USER AUTHENTICATION WITH KEYSTROKE DYNAMIC USING MACHINE LEARNING ALGORITHMS

Project work submitted to Avinashilingam Institute for Home Science and Higher Education for Women

MASTER OF SCIENCE IN INFORMATION TECHNOLOGY

SUBMITTED BY

SRIDEVI M

21PIT008

Under the Guidance of

Dr. D. SHANMUGAPRIYA, M.Sc., M.Phil., Ph.D., SET.,

Assistant Professor and Head

Department of Information Technology

AVINASHILINGAM INSTITUTE FOR HOME SCIENCE AND HIGHER EDUCATION FOR WOMEN

SCHOOL OF PHYSICAL SCIENCES AND COMPUTATIONAL SCIENCES

DEPARTMENT OF INFORMATION TECHNOLOGY

COIMBATORE-641043

MAY 2022

DECLARATION

DECLARATION

I hereby declare that the project entitled **"User Authentication With Keystroke Dynamic Using Machine Learning Algorithms"** is a record of the original work done by SRIDEVI M(21PIT008) under the guidance of Dr. D. Shanmugapriya MSC., M.Phil., Ph.D., Assistant Professor and Head, Department of Information Technology, School of Physical Sciences and Computational Sciences, Avinashilingam Institute for Home Science and Higher Education for Women inthe partial fulfillment for the award of the degree of Master of Science in Information Technology, and this project work has not formed the basis for any Degree/Diploma/Associates.

Place: Coimbatore

Date: 19/05/2023

Signature of the candidate

Countersigned By,

Dr.D.Shanmugapriya, M.Sc., M.Phil., PhD., SET.,
Assistant Professor and Head,
Department of Information Technology,
School of Physical Sciences and Computational Sciences.

CERTIFICATE

CERTIFICATE

Avinashilingam Institute for Home Science and Higher Education for Women

(Deemed to be University under Estd. u/s 3 of UGC Act 1956, Category 'A' by MHRD)

Re-accredited with 'A++' Grade by NAAC. CGPA 3.65/4, Category 1 University by UGC

Coimbatore - 641 043, Tamil Nadu, India

DST - CURIE - AI Sponsored

Centre for Cyber Intelligence

CERTIFICATE OF PROJECT COMPLETION

This is to certify that Ms. Sridevi M (21PIT008), Master of Information Technology, Avinashilingam Institute for Home Science and Higher Education for Women, has successfully completed the project entitled "Multi-Factor Authentication with Keystroke Dynamics and Image CAPTCHA using Machine Learning Methods" under Centre for Cyber Intelligence - Centre for Machine Learning and Intelligence - a DST - CURIE - AI facility during December 2022 - May 2023.

Dr. G. Padmavathi
Dean, School of PSCS
CCI - Principal Investigator

Dr. P. Subashini
Project Coordinator - DST - CURIE - AI

Dr. S. Kowsalya
Registrar

CERTIFICATE

This is to certify that this project work entitled "**User Authentication with Keystroke Dynamic Using Machine Learning Algorithms**" done by **SRIDEVI M**(21PIT008) has been submitted to Avinashilingam Institute for Home Science and Higher Education for Women, Coimbatore-641 043 in partial fulfillment of the requirement for the award of the degree of **MASTER OF SCIENCE IN INFORMATION TECHNOLOGY.** This Project has not found the basis for the award of any Degree/Associate/fellowship or similar title to any Candidate of any University. Certified as a Bonafide record of the work submitted for the Viva-voce held on 25/5/23

Signature of the Head of the Department

Signature of the Supervisor

Signature of the Examiner(s)

ACKNOWLEDGEMENT

ACKNOWLEDGEMENT

I owe my sincere thanks to Lord Almighty and My lovable parents for showering their generous blessing supreme in all endeavors.

I wish to express my gratitude to **Prof. S.P. Thyagarajan**, Chancellor, Avinashilingam Institute for Home Science and Higher Education for Women, Coimbatore, for providing the facilities to conduct this study.

I extend my thanks to **Dr. Bharathi Harishankar**, Ph.D., FRSA., Vice Chancellor, Avinashilingam Institute for Home Science and Higher Education for Women, Coimbatore, for providing flamboyant help towards the completion of the study.

I record my deep sense of gratitude and indebtedness to **Dr. S. Kowsalya**, M.Sc., M.Phil., Ph.D., and Registrar, Avinashilingam Institute for Home Science and Higher Education for Women, Coimbatore, for providing adequate help for the study

I grateful record my sincere thanks to **Dr. G. Padmavathi** M.Sc., M.Phil., Ph.D., Dean and Professor, School of Physical Sciences & Computational of Sciences, Avinashilingam Institute for Home Science and Higher Education for Women, Coimbatore, for timely help rendered throughout the course of this work.

I heartily Thank my esteemed project guide **Dr. D. Shanmugapriya** MSC.,M.Phil., Ph.D., , Head of the Department Assistant Professor Department of Information Technology, for imparting tremendous assistance and well-timed support for the triumph four projects.

I express my honorable thanks to our project coordinator Department of Information Technology, for imparting tremendous assistance and well-timed support for the triumph of our project.

I sincerely thank all the staff members of the Department of Information Technology, Avinashilingam Institute for Home Science and Higher Education for Women, Coimbatore, for their help and support.

I like to extend my gratitude to Ms.A.Roshini, Technical Assistant –Center of cyber intelligence, Department of Computer Science, For providing Project guidelines and always supported me and encouraged me with valuable advice and Profound belief in my work and abilities-CURIE – AI Sponsored Phase II for providing the laboratory facilities to execute my project. I would like to acknowledge the help rendered by Center for Cyber Intelligence, DST.

I would like to express my special thanks to my parents, my friends and all my well-wishers for their constant encouragement, support and help in carrying out this work successfully.

ABSTRACT

INTRODUCTION

ABSTRACT

Keystroke dynamics is a biometric authentication technique, this technique uses the unique patterns and rhythms of an individual's typing behavior to create a unique typing pattern that can be used to authenticate users. When used in combination with other authentication factors such as passwords or fingerprint recognition, keystroke dynamics can provide an additional layer of security that makes it much more difficult for attackers to gain access to sensitive systems or data.

This project aims to concentrate on user authentication Keystroke Dynamic using Support Vector Machine (SVM) and Manhattan Distance Filter (MDF) algorithm to detect the imposter in the data. The Keystroke dataset CNS-0430474 is used to identify the imposters using user authentication mechanism based on machine learning methods. The dataset consists of 20400 instances, 60 features with 51 subjects. The dataset is the benchmark one provided by CMU (2009) repository.

The developed machine learning models SVM & MDF are evaluated using Equal Error Rate (EER) parameter to identify the significant one. From this evaluation, the SVM based user authentication model performs better in identifying genuine user with low equal error rate of 0.18.

Keywords: Equal Error Rate, Genuine user, Imposter User, Keystroke Dynamic, User Authentication.

TABLE OF CONTENT

CHAPTER-I

1.INTRODUCTION

1.1 Biometric System:

A biometric system allows for the systematic recognition of an individual by using some form of distinguishing feature or characteristic of human body. Fingerprints, facial traits, voice traits, geometry of hand, handwriting, the iris, and the retina, have all been used to construct a biometric system.
Biometric systems begin by obtaining the sample feature, such as a sound signal captureddigitally for voice recognition or a digital captured color image for face recognition.

The obtained sample is converted as a biometric template using a mathematical formula. It will generate are presentation of the characteristic that is normalized, efficient, and discriminating, which can be compared objectively to other templates to ascertain identity. Most biometric systems have two operating modes. An enrollment mode for adding templates to a database, andan identification mode for creating a template for a person and then searching the database of pre-enrolled templates for a match.
Easier fraud detection

- Better than password/PIN or smart cards.
- No need to remember passwords
- Requires actual presence of the individual to be identified
- Unique physical or behavioral trait and cannot be borrowed, stolen, or forgotten.

1.1.1 Biometrics Types

There are two types of Biometrics. They are described in the following figure 1.

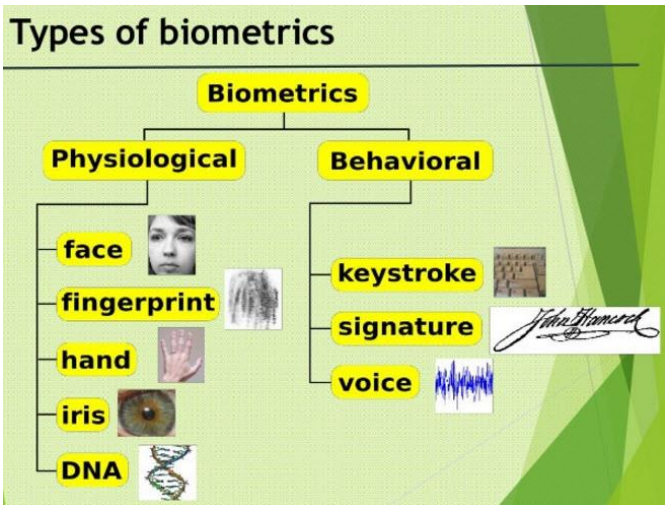


Figure 1.1 Types of Biometrics

1.2 User Authentication

Authentication is the process of identifying users that request access to a system, network, or device. Access control often determines user identity according to credentials like username and password. Other authentication technologies like biometrics and authentication apps are also used to authenticate user identity.

User authentication is a method that keeps unauthorized users from accessing sensitive information. Cybercriminals can gain access to a system and steal information when user authentication is not secure. The data breaches companies like Adobe, Equifax, and Yahoo faced are examples of what happens when organizations fail to secure their user authentication. Hackers gained access to Yahoo user accounts to steal contacts, calendars and private emails between 2012 and 2016.

1.2.1 Levels of Authentication

Cybercriminals always improve their attacks. As a result, security teams are facing plenty of authentication-related challenges. This is why companies are starting to implement more sophisticated incident response strategies, including authentication as part of the process. The list below reviews some common authentication methods used to secure modern systems.

The levels of Authentication are listed below:

- Single-factor authentication (SFA)
- Two-factor authentication (2FA)
- Multi-factor authentication (MFA)

Single-factor authentication (SFA)

Single-factor authentication is the simplest form of authentication method. With SFA, a person matches one credential to verify himself or herself online. The most popular example of this would be a password (credential) to a username. Most verification today uses this type of authentication method.

Two-factor authentication (2FA)

Two-factor authentication uses the same password/username combination, but with the addition of being asked to verify who a person is by using something only he or she owns, such as a mobile device. Putting it simply: it uses two factors to confirm an identity.

Multi-factor authentication (MFA)

MFA authentication methods and technologies increase the confidence of users by adding multiple layers of security. MFA may be a good defence against most account hacks, but it has its own pitfalls. People may lose their phones or SIM cards and not be able to generate an authentication code.

1.3 Keystroke Dynamics

Keystroke dynamics or typing biometrics refers to the automated method of identifying or confirming the identity of an individual based on the manner and the rhythm of typing on a keyboard. **Keystroke dynamics** is a behavioral biometric, this means that the biometric factor

Keystroke dynamics is the study of whether people can be distinguished by their typing rhythms, much like handwriting is used to identify the author of a written text. Possible applications include acting as an electronic fingerprint, or in an access-control mechanism.

Keystroke dynamics, or typing dynamics, is the detailed timing information that describes exactly when each key was pressed and when it was released as a person is typing at a computer keyboard.

Keystroke dynamics can run into two modes:

- Identification
- Verification.

1.4 About the Project

In user authentication, keystroke dynamics can be used in combination with other authentication factors, such as passwords or fingerprint recognition, to create a more robust and secure authentication process.

Keystroke dynamics can be applied in a variety of scenarios, including login systems for computers, mobile devices, and web applications, as well as physical access control systems that provides an additional layer of security without requiring additional hardware or user effort.

It also analyzes the timing and duration of keystrokes, as well as the pauses between keystrokes, to create a unique typing pattern that can be used to authenticate users using dwell time, flight time, digraph, trigraph in the dataset.

This project investigate the CNS-0430474 benchmark dataset to identify the genuine and imposter users with keystroke dynamic using machine learning methods.

1.5 Motivation and Justification

Biometric-based authentication is categorized into the physiological and behavioral property of the user. The physiological property covers the visible part of the human body such as the retina, fingerprint, etc. On the other hand, behavioral property analyzes the behavior of a user through user profiling, gait, mouse dynamics, keystroke dynamics, etc.

These unique behavior properties can be used to enhance the user verification process and develop a multi-model user authentication system. For instance, by implementing keystroke dynamics alongside with password-based authentication system, the impostor will not only need to obtain the knowledge of the password but also the knowledge of how the password is typed. Thus, better security is provided by using multi-modal user authentication.

Keystroke dynamics is a user authentication method, which validates the user's typing rhythm to allow access to the system. It is an emerging field of interest for security especially in user authentication due to its various advantages.

Keystroke authentication is a biometric authentication technique that uses the unique typing patterns of an individual to verify their identity. Support Vector Machines (SVMs) are a popular machine learning algorithm that can be used for classification tasks, including keystroke authentication.

Overall, keystroke authentication using SVM and Manhattan Project is a powerful and effective way to verify the identity of users based on their typing patterns

1.6 Problem Statement

Using keystroke dynamic authentication in conjunction with machine learning techniques for authenticating the genuine users.

1.7 Objective

To authenticate genuine/imposter user with Keystroke Dynamic using Machine Learning Techniques.

1.8 Statistics

The following are some of the recent statistics about the impact of User Authentication and its security are described below

By the end of 2021, Google auto-enrolled 150 million users into using two-factor authentication to access their accounts. As reported by 9 to5 Google, the move saw a 50% decline in compromised accounts. This huge achievement speaks volumes about the positive impact 2FA can have on a login process versus using a traditional username and password to authenticate.

In 2021, LastPass found that businesses in the technology and software industry were the most proactive in using MFA, with 39% of respondents stating they were already using it. Education closely followed with 33%, but worryingly, the industries that handle some of the most sensitive customer data – legal and insurance had much less uptake of MFA, both with 20% of employees using the authentication method.

