Authentication using Biometric Keystroke Dynamics

Siddharth Varshney *2K17/CO/340*

INTRODUCTION

With the prevalence of Skype, Line, WeChat, and other instant messaging (IM) software, frauds involving IM applications have increased significantly. For example, users clicking on an unknown website displayed in certain IM software, causing usernames and passwords to be stolen and used by scam gangs to defraud the user's friends or families. In another typical scam, a scam gang steals a user's username and password from the IM software and, subsequently, sends messages to the user's friends by requesting the purchase of game points. To strengthen the security of usernames and passwords, researchers have recently applied different biometric techniques relating to iris, retina, fingerprint, face, palm print, voice, online signature, or **keystroke dynamics as login-identification mechanisms**. However, biometrics identification technologies for retina, fingerprint, and face require additional hardware costs. Because a keyboard, a typical computer peripheral, which does not require any changes in computer-use habits, most recent studies have focused on keystroke dynamics as a method of increasing user account security. In recent years, the method of keystroke dynamics has been applied in the research of smartphones, touch devices, and emotion recognition. However, traditional keystroke-dynamics verification methods are not applicable to detect fraudulent messages in IM software.

Keystroke dynamics is the study of the typing patterns of people to distinguish them from one another, based on of these patterns. Every user has a certain way of typing that separates him from other users; for example, for how long does a user press the keys, how much time between consecutive key presses, etc.

Keystrokes are an upcoming area of research in biometrics. One famous application is the Coursera website using a typing test as the method to verify the students. No additional hardware is required to collect keystrokes; only a keyboard. Keystroke dynamics, together with security measures already in place (eg, password) can hence serve as a convenient 2-factor authentication.

Problem Statement

Design a technique which distinguishes between real users and fake users depending on their typing speed, rhythm, keypress time etc. This can be achieved by using Convolutional Neural Network. Our main function is to identify factors of keyboard flight time, dwell time, frequency of typing errors and use of particular control keys which is the most indicative of typing habits. As such, these factors guided our selection of typing features to examine. To explain more about our project, we have used keystroke dynamics which is the detailed timing information which describes exactly when each key was pressed and when it was released as a person is typing at a computer keyboard.

The keystroke rhythms of a user are measured to develop a unique biometric template of the user's typing pattern for future authentication. Keystrokes are separated into static and dynamic typing, which are used to help distinguish between authorized and unauthorized users. Vibration information may be used to create a pattern for future use in both identification and authentication tasks.

Motivation

Nowadays social media has become a daily life use for almost their social media handle has become their virtual representation. With this social media misuse has become very common and there is a need to make these social media platforms more secure as we have seen in recent past how bill gates twitter account was used to fraud hundreds of people.

So our motivation for the project is to make a system that will make ensure incidents like these don’t take place in future. Making social media websites safe for everyone use.

1. Securing the sensitive data and computer systems by allowing ease access to authenticated users and withstanding the attacks of imposters is one of the major challenges in the field of computer security. Traditionally, ID and password schemes are most widely used for controlling the access to computer systems. But, this scheme has many flaws such as Password sharing, Shoulder surfing, Brute force attack, Dictionary attack, Guessing, Phishing and many more. Biometrics technologies provide more reliable

and efficient means of authentication and verification. Keystroke Dynamics is one of the famous biometric technologies, which will try to identify the authenticity of a user when the user is working with a keyboard. In this paper, neural network approaches with three different passwords namely weak, medium and strong passwords are taken into consideration and accuracy obtained is compared.

2. Almost all the people rely on computers at a certain level in day today life. Many of these systems store highly sensitive, personal, commercial, confidential or financial data. Unauthorized access to such data will lead to loss of money or unwanted disclosure of highly confidential data by threatening the Information security. The first and foremost step in preventing unauthorized access of information for providing information security is user authentication. Biometric-based authentication is one of the authentication techniques which is based on something you are and depends on behavioral and physiological characteristics of individuals. In this authentication, the person presents a characteristic that cannot be forged and nor be forgotten. Biometrics is classified into physiological and behavioral biometrics. Physiological biometric refers to what the person is and the Behavioral biometrics are related to what a person does, or how the person uses the body. Keystroke dynamics is considered as a strong behavioral biometric based authentication system.

3. It is a process of analyzing the way a user types at a terminal by monitoring the keyboard in order to identify the Keystroke Dynamics Authentication Using Neural Network Approaches 687 users based on habitual typing rhythm patterns. Moreover, unlike other biometric systems which may be expensive to implement, keystroke dynamics is cheap and does not require any sophisticated hardware as the only hardware required is the keyboard.

4. Keystroke analysis contains two approaches: static and dynamic. In static approach, the system checks the user one time, that is at authentication time. In the dynamic approach, the system checks the user continuously throughout the session.

