

University of Sheffield

# Using Machine Learning of User Keystrokes to Identify Hijacked Sessions



Kunal Das

*Supervisor:* Andrew Stratton

A report submitted in fulfilment of the requirements  
for the degree of MSc in Cybersecurity and AI

*in the*

Department of Computer Science

September 14, 2022

## Declaration

All sentences or passages quoted in this report from other people's work have been specifically acknowledged by clear cross-referencing to author, work and page(s). Any illustrations that are not the work of the author of this report have been used with the explicit permission of the originator and are specifically acknowledged. I understand that failure to do this amounts to plagiarism and will be considered grounds for failure in this project and the degree examination as a whole.

Name: Kunal Das

---

Signature: Kunal Das

---

Date: 14/09/2022

---

## Abstract

Security is one of the most important concerns of modern society. As a result of concerns about their data, individuals and companies are striving to improve authentication techniques. Secure passwords are adopted by modern society, and two-factor authentication is implemented by most businesses. What is the next step? By implementing continuous authentication, these two problems can be addressed from both sides. There is no need for users to be concerned about their active login session. By securing their services more widely and protecting themselves from automated attacks, companies can increase the level of security for their services.

The purpose of the project is to develop an effective application for detecting a non-legitimate user acting like a genuine user to prevent this kind of spoofing attack automatically. While a user is typing the system. Aim to obtain some new information rather than merely reiterating previous findings. An authentication method based on password typing keystroke dynamics has been previously developed. Based on the results, the system will decide whether or not to make the session active. Here is the goal to capture writing data and validate that for the user.

As a result, future versions of this software might be able to authenticate any device with the use of other features. A web application can include mouse dynamics along with keystroke dynamics. In this work, user identification is the main objective that has been achieved by writing patterns. In addition, users are now able to leave their work whenever they wish, and when another user begins to use the system the session will be closed by the system. A shell script authentication system will also be available in the near future. By implementing this methodology, server attacks can be reduced. Active sessions can save users time and effort by reducing the need to enter passwords or use 2FA every time.

# Contents

<b>1</b>	<b>Introduction</b>	<b>1</b>
1.1	Aims and Objectives . . . . .	1
1.2	Overview of the Report . . . . .	2
<b>2</b>	<b>Literature Survey and Background</b>	<b>4</b>
2.1	Authentication . . . . .	4
2.1.1	Password Authentication . . . . .	5
2.1.2	Two-Factor Authentication . . . . .	5
2.1.3	Biometric Authentication . . . . .	6
2.1.4	Token Authentication . . . . .	7
2.1.5	CAPTCHA . . . . .	7
2.2	Languages . . . . .	8
2.2.1	HTML . . . . .	8
2.2.2	CSS . . . . .	8
2.2.3	JavaScript . . . . .	9
2.2.4	Python . . . . .	9
2.3	Use of Artificial Intelligence (AI) in Real Time Application . . . . .	9
2.3.1	Navigation System . . . . .	10
2.3.2	Language Processing Applications . . . . .	10
2.4	Machine Learning . . . . .	10
2.4.1	Supervised Learning . . . . .	11
2.4.2	K-Nearest Neighbors . . . . .	12
2.5	Continues Authentication . . . . .	15
2.5.1	Keystroke Dynamics . . . . .	16
<b>3</b>	<b>Implementation and Analysis</b>	<b>20</b>
3.1	Design . . . . .	20
3.1.1	Front-end . . . . .	21
3.1.2	Back-end . . . . .	22

<b>4</b>	<b>Results</b>	<b>36</b>
4.1	Performance Measure of KNN . . . . .	36
4.1.1	Keystroke Patterns . . . . .	37
4.1.2	Confusion Matrix . . . . .	38
<b>5</b>	<b>Conclusions and Future Work</b>	<b>42</b>

# List of Figures

2.1	Simple Authentication . . . . .	4
2.2	Password Authentication . . . . .	5
2.3	Two-Factor Authentication . . . . .	6
2.4	Biometric Authentication . . . . .	6
2.5	Token Authentication . . . . .	7
2.6	CAPTHA . . . . .	8
2.7	Supervised Learning for label based training . . . . .	11
2.8	Regression . . . . .	12
2.9	K-Nearest Neighbors . . . . .	14
2.10	K-Nearest Neighbors with New Instance . . . . .	14
2.11	Keystroke Authentication Process . . . . .	17
3.1	Registration Page . . . . .	21
3.2	Login Page . . . . .	21
3.3	Home Page . . . . .	22
3.4	Logout Page . . . . .	22
3.5	Secret Key . . . . .	22
3.6	Session Key . . . . .	23
3.7	Registration Back-end . . . . .	24
3.8	Secuirty Function . . . . .	25
3.9	Authentication Function . . . . .	26
3.10	Key Example . . . . .	27
3.11	Keystroke Collection Process . . . . .	28
3.12	Feature Extraction . . . . .	30
3.13	Feature Extraction . . . . .	30
3.14	Data Flow Diagram . . . . .	31
3.15	Data Collect . . . . .	32
3.16	Normalizaton . . . . .	33
3.17	Training Model . . . . .	34
3.18	Prediction Model . . . . .	35
4.1	Keystroke Patterns of two user . . . . .	38

4.2	Genuine Matrix . . . . .	39
4.3	Imposter Matrix . . . . .	39
4.4	JSON Data Structure . . . . .	40
4.5	Authentication Writing UI . . . . .	41

# List of Tables

2.1	Spam and Ham Data Sample . . . . .	11
2.2	Used Car Data-Set . . . . .	13
2.3	User Typing Characteristics . . . . .	16
2.4	Classifier Comparison with 2009 Databse . . . . .	19
3.1	Keystroke Dataset . . . . .	29
4.1	Genuin User Data . . . . .	36
4.2	Imposter Data . . . . .	37



# Chapter 1

## Introduction

Nowadays, people are increasingly dependent on technology. The physical card people use is also not fully identified that this is the genuine user. In every application from an online store to banking or social media. The Internet is a necessity for most people. People started storing their data on online cloud storage (Siddiqui et al., 2021). Hackers can also gain access to local WiFi networks. Passwords and accounts can be hacked by external or Internal hackers (Killourhy and Maxion, 2009). The companies implemented authentication methods in order to ensure that the network and web-based applications were secure. In order to implement biometric features, it is very expensive. A further disadvantage of biometrics is that it does not have the capability of continuously authenticating a person. A building or other location can be accessed by anyone by using another person's key card, just as in movies. It can be a serious problem if someone gets access to someone else's account without their consent. There is also the possibility that people sometimes lose their phones and then someone else is able to use them. In situations like these, it would be helpful to add an additional layer of security. Keyboard and mouse usage can be a biometric feature for users. It will require less data to be authenticated. It can be used in future for online behavioral authentication schemes. For security-sensitive apps, continuous authentication is critical. A session can be attacked in several ways like a Man-in-the Middle (MITM) attack where an attacker can steal the session key between a server and client. After that they can access the system later. The user will know the information after the breaking. By that time there may be some changes already that can not be revoked. Spoofing attacks can be prevented by continued authentication (Siddiqui et al., 2021).

### 1.1 Aims and Objectives

In the project, the aim was to develop an application that would detect whether or not a user is authentic. Prior research has revealed a number of challenges related to data being able to change based on physical equipment or being unstable during the collection of data. It is possible for a coat to slow down hand movements or a scratch on a finger make typing slower than usual. When someone wears gloves, that will result in a different pattern for

learning. In the real world, there will be several challenges. According to previous research, researchers initially analyzed user keystrokes for two features. 1% of the results were accurate. In the past, a variety of machine learning algorithms have been used. Random Forest, Neural Networks, K-Means, KNN but K-Nearest Neighbor gives the best result in every research. So, KNN has chosen to implement this application. KNN distance metrics help to generate clusters. Cluster is a chunk of a dataset which is explained further in this paper. Several distance metrics are used in the past research papers like Manhattan, Euclidean, Minkowski and others. As part of this paper, these distances are also explained. The last timing was 5 minutes and in this application the target was 3 minutes to detect the user. For this particular topic always the aim will be time consideration.

## 1.2 Overview of the Report

An application has been developed for the project. The project developed on HTML, CSS, JavaScript and python. Several libraries used to develop the application. It is important to note that Python flask serves as the main framework for balancing the project architecture. The application was developed with several front-end pages. The application generates data which is stored in an online database called MongoDB. In chapter 2, several authentication methods are discussed, from basic authentication to biometric authentication. It is also necessary to be aware of the bugs in the system so that the problem can be fixed from all angles. To use the secured login process, a registration form is created for new users and a login page is created for existing users. For pattern learning, Machine Learning uses the K-Nearest Neighbors algorithm to distinguish between real and fake users. Browser-based interval-based data collection. The browser sends the data to the server at regular intervals. AJAX is essential to making an interactive client-server interface. Data was collected by counting the interval between key entries using timestamps. This program measures the time between pressing two keys or releasing two keys. For the generation of a shared key, a key exchange procedure is also employed. This encryption is based on NIST P-256. During the diffie hellman key exchange between a server and a client, the procedure is carried out. An encrypted password is assigned to the secret, which will be used to create a session key. In order for a browser to be secure, dynamic key exchange is not necessary. Using dynamic secret keys in flask will prevent the session key from being stored. Sessions will be cleared every time they are refreshed. If someone logs out of the web application, they can't log in again without their username and password. Flask handles navigation in the web application. Two research backgrounds were selected for the project to discuss where both research conducted on password typing characteristics. In 2009, more than 100 users were surveyed for the last research study. As part of the evaluation of the performance of new distance matrices, the same dataset was used. The following set of data was collected from users who manually entered text into a text field to generate the data. It should be noted, however, that data is the primary concern in this area for every research. This has been the case in the past when researchers have applied their models to the dataset. However, they did not develop a

software application to facilitate password typing. It may be helpful, however, if the model is capable of detecting a good level of accuracy. Future research will focus not only on keystroke dynamics, but also on mouse and smart phone touch sensitive data. During the collection of data, some challenges were also encountered. Data can be affected by changes in user behavior resulting from fast or slow typing speeds.

## Chapter 2

# Literature Survey and Background

### 2.1 Authentication

An authenticating process involves verifying the identity of an authorised individual or thing. The purpose of authentication is to prove the authenticity of something which was made by a person or a person or an incident in history. Dealing with security is a common problem in every field. In practice, authentication can be performed by testimony, identification documents, or when the authenticator already knows the individual or thing (Contributors, 2019). In Fig. 2.1 , the figure shows examples of authentication processes in different locations. A security check at an airport is illustrated in Example 1. Security reasons require airport authenticators to ensure that travellers do not carry any harmful things or devices. Example 2 illustrates how ID cards are used as identity cards throughout any organisation. Authenticators will not allow it without the ID card or if the person's details do not match in their existing database if they forgot it. In the last example, we check the validity of car number plates.

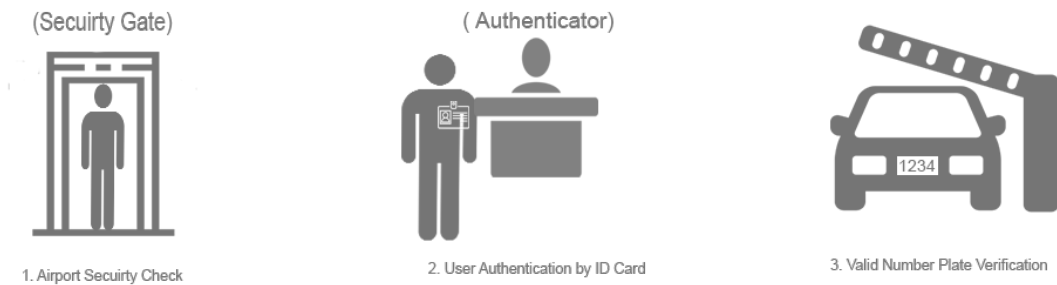


Figure 2.1: Simple Authentication

### 2.1.1 Password Authentication

The password authentication process is straightforward to follow. When using password authentication, usernames and passwords are critical considerations. Access to a system or website will be granted if the inputs are correct. There are times when users use unsecured passwords like 'password', 'abcdefg', '12345' etc. There is a high probability of cracking or guessing these passwords. It is common for hackers to use programs to decrypt passwords. If the password is simple, it will be much easier for the hacker to gain access to the account. Now, programmers are striving to make sure that the system is secure and that users are not able to access any simple passwords. The minimum length is usually 8 digits, with uppercase letters, lowercase letters, numbers, and special characters mandatory on most websites (N-Able, 2019). In Fig. 2.2, an authorised user must enter a username and password in order to access the system. The use of password authentication is common practice on all electronic devices and software's, including smartphones, tablets, computers, and most websites. Occasionally, websites set an expiration date for passwords, and users need to change their passwords before or on that date. Users with old passwords may be unable to reset their passwords in the system.



Figure 2.2: Password Authentication

### 2.1.2 Two-Factor Authentication

Two-factor authentication is similar to password authentication, except it involves the use of a physical device registered by the user or provided by the organisation where the user is registered. A one-time password (OTP) is sent to the user when he or she enters their username and password, or through the use of an authenticator app generates a pass-code or approves a notification sent by the system when the user attempts to log in. Using the second authentication method will prevent someone from accessing the system if they already know the password. A number of examples can be found in our daily lives, such as ATM's. ATM withdrawals require the user to simultaneously use their debit card and pin (N-Able, 2019). In Fig. 2.3, it is shown in the figure that the same process is used for logging in as with normal password authentication, but with an added layer of security. The user receives a text message with a one-time password (OTP) that enables them to successfully log into their account after entering the OTP.

