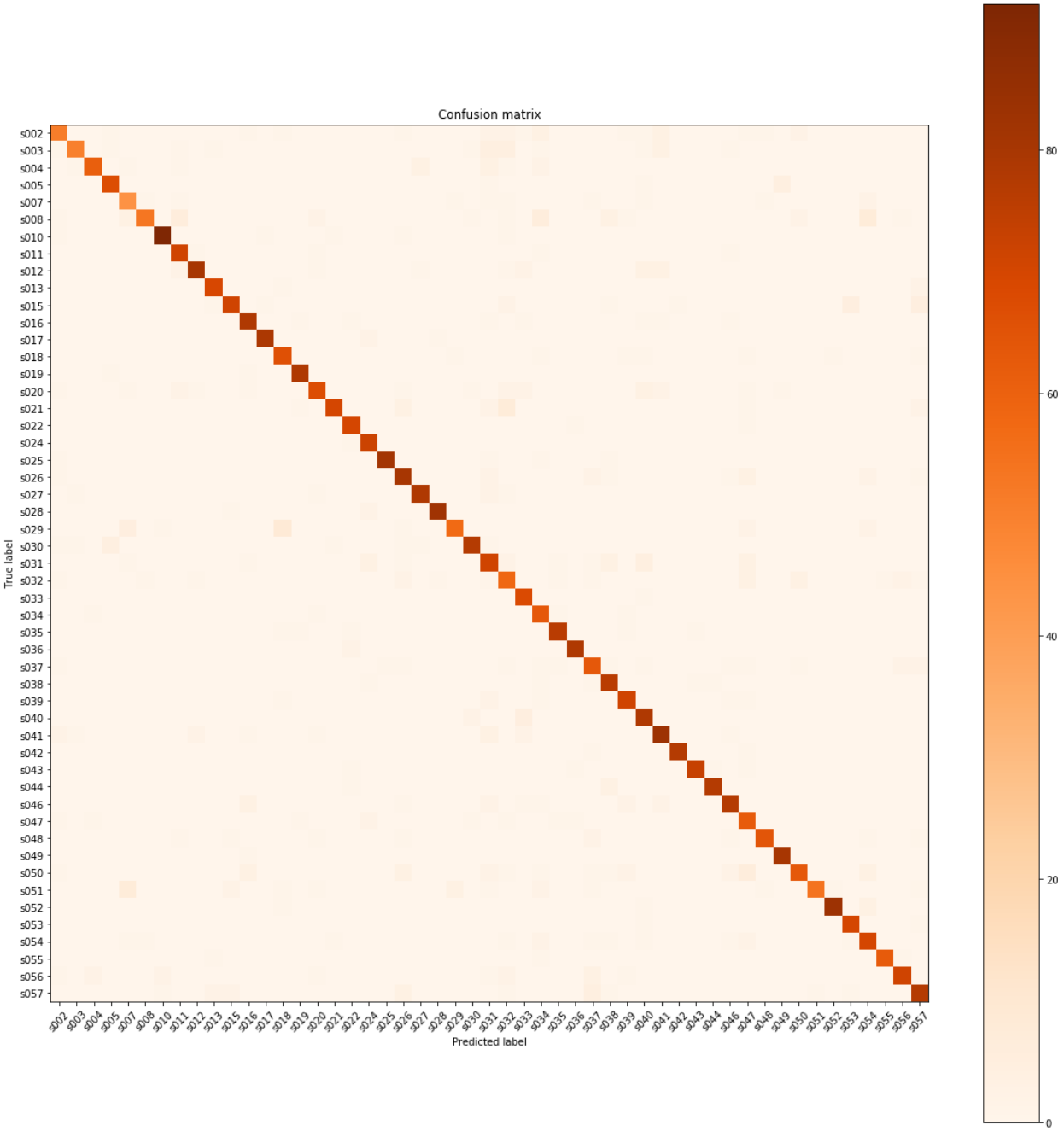
**Appendix 2: Number of unique users in the dataset.**

```
In [5]: #number of unique classes of users
        keystroke_data['subject'].unique()

Out[5]: array(['s002', 's003', 's004', 's005', 's007', 's008', 's010', 's011',
               's012', 's013', 's015', 's016', 's017', 's018', 's019', 's020',
               's021', 's022', 's024', 's025', 's026', 's027', 's028', 's029',
               's030', 's031', 's032', 's033', 's034', 's035', 's036', 's037',
               's038', 's039', 's040', 's041', 's042', 's043', 's044', 's046',
               's047', 's048', 's049', 's050', 's051', 's052', 's053', 's054',
               's055', 's056', 's057'], dtype=object)
```

**Appendix 3: Un-normalized confusion matrix**




Confusion matrix
True label
Predicted label
s002 s003 s004 s005 s007 s008 s010 s011 s012 s013 s015 s016 s017 s018 s019 s020 s021 s022 s024 s025 s026 s027 s028 s029 s030 s031 s032 s033 s034 s035 s036 s037 s038 s039 s040 s041 s042 s043 s044 s046 s047 s048 s049 s050 s051 s052 s053 s054 s055 s056 s057
0
20
40
60
80