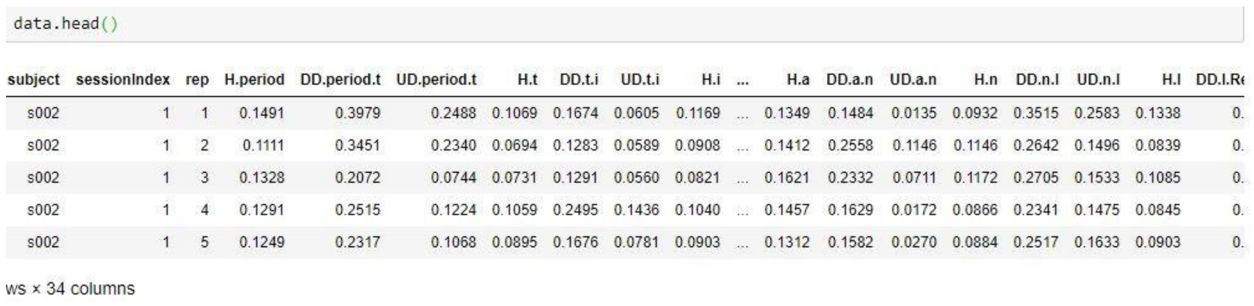
Objectives

Objectives of the project is to make a system that will

- Authenticate users on the basis of their keystroke dynamics.
- With increasing security concerns in traditional approaches like username password keystroke dynamics will be a key changer in the field of online security and human identification.
- Make online communication more secure as at any time to use the system we require the person not his/her username and password.

Datasets Used

To interpret the values given in the dataset. Given below are rows from the .csv file in the dataset.



```
data.head()
```

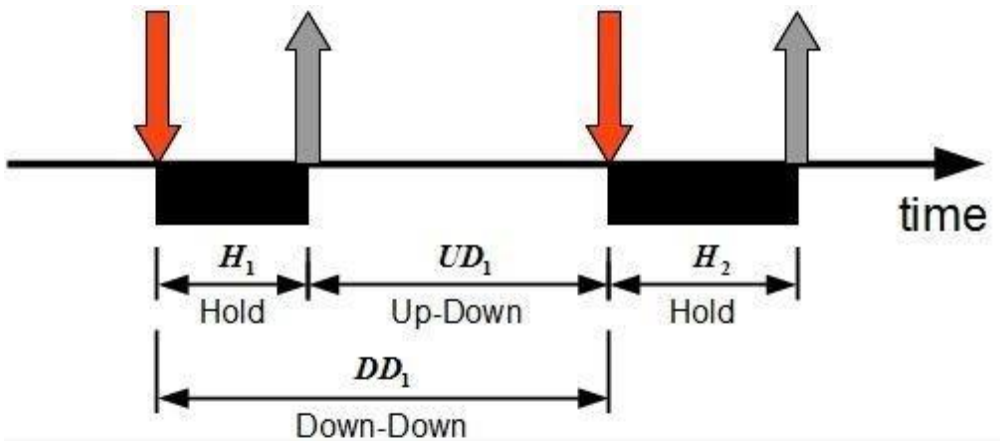
	subject	sessionIndex	rep	H.period	DD.period.t	UD.period.t	H.t	DD.t.i	UD.t.i	H.i	...	H.a	DD.a.n	UD.a.n	H.n	DD.n.l	UD.n.l	H.l	DD.l.Re
	s002	1	1	0.1491	0.3979	0.2488	0.1069	0.1674	0.0605	0.1169	...	0.1349	0.1484	0.0135	0.0932	0.3515	0.2583	0.1338	0.
	s002	1	2	0.1111	0.3451	0.2340	0.0694	0.1283	0.0589	0.0908	...	0.1412	0.2558	0.1146	0.1146	0.2642	0.1496	0.0839	0.
	s002	1	3	0.1328	0.2072	0.0744	0.0731	0.1291	0.0560	0.0821	...	0.1621	0.2332	0.0711	0.1172	0.2705	0.1533	0.1085	0.
	s002	1	4	0.1291	0.2515	0.1224	0.1059	0.2495	0.1436	0.1040	...	0.1457	0.1629	0.0172	0.0866	0.2341	0.1475	0.0845	0.
	s002	1	5	0.1249	0.2317	0.1068	0.0895	0.1676	0.0781	0.0903	...	0.1312	0.1582	0.0270	0.0884	0.2517	0.1633	0.0903	0.

ws × 34 columns

The 3 most widely used features for keystroke dynamics are:

1. <u>Hold time</u> – time between press and release of a key.
2. <u>Keydown-Keydown time</u> – time between the pressing of consecutive keys.
3. <u>Keyup-Keydown time</u> – time between the release of one key and the press of next key.

The CMU keystroke data-set provides us with these three features. The below picture depicts the three time periods used as features in keystroke data-set.



These feature vector contains the data for subject s002 (user) from his first session and first repetition in that session. The feature H.*key* tells the hold time for the key; eg, H.period is the hold time for the dot key. DD.*key1.key2* is the keydown-keydown time for the key pair key1 and key2; eg, DD.period.t is the keydown-keydown time for pressing the dot key and the t key. Lastly, UD.*key1.key2* is the keyup-keydown time for the keys key1 and key2; eg, UD.period.tis the keyup-keydown time for the dot key and tkey.