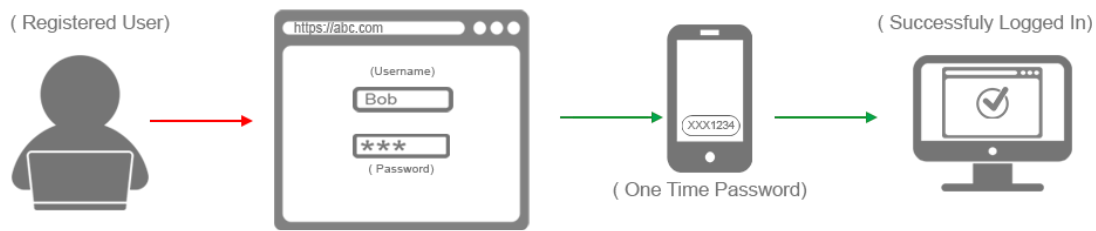


Figure 2.3: Two-Factor Authentication

### 2.1.3 Biometric Authentication

Biometric authentication is one of the most secure authentication methods available. In order to authenticate a person, physical features are required. An individual may authenticate by using his or her face, retinal scan, voice or fingerprint. For a legitimate user, it is very difficult to break or pass this authentication. This authentication method has the advantage that users do not have to remember usernames or passwords, nor do they have to carry any cards or devices in order to authenticate. Occasionally, devices such as phones, laptops and tablets are unable to authenticate a particular feature, such as fingerprints and retinal scans. In order to accomplish this, companies use special devices. It is not uncommon for users to refuse to share their physical characteristics or identities with companies or government agencies unless they have a legitimate reason to do so (N-Able, 2019).

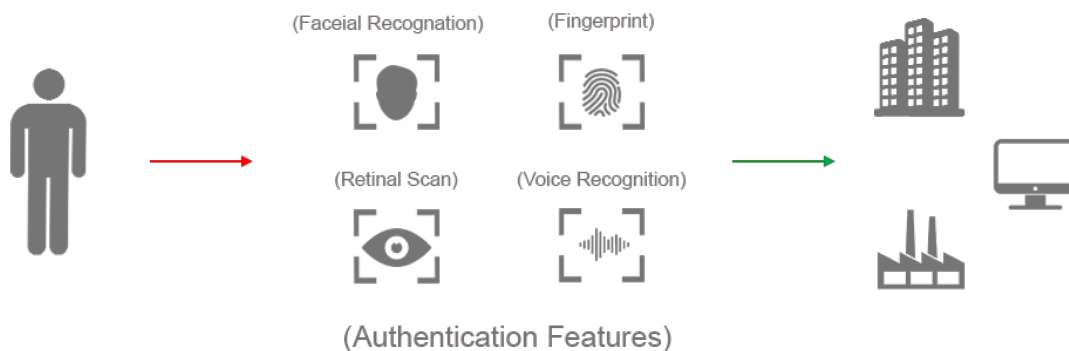


Figure 2.4: Biometric Authentication

### 2.1.4 Token Authentication

In some cases, companies do not want to use cell phones or rely on any other authentication layer (N-Able, 2019). Hardware and software are both involved in token authentication (inwedo, 2022). The token authentication process is built to support two-factor authentication. In place of a cell phone, companies offer smart cards or USB tokens (N-Able, 2019), which must be used with a card reader or USB port in order to operate (inwedo, 2022). Cards like RFIDs with radio frequency communication and USB dongles. The use of authentication devices also requires users to be cautious in order to prevent their use by unauthorised individuals (N-Able, 2019). Unlike hardware tokens, software tokens are not visible and can be used by any electronic device, including smartphones, tablets, laptops, etc (inwedo, 2022). In Fig. 2.5, this figure illustrates two examples. One is hardware that uses RFID technology. Once a RFID tag is placed near an antenna, the antenna communicates with the RFID tag and activates an electric current to unlock the door. An example of server authentication is shown in the second illustration, in which the user logs in and a token of authentication is generated from the authentication server, and based on that token, the user receives the data from the resource server.

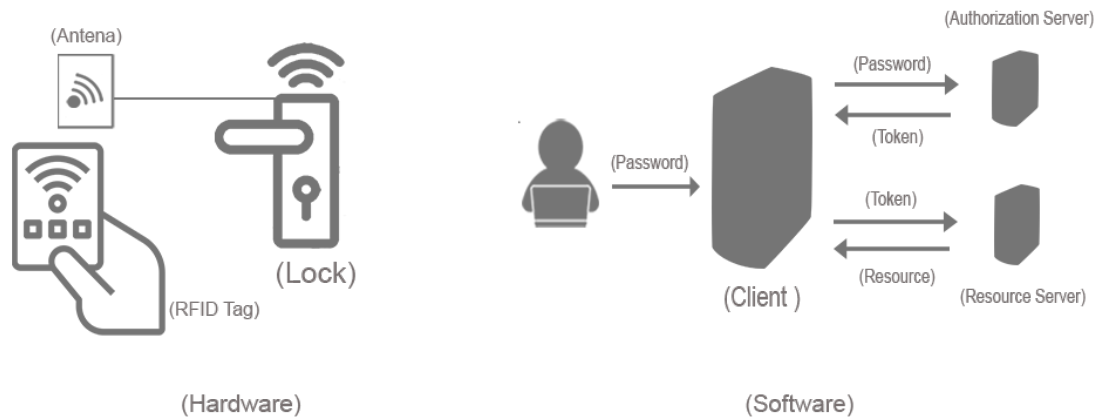


Figure 2.5: Token Authentication

### 2.1.5 CAPTCHA

Automated attacks are increasingly being made by hackers in an effort to gain access to the system. The use of captcha is effective in preventing this type of attack. As opposed to detecting an automatic attack, it focuses on verifying an actual human is attempting to use the service. By using numbers, pictures and letters in a different format, captcha attempt to generate a unique identifier that only a human is capable of deciphering and not only.

Although it provides an additional layer of security against hackers, it only causes problems for disabled individuals such as blind individuals in auditory screen readers. The captcha are inaccessible to them. It is not uncommon for non-disabled users to encounter some difficulty in solving captcha (N-Able, 2019). In Fig. 2.6 (okta, 2021) this section, you will find some examples of captchas that are used to authenticate users.

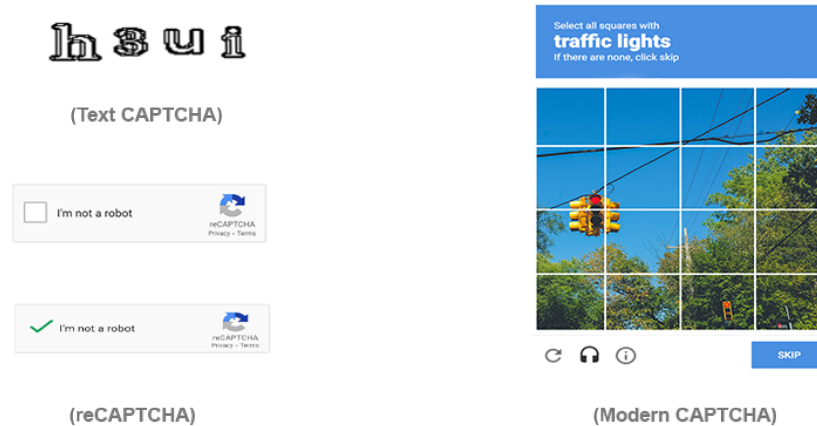


Figure 2.6: CAPTCHA

## 2.2 Languages

### 2.2.1 HTML

The HTML language is used for the distribution of information worldwide. The language is understandable globally. In addition to being understandable by most computers, it is also used for computers to publish content. The language known globally as HTML is used by the World Wide Web (WWW). HTML stands for Hypertext Markup Language. Publishing, data, media content, files, and finding information online are all made easier by HTML. It was originally developed by Tim Berners-Lee. After authors and vendors began using HTML in the same way, HTML started expanding in different directions. The development is boosted by a joint effort between different organisations. There is now a version 5 of HTML available. This mechanism changed in various ways, such as developers now being able to use style sheets, scripting, frames, embedding objects, as well as improving accessibility for persons with disabilities (Raggett et al., 1999).

### 2.2.2 CSS

Contents published in HTML can be designed with CSS (Cascading Style Sheets). Published documents are presented in an attractive way using this technique. HTML without CSS looks like a basic designed page. It has become critical for companies to develop front-end development strategies to attract or maintain meaningful information for their customers



(Meyer, 2006). The CSS 4 standard is currently available. In addition, it offers a dynamic interactive option using JavaScript or some JavaScript libraries such as AJAX and jQuery. In addition to allowing the user to change the colors, spacing, borders, and dimensions, there is also the option of adding animation and transformation. In order to make the libraries more dynamic, some features have already been added (Nixon, 2014).

### 2.2.3 JavaScript

For web browsers, JavaScript is an important language. As a result of its integration with browsers, it has become the most popular language. It was after the failure of Java<sup>TM</sup> applets that JavaScript was introduced on its own to the industry. Aside from that, many programmers consider it to be a distasteful language as well. The majority of people do not learn JavaScript because if they wish to develop something in an environment which only supports JavaScript, they will choose to learn an alternative language which can be supported in most of the environment. Since JavaScript is an easy-to-learn language, it can be used by anyone. It also supports object-oriented programming, so users can structure data based on objects and store them as JSON files (Crockford, 2008).

### 2.2.4 Python

Python was developed approximately in 1991. In other words, it is an interpreted language (Python, 2021). A program is executed by an interpreter, which is why it's called an interpreted language (Downey, 2012). Python covers a wide range of areas including procedural, functional and object-oriented programming. It is inspired by different programming languages, including C, Java, Lisp, and Haskell (Python, 2021). The Python programming language optimizes the accuracy, performance, integration, efficiency and of the system. In addition to product design, web development, user interfaces and system programming, it is now used for a variety of tasks. As a result of its ease of implementation, Python is often called a scripting language (Van Rossum and Drake Jr, 1995). As a result of its ease of implementation, Python is often called a scripting language. As well as that, it has the fastest expansion rate in the world. Developers are able to learn and implement machine learning with the help of well-designed packages (Vallat, 2018).

## 2.3 Use of Artificial Intelligence (AI) in Real Time Application

The purpose of artificial intelligence is to act like a human and to assist humans in their daily activities. Artificial intelligence is the combination of a number of methodologies, including deep learning and machine learning (Javatpoint, 2021). In everyday life, people use AI for a variety of applications. In addition to being used in different domains for different purposes, the improved version of the applications is able to perform correct calculations and process large data with accurate results. Maps, social media, search engines, banks, and marketing are just a few examples of industries. AI has become dominant in every field.

### 2.3.1 Navigation System

Most of the people depend on google maps. People find it useful to find new places easily. Previously maps used online satellite navigation to guide users. As a result of AI, now a user not only has a better experience, but also has the opportunity to see surroundings. The use of image processing to recognize handwriting labels is also helpful for users to find their exact location. An application that understands traffic and learns about it. Users can then choose the best route for their journey based on that information (Javatpoint, 2021).

### 2.3.2 Language Processing Applications

There are times when users have difficulty typing an email or sending a message to someone. Sometimes people forget to correct typing mistakes. In addition to preventing this mistake, AI helps recommend keywords that can help users write in a better way and provides a writing tone that the user is trying to achieve. The process of learning a new language requires a lot of patience. In the model, the data feed helps to identify mistakes and show them in colored markings. Additionally, it makes learning easier for users. The use of language processing can help users solve their problems in other ways as well. Frequently, users encounter difficulties getting answers for a specific product or service and they try to get instant solutions at that moment. They may have to wait for someone to assist them in person or over the phone. By the help of AI complex questions can also be answered online with the assistance of chat bots. Users received good recommendations online for their searches. The majority of the time, they are able to locate the answer or information that the user was looking for (Javatpoint, 2021).

## 2.4 Machine Learning

Programming techniques based on machine learning aim to learn from data sets. Typical daily mail systems receive a large number of emails, including spam. Whenever a user marks an email as spam, the machine learning model learns that, and marks unmarked or regular emails as ham. Data records such as these can be used to train a machine learning algorithm. The training process is considered as an instance (Géron, 2017).

To learn and categorise the data, machine learning uses a variety of systems. Human supervision is provided during the implementation of the methods. A number of methods are available, including supervised, unsupervised, semi-supervised, and reinforcement learning. The learning of systems can be done online or in batches. In the system, data points are compared with previous data or identify patterns and develop models to predict the outcome. It is not necessary to use all criteria. Depending on the requirements or desired outcome, Machine Learning can be customised in any way. It is possible to classify machine learning based on supervision and time provided during training (Géron, 2017).

### 2.4.1 Supervised Learning

In supervised learning, the data collected for the training already have the result and it is called labels. A supervised learning program is a classification program. In Fig. 2.7 (Géron, 2017), showing the example of email trained with classes of spam and ham email and try to predict the result for a new instance. Emails should be classified by the system as they are received (Géron, 2017).