The average cost of recovery from eCommerce fraud was set to reach $6.4 billion by 2021.

61% of people reuse the same password across multiple accounts

In Twitter's 2021 transparency report, the social media giant revealed that its two-factor authentication method in its reporting period between July and December 2020, a meager 2.5% of users adopted 2FA, a rise of 8.7%.

Duo Labs reports that the most concerning industry were it to be hacked would be related to finance – backed by 93% of respondents.

Email and social media were the following most essential categories with 58% and 40% of responses, respectively.

1.9 Contribution Of the Project

The Contribution of this project is to identify genuine/imposter user in the taken dataset CNS-0430474 using SVM and MDF in Machine Learning algorithms. Also the taken dataset has been enhanced by adding some of the Keystroke Dynamic features such as dwell time, flight time, digraph

and trigraph properties by adding these significant properties contribute the user authentication effectively.

After the introduction the document is organised as Chapter-II deals with system configuration, Chapter - III presents the recent research work carried out with user authentication with keystroke dynamic using machine learning methods, Chapter – IV explains the step by step methodology applied to develop the suitable model for User Authentication, Chapter-V entitles the results obtained from each phases of the project , Chapter-VI concludes with Future Scope of the project.

CHAPTER-II

2. SYSTEM CONFIGURATION

2.1 Hardware requirement

Processor: intel i7 and an above ram: 8 GB

Hard disk capacity: 1TB

2.2 Software requirement

Operating system: windows10

package: anaconda

Programing platform: python

2.3 About the environment

In this project the following Python environment is used to develop a significant model

The tools used in this project are listed below.

- Python 3.11.3
- Anaconda

2.3.1 Anaconda

Rather than using command lines, Anaconda Navigator's graphical interface can be used to install packages, manage environments, and run standard Python tasks. Moreover, it enables easy management of channels, environments, and packages for Anaconda without the need for command-line input. On the Anaconda Cloud or in a local Anaconda Repository, Navigator can search for packages. Windows, macOS, and Linux are all supported**.**

2.3.2 Jupyter Notebook

An open-source web program called the Jupyter Notebook allows users to create and share documents with live code, equations, visualizations, and text. The folks who work on Project Jupyter maintain Jupyter Notebook. The I-Python project, which once had its own I-Python Notebook project, is where Jupyter Notebooks got their start. Jupyter's name is derived from the three primary programming languages it supports: Julia, Python, and R. There are already more than 100 additional kernels available; however, Jupyter comes with the I-Python kernel, which enables Python programmed writing.

2.33 Some of the Python package and libraries involved in this project to develop the dep learning models

Python Package

a) Scikit-Learn

Scikit-learn, a free Python package that is frequently seen as a direct extension of SciPy, isbased on NumPy and SciPy. It is specially made for creating supervised and unsupervised machine learning algorithms and data modeling.

Scikit-learn is user-friendly and beginner-friendly because of its straightforward, intuitive, and consistent interface. Scikit-learn performs admirably by enabling users to alter and exchange data as they need, despite the fact that its utility is constrained because it only excels at data modelling

b) Pandas

Python's Pandas package for data research and analysis enables programmers to create simple, seamless high-level data structures. Pandas, which is based on NumPy, is in charge of getting data sets and data points ready for machine learning. Pandas uses one-dimensional (series) and two-dimensional (Data Frame) data structures. These two types of data structures allow Pandas to be used in a range of industries, from science and statistics to banking and engineering.

Due to its adaptability, the Pandas library can be used with other scientific and numerical libraries. Because they are rapid, compliant, and highly descriptive, their data structures are simple to use. By aggregating, integrating, and re-indexing data with Pandas, one can modify data functionality with a minimum of keystrokes.

c) Matplotlib

Matplotlib is a data visualization library that is used for making plots and graphs. It is an extension of SciPy and is able to handle NumPy data structures as well as complex data models made by Pandas. Although it is expertise is limited to 2D plotting, Matplotlib can produce high-quality and publish-ready diagrams, graphs, plots, histograms, error charts, scatter plots, and bar charts.

Matplotlib is intuitive and easy to use, making it a great choice for beginners. It is even easier to use for people with pre-existing knowledge of various other graph-plotting tools. It offers GUI toolkits support, including wxPython, Tkinter, and Qt.

d) NumPy

Open-source and well-known Python library for numbers, NumPy. It is capable of carrying out a wide range of mathematical operations on matrices and arrays. One of the most popular libraries for scientific computing, it is frequently used by scientists to analyze data.

It is perfect for machine learning and artificial intelligence (AI) projects since it can process multidimensional arrays, handle linear algebra, and perform Fourier transformation. 10

NumPy arrays demand a considerable reduction in storage space when compared to standard Python lists. They are also lot easier to operate and considerably faster than the earlier. One can reshape, transpose, and modify data in matrix form with NumPy. Combining NumPy's capabilities makes it easy to enhance the machine learning model's performance.

e) Seaborn

An open-source Python package for data visualization and graphing is called Seaborn. It uses sophisticated Panda's data structures and is based on the graphing software Matplotlib. Seaborn offers a high-level, feature-rich interface for creating precise, illuminating statistical graphs on its own. Because it can produce logical graphs of learning and execution data, it is employed in machine learning and deep learning applications.

The most beautiful and eye-catching graphs and plots are produced by Seaborn, which makes it ideal for use in publishing and marketing. Seaborn can also save you time and effort because it enables you to build complex graphs with little code and basic instructions.

f) Train-Test Split

The train-test split is used to determine how well machine learning algorithms work when employed with prediction-based methods and applications. To compare the output of one's own machine learning model, one can use this quick and simple procedure.

CHAPTER-III

3.REVIEW OF LITERATURE

The chapter 2 covers the literature review done towards the proposed system. The following table 3 consists of the title, techniques and the observation of the research in the literature survey.

S.No	Title of the Article	Author & Year	Dataset Used	Algorithm Used	Accuracy Obtained
1.	An improved user identification based on keystroke-dynamics and transfer learning.	Tewari, Anurag, and Prabhat Verma. (2022).	pre-trained models are utilized by applying fine-tuning	Recognition based system	98.57% accuracy
2.	Detecting human attacks using the keystroke dynamic approach.	Alsuhibany, Suliman A., and Latifah A. Alreshoodi.(2021)	Novel defence system using the keystroke dynamic approach	Automated Public Turing Test to Tell Computers and Humans Apart was introduced	100% detection rate
3.	Investigation of using keystroke dynamics to enhance the prevention of phishing attacks.	Alamri, Emtethal K., Abdullah M. Alnajim, and Suliman A. Alsuhibany. (2022).	keystroke dynamics dataset	Highly effective in protecting online services from phishing attacks	0% false-positive rate and 17.8% false-negative rate

4.	Contributions to keystroke dynamics for privacy and security on the Internet.	Migdal, Denis. Diss. Normandie University (2019).	pre-trained models dataset	To protect information from malicious websites	97% detection rate
5.	Comparing anomaly-detection algorithms for keystroke dynamics.	Killourhy, Kevin S., and Roy A. Maxion. (2009).	Keystroke-dynamics data set	Anomaly detectors for password timing	equal-error rates between 9.6% and 10.2%
6.	Keystroke biometric systems for user authentication.	Ali, Md Liakat, et al. (2017)	Survey of the most recent researches on keystroke dynamic authentication	Identifies some issues that need to be addressed in designing keystroke dynamic biometric systems	Overall accuracy of KB is still lower than other biometric authentication systems
7.	Keystroke-Based User Authentication System Using Support Vector Machine	Singh and Saxena. (2012)	Tested on a dataset of 100 users.	SVM algorithm and a Manhattan Distance filter	Accuracy rate of 97.5%
8.	Keystroke dynamics based user authentication using deep multilayer perceptron	Andrean, Alvin, Manoj Jayabalan, and Vinesh Thiruchelvam (2020)	Deep learning model on keystroke dynamics dataset	Deep learning model using Multilayer Perceptron	MLP achieved optimum EER of 4.45%

					compared to original benchmark classifiers such as 9.6% (scaled Manhattan)
9.	User Authentication System Based on Keystroke Dynamics Using SVM and Euclidean Distance	Yousfi et al. (2012).	Dataset of 30 users using supervises learning	SVM algorithm with Manhattan Distance filter	Achieved the highest accuracy rate of 97.72%
10.	KeyDetect-Detection of anomalies and user based on Keystroke Dynamics	Kar, Soumyatattwa, et al. (2023).	Data set of 51 users typing a password in 8 sessions	Keystroke dynamics (typing pattern) of a user	Accuracy of 95.05%

CHAPTER-IV

4. METHODOLOGY

This section explains the step by step methodology used to develop a machine learning model for User Authentication with Keystroke Dynamic to identify Genuine/Imposter user in the data. Figure 4 shows the proposed methodology applied for the project.

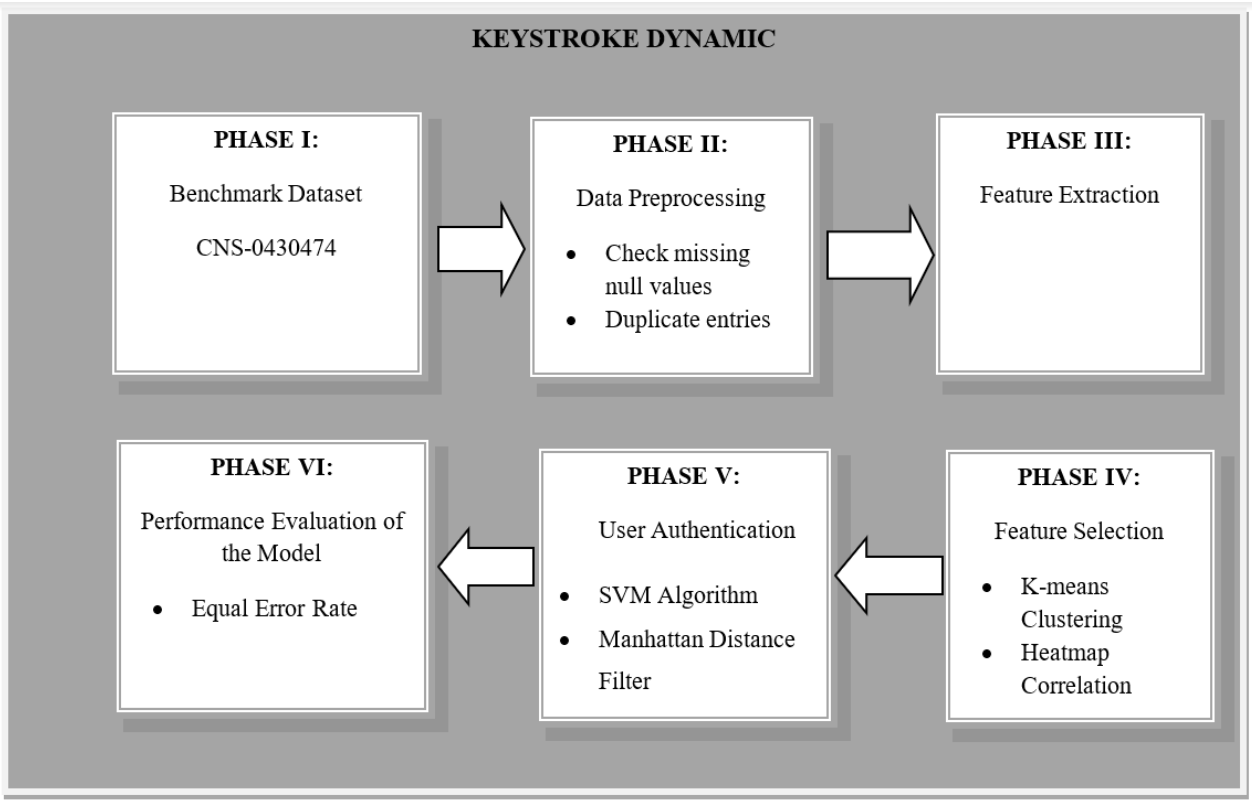


Figure 4.1 Proposed Methodology

4.1 Data Collection

The process of collecting data include compiling datasets from relevant sources in order to evaluate and test the suggested system. This dataset was provided by the National Science Foundation under grant numbers CNS-0430474 and CNS-0716677, and by the Army Research Office through grant number DAAD19-02-1-0389 to Carnegie Mellon University's CyLab. The datasets are applied for Comparing Anomaly-Detection Algorithms for Keystroke Dynamics in (DSN-2009).

The data consist of keystroke-timing information from 51 subjects (typists) 60 features with 20400 instances was attempted by the typists by writing a passwords as .tie5Roanl for 400 time to capture the user behaviour with different keystrokes.

Every time the subject presses or releases a key, the software application logs the keystroke event (key down or key up), the key's name, and the timestamp for when it happened. To create incredibly precise timestamps, an external reference clock was used. The reference clock was shown to be accurate to within 200 microseconds (by simulating key presses at predetermined intervals with a function generator).

Dataset Description Dataset Name: CNS-0430474
Number of Instances: 20400
Number of Features: 60 features and 51 subjects
Link: https://www.cs.cmu.edu/~keystroke/

4.1.1 Dataset Feature Description

The features of the dataset are described the feature in Table 4.1

Table 4.1 Dataset Feature Description

S.No	Features of the Dataset	Description
1	Subject	Subject ID or class label for 51 users involved in typing task.
2	sessionIndex	A number of the session in the typing task; consists of 8 sessions in total.
3	Rep	A number of repetition in the typing task; consists of 50 repetitions for each session.
4	H.period	The duration between pressing and releasing '.' key.
5	DD.period.t	The duration between pressing '.' key and pressing 't' key.
6	UD.period.t	The duration between releasing '.' key and pressing 't' key.
7	H.t	The duration between pressing and releasing 't' key.
8	DD.t.i	The duration between pressing 't' key and pressing 'i' key.
9	UD.t.i	The duration between releasing 't' key and pressing 'i' key.
10	H.i	The duration between pressing and releasing 'i' key.
11	DD.i.e	The duration between pressing 'i' key and pressing 'e' key.
12	UD.i.e	The duration between releasing 'i' key and pressing 'e' key.