The columns 4-34 thus have these time information for the various keys and key pairs in .tie5Roanl. There are 8 sessions per user and 50 repetitions per session, hence 400 keystroke vectors for one user.

The Python module pandas has been used to load the keystroke.csv file. The data variable is a pandadata frame and contains the entire contents of the .csvfile. subjectsarray has the labels of all the 51 subjects.

System Requirements and Language Used for Coding

Operating System: Windows 7 or 10, Mac OS X 10.11 or higher, 64-bit, Linux:

RHEL 6/7, 64-bit

CPU: x86 64-bit (Intel / AMD architecture)

RAM: 4 GB

Storage: 4 GB free disk space

Programming Language: Python (version 3.6 or higher)

IDE: Jupyter Notebook/Google Colab

Packages:

1. Pip
2. Numpy
3. Pandas
4. Tensorflow
5. Matplotlib
6. Scipy
7. scikit-learn

Proposed Methodology

How is the model getting trained?

From the 400 vectors (repetitions of password) available for every user, we employ the first 200 for training the model for the user, to capture his typing behaviour. Train variable (a pandadata frame) contains the training data.

We will be presenting the results on 5 different learning models.

- Manhattan Distance
- Manhattan Filtered Distance
- Manhattan Scaled Distance
- Gaussian Mixture Model (GMM)
- Support Vector Machines (SVM)

Our first detector is Manhattan Detector (MD). The taining() function in the Python code calculates the mean_vector for each user from the samples in train(training set). Here, only the mean of feature vectors is considered as a user model.

Performance Evaluation

We will now supply the model of a user with unseen test sample as explained here- a test_genuine list which has the remaining 200 feature vectors (repetitions of password) of the same user and a test_imposter list which has 5 vectors each from all the other 50 users, making a total of 250 imposter samples. In total, each user will be tested 450 times for user's authenticity. Again, keep in mind that this process will be repeated for all the 51 users.

Performance evaluation through Manhattan Distance

The detector calculates the city block distance (Manhattan distance) between the test samples and mean_vector for the subject. Its easy to understand that smaller values of this distance indicate higher similarity of the sample to the subject's model; however if the score has a larger value, it means the sample is quite dissimilar to the model and should get not get verified as the subject.

Having obtained the scores for both kind of samples, genuine and imposter, in lists user_scores and imposter_scores respectively, we evaluate the equal error rate (EER) for the detector. We report the average EER as the performance metric for the detectors.

Various types of other Detectors

Manhattan Filtered Detector (MFD)

The training() function of the MFD takes into account any outliers in a subject's typing habits. Such deviations from his/her usual typing habits may occur due to a variety of reasons, like the user being tired or bored and hence typing exceptionally slower than normal, etc. MFD simply filters (removes) such outliers.

We calculate mean_vector and standard deviation vector, std_vector for the user from his training vectors. To reject the outliers, first the euclidean distance between each of the training vectors and mean_vector is determined. Then, any vector for whom this distance is greater than three times the std_vector is an outlier and is dropped from train. Dropping_indices consists of the indices of all such outliers. Having eliminated all such training vectors from the set, mean_vector for the user is re-calculated from the remaining samples in train. This mean_vector is now the model for the user's typing behavior.

Manhattan Scaled detector (MSD)

While training, we calculate the `mean_vector()` as well as the `mad_vector()` which has the mean absolute deviation (MAD) of each feature of the training data. In `testing()`, score for a test sample is being calculated as $\sum_{i=1}^{p} \frac{|x_i - y_i|}{\alpha_i}$, where x_i and y_i are the i^{th} feature in the test sample and `mean_vector()` respectively, and α_i is that feature's MAD , taken from the `mad_vector()`. Thus, we are essentially calculating the city-block distance but each feature is getting scaled by its MAD.

One-Class SVM

One class SVMs learn a decision function from the data of one class only and test a new sample to found out whether it is like the training data or not. This does exactly what we need, right? We will use sklearn.svm.OneClassSVM sub-module's fit() function to train a OneClassSVM object, named clf here, and the decision_function() function to calculate the similarities scores for the test samples.

Manhattan Filtered Scaled Detector

This model is a combination of previous two models, Manhattan Filtered and Manhattan Scaled. Just like Manhattan Filtered the training() function of the takes into account any outliers in a subject's typing habits. Such deviations from his/her usual typing habits may occur due to a variety of reasons, like the user being tired or bored and hence typing exceptionally slower than normal, etc.

While training, we calculate the `mean_vector()` as well as the `mad_vector()` which has the mean absolute deviation (MAD) of each feature of the training data. In `testing()`, score for a test sample is being calculated as $\sum_{i=1}^{p} \frac{|x_i - y_i|}{\alpha_i}$, where x_i and y_i are the i^{th} feature in the test sample and `mean_vector()` respectively, and α_i is that feature's MAD , taken from the `mad_vector()`.

One-Class SVM Filtered

This model is a combination of the previous two models, Manhattan Filtered and SVM detector. Just like Manhattan Filtered the training() function of the takes into account any outliers in a subject's typing habits. Such deviations from his/her usual typing habits may occur due to a variety of reasons, like the user being tired or bored and hence typing exceptionally slower than normal, etc.

