7-29-2020

Towards Engineering Reliable Keystroke Biometrics Systems

Anjali Parag Shah
University of Windsor

Towards Engineering Reliable Keystroke Biometrics Systems

By

Anjali Shah

A Thesis
Submitted to the Faculty of Graduate Studies
through the School of Computer Science
in Partial Fulfillment of the Requirements for
the Degree of Master of Science
at the University of Windsor

Windsor, Ontario, Canada

2020

Towards Engineering Reliable Keystroke Biometrics Systems

by

Anjali Shah

APPROVED BY:

A. Azab
Department of Mechanical, Automotive and Materials Engineering

---

B. Boufama
School of Computer Science

S. Saad, Advisor
School of Computer Science

June 11, 2020

DECLARATION OF ORIGINALITY

I hereby certify that I am the sole author of this thesis and that no part of this thesis has been published or submitted for publication.

I certify that, to the best of my knowledge, my thesis does not infringe upon anyone's copyright nor violate any proprietary rights and that any ideas, techniques, quotations, or any other material from the work of other people included in my thesis, published or otherwise, are fully acknowledged in accordance with the standard referencing practices. Furthermore, to the extent that I have included copyrighted material that surpasses the bounds of fair dealing within the meaning of the Canada Copyright Act, I certify that I have obtained a written permission from the copyright owner(s) to include such material(s) in my thesis and have included copies of such copyright clearances to my appendix.

I declare that this is a true copy of my thesis, including any final revisions, as approved by my thesis committee and the Graduate Studies office, and that this thesis has not been submitted for a higher degree to any other University or Institution.

ABSTRACT

In this thesis, we argue that most of the work in the literature on behavioural-based biometric systems using AI and machine learning is immature and unreliable. Our analysis and experimental results show that designing reliable behavioural-based biometric systems requires a systematic and complicated process. We first discuss the limitation in existing work and the use of conventional machine learning methods. We use the biometric zoos theory to demonstrate the challenge of designing reliable behavioural-based biometric systems. Then, we outline the common problems in engineering reliable biometric systems. In particular, we focus on the need for novelty detection machine learning models and adaptive machine learning algorithms. We provide a systematic approach to design and build reliable behavioural-based biometric systems. In our study, we apply the proposed approach to keystroke dynamics. Keystroke dynamics is behavioural-based biometric that identify individuals by measuring their unique typing behaviours on physical or soft keyboards. Our study shows that it is possible to design reliable behavioral-based biometrics and address the gaps in the literature.

DEDICATION

I would like to dedicate this thesis to my mom for her incredible love and support. Because I believe that she is the real backbone of our family, this is to appreciate her selfless hard work and efforts towards the family.

Furthermore, I dedicate it to my dad to raise me like a son and give me wings to fly. To my grandfather, for always trusting me and supporting me in my hard times, without his encouragement, nothing would have been easy. And to my entire family for their unconditional affection towards me.

AKNOWLEDGEMENTS

I would like to sincerely express my most profound gratitude towards my supervisor Dr. Sherif Saad Ahmed, whose input helped me immensely. With his input, I was able to look at my research with a different perspective and a more critical eye.

Secondly, I would like to express my gratitude to my thesis committee members for their beneficial advices and suggestions for my thesis.

I would also like to thank my friend Saurav for always encouraging and supporting me.

I humbly extend my thanks to the School of Computer Science and all concerned people who helped me in this regard.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

TP	True Positive
FN	False Negative
FP	False Positive
TN	True Negative
TPR	True Positive Rate
FPR	False Positive Rate
FNR	False Negative Rate
TNR	True Negative Rate
FRR	False Rejection Rate
FNMR	False Non Match Rate
FAR	False Rejection Rate
FMR	False Match Rate
EER	Equal Error Rate
SVM	Support Vector Machine
DT	Decision Tree
KNN	K Nearest Neighbors
NB	Naive Bayes
LR	Logistic Regression
RF	Random Forest
MLP	Multi Layer Perceptron
LGB	Light Gradient Boosting Machines

NN	Neural Network
GA-KNN	Genetic Algorithm -K Nearest Neighbors
IForest	IsolationForest
One-Class SVM	One Class Support Vector Machine

CHAPTER 1

Introduction

1.1 Access Control System

Any hardware, software, or administrative policy or process that controls access to resources is called an access control system [27]. In other words, it is a security technique that controls who or what resources can be viewed or used in a computing environment. It is a fundamental concept of security that reduces the risk to a company or organization. Its main goal is to provide authorized access and to avoid unauthorized access to resources. It performs the following overall steps: 1.) Identify and authenticate users or other subjects attempting to access the resources 2.) Determine whether access is authorized 3.) Grant or restrict access based on the subject's identity 4.) Monitor and record access attempts

1.1.1 Authentication Methods

Authentication verifies the identity of the subject by comparing one or more factors with a database of valid identities [27]. There are three basic methods of authentication:

Type-1: Something you know. Examples: passwords, personal identification number(PIN) or passphrase

Type-2: Something you have. Examples: smart card, hardware token, memory card, universal serial bus drive(USB)

Type-3: It is a physical attribute of a person identified with different kinds of biometrics, which includes Something you are. Examples: fingerprints, iris patterns, face

patterns, etc. OR Something you do. Examples: signature, keystroke dynamics and voice.

1.2 Biometrics

Biometrics are physical or behavioural characteristics that can be used to identify a person electronically in order to provide access to systems, devices or data. Examples of biometrics include finger print, facial patterns, iris patterns, voice, typing cadence. These identifiers are unique to an individual and can be used in conjunction to ensure greater identification accuracy.

1.3 Machine Learning

It is a branch of artificial intelligence focused on the premise that, with minimal human input, systems can learn from data, recognize patterns, and make decisions. Fig. 1.3.1 depicts three significant parts of the machine learning system:

Model: the system that makes predictions or identifications.

Parameters/features: the signals or factors used by the model to form its decisions.

Learner: the system that adjusts the parameters and, in turn, the model by looking at differences in predictions versus actual outcome.

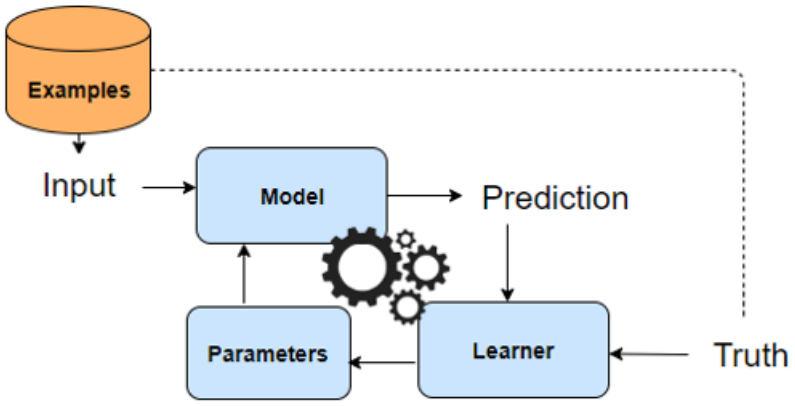

Fig. 1.3.1: Machine Learning Process

1.4 Motivation

The most important aspect of a biometric device is its accuracy. To use a biometric system for identification, a biometric device must be able to detect minute differences in the information. It is essential to learn the characteristics of the particular system. By doing that, one can get more understanding of the behavior of algorithms applied in such systems. This information helps to take appropriate actions when it is needed to maintain the system's reliability. In the keystroke systems, the user gets used to the typing device in a short time, so the typing patterns are likely to change over time. To adapt the changing typing behavior, a system has to have some policy to handle the situation. But what methodology a system should follow to address the condition? We are trying to answer this question through our research.

1.5 Problem Statement

Throughout the years, advancements in software and hardware technologies have reduced the costs of biometric authentication in addition to the advancement of computing resources, networking and database systems have made it easy to connect across a wide variety of geographic and networked areas. Thus, the acceptance of the biometric authentication systems has increased over time. The current state-of-the-art methods are used in various biometric systems to either identify or verify the user. The techniques include statistical learning, machine learning, anomaly detection, etc. Less emphasis is given on designing reliable behavioural-based biometric systems. In this aspect, the research focuses on providing a systematic approach to design and build reliable behavioural-based biometric systems. To implement proposed methodology experiments are performed using keystroke dynamics based access control system.

1.6 Thesis Contribution

In summary, we make the following contributions:

- We conducted a detailed study of behavioral biometrics' based access control systems which are using machine learning techniques. We studied how the previous researches have built different biometrics based on various machine learning algorithms and provided an in-depth analysis of how machine learning works in keystroke biometrics.

- To better understand the system's quality and performance, we have experimented using the widely used biometric zoo's [13] concept and provided our findings and observations on it.

- Further, we investigated anomaly detection techniques techniques and used two more datasets (from the android platform) to generalize our findings and also to look for platform-dependent variances in our experimental results.

- Our main objective here is to provide guidelines of how to design and build reliable keystroke dynamics based access control system. Furthermore, through this research, we tried to understand the behavior of keystroke biometrics over time. So, if the user's typing patterns gradually change with time, then how to update the user profile passively without altering user and maintain the system's reliability. All these research and findings led us to understand the steps that one can take to engineer building the reliable keystroke biometrics-based authentication system.

1.7 Thesis Organization

The rest of the thesis is organized as follows:

- In Chapter 2, we discuss how biometric zoos got proposed and how its evolution took place. Additionally, we present an analysis of the works that have been done using biometric zoo concepts for performance evaluation and enhancement of different biometric systems. A study of machine learning-based biometric systems, as well as anomaly detection based keystroke system, is also presented.

- In Chapter 3, we discuss the access control systems and authentication methods. The biometric authentication method with it's authentication process and types are described in detail. An introduction to the keystroke biometrics is also provided.

- In Chapter 4, we mentioned various machine learning techniques including multi-class classification and anomaly detection models. Additionally, evaluation metrics for the classification is discussed.

- In Chapter 4, we show the design and implementation of our approach to test the reliability of the keystroke biometrics based access control system.

- In Chapter 5, experiments and results of different machine learning-based biometric access control system is demonstrated using biometric zoo concepts to indicate the reliability of the existing techniques. In addition, experimental results of an anomaly detection based biometric authentication system with three datasets, one with a regular PC keyboard and two additional datasets from having touch screen keypads from the android platform, is also shown. In the end, the observations are drawn from the comparison and similarities of the different anomaly-based results.

- Finally, In Chapter 6 we conclude the thesis and discuss the potential future work

CHAPTER 2

Related Works

Biometrics is a widely used technique to authenticate the users using their physical or behavioral traits. There are many ways by which one can evaluate the biometrics system's performance. The most general evaluation metrics for biometrics are false rejection rate, false acceptance rate, equal error rate and failure to enroll rate which are described in detail in the section 3.3.2. Additionally, there is one more approach familiar to biometric researchers and practitioners to evaluate the biometric systems, that is the concept of biometric zoo.

2.1 Biometric Zoo

In a biometric authentication system, there are often the cases that some users are consistently performing poor. The researchers and integrators are interested in finding this type of poor performing user groups. Generally, majority of the users faces few issues like they are rarely falsely accepted or falsely rejected by the system. However, a small group of users may always behave in a way which increases system's verification errors. This type of users are naturally difficult to recognize and analysis of this kind of users and their common characteristics can expose the fundamental weaknesses of the biometric system. By considering these traits of the users one may be able to develop more robust authentication systems. Several problem groups have been characterized and given animal names which describe their behavior. The concept of biometric zoo is introduced by Doddington et al. [13] which is described in the following section.

2.1.1 Doddington's Biometric Zoo

In 1998, while performing statistical tests for the performance discrepancy in speech and speaker recognition systems, Doddington et al. [13] considered differences in distinguishability of individual types of users. In particular, the authors proposed a menagerie in which speaker differences were characterized using animal names, like Sheep, Goat, Lamb, and Wolf [13]. They considered two scores, i.e., genuine score and imposter score, to categorize the users in animal classes. **Genuine Score** is the set of scores in which user k is matched with user k's template. For example; when an apple has a match with an apple (True Positive). In other words, it derives number of times one can be easily matched with their samples. **Imposter Score** is the set of scores in which either any j user is matched with user k or k user is matched with any user j. For example, when an apple has a match with a peach or a peach has a match with an apple (False Positive). In other words, it derives the number of times when someone can successfully pretend to be someone else. The members of the Doddington's biometric zoo are described below:

- **Sheep:**

  Sheep users have very high genuine scores meaning these are the kind of the users who can easily access their accounts. So for them, the system performs quite well. When these types of users provide their biometric for a match, the sample matches nicely with saved samples of themselves and poorly with other user's saved samples.

- **Goat:**

  Goat users have low genuine scores meaning these are the kind of the users who are unable to access their accounts. Users categorized as goats are very hard to recognize by the system. When they provide their biometric for a match, they cannot match with one of their saved samples and as a result, the system rejects the user. They are the ones who are responsible for the majority of the system's False Rejection Rate (FRR).

- **Lamb:**

Lamb users have high imposter scores. They are easy to mimic, meaning any random user can get into the system pretending to be a lamb. When lambs biometric is combined with a different person's biometric, the resulting match score will be higher than average. The majority of the system's False Acceptance Rate is due to the lambs.

- **Wolf:**

  Wolf users also have high imposter scores. They are experts in impersonation, meaning they can get into other users' accounts by imitating their patterns. When these users provide their biometric for a match, they have a higher chance of getting matched with a different person's stored biometric sample. It is said that the wolves prey upon the lambs because, by definition, lambs are easy to imitate.

2.1.2 Yager and Dunstone's Biometric Zoo

Yager and Dunstone introduced four additional animals: Dove, Phantom, Chameleon, and Worm in 2007 [53, 54]. It is based on the user's relationship between their genuine and imposter match scores. Following are the definitions for the animals introduced in this category:

- **Dove:**

  These users are the best possible users for a biometric system. The users have high genuine score and low imposter scores. Their samples match well with other saved samples of themselves and poorly against others. Doves are rarely involved in any type of verification error.

- **Phantom:**

  User categorized as Phantoms have low genuine and imposter scores. It is implicit that they generate lower match scores regardless of who they are being matched against. They can be the cause of False rejects but are unlikely to be involved in False Accepts.

- **Chameleon:**

  Chameleons always appear similar to others receiving high match scores for all verifications. The users have high genuine and imposter scores. In other words, they receive high match scores for all verifications, both genuine and imposter. They are likely to cause False accepts.

- **Worm:**

  Worms are the worst types of users that a system can have. They fall in the range of highest imposter score and lowest genuine score. They are lowly creatures, having few distinguishing characteristics, and hence match poorly against themselves also they can be parasitic, leading to high match scores when matched against others. They are said to be the cause of a disproportionate number of system errors.

2.1.3 Doddington's Biometric Zoo Vs. Yager and Dunstone's Biometric Zoo

Doddington's zoo is based on the user's average genuine or imposter match scores, on the other hand, Yager and Dunstone's zoo considers the relationship between genuine and imposter match scores.

Goats are the users who are difficult to match and have lower genuine scores. In here, Doddington has not considered the associated imposter score for each user. shown on Fig. 2.1.1 [53], if the imposter score is also low, then the user falls into the 'Phantom' category, but if the imposter score is high, then one can fall in the 'Worms' category.

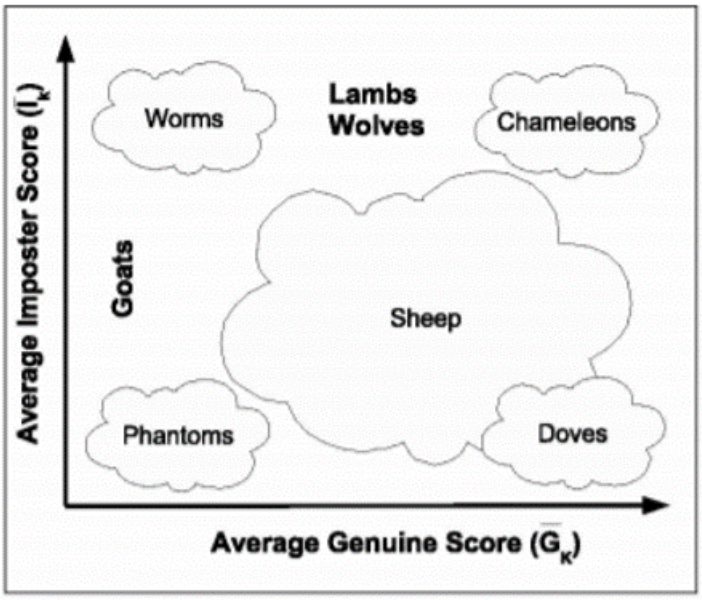

Fig. 2.1.1: Relationships between genuine and imposter match scores and the resulting animal

Lambs tend to produce high match scores when being matched against by other users. Wolves are the users who get high scores when matching against others. In both cases, imposter match scores are high, so, as per the fig. 2.1.1, if we consider high genuine score, then the user falls in 'Chameleons' category, and if the genuine score is low, then one falls in the 'Worms' category.