13	H.e	The duration between pressing and releasing ‘e’ key.
14	DD.e.five	The duration between pressing ‘e’ key and pressing ‘five’ key.
15	UD.e.five	The duration between releasing ‘e’ key and pressing ‘five’ key.
16	H.five	The duration between pressing and releasing ‘.’ key.
17	DD.five.shift.r	The duration between pressing ‘five’ key and pressing ‘shift.r’ key.
18	UD.five.shift.r	The duration between releasing ‘five’ key and pressing ‘shift.r’ key.
19	H.shift.r	The duration between pressing and releasing ‘r’ key.
20	DD.shift.r.o	The duration between pressing ‘shift.r’ key and pressing ‘o’ key.
21	UD.shift.r.o	The duration between releasing ‘shift.r’ key and pressing ‘o’ key.
22	H.o	The duration between pressing and releasing ‘o’ key.
23	DD.o.a	The duration between pressing ‘o’ key and pressing ‘a’ key.
24	UD.o.a	The duration between releasing ‘o’ key and pressing ‘a’ key.
25	H.a	The duration between pressing and releasing ‘a’ key.
26	DD.a.n	The duration between pressing ‘a’ key and pressing ‘n’ key.
27	UD.a.n	The duration between releasing ‘a’ key and pressing ‘n’ key.
28	H.n	The duration between pressing and releasing ‘n’ key.
29	DD.n.l	The duration between pressing ‘n’ key and pressing ‘l’ key.
30	UD.n.l	The duration between releasing ‘n’ key and pressing ‘l’ key.
31	H.l	The duration between pressing and releasing ‘l’ key.
32	DD.l.return	The duration between pressing ‘l’ key and pressing ‘return’ key.
33	UD.l.return	The duration between releasing ‘l’ key and pressing ‘return’ key.
34	H.return	The duration between pressing and releasing ‘return’ key.

4.2 Data Pre-processing

Data pre-processing is a vital phase in machine learning that improves the quality of the data to promote the extraction of valuable insights from the data. Preparing (cleaning and arranging) raw data in order to make it acceptable for creating and training Machine Learning Model.

It raises reliability and accuracy. Pre- processing data can increase the correctness and quality of a dataset, making it more stable by removing missing or inconsistent data values brought on by human or computer mistake. It ensures consistency in data.

4.2.1 Types of Data Pre-processing

1. Acquire the dataset
2. Import all the crucial libraries
3. Import the dataset
4. Identifying and handling the missing values
5. Encoding the categorical data
6. Splitting the dataset

4.2.2 Data Pre- Processing Techniques Implemented In the Datasets

a) Handling Null Values

There are almost never any null values in a real-world dataset. No model can handle these NULL or NaN values on its own, thus we must step in regardless of whether the issue is one of regression, classification, or any other kind. Python represents NULL with NaN. So, don't mix the two as they can be used alternately. Methods Handled for Null values are

- Impute missing values for continuous variable
- Impute missing values for categorical variable
- Other Imputation Methods
- Using Algorithms that support missing values
- Prediction of missing values
- Imputation using Deep Learning Library
- Deleting Rows with missing values

Dropping the null-valued rows or columns is the simplest solution to this issue.
If the missing values are not handled correctly, you can wind up creating a machine learning model that is biased and produces inaccurate results. Missing data can make the statistical analysis less precise.

b) Removing Duplicate Values

A dataset contains many instances of a duplicate value. It is frequently discovered when using Excel to work with huge databases.

Data processing will be unsuccessful if duplicate records are not eliminated. The goal of this control is to eliminate multiple records from the dataset in order to make it ready for further processing.

Since the dataset is the standard data it does not contain any irrelevant values or duplicate entries.

4.3 Feature Extraction

Feature engineering is the pre-processing step of machine learning, which extracts features from raw data**.** It helps to represent an underlying problem to predictive models in a better way, which as a result, improve the accuracy of the model for unseen data. The predictive model contains predictor variables and an outcome variable, and while the feature engineering process selects the most useful predictor variables for the model.

Feature engineering is the process of selecting and transforming variables when creating a predictive model using machine learning. It's a good way to enhance predictive models as it involves isolating key information, highlighting patterns and bringing in someone with domain expertise.

These processes are described as below:

1. **Feature Creation**: Feature creation is finding the most useful variables to be used in a predictive model. The process is subjective, and it requires human creativity and intervention. The new features are created by mixing existing features using addition, subtraction, and ration, and these new features have great flexibility.
2. **Transformations**: The transformation step of feature engineering involves adjusting the predictor variable to improve the accuracy and performance of the model. For example, it ensures that the model is flexible to take input of the variety of data; it ensures that all the variables are on the same scale, making the model easier to understand. It improves the model's accuracy and ensures that all the features are within the acceptable range to avoid any computational error.
3. **Feature Extraction**: Feature extraction is an automated feature engineering process that generates new variables by extracting them from the raw data. The main aim of this step is to reduce the volume of data so that it can be easily used and managed for data modelling. Feature extraction methods include cluster analysis, text analytics, edge detection algorithms, and principal components analysis (PCA).
4. **Feature Selection:** While developing the machine learning model, only a few variables in the dataset are useful for building the model, and the rest features are either redundant or irrelevant. If we input the dataset with all these redundant and irrelevant features, it may negatively impact and reduce the overall performance and accuracy of the model. Hence it is very important to identify and select the most appropriate features from the data and remove the irrelevant or less

important features, which is done with the help of feature selection in machine learning. "Feature selection is a way of selecting the subset of the most relevant features from the original features set by removing the redundant, irrelevant, or noisy features.

4.3.1 Feature extraction can be accomplished manually or automatically:

- Manual feature extraction requires identifying and describing the features that are relevant for a given problem and implementing a way to extract those features. In many situations, having a good understanding of the background or domain can help make informed decisions as to which features could be useful.
- Over decades of research, engineers and scientists have developed feature extraction methods for images, signals, and text. An example of a simple feature is the mean of a window in a signal.
- Automated feature extraction uses specialized algorithms or deep networks to extract features automatically from signals or images without the need for human intervention. This technique can be very useful when you want to move quickly from raw data to developing machine learning algorithms. Wavelet scattering is an example of automated feature extraction.

In this project feature extraction is applied to create the four significant features based on the available features of the data which helps to detect the genuine user authentication. The features extracted based on the available features of the dataset using latency parameters.

- Dwell time
- Flight time
- Digraph
- Trigraph

Dwell time

Dwell time is the length of time a key is pushed. Dwell time describes how long a key is held down before being released. It is the interval between when you press and when you release a key. That can be calculated as

$$DT = UD + DD$$

Whereas,

DT-Dwell time

UD- The duration between releasing '.' key and pressing 't' key.

DD- The duration between pressing '.' key and pressing 't' key.

Flight time

The time interval between releasing a key and pressing the next one is known as the flight time. Flight time is the amount of time between pressing a key and releasing it. That can be calculated as

$$FT = H.i + H.e$$

Whereas,

FT-Flight time

H.i- The duration between pressing and releasing ‘i’ key.

H.e- The duration between pressing and releasing ‘e’ key.

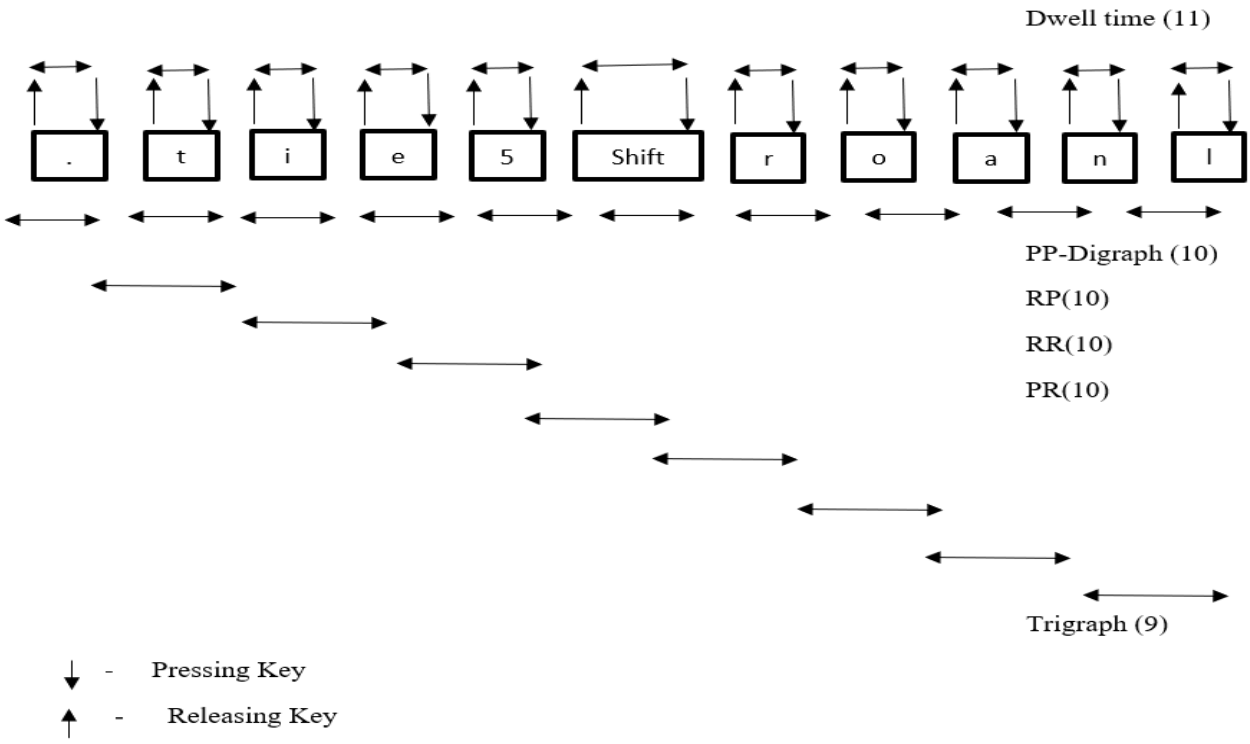


Figure 4.2 FEATURE EXTRACTION IN KEYSTROKE DYNAMICS

Digraph

Digraph is the length of -2 Sequence characters. Digraphs are no longer discriminative when computed without taking into account the word that was inputted. That can be calculate as

$$Digraph = H.i – H.e$$

Whereas,

H.i- The duration between pressing and releasing ‘i’ key.

H.e- The duration between pressing and releasing ‘e’ key.

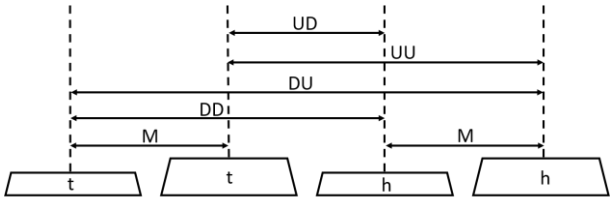


Figure 4.3.1.2 Digraph

Trigraph

Trigraph is the length of -3 Sequence characters. Trigraph is the interval of time between pressing the first key and pressing the third key.

That can be calculated as

$$\text{Trigraph} = H.i + H.e - H.five$$

Whereas,

H.i - The duration between pressing and releasing ‘i’ key.

H.e - The duration between pressing and releasing ‘e’ key.

H.five - The duration between pressing and releasing ‘.’ key.

4.4 Feature selection

The time interval between keystrokes might occur at any interval and is not recorded at a continuous rate, it is non-stationary in its raw form. Working with nonstationary time series data may be quite difficult, and differencing is a popular method for obtaining stationary data.

The standard metrics that are used in keystroke dynamics research dwell time, flight time, Trigraph and Diagraph.

Feature Selection has done using K-means Clustering and Heatmap Correlation in this project.

4.4.1 Types of Feature Selection Methods in Machine Learning

1. Filter Methods

Filter methods pick up the intrinsic properties of the features measured via univariate statistics instead of cross-validation performance. These methods are faster and less computationally expensive than wrapper methods. When dealing with high-dimensional data, it is computationally cheaper to use filter methods.

- Information Gain
- Chi-square Test
- Fisher’s Score

- Correlation Coefficient
- Missing value

2. Wrapper Methods

Wrappers require some method to search the space of all possible subsets of features, assessing their quality by learning and evaluating a classifier with that feature subset. The feature selection process is based on a specific machine learning algorithm we are trying to fit on a given dataset. It follows a greedy search approach by evaluating all the possible combinations of features against the evaluation criterion. The wrapper methods usually result in better predictive accuracy than filter methods.

- Forward Feature Selection
- Backward Feature Elimination
- Exhaustive Feature Selection
- Recursive Feature Elimination

3. Embedded Methods

These methods encompass the benefits of both the wrapper and filter methods by including interactions of features but also maintaining reasonable computational costs. Embedded methods are iterative in the sense that takes care of each iteration of the model training process and carefully extract those features which contribute the most to the training for a particular iteration.

- LASSO Regularization (L1)
- Random Forest Importance

In this project the following feature selection methods are used to select the significant features of K means Clustering and Heatmap Correlation.

4.4.2 K-means Clustering

The process of organizing objects (data) into groups based on similar features within the members (data points) of the group. Heatmaps are one way to visualize the results of clustering.

After locating clusters in the training vectors using the k-means clustering technique, it determines whether the test vector is near any of the clusters. The detector simply applies the k-means algorithm to the training data (with k = 3) during the training phase.

Each training vector should be near to at least one of the three centroids produced by the algorithm, which creates three of them. The Euclidean distance between the test vector and the closest of these centroids is used to determine the anomaly score during the test phase.

Benefits for performing K-means clustering

- Organize similar items close to each other
- Get experiment-wide look at an interesting subset of data
- Visually identify patterns for further analysis
- Order features and/or samples in a sensible way
- Split features and/or samples into a predefined number of groups
- As one method of quality control
- Explore a large data matrix (such as of expression measurements)

In order to assign similar components to the same cluster, clustering algorithms divide a collection of elements into subsets or clusters. The most typical unit of measurement needed for this is the Euclidean distance, which must be defined. Unsupervised learning is a technique used in clustering, so no human expert is engaged in the procedure.