One class SVMs learn a decision function from the data of one class only and test a new sample to find out whether it is like the training data or not. This does exactly what we need. We will use sklearn.svm.OneClassSVM sub-module's fit() function to train a OneClassSVM object, named clf here, and the decision_function() function to calculate the similarities scores for the test samples.

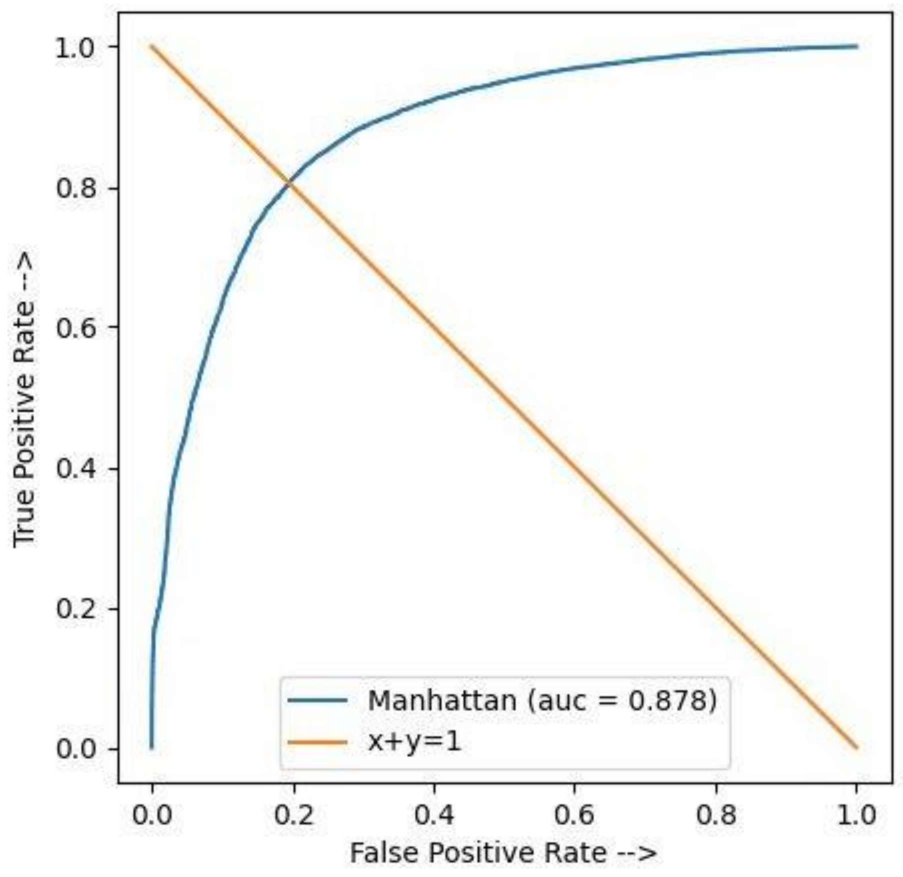
Results and Graphs

Performance Evaluation Graphs through ROC Curve

The area under the ROC curve signifies the accuracy of the predictions. More the area under the curve in comparison to a similar curve, more accuracy that model will have.

Manhattan Distance

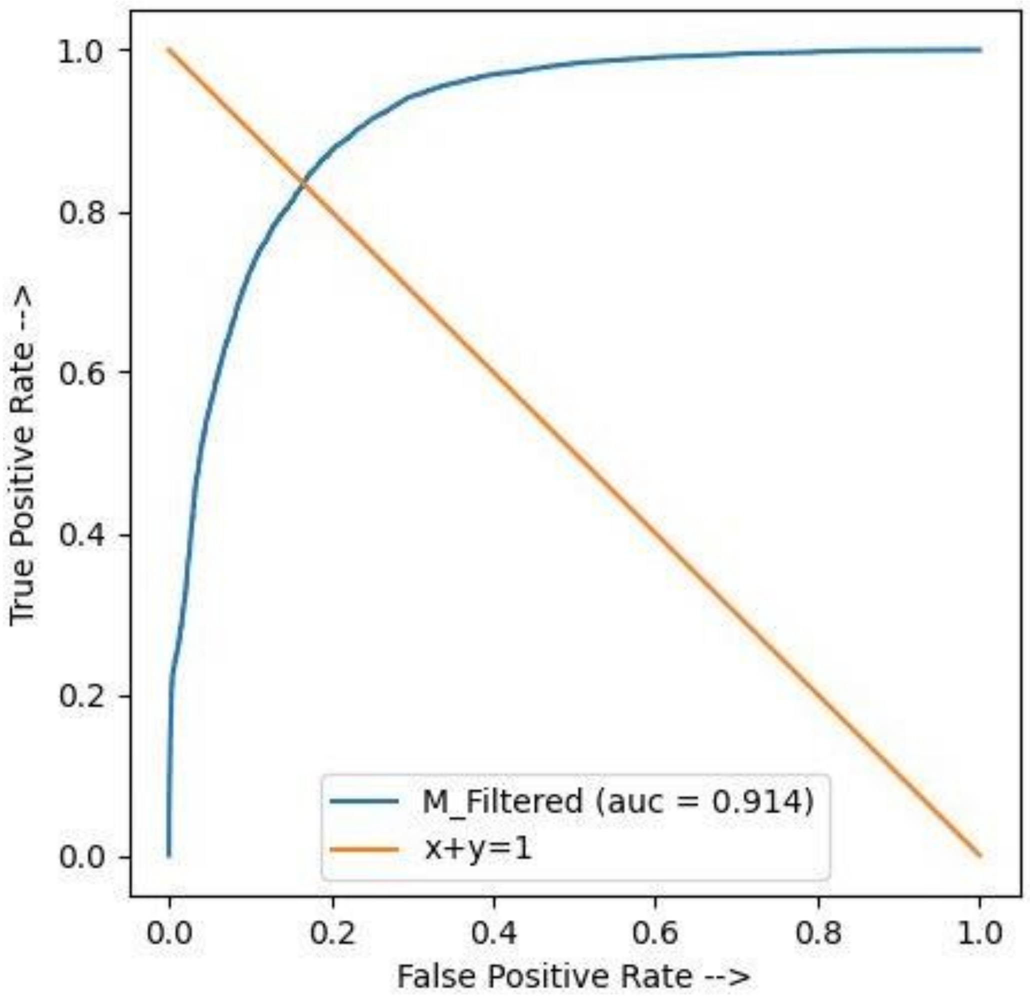
The detector calculates the city block distance (Manhattan distance) between the test samples and mean_vectorfor the subjects.