We can say that the Yager and Dunstone's zoo is making the Doddington's zoo complete by introducing the concept of correlation between genuine and imposter match scores.

2.1.4 Significance Of The Biometric Zoo

The biometric performance evaluation metrics like false rejection rate, false acceptance rate or equal error rate provides overall idea of how a system is performing while, to get an insight of system's issues the various categories of biometric menagerie helps. The three metrics average out individuals and problems associated with the subgroups of populations. The biometric menagerie serves as a diagnostic tool that takes a more user-centric approach. It helps in finding the users who are affecting

the system's performance and also what kind of behaviors do they exhibit. The three animal categories Goats, Lambs or Wolves makes this task easy for any researcher or any biometric authentication device development companies to know their device's characteristics.

Biometric zoos can be used to improve the design quality of biometric based authentication systems. By looking at each animal groups' characteristics, it can be said that if one apply the biometric zoo concepts on any biometric authentication system and find out the possible goats/lambs and wolves categories then they can take actions to handle and reduce those kind of user effects on the system. Additionally, it can also help one to develop techniques to convert poor performing users like goat, lamb or wolf into the sheep or dove categories, which can help to enhance the reliability of the biometric system. Also, just to check the performance of the system one can check for number of goats/lambs and wolves, the lower the value the better the performance of the system.

2.2 Biometric Zoo Evolution

The works under this category show that the biometric zoo does exist in every biometric technology and the authors try to understand why the zoo exists.

Teli et al. [49] proposed a theoretical framework to generalize biometric zoos across algorithms and datasets by experimenting on the human face recognition dataset. They tried to demonstrate the tests which show the existence of different levels of biometric zoos like zeroth level, first level, second-level, and third-level zoos. Experiments were carried out on two face recognition algorithms using two previously proposed approaches by Doddington et al. [13] and Yager and Dunstone et al. [53, 54], respectively. X^2 test was performed to check the existence of different animals. Results concluded that zoos of order greater than one are rare, or it is non-existent.

Bodorin et al. [37] claimed that the biometric menagerie previously proposed by Doddington et al. [13] and Yager et al. [53, 54] is fuzzy and inconsistent for the iris recognition system, whether it refers to the user or its biometric templates.

It was tested using 12 iris recognition tests on the iris image dataset. All tests were performed using the second version of the Circular Fuzzy Iris Segmentation procedure. The results suggest that the Goat concept is the most consistent, while the wolf is not a fuzzy concept. It also presents that biometric menagerie in any terms depends on the calibration of the iris recognition system.

Zheng et al. [16] believed that both the theories carried out by Doddington et al. [13] and Yager et al. [53, 54] disregarded threshold in biometric systems when classifying animals which intern might reduce the accuracy of the animal detection. To prove their belief, they experimented with both the concepts on a 100 % accurate finger vein dataset with 0.3 as a threshold using Kruskal Wallis Test to test Doddington Zoo's presence and to detect Yager zoo genuine and imposter scores were considered. Results showed the existence of both the menagerie animals (Goat, Lamb, wolf, Chameleon, Worm, and Phantom) in the dataset, which should not be the case. They proposed an optimized method Biometric Menagerie Detection with threshold (BMDT) based on Yager's theory and experimentally demonstrated that its accuracy is better.

2.3 Performance Evaluation and Enhancement Of Biometric Systems Using Doddington's Zoo

Authors under this category have used biometric zoo to check for the number of goats/lambs/wolves that are considered as the system's flaws. Some of them have utilized this knowledge to enhance their system's performance by proposing different techniques.

In the context of fingerprint and iris datasets, Ross et al. [42] proposed a selective fusion approach by getting the weakest users, which contributes to the majority of FAR (False Acceptance Rate) and FRR (False Rejection Rate) with the help of Doddington's zoo. Statistical framework based on the concept of percentiles of match scores and F-ratio was used to categorize the users. Only weak users were asked

for further information. Through this incremental method, they claimed that the system's overall matching accuracy increased and computational time got decreased. Jeffery et al. [33] classified subjects from the iris dataset as biometric zoo animals for several algorithms and studied the consistency of the classification algorithms. iris-BEE, MIRLIN and OSIRIS algorithms were compared and to present results ROC (receiver operating characteristic) curve depicting FAR and FRR was plotted. Their results showed that biometric menagerie classification is algorithm dependent and it also relies on which kind of iris is chosen(left/right). The authors also claimed that a person classified as weak should not be considered as weak because of algorithms' disagreement and the mismatched classification. Howard et al. [17] observed that biometric menagerie is based on the assumption that match scores are partially dependent on the specific subjects involved in the comparison operation. They claimed that the rate of identification is significantly affected by certain inherent properties of the subject, such as its ethnicity, gender and color of the eye, as well as the characteristics of the image, in particular the wavelength of light used by the sensor. To understand the iris recognition system's performance, a regression tree model was used to perform experiments.. Results demonstrated that only a single factor difference was found to cause a 2-3 times increase in the false rejection rate(number of goat users).

Ahmad et al. [45] analyzed the goat user within the population of an offline signature biometrics using HMM (Hidden Markov Model) based computational approach. They identified four goat populations on the basis of four local features (pixel density, center of gravity, angle, and distance) to interpret if they have any correlation. They tried to test whether different features influence the goat results, experiments demonstrated that they were highly correlated with each other. EER (Equal Error Rate) was considered to analyze their co-relationship. Sundararajan et al. [47] studied the challenges of a biometric system based on the writing style of individuals by investigating the system for the presence of goats, wolves, or lambs. The presence of the animals was verified using match score, FAR and FRR. They Suggested a method using person-specific characteristics referred to as "style signatures" To obtain Style

signatures for each person, they trained N One-vs-Rest (OVR) binary classifiers for N individuals. Experimental results showed that the use of person-specific Style signatures might be better at lowering false acceptance and false rejection rates, thereby addressing the goat/wolf/lamb behavior of individuals to a certain extent.

DeCann et al. [12] studied the impact of the biometric zoo on the relationship between the ROC curve and CMC (Cumulative Match Characteristic) curve created by the collection of genuine and impostor match scores. They designed a sampling procedure that reassigns the match scores to the animals of Doddington's zoo [13]. Experiments were performed using figure print and gait scores. ROC curves were plotted using false match rate (FMR) and the false non-match rate (FNMR) CMC curves were potted using the top 10 match scores, which exceeded threshold and system's identification accuracy for the same. Observations showed that several CMC curves could be linked to a single ROC curve. Kirchgasser et al. [21] used Doddington's zoo [13] concept to describe the Fingerprint template ageing influence. To investigate the same, they labelled data of users, including time separation of 4 years with different animal category names described by biometric menagerie concept. F-Test and Kruskal – Wallis tests were used to generate animal groups. The results demonstrated that regardless of which dataset and recognition systems were considered, the labelled users in the older datasets were not the same as in the new ones. They confirmed that fingerprint ageing could be the cause of the high amount of fluctuation in the detection results.

Neal et al. [31] explored the soft biometric classification of demographic and behavioral attributes using phone data collected from several subjects. Due to people exhibiting high intra-class variance templates and queries are affected in terms of how well they match. To analyze further in the matching error, biometric menagerie has been used. The similarity matrix was generated to evaluate animal results. Findings showed that many subjects are characterized as goats or wolves in their calling and SMS habits.

Poh et al. [36] claimed that Doddington's categorization [13] does not provide a criterion for ranking users in a database on the basis of their performance variability.

They proposed a user-dependent performance criterion that requires a limited number of genuine training scores. Three Log Likelihood Ratios were discussed: Z-norm, Z-shift and F-norm and developed a user specific score normalization scheme, which is a constrained F-norm ratio. Experimental results proved that user-dependent variability could be decreased by this scheme. Benchmark dataset XM2VTS containing match scores of 7 face systems and 6 voice systems were used. Schnitzer et al. [43] investigated the relationship of the animals of the Doddington Zoo [13] to the concentration of distances and the problem of hubness in a speaker verification system. Experiments have shown that, due to the high feature dimensions, goats and wolves are likely to emerge. For evaluation, GMM (Gaussian Mixture Model) trained on Mel frequency cepstrum coefficient (MFCC) is used. They claimed that hubness is an integral part of the development of the Doddington Zoo for speaker verification systems.

2.4 Doddington's Zoo Effects On Keystroke Dynamics

Wang et al. [51] presented frog boiling attacks that stealthily leverages the template update scheme of the keystroke verification system to poison user templates. The impact of the attack on the user groups identified by the biometric menagerie was investigated. They illustrated how the attack mutates the "Doddington Zoo [13]," as it turns historically well-performing animals (sheep) into ill-performing animals (lambs or goats) systematically. Attacks were performed on scaled manhattan verifier, selective fusion verifier and all fusion verifier.

Mhenni et al. [25] proposed method that used keystroke dynamics to help password-based applications to overcome hacking attacks. Users were classified into multiple categories according to the Doddington Zoo classification [13]. They applied an adaptive strategy specific to each category of users. Three different adaptive mechanisms were used: the growing window mechanism, the sliding window mechanism and the

least frequently used mechanism. An update strategy specific to the user class has been identified, which improved the obtained performances. Users with significant intra-class differences (Goats) have a greater comparison scale. Also, users who were more vulnerable to hacker attacks (lambs) were given higher decision thresholds. Additionally, Mhenni et al. [26] proposed a user-dependent adaptive strategy based on the Doddington zoo [13] as well as Yager and Dunstone's menagerie [53, 54], for the recognition of the user's keystroke dynamics. They applied an adaptive strategy specific to the characteristics of each user of both the menagerie aiming to solve the intra-class variation problems. Experiments were performed using the GA-KNN classification algorithm. An update strategy specific to the user class has been identified, which improved the obtained performances.

2.5 Machine Learning Based Biometric Systems

Under this category, we have reviewed the biometric systems that are using machine learning-based algorithms for either identification or verification task. Additionally, the evaluation metrics considered are also mentioned. Please note that none of the mentioned researches has considered quality score factor while taking its decisions. Quality score denotes the quality of the user's provided sample, in other words it derives how good is quality of the user's provided sample.

Boles et al. [9] used the Support Vector Machine algorithm for identification task in a voice biometric system . They considered accuracy for evaluation of the system. Marsico et al. [11] applied Support Vector Machine and various neural network algorithms like Wavelet Probabilistic NN, Back Propagation NN, Radial Basis Function NN, Restricted Boltzman Machine and Multi Layer Perceptron for identification of Iris biometrics. They have considered the threshold in the system and evaluated the system with accuracy, false acceptance rate(FAR), false rejection rate(FRR) and equal error rate(EER). For identification of brain EEG signals Bashar et al. [7] used machine learning algorithms like multiscale shape description (MSD), multiscale wavelet packet statistics (WPS), multiscale wavelet packet energy statistics (WPES)

for feature extraction and for matching purposes they have considered error-correcting output code multiclass model (ECOC) using SVM. They calculated accuracy to evaluate the system's performance. Sundararajan et al. [48] have presented a survey on deep learning based biometric systems. In the survey, they have mentioned various other works that have used deep learning algorithms like Convolutional Neural Network (CNN), Deep Belief Network (DBN), Deep Boltzman Machine (DBM), Convolutional DBN (CDBN), Restricted Boltzmann Machines (RBM), Recurrent Neural Networks, etc. for both identification and verification tasks for the biometric modalities like Face, Fingerprint, Palm Print, Iris, Voice, Signature, Gait and Keystroke. The works mentioned have considered accuracy, false acceptance rate (FAR), false rejection rate (FRR), mean absolute error (MAE), equal error rate (EER) or their combination for the assessment of the system.

Alghamdi et al. [3] used Medians Vector Proximity (MVP), K-Nearest Neighbor (KNN) and Random Forest Classifier for identification of smartphone user's gestures. They have considered the threshold in the system and evaluated the system on the basis of equal error rate(EER) and classification time. Bo et al. [8] applied two class SVM model for identification of touch and movement based biometric. They have used threshold in the system and considered accuracy, false acceptance rate and false rejection rate for system evaluation.

For identification in keystroke biometrics, Krishnamoorthy et al. [22] have applied SVM-RBF with one vs. one decision shape function. They have considered the threshold in the system and evaluated the system using F1-Score and accuracy. Ramu et al. [40] used the Gaussian probability density function and SVM with linear kernel for identification of keystroke biometric. They have used threshold in the system and considered accuracy, false acceptance rate and false rejection rate for system evaluation. In the context of tap and Keystroke biometric, Miluzzo et al. [28] considered Ensemble Classification Technique by using K-Nearest Neighbor (KNN), Random Forest Classifier, Multinomial Logistic Regression, Support Vector Machine and Bagged Decision Trees for identification task. They used accuracy to assess the system's performance.

2.6 Use Of Anomaly Detection In Keystroke Biometrics

Kevin Killourhy and Roy Maxion [19] applied various anomaly detection algorithms on the benchmark keystroke dataset and examined which are the top-performing algorithms for the keystroke biometrics. The main goal of their research was to collect a data set, establish an evaluation procedure and equally evaluate the performance of several anomaly detection algorithms like; Manhattan (scaled), Nearest Neighbor (Mahalanobis), Outlier Count (z-score), SVM (one-class), Mahalanobis, Mahalanobis (normed), Manhattan (filter), Manhattan, Neural Network (auto-assoc), Euclidean, Euclidean (normed), Fuzzy Logic, k Means, Neural Network (standard). In the process, they identified which detectors have the lowest error rate on their collected dataset (e.g., the Nearest Neighbor (Mahalanobis) detector) and they provided a data set and evaluation methodology that can be used by the community to evaluate new detectors and report comparative results. Furthermore, Kevin Killourhy and Roy Maxion [20] tried to find out the factors which affect the error rates of the anomaly detectors in the context of keystrokes dynamics. The factors that they considered for testing are the algorithm itself, amount of training, choice of features, use of updating, impostor practice, and typist-to-typist variation. They also experimented to know the approach that can be used to assess the effects of the factors on the anomaly detectors. In their approach, they experimented using a benchmark dataset, done statistical analysis using linear mixed analysis models and validated the model's predictions using new data. Their results showed that all the factors had a major influence on error rates except impostor practice and feature set.

Ivannikova et al. [18] proposed two approaches for detecting anomalies in the CMU dataset [19]. Dependence Clustering based approach and a k-NN-based approach that demonstrated strong results. Some of the current methods do use real data from users for training and validation. They designed a cross-validation procedure with artificially generated impostor samples that improve the learning process and allows for a fair comparison with previous works. They adapted a spectral clustering style

algorithm previously used only for clustering problems for the anomaly detection task. Experimental results demonstrated that both proposed approaches outperformed the previous state-of-the-art results for the CMU dataset for unsupervised learning.

John V. Monaco [30] described fifteen anomaly detection systems submitted to the Keystroke Biometrics Ongoing Competition (KBOC). The competition presented a task to identify anomaly with a public keystroke dataset containing over 300 subjects typed case-insensitive repetitions of their first and last name, and as a result, keystroke sequences could vary in length and order depending on the usage of modifier keys. Participants had the task of designing biometric keystroke verification systems that achieved a low cross over rate on a set of unlabeled query samples. To counter this, a preprocessing algorithm for keystroke alignment was developed in order to obtain a semantic correspondence between keystrokes in inconsistent sequences.

Mudhafar M. Al-Jarrah [2] presented an anomaly detector for keystroke dynamics authentication, based on a medians vector proximity method, validated by an empirical analysis of an independent keystroke data benchmark. A password typing-rhythm classifier is introduced, which can be used as an anomaly detector in genuine and impostor users' authentication.

2.7 Gap Between The Production Based And Research Based Approaches

In our opinion, there are two cases:

1. While doing their experiments with the benchmark keystroke dataset Killourhy and Maxion [19] have considered half of the samples from the total available samples to train the 14 different anomaly detectors and provided their findings based on its results. But in reality, that is not a feasible task. One can not ask the user to enter the password for say 200 times or 300 times. It will take long time and meanwhile, the typing behavior of the user may change or the user might get frustrated. In other kinds of literature, as mentioned in the category

"Machine Learning Based Biometric Systems" in the section 2.5, they are using various multi-class classification techniques, which is fine, but again majority of these researches are using a large number of data for training. Also, in the machine learning classification for the unseen sample, it is easy to identify a user because the classifier has a tendency always to find an identity, it will never say no.

2. The best results that the authors are getting in [19] are mostly the distance-based anomaly detectors. But in the keystroke dynamics, user typing patterns change over time, so the distance may not remain the same all the time. On the other hand, If we consider the fingerprint detection system, then the distance-based detectors work well because the distances almost remain the same, so the system accepts the user as a genuine user for a longer duration. So, it is not clear from the works of literature on how keystroke biometrics impacts on the accuracy of the system over time.