As opposed to hierarchical clustering, partitional or flat clustering methods do not specify any structure or relationship between clusters. K-means, the most well-liked flat clustering technique, places observations in the cluster with the closest mean.

By the total number of user appearances, digraphs are arranged in decreasing order. As a result, the values of the feature vector's first dimension, which corresponds to the most frequent digraphs, represent the mean of a larger population of observations and, logically, ought to be more important during the clustering process than the last dimension, which corresponds to infrequently typed digraphs.

This raises the issue of how many and which specific digraphs are required to correctly identify the user of a subsession. By changing the feature vector size, the most practical number of digraphs will be determined based on the effectiveness of the clustering.

4.4.3 Heatmap Correlation

A heatmap that displays a 2D correlation matrix between two discrete dimensions and uses coloured cells to represent data from typically a monochromatic scale is called a correlation heatmap. The first dimension's values are displayed as the table's rows, while the second dimension's values are displayed as columns. The percentage of measurements that match the dimensional value is shown in the cell's colour.

Because they show differences and variance in the same data and make patterns easy to comprehend, correlation heatmaps are perfect for data analysis. A colorbar helps a correlation heatmap, like a conventional heatmap, by making the data more legible and understandable.

Data visualisation is accomplished with the help of the Python module Seaborn, which is based on matplotlib. It offers a way to exhibit data in the form of a statistical graph as an interesting and appealing way to communicate certain information.

One of the components offered by Seaborn is a heatmap, which uses a colour scheme to depict variation in linked data. This article mostly focuses on correlation heatmaps and how to create one for a data frame using seaborn, pandas, and matplotlib.

A correlation heatmap, which shows the correlation matrix, visualises the strength and direction of the correlation between two sets of variables in a dataset. It is a useful method for uncovering relationships and patterns in huge datasets.

Seaborn, a Python data visualisation library, provides straightforward tools for creating statistical graphics. Thanks to its feature for producing correlation heatmaps, users can immediately visualise a dataset's correlation matrix.

To create a correlation heatmap, we must first import the dataset, compute the variables' correlation matrix, and then use the Seaborn heatmap function to create the heatmap. A matrix with colours designating the strength of the correlation between the variables is shown in the heatmap. The correlation coefficients can also be displayed by the user on the heatmap.

Seaborn correlation heatmaps are a useful visualisation approach for analysing links and trends in datasets and can be used to identify important factors for further research.

Training and testing can be done in the feature reduction which can be done using SVM algorithm and Manhattan distance filter. While, testing and training random users will be selected in SVM ,the same user will be carried out to the Manhattan distance filter to compare the Baseline NRI Score.

4.5 User Authentication

User authentication is a method that keeps unauthorized users from accessing sensitive information. Authentication is the process of verifying the identity of a user or system before granting access to resources or information. It is the process of confirming that a person, device, or application is who or what it claims to be, and not an imposter or malicious entity.

In computer security, authentication is typically used to ensure that a user is who they claim to be before allowing access to a system or network. Authentication mechanisms may include passwords, biometric data, smart cards, security tokens, or other methods that prove the identity of the user or device.

Authentication is an important aspect of cybersecurity because it helps to prevent unauthorized access to sensitive information or resources. Without proper authentication, malicious actors may be able to gain access to systems and networks, potentially causing harm or stealing confidential data.

Authentication is often used in conjunction with other security measures, such as encryption and access control, to provide a comprehensive security solution. By properly authenticating users and devices, organizations can ensure that only authorized entities are allowed to access their systems and data.

Keystroke authentication is a biometric authentication technique that uses a person's typing patterns or keystroke dynamics to verify their identity. It involves capturing and analyzing the rhythm, timing, and pressure of a user's keystrokes as they type on a keyboard, and comparing these patterns to a pre-registered profile to determine if the user is authentic.

4.5.1 Methods of User Authentication

1. Password-based authentication

Passwords are the most common methods of authentication. Passwords can be in the form of a string of letters, numbers, or special characters. To protect yourself you need to create strong passwords that include a combination of all possible options.

However, passwords are prone to phishing attacks and bad hygiene that weakens effectiveness. An average person has about 25 different online accounts, but only 54% of users use different passwords across their accounts.

2. Multi-factor authentication

Multi-Factor Authentication (MFA) is an authentication method that requires two or more independent ways to identify a user. Examples include codes generated from the user’s smartphone, Captcha tests, fingerprints, voice biometrics or facial recognition.

3. Certificate-based authentication

Certificate-based authentication technologies identify users, machines or devices by using digital certificates. A digital certificate is an electronic document based on the idea of a driver’s license or a passport.

The certificate contains the digital identity of a user including a public key, and the digital signature of a certification authority. Digital certificates prove the ownership of a public key and issued only by a certification authority.

4. Biometric authentication

Biometrics authentication is a security process that relies on the unique biological characteristics of an individual. Here are key advantages of using biometric authentication technologies:

- Biological characteristics can be easily compared to authorized features saved in a database.
- Biometric authentication can control physical access when installed on gates and doors.
- You can add biometrics into your multi-factor authentication process.

Biometric authentication technologies are used by consumers, governments and private corporations including airports, military bases, and national borders. The technology is increasingly adopted due to the ability to achieve a high level of security without creating friction for the user.

5. Token-based authentication

Token-based authentication technologies enable users to enter their credentials once and receive a unique encrypted string of random characters in exchange. You can then use the token to access protected systems instead of entering your credentials all over again. The digital token proves that you already have access permission. Use cases of token-based authentication include RESTful APIs that are used by multiple frameworks and clients.

Keystroke authentication typically involves the following steps:

Enrollment: The user types a series of predefined phrases or sentences into a computer or device. These keystrokes are recorded and used to create a profile of the user's typing patterns.

Authentication: The user types the same predefined phrases or sentences, and the system compares their keystroke patterns to the previously recorded profile to verify their identity.

Keystroke authentication can provide a relatively simple and low-cost way to authenticate users, without requiring specialized hardware or software. However, it has some limitations, such as the potential for false positives or false negatives due to changes in the user's typing patterns over time, or variations in the typing environment (e.g. typing speed, typing angle, or keyboard layout).

Despite these limitations, keystroke authentication can be a useful additional layer of security when combined with other authentication methods, such as passwords or biometric authentication, to provide a more robust authentication system.

4.5.2 Model Building

The science and practice of machine learning may allow programmed computers to learn from the data provided to them. Computers are frequently trained on the data (training set) provided to them throughout the machine learning process, and they can demonstrate their performance on a specific data set. (Test set). In this approach, the problem is resolved with the least amount of human involvement. Where traditional methods are ineffective, machine learning is widely used.

Following is a list of possible uses:

- It has the ability to tackle complex problems, which traditional approaches are unable to do.
- It is capable of interpreting enormous amounts of complex data
- It is capable of resolving complex issues for which standard approaches are ineffective.
- Machine learning techniques can offer better answers when the state-of-the-art ones call for excessively frequent external updates or external intervention, respectively.
- Different surroundings can support it. Data analysis is frequently used to apply machine learning techniques to a new circumstance.

Supervised Learning

The training data had been appropriately identified and labelled using this way. For instance, the dataset contains information (tags/labels) regarding the type of each flow. (such as normal or harmful). These tags are compared to the findings discovered by the algorithm in the following stage, the test/prediction phase, and the algorithm's success is also determined. The strategy has a good track record of success. Although expensive, supervised learning relies on outside services (like manual tagging) for labeling, and this process is repeated until the algorithm performs at a high degree of accuracy.

Support Vector Machines (SVM) Algorithm

Support Vector Machines (SVM) can be used for keystroke dynamics, which is the process of identifying individuals based on their typing patterns. SVM is a popular machine learning algorithm that can be used for classification tasks, including keystroke dynamics.

The first step in using SVM for keystroke dynamics is to collect data from individuals. This data should include the typing patterns of individuals, such as the duration of key presses, the time between key presses, and the number of errors made during typing.

Once the data is collected, it needs to be preprocessed to extract relevant features. Feature extraction is an important step in machine learning because it helps to reduce the dimensionality of the data and identify the most relevant features for the classification task. For keystroke dynamics, features can include the duration of key presses, the time between key presses, and the number of errors made during typing.

Once the features are extracted, the SVM model can be trained using a labeled dataset. The labeled dataset should include examples of keystroke patterns from different individuals and their corresponding labels. The labels indicate which individual the keystroke pattern belongs to.

After the SVM model is trained, it can be used to classify new keystroke patterns. When a new keystroke pattern is presented to the model, it will use the features extracted from the pattern to predict which individual it belongs to.

Overall, SVM is a powerful machine learning algorithm that can be used for keystroke dynamics. It is important to collect a large and diverse dataset for training the model and to carefully select the relevant features for the classification task.

Advantages of SVM

- Effective in cases with big dimensions.
- Due to the decision function's use of support vectors, a subset of training points, its memory usage is effective.
- For the decision functions, various kernel functions can be supplied, as well as bespoke kernels.

Limitations of SVM

- Unsuitable to Large Datasets.
- Large training time.
- More features, more complexities.
- Bad performance on high noise.
- Does not determine Local optima.

Manhattan Distance Filter

Manhattan Distance is extremely similar to Euclidean distance. We sum the absolute value of the difference between each dimension rather than the squared difference between them. Manhattan distance is so named because it's comparable to how you may walk through a city block.

The Manhattan distance filter, also known as the city block distance filter or L1 norm filter, is a technique used in image processing and computer vision to filter out noise and preserve edges in an image.

The filter er works by computing the Manhattan distance between each pixel and its surrounding neighbors. The Manhattan distance between two points is the sum of the absolute differences between their x and y coordinates. For example, the Manhattan distance between the points (1, 3) and (5, 7) is |1 - 5| + |3 - 7| = 8.

To apply the filter to an image, a sliding window is moved over each pixel, and the Manhattan distance between the pixel and its neighbors within the window is computed. If the distance is below a certain threshold, the pixel is considered part of an edge and is preserved. If the distance is above the threshold, the pixel is considered to be noise and is filtered out.

The Manhattan distance filter is particularly effective at preserving edges in images because it does not blur or smooth them out, as some other filters can do. Instead, it only removes noise that is not part of an edge, while leaving the edges intact. This can result in sharper and more detailed images.

Overall, the Manhattan distance filter is a useful tool for improving the quality of images in computer vision and image processing applications, particularly those that require edge preservation.

Advantages of Manhattan distance filter

1. Simplicity: The Manhattan distance filter is straightforward to implement and understand. It calculates the distance between two points by summing the absolute differences between their coordinates. This simplicity makes it easy to incorporate into algorithms and systems.
2. Computational efficiency: The Manhattan distance filter is computationally efficient to calculate compared to other distance metrics, such as the Euclidean distance. Since it involves only the sum of absolute differences, it avoids the need for expensive square root operations, which can be time-consuming, especially in large-scale applications.
3. Feature relevance: The Manhattan distance filter gives equal importance to all features or dimensions of the data. It is suitable for scenarios where each feature contributes independently to the similarity or dissimilarity between two points. This can be advantageous when dealing with high-dimensional data, as it prevents any single feature from dominating the distance calculation.
4. Robustness to outliers: The Manhattan distance filter is less sensitive to outliers compared to the Euclidean distance filter. Outliers have a larger impact on the Euclidean distance due to the squared term in the formula, whereas the Manhattan distance considers only the absolute differences. As a result, the Manhattan distance filter can provide more robust results when dealing with noisy or outlier-prone data.

5. Spatial proximity: The Manhattan distance filter is particularly useful in scenarios where spatial proximity matters. It accurately captures the "city block" or "taxicab" distance between two points in a grid-like or city-like environment. This property makes it suitable for applications such as route planning, urban planning, and image processing tasks involving grid-like structures.
6. Interpretability: The Manhattan distance filter produces interpretable results because the distance value corresponds directly to the sum of absolute differences between coordinates. This interpretability makes it easier to understand and explain the reasoning behind the filtering or decision-making process.

Limitations Of Manhattan distance filter

1. Directional insensitivity
2. Sensitivity to scale
3. Limited representation of continuous relationships
4. Inability to capture non-linear relationships
5. Sensitivity to irrelevant features

4.6 Performance Metrics

Performance metrics, also known as evaluation metrics or assessment measures, are quantitative measures used to assess the performance or effectiveness of a system, model, algorithm, or process. These metrics provide objective criteria for evaluating the quality, accuracy, efficiency, or other desirable attributes of a particular solution or approach. Performance metrics are commonly used in various fields, including machine learning, data analysis, optimization, and system evaluation.

Performance metrics can vary depending on the specific problem and context. Here are some commonly used performance metrics in different domains:

1. Classification Metrics:

a) Accuracy: The proportion of correctly classified instances.
b) Precision: The proportion of true positive predictions among positive predictions.
c) Recall (Sensitivity or True Positive Rate): The proportion of true positive predictions among actual positive instances.
d) F1 Score: The harmonic mean of precision and recall, provides a balanced measure.
e) Specificity (True Negative Rate): The proportion of true negative predictions among actual negative instances.

f) ROC Curve (Receiver Operating Characteristic Curve): A graphical representation of the trade-off between true positive rate and false positive rate for different classification thresholds.
g) Area Under the ROC Curve (AUC-ROC): A metric that summarizes the ROC curve, provides a measure of the overall performance of a classifier.

2. Regression Metrics:

a) Mean Absolute Error (MAE): The average absolute difference between predicted and actual values.
b) Mean Squared Error (MSE): The average squared difference between predicted and actual values.
c) Root Mean Squared Error (RMSE): The square root of MSE, provides a more interpretable measure in the same unit as the target variable.
d) R-squared (coefficient of determination): The proportion of the variance in the target variable explained by the model.
e) Mean Absolute Percentage Error (MAPE): The average percentage difference between predicted and actual values, commonly used for relative error assessment.