Area under the ROC curve is 0.878 out of 1. As we can see, to increase true positives we will have to endanger security of the model. If we take the threshold below 8 of true positive rate, False positive rate is negligible.

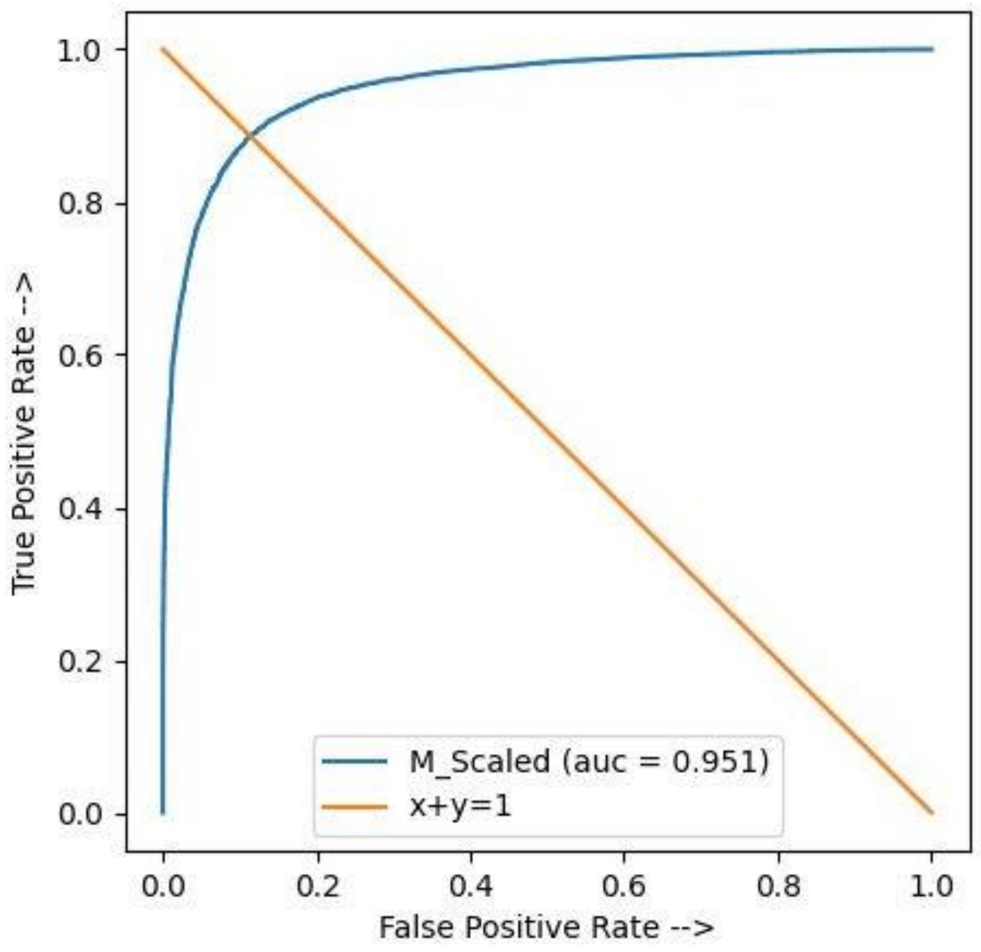
Manhattan Distance Filtered

The training() function of the MFD takes into account any outliers in a subject’s typing habits. Such deviations from his/her usual typing habits may occur due to a variety of reasons, like the user being tired or bored and hence typing exceptionally slower than normal, etc. MFD simply filters (removes) such outliers.