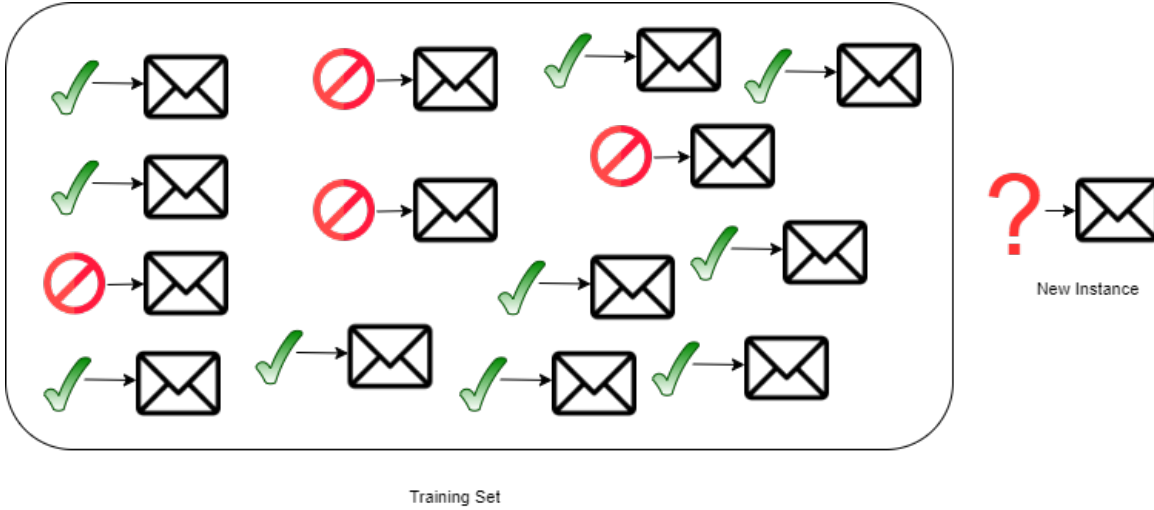


Figure 2.7: Supervised Learning for label based training

The number of indicated words or characters is equal to the number of words or characters in an email. The example in Table 4.1 illustrates the difference between spam and ham in words and characters (Hastie et al., 2009).

	george	you	your	hp	free	hpl	!	our	re	edu	remove
spam	0.00	2.26	1.38	0.02	0.52	0.01	0.51	0.51	0.13	0.01	0.28
ham	1.27	1.27	0.44	0.90	0.07	0.43	0.11	0.18	0.42	0.29	0.01

Table 2.1: Spam and Ham Data Sample

In order for the prediction model to learn, the data is collected. As a result of learning, an outcome can be predicted for unknown data. A good learning process will produce accurate results. Using the example, we can see how supervised learning works. The term supervised name is derived from label guidance learning. Previously, a study was conducted to predict whether emails would be classified as junk mail or not. This project was designed to detect and judge whether an email should be categorised as spam for the user or not. 4601 email messages were used for training and classified as spam or email (legitimate user). According to the analysis of the research, 57 words and punctuation's appear frequently. This problem

falls under the category of supervised learning. An example of a classification problem is a problem with labels attached to spam/email.

For another example. It is possible to measure the price of some items based on their features. Predictors like age, brand, and mileage are used for cars. Regression is a technique for sorting tasks. Various car samples with different features are needed for the analysis. Mileage may be defined differently for machine learning purposes. You can use the feature for different purposes in different circumstances. In this case, it is denoting a numerical value. Despite the fact that attributes and features are often used equally. Regression can also be used as a classifier sometimes. The logistic regression method, for example, is most often used as a classifier (Géron, 2017).

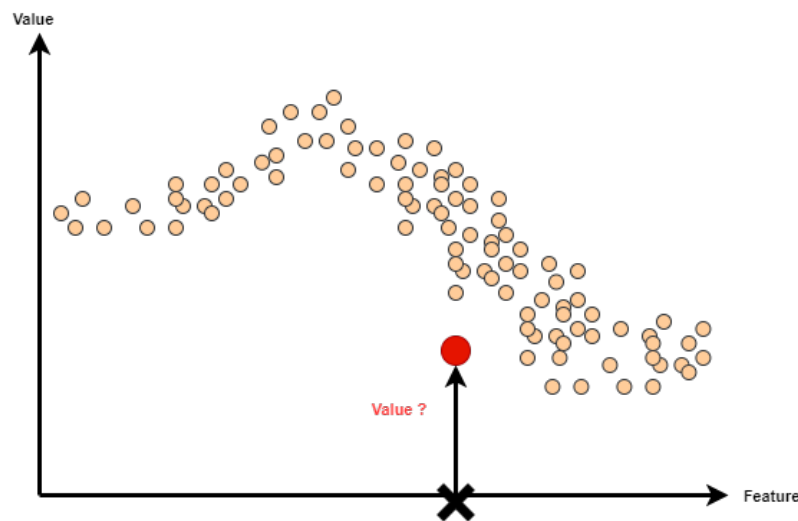


Figure 2.8: Regression

### 2.4.2 K-Nearest Neighbors

The KNN algorithm is a type of supervised learning algorithm. That has been developed for the purpose of data mining and machine learning. It is an example of classification, where it compares past and present data using vectors. In terms of algorithms, KNN is a simple one. This Table 2.2 (kaggle, 2020) illustrates how a database can be fed into a machine learning model to predict a new used car price. The machine learning algorithm predicts a new price based on the similar features in the used cars dataset. Each type of combination refers to the price of each individual vehicle. The price of any car can be predicted by using each row of data as a classification. In the case of a car purchased in 2012 with 80000 kilometres of mileage, the model will attempt to find the closest matching set of data and estimate the price (José, 2018).

brand	model	year	price	transmission	mileage	tax	mpg	engineSize
focus	5 Series	2014	11200	Automatic	67068	125	57.6	2
focus	6 Series	2018	27000	Automatic	14827	145	42.8	2
focus	5 Series	2016	16000	Automatic	62794	160	51.4	3
focus	1 Series	2017	12750	Automatic	26676	145	72.4	1.5
audi	A1	2016	11000	Manual	29946	30	55.4	1.4
audi	A4	2017	16800	Automatic	25952	145	67.3	2
audi	A3	2019	17300	Manual	1998	145	49.6	1
audi	A1	2016	13900	Automatic	32260	30	58.9	1.4
audi	A6	2016	13250	Automatic	76788	30	61.4	2
audi	A4	2016	11750	Manual	75185	20	70.6	2
audi	A3	2015	10200	Manual	46112	20	60.1	1.4
skoda	Octavia	2017	10550	Manual	25250	150	54.3	1.4
skoda	Citigo	2018	8200	Manual	1264	145	67.3	1
skoda	Octavia	2019	15650	Automatic	6825	145	67.3	2
skoda	Yeti Outdoor	2015	14000	Automatic	28431	165	51.4	2
skoda	Superb	2019	18350	Manual	10912	150	40.9	1.5
skoda	Yeti Outdoor	2017	13250	Automatic	47005	145	51.4	2

Table 2.2: Used Car Data-Set

KNN follows distance measurement algorithms like Euclidian, Manhattan, Minkowski. Where these formulas help to find the labels for prediction. Where in 2.1 euclidean distance  $p$  and  $q$  measure as two points in Euclidean  $n$ -space. Each side of the triangle is calculated by subtracting one point from another point in every dimension. Add the squared result to the total. Finally, the square root of that measurement represents the Euclidean distance. For 2.2 Manhattan distance, sum all absolute differences between all dimensions between any two points. Where  $n$  is number of dimensions. Two dimensions are used to represent Manhattan distance. The 2.3 Minkowski distance metric can be defined as the generalisation of the distance metric across an normed space of vectors. It is simply a generalisation of Euclidean and Manhattan distances (José, 2018).

$$d(P, Q) = \sqrt{\sum_{i=0}^n (q_i - p_i)^2} \quad (2.1)$$

$$d(P, Q) = \sum_{i=1}^n |p_i - q_i| \quad (2.2)$$

$$d(X, Y) = \left( \sum_{i=1}^n (|x_i - y_i|)^c \right)^{\frac{1}{c}} \quad (2.3)$$

An instance of a cluster is generated by a KNN model in Fig. 2.9. The colours of each cluster correspond to the cluster in which they belong. It is possible for clusters to be scattered if classification is complicated or overlapped.

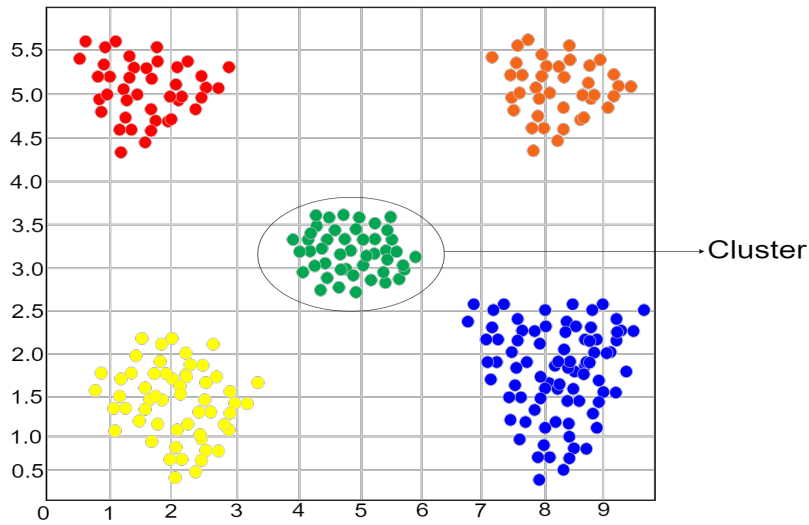


Figure 2.9: K-Nearest Neighbors

According to Fig. 2.10, it will show the distance between clusters based on the value of  $k$ . If  $k$  is 3, the distance between the 3 clusters closest to it will be calculated. In the figure, the circle around 3 classes represents the nearest 3 clusters. Assuming  $k = 8$ , it will calculate distances for the 8 nearest classes as shown in the graph. The prediction for the three classes is blue since the blue number class received the most votes. As the result of a value of 8 is orange, the result is due to the fact that it received four votes (José, 2018).

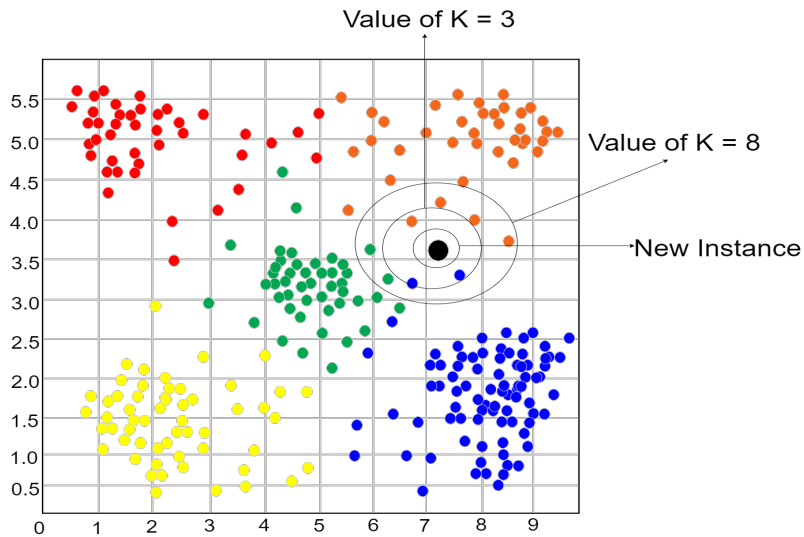


Figure 2.10: K-Nearest Neighbors with New Instance

## 2.5 Continues Authentication

The authentication of authorised users poses one of the biggest problems in implementing secure systems. In our real-world application, password authentication or two-factor is available. New implementations are essential to getting stronger authentication. In the first level of security, strong passwords or strong encryption methods have already been developed along with encrypted communications. Authentication devices are more expensive or have a much higher risk of being lost. One of the strengths will be being able to develop something that can't be transferred like hardware devices or difficult to understand. The implementation of biometric authentication requires a substantial budget. Small businesses cannot afford it or mid-sized businesses find it difficult. The implementation process can also be lengthy and cannot be updated easily with new technology. It is feasible to implement low-cost authentication techniques such as continuous authentication on a low budget and to authenticate without requiring the user to provide any personal information. Analysing handwritten letters with their previous letters is similar to this process. The system should detect whether the user is acceptable or not. The system can prevent potential damage caused by imposters. The detection of imposters in less than 100 words had already been studied in a previous experiment. A slight change or temporary difficulty can change the typing pattern. It is possible, for instance, for a user to use his coat to fill up a heater in a cold room. Typing speed may slow down when a hand moves slowly due to a coat, or changes may occur in writing pattern. For the purpose of identification, Keystroke characteristics are crucial. A unique digital print is created by the time interval between keys. Also the error frequency can be tracked for authentication. Keystroke force, typing rate and text statistics also can be a helpful feature with Keystroke timings. A key rollover and key lockout system is already built into the keyboard hardware. The most natural typing can be done with these facilities. In the absence of it, the user may need to press one key at a time, which may result in slower performance. A single key is pressed and another is pressed before releasing or after a few milliseconds, handled by key rollover. If a key is repeatedly pressed, the data can be affected, which can be dealt with by a key lockout that eliminates that particular key data (Shepherd, 1995).

There are times when the keyboard hardware interrupts for key presses and releases. A hardware scanner identifies the key codes from key press or release actions, which are then processed by the system for identification. Most of the information is lost after data is transmitted to the BIOS. As an example, the capital key is not valid unless you press the shift key. This is why alt, ctrl, and shift keys provide left-handed and right-handed users with two options on the left and right side of the keyboard. In BIOS, both sides produce the same results. No matter which way it is pressed, the system discards it. That is not the logical part. The time of each key is recorded and it is known as the key input handler. By using the system, the keystroke interval and overlap are calculated. In the past, an application using Turbo Pascal was developed. In order to analyse keystroke logs, these systems utilise TSR. The application was based on PASSWORD typing detection and required a password

input 5 times. The stats generated from duration of individual keys and duration. Then they calculated the mean and variance. The authorization will be based on these measurements (Shepherd, 1995).

The measurement is accurate to within 1%. The challenges were related to timing. The data was affected by long pauses. As a result of breaks, the system was suspended in order to stop calculating means and variances. Data from four users was presented as an example in Table 2.3. Two of them were professionals and their typing rate per second was similar but learned from different people. The third user was not a professional typer but a frequent keyboard user. An irregular individual was present in the fourth case (Shepherd, 1995).

User	Duration		Interval	
	Mean	Var	Mean	Var
1	66ms	12ms	122ms	18ms
2	59ms	8ms	104ms	22ms
3	81ms	20ms	225ms	140ms
4	102ms	34ms	665ms	320ms

Table 2.3: User Typing Characteristics

In Table 2.3, the professional user strikes 6 keys per second, that's 60 words per minute with an average of 6 letters per word. In addition to the keystroke durations being different, the durations were also remarkably different. It takes the third user longer to find keys than it does for the professionals. The fourth and last user is demonstrating the typing experience by comparing data with others.