3. Clustering Metrics:

a) Silhouette Coefficient: Measures the compactness of clusters and separation between clusters.
b) Calinski-Harabasz Index: Measures the ratio of between-cluster dispersion to within-cluster dispersion.
c) Davies-Bouldin Index: Measures the average similarity between each cluster and its most similar cluster.

4. Information Retrieval Metrics:

a) Precision at K: The proportion of relevant items among the top K retrieved items.
b) Recall at K: The proportion of relevant items retrieved among all relevant items.

4.6.1 Equal Error Rate

Equal Error Rate (EER) is a performance metric commonly used in biometric systems and binary classification tasks. It provides a balance between false acceptance rate (FAR) and false rejection rate (FRR) by finding the point at which these rates are equal.

In biometric systems, such as face recognition or fingerprint identification, the goal is to correctly classify individuals as either genuine (positive) or impostor (negative). The EER represents the point at which the system makes an equal number of false acceptances and false rejections. In other

words, the EER is the threshold or operating point where the rates of false acceptance and false rejection intersect.

To calculate the EER, a receiver operating characteristic (ROC) curve is constructed by varying the decision threshold of the classifier, which determines the classification boundary between positive and negative classes. The ROC curve plots the true positive rate (TPR) against the false positive rate (FPR) at different threshold settings.

The EER is obtained by finding the threshold at which the FPR and FRR are closest to each other. This means that the EER represents the point where the probability of incorrectly accepting a negative sample is approximately equal to the probability of incorrectly rejecting a positive sample.

The EER is often used as a single summary metric to compare different biometric systems or classification algorithms. It provides a concise measure of performance, but it is important to note that it may not capture the entire performance profile of the system. Additional metrics, such as the area under the ROC curve (AUC-ROC) or precision-recall curves, can provide more detailed insights into the performance characteristics of the system.

In summary, the Equal Error Rate (EER) is a threshold-based performance metric that represents the point at which false acceptance rate (FAR) and false rejection rate (FRR) are equal in biometric systems and binary classification tasks.

The Equal Error Rate (EER) is calculated by finding the threshold at which the False Acceptance Rate (FAR) and the False Rejection Rate (FRR) are equal.

$$EER=FAR+FRR2$$

$$FAR = FP / (FP + TN)$$

$$FRR = FN / (FN + TP)$$

where:

FP: False Positives (incorrectly classified negative samples as positive)

TN: True Negatives (correctly classified negative samples as negative)

FN: False Negatives (incorrectly classified positive samples as negative)

TP: True Positives (correctly classified positive samples as positive)

To calculate the EER, you need to vary the decision threshold of the classifier and calculate the FAR and FRR at each threshold. The EER is then obtained by finding the threshold where the FAR and FRR are approximately equal.

CHAPTER-V

5.RESULTS AND DISCUSSION

The result of keystroke dynamic authentication is typically a binary output indicating whether the system has determined that the person typing is the same as the enrolled user. This result can be used to grant access to secure systems, applications, or physical spaces. Average EER for SVM detector and Manhattan Distance Filter Algorithm for Baseline NMI score.

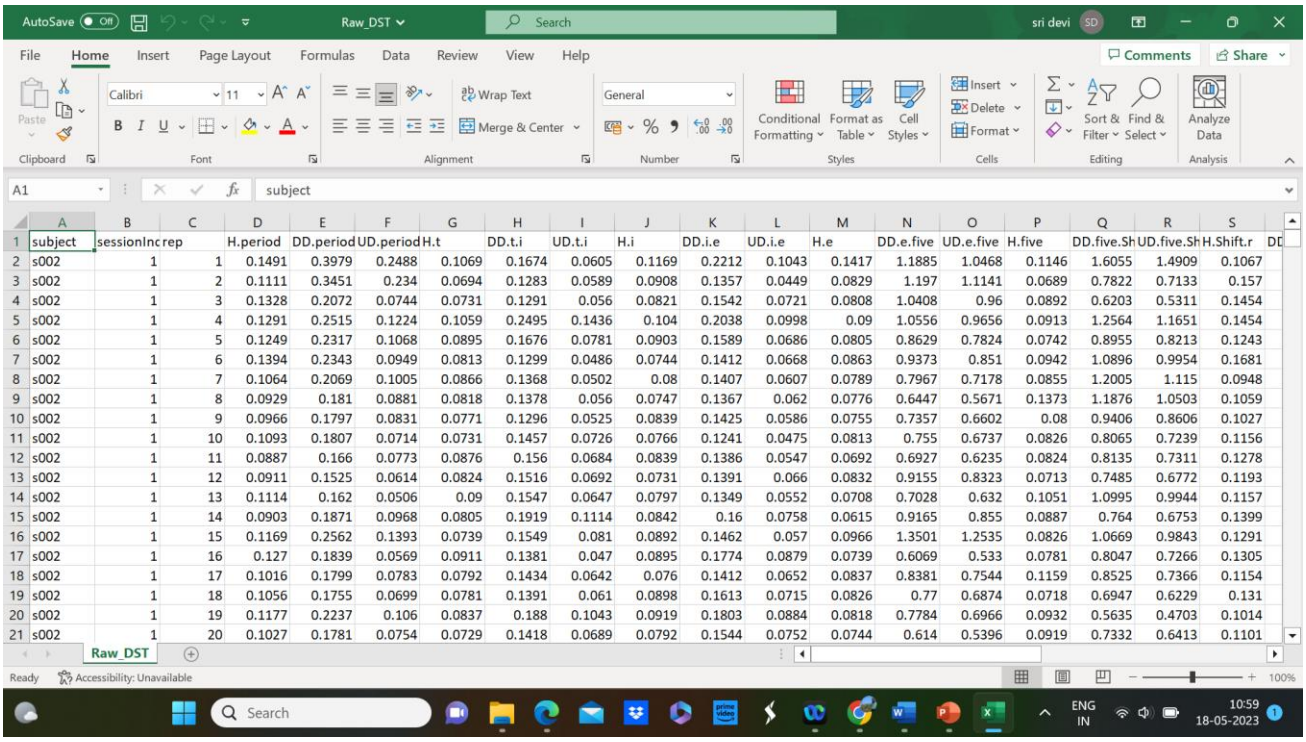
This section briefly shows the results obtained under each phase of the user authentication with keystroke dynamic

PHASE - I

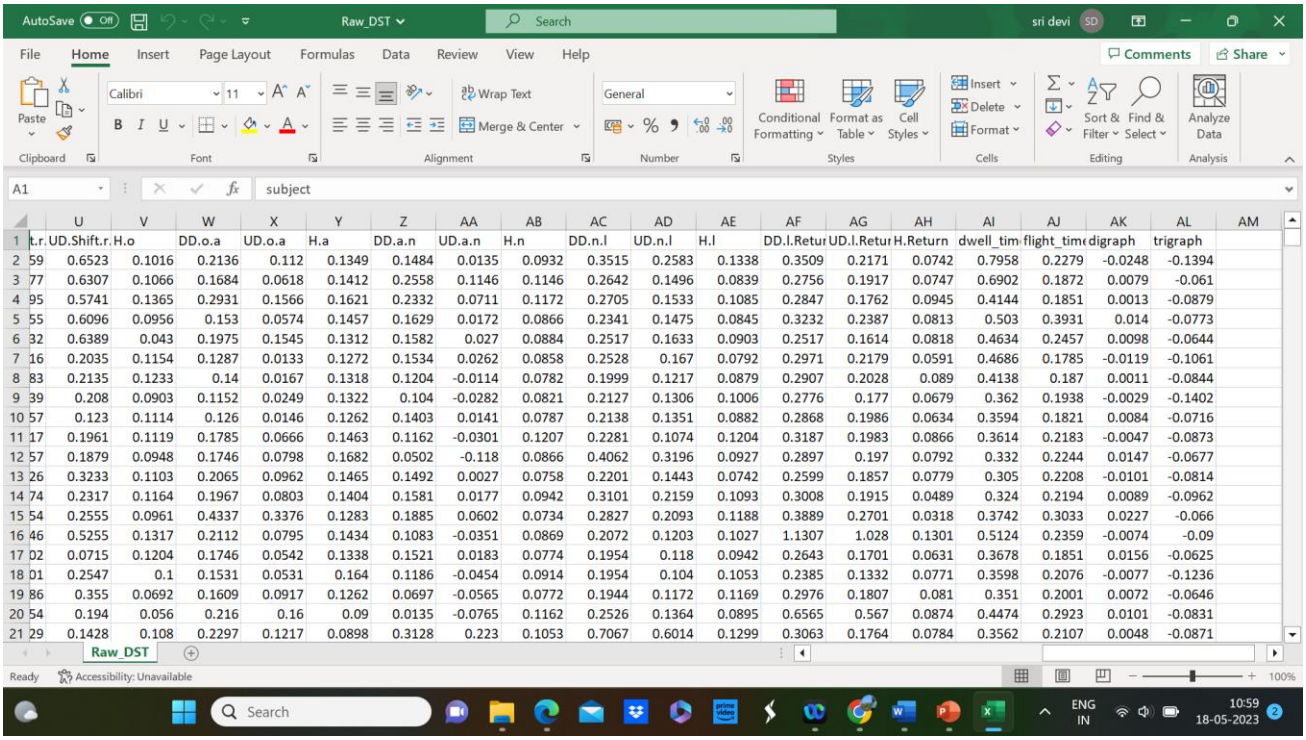
5.1 Data Collection

Benchmark dataset is collected.

Adding dataset screenshot



subject	sessionIn	crep	H.period	DD.period	UD.period	H.t	DD.t.i	UD.t.i	H.i	DD.i.e	UD.i.e	H.e	DD.e.five	UD.e.five	H.five	DD.five.Sh	UD.five.Sh	H.Shift.r
s002	1	1	0.1491	0.3979	0.2488	0.1069	0.1674	0.0605	0.1169	0.2212	0.1043	0.1417	1.1885	1.0468	0.1146	1.6055	1.4909	0.1067
s002	1	2	0.1111	0.3451	0.234	0.0694	0.1283	0.0589	0.0908	0.1357	0.0449	0.0829	1.197	1.1141	0.0689	0.7822	0.7133	0.157
s002	1	3	0.1328	0.2072	0.0744	0.0731	0.1291	0.056	0.0821	0.1542	0.0721	0.0808	1.0408	0.96	0.0892	0.6203	0.5311	0.1454
s002	1	4	0.1291	0.2515	0.1224	0.1059	0.2495	0.1436	0.104	0.2038	0.0998	0.09	1.0556	0.9656	0.0913	1.2564	1.1651	0.1454
s002	1	5	0.1249	0.2317	0.1068	0.0895	0.1676	0.0781	0.0903	0.1589	0.0686	0.0805	0.8629	0.7824	0.0742	0.8955	0.8213	0.1243
s002	1	6	0.1394	0.2343	0.0949	0.0813	0.1299	0.0486	0.0744	0.1412	0.0668	0.0863	0.9373	0.851	0.0942	1.0896	0.9954	0.1681
s002	1	7	0.1064	0.2069	0.1005	0.0866	0.1368	0.0502	0.08	0.1407	0.0607	0.0789	0.7967	0.7178	0.0855	1.2005	1.115	0.0948
s002	1	8	0.0929	0.181	0.0881	0.0818	0.1378	0.056	0.0747	0.1367	0.062	0.0776	0.6447	0.5671	0.1373	1.1876	1.0503	0.1059
s002	1	9	0.0966	0.1797	0.0831	0.0771	0.1296	0.0525	0.0839	0.1425	0.0586	0.0755	0.7357	0.6602	0.08	0.9406	0.8606	0.1027
s002	1	10	0.1093	0.1807	0.0714	0.0731	0.1457	0.0726	0.0766	0.1241	0.0475	0.0813	0.755	0.6737	0.0826	0.8065	0.7239	0.1156
s002	1	11	0.0887	0.166	0.0773	0.0876	0.156	0.0684	0.0839	0.1386	0.0547	0.0692	0.6927	0.6235	0.0824	0.8135	0.7311	0.1278
s002	1	12	0.0911	0.1525	0.0614	0.0824	0.1516	0.0692	0.0731	0.1391	0.066	0.0832	0.9155	0.8323	0.0713	0.7485	0.6772	0.1193
s002	1	13	0.1114	0.162	0.0506	0.09	0.1547	0.0647	0.0797	0.1349	0.0552	0.0708	0.7028	0.632	0.1051	1.0995	0.9944	0.1157
s002	1	14	0.0903	0.1871	0.0968	0.0805	0.1919	0.1114	0.0842	0.16	0.0758	0.0615	0.9165	0.855	0.0887	0.764	0.6753	0.1399
s002	1	15	0.1169	0.2562	0.1393	0.0739	0.1549	0.081	0.0892	0.1462	0.057	0.0966	1.3501	1.2535	0.0826	1.0669	0.9843	0.1291
s002	1	16	0.127	0.1839	0.0569	0.0911	0.1381	0.047	0.0895	0.1774	0.0879	0.0739	0.6069	0.533	0.0781	0.8047	0.7266	0.1305
s002	1	17	0.1016	0.1799	0.0783	0.0792	0.1434	0.0642	0.076	0.1412	0.0652	0.0837	0.8381	0.7544	0.1159	0.8525	0.7366	0.1154
s002	1	18	0.1056	0.1755	0.0699	0.0781	0.1391	0.061	0.0898	0.1613	0.0715	0.0826	0.77	0.6874	0.0718	0.6947	0.6229	0.131
s002	1	19	0.1177	0.2237	0.106	0.0837	0.188	0.1043	0.0919	0.1803	0.0884	0.0818	0.7784	0.6966	0.0932	0.5635	0.4703	0.1014
s002	1	20	0.1027	0.1781	0.0754	0.0729	0.1418	0.0689	0.0792	0.1544	0.0752	0.0744	0.614	0.5396	0.0919	0.7332	0.6413	0.1101