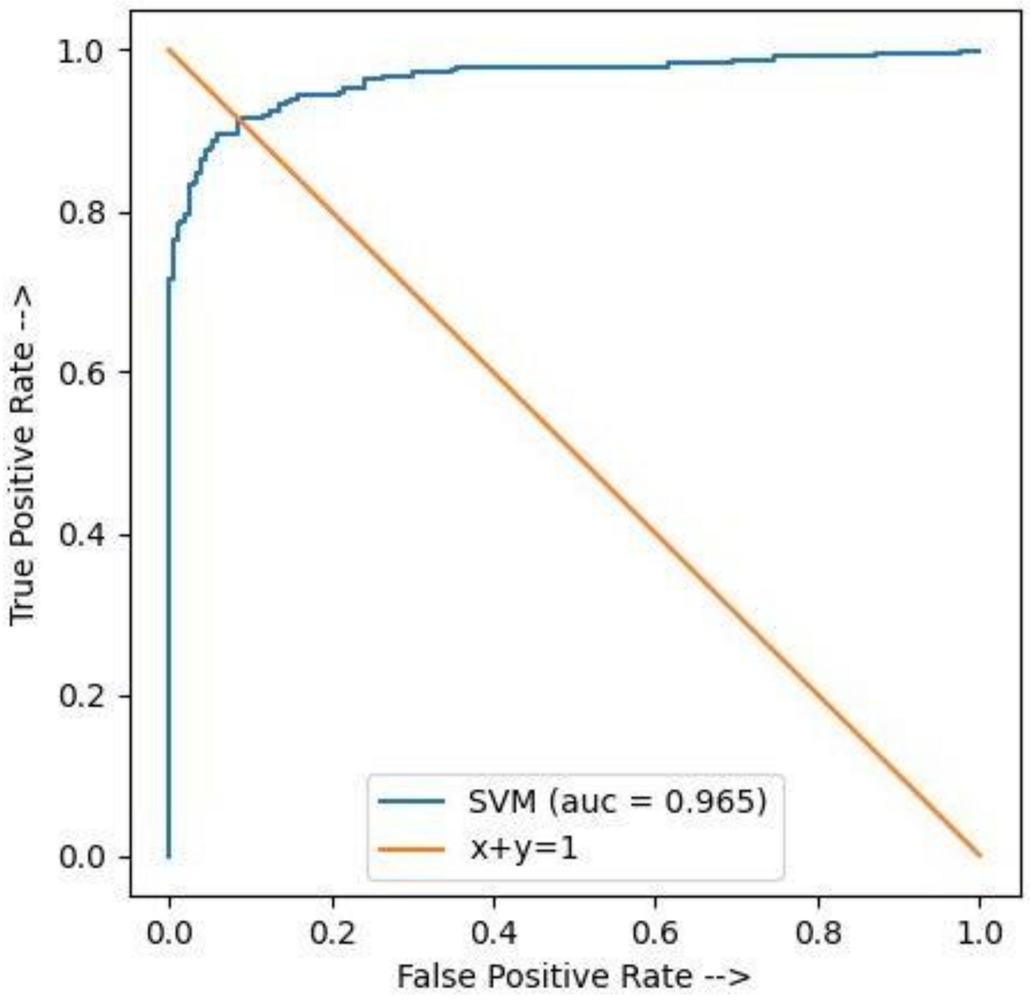
In this model, we have slightly more area than the regular manhattan model, which signifies greater true positives at equal amount of security breacher, or we say, less security breaches at same number of secure logins. This would also mean that the equal error rate is lower than the regular manhattan model.

Manhattan Distance Scaled



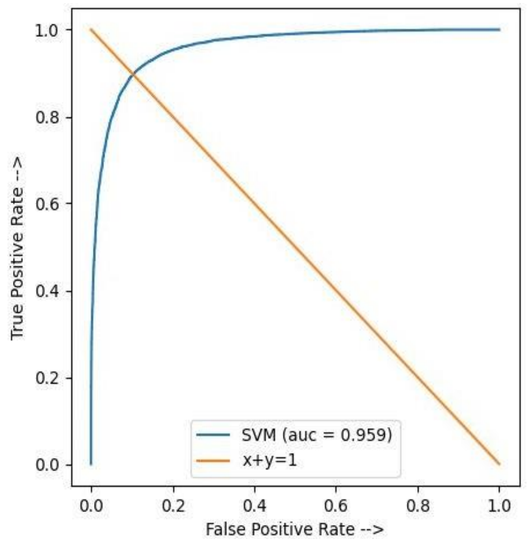
In this model, we have slightly more area than the regular manhattan and manhattan filtered model, which signifies greater true positives at equal amount of security breacher, or we say, less security breaches at same number of secure logins. This would also mean that the equal error rate is lower than the regular manhattan model and manhattan filtered model.