### 2.5.1 Keystroke Dynamics

A second study was conducted on authentication. Based on novel keystroke dynamics authentication for industrial use. Where a system can detect a non-legitimate user. A second approach to analysing password typing data. A method of collecting data from a web browser by using JavaScript. Therefore, they determine four characteristics: the interval between key pressure events, the interval between two key releases, and the interval between one key press and release, also the opposite. The dataset comes from a 2009 study on keystroke dynamics (Mhenmi et al., 2018).

For verification they used classification methods by using K-nearest neighbours (KNN). It provides high accuracy when it comes to keystroke dynamics. They used different distance metrics to increase the performance and accuracy. One was stastical distance another one



was hamming distance (Mhenni et al., 2018).

$$D_{STAT} = 1 - \frac{1}{n} \sum_{i=1}^n e^{-\frac{|q_i - p_i|}{\sigma_i}} \quad (2.4)$$

$$D_{HAMMING} = (\#(q_j \neq g_j(k))/n) \quad (2.5)$$

In order to calculate 2.4 statistical distance metrics, biometric features are individually analysed for statistical data. For competitive performances, the distance calculation is commonly used. Predictions are generated based on the calculation speed. In 2.5 Hamming, the percent difference between the labels and the novel query is calculated (Mhenni et al., 2018).

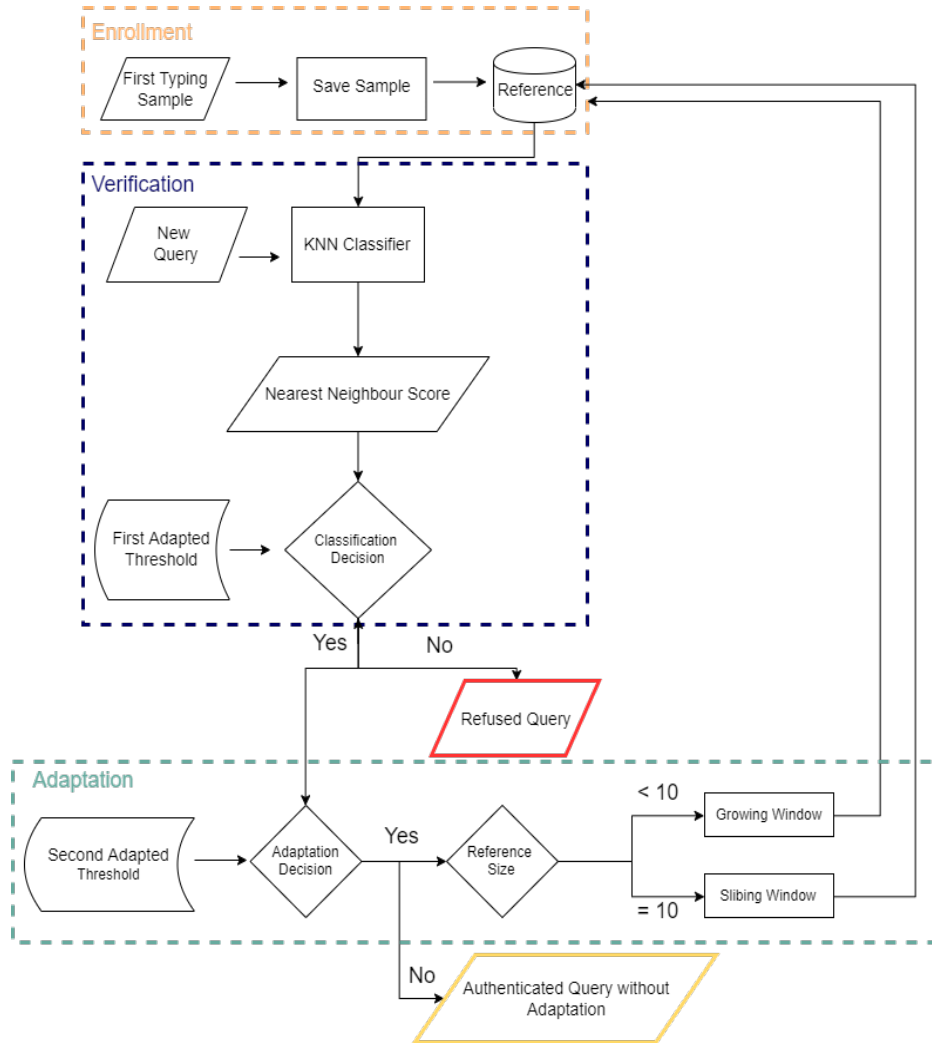


Figure 2.11: Keystroke Authentication Process

$$D_{EUCLIDEAN} = \sqrt{\sum_{k=1}^m (q_i - g_j(k))^2} \quad (2.6)$$

$$D_{MANHATTAN} = \sum_{k=1}^m |(q_i - g_j(k))| \quad (2.7)$$

To calculate the distance metrics in KNN, 2.6 Euclidean and 2.7 Manhattan distance calculation methods were also used. To calculate the distance metrics in KNN, Euclidean and Manhattan distance calculation methods were also used.  $q_i$  is the query for a user and  $j$ ,  $g_j(k)$  is the  $k^{th}$  reference sample of the user.  $j$  and  $m$  are the numbers of samples in the reference and  $g_j$  and  $\mu$  is the mean vector and sigma was the standard deviation vector of references.  $N$  is the length of the user, where  $i$  ranges from 1 to  $n$  (Mhenni et al., 2018).

The user was asked to enter password type data during the enrollment phase. The user data was processed as a sample. After the data was received, it was added to the queue. The query limit was set to 10. References do not rely on more than 10 datasets. To make fast communication between server and database. After validating the dataset it goes to the adaptation part. To make a decision, a standard threshold was set. It was based on a double threshold to standardize. The first one made a decision to accept or reject the data and the second one made a decision to update or not. Adaptation of the data update to the system during use of the system (Mhenni et al., 2018).

$$T_j^{i+1} = T_j^i - e^{-\frac{\mu_j}{\sigma_j}} \quad (2.8)$$

$\mu_j$  and  $\sigma_j$  represent each user's mean and standard deviation. In adaptation I,  $T_j^i$  was the threshold and the  $j$  was the user. A semi-supervised learning approach is applied in these modes. The labels were assigned by KNN classifiers. Different distance matrixes were used to get an optimal distance. Using each query to update the dataset, the whole dataset works online. In addition, they proposed a double-serial mechanism for their experiments. It waits until each data set has a length of 10, then switches to the old dataset to check the experience. This works properly without any modification as a self-learning process. In the case of a verified decision, a consequence follows. Real-time updates are made to the reference and threshold. With the help of the double serial mechanism they developed a stable model of keystroke dynamics for users. A method of intra-class verification was used (Mhenni et al., 2018).

This study used two sets of data for validation: GREYC 2009 and WebGREYC. The dataset was generated through the participation of 133 users. They are looking for 100 users for five acquisition sessions over two months and 60 samples per user. In the webGREYC dataset 118 users participated but 45 of them only participated for 60 patterns data collection process. For both the datasets, they consider 60 users each (Mhenni et al., 2018).

The stream generation method is used to extract the data from the user. The 60 samples collected from users were entered into an evaluation protocol. They divided the process into two parts. Each process had eight queries to the system. The data consists of five genuine user records and three imposter records. A sample of ten genuine users was collected in a batch as part of the study to collect a true user sample. In the database, the original user data is arranged in sequential order. In this case, the imposter query runs at random. As a result of that they obtain 12 sessions of adaptation where it calculates at 60/5. They divided the dataset into two parts where the original user data was taken 37.5% which was calculated from 3 by 8 and for the imposter they took 62.5% data. They calculated the data from 5 by 8. The attack rates were higher than general keystroke attacks with 70% genuine and 30% imposter data. They set the EER rate to 3%. They used 4 distance matrices to calculate the result (Mhenni et al., 2018).

Referrene Size					
Adaptive Mechanism	Minimum	Mazimum	Classifier	EER	AUC
Double	1	10	KNN(Hamming)	6.1%	0.013
Serial	1	10	KNN(Statistical)	6.3%	0.017
	1	10	KNN(Eculidean)	7.8%	0.033
	1	10	KNN(Manhattan)	8.9%	0.031
Avergae	5	15	SVM	6.96%	-
Mechanism	5	15	Neural Network	8.75%	-
[8]	5	15	Statistical	10.75%	-

Table 2.4: Classifier Comparison with 2009 Databse

In Table 2.4, EER rate of 6.69% was the performance they received. When they used a SVM classifier, they compiled a sample of 5 samples of a minimum size and 15 samples of maximum data. They used the same dataset from 2009 research. They got better results in EER and Statistical distances. In static distance, the EER rate was 6.3%, and in hamming distance, it was 6.1%. Offer a facility for deploying the application to the server to take advantage of low computing time. Their experiments refer to the evolution of size over a long period of time. It is due to the fact that the number of set queries was not the same for all the users. A rapid increase in the number of samples of data caused an interruption in the flow of data as a result. This is mainly due to the fact that sliding windows work better here because of that. In addition, they found that the performance was being negatively affected by a slower growing window. Initially, the performance rate for the research was lower than expected due to the weak recognition that was provided to new users at the beginning. This refers to Manhattan distance in KNN. Other distances works really well specially Statistical distance for the model. KNN struggle more than the other algorithms in terms of intra class verification since it has a smaller data size. The advantages of the algorithm are also used for other algorithms in this literature. They tested the result for single data then checked for multiple data (Mhenni et al., 2018).

## Chapter 3

# Implementation and Analysis

In this section of the report, the implementation and analysis will be discussed in extensive detail. In the literature review, the previous work has been discussed in detail. In this study, it was significant to collect data from different categories of people, and a total of 10-15 user data were collected. Previously, a public dataset was available from 2009 on password typing verification, and the application in these projects was implemented to authenticate users based on their writing pattern. Therefore, the previous dataset could not be used for implementation or training purposes.

### 3.1 Design

This project has been implemented using a number of libraries and languages. HTML, Bootstrap CSS, and JavaScript are the components used to build the front end. As the key to managing the layers of the application, the Flask framework has been used to develop the application. Several challenges were faced during the implementation of the project. All the web pages have been created in HTML and navigation are managed by flask. A key requirement of the project is the ability to determine whether the user is a legitimate user or not. In order to achieve this, a database was required, and MongoDB was selected as the online database server. In order to make it more dynamic and accessible to anyone with a keyboard of any size. Additionally, it is capable of updating the database on a dynamic basis. For the project to be successful, it has been necessary to engage in an interactive session in order to experiment and achieve the desired results. In order to maintain secure sessions between client and server, maintaining secure transactions has been of paramount importance. Flask has that option to maintain the session securely with security key and session key. Javascript was used as the first target for collecting user data from web pages and standardizing it into JSON format. After sending the message to the server, the server should check the status of the user and return a response.

### 3.1.1 Front-end

It is developed using HTML, CSS, and JavaScript for the front end of the application. There is a registration form Fig. 3.1 for the user to fill out. The user can register using their actual data or he or she can register anonymously using fake name and email id. In the case of email, they can use fake addresses such as "abc@mail.com", "123@mail.com". At present, the system does not validate email addresses because the actual user is not required for testing. Furthermore, storing their personal details without their consent is unethical. There is also an option for checking the email address to make sure there are no duplicates. A login page Fig. 3.2 has also been created for a secure login system. Here, users can log in

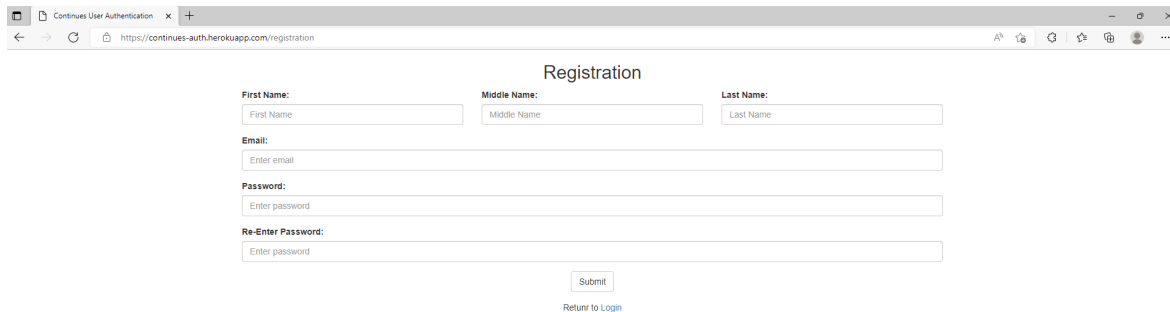
A screenshot of a web browser showing a registration form titled "Registration". The form is located at the URL "https://continues-auth.herokuapp.com/registration". It contains several input fields: "First Name:" with a placeholder "First Name", "Middle Name:" with a placeholder "Middle Name", and "Last Name:" with a placeholder "Last Name". Below these are "Email:" with a placeholder "Enter email", "Password:" with a placeholder "Enter password", and "Re-Enter Password:" with a placeholder "Enter password". A "Submit" button is at the bottom right, and a "Return to Login" link is at the bottom center.

Figure 3.1: Registration Page

using their email address and password that they used when registering with the system. A new session is started for the specific user when the system confirms that the email address and password are identical.

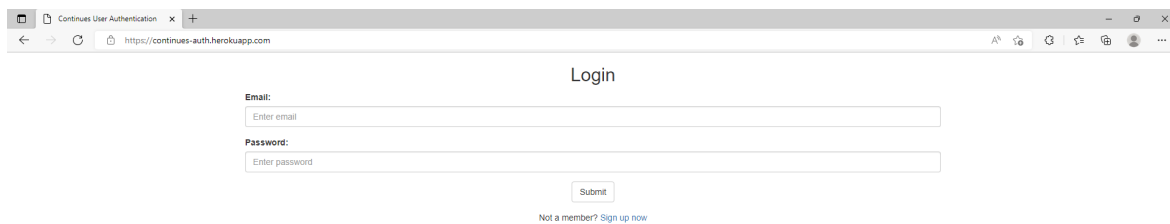
A screenshot of a web browser showing a login form titled "Login". The form is located at the URL "https://continues-auth.herokuapp.com". It contains two input fields: "Email:" with a placeholder "Enter email" and "Password:" with a placeholder "Enter password". A "Submit" button is at the bottom right. Below the button is a link that says "Not a member? Sign up now".