3. The majority of the works in the literature are using same password to record users' keystroke samples. However, in the keystroke dynamics, if the user changes his password then the typing pattern completely changes. In this case the feature selection will not work. Because, in reality, different users will have different passwords with different length. One would not be able to build the classifiers using the feature selection when the samples don't share the same feature dimensions.

CHAPTER 3

Access Control Systems

In this chapter, we start by describing what is access control systems and what are the various authentication methodologies used in those systems. The chapter also talks about biometrics in detail including it's authentication process and general evaluation metrics. Furthermore, a brief introduction of physical and behavioral biometrics is provided. Additionally, keystroke dynamics a type of behavioral biometrics is described in detail with digraph representation.

3.1 Access Control System

Access control is a selective restriction to the access of the data. It performs identification, authentication and authorization of users and entities by evaluating required login credentials. Where, identification is the process of subject claiming an identity. A subject is required to provide its identity to proceed further for authentication and authorization operations. Authentication is a method used to determine whether someone is what they appear to be. It verifies the identity of the subject by comparing one or more factors with a database of valid identities [27]. Authorization indicates who is trusted to perform certain actions. If the operation is permitted the subject is authorized otherwise it is not authorized. The access controls can be implemented in three ways: administratively, logically/technically, or physically [27].

- **Administrative Access Controls:** Administrative access controls are the policies and procedures established by the safety policy and other regulations or specifications of an organization. Examples of administrative access controls

include policies, procedures, hiring practices, background checks, classifying and labeling data, security awareness and training efforts, reports and reviews, personnel controls, and testing [27].

- **Logical/Technical Controls:** Logical access controls are the tools used to monitor access and to secure systems and resources through hardware or software. Examples of logical or technical access controls include authentication methods (such as passwords, smartcards, and biometrics), encryption, constrained interfaces, access control lists, protocols, firewalls, routers, intrusion detection systems, and clipping levels [27].

- **Physical Controls:** Physical access controls involve specific measures used to prevent, monitor or identify external interaction with devices or locations inside the facility. Examples of physical access controls include guards, fences, motion detectors, locked doors, laptop locks, badges, swipe cards, video cameras and alarms [27].

3.2 Authentication Factors/Methods

To verify the user's identity authentication system considers some factors or methods. There are three factors or methods for authentication:

- **Knowledge Based Factor (Type-1):** It is depended on what the user knows. Examples of knowledge based or type-1 factors include passwords, personal identification number(PIN) or passphrase.

- **Possession-based factors (Type-2):** It is based on what a user possess(has). Illustrations of knowledge based or type-2 factors include smart card, hardware token, memory card, universal serial bus drive(USB)

- **Inherence-based factors (Type-3):** This is defined according to what the user is, or how he does. It is a physical attribute of a person identified with different kinds of biometrics [27] Examples in what user is includes fingerprints,

iris patterns and face patterns. Examples in how he does category include signature and keystroke dynamics

All these factors become stronger over time if they are implemented correctly. Authentication can be performed using the combination of one or more above mentioned factors. Based on number of factors considered authentication can be divided into three categories:

- **Single Factor Authentication:** As the name suggests it uses only one factor to authenticate a user trying to get access into the system. It is more vulnerable to attacks.
- **Two Factor Authentication:** It combines any two factors to increase the system security. For example, there are many banking applications which requires users to enter their password as well as a one time password(OTP) sent by the bank to the individual's device to authenticate the user.
- **Multi Factor Authentication:** It uses more than one authentication factors to generate a layered structure of authentication. In simple words, It requires the user to enter two or more credentials to login into the system. It not only increases the security but generates a reliable false proof system.

3.3 Biometrics

It refers to any automatically measurable physiological or behavioral traits which are distinctive to an individual. Physiological characteristics are related to the shape of the body (Something you are). As shown on Fig. 3.3.1 [55], examples of physiological characteristics include finger print, face, hand, iris, finger vein. Behavioral characteristics are related to the pattern of behavior of a person (Something you do). As shown on Fig. 3.3.1 [55], examples of behavioral characteristics include voice, keystroke, signature. By using unique biological characteristics, biometrics is the most suitable means of identifying and authenticating individuals in a reliable and fast way.

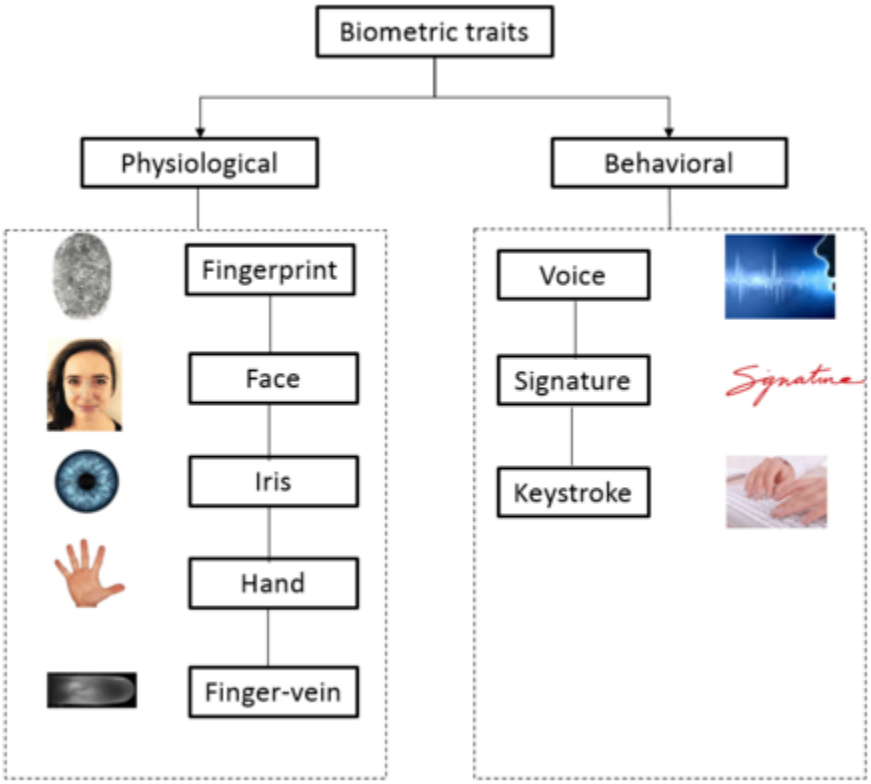

Fig. 3.3.1: Biometrics

With the biometric technology there is nothing to lose or forget since the characteristics or traits of the person serve as the identifiers [55]. Also, many of these identifiers remain intact for longer time. The authentication factors like passwords and PINs can be stolen easily. Biometrics should reduce the risk of compromise the likelihood that an adversary can present a suitable identifier and gain unauthorized access [55].

According to the requirements, biometrics are utilized for either of the two purposes: identification or verification. In identification, the biometric system asks and tries to answer the question "Who is X?" In this process, the biometric device reads and compares the samples against each record or template in the database. This type of comparison is referred to as one to many (1:n) search. Verification is when the systems ask and try to answer the question, "Is this X?" After the user claims the identity of X. In a verification procedure, the biometric device needs input from the

user, at which time the user asserts his or her identity through a password, token, or user name (or any combination of the three). The user input points the system to a template in the database. It then processes and compares the sample with or against a user-defined template. This kind of search is called one to one (1:1). In this case, the system either finds a match or fails to find a match.

3.3.1 Biometric Authentication Process

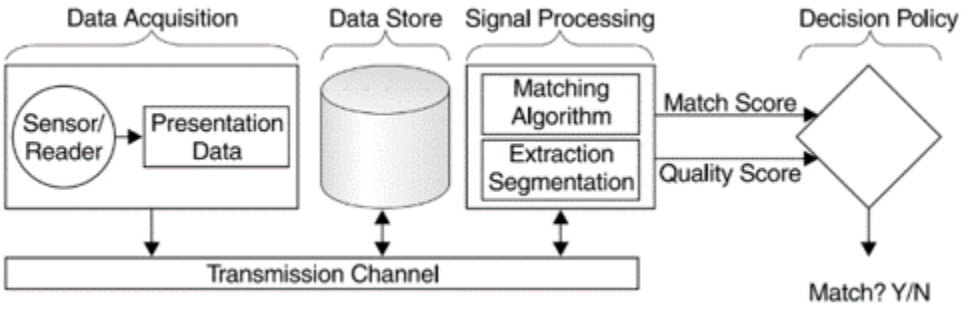

Fig. 3.3.2: Biometric Authentication Process

As shown on Fig. 3.3.2 [32], the biometric authentication process starts with the "Data Acquisition", where the user provides there biometric sample through any biometric sensor. The sample is acquired and forwarded to the "Signal Processing" unit for matching purposes through the transmission channel. When the sample arrives at the signal processing unit, segmentation is performed, and any unwanted data(noise) is removed from the sample. Next, the segmented sample is provided to the feature extractor unit that extracts essential features out of the sample and generates a template for matching. The output of the "Extraction and Segmentation" unit is "Quality Score" which scores the quality of the user-provided sample. The matching algorithm considers template generated through "Extraction and Segmentation" and based on the application, it matches the template with one or more reference templates and produces a matching score that describes how well a template matches the reference template(s). Both the scores are considered by "Decision Policy" of the system to decide whether there is a match (Yes?) or not (No?). Generally, a predetermined

threshold for both scores are considered. If both scores are above that threshold, it is said to be a match (Yes). Otherwise, if the quality score is above threshold and match score is below the threshold, then the system rejects the user and if the match score is above the threshold and quality score is below the threshold, then the system asks the user to provide the biometric sample again.

3.3.2 Evaluation Metrics

False Rejection Rate (FRR) / False Non-Match Rate (FNMR):
It describes the number of times someone who should be identified positively is instead rejected.

$$FRR = \frac{FR}{N} \times 100 \tag{1}$$

where FR = Number of incidents of False Rejections
N = Total number of samples

False Acceptance Rate (FAR) / False Match Rate (FMR):
Describes the number of times someone is inaccurately positively matched

$$FAR = \frac{FA}{N} \times 100 \tag{2}$$

where FA = Number of incidents of False Acceptance
N = Total number of samples

Equal Error Rate (EER) / Crossover Rate:
Combination of FAR and FRR helps to understand the usefulness of a particular biometric device in the given scenario. Equal Error rate or Cross Over Rate is the intersection of false rejection rate and false acceptance rate (Fig. 3.3.3 [32]). The lower the rate the better the biometric system.

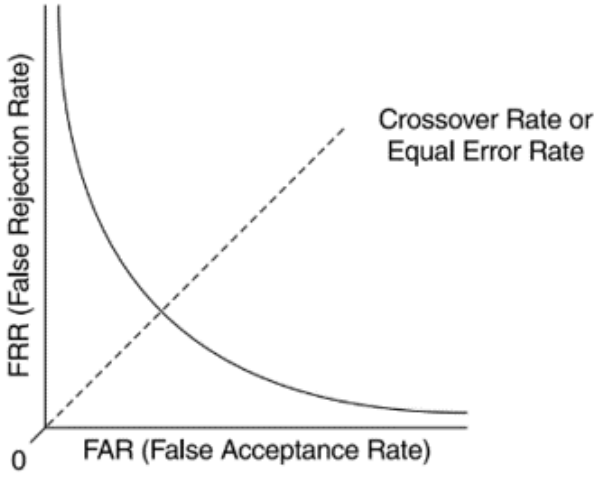

Fig. 3.3.3: Equal Error Rate or Crossover Rate

Failure To Enroll Rate (FER) It is the percentage of population that are unable to enroll into the system.

3.3.3 Physical Biometrics

Physiological characteristics such as fingerprints, iris, finger vein patterns and face geometry play a vital role in user verification to verify users belonging to a large population. It is very convenient way of presenting identity because different shape of body presents the identity so there are no risks that a user tend to forget. Also, it almost remains the same for several years so there is no need to update it now and then. It is stable and reliable as well as easy to use and setup. Though these structures make each individual body unique, they are static, which leaves them more vulnerable to being scanned or photographed, then reconstructed for malicious use [35].

3.3.4 Behavioral Biometrics

Behavioral biometrics is used to uniquely identify a user through their particular behavioral pattern or actions. The benefit of using behavioral biometrics over the physical biometrics is that it is stored in terms of numeric timing, position and statistic

data which is not a physical representation like a shape of a finger print or a face. So, even if the information is stolen the attacker is unable to interpret the data and regenerate certain behavior. Also, unlike physical biometrics the stored data keeps evolving and changes over time which means even if the behavior biometrics is stolen after some time it is of no use because it keeps on changing. Whereas, physical biometrics like finger print once stolen then it can be used for multiple purposes for longer duration. Every person behaves in a completely individual way. The gait with which someone walks, the fluctuations in vocal tone as they speak, and the cadence with which they type are as unique as fingerprints but are much harder for malicious actors to capture, much less duplicate. Behavioral biometrics uses these patterns to authenticate users and protect data [35]. Behavioral biometrics includes gait, voice patterns, keystroke dynamics, touch screen swipes/ mobile interactions and cursor movements.

3.3.5 Keystroke Dynamics

It is a type of behavioral biometrics that measures how a subject uses a keyboard based on the timing and latency between a key press and key release event on a keyboard. Software is used to capture it, so the technique can be applied to any system that accepts and processes keyboard input events [32]. It can be used for single authentication events or continuous monitoring. For example; It can be used to harden the passwords, which means that the keystroke dynamics can be deployed for each user to augment the existing password by requiring that the password should be entered in a manner consistent with the intended user [32].

It is a strong authentication procedure which involves typing a password and the way of typing it. This double-layer defence provides better protections against all threats on the internet, such as brute force attack, dictionary attack, and actual shoulder surfing. The brute-force attack includes a hacker trying out all possible combinations of passwords, and is easier to interpret if the intruder knows what we know. The dictionary attack is a type of brute force attack which consists of trying every word of the dictionary as a password which works in most of the cases.

In the shoulder surfing the attacker observes the password while one is typing it. The keystroke dynamics serves as the best solution for all these attacks, because it combines the biometric pattern with the password. By doing so, it becomes hard for an attacker to impersonate means that it is tough to regenerate if observed or even if one knows the correct password. One's typing pattern is a unique behavioral characteristic of an individual. Additionally, keystroke dynamics has a resettable signature which means that if you change the text, then the typing behavior also changes. So, if an intruder somehow get the typing traces for a particular password typed by a user then the legitimate user can reset the typing traces by changing the password.

3.3.5.1 Digraph Representation

In reality, timing traces used for pattern matching are commonly represented as a set of digraphs. A digraph is an adjacent pair of characters in typing sequence and associated timing delay between the pressing of the first key and the pressing of the second key [32]. Fig. 3.3.5 [32] shows an example digraph representation for typing no.

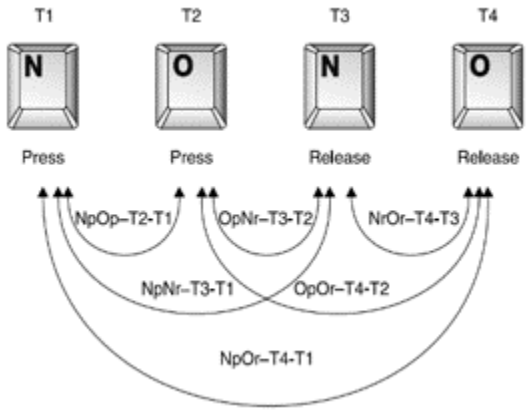

Fig. 3.3.4: Digraph representation for typing no

The data is kept as a table of time stamps containing key-down and key-up events. The resulting features are expressed as dwell time (time that key is depressed) and flight time (latency between key down events) for the various digraphs. Moreover,

some character sequences and words can be represented as trigraphs or tetragraphs (for instance, ing and tion) or word graphs.

CHAPTER 4

Machine Learning Techniques

The chapter introduces machine learning techniques. Two types of classifications: multi-class classification and one class classification (anomaly detection) are discussed in detail. The multi-class classification working is explained and eight different types of classification algorithms are described in depth. Also, anomaly detection's working is explained and four kinds of anomaly detectors are discussed. In the end, the evaluation metrics for the classification methods are explained.

4.1 Machine Learning

Machine learning is an application of artificial intelligence (AI) that provides systems the ability to automatically learn and improve from experience without being explicitly programmed [14]. It is generally categorized as supervised or unsupervised.

Supervised algorithms for machine learning are developed to learn from labelled data. The term "supervised" comes from the idea that training this kind of algorithm is like having an instructor monitor the entire operation. The training data for the supervised algorithm's training consist of inputs with the correct outputs. During the training, the algorithm looks for the patterns in the training data, which corresponds with the correct output. Once the training is complete, the algorithm can take unseen data as inputs and determine the correct label for it based on the data it has seen during the training phase. Supervised learning aims to predict the correct label for the unseen sample. The supervised learning can be divided into two sub-types: classification and regression. In this thesis, we are focusing on supervised machine

learning methods, specifically classification for user identification.