	U	V	W	X	Y	Z	AA	AB	AC	AD	AE	AF	AG	AH	AI	AJ	AK	AL	AM
1	t.r	UD.Shift.r	H.o	DD.o.a	UD.o.a	H.a	DD.a.n	UD.a.n	H.n	DD.n.l	UD.n.l	H.l	DD.l.Retur	UD.l.Retur	H.Return	dwell_tim	flight_time	digraph	trigraph	
2	59	0.6523	0.1016	0.2136	0.112	0.1349	0.1484	0.0135	0.0932	0.3515	0.2583	0.1338	0.3509	0.2171	0.0742	0.7958	0.2279	-0.0248	-0.1394	
3	77	0.6307	0.1066	0.1684	0.0618	0.1412	0.2558	0.1146	0.1146	0.2642	0.1496	0.0839	0.2756	0.1917	0.0747	0.6902	0.1872	0.0079	-0.061	
4	95	0.5741	0.1365	0.2931	0.1566	0.1621	0.2332	0.0711	0.1172	0.2705	0.1533	0.1085	0.2847	0.1762	0.0945	0.4144	0.1851	0.0013	-0.0879	
5	55	0.6096	0.0956	0.153	0.0574	0.1457	0.1629	0.0172	0.0866	0.2341	0.1475	0.0845	0.3232	0.2387	0.0813	0.503	0.3931	0.014	-0.0773	
6	32	0.6389	0.043	0.1975	0.1545	0.1312	0.1582	0.027	0.0884	0.2517	0.1633	0.0903	0.2517	0.1614	0.0818	0.4634	0.2457	0.0098	-0.0644	
7	16	0.2035	0.1154	0.1287	0.0133	0.1272	0.1534	0.0262	0.0858	0.2528	0.167	0.0792	0.2971	0.2179	0.0591	0.4686	0.1785	-0.0119	-0.1061	
8	83	0.2135	0.1233	0.14	0.0167	0.1318	0.1204	-0.0114	0.0782	0.1999	0.1217	0.0879	0.2907	0.2028	0.089	0.4138	0.187	0.0011	-0.0844	
9	39	0.208	0.0903	0.1152	0.0249	0.1322	0.104	-0.0282	0.0821	0.2127	0.1306	0.1006	0.2776	0.177	0.0679	0.362	0.1938	-0.0029	-0.1402	
10	57	0.123	0.1114	0.126	0.0146	0.1262	0.1403	0.0141	0.0787	0.2138	0.1351	0.0882	0.2868	0.1986	0.0634	0.3594	0.1821	0.0084	-0.0716	
11	17	0.1961	0.1119	0.1785	0.0666	0.1463	0.1162	-0.0301	0.1207	0.2281	0.1074	0.1204	0.3187	0.1983	0.0866	0.3614	0.2183	-0.0047	-0.0873	
12	57	0.1879	0.0948	0.1746	0.0798	0.1682	0.0502	-0.118	0.0866	0.4062	0.3196	0.0927	0.2897	0.197	0.0792	0.332	0.2244	0.0147	-0.0677	
13	26	0.3233	0.1103	0.2065	0.0962	0.1465	0.1492	0.0027	0.0758	0.2201	0.1443	0.0742	0.2599	0.1857	0.0779	0.305	0.2208	-0.0101	-0.0814	
14	74	0.2317	0.1164	0.1967	0.0803	0.1404	0.1581	0.0177	0.0942	0.3101	0.2159	0.1093	0.3008	0.1915	0.0489	0.324	0.2194	0.0089	-0.0962	
15	54	0.2555	0.0961	0.4337	0.3376	0.1283	0.1885	0.0602	0.0734	0.2827	0.2093	0.1188	0.3889	0.2701	0.0318	0.3742	0.3033	0.0227	-0.066	
16	46	0.5255	0.1317	0.2112	0.0795	0.1434	0.1083	-0.0351	0.0869	0.2072	0.1203	0.1027	1.1307	1.028	0.1301	0.5124	0.2359	-0.0074	-0.09	
17	02	0.0715	0.1204	0.1746	0.0542	0.1338	0.1521	0.0183	0.0774	0.1954	0.118	0.0942	0.2643	0.1701	0.0631	0.3678	0.1851	0.0156	-0.0625	
18	01	0.2547	0.1	0.1531	0.0531	0.164	0.1186	-0.0454	0.0914	0.1954	0.104	0.1053	0.2385	0.1332	0.0771	0.3598	0.2076	-0.0077	-0.1236	
19	86	0.355	0.0692	0.1609	0.0917	0.1262	0.0697	-0.0565	0.0772	0.1944	0.1172	0.1169	0.2976	0.1807	0.081	0.351	0.2001	0.0072	-0.0646	
20	54	0.194	0.056	0.216	0.16	0.09	0.0135	-0.0765	0.1162	0.2526	0.1364	0.0895	0.6565	0.567	0.0874	0.4474	0.2923	0.0101	-0.0831	
21	29	0.1428	0.108	0.2297	0.1217	0.0898	0.3128	0.223	0.1053	0.7067	0.6014	0.1299	0.3063	0.1764	0.0784	0.3562	0.2107	0.0048	-0.0871	

Figure 5.1 Data Collection

5.1.1 Importing Libraries

```
import pandas as pd
from sklearn.preprocessing import LabelEncoder
from sklearn.feature_selection import SelectKBest, f_classif
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score
result_df = pd.DataFrame(columns=['User','Authenticity','Algorithm','EER Rate'])

# Load the data
keystroke_data = pd.read_csv('D:\Project\Project dataset\Raw_DST.csv')
```

PHASE-II

5.2 Data Pre-processing

In this project the data pre-processing used to check the redundancy of the dataset with null values and duplicate entries. Figure 5.2 and 5.3 shows the results of data pre-processing

Data preprocessing

```
##DATA PREPROCESSING##
##Checking null values##
keystroke_data.isnull()
```

	subject	sessionIndex	rep	H.period	DD.period.t	UD.period.t	H.t	DD.t.i	UD.t.i	H.i	...	DD.n.l	UD.n.l	H.l	DD.l.Return	UD.l.Return	H.Return
0	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False	False	False
1	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False	False	False
2	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False	False	False
3	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False	False	False
4	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False	False	False
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
20395	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False	False	False
20396	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False	False	False
20397	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False	False	False
20398	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False	False	False
20399	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False	False	False

20400 rows × 38 columns

Figure 5.2 After checking null values

```
#Duplicate Entries
duplicates = keystroke_data.duplicated()
```

```
print(keystroke_data[duplicates])
```

```
Empty DataFrame
Columns: [subject, sessionIndex, rep, H.period, DD.period.t, UD.period.t, H.t, DD.t.i, UD.t.i, H.i, DD.i.e, UD.i.e, H.e, DD.e.f
ive, UD.e.five, H.five, DD.five.Shift.r, UD.five.Shift.r, H.Shift.r, DD.Shift.r.o, UD.Shift.r.o, H.o, DD.o.a, UD.o.a, H.a, DD.
a.n, UD.a.n, H.n, DD.n.l, UD.n.l, H.l, DD.l.Return, UD.l.Return, H.Return, dwell_time, flight_time, digraph, trigraph]
Index: []

[0 rows x 38 columns]
```

Figure 5.3 After checking duplicate entries

From figure 5.2 and figure 5.3 it is observed that, the dataset does not contain any null values and duplicate entries

PHASE -III

Feature Extraction

In this project Feature extracted from H.period to trigraph Figure 5.4 and 5.5 shows the results of data pre-processing.

Feature extraction

```
df=keystroke_data.loc[:, "H.period":"trigraph"]
df
```

	H.period	DD.period.t	UD.period.t	H.t	DD.t.i	UD.t.i	H.i	DD.i.e	UD.i.e	H.e	...	DD.n.l	UD.n.l	H.l	DD.l.Return	UD.l.Return	H.Return
0	0.1491	0.3979	0.2488	0.1069	0.1674	0.0605	0.1169	0.2212	0.1043	0.1417	...	0.3515	0.2583	0.1338	0.3509	0.2171	0.0742
1	0.1111	0.3451	0.2340	0.0694	0.1283	0.0589	0.0908	0.1357	0.0449	0.0829	...	0.2642	0.1496	0.0839	0.2756	0.1917	0.0747
2	0.1328	0.2072	0.0744	0.0731	0.1291	0.0560	0.0821	0.1542	0.0721	0.0808	...	0.2705	0.1533	0.1085	0.2847	0.1762	0.0945
3	0.1291	0.2515	0.1224	0.1059	0.2495	0.1436	0.1040	0.2038	0.0998	0.0900	...	0.2341	0.1475	0.0845	0.3232	0.2387	0.0813
4	0.1249	0.2317	0.1068	0.0895	0.1676	0.0781	0.0903	0.1589	0.0686	0.0805	...	0.2517	0.1633	0.0903	0.2517	0.1614	0.0818
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
20395	0.0884	0.0685	-0.0199	0.1095	0.1290	0.0195	0.0945	0.0757	-0.0188	0.1328	...	0.1329	0.0509	0.1005	0.2054	0.1049	0.1047
20396	0.0655	0.0630	-0.0025	0.0910	0.1148	0.0238	0.0916	0.0636	-0.0280	0.1256	...	0.0868	-0.0169	0.1445	0.2206	0.0761	0.1198
20397	0.0939	0.1189	0.0250	0.1008	0.1122	0.0114	0.0721	0.0462	-0.0259	0.0903	...	0.1311	0.0622	0.1034	0.2017	0.0983	0.0905
20398	0.0923	0.1294	0.0371	0.0913	0.0990	0.0077	0.0992	0.0897	-0.0095	0.1016	...	0.0697	0.0121	0.0979	0.1917	0.0938	0.0931
20399	0.0596	0.1310	0.0714	0.0992	0.1103	0.0111	0.0998	0.0813	-0.0185	0.1493	...	0.1133	0.0343	0.0807	0.1993	0.1186	0.1018

20400 rows × 35 columns

Figure 5.4 After Feature extracting

dwell_time	flight_time	digraph	trigraph
0.7958	0.2279	-0.0248	-0.1394
0.6902	0.1872	0.0079	-0.0610
0.4144	0.1851	0.0013	-0.0879
0.5030	0.3931	0.0140	-0.0773
0.4634	0.2457	0.0098	-0.0644
...	...	...	...
0.1370	0.1485	-0.0383	-0.1383
0.1260	0.1386	-0.0340	-0.1327
0.2378	0.1236	-0.0182	-0.0915
0.2588	0.1067	-0.0024	-0.0844
0.2620	0.1214	-0.0495	-0.1632

Figure 5.5 Added latency parameters

```
X=keystroke_data[['dwell_time', 'flight_time','digraph', 'trigraph']]
```

```
print(keystroke_data[['dwell_time', 'flight_time', 'digraph', 'trigraph']])
```

```
       dwell_time  flight_time  digraph  trigraph
0          0.7958       0.2279  -0.0248   -0.1394
1          0.6902       0.1872   0.0079   -0.0610
2          0.4144       0.1851   0.0013   -0.0879
3          0.5030       0.3931   0.0140   -0.0773
4          0.4634       0.2457   0.0098   -0.0644
...           ...          ...      ...       ...
20395      0.1370       0.1485  -0.0383   -0.1383
20396      0.1260       0.1386  -0.0340   -0.1327
20397      0.2378       0.1236  -0.0182   -0.0915
20398      0.2588       0.1067  -0.0024   -0.0844
20399      0.2620       0.1214  -0.0495   -0.1632

[20400 rows x 4 columns]
```

Figure 5.5 Measures Of latency parameters

From figure 5.3 and figure 5.5 it is observed that the feature dwell time, flight time, digraph and trigraph has added on the CNS-0430474 dataset

PHASE IV

Feature Selection

Feature Selection has done with K-means Clustering and Heatmap Correlation. To significant user Authentication using SVM and MDF algorithms to authenticate the Genuine/Imposter User.

SVM for Modeling

```
from sklearn.svm import OneClassSVM
import numpy as np
np.set_printoptions(suppress = True)
import pandas
import random
import scipy.stats as stats
from EER import evaluateEER
from sklearn.cluster import KMeans, AgglomerativeClustering
from sklearn.metrics.cluster import normalized_mutual_info_score, adjusted_rand_score
```

```
class SVMDetector:
#just the training() function changes, rest all remains same.

    def __init__(self, subjects):
        self.u_scores = []
        self.i_scores = []
        self.mean_vector = []
        self.subjects = subjects


    def training(self):
        self.clf = OneClassSVM(kernel='rbf',gamma=26)
        self.clf.fit(self.train)

    def testing(self):
        self.u_scores = -self.clf.decision_function(self.test_genuine)
        self.i_scores = -self.clf.decision_function(self.test_imposter)
        self.u_scores = list(self.u_scores)
        self.i_scores = list(self.i_scores)

    def evaluate(self):
        eers = []

        print(subjects)
        genuine_user_data = keystroke_data.loc[keystroke_data.subject == subjects, \
                                        "dwell_time":"trigraph"]
        print(genuine_user_data)
        # prints data that will be plotted
        # columns shown here are selected by corr() since
        # they are ideal for the plot
        import seaborn as sb
        print(genuine_user_data.corr())
```

Figure 5.6 Correlation for SVM

```
print(subjects)
genuine_user_data = keystroke_data.loc[keystroke_data.subject == subjects, \
                                "dwell_time":"trigraph"]
print(genuine_user_data)
# prints data that will be plotted
# columns shown here are selected by corr() since
# they are ideal for the plot
import seaborn as sb
print(genuine_user_data.corr())

# plotting correlation heatmap
dataplot=sb.heatmap(genuine_user_data.corr())

# displaying he
# We are going to use the K-Means algorithm
model_kmeans = AgglomerativeClustering(n_clusters=2)
preds = model_kmeans.fit_predict(genuine_user_data)
print(preds)

# We are going to use the NMI metric to measure the quality/performance of the clustering
baseline_score = normalized_mutual_info_score(model_kmeans.labels_, preds)
print("Baseline NMI Score:", baseline_score)

print("Selected Features : ", model_kmeans.feature_names_in_)

imposter_data = keystroke_data.loc[keystroke_data.subject != subjects, :]

#feature Reduction
self.train = genuine_user_data[:200]
self.test_genuine = genuine_user_data[200:]
self.test_imposter = imposter_data.groupby("subject"). \
                        head(5).loc[:, "dwell_time":"trigraph"]

self.training()
self.testing()
eers.append(evaluateEER(self.u_scores, \
                            self.i_scores))
return np.mean(eers)
```