Figure 3.2: Login Page

The Fig.3.3 homepage is created for the user to write on and the user name will appear at the top of the page as the name they registered with. Those wishing to manually end their session can do so by clicking the logout button.

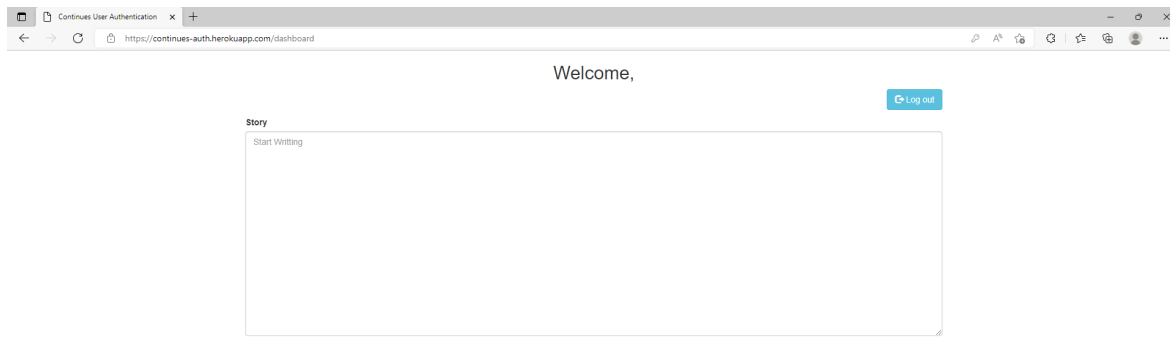


Figure 3.3: Home Page

A Fig.3.4 logout page is also created if a user ends the session manually or their system detects an unauthorized user.

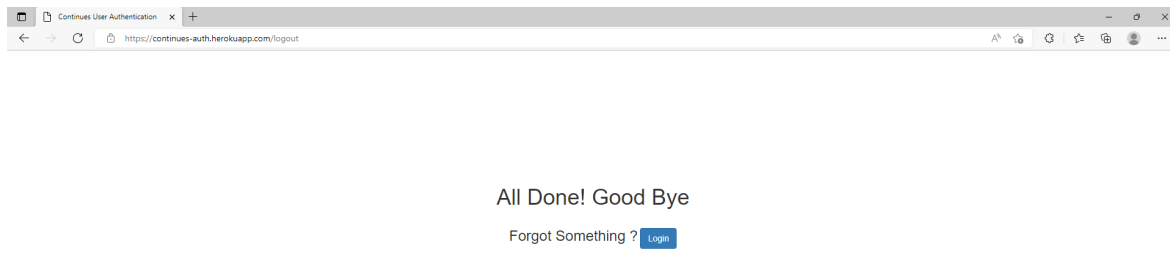


Figure 3.4: Logout Page

### 3.1.2 Back-end

#### Flask

Compact frameworks, such as Flask, are micro-frameworks. Small enough to become easily understandable and easy to learn from source code. It is important to note that size doesn't matter when it comes to functionality compared to other frameworks (Grinberg, 2018).

Flask app created for this application. It is necessary to generate a single secret key in order to maintain a dynamic session key. Similarly, if the secret key is also set to dynamic in every window slide, the browser will generate a new secret key and can't compare or remember the previous key, resulting in the session being cleared. Because the session is an important key in this research, it won't be a solution.

```
app.config['SECRET_KEY'] = 'h\xd0Pp\x17\xe0Z6\xfe\xa9\xa9\xf1\xe6\xb9\xc1\xedQ\xf1\xcd0=\x00\x01\xd6[\xff\x88k\xae\xfd\xa6\x9c' # Create Secret Key which is encrypted
```

Figure 3.5: Secret Key

```

server_private_key = ec.generate_private_key(ec.SECP256K1()) # Generate private key for server
private_key = ec.generate_private_key(ec.SECP256K1()) # Generate another private key
shared_key = server_private_key.exchange(ec.ECDH(), private_key.public_key()) # Generate shared key

session['session_key'] = shared_key # Create a dynamic session for secure connection

```

Figure 3.6: Session Key

A text encryption key is set in Fig. 3.5. In Flask, the key is stored in the back-end and is used to generate the session based on that key. As shown in Fig. 3.6, a shared key is generated following an elliptic curve diffie-hellman key exchange. In view of the fact that the flask handles the entire architecture, it is sufficient to use the shared key generated by the server for the session key. An environment in which a server private key and a private key are generated dynamically. The shared key is then derived from both the keys generated here. The curves are smaller than 224 bits, but they are faster because they are based on NIST p curves. As a result, the risk of a security breach is also reduced. According to safeCurves, NIST is the only cryptography standard supported by the software. An elliptic curve follows equation 3.1. P-256 is an elliptic curve that follows equation 3.2. Here,  $q$  represents the underlying field, and  $a, b$  represents the elliptic curve parameter, which in the case of P256 is 3. In a curve, point  $G$  is called the base point and point  $n$  is its base point (Adalier and Teknik, 2015).

$$y^2 = x^3 + ax + b \bmod q \quad (3.1)$$

$$y^2 = x^3 + 3x + b \bmod q \quad (3.2)$$

A user fills out a form, submits it, and data is received in the Fig. 3.7 "user\_registration" function after it has been submitted using a POST method. Data is received and passed to each variable separately. Combine name variables for storing into the database and in session. To avoid redundancies, the email is checked before submission. The password is encrypted using the bcrypt hashing algorithm.

```

@app.route('/submit-details', methods=['POST', 'GET'])
def user_registration():
    if request.method == 'POST':
        firstName = Validate.valText(request.form['first_name']) # get user first name
        middleName = Validate.valText(request.form['middle_name']) # get user middle name
        lastName = Validate.valText(request.form['last_name']) # get user last name
        fullName = combine(firstName,middleName,lastName) # Create full name for future use
        email = Validate.valEmail(request.form['email']) # Getting user email
        email = email.lower() # Lowering the email
        checkEmail = {'email': email} # Create query for MongoDB
        checkUser = userCollection.count_documents(checkEmail); # Run query for that email
        # Check if user is available or not
        if checkUser > 0:
            alert = [{"type": "warning", "name": "Warning", "description": "User Already Exist."}] # Set warning
            return render_template('registration_form.html', alert=alert) # If user already exist
        password = bcrypt.generate_password_hash(request.form['pwd']) # Encrypting password
        #Inserting User Data
        entry = userCollection.insert_one(
            {'fullName':fullName, 'firstName': firstName, 'middleName': middleName, 'lastName': lastName, 'email': email, 'password': password})
        return redirect(url_for('index')) #Successful entry goes to login page

```

Figure 3.7: Registration Back-end

A separate Fig. 3.8 login security function is developed. When the user submits his or her email and password, the data will be sent to the security function. To begin with, it checks whether the email type sent by the user is valid or not to prevent phishing attacks. It will then determine whether the user is available and if so, it will obtain the users current password. After that, encrypt the given password and compare it with the registered password if they match. A session will be created and the session key will be set to the application if every authentication is successful. In addition, the user's full name and id will be stored in the session cookie.



```

@app.route('/security', methods=['POST', 'GET'])
def authentication():
    if request.method == 'POST':
        userEmail = request.form['user_email'] # Stroing email from html form submission
        userPassword = request.form['user_pwd'] # Stroing password from html form submission

        inputEmail = re.match(r'^([a-zA-Z0-9_\.\-])+\@(([a-zA-Z0-9\-\-])+\.)+([a-zA-Z0-9]{2,4})+$', userEmail) # Check input type is email or not for an extra lyer of security
        if bool(inputEmail) == True:
            userEmail = userEmail.lower() # set email words to lower

        myquery = {'email': userEmail} # set query for MongoDB
        checkUser = userCollection.count_documents(myquery); # Document Count from a single user email

        # Checking if user is available or not
        if checkUser > 0:
            userData = userCollection.find(myquery) # Executing query to get data from the user
            dbPassword = userData[0]['password'] # Get user password from database
            dbUserId = userData[0]['_id'] # Get User Id
            dbUserName = userData[0]['fullName'] # Get user password
            dbUserId = json.loads(json_util.dumps(dbUserId))['_id'] # Remove user id object and convert into string
            print(dbUserName)
            hashedUserPassword = bcrypt.generate_password_hash(userPassword) # Encrypt current password
            checkPassword = bcrypt.check_password_hash(dbPassword, userPassword) # Check current password and will return true or false

            # Check if password is true or false
            if checkPassword == True:
                server_private_key = ec.generate_private_key(ec.SECP256K1()) # Generate private key for server
                private_key = ec.generate_private_key(ec.SECP256K1()) # Generate another private key
                shared_key = server_private_key.exchange(ec.ECDH(), private_key.public_key()) # Generate shared key

                session['session_key'] = shared_key # Create a dynamic session for secure connection
                session['profile'] = {'userId': dbUserId, 'userName': dbUserName} # Create session value
                gmt = time.gmtime() # Get current time
                timestamp = calendar.timegm(gmt) # Get current timestamp
                dateTime = datetime.now().strftime('%Y-%m-%d %H:%M:%S') # Get current date and time
                # Recording session entry time for every user
                sessionEntry = sessionCollection.insert_one({'sessionKey': session.get('session_key'), 'name': session.get('profile')['userName'], 'userId': session.get('profile')['userId']})
                return redirect(url_for('dashboard')) # If successful return to dashboard
            else:
                session.clear() # Clear the session
                alert = [{"type": "warning", "name": "Invalid", "description": "Username or Password!"}] # Message for error or warning
                return render_template('index.html', alert=alert)
        else:
            session.clear()
            alert = [{"type": "danger", "name": "Error", "description": "User Doesn't Exist."}] # Message for error or warning
            return render_template('index.html', alert=alert)

```

Figure 3.8: Security Function

Authentication is performed with this last Fig. 3.9 function. An AJAX setup was used in the front end to provide the data here. The JavaScript functions assist in the collection of the data and its transmission to Flask via AJAX. In order to merge the user id with the data, the function collects the user id from the session as soon as it receives the data. As a starting point, a row is stored every 10 seconds for three minutes based on the current behavior of that particular session and sent the data to the database. Afterwards, the data will be gathered in the same manner, but rather than sending it to the database, it will be stacked in a queue. It will be sent to a machine learning model for determining the status of the user, and in return a response will be received. Depending on the response, the system either stores the data in the database or clears the session by responding back to AJAX.

```

@app.route('/auth', methods=['GET', 'POST'])
def auth():
    global behaviour
    global validator
    global biometricTemp
    if request.method == "POST":
        sessionKey = session.get('session_key')
        if sessionKey == None:
            return jsonify({'status': 'fail'}) # Return fail if session out
        else:
            sessionUser = session.get('profile')['userId'] # Select user id from session
            data= request.get_json()
            data['user'] = session.get('profile')['userId'] # Select user id from session
            structure = json.loads(str(json.dumps(data))) # Create json stucture with user id
            # Check if current behaviour goes more than 18 or not
            if behaviour >= 18:
                print(f'In Behaviour End State {len(biometricTemp)}')
                # Check if suspicious behaviour goes more than 18 or not
                if len(biometricTemp) >= 18:
                    modelVal = str(Validation(sessionUser,biometricTemp)) # Check into model that is imposter or not
                    print('Final',type(modelVal))
                    #If imposter then it will return imposter status and logout from the system othe wise continue
                    if modelVal == 'imposter':
                        print('In Imposter')
                        return jsonify({'status': 'imposter'})
                    else:
                        print('In User')
                        keyCollection.insert_many(biometricTemp)
                        print('Prediction:',modelVal)
                        biometricTemp = []
                        return jsonify({'status': str(modelVal)})
                else:
                    biometricTemp.append(structure) # Add single data to queue to check
                    return jsonify({'status': 'continue'})
            else:
                print(f'In Behaviour State {behaviour}')
                behaviour += 1
                keyCollection.insert_one(structure) # Add single data for current behaviour
                return jsonify({'status': 'continue'})

```

Figure 3.9: Authentication Function

## Database

An example of a NoSQL database is MongoDB. For Craigslist use MongoDB is very popular in the market. A document can easily be stored in MongoDB rather than a relational database management system. The scalability and reliability is very high in MongoDB. A NoSQL database is a different kind of database from a relational database. Data no longer needs to be fixed based on the table like in a relational database. MongoDB follows JSON data storage architecture where it creates documents with dynamic storage. It is called BSON. MongoDB already has significant capabilities and versatility for multiple languages (Boicea

et al., 2012). The purpose of this system is to store keystroke data with 20 different features of data within the keystroke dynamics collection. There are two types of user directories: the user directory that maintains the login system, and the session that records the login time at the time of every login.

### Data Collect

Previous research has provided some key insights into the data collection process. For the purpose of recognizing the pattern, five major key techniques were used. One is the Fig. 3.10 Key Hold process, which is the period of time between pressing a key and releasing it. A second calculation is to determine the time interval between two keys being released. In other words, it's like when a user presses a key and then presses another without releasing the first. There is a process called Fig. 3.10 key down down. A third process is the opposite of a down-down process. As a result of the user releasing two keys, the time between those two events is called the Fig. 3.10 key up up period. Lastly, there is the Fig. 3.10 key up down and the Fig. 3.10 key down up. An additional key press and release is calculated in the same manner as a key release and a key press and release.