Unsupervised learning is where you simply have input data (X) with no associated output variables. The aim of unsupervised learning is to simulate the underlying structure or distribution of data to get more knowledge about the data. It is considered unsupervised, since unlike supervised learning above there are no correct labels associated with the data. The system doesn't work out the right output, but examines the details and can draw inferences from datasets to explain hidden constructs from unlabeled input. Unsupervised learning is further divided into two subcategories: clustering and association.

4.2 Multi-Class Classification

Classification is the method of determining the class of given data points. Often the classes are called targets/labels or categories. Classification predictive modeling is a process in which input variables (X) and it's related independent output variables (Y) are used in an algorithm to learn the mapping function from input to output $Y = f(X)$. The goal is to estimate the mapping function so efficiently that the output variables (Y) can be predicted for every new input variable (X). The process of learning the mapping function is called training and the process of getting the output variable for new(unseen) input variable is called testing.

Fig. 4.2.1 illustrates how the classification works to identify whether the user is a genuine user or an imposter user. To train the model combination of both the types of users with their correct labels, genuine/imposter is used. Once a predictive model is ready, a random unknown user is fed into the predictive model to get its label(class) as either genuine or imposter user. It is an example of a two-class classification wherein there are only two classes (genuine / imposter). There can be more than two classes; those types of classification techniques are called multi-class classification techniques. Mostly, the systems have more than two users, so generally, the multi-class classification is deployed to identify the users.

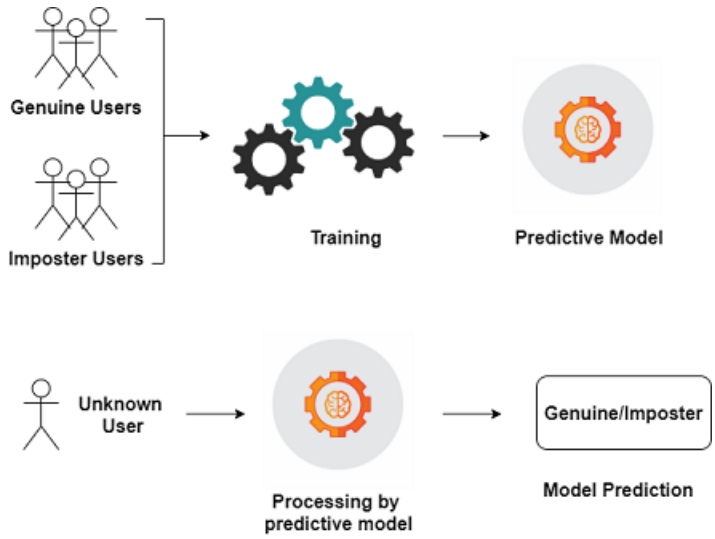

Fig. 4.2.1: Classifying users into genuine or imposter user category

4.2.1 Support Vector Machine (SVM)

The objective of the support vector machine algorithm is to find a hyperplane in N-dimensional space (N — the number of features) that distinctly classify the data points [41]. As shown on Fig. 4.2.2, generally, the model finds a group of hyperplanes and based on the margin value, it selects an optimal hyperplane that has maximum margin value, i.e., the maximum possible distance between two classes' data points. Support vectors are data points that are closer to the hyperplane and influence the position and orientation of the hyperplane [41]. The hyperplanes act as decision boundaries. Points falling on either side of the plane can be given different classes accordingly. The dimensions of hyperplanes changes with the number of features. So, for example, for 2 input features the hyper plane is just a line and the hyper plane for 3 features is a two-dimensional plane.

To measure the similarity between the data points, a kernel function is used. The function is data dependent and hence it can be selected according to the problem at hand. If the data points are not linearly separable then the kernel function maps them into higher dimensions to make it linearly separable and predicts their target class.

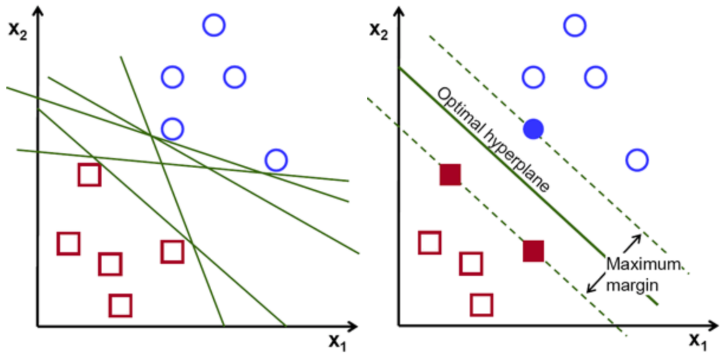

Fig. 4.2.2: Support Vector Machine: (left) Possible Hyper planes (right) model selected optimal hyper plane with maximum margin

4.2.2 Decision Tree

The goal of the decision tree is to create a model that predicts the value of a target variable by learning simple decision rules inferred from the data features [44]. A tree can be "learned" by splitting the source set into subsets based on a test of the value of the attribute. This procedure is repeated recursively, called recursive partitioning on each derived subset. The recursion is completed when the subset at a node all has the same target variable value, or when splitting does not add value to the predictions anymore. Fig. 4.2.3 [56] describes an example of a binary tree for predicting a person's fitness. It predicts the label from fit/unfit based on the parameters like age, eating habits and exercise habits.

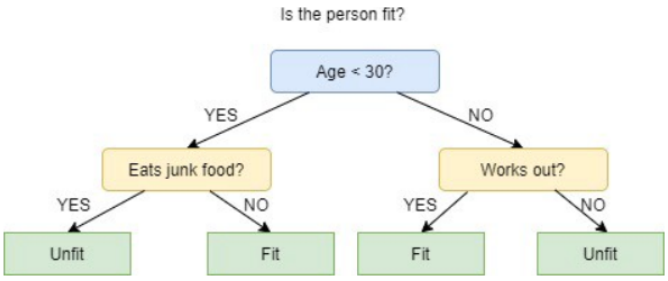

Fig. 4.2.3: Decision Tree example

At each new node of the tree, the algorithm specifies new rules which led it to the final target class label.

4.2.3 K Nearest Neighbor (KNN)

KNN is a lazy learning algorithm since it does not have a dedicated training process and uses all the data during classification for training. Additionally, It is a non-parametric learning algorithm because it assumes nothing about the underlying data. KNN operates by finding the distances between a query and all the examples in the data, choosing the K number of examples listed nearest to the query, then votes for the most common label. For instance, we have a dataset having 2 classes in it (red/blue) and its plotting look like the figure 4.2.5 [50].

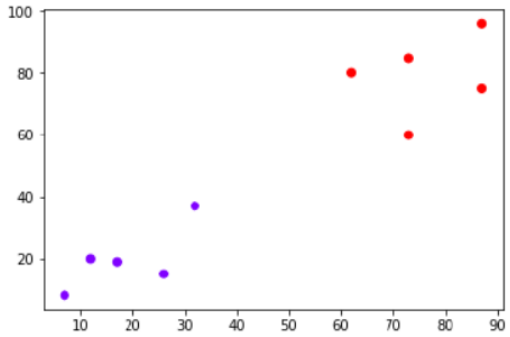

Fig. 4.2.4: KNN training set plotting

Suppose now a new data point comes in as the one in black colour. Let's take K=3. So, here the K nearest neighbor will find the 3 most nearest data points to the new data point, As shown on the figure below.

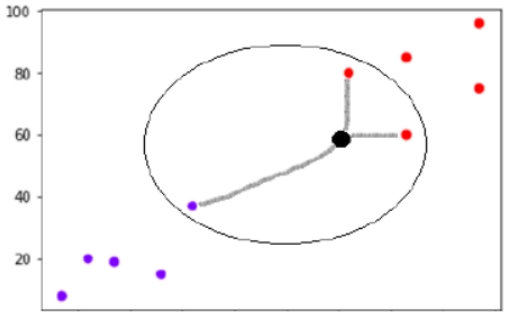

Fig. 4.2.5: KNN find K nearest neighbors

Fig. 4.2.5 [50] demonstrates that the two most nearest data points are from the

red class and one point belongs to the black class. So, by majority voting, the black coloured data point will be assigned the label of the red class.

4.2.4 Naïve Bayes

Naive Bayes is based on the Bayes theorem, which assumes that each feature of the feature set is independent of each other. The key purpose of the Bayesian classification is to determine the posterior probabilities, i.e. the likelihood of a category given certain observable characteristics. It estimates the probabilities of membership for each class, such as the likelihood that a given record or data point belongs to a specific class. The class with the highest likelihood is considered to be the most likely class.

4.2.5 Logistic Regression

It is an algorithm for predictive analysis and based on the principle of probability. Logistic regression uses a cost function named 'Sigmoid function' or also known as 'Logistic function'. The hypothesis of logistic regression tends to limit the cost function between 0 and 1 [6].

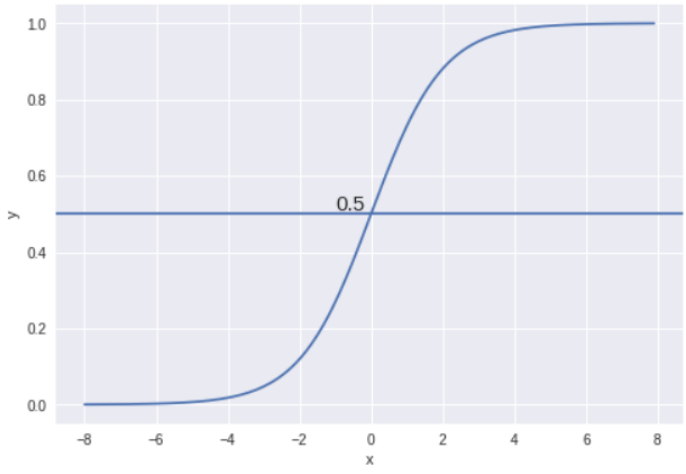

Fig. 4.2.6: Logistic Regression Example

The classifier is expected to provide the classes or labels when we provide the input features through a prediction function and it gives the probability scores in between

0 and 1. As shown on Fig. 4.2.6 [6] for example, we have two classes genuine user and imposter user and we keep a threshold of 0.5 to decide which class a sample belongs. If the score values go below the threshold, then it will be assigned an imposter class and if it is above the threshold, then it is genuine. Now, suppose the classifier outputs a probability score of 0.7, then the sample will be labelled as genuine.

4.2.6 Random Forest

Random forests or random decision forests are an ensemble learning method for classification. Random forest, as the name suggests, is made up of a large number of individual decision trees that act as an ensemble. Every single tree in the random forest provides class predictions and in the end, a final class is decided through majority voting (Fig.4.2.7 [1]).

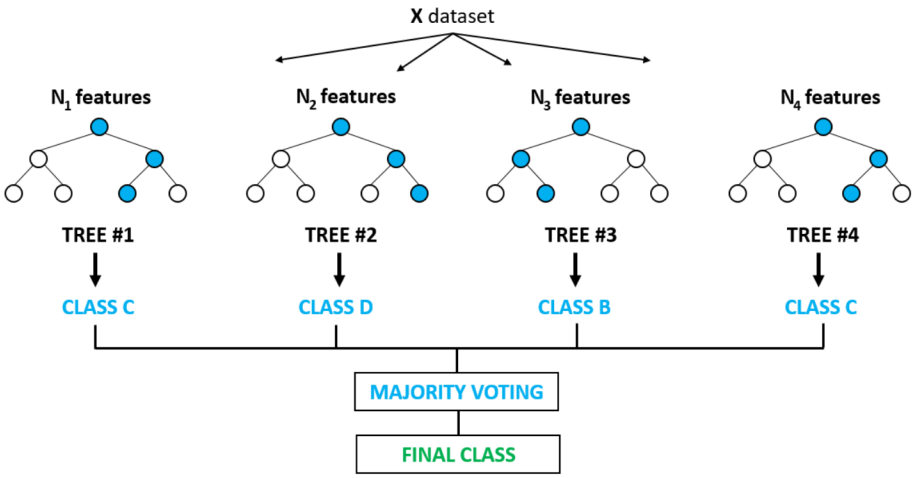

Fig. 4.2.7: Random Forest Example

The benefit of using the random forest over a decision tree is that each tree provides the prediction, which helps to correct the error in any other individual tree's class prediction.

4.2.7 Multi Layer Perceptron (MLP)

It is a deep artificial neural network algorithm. Fig.4.2.8 [29] shows the structure of MLP. The network consists of an input layer which contains a set of neurons representing the input features $\{x_1, x_2, ..., x_n\}$, an output layer which makes the predictions and in between these two layers there can be N number of hidden layers which performs the intermediate computations. The model is trained using a set of input-output variables from which the model learns the association or dependence between the input and the target variables. The training requires tuning the parameters like weights and biases of the model to eliminate the error.

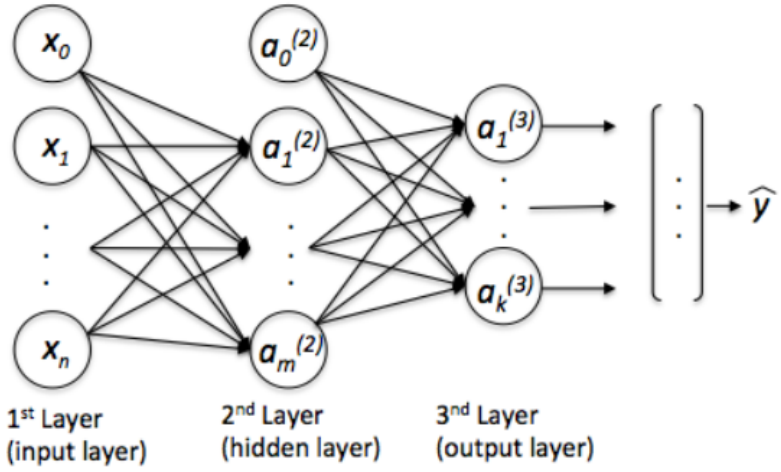

Fig. 4.2.8: Structure of Multi Layer Perceptron

4.2.8 Light Gradient Boosting Machines (LightGBM)

Light GBM is a gradient boosting framework that uses a tree-based learning algorithm. What makes it different from other tree-based algorithms is the way it expands. It grows vertically instead of horizontally (Fig.4.2.9 [39]), which means that it expands leaf-wise and not level-wise. It always chooses a leaf with a maximum delta loss to expand, which helps it to reduce more loss than any other level-wise algorithm.

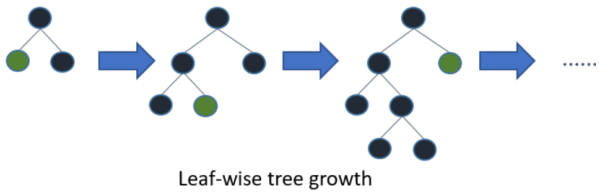

Fig. 4.2.9: How a Light GBM works

Fig. 4.2.10: How other boosting algorithm works

Light GBM has gained popularity because it is a high-speed algorithm and it can handle a large amount of data with fewer memory requirements. Also, it majorly focuses on the accuracy of the results.

4.3 Anomaly Detection (One-Class Classification)

Anomaly detection is the process of finding data objects with behaviors that are very different from expectation. Such objects are called outliers or anomalies. Many systems need the ability to determine whether a new observation belongs to the same distribution as existing observations (it is an inlier), or should be considered to be different (it is an outlier).

There are three broad categories of anomaly detection techniques. Unsupervised anomaly detection techniques detect anomalies in an unlabeled test data set on the assumption that most instances in the data set are normal by searching for instances that appear to fit the least to the rest of the data set. Supervised anomaly detection techniques require a collection of data that has been labelled “normal” and “anomalous” and requires training a classifier. Semi-supervised anomaly detection

techniques create a model that reflects normal behavior from a given standard training data set and then evaluates the probability of a test instance being created by the learned model. Among all the unsupervised anomaly detection is the most preferred approach because, in reality, we don't have a data set that is explicitly labelled as normal or anomalous.

4.3.1 Working Of An Anomaly Detector

Every model, in some way, scores a data point than uses threshold value to determine whether the point is an outlier or not. According to the data given as an input to the anomaly detector, it decides the threshold value for considering a point to be an inlier or outlier. For illustration, let us understand it using the concept of the imposter and genuine user. Fig. 4.3.2 shows the training of an anomaly detector for a single genuine user. The detector is trained using a part of the genuine user's data. The detector decides the threshold and learns a decision boundary according to the threshold value.