Figure 5.7 Training and Testing in SVM

ManhattanFilter

```
from scipy.spatial.distance import euclidean, cityblock
import numpy as np
np.set_printoptions(suppress = True)
import pandas
from EER import evaluateEER
import seaborn as sb
```

```
class ManhattanFilteredDetector:
#just the training() function changes, rest all remains same.

    def __init__(self, subjects):
        self.user_scores = []
        self.imposter_scores = []
        self.mean_vector = []
        self.subject_choosen = subject_choosen

    def training(self):
        self.mean_vector = self.train.mean().values
        self.std_vector = self.train.std().values
        dropping_indices = []
        for i in range(self.train.shape[0]):
            cur_score = euclidean(self.train.iloc[i].values,
                                  self.mean_vector)
            if (cur_score > 3*self.std_vector).all() == True:
                dropping_indices.append(i)
        self.train = self.train.drop(self.train.index[dropping_indices])
        self.mean_vector = self.train.mean().values

    def testing(self):
        for i in range(self.test_genuine.shape[0]):
            cur_score = cityblock(self.test_genuine.iloc[i].values, \
                                  self.mean_vector)
            self.user_scores.append(cur_score)
```

Figure 5.8 Correlation for Manhattan Distance Filter

The train and test method is applied to divide the dataset for training phase and testing phase.

```
print(subjects)
genuine_user_data = keystroke_data.loc[keystroke_data.subject == subject_choosen, \
                                "dwell_time":"trigraph"]
print(genuine_user_data)
# prints data that will be plotted
# columns shown here are selected by corr() since
# they are ideal for the plot
import seaborn as sb
print(genuine_user_data.corr())

# plotting correlation heatmap
dataplot=sb.heatmap(genuine_user_data.corr())

# displaying he
# We are going to use the K-Means algorithm
model_kmeans = AgglomerativeClustering(n_clusters=2)
preds = model_kmeans.fit_predict(genuine_user_data)
print(preds)

# We are going to use the NMI metric to measure the quality/performance of the clustering
baseline_score = normalized_mutual_info_score(model_kmeans.labels_, preds)
print("Baseline NMI Score:", baseline_score)

print("Selected Features : ", model_kmeans.feature_names_in_)
imposter_data = data.loc[data.subject != subject_choosen, :]
self.train = genuine_user_data[:200]
self.test_genuine = genuine_user_data[200:]
self.test_imposter = imposter_data.groupby("subject"). \
                        head(5).loc[:, "dwell_time":"trigraph"]
self.training()
self.testing()
eers.append(evaluateEER(self.user_scores, \
                            self.imposter_scores))
return np.mean(eers)
```

Figure 5.9 Training and Testing in Manhattan Distance Filter

From figure 5.6 to figure 5.9 it is observed that k-means clustering and correlation heatmap of the training and testing and has done on the CNS-0430474 dataset form dwell time to trigraph.

Selecting Random User for SVM

```
subjects = keystroke_data["subject"].unique()
print(subjects)
subjects=random.choice(subjects)
subject_choosen=subjects
subject_choosen
```

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

's051'
```

Figure 5.10 Selecting Random User for SVM

Selecting Random User for Manhattan Distance Filter

```
subjects = keystroke_data["subject"].unique()
print(subjects)
subjects=random.choice(subjects)
subject_choosen=subjects
subject_choosen
```

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

's051'
```

Figure 5.11 Selecting Random User for Manhattan Distance Filter

The figure 5.10 and figure 5.11 selecting random user for SVM and Manhattan Distance Filter.

PHASE V

User Authentication

Equal Error Rate performance used to validate the performance of the SVM and MDF model.

```
print ("average EER for SVM detector:")
print(SVMDetector(subjects).evaluate())
svm_eer=SVMDetector(subjects).evaluate()
```

```
average EER for SVM detector:
s051
       dwell_time  flight_time  digraph  trigraph
17600      0.6096       0.1770  -0.0068   -0.0875
17601      0.3160       0.1831   0.0129   -0.0546
17602      1.0472       0.2005  -0.0013   -0.0794
17603      0.5426       0.1642  -0.0127   -0.0853
17604      0.4758       0.1506   0.0111   -0.0678
...           ...          ...      ...       ...
17995      0.2552       0.1195  -0.0634   -0.1421
17996      0.2790       0.0930  -0.0567   -0.1314
17997      0.2610       0.1075  -0.0250   -0.1052
17998      0.3480       0.0840  -0.0623   -0.1222
17999      0.4208       0.2601  -0.0092   -0.0831

[400 rows x 4 columns]
             dwell_time  flight_time   digraph  trigraph
dwell_time     1.000000     0.177942  0.188303  0.124141
flight_time    0.177942     1.000000  0.038712  0.030598
digraph        0.188303     0.038712  1.000000  0.852699
trigraph       0.124141     0.030598  0.852699  1.000000
[0 1 0 0 0 1 0 1 0 1 1 0 0 0 1 0 0 1 1 1 1 0 0 0 0 1 1 1 1 1 0 1 1 1 0 1 1
 0 1 0 1 1 1 1 1 1 0 0 0 0 1 0 1 1 0 1 0 1 1 1 1 1 1 1 1 0 0 0 0 0 1 1 1 1
 1 0 0 0 1 1 1 1 1 0 1 1 1 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 1 1 1 1 1 0 1 0 0 1
 1 1 1 0 0 0 1 1 1 1 1 1 1 0 1 0 1 0 1 1 1 0 0 1 0 1 0 1 1 1 1 1 1 1 1 0 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1
 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 0 0 1 1 1 0 1 1 1 1 1 1 1 0 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 0 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 0 1 0 1 1 1 0 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 0]
Baseline NMI Score: 1.0
```

Figure 5.12 Performance analysis of EER for SVM

```
print ("average EER for Manhattan Filtered detector:")
print(ManhattanFilteredDetector(subject_choosen).evaluate())
eer_manhattan=ManhattanFilteredDetector(subject_choosen).evaluate()
```

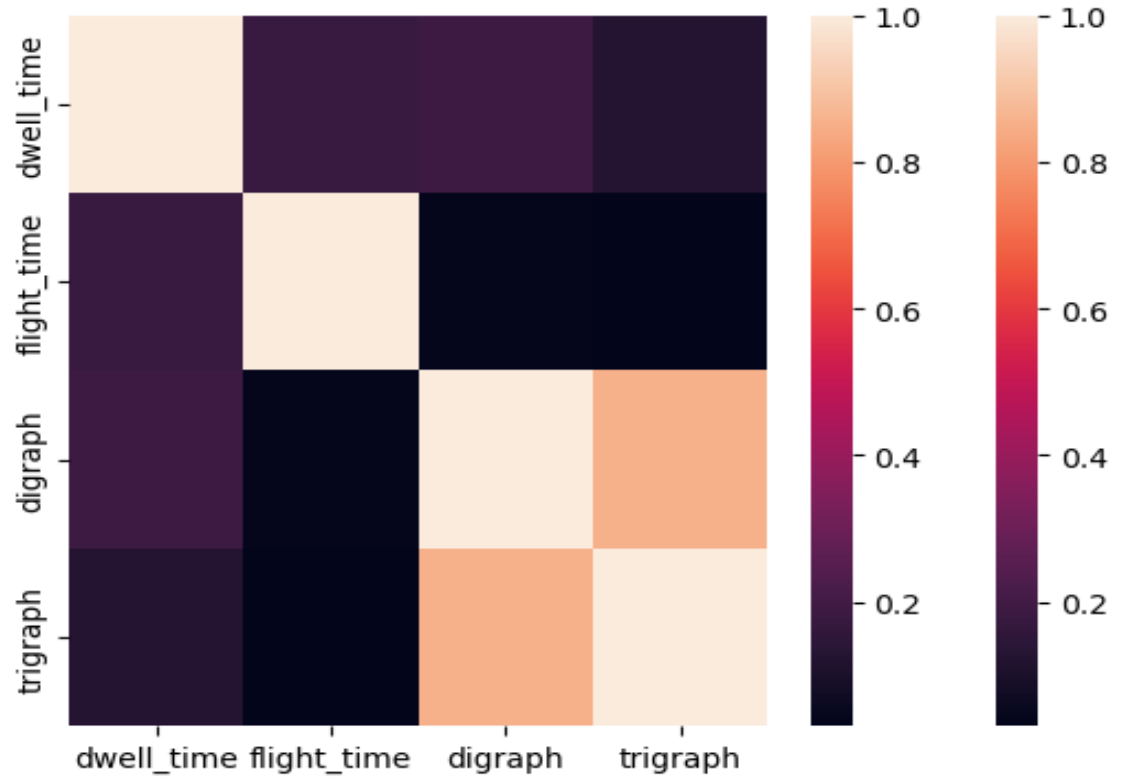
```
average EER for Manhattan Filtered detector:
s051
       dwell_time  flight_time  digraph  trigraph
17600      0.6096       0.1770  -0.0068   -0.0875
17601      0.3160       0.1831   0.0129   -0.0546
17602      1.0472       0.2005  -0.0013   -0.0794
17603      0.5426       0.1642  -0.0127   -0.0853
17604      0.4758       0.1506   0.0111   -0.0678
...           ...          ...      ...       ...
17995      0.2552       0.1195  -0.0634   -0.1421
17996      0.2790       0.0930  -0.0567   -0.1314
17997      0.2610       0.1075  -0.0250   -0.1052
17998      0.3480       0.0840  -0.0623   -0.1222
17999      0.4208       0.2601  -0.0092   -0.0831

[400 rows x 4 columns]
             dwell_time  flight_time   digraph  trigraph
dwell_time     1.000000     0.177942  0.188303  0.124141
flight_time    0.177942     1.000000  0.038712  0.030598
digraph        0.188303     0.038712  1.000000  0.852699
trigraph       0.124141     0.030598  0.852699  1.000000
[0 1 0 0 0 1 0 1 0 1 1 0 0 0 1 0 0 1 1 1 1 0 0 0 0 1 1 1 1 1 0 1 1 1 0 1 1
 0 1 0 1 1 1 1 1 1 0 0 0 0 1 0 1 1 0 1 0 1 1 1 1 1 1 1 1 0 0 0 0 0 1 1 1 1
 1 0 0 0 1 1 1 1 1 0 1 1 1 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 1 1 1 1 0 1 0 0 1
 1 1 1 0 0 0 1 1 1 1 1 1 1 0 1 0 1 0 1 1 1 0 0 1 0 1 0 1 1 1 1 1 1 1 0 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1
 1 1 1 1 1 0 1 1 1 1 1 1 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 0 0 1 1 1 0 1 1 1 1 1 1 1 0 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 0 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 0 1 0 1 1 1 0 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 0]
Baseline NMI Score: 1.0
```

Figure 5.12 Performance analysis of EER for MDF

PHASE VI

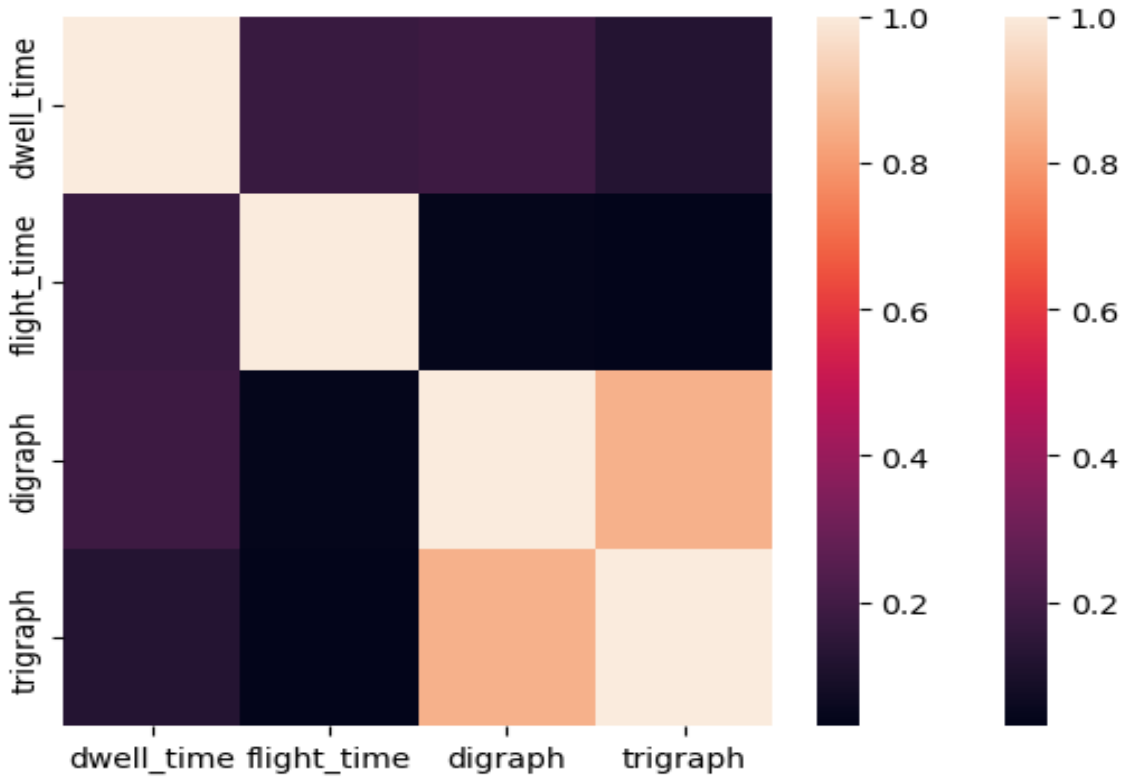
Performance Evaluation of the Model




```
svm_eer
```

```
0.18218623481781376
```