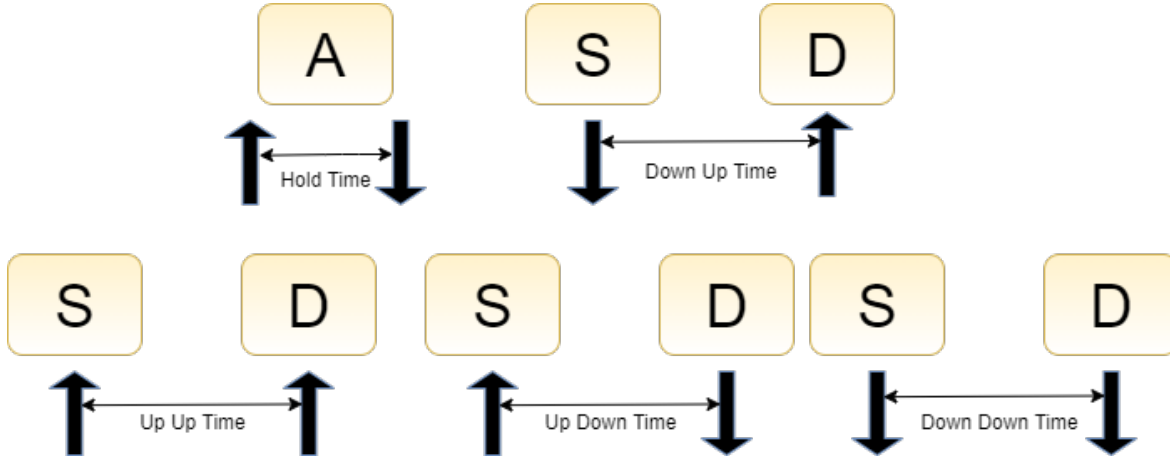


Figure 3.10: Key Example

There are a number of values coming from this pattern, and they are all calculated along with their minimum value, maximum value, 3.3 average, and 3.4 standard deviation. To get a better result.

$$A = \frac{1}{N} \sum_{i=1}^n a_i \quad (3.3)$$

$$\sigma = \sqrt{\frac{\sum (x_i - \mu)^2}{N}} \quad (3.4)$$

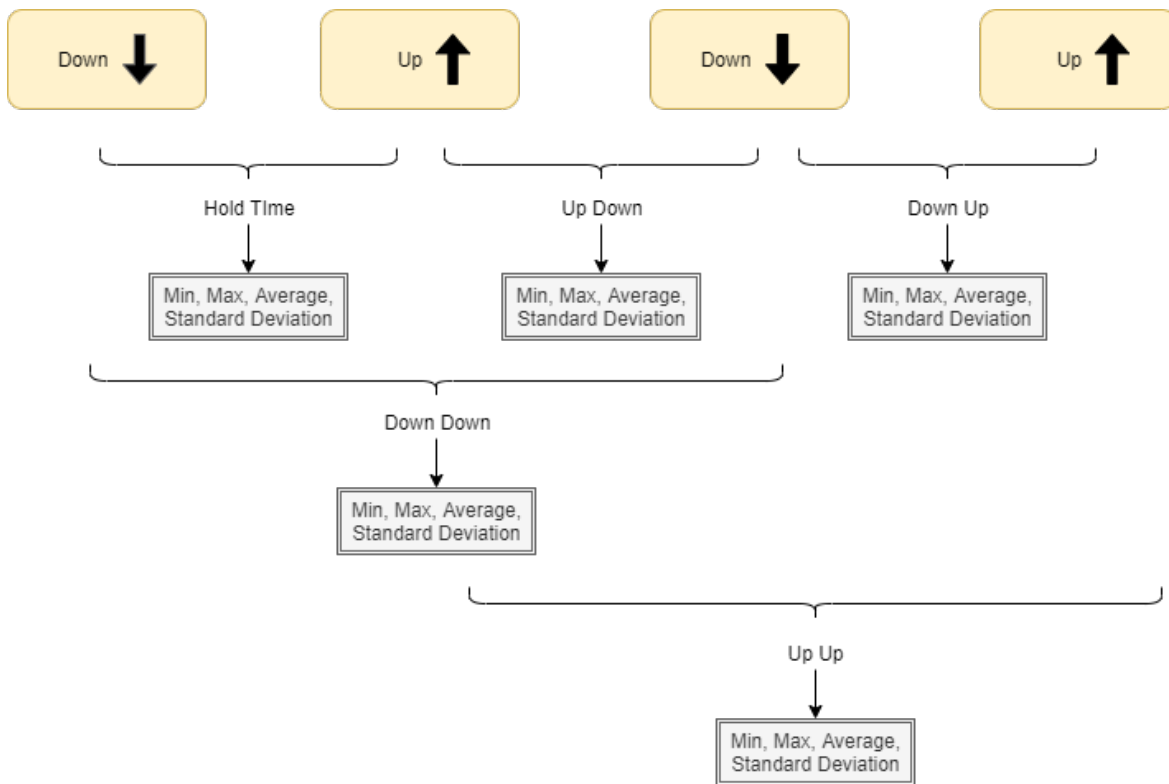


Figure 3.11: Keystroke Collection Process

The Table 3.1 data sample shows the values that were collected from different users. The points where different user values differ from each other for the same fields can be an excellent learning method for machine learning. In the analysis of the data, it is found that the maximum hold time detected by a user is 152 and the lowest is 124. A user has an 18.56 standard deviation and a maximum hold time of 144, but many other users have lower maximums although their standard deviation value is higher than the user.. The standard deviation is not affected by the hold maximum. Each field of data depends on the total time period over which the data was generated. As a result of the max value instead of the hold value, this feature has a higher value than the others. Among the lowest values in the table, 5 and 9 both represent up-down minimum values. This illustration shows the speed at which a particular type of typer moves up and down. From this data, one can determine the speed of the user. The maximum value for up down is too different from most other users, where one user has a value close to 1000 while another user has a value of 3592.

hold_min	64.00	58.00	42.00	70.00	53.00	50.00	59.00	55.00	73.00
hold_max	134.00	124.00	152.00	144.00	149.00	138.00	134.00	148.00	150.00
hold_avg	92.23	86.40	95.77	98.85	102.90	85.00	83.45	104.36	104.22
hold_std	21.35	15.84	23.75	18.56	26.26	21.90	21.66	21.37	21.60
down_down_min	184.00	87.00	111.00	69.00	138.00	121.00	121.00	130.00	94.00
down_down_max	2617.00	1741.00	1599.00	1084.00	2487.00	2566.00	3656.00	1050.00	1171.00
down_down_avg	765.31	317.76	418.76	351.41	457.05	413.13	971.60	406.60	322.04
down_down_std	786.82	307.38	439.82	268.03	521.01	514.18	1190.22	321.51	280.70
up_up_min	138.00	109.00	102.00	85.00	129.00	58.00	94.00	114.00	68.00
up_up_max	2674.00	1742.00	1614.00	1058.00	2482.00	2624.00	3671.00	1084.00	1240.00
up_up_avg	796.25	316.69	418.00	354.26	455.80	411.74	973.10	395.90	323.42
up_up_std	811.14	308.27	442.41	270.31	525.13	526.94	1196.90	328.68	289.62
up_down_min	60.00	5.00	12.00	18.00	35.00	21.00	9.00	35.00	14.00
up_down_max	2550.00	1642.00	1518.00	986.00	2412.00	2516.00	3592.00	971.00	1090.00
up_down_avg	673.08	230.38	322.38	258.96	353.00	338.13	887.70	293.62	245.19
up_down_std	786.18	307.59	446.54	271.58	528.80	518.42	1201.58	326.38	281.02
down_up_min	262.00	201.00	181.00	83.00	232.00	136.00	209.00	244.00	123.00
down_up_max	2741.00	1841.00	1691.00	1156.00	2557.00	2674.00	3735.00	1163.00	1321.00
down_up_avg	889.50	404.07	514.38	444.85	559.85	488.96	1057.00	509.30	402.35
down_up_std	811.00	308.01	435.62	269.91	517.10	522.29	1185.60	322.91	292.62

Table 3.1: Keystroke Dataset

```

document.getElementById("writingPad").addEventListener("keydown", function(event){
    const currentTime = Date.now();
    const keyDownDistanceCount = keyDownDistance.length;
    const keyUpDistanceCount = keyUpDistance.length;
    if(keyDownDistanceCount > 0)
    {
        twoKeyDistance = currentTime - keyDownDistance[keyDownDistanceCount-1];
        keyDownLatency.push(twoKeyDistance);
    }
    if(keyUpDistanceCount > 0)
    {
        upDownDistance = currentTime - keyUpDistance[keyUpDistanceCount-1];
        keyUpDown.push(upDownDistance);
    }
    keyDownDistance.push(currentTime);
    if (!startKey[event.key])
    {
        startKey[event.key] = currentTime;
    }
});

```

Figure 3.12: Feature Extraction

```

document.getElementById("writingPad").addEventListener("keyup", function(event){
    const currentTime = Date.now();
    const keyUpDistanceCount = keyUpDistance.length;
    const keyDownDistanceCount = keyDownDistance.length;
    if(keyUpDistanceCount > 0)
    {
        twoKeyDistance = currentTime - keyUpDistance[keyUpDistanceCount-1];
        keyUpLatency.push(twoKeyDistance);
    }
    if(keyDownDistanceCount > 1)
    {
        downUpDistance = currentTime - keyDownDistance[keyDownDistanceCount-2];
        keyDownUp.push(downUpDistance);
    }
    keyUpDistance.push(currentTime);
    keyHoldTime = currentTime - startKey[event.key];
    startKey[event.key] = null;
    if(isNaN(keyHoldTime) == false)
    {
        keyHold.push(keyHoldTime);
    }
});

```

Figure 3.13: Feature Extraction

JavaScript is used to get features from HTML text boxes. Whenever a key is held, the current timestamp is selected, stored with the key id in an object, and when the key is released, the object value timestamp is deducted from the value with the current timestamp to calculate the duration. All other methods require setup of two arrays for down and up. If someone presses a key on the same key down function, check to see if the array length is greater than 0, then subtract that value from the current timestamp to calculate the down down time. By subtracting that same timestamp from the last up time registered in the array, one will find the result of up down. It has been implemented in the same manner for the release of keys. By subtracting the current timestamp from the last key up value, one can obtain the up up time, and by subtracting last to last key down, one can obtain the time difference between down and up.

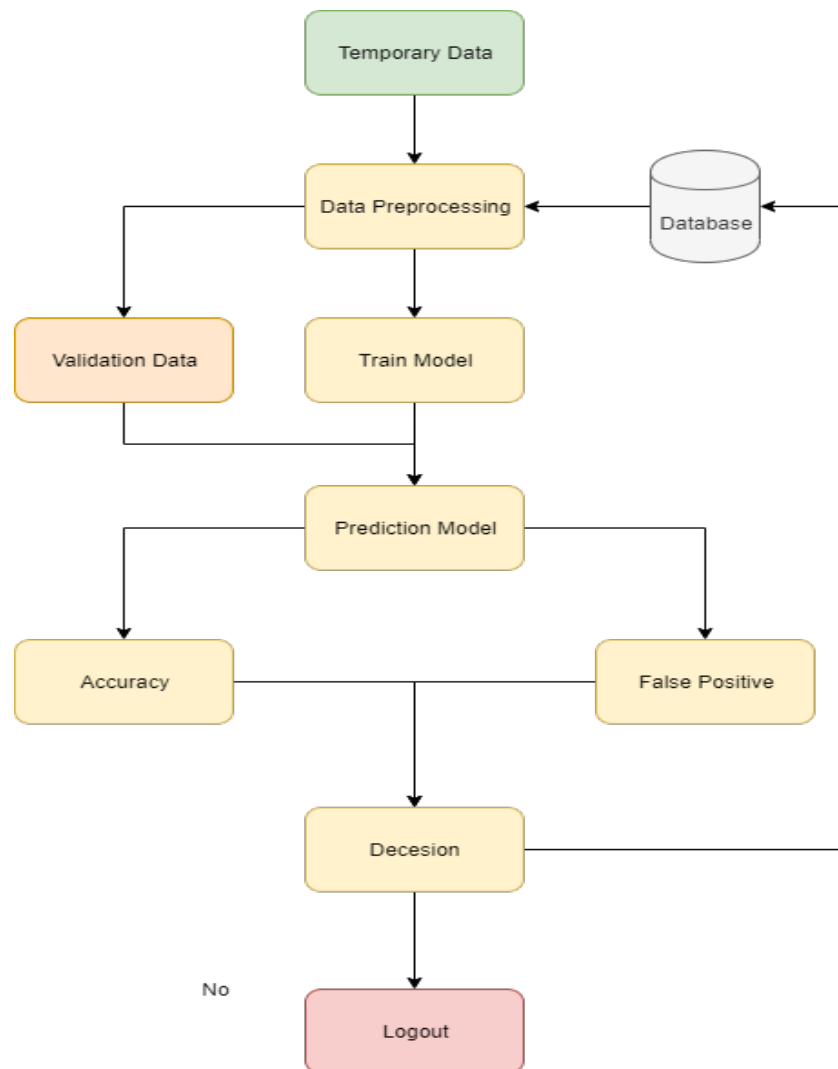


Figure 3.14: Data Flow Diagram

The Fig. 3.14 illustrates. Throughout the process of authentication, a small amount of data will be collected and sent to the validate model every three minutes at a time interval of ten seconds as part of the authentication component. Additionally, a set of data will be fetched every time an authentication request is sent to the server so that it can be processed. There will be two sets of data, one of which will verify the authenticity of user data, and another one of which will retrieve all the user data available in the database except the actual user data. The preprocessing part will clean the data and convert that data into data frame. It is easy to understand the normalization process, in which the values are normalized to a single place. Separated the labels from the dataset as well. After normalization data needs to prepare for the learning. So, the data split into two parts for both the databases. Testing was conducted with a number of 50 users due to the minimum user participation requirement, but the logic was designed to calculate the length of the actual user data as compared to the imposter data. The length of the actual user data will determine the length of the imposter data. Then, 50 percent of the data was used for training the model and the remainder was used for testing. In the application, it will be 100% of the data since the testing data or validation data will be derived from the actual session data. The labels of imposter datasets will be changed to imposter instead of user id. Clusters will be created after splitting. Prediction model and distance calculation are based on K-Nearest Neighbors. A cluster size of 5 has been set. A model validation is a procedure in which a set of data will be sent for model validation, and then the model will be predicted using the received accuracy and false positive accuracy information. A result that appears to be an authorized user will be sent to the database, and if it does not appear to be an authorized user, a response will be sent in return and the session will be cleared and a redirection to the logout page will occur.

```
def dataCollect(self):
    biometrics = keyCollection.find({'user':self.userId}) # Collect actual user data from database
    impBiometrics = keyCollection.find({'user':{'$nin': [self.userId] }}) # Collect rest of the users details

    self.dataFrame = pd.DataFrame.from_dict(biometrics, orient='columns') # Covert into pandas Dataframe
    self.dataFrame = self.dataFrame.mask(self.dataFrame==0).fillna(self.dataFrame.mean()) # Replace 0 with Mean

    self.impDataFrame = pd.DataFrame.from_dict(impBiometrics, orient='columns') # Covert into pandas Dataframe
    self.impDataFrame = self.impDataFrame.mask(self.impDataFrame==0).fillna(self.impDataFrame.mean()) # Replace 0 with Mean

    self.suspiciousData = pd.DataFrame.from_dict(self.data, orient='columns') # Covert into pandas Dataframe
    self.suspiciousData = self.suspiciousData.mask(self.suspiciousData==0).fillna(self.suspiciousData.mean()) # Replace 0 with Mean
```