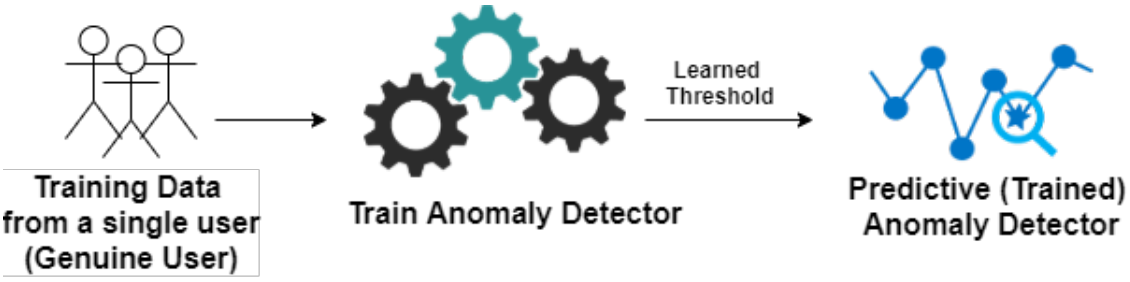

Fig. 4.3.1: Training of an anomaly detector

The trained anomaly detector is tested using the remaining part of the genuine user's data (Fig. 4.3.3). The anomaly detector outputs the information of inliers, outliers and the anomaly score. As the training and testing set are both from the same user, the anomaly score is called genuine score.

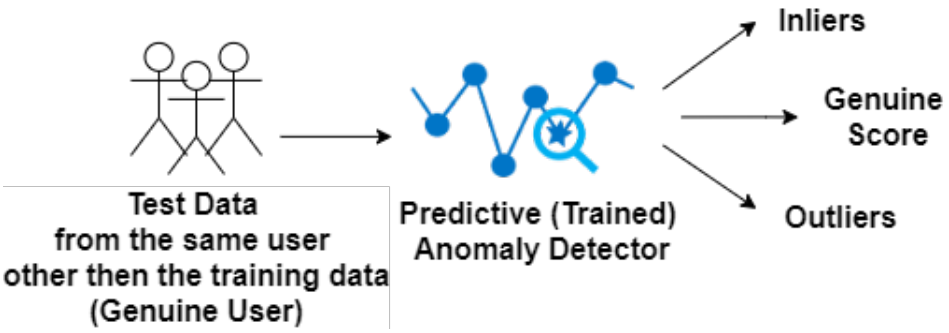

Fig. 4.3.2: Anomaly Detection for Genuine User

The trained anomaly detector is tested using a few samples of the imposter user (Fig. 4.3.4). The anomaly detector outputs the information of inliers, outliers and the anomaly score for the imposter user. As the training and testing set are from a different user, the anomaly score is called imposter score.

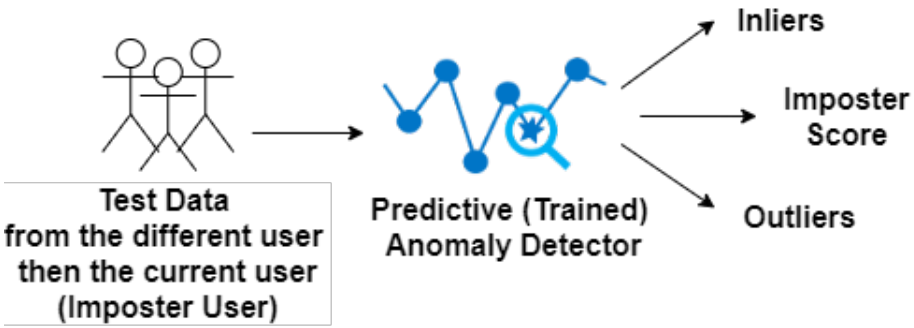

Fig. 4.3.3: Anomaly Detection for Imposter User

The trained anomaly detector outputs anomaly scores as well as inliers and outliers. So, how an anomaly detector decides a point to be an inlier or outlier? Fig 4.3.5 demonstrates how a trained anomaly detector makes its predictions. Firstly, the anomaly score is calculated for a data point; then, it is provided to the decision function. The decision function compares the anomaly score with the predefined threshold value. If the score is equal to or higher than the threshold value, then the data point is categorized as an outlier; otherwise, it is considered as an inlier.

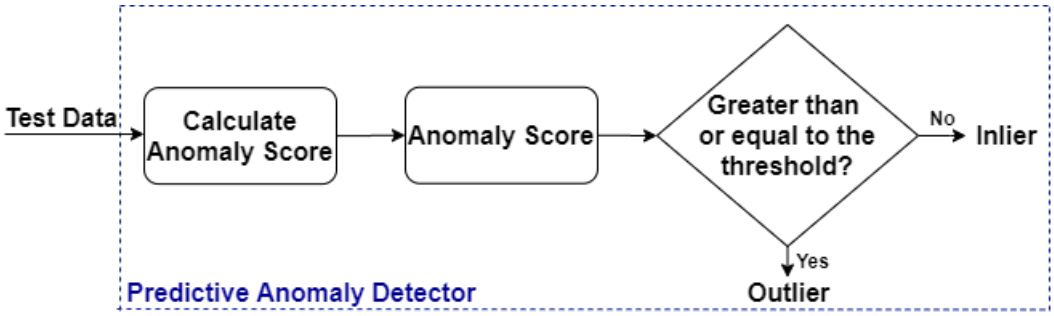

Fig. 4.3.4: How a trained anomaly detector works (internally)

4.3.2 K Nearest Neighbor (KNN)

KNN anomaly detector saves a list of training vectors during the training and learns the covariance matrix. During testing, it calculates Mahalanobis distance between the training and test vectors. Mahalanobis distance is the distance between a point and a distribution. The distance provides a way to measure how similar a data point is to a known set of data points. The distance of the test vector to its kth nearest neighbor is considered to calculate the anomaly scores.

Average - K Nearest Neighbor (AVG-KNN)

Average KNN is a variant of KNN detector with a minor change at the time of its application. While calculating the anomaly score, the mean(average) of all k neighbors are considered.

4.3.3 IsolationForest (IForest)

The IsolationForest 'isolates' observations by randomly selecting a feature and then randomly selecting a split value between the maximum and minimum values of the selected feature [34]. The algorithm partitions the data into a set of trees. Anomaly score is provided by looking at how isolated the point is in the tree structure [24].

4.3.4 One-Class Support Vector Machine (One-Class SVM)

It is the one-class variant of the standard Support Vector Machine (SVM) developed explicitly for anomaly detection. A data point from a single class is projected in a high dimensional space. A separator is found between the origin and the projection. Following the same process during the training, it builds the model using the training data vectors. The test vectors are also projected in the same space. The distance between the test vector and the partition is calculated as the anomaly score.

4.4 Evaluation Metrics

Any classification's performance is generally evaluated based on metrics named the confusion matrix (Fig. 4.4.1 [38], Fig. 4.4.2 [23]). It is plotted using actual values against the predicted values. In the case of two-class classification, one class is considered as positive and another one is considered as negative. For example, we have two classes genuine user (+ve) and imposter user (-ve). based on the confusion matrix, some parameters are deduced as follows:

- **True Positive**: When positive class is predicted as positive. For example, when a genuine user is predicted as a genuine user.
- **False Negative**: When positive class is predicted as negative. For example, when a genuine user is predicted as an imposter user.
- **False Positive**: When negative class is predicted as positive. For example, when an imposter user is predicted as a genuine user.
- **True Negative**: When negative class is predicted as negative. For example, when an imposter user is predicted as an imposter user.

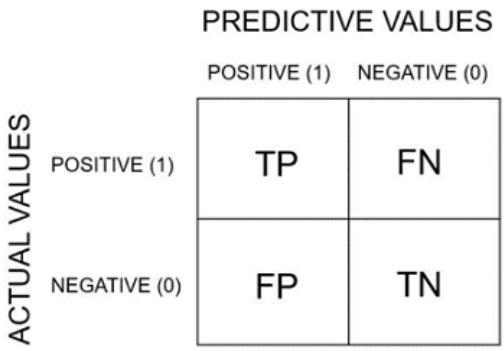

Fig. 4.4.1: Two Class Confusion Matrix

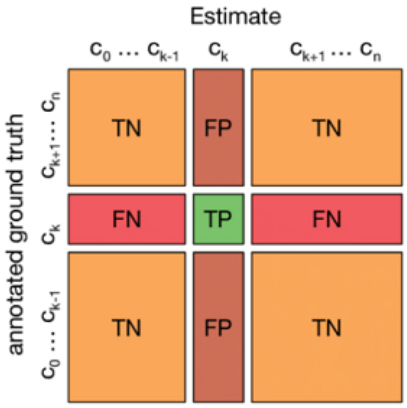

Fig. 4.4.2: Multi class Confusion Matrix

Based on the above parameters, some metrics are generated:

- **False Positive Rate (FPR)**: Proportion of negative cases classified as positive cases.

$$FPR = \frac{FP}{FP + TN} \times 100 \tag{1}$$

- **False Negative Rate (FNR)**: Proportion of positive cases classified as negative cases

$$FNR = \frac{FN}{FN + TP} \times 100 \tag{2}$$

- **True Positive Rate (TPR)**: Proportion of positive cases classified as positive

cases.

$$TPR = \frac{TP}{FN + TP} \times 100 \tag{3}$$

- **True Negative Rate (TNR)**: Proportion of negative cases classified as negative cases

$$FPR = \frac{TN}{FP + TN} \times 100 \tag{4}$$

- **Accuracy**: The fraction of predictions classification model got right.

$$Accuracy = \frac{Numberofcorrectpredictions(TP + TN)}{Totalnumberofpredictions(TP + FN + TN + FP)} \times 100 \tag{5}$$

CHAPTER 5

Methodology

The previous chapters have introduced the access control systems, keystroke biometrics and various machine learning techniques. This chapter walks you through various methods used to build a keystroke-based access control system. Firstly, we show how multi-class machine learning algorithms can be implemented as a keystroke based access control system and how it is evaluated using the widely known biometric menagerie concepts. Secondly, the procedure to use the anomaly detection technique in the keystroke based authentication system is introduced. This chapter also includes the process of doing the feature selection on the available feature set. An insight into the various research based approaches of machine learning and anomaly detection is provided. The differences between the research based and production based approaches are also described. In the keystroke systems, the user gets accustomed to the typing device in a short time consequently, their typing patterns change with time. If the system continues to rely only on the previously trained user profiles then the chances of user rejections gets increased over time. To lay emphasis on the problem, two approaches of how a keystroke biometrics-based authentication system can be updated with the time is provided.

Fig.5.0.1 demonstrates the keystroke authentication process. The procedure starts with a keystroke sensor like a normal computer keyboard, touch screen keypad or from any device which can process keystroke. The user provides its keystroke sample by typing a password, passphrase or a small paragraph. The keystroke sample is captured in the form of digraph or trigraph. The sample is next forwarded to the feature extractor unit which processes the raw samples and prepares the template for match-

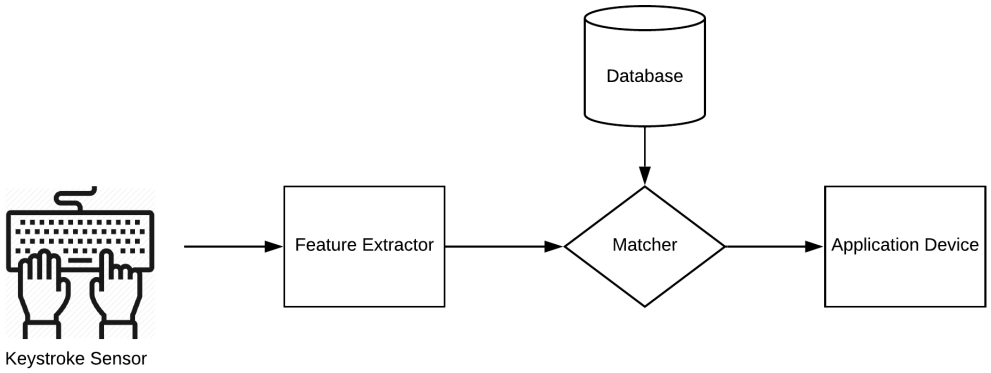

Fig. 5.0.1: Working of keystroke authentication system

ing process. It also generates the quality score which indicates how good is the quality of the provided sample. In the matching phase the system considers the template and according to the requirement (i.e, verification or identification) it tries to match the given template with one or more available templates and generates a matching score. The template matching unit is generally a trained machine learning or statistical model which does the pattern matching process and generates the matching score. Matching score denotes how well a template matches with the reference template. In the end, the quality score and match score are considered for deciding the match. According to the threshold value considered by the system, a decision for accepting or rejecting a user is taken. If the match score and quality score are above the threshold value the user is accepted otherwise the system may reject the user or simply asks the user to reenter the sample.

The work in this thesis focuses on the template matching part of the authentication process. It is one of the most important and talked about aspects of the biometrics. It is the critical part of any biometric authentication system's decision policy. We have considered conventional machine learning as well as anomaly detection techniques for our research. As observed from the fig 5.0.1 every unit contributes in the matching process. So, the matching process gets affected by variances introduced by individual units.

5.1 Multi-class Machine Learning Methods

In this section, we demonstrate how multi-class machine learning methods perform in a keystroke based biometrics system to identify users. In the beginning, we discuss some of the machine learning approaches implemented in the literature. To understand the behavior of a machine learning based keystroke access control system, we have experimented on a benchmark keystroke dataset [19]. In our work, we have done classification using eight different classification models. Various researches in the fields of biometrics have used the concept of biometric menagerie to get better understanding of the system's performance. The most commonly used approach is Doddington's Biometric Zoo. So, we implemented the Doddington's Biometric Zoo concepts in our classification experiments to get an insight into the system's behavior.

5.1.1 Methodologies Used In The Literature

In the literature 'Machine learning based soft biometrics for enhanced keystroke recognition system' [40] Ramu et al. have considered the use of Support Vector Machine (SVM) for classification. They have performed 5-fold cross validation to evaluate the soft biometric accuracy. The data was partitioned into 5 subsets. For each experiment run, 4 subsets are used for training the classifier and the remaining 1 subset is used for testing. The biometric recognition accuracy was calculated by taking an average of the accuracies of all the runs.

Margit Antal and Lehel Nemes in [5] have implemented two class classification techniques like KNN, Bayes Net and Random Forest on their collected android keystroke dataset. Firstly, they select genuine and imposter samples from their dataset. To create the negative samples set they have selected random 2 samples from each other user (users other than the current user). Then they perform N-runs of the randomization followed by N-Fold cross validation for the given user data and all these steps were repeated for all the users of the dataset. In other words, if the value of N is 10 then for each run of cross validation process 90% of the user's data is used for the training and rest 10% data is considered for testing. To evaluate the classification based on

the users' score they select a threshold and calculate False Positive and False Negative rate values and derive the Equal Error Rate (EER) value through their intersection.

To perform keystroke identification, Antal et al. in [4] have considered classification methods like Naive Bayes, K Nearest Neighbors (KNN), Support Vector Machines (SVM), Decision Trees and Multi Layer Peceptron (MLP). They have experimented on an android keystroke data set. For comparison purposes, they performed classification for keystroke data with and without touch screen based features. They have experimented by executing 10 runs of 10-folds cross validation on entire dataset. Consequently, 90% data is considered for training the model and rest is used for the testing purpose. They have reported the accuracy values based on the average of the 10 10-fold cross validation accuracies.

5.1.2 Data Preprocessing

Data preprocessing is a procedure to convert the data into the format suitable for a machine learning model. Cleaning and putting the data in a formatted way is essential. As a part of data preprocessing, we verified our data for missing values, class imbalances and duplicates. Finally, we removed some features from the dataset, which are qualitative and not crucial for the classification task. Additionally, before doing classification, we found the best hyper parameters for the classifier by tuning the parameters for the classifier using the 'Grid Search' method.

5.1.2.1 Grid Search Method

We applied the grid search method for parameter tuning using the function 'GridSearchCV' of the sklearn's library model_selection. The method finds the best parameter set for the given development set. We started by providing three parameters to the 'GridSearchCV' method: classification model, parameter set for tuning and scoring method. Next, we fit the model for the training set and training label to get the best parameters. We perform the grid search to get the best parameter set for both 'precision' and 'recall' scores. **Precision** expresses the proportion of the data points

our model says was relevant actually were relevant [$Precision = \frac{TP}{FP+TP}$] [52]. **Recall** expresses the ability to find all relevant instances in a dataset [$Recall = \frac{TP}{FN+TP}$] [52]. At the end of each loop iteration, the algorithm outputs the best parameter set found for the particular scoring method selected during the iteration.

The Methodology is divided into two parts, in the first part we discussed the methodology to categorize individual users into sheep, goat or lamb animal class. In the second part we describe the methodology for finding the system generated errors.