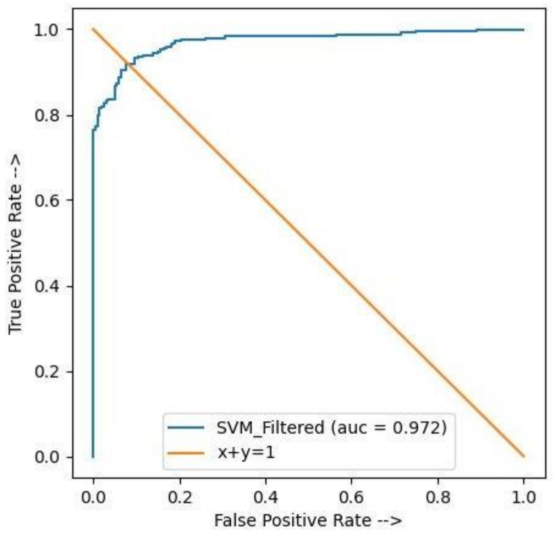
SVM(Support Vector Machine)



SVM is one of the most powerful classifiers in Machine learning. In Keystroke authentication, SVM has the highest area under curve among four models. As we can see , until approximately 0.75 true positive rate, the model can easily identify every false login attempt. To make machine breach proof, nearly 1 in every 4 true attempts is also classified as a false attempt, but this a perfect threshold for high security authentication.

Manhattan Filtered scaled and One-class SVM Filtered



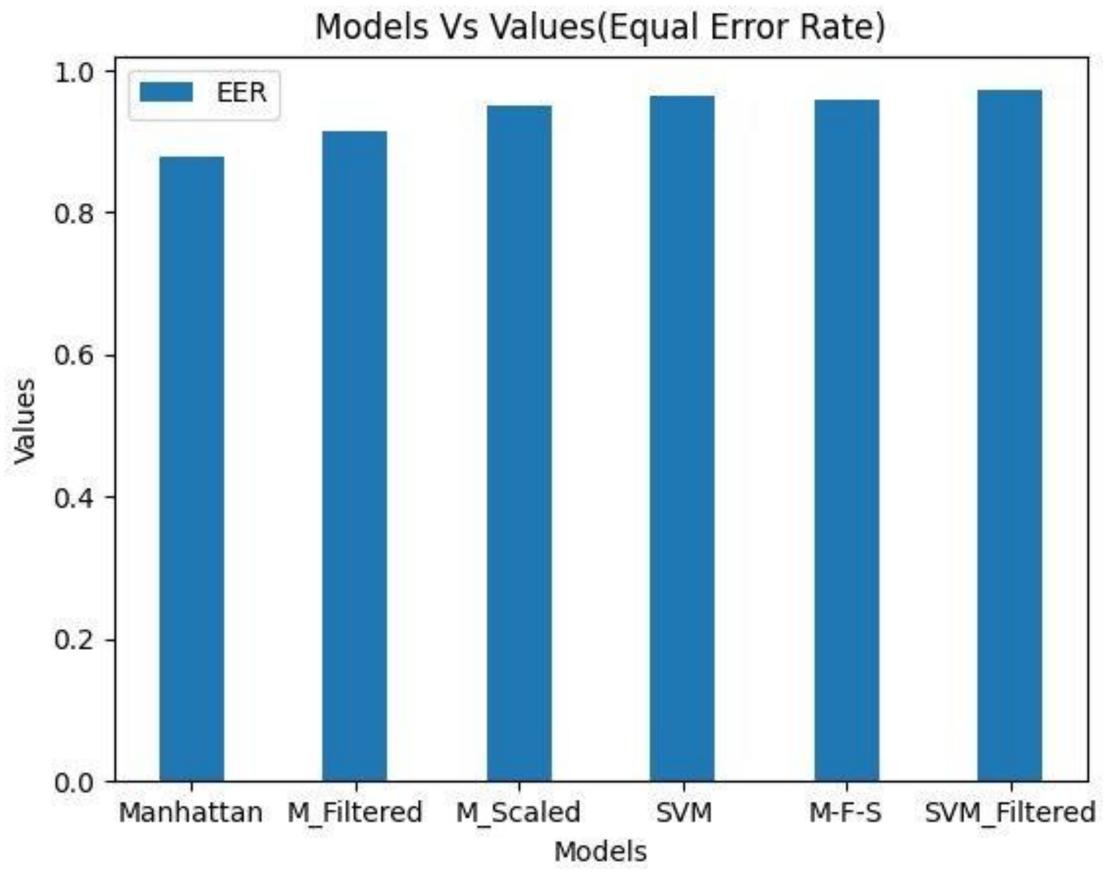


Filtering the training data before feeding in the model gives us fairly better results on previously implemented algorithms.

AUC of SVM Filtered and Manhattan Filtered Scaled have both increased significantly.
AUC of Manhattan Filtered Scaled is still less than that of SVM. This gives us proof that SVM is the most powerful classification algorithm here.

EER value of One-Class SVM Filtered is less than EER value of Manhattan Filtered Scaled. This gives us the idea that EER is just a means to give us an idea where to place the threshold where x(axis) +y(axis) = 1. EER value has a smaller influence in comparing the better algorithm.