Figure 3.15: Data Collect

In Fig.3.15 pandas were used to manage and structure the data. To stop overfitting or underfitting, replace 0 with the median of each column in the data cleaning part.



```

def normalize(self):
    dfFeatures = self.dataFrame.iloc[:,1:-1] # Select only the rows with data
    impdfFeatures = self.impDataFrame.iloc[:,1:-1] # Select only the rows with data
    suspiciousFeatures = self.suspiciousData.iloc[:, :-1] # Select only the rows with data

    self.dfNormalize = zscore(dfFeatures) # normalize the data with zscore normalization
    self.impDfNormalize = zscore(impdfFeatures) # normalize the data with zscore normalization
    self.suspiciousDataNormalize = zscore(suspiciousFeatures) # normalize the data with zscore normalization

    self.dfLabels = self.dataFrame.iloc[:, -1:] # select the lables of that data
    self.impDfLabels = self.impDataFrame.iloc[:, -1:] # select the lables of that data
    self.suspiciousLaels = self.suspiciousData.iloc[:, -1:] # select the lables of that data

```

Figure 3.16: Normalization

In Fig. 3.16 a first set of features was extracted from the dataset and the data was normalized using the equation 3.5 Z-Score method. Labels were taken from the actual dataset and separated out.

$$x' = \frac{x - \mu}{\sigma} \quad (3.5)$$

```

def trainig(self):

    scaler = StandardScaler() # Select Standard Scaler

    limit = 50 # Set a limit for the dataset to balance between two datasets

    print(f'Dataframe Count: {len(self.dfNormalize)}')
    print(f'Dataframe Label Count: {len(self.dfLabels)}')

    print(f'Imposter Count: {len(self.impDfNormalize)}')
    print(f'Imposter Label Count: {(self.impDfLabels)}')

    # Select a limited amount of dataset if it's less than that then it will select the length of the actual data set
    if len(self.dfNormalize) >= 50:
        self.dfNormalize = self.dfNormalize.sample(n = 50)
        self.dfLabels = self.dfLabels.sample(n = 50)

        self.impDfNormalize = self.impDfNormalize.sample(n = 50)
        self.impDfLabels = self.impDfLabels.sample(n = 50)
    else:
        self.impDfNormalize = self.impDfNormalize.sample(n = len(self.dfNormalize))
        self.impDfLabels = self.impDfLabels.sample(n = len(self.dfNormalize))

    print(f'Dataframe Count: {len(self.dfNormalize)}')
    print(f'Dataframe Label Count: {len(self.dfLabels)}')

    print(f'Imposter Count: {len(self.impDfNormalize)}')
    print(f'Imposter Label Count: {(self.impDfLabels)}')

    xTrain, xTest, yTrain, yTest = train_test_split(self.dfNormalize, self.dfLabels, test_size=0.2, random_state = 4) # Split actual dataset
    impXTrain, impXTest, impYTrain, impYTest = train_test_split(self.impDfNormalize, self.impDfLabels, test_size=0.6, random_state = 4) # Split imposter dataset

    tempData = self.labelReplace(impYTrain, 'imposter') # Replace imposter labels

    xTrain = xTrain.append(impXTrain, ignore_index=True) # Append imposter data into actual dataset
    yTrain = yTrain.append(tempData, ignore_index=True) # Append imposter labels into actual label

    # Set a random state
    xTrain = xTrain.sample(frac=1, random_state=1).reset_index(drop=True)
    yTrain = yTrain.sample(frac=1, random_state=1).reset_index(drop=True)

    scaler.fit(xTrain)
    xTrain = scaler.transform(xTrain) #real user train data

    self.knn = KNeighborsClassifier(n_neighbors=5) # Set Cluster Size
    self.knn.fit(xTrain, yTrain) # Fit model for training
    self.suspiciousDataNormalize = scaler.transform(self.suspiciousDataNormalize) # Scale suspicious data

```

Figure 3.17: Training Model

In Fig.3.17 a scaler is selected to fit the data. The calculation setup used the actual length of the data in order to avoid an unbalanced model from arising. After that, the actual data is split into XTrain and YTrain, and the imposter data is split into impXTrain and impYTrain. A random state has been chosen for a consistent result. Again scale the data for error redundancy. Imposter data labels set into imposter. xTrain merges with impXTrain and yTrain is merged with impYTrain. A cluster taken 5 for KNN. This stage involves training the model and scaling the temporary data.

```

def prediction(self):

    testLength = len(self.suspiciousDataNormalize) # Get the length of suspicious data

    self.yPred = self.knn.predict(self.suspiciousDataNormalize) # Predict with the suspicious dat

    confusionMatrix = confusion_matrix(self.suspiciousLaels, self.yPred) # Generate confusion matrix
    score = accuracy_score(self.suspiciousLaels, self.yPred) # Get the score of the prediction
    report = classification_report(self.suspiciousLaels, self.yPred , output_dict=True) # Generate Classification report

    reportDf = df = pd.DataFrame(report) # Convert that report into data frame

    falsePositive = confusionMatrix.sum(axis=0) - np.diag(confusionMatrix) # Calculate false postive
    support = int(100 * falsePositive[1]/ testLength) # Calculate support based on fasle positie
    score = int(score * 100)
    print(f'Support:{support},Score:{score}')
    # if support of imposter getter than 40 or score < 60 predict as an imposter
    if support > 40 or score < 60:
        print('imposter')
        return 'imposter'
    else:
        print('user')
        return 'user'

```

---

Figure 3.18: Prediction Model

In Fig.3.4, the length of the temporary data will be calculated. One confusion matrix will be generated based on the data sent for prediction. The accuracy of the result will be determined based on the results. Also classification will be generated and that will be converted into a data frame. In this way, a false positive will be removed and put into a condition where it will be detected as an imposter if the false positive support reaches 40%. The result will be the same even if the accuracy comes to 60% according to the user.

## Chapter 4

# Results

The result section is based on the analysis generated from collected data. 8-10 users participated in the data collection procedure. The results are obtained based on the analysis of the prediction for individual users.

### 4.1 Performance Measure of KNN

User	f1-score	Accuracy	Correct	Incorrect
User 1	75	60%	9	6
User 2	80	66%	10	5
User 3	93	86%	13	2
User 4	89	80%	12	3
User 5	87	76%	10	3

Table 4.1: Genuin User Data

As you can see in the Fig. 4.1, this is based on actual data used by the user for a machine learning training prediction. As a result, using the example of user 1, user got an accuracy of 60% with 9 correct predictions and 6 incorrect predictions. In the case of this particular user, the f1 score achieved was 75. There have already been challenges associated with the accuracy of the previous research. In addition to the data, there are a number of other factors that can affect it. It is possible that a user is wearing gloves while typing or may be injured or may only be using one hand while typing. To achieve a higher level of accuracy, data is necessary and user interaction will make the process of gathering true data more efficient. Earlier research has shown that to make a positive decision, a minimum of 100 keystrokes are needed. In the event that user data is incorporated well into the system, then there is a higher chance of accuracy coming into play. In similar fashion, users 4 and 5 received accuracy rates of 80% and 76%, respectively. A higher F1-score of 89 and 87 was received.

User	f1-score	Accuracy	Correct	Incorrect
User 1	33	80%	40	10
User 2	43	72%	36	14
User 3	83	28%	14	36
User 4	93	22%	44	6
User 5	95	10%	40	4

Table 4.2: Imposter Data

Fig. 4.2 shows the imposter data. The calculated in a different way because the labels were using genuine user labels that contained imposter data. Besides the logged in user, the rest of the users in the database behave as if they were imposters. If the system trained well with imposter data then the prediction will be most of the time imposter because they learned most of the features of imposter. The system should learn about users more correctly with imposter. The patterns are the key feature for the machine learning model. Here, user 1 and 2 received a good accuracy of 80% and 72%. The users data trained well to distinguish. As for the rest of the users, they were not well trained and were mostly detected as genuine users of the website.

#### 4.1.1 Keystroke Patterns

The Fig. 4.1 shows the data length refers to the length of the keystroke rows collected from the user, and frequency is determined by normalizing the keystroke values. Each pattern differs from the other. In spite of the fact that some patterns look similar, their fluctuation is different. As a result of using equation 4.1 MinMax normalization, the data have been normalized. A row of data is gathered from the user and the data varies from one row to another based on the type of data collected. In some cases, graphs show stable patterns that depict a stable writing pace and a steady writing pattern. If the user is unfamiliar with the keyboard or has difficulty finding the keys while typing, the data might differ depending on whether the user is a regular user of the keyboard. Although some graphs look similar to each other, they are entirely different from each other in terms of their content and shape. There is a huge difference between their starting point and their endpoint, as well as a vast difference between their fluctuation.

$$z_i = \frac{x_i - \min(x)}{\max(x) - \min(x)} \quad (4.1)$$

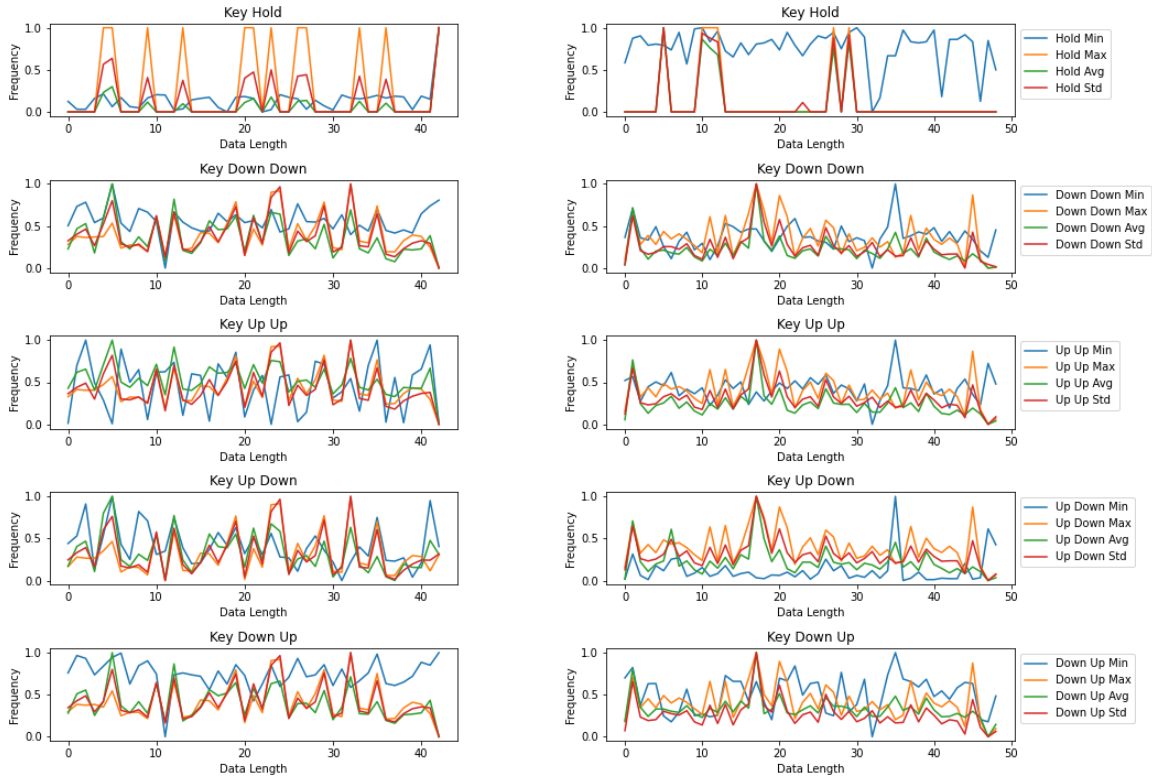


Figure 4.1: Keystroke Patterns of two user

#### 4.1.2 Confusion Matrix

It is shown in the confusion matrix whether there are true positives or false positives. As shown in the figure, it was initially detected as true 12 times and was originally a true label for genuine users. 3 times it was detected as an imposter but it was a genuine user. It is possible to improve the accuracy of the model by learning more about the user.

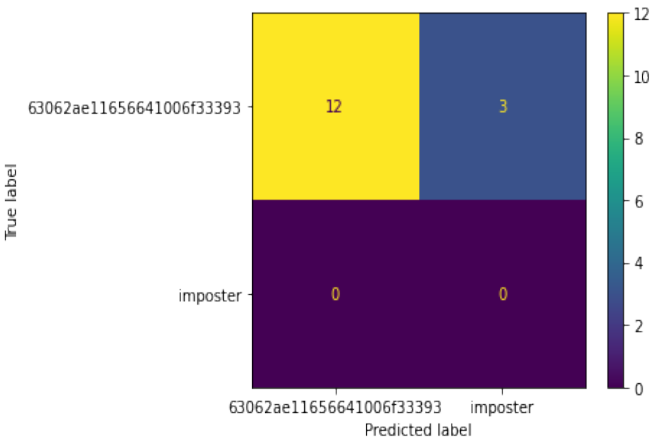


Figure 4.2: Genuine Matrix

There are two different labels available in the confusion matrix: user id and imposter, where in the case of false data the genuine label has been merged. It is illustrated in the Fig. ?? that 15 times the user was detected as a genuine user, whereas 35 times the user was detected as an imposter. This is a case in which the data is actually an imposter, so the prediction is an imposter as well. If imposter varieties are available and stored in the database, or if the imposter data is increased, the true positive of the model can be reduced in this section.