5.1.3 Part-I: Find Sheep, Goat, Lamb

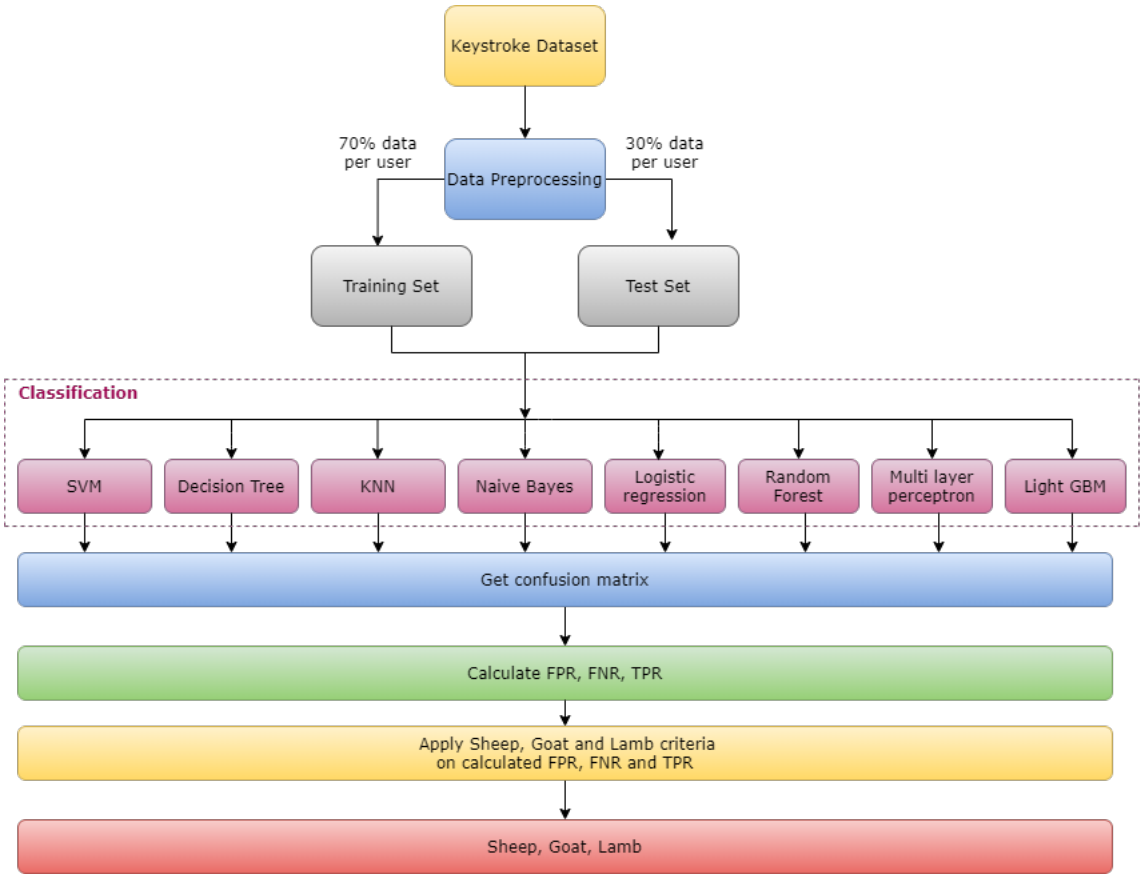

Fig. 5.1.1: Working of part-I: find sheep, goat, lamb

Fig. 5.1.1 shows the overall flow for part-I, we first divide the preprocessed feature set into two parts: Training set consisting of 70% data per user and testing set has

remaining 30% data per user. We forward this data to the classifiers for classification. As a result of classification, we get a confusion matrix from which we calculate FPR, FNR and TPR and apply the constraints of sheep, goat and lamb.

5.1.4 Part-II: System Generated Errors

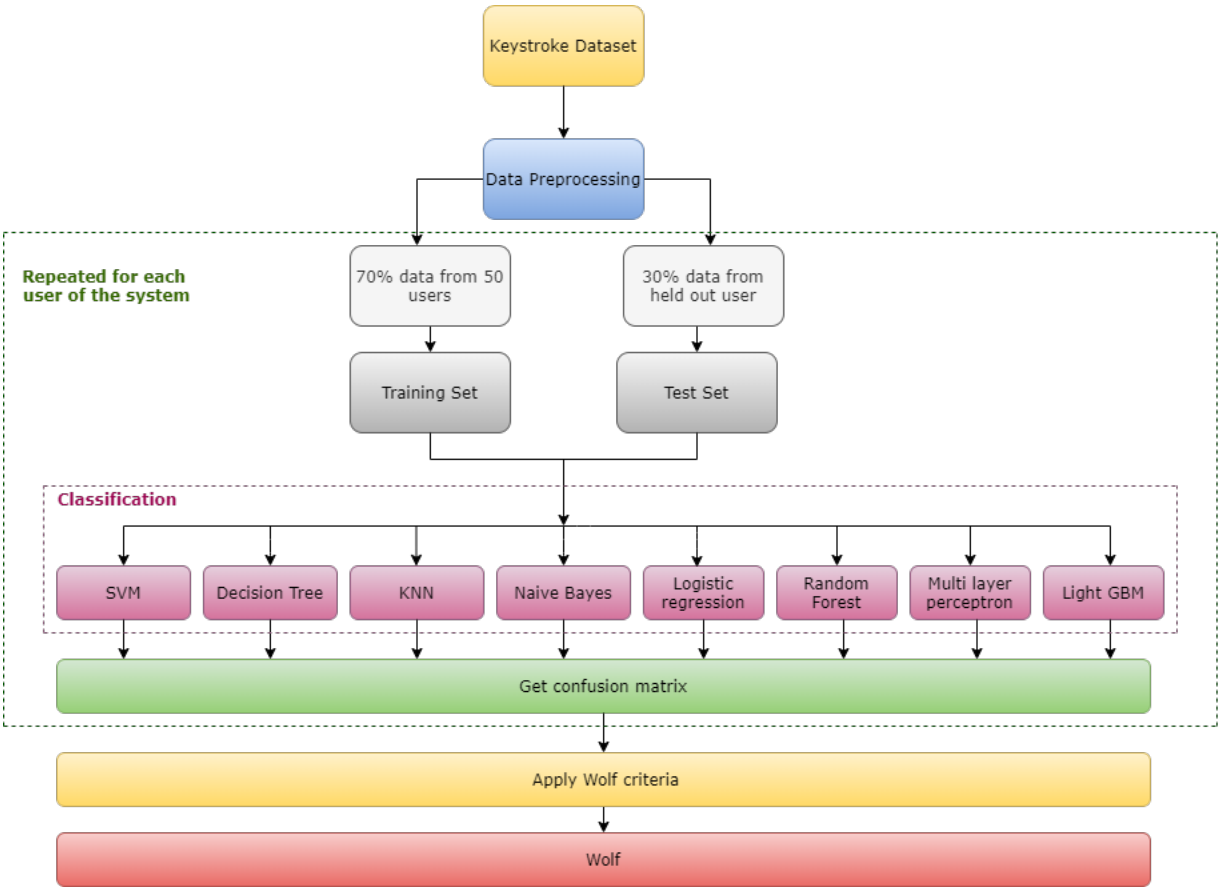

Fig. 5.1.2: Working of part-II: find system generated errors

In this approach, we held one user out for testing and rest are used for the training. That is, in each iteration, one out of N users is used as a test set. For training, 70% data from the N-1 users and for testing 30% data from the held-out user is considered for each iteration. The output will be plotted in terms of the confusion matrix in order to visualize the distribution of the test user's samples in other classes. The reason for doing this is to address our need to understand system's response towards

a sample unavailable in it's training. We decide the wolf by verifying that in how many classes the test user is present, in other words, in how many classes the test user's samples are getting distributed. The higher the number of classes, the higher the chances of the test user for being a wolf. We believe that the machine learning based keystroke access control systems will not be able to provide correct predictions when it encounters unknown samples. It will categorize the given unknown sample as one of the available categories.

5.2 Anomaly Detection

This section presents the design and implementation of the anomaly detection for keystroke biometrics based access control system. Firstly, it discusses about the already implemented works in the literature and their approach to implement anomaly detection in keystroke biometrics. It also demonstrates the use of feature selection and normalization. Furthermore, it states differences in the practical and research-based approaches in addition to introducing practical methodologies to implement in the system with a limited number of user samples. We claim that the keystroke biometrics has a tendency to change over time and hence the users' profiles must be updated periodically. We describe the methodologies for with and without updating the users' profiles. To generalize our findings and also to look for platform-dependent variances in our experimental results we have performed anomaly detection on the three datasets proposed in [19], [4] and [5] respectively. The first dataset was collected through a normal PC keyboard; the other two are android device based keystroke datasets. In the methodology the experiments are described for four different anomaly detectors namely, one-class support vector machine, isolation forest, k-nearest neighbors and average k-nearest neighbors.

5.2.1 Methodologies Used In The Literature

Killourhy and Maxion, who proposed the benchmark keystroke dataset [19], have considered 50% data from the total data to train the model. They started by desig-

nating one of the 51 subjects as the genuine user, and the rest as impostors. They run the training phase of the detector using the timing information of the first 50% (200) password repetitions typed by the genuine user. Once the detector builds a model of the user's typing behavior, they ran the test phase of the detector using the remaining 50% (200) repetitions typed by the genuine user and recorded the anomaly scores as user scores. Finally, they run the test phase of the detector on the timing vectors from the first five repetitions typed by each of the 50 impostors that are in total 250 test samples and again recorded the anomaly scores as impostor scores. The process was then repeated, designating each of the other subjects as the genuine user in turn. They performed the same process for all the 14 detectors.

Margit Antal and Lehel Nemes in [4] have used five detectors implemented in the R script provided by Killourhy and Maxion [19] to experiment on the android keystroke dataset. In their script, they split the data into three equal parts, each containing 20 samples from each user and there are 54 users in their dataset. So, each part contains 54 x 20 = 1080 samples in total. The detectors are trained separately for each user using two-third (66.67%) of the data (40 samples/user). The evaluation was performed on the remaining one-third positive (33.33 %) samples (20 samples/user) and two negative samples selected from each of the other users (106 samples); The previous step is repeated thrice (threefold cross-validation), and the mean EER and its standard deviation is computed.

A similar kind of methodology as [4] is followed by Antal et al. in [5] to experiment on an android dataset. They also used the R scripts provided by Killourhy and Maxion [19] to perform verification using the anomaly detectors. The data was divided into three parts, each part having 17 samples/user. So, in total, each part had (3 x 42 = 126 samples). Two thirds (66.67%) of the data was used for building the user's profile (training the anomaly detectors) and the remaining one third (33.33 %) of the data was utilized for testing FRR. The first five samples from each user excluding the current user (41 x 5 = 205 samples) were used to test the imposter rate (FAR).

5.2.2 Data Preprocessing

As a part of data preprocessing, we verified our data for missing values, class imbalances and duplicates. In the end, we balanced the classes in one of the dataset where required. Additionally, we removed some features from the dataset, which are qualitative and not important for the anomaly detection. We have also performed feature selection and normalization which is described in detail in the following sections.

5.2.3 Anomaly Detection With 70 - 30 Train/Test Ratio

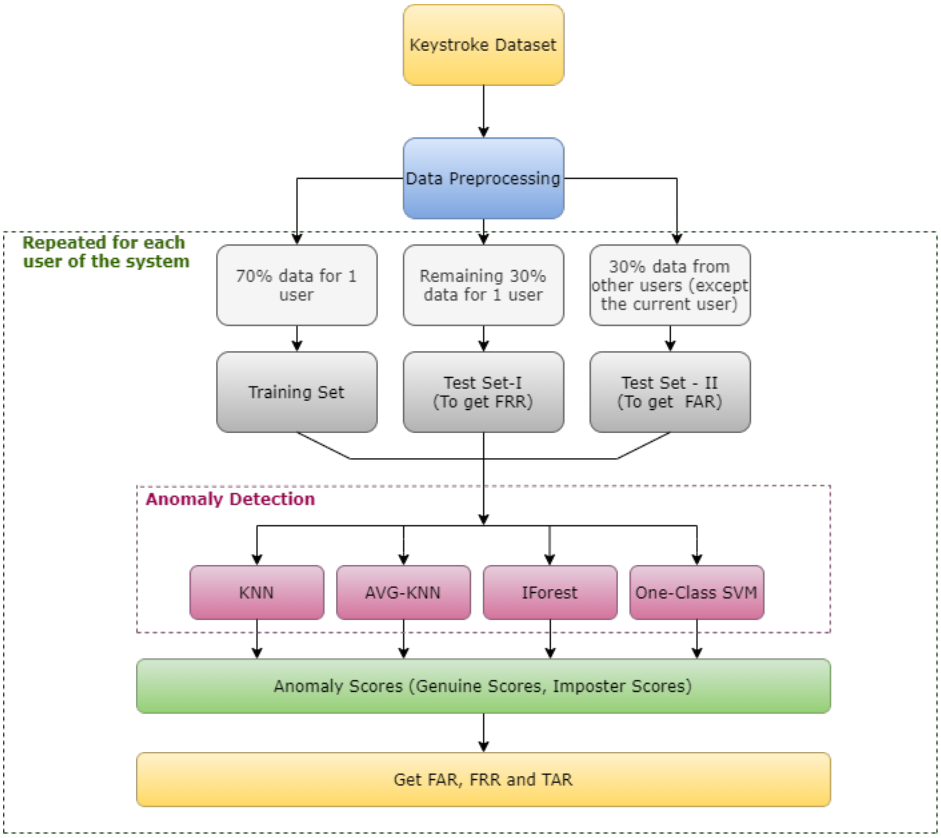

Fig. 5.2.1: Flowchart for Anomaly Detection with 70 - 30 ratio

Unlike the machine learning experiments' methodology, here, we divide the data into three parts. There are 2 test sets; one is to get the false rejection rate (FRR) and another one is to get the false acceptance rate (FAR). As the flow chart demonstrates, the anomaly detector is trained using 70% samples from one user. To get the false

rejection rate, 30% test data are considered from the same user and 30% of samples randomly from other users (except the current user) are used to get the false acceptance rate. The same process is repeated for all subjects' samples. We have followed this methodology to be consistent with our machine learning experiments and also to match with the literature based approaches. The approach will help us to compare our outcomes with the ones in the literature.

5.2.4 Effects Of Feature Selection And Normalization

To compare the effects of feature selection and normalization on anomaly detection's performance, previously we applied the anomaly detection without using the feature selection and normalization methods. To test the impact of feature selection and normalization on anomaly detector's performance, we have implemented six types of feature selection methods on the feature set: Pearson Correlation, Chi-Square, Recursive Feature Elimination, Lasso: SelectFromModel, Tree-based: SelectFromModel, LGB: SelectFromModel. The features which are not marked important ('True') by any of the feature selectors are removed from the feature set. Once the selected feature set is ready, normalization is applied using the 'PowerTransformer' using method 'yeo-johnson' of the sklearn's 'preprocessing' library. The same experiment for anomaly detection (70-30 train/test ratio) is performed again, but with the selected feature set and the observations are derived from the comparison of the anomaly detectors' performances in both the scenarios.

- **Pearson Correlation**

  We provide the feature vector with the correct feature label vector (target variables) and the number of features to select from the entire feature set as an input to the feature selector [34]. The method finds the correlation between each feature and the target variable and provides its results as select ('True') or not select ('False')

- **Chi-Square**

  We provide the scaled feature vector with the correct feature label vector (target

variables) and the number of features to select from the entire feature set as an input to the feature selector [34]. This method calculates the chi-square metric between the target and the feature variables and only select the variables with the maximum chi-squared values. The result is provided as select ('True') or not select ('False').

- **Recursive Feature Elimination**

  The recursive feature elimination (RFE) selects features by recursively considering smaller and smaller sets of features [34]. It first trains the initial feature set and obtains the importance of each feature through a coefficient attribute. For our experiment, we have used 'Logistic Regression', and the Recursive Feature Elimination observes the coefficient attribute of the 'Logistic Regression' object. The recursion continues until the method finds the best possible features for the dataset and outputs the result as either select ('True') or not select ('False') for a particular feature.

- **Lasso: SelectFromModel**

  Lasso is an embedded method that uses algorithms that have built-in feature selection methods [34]. To implement Lasso, we have used the 'Logistic Regression' as the selected model and L1 as the regularizer. The result is provided as select ('True') or not select ('False').

- **Tree-based: SelectFromModel**

  Like Lasso, this is also an embedded method that uses tree-based algorithms that have built-in feature selection methods [34]. To implement it, we have used 'Random Forest' as the selected model. The result is provided as select ('True') or not select ('False').

- **LGB: SelectFromModel**

  To apply boosting algorithms like Light GBM(LGB) as a feature selector, we have used 'LGB' as the selected model [34]. The result is provided as select ('True') or not select ('False').

5.2.5 Differences In Production Based And Research-based Approaches

As seen in the sections 5.1.1 and 5.2.1 "Methodologies used in the literature", it is evident that all the experiments and works in the literature are performed using a large amount of data which is not feasible in real world. One may not ask the user to enter the password say 200 times or 300 times. It will take long time and meanwhile, the typing behavior of the user may change, or the user might get frustrated.. In other kinds of literature, as in the Machine Learning based biometric systems, they are using various classification techniques, which is fine, but again majority of these researches are using a huge quantities of data for training.