Figure 5.13 Correlation graph for SVM model with EER Score




```
eer_manhattan
```

```
0.21695257315842584
```

Figure 5.14 Correlation graph for MDF model with EER Score

Algorithm Comparison

Algorithm comparison

```
threshold=30
if svm_eer < eer_manhattan:
    best_eer=svm_eer
    Alg = "Support Vector Machine"

elif svm_eer > eer_manhattan:
    best_eer=eer_manhattan
    Alg = "Manhattan Filter"

if best_eer < threshold:
    user = "Genuine user"

else:
    user = "Imposter"
```

Figure 5.15 Comparing SVM and MDF model

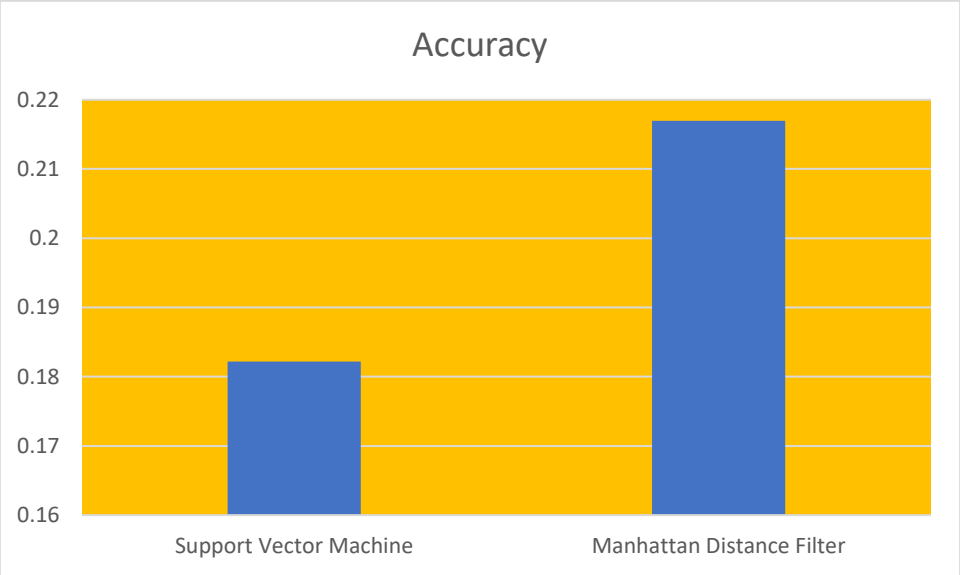
```
result_df
```

	User	Authenticity	Algorithm	EER Rate
0	s051	Genuine user	Support Vector Machine	0.182186

Figure 5.16 Authenticated User

Figure 5.16 shows that the result of the authenticated user by comparing the Support Vector Machine and Manhattan Distance Filter model and it derives that SVM model gives us the lower EER score.

Accuracy



	User	Authenticity	Algorithm	EER Rate
0	s051	Genuine user	Support Vector Machine	0.182186

Figure 5.17 Accuracy Score

6. CONCLUSION AND FUTURE SCOPE

Previously, it was not possible to compare the performance of different anomaly detectors across studies in the keystroke dynamics literature. Our objective in this work to develop an evaluation procedure, and measure the performance of many algorithms on an equal basis.

In the process, we established which detectors have the lowest error rates on our data (e.g SVM), and we provide a data set and evaluation methodology that can be used by the community to assess new detectors and report comparative results..

In summary, the future scope of keystroke authentication using SVM and Manhattan distance filter is vast, and there are several exciting directions for research and development. With the advancement of technology and the increasing demand for secure authentication systems, the field of keystroke authentication is expected to continue to grow and evolve in the coming years. Also for the extra layer of security Image Captcha can be included and authenticate the user in the basis of Multifactor Authentication.

These create the basis of a potentially effective way of enhancing overall security rating by playing a significant role in part of a larger multifactor authentication mechanism low implementation cost, high user acceptance, and ease of integration to existing security systems.

6.1 Outcome

In this project SVM and MDF Machine learning models are developed to identify the genuine imposter user with user authentication-based Keystroke Dynamic. By evaluating the SVM and MDF model on the basic of EER from figure 5.13 and figure 5.14 it is evident that the EER value is lower for SVM model with 0.18 compared with MDF model with equal ERR 0.218. Based on the analysis it suggested that SVM model identifies the genuine user more precisely for user authentication problem

7.REFERENCES

[1] Tewari, Anurag, and Prabhat Verma. "An improved user identification based on keystroke-dynamics and transfer learning." WEB. Vol. 19. 2022.

[2] Alsuhibany, Suliman A., and Latifah A. Alreshoodi. "Detecting human attacks on text-based CAPTCHAs using the keystroke dynamic approach." *IET Information Security* 15.2 (2021): 191-204.

[3] Alamri, Emtethal K., Abdullah M. Alnajim, and Suliman A. Alsuhibany. "Investigation of using captcha keystroke dynamics to enhance the prevention of phishing attacks." Future Internet 14.3 (2022): 82.

[4] Migdal, Denis. Contributions to keystroke dynamics for privacy and security on the Internet. Diss. Normandie Université, 2019.

[5] Ali, Md Liakat, et al. "Keystroke biometric systems for user authentication." Journal of Signal Processing Systems 86 (2017): 175-190.

[6] Von Ahn, Luis, et al. "recaptcha: Human-based character recognition via web security measures." *Science* 321.5895 (2008): 1465-1468.

[7] Andrean, Alvin, Manoj Jayabalan, and Vinesh Thiruchelvam. "Keystroke dynamics based user authentication using deep multilayer perceptron." *International Journal of Machine Learning and Computing* 10.1 (2020): 134-139.

[8] Killourhy, Kevin S., and Roy A. Maxion. "Comparing anomaly-detection algorithms for keystroke dynamics." *2009 IEEE/IFIP International Conference on Dependable Systems & Networks*. IEEE, 2009.

[9] Kar, Soumyatattwa, et al. "KeyDetect--Detection of anomalies and user based on Keystroke Dynamics." *arXiv preprint arXiv:2304.03958* (2023).

[10] Singh and Saxena. "Keystroke-Based User Authentication System Using Support Vector Machine", WEB. Vol. 19. 2021

Dataset Link

https://www.cs.cmu.edu/~keystroke/

Website Reference

https://vitalflux.com/pandas-dataframe-how-to-add-rows-columns/#:~:text=There%20are%20multiple%20ways%20of,use%20Dataframe.insert()%20method.

https://trumpexcel.com/apply-formula-to-entire-column-excel/

https://scikit-learn.org/stable/modules/generated/sklearn.metrics.pairwise.manhattan_distances.html

8.APPENDIX

8.1 Sample Coding

Importing Libraries

```
import pandas as pd
from sklearn.preprocessing import LabelEncoder
from sklearn.feature_selection import SelectKBest, f_classif
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score
result_df = pd.DataFrame(columns=['User','Authenticity','Algorithm','EER Rate'])
```

Loading the Dataset

```
keystroke_data = pd.read_csv('D:\Project\Project dataset\Raw_DST.csv')
```

Data Pre-processing

```
##DATA PREPROCESSING##
##Checking null values##
keystroke_data.isnull()
#Duplicate Entries
duplicates = keystroke_data.duplicated()
print(keystroke_data[duplicates])
```

Feature Extraction

```
df=keystroke_data.loc[:, "H.period":"trigraph"]
df
keystroke_data
pd.Series(keystroke_data.values[:, -1], name=keystroke_data.columns[-1])
X=keystroke_data[['dwell_time', 'flight_time','digraph', 'trigraph']]
print(keystroke_data[['dwell_time', 'flight_time', 'digraph', 'trigraph']])
```

SVM for Modeling

```
from sklearn.svm import OneClassSVM
import numpy as np
np.set_printoptions(suppress = True)
import pandas
import random
import scipy.stats as stats
from EER import evaluateEER
from sklearn.cluster import KMeans, AgglomerativeClustering
from sklearn.metrics.cluster import normalized_mutual_info_score, adjusted_rand_score

class SVMDetector:
#just the training() function changes, rest all remains same.

 def __init__(self, subjects):
        self.u_scores = []
        self.i_scores = []
        self.mean_vector = []
        self.subjects = subjects

 def training(self):
        self.clf = OneClassSVM(kernel='rbf',gamma=26)
        self.clf.fit(self.train)

    def testing(self):
        self.u_scores = -self.clf.decision_function(self.test_genuine)
        self.i_scores = -self.clf.decision_function(self.test_imposter)
        self.u_scores = list(self.u_scores)
        self.i_scores = list(self.i_scores)
```

```
def evaluate(self):
    eers = []
    print(subjects)
    genuine_user_data = keystroke_data.loc[keystroke_data.subject == subjects, \
                                "dwell_time":"trigraph"]
    print(genuine_user_data)
    # prints data that will be plotted
    # columns shown here are selected by corr() since
    # they are ideal for the plot
    import seaborn as sb
    print(genuine_user_data.corr())

    # plotting correlation heatmap
    dataplot=sb.heatmap(genuine_user_data.corr())

    # displaying he
    # We are going to use the K-Means algorithm
    model_kmeans = AgglomerativeClustering(n_clusters=2)
    preds = model_kmeans.fit_predict(genuine_user_data)
    print(preds)

    # We are going to use the NMI metric to measure the quality/performance of the clustering
    baseline_score = normalized_mutual_info_score(model_kmeans.labels_, preds)
    print("Baseline NMI Score:", baseline_score)

    print("Selected Features : ", model_kmeans.feature_names_in_)

    imposter_data = keystroke_data.loc[keystroke_data.subject != subjects, :]

    #feature Reduction
    self.train = genuine_user_data[:200]
    self.test_genuine = genuine_user_data[200:]
    self.test_imposter = imposter_data.groupby("subject"). \
```

```
            head(5).loc[:, "dwell_time":"trigraph"]

            self.training()
            self.testing()
            eers.append(evaluateEER(self.u_scores, \
                                    self.i_scores))
            return np.mean(eers)
        subjects = keystroke_data["subject"].unique()
        print(subjects)
        subjects=random.choice(subjects)
        subject_choosen=subjects
        subject_choosen



        print ("average EER for SVM detector:")
        print(SVMDetector(subjects).evaluate())
        svm_eer=SVMDetector(subjects).evaluate()
        svm_eer
```

Manhattan Distance Filter for Modeling

```
        from scipy.spatial.distance import euclidean, cityblock
        import numpy as np
        np.set_printoptions(suppress = True)
        import pandas
        from EER import evaluateEER
        import seaborn as sb

        class ManhattanFilteredDetector:
        #just the training() function changes, rest all remains same.

         def __init__(self, subjects):
         self.user_scores = []
         self.imposter_scores = []
         self.mean_vector = []
```

```
self.subject_choosen = subject_choosen

def training(self):
 self.mean_vector = self.train.mean().values
 self.std_vector = self.train.std().values
 dropping_indices = []
  for i in range(self.train.shape[0]):
      cur_score = euclidean(self.train.iloc[i].values,
                        self.mean_vector)
      if (cur_score > 3*self.std_vector).all() == True:
          dropping_indices.append(i)
  self.train = self.train.drop(self.train.index[dropping_indices])
  self.mean_vector = self.train.mean().values

 def testing(self):
     for i in range(self.test_genuine.shape[0]):
         cur_score = cityblock(self.test_genuine.iloc[i].values, \
                             self.mean_vector)
         self.user_scores.append(cur_score)

     for i in range(self.test_imposter.shape[0]):
         cur_score = cityblock(self.test_imposter.iloc[i].values, \
                             self.mean_vector)
         self.imposter_scores.append(cur_score)

 def evaluate(self):
     eers = []

print(subjects)
genuine_user_data = keystroke_data.loc[keystroke_data.subject == subject_choosen, \
                             "dwell_time":"trigraph"]
print(genuine_user_data)
# prints data that will be plotted
```

```
# columns shown here are selected by corr() since
# they are ideal for the plot
import seaborn as sb
print(genuine_user_data.corr())

# plotting correlation heatmap
dataplot=sb.heatmap(genuine_user_data.corr())

# displaying he
# We are going to use the K-Means algorithm
model_kmeans = AgglomerativeClustering(n_clusters=2)
preds = model_kmeans.fit_predict(genuine_user_data)
print(preds)

# We are going to use the NMI metric to measure the quality/performance of the clustering
baseline_score = normalized_mutual_info_score(model_kmeans.labels_, preds)
print("Baseline NMI Score:", baseline_score)

print("Selected Features : ", model_kmeans.feature_names_in_)
imposter_data = data.loc[data.subject != subject_choosen, :]
self.train = genuine_user_data[:200]
self.test_genuine = genuine_user_data[200:]
self.test_imposter = imposter_data.groupby("subject"). \
                    head(5).loc[:, "dwell_time":"trigraph"]
self.training()
self.testing()
eers.append(evaluateEER(self.user_scores, \
                        self.imposter_scores))
return np.mean(eers)
```

```
path = "D:\Project\Project dataset\Raw_DST.csv"
data = pandas.read_csv(path)
#subjects = data["subject"].unique()
subject_choosen
print ("average EER for Manhattan Filtered detector:")
print(ManhattanFilteredDetector(subject_choosen).evaluate())
eer_manhattan=ManhattanFilteredDetector(subject_choosen).evaluate()
eer_manhattan
```

Algorithm Comparsion

```
threshold=30
if svm_eer < eer_manhattan:
 best_eer=svm_eer
 Alg = "Support Vector Machine"

elif svm_eer > eer_manhattan:
 best_eer=eer_manhattan
 Alg = "Manhattan Filter"

if best_eer < threshold:
 user = "Genuine user"

else:
 user = "Imposter"
```

Dataframe

```
#result=[subject_choosen,user,Alg, best_eer]
# Define the new row to be added
new_row = {'User': subject_choosen, 'Authenticity': user, 'Algorithm': Alg, 'EER Rate' : best_eer}

# Use the loc method to add the new row to the DataFrame
result_df.loc[len(result_df)] = new_row
result_df
```