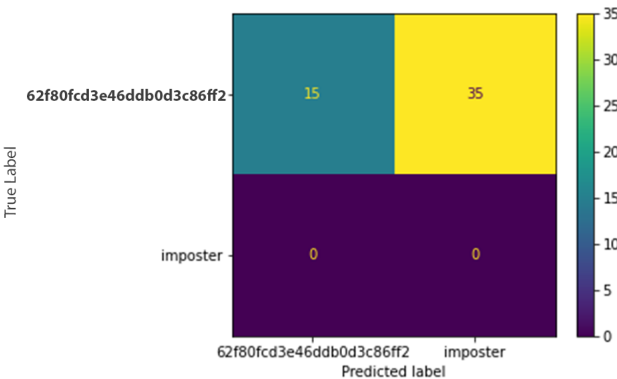


Figure 4.3: Imposter Matrix

A data structure has been built using the JSON format for collecting and storing data. As soon as the data is verified, it is sent to the database after authentication has been successful. As shown in the Fig. 4.4.

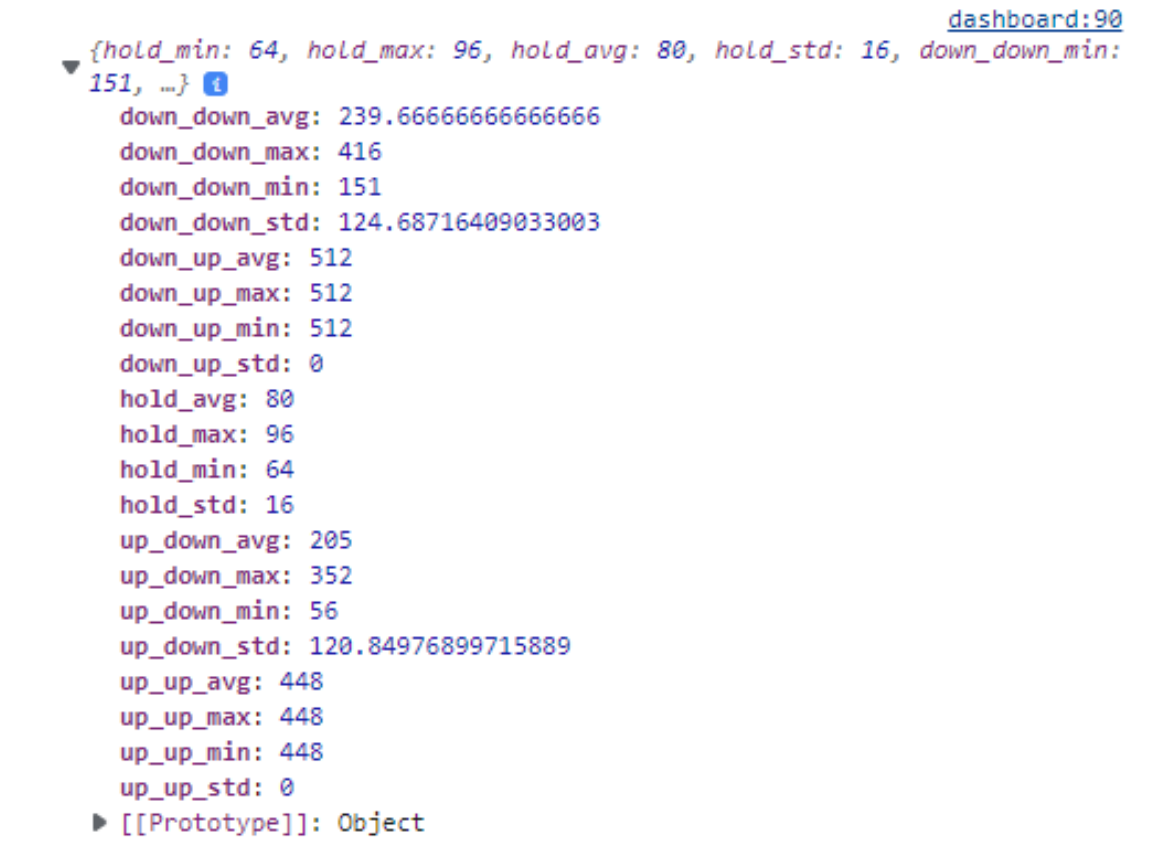


Figure 4.4: JSON Data Structure

As shown in Fig. 4.5, there is a text box where the user can write, while the data is being gathered in the background. Upon submitting a request, a response will be provided. It is expected that the response will continue if the stack is collecting. A successful authentication will be determined by the response to the authentication request. Otherwise, the system will send an imposter and reroute the session to the logout page.



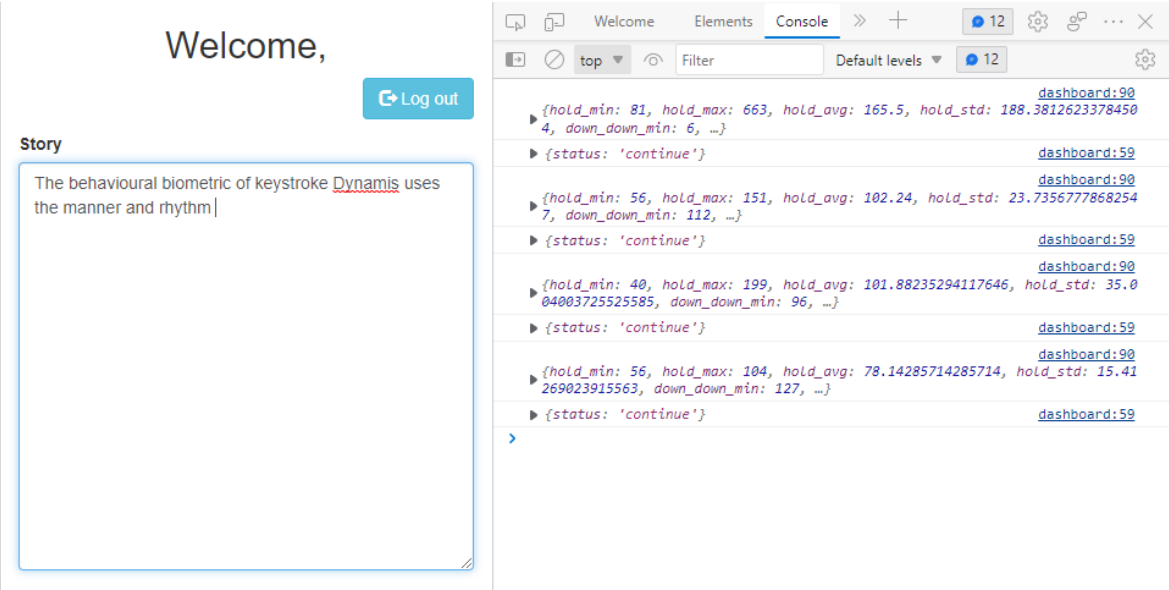


Figure 4.5: Authentication Writing UI

## Chapter 5

# Conclusions and Future Work

A number of research studies have been conducted in the past. Occasionally, the target was to analyse the user's behaviour or to verify whether the user was genuine without identifying the user or checking any additional verification. The initial stage of the training includes learning about keystrokes, as mentioned in Chapter 2. In order to verify a user, they stated that at least 100 keystrokes are required. A study was conducted on the patterns of password typing. Afterward, a research was conducted in 2009, and a dataset was generated based on the results of this research. In this study, more than 100 individuals participated, and the research took place over a period of two months.

As compared to previous research, the system is able to detect a person within 3 minutes with only 18 keystrokes, which is much faster than previous research methods. I think that the application can be improved if a greater number of people are able to participate in it. It is very important that a balance is maintained between genuine people and those who are imposters while learning. In the process of collecting data, there were several challenges that had to be overcome. Typing at a higher speed and collecting data at a higher speed will generate errors and large values. In addition, this affects the training model and makes it difficult to normalise the data in a meaningful way. A semi supervised learning approach is used to develop it. This model works as supervised learning, but the data changes on every run if the user is genuine. A reinforcement learning model can be developed after collecting effective data from users or an API can be developed so that any web application can use it.

The number of data required to make a professional application for companies depends on the type of user and the type of information they provide. There can be professional typist, occasional users of the keyboard, or people who type on a daily basis, but not a professional typist. Additionally, different types of keyboards have to be collected along with the data in a number of different scenarios. As soon as the research is completed, the data can be useful for authentication. It will take a long time to implement and fund the research. Researchers have already developed some applications for smartphones that have not been implemented in real life. Future web and mobile applications will be able to combine keystroke dynamics,

mouse dynamics, and touch sensitivity. There is a major challenge to implement to ensure that companies do not monitor their activities such as storing what they type and what they do in their daily activities. A major security concern for users can result from this situation. If this is implemented and companies begin storing user patterns, security measures must be taken to prevent replication in the future.

A variety of applications can be developed based on the keystroke dynamics. However, mouse dynamics must also be integrated with keystrokes for it to be effective for social media websites. It is important to consider mouse dynamics such as the x and y axis, button action, button type, and mouse speed. The system may also be implemented by companies in order to secure the internal system used by their employees. It is possible to prevent third parties from accessing the system. Detecting shell script users can also be solved with this technique. Hackers can be prevented from making use of it, and automated attacks can also be prevented. In order to accomplish that, the application must have full control over the DOM or it must be integrated with the operating system. Furthermore, if the model is not trained, then this can also pose a problem.

# Bibliography

- Adalier, M. and Teknik, A. (2015). Efficient and secure elliptic curve cryptography implementation of curve p-256. In *Workshop on elliptic curve cryptography standards*, volume 66, pages 2014–2017.
- Boicea, A., Radulescu, F., and Agapin, L. I. (2012). MongoDB vs oracle–database comparison. In *2012 third international conference on emerging intelligent data and web technologies*, pages 330–335. IEEE.
- Contributors, W. (2019). Authentication. [online] Wikipedia. Available at: <https://en.wikipedia.org/wiki/Authentication> [Accessed Jun. 24AD].
- Crockford, D. (2008). *JavaScript: The Good Parts: The Good Parts.* ” O’Reilly Media, Inc.”.
- Downey, A. (2012). *Think python.* ” O’Reilly Media, Inc.”.
- Géron, A. (2017). Hands-on machine learning with scikit-learn and tensorflow: Concepts. *Tools, and Techniques to build intelligent systems.*
- Grinberg, M. (2018). *Flask web development: developing web applications with python.* ” O’Reilly Media, Inc.”.
- Hastie, T., Tibshirani, R., Friedman, J. H., and Friedman, J. H. (2009). *The elements of statistical learning: data mining, inference, and prediction*, volume 2. Springer.
- inwedo (2022). What is token-based authentication? [online] inwebo. Available at: <https://www.inwebo.com/en/authentication-token/> [Accessed Jul. 27AD].
- Javatpoint (2021). (examples of ai (artificial intelligence) - javatpoint, 2022). [online] Available at: <https://www.javatpoint.com/examples-of-ai> [Accessed Aug. 30AD].
- José, I. (2018). (knn (k-nearest neighbors) 1). [online] Medium. Available at: <https://towardsdatascience.com/knn-k-nearest-neighbors-1-a4707b24bd1d> [Accessed 9 Sep. 2022].
- kaggle (2020). (100,000 uk used car data set). [online] <https://www.kaggle.com/datasets/adityadesai13/used-car-dataset-ford-and-mercedes> [Accessed 9 Sep. 2022].

- Killourhy, K. S. and Maxion, R. A. (2009). Comparing anomaly-detection algorithms for keystroke dynamics. In *2009 IEEE/IFIP International Conference on Dependable Systems & Networks*, pages 125–134. IEEE.
- Meyer, E. A. (2006). *CSS: The Definitive Guide: The Definitive Guide*. ” O’Reilly Media, Inc.”.
- Mhenni, A., Cherrier, E., Rosenberger, C., and Amara, N. E. B. (2018). Towards a secured authentication based on an online double serial adaptive mechanism of users’ keystroke dynamics. In *International Conference on Digital Society and eGovernments (ICDS)*.
- N-Able (2019). Understanding network authentication methods. [online+e] N-able. Available at: <https://www.n-able.com/blog/network-authentication-methods> [Accessed Jun. 24AD].
- Nixon, R. (2014). *Learning PHP, MySQL & JavaScript: With jQuery, CSS & HTML5*. ” O’Reilly Media, Inc.”.
- okta (2021). Authentication token: what is it? how does it work? [online] Available at: <https://www.okta.com/uk/identity-101/what-is-token-based-authentication/> [Accessed Jul. 28AD].
- Python, W. (2021). Python. *Python Releases for Windows*, 24.
- Raggett, D., Le Hors, A., Jacobs, I., et al. (1999). Html 4.01 specification. *W3C recommendation*, 24.
- Shepherd, S. (1995). Continuous authentication by analysis of keyboard typing characteristics. In *European Convention on Security and Detection, 1995.*, pages 111–114. IET.
- Siddiqui, N., Pryor, L., and Dave, R. (2021). User authentication schemes using machine learning methods—a review. In *Proceedings of International Conference on Communication and Computational Technologies*, pages 703–723. Springer.
- Vallat, R. (2018). Pingouin: statistics in python. *J. Open Source Softw.*, 3(31):1026.
- Van Rossum, G. and Drake Jr, F. L. (1995). *Python tutorial*, volume 620. Centrum voor Wiskunde en Informatica Amsterdam, The Netherlands.