It is impractical to obtain more than some 10 to 15 password samples from the users. Additionally, while we have limited samples (let's say 10 samples) for the user, then it is not feasible to apply any feature selection. Because, we think that, the feature selection results may vary when we do it on the entire dataset and on the small subset of the dataset.

5.2.6 Feature Selection With Less Data

To review the hypothesis that the feature selection will not be much effective in the case of fewer amount of data we derive a methodology which does the feature selection with the small subset of the dataset. In this method, we select 10 samples from each class (i.e, user/subject) of the dataset and create a subset to perform feature selection. For example, we have 10,000 samples and there are 20 classes in total then we take 20 X 10 = 200 samples and create a subset to perform feature selection on it. Next, we apply all six feature selectors on the derived subset and record the result to compare it with the results of feature selection when we considered the entire dataset. If the outcomes of the experiment demonstrate that there is a difference in the results of feature selection on entire dataset and the smaller subset then our hypothesis will be proved correct. Otherwise, if there is no effect on the results, in other words if the results of both the feature selections gives same feature set then our hypothesis will

be proved wrong. By following this methodology, we will be able to finalize whether one should prefer to use feature selection for smaller subsets of data or not.

5.2.7 Without Updating The User Profile

We believe that the user's keystroke patterns changes over time and hence, the profiles should be updated periodically. To test this hypothesis we followed the approach described in this section. Also, from this section we are considering the practical approach described in the previous section. We are going to consider a batch of 10 samples for training as well as testing purposes.

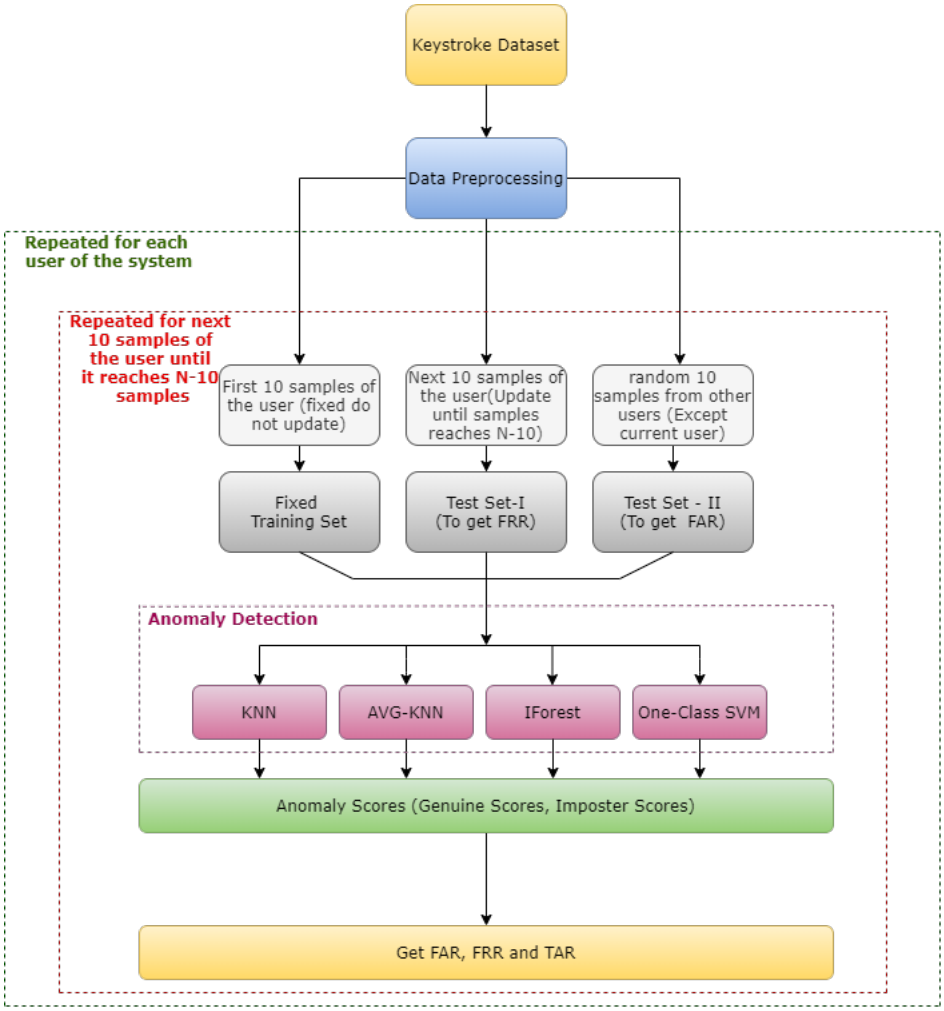

Fig. 5.2.2: Flowchart illustrating the methodology for user profile without update

We kept the training sample the same for all the user iterations (All the logins).

We trained the anomaly detectors with the first 10 samples fixed and tested using the next 10, 20, 30, ... , N-10 samples for the same user to get the false rejection rate. To get the false acceptance rate random 10 samples from other users (except the current user) are used. The same process is repeated for the particular user until the last sample of the user. The process is repeated for all the users of the dataset. If the output of the experiment suggest that the update is required and without the update the system will perform poorly in terms of rejection rates. We will be able to claim that the typing pattern do change over time. Otherwise, it doesn't change and hence we don't need any techniques to update the users' profiles.

5.2.8 Methods To Update User Profile

On the basis of our belief that the user profiles do need some kind of update regularly. We propose two adaptive methods to update the user profile systematically namely batch mode and sliding window approaches. Both the approaches are novel to the best of our knowledge.

5.2.8.1 Batch Mode Approach

Experiments are performed with the same anomaly detectors but by using a different approach. For any user, the first 10 samples are used for training and the next 10 samples are considered for testing (to get FRR). Likewise, for the second iteration, the second 10 samples (which were used for testing (to get FRR) in 1st iteration) is used for training and the next 10 samples from the current 10 samples are considered for testing (to get FRR). At the same time, 10 samples from random users (except the current user) are used to find false acceptance rate (FAR). The same process is repeated for the particular user until the last sample of the user. The process is repeated for all the users of the dataset.

5.2.8.2 Sliding Window Approach

In this approach we have a window that we move 1 sample per user's successful login. We consider the first 10 samples (1,2,....,10) for anomaly detector's training for a particular user. To test the detector for the false rejections we fed it the next sample (11) of the same user. Also at the same time to check for the false accepts we select random 10 samples from the other users except the current user. For the second iteration we remove the oldest sample (1) from the training set and append the training set with the next successful sample (11). Also we update the test set by moving the test window to the next sample i.e, 12. All these steps are repeated until N-10 number of samples of the current user. The entire process in repeated for all the users of the system.

Fig. 5.2.3: Working of the Sliding Window

We believe that this approach will be able to capture users' behavior better than the batch mode approach as we are moving one sample at a time. So, it helps train the

system with the user's most recent behaviors which in turn helps system to recognize the user in a consistent manner.

5.3 Summary

To summarize, we established the methodologies to use different multi-class classification and anomaly detection methods to build a keystroke-based access control system. Firstly, we showed how multi-class machine learning algorithms can be implemented as a keystroke based access control system and how it is evaluated using the widely known biometric menagerie concepts. Secondly, the procedure to use the anomaly detection technique in the keystroke based authentication system is introduced. The chapter also talked about the process of doing the feature selection and normalization on the given dataset. Additionally, it demonstrated the procedure to perform feature selection on the small subset of the data. The approaches used by the literature works for the multi-class as well as anomaly detection is discussed. Furthermore, the differences between the literature and the practical approaches are reviewed. In the end, two approaches of how a keystroke biometrics-based authentication system can be updated with the time are provided.

CHAPTER 6

Experiments and Results

In this chapter, firstly, we give brief introduction of the experimental environment and toolkits. We also discuss about the datasets we have used for the experiments. We perform analysis of the machine learning based access control system with the Doddington's biometric zoo. Additionally, we investigate the anomaly detection based approaches to know how it behaves in a literature based setting. We also demonstrate the effects of the feature selection and normalization by experimenting on the anomaly detection techniques. This analysis will give us an idea about how effective and efficient are the existing state-of-the art literature based access control systems. In addition, we show the implementation and evaluate the performance and efficiency of our proposed practical approaches.

6.1 Environment and Toolkits

We have used a machine having windows 10 installed in it. To perform our machine learning related experiments in python we installed the anaconda platform for python on our machine. From various environments offered by anaconda we selected jupyter notebook as our IDE. Several anomaly detection experiments are also performed on pycharm IDE for python. The machine learning experiments are executed using various Scikit-learn methods and anomaly detection is performed using PyOD anomaly detection techniques.

- Scikit is a free software machine learning library for the Python programming language [34]. Scikit-Learn offers a wide variety of methods for data mining

and analysis. It includes various classification, regression and clustering algorithms like Random Forest, K-nearest Neighbors, Support Vector Machine, K Means, Gradient Boosting algorithms which works with the Python numerical and scientific libraries NumPy and SciPy.

- PyOD is a scalable Python toolkit for detecting outliers in multivariate data. It provides access to around 20 outlier detection (Anomaly Detection) algorithms under a single well-documented API [24]. It supports advance models like neural networks, deep learning and outlier ensembles.

6.2 Dataset Description

This section discusses about the three benchmark keystroke datasets that we have considered for our research experiments. The sections mention about the various features and number of samples available for each class of the dataset. We are considering datasets which have been collected from different platforms by typing the same password '.tie5Roanl'. We are using the normal personal computer based dataset for our machine learning experiments and for anomaly detection we have considered all the three datasets.

6.2.1 Personal Computer Keyboard Based Keystroke Dataset

The dataset is a benchmark data set for keystroke dynamics proposed by Kevin Killourhy and Roy Maxion [19]. 51 subjects (typists) typed the password (.tie5Roanl) 400 times over 8 sessions (50 repetitions per session). They waited at least one day between sessions, to capture some of the day-to-day variation of each subject's typing. The data are arranged as a table with 34 columns. Each row of data corresponds to the timing information for a single repetition of the password by a single subject.The first column, subject, is a unique identifier for each subject (e.g., s002 or s057). The second column, sessionIndex, is the session in which the password was typed (ranging from 1 to 8). The third column, rep, is the repetition of the password within the

session (ranging from 1 to 50).

The remaining 31 columns present the timing information for the password. The name of the column encodes the type of timing information. Column names of the form H.key designate a hold time for the named key (i.e., the time from when key was pressed to when it was released). Column names of the form DD.key1.key2 designate a keydown-keydown time for the named digraph (i.e., the time from when key1 was pressed to when key2 was pressed). Column names of the form UD.key1.key2 designate a keyup-keydown time for the named digraph (i.e., the time from when key1 was released to when key2 was pressed).

subject	sessionIndex	rep	H.period	DD.period.t	UD.period.t	...
s002	1	1	0.1491	0.3979	0.2488	...

Fig. 6.2.1: Benchmark dataset snapshot

Fig. 6.2.1 presents typing data for subject 2, session 1, repetition 1. The period key was held down for 0.1491 seconds (149.1 milliseconds); the time between pressing the period key and the t key (keydown-keydown time) was 0.3979 seconds; the time between releasing the period and pressing the t key (keyup-keydown time) was 0.2488 seconds; and so on.

6.2.2 Android Keystroke Dataset - I

This benchmark keystroke dataset is proposed by Margit Antal and Lehel Nemes in [5]. The data is collected from 54 volunteers through an android application. 13 identical Nexus 7 tablets were used to collect the data. The password used to record the data was '.tie5Roanl'. The data were collected in 3 sessions which were one week apart. In each session users entered 20 entries of the password which summed up to total of 60 entries per user in the dataset.

The application implemented a custom keyboard to store the timing , touch screen and other related raw data from the user's typing. Typing of the password required typing 13 keys: 8 letters, a digit, a period character, a shift key to type the capital

letter and two times the numerical key to switch to and from numerical keypad. The features in the dataset are summarized in the table 6.2.1 [5]:

Table 6.2.1: Feature Set - Android Keystroke Dataset-I

Feature	**Feature Explanation**	**No. of Features**
Hold time (HT)	Time between key press and release	13
Down Down Time (DD)	Time between consecutive key presses	12
Up-down time (UD)	The time between key release and next key press	12
Pressure (P)	Pressure at the moment of key press	13
Finger area (FA)	Finger area at the moment of key press	13
Mean Hold Time (MHT)	Average of key hold time values	1
Mean Pressure (MP)	Average of key pressure values	1
Mean finger area (MFA)	Average of finger areas	1
Mean X acceleration (MAX)	Mean X acceleration	1
Mean Y acceleration (MAY)	Mean Y acceleration	1
Mean Z acceleration (MAZ)	Mean Z acceleration	1
Total distance (TD)	Sum of the distances (in pixels) between two consecutive buttons	1
Total time (TT)	Time needed to type in the password	1
Velocity (V)	Quotient of the distance and the total time	1
Total		**72**

6.2.3 Android Keystroke Dataset - II

An android application was developed to collect the user's typing data by Antal et al. [4]. The data were collected in two sessions in which each participant entered the password '.tie5Roanl' for 30 times per session. All an all 42 people participated in the study. Though the passwords were entered 60 times by the users, some samples having deletions were dropped from the final dataset. The final dataset has minimum 51 entries per user.

For data collection Nexus 7 tablet and LG Optimus L7 II P710. In total 37 tablet users and 5 mobile phone users supplied the data. Typing the chosen password required to press 8 letter keys, a digit, a period character, twice the shift key in order to type capital letter and twice the numerical keyboard switch key. The feature

vectors are summarized in the table 6.2.2 [4]:

Table 6.2.2: Feature Set - Android Keystroke Dataset-II

Feature	**Feature Explanation**	**No. of Features**
Key Hold Time (H)	Time between key press and release	14
Down-down time (DD)	Time between consecutive key presses	13
Up-down time (UD)	The time between key release and next key press	13
Key Hold Pressure (P)	Pressure at the moment of key press	14
Finger area (FA)	Finger area at the moment of key press	14
AH (Average Hold Time)	Average of key hold times	1
AP	Average of key pressures	1
Total		**71**

6.3 Multi-class Machine Learning Methods

In this section, we demonstrate the experiments for the multi-class machine learning methods and how it performs in a keystroke-based access control systems. A benchmark keystroke dataset [19] for the normal personal computer keyboard is used to run the experiments.

6.3.1 Data Preprocessing

As a part of data preprocessing we removed qualitative features like 'subject', 'sessionIndex', 'rep' from the dataset. In classification there are always two vectors: Feature vector (X) and Target vector (Y). Here, 'subject' is the target vector (y) and other features are considered in the feature vector (X).

6.3.1.1 Grid Search Method

To improve the performance of classification various model parameters are tuned using the 'Grid Search' method. The detail of how the grid search is performed and

as a result best parameters for a particular model are described in this section.

- **Support Vector Machine:**

  To perform grid search on support vector machine we provided three parameters to it:

  1. 'SVC (Support Vector Classification)' function from sklearn library 'svm'
  2. tuned_parameters = ['Kernel': ['linear', 'rbf', 'poly'], 'c': [0.1, 1, 10, 100, 1000], 'degrees': [0, 1, 2, 3, 4, 5, 6]] where, the kernel parameters choose the type of hyperplane used to isolate the data. 'linear' is used for a linear hyperplane. 'rbf' and 'poly' is used for a non-linear hyper-plane, C is a regularizer which controls the trade off between the decision boundary and the correct classification of training points, degree parameter is considered when the kernel is set to 'poly'. It is the degree of the polynomial to find the hyperplane to split the data.
  3. scores = ['precision', 'recall']

  for the support vector machine the best parameter set returned by the grid search method is: kernel = 'linear', C = 100.

- **Decision Tree:**

  The parameters provided to tune for decision tree classifier are:

  1. 'DecisionTreeClassifier' method from sklearn library 'tree'
  2. tuned_parameters = ['criterion': ['gini','entropy'],'max_depth': np.arange(3, 100)] Where, max_depth denotes how deep the tree is. The more deeper the tree the more splits it has and it can capture more information from the data. 'criterion' is the function to measure the quality of a split. Supported criteria are "gini" for the Gini impurity and "entropy" for the information gain.
  3. scores = ['precision', 'recall']

The best parameter set found for decision tree classifier is criterion= 'entropy', max_depth = 78

- **KNN:**

  To perform grid search on KNN we provided three parameters to it:

  1. 'KNeighborsClassifier' method from sklearn library 'neighbors'.
  2. tuned_parameters = ['n_neighbors': np.arange(1, 143), 'weights': ['uniform','distance'], 'metric': ['euclidean','manhattan']] Where, n_neighbors represents the number of neighbors to use. It require to get the best value of K (number of neighbors) which gives the best performance, 'weight' is the weight function used in prediction. 'uniform' : uniform weights. All points in each neighborhood are weighted equally. 'distance' : weight points by the inverse of their distance. in this case, closer neighbors of a query point will have a greater influence than neighbors which are further away. metrics is the distance metric to use for the tree.
  3. scores = ['precision', 'recall']

  The best parameter set found for KNN classifier is n_neighbors = 3, weights='distance', metric='manhattan'

- **Logistic Regression:**

  The parameters provided to tune for naive bayes classifier are:

  1. skleran's linear model library's LogisticRegression method
  2. tuned_parameters = ['solver': ['newton-cg', 'lbfgs', 'liblinear', 'sag', 'saga'], 'max_iter': np.arange(100, 2000) , 'multi_class' : ['ovr', 'multinomial']] Where, solver is the algorithm to use in the optimization problem, 'max_iteration' is the maximum number of iterations taken for the solvers to converge and In the multi_class parameter, if the option chosen is 'ovr', then a binary problem is fit for each label. For 'multinomial' the loss minimised is the multinomial loss fit across the entire probability distribution, even when the data is binary.