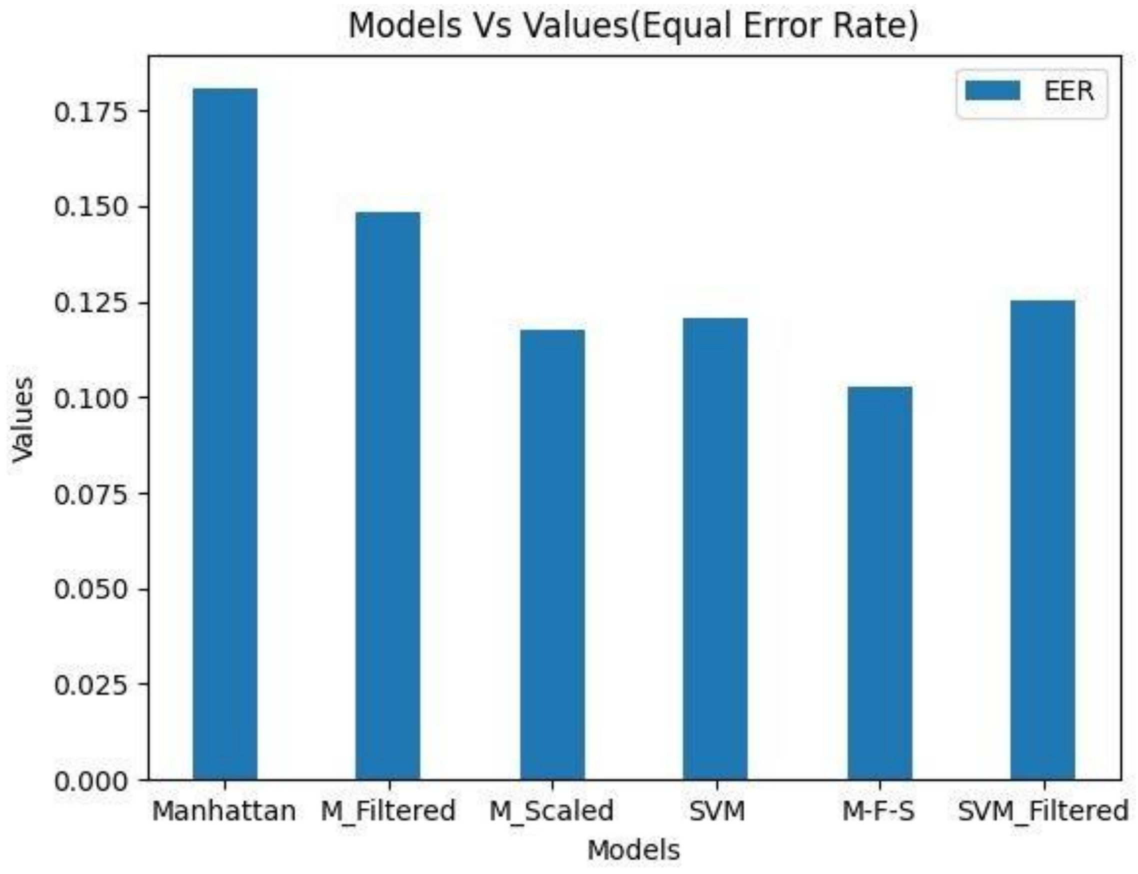
Area under curve graphical representation comparison



The area under the curve represents how steep is the initial part of the graph where the threshold is calculated. AT the threshold, we get maximum true positive rate with minimal false positive rate which means maximum security even if it costs a little false negatives in login decisions.

EQUAL ERROR RATE (EER) comparison

Model	Rate
Manhattan -	0.18065765645731371
Manhattan Filtered -	0.1484870487058271
Manhattan Scaled -	0.11763692496206396
SVM -	0.12054244703221502
M Filtered Scaled-	0.10284572835503178
SVM Filtered-	0.12557263803596627



Equal error rate is the point at x-axis at which line (0,1)-(1,0) intersects the ROC curve. Lower the x-axis means lower equal error rate which in turn means less number of false security access by third parties and lower number of cases where true login is identified as a security breach.

References

1. Keyword-based approach for recognizing fraudulent messages by keystroke dynamics, 2020, https://www.sciencedirect.com/science/article/abs/pii/S0031320319303681

2. Keystroke dynamics: Characteristics and opportunities https://ieeexplore.ieee.org/document/5593258

3. A Survey of Keystroke Dynamics Biometrics https://www.hindawi.com/journals/tswj/2013/408280/

4. Comparing Anomaly-Detection Algorithms for Keystroke Dynamics https://www.cs.cmu.edu/~maxion/pubs/KillourhyMaxion09.pdf

Summary and conclusion

Previously, it was not possible to compare the performance of different anomaly detectors across studies in the keystroke dynamics literature. The objective in this work has been to collect a data set, develop an evaluation procedure, and measure the performance of many anomaly detection algorithms on an equal basis. In the process, we established which detectors have the lowest error rates on our data (e.g., SVM detector).