3. scores = [‘precision’, ‘recall’]

The best parameter set found for logistic regression classification is max_iter=2000, solver=‘saga’, multi_class=‘multinomial’

- **Random Forest:**

  To perform grid search on Random Forest we provided three parameters to it:

  1. ‘RandomForestClassifier’ method from sklearn library ‘ensemble’
  2. tuned_parameters = [‘n_estimators’: np.arange(50, 1050) , ‘max_depth’: np.arange(3, 100)] Where, ‘n_estimators’ represents the number of trees in the forest, ‘max_depth’ denotes the maximum depth of the individual tree in the forest.
  3. scores = [‘precision’, ‘recall’]

  The best parameter set found for Random Forest classifier is n_estimators=1000, max_depth=10

- **Multi Layer Perceptron:**

  To perform grid search on MLP we provided following parameters to it:

  1. ‘MLPClassifier’ method from sklearn library ‘neural_network’ is used.
  2. tuned_parameters = [‘solver’: [‘lbfgs’, ‘sgd’, ‘adam’], ‘alpha’: uniform(0.0001, 0.9), ‘hidden_layer_sizes’: [(sp_randint.rvs(100,300), sp_randint.rvs(100,300)], ‘max_iter’: np.arange(100, 1000)] Where, ‘solver’ is for weight optimization, Alpha is L2 penalty (regularization term) parameter. In the ‘hidden_layer_sizes’ the ith element represents the number of neurons in the ith hidden layer. ‘max_iter’ is the maximum number of iterations. The solver iterates until convergence or this number of iterations.
  3. scores = [‘precision’, ‘recall’]

  The best parameter set found for MLP classifier is solver=‘adam’, alpha=0.001, hidden_layer_sizes=(150, 100), max_iter=1000

- **Light GBM:**

  To perform grid search on LGB we provided following parameters to it:

  1. 'lgb' method from the library 'lightgbm'
  2. tuned_parameters = ['num_leaves': np.arange(0, 25), 'colsample_bytree': uniform(0, 0.9), 'learning_rate': uniform(0, 0.9), 'min_child_samples': np.arange(100, 500), 'min_child_weight': uniform(0, 0.1), 'reg_alpha': np.arange(0,1), 'reg_lambda': np.arange(0,1), 'subsample': np.arange(0,1)] Where, 'num_leaves' is maximum tree leaves for base learners, colsample_bytree is subsample ratio of columns when constructing each tree, learning_rate is the boosting learning rate, 'min_child_samples' denotes minimum number of data needed in a child(leaf), 'min_child_weight' is the minimum sum of instance weight needed in a child (leaf), 'reg_alpha' is the L1 regularization term on weights, 'reg_lambda' is the L2 regularization term on weights, 'subsample' is the subsample ratio of the training instance.
  3. scores = ['precision', 'recall']

  The best parameter set found for LGB classifier is 'num_leaves': 22, 'colsample_bytree': 0.87, 'learning_rate': 0.05, 'min_child_samples': 475, 'min_child_weight': 1e-05, 'reg_alpha': 0.1, 'reg_lambda': 1, 'subsample': 0.587

6.3.2 Part-I: Find Sheep, Goat, Lamb

In this section we perform experiments to get the sheep, goat and lamb type of users from the given dataset. Firstly, we performed classification on the given dataset by keeping the train/test ratio of 70-30 %. From the classification predictions we plotted a multiclass confusion matrix and calculated the metrics: False Negative Rate, False Positive Rate and True Positive Rate. To get the sheep, goat and lamb we formulated the criterion by taking the reference from the doddington's zoo.

6.3.2.1 Classification

From the grid search method we could get the best possible parameters for the classifiers. We took 280 samples out of 400 samples per user to train the classification model and tested the model with remaining 120 samples per user. Classification is performed by first training the classification model by training data and to get the predictions for the new samples testing set is used. Output of the classification is shown in terms of classification accuracy in Table 6.3.1.

Table 6.3.1: Classification results

No.	**Classifiers Used**	**Accuracy**
1	Support Vector Machine (SVM)	85.9 %
2	Decision Tree	73.0 %
3	K Nearest Neighbor (KNN)	82.5 %
4	Naïve Bayes	66.7 %
5	Logistic Regression	71.2 %
6	Random Forest	87.3 %
7	Multi Layer Perceptron (MLP)	91.7 %
8	Light GBM	94.7 %

We can see from the table 6.3.1 that, some of the classifiers are working quite well with the dataset like light GBM (94.7%), multi layer perceptron(91.7%), random forest (87.3%), support vector machine (85.9%) and k nearest neighbors (82.5%).

6.3.2.2 Problem In Finding The Lamb

As per the lamb's definition, lambs are responsible for the majority of false acceptance rate, which means any other user is accepted by a system as a lamb. So, false acceptance rate is nothing but False Positive Rate. FPR is defined by proportion of negative cases classified as positive cases.

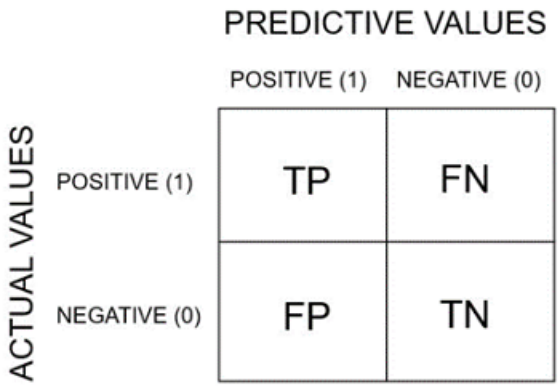

Fig. 6.3.1: Two Class Confusion Matrix [38]

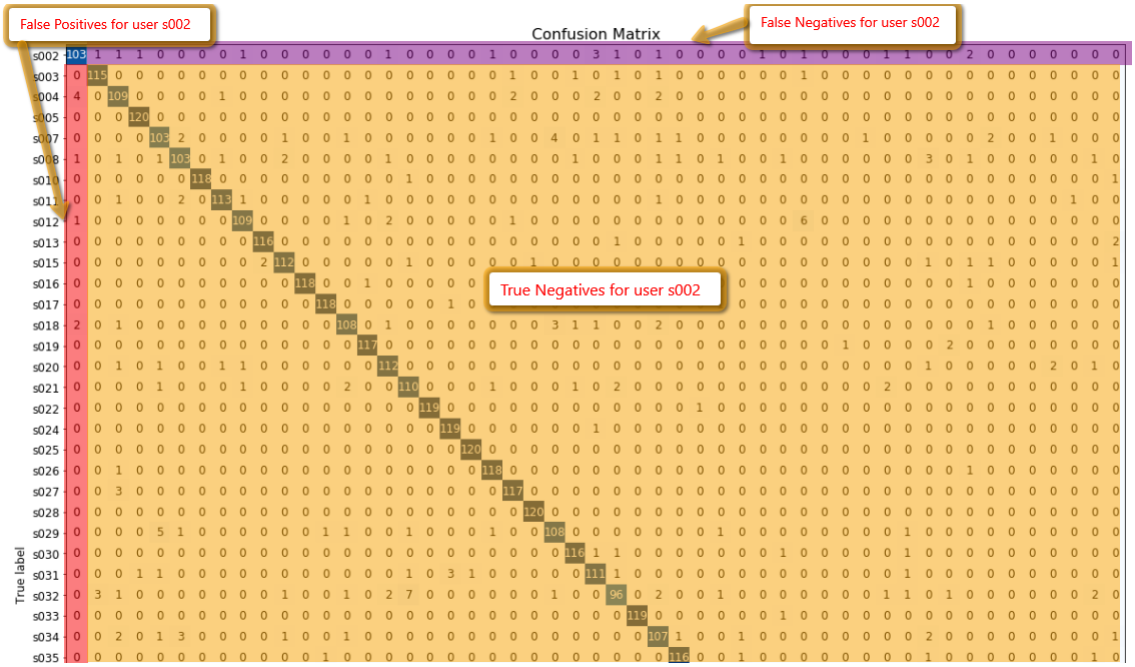

Fig. 6.3.2: Illustrating FN, FP, TP, TN in resultant multiclass confusion matrix

In the case of multiclass classification, the False Positives (False Accepts) of one user appears in the entries of the False Negatives (False Rejects) of other users. Which is not the case with the two class classification confusion matrix. Fig. 6.3.2 demonstrates the resultant multiclass classification matrix. The column highlighted with red color denotes the false positives for user 's002', The row marked in violet is the false negatives for the user 's002', The square at the top left corner in the

blue color is the truly predicted samples (i.e, True Positives) of the user 's002' and everything else except TP, FP and FN is TN. Consider the row of the user 's004', the very first 4 samples belongs to the false positives of the user 's002' and the same 4 samples are counted as one of the false negatives while calculating the false negatives of the user 's004'!

Also, for multiclass confusion matrix true negative (true rejects) is everything else from the confusion matrix that is not true positive (true accepts), false positive (false accepts) or false negative (false rejects). So, the value of true negative (true rejects) will be huge in this case, which will not give equivalent value for false positive rate (false acceptance rate) as compared to false negative rate (false rejection rate).

Based on above two findings, it can be said that the concept of lamb becomes fuzzy when we talk about calculating the false acceptance rate from the multiclass confusion matrix. The same applies to the wolf because if there is no lamb, then there is no wolf. The sheep and the goat can be easily found.

The findings show that when there are more than two classes, the multi-class classification techniques are useful only to get the False Rejection Rate (FRR) and also the True Acceptance Rate (TAR). Which alone is not sufficient to understand the system's behavior. Consequently, by using multi-class classifiers we are unable to debug the biometric zoo model entirely.

6.3.2.3 Find Sheep And Goat

- **What can be a Goat?**

  Goat users have highest chances of rejections when one tries to login with his own true identity. In Doddington's zoo goats are defined as below the 2.5 percentile of average match score. So, we considered TPR as the match score and calculated the 2.5th percentile of TPR. To calculate 'Goat' following constraint is considered:

  TPR $<$ 2.5th percentile of average TPR

- **What can be a Sheep?**

System performs nominally well for them and it is easily detected as a true user. Sheep is any user which is not a goat. Goats are calculated with TPR less than 2.5th percentile of TPR then sheep should be somebody at and over 2.5th percentile of TPR. To calculate 'Sheep' following constraint is considered:

TPR >= 2.5th percentile of average TPR

6.3.2.4 Sheep And Goat Results

In this section we demonstrate the results of applying sheep and goat constraints on the classification metrics (TPR) for each user to categorize them in one of the animal classes (sheep/goat). The results demonstrates that according to different classifiers users in goat and sheep category varies. In the resultant tables there are few goat users marked in red for all the classifiers like users- 2, 7, 8, 20, 32, 34, 37, 50, 51, 56 and 57. They are frequently(commonly) categorized into the goat category by the classifiers.

Table 6.3.2: Sheep and Goat Results for Support Vector Machine

Support Vector Machine (SVM) (Accuracy: 85.9%)		
User Type	**Number of Users**	**Users**
Sheep	31	3, 5, 10, 12, 16, 17, 18, 19, 22, 24, 25, 26, 27, 28, 29, 30, 33, 35, 36, 38, 39, 40, 41, 42, 43, 44, 48, 49, 52, 53, 55
Goat	20	2, 4, 7, 8, 11, 13, 15, 20, 21, 31, 32, 34, 37, 46, 47, 50, 51, 54, 56, 57

Table 6.3.3: Sheep and Goat Results for Decision Tree

Decision Tree (DT) (Accuracy: 73.0 %)		
User Type	**Number of Users**	**Users**
Sheep	29	3, 5, 10, 11, 12, 15, 16, 17, 18, 19, 22, 24, 25, 27, 28, 30, 33, 35, 36, 38, 39, 40, 42, 43, 44, 49, 52, 53, 55
Goat	22	2, 4, 7, 8, 13, 20, 21, 26, 29, 31, 32, 34, 37, 41, 46, 47, 48, 50, 51, 54, 56, 57

Table 6.3.4: Sheep and Goat Results for K - Nearest Neighbors

K- Nearest Neighbors (KNN) (Accuracy: 82.5 %)		
User Type	**Number of Users**	**Users**
Sheep	32	5, 10, 11, 12, 13, 16, 17, 19, 21, 22, 24, 25, 26, 27, 28, 29, 30, 33, 36, 38, 39, 41, 42, 43, 44, 47, 48, 49, 52, 53, 54, 55
Goat	19	2, 3, 4, 7, 8, 15, 18, 20, 31, 32, 34, 35, 37, 40, 46, 50, 51, 56, 57

Table 6.3.5: Sheep and Goat Results for Naive Bayes

Naive Bayes (NB) (Accuracy: 66.7%)		
User Type	**Number of Users**	**Users**
Sheep	31	3, 4, 5, 7, 8, 10, 11, 12, 13, 16, 17, 19, 22, 24, 25, 26, 27, 28, 29, 30, 33, 36, 40, 42, 43, 44, 48, 51, 52, 53, 55
Goat	20	2, 15, 18, 20, 21, 31, 32, 34, 35, 37, 38, 39, 41, 46, 47, 49, 50, 54, 56, 57

Table 6.3.6: Sheep and Goat Results for Logistic Regression

Logistic Regression (LR) (Accuracy: 71.2 %)		
User Type	**Number of Users**	**Users**
Sheep	30	3, 5, 10, 11, 12, 16, 17, 18, 19, 22, 24, 25, 27, 28, 30, 33, 35, 36, 38, 39, 40, 41, 42, 43, 44, 46, 48, 52, 53, 55
Goat	21	2, 4, 7, 8, 13, 15, 20, 21, 26, 29, 31, 32, 34, 37, 47, 49, 50, 51, 54, 56, 57

Table 6.3.7: Sheep and Goat Results for Random Forest

Random Forest (RF) (Accuracy: 87.3%)		
User Type	**Number of Users**	**Users**
Sheep	35	3, 4, 5, 10, 11, 12, 13, 16, 17, 18, 19, 21, 22, 24, 25, 27, 28, 29, 30, 33, 35, 36, 38, 39, 40, 41, 42, 43, 44, 47, 48, 49, 52, 53, 55
Goat	16	2, 7, 8, 15, 20, 26, 31, 32, 34, 37, 46, 50, 51, 54, 56, 57

Table 6.3.8: Sheep and Goat Results for Multi-layer Perceptron

Multi-layer Perceptron (MLP) (Accuracy: 91.7 %)		
User Type	**Number of Users**	**Users**
Sheep	37	3, 5, 10, 11, 12, 13, 15, 16, 17, 18, 19, 22, 24, 25, 26, 27, 28, 29, 30, 33, 34, 35, 36, 38, 39, 40, 41, 42, 43, 44, 48, 49, 50, 52, 53, 54, 55
Goat	14	2, 4, 7, 8, 20, 21, 31, 32, 37, 46, 47, 51, 56, 57

Table 6.3.9: Sheep and Goat Results for LightGBM

LightGBM (LGB) (Accuracy: 94.7%)		
User Type	**Number of Users**	**Users**
Sheep	35	3, 5, 10, 11, 13, 15, 16, 17, 19, 20, 22, 24, 25, 26, 27, 28, 30, 31, 33, 35, 36, 38, 39, 40, 41, 42, 43, 44, 46, 47, 49, 52, 53, 54, 55
Goat	16	2, 4, 7, 8, 12, 18, 21, 29, 32, 34, 37, 48, 50, 51, 56, 57

6.3.2.5 How Close Are Some Goats From Being A Sheep?

The analysis is done on the goat users which are not frequently (i.e, commonly) classified as a goat by the classifiers. The goat users from all the classifiers are plotted with their match rate (TPR) and the threshold for being a sheep. The purpose of this experiment is to know whether there is any classifier specific effects on the categorization of sheep and goat users.

From the plots, it can be seen that some goats are actually sheeps but due to category constraint and also due to different classifier behavior variances they are

classified in the goat category in one classifier while in others they are a sheep. For instance, Fig. 6.3.6 shows two plots for MLP and LGB, we can see that there are some goat users in MLP like users - 20, 31, 46 and 47 which are not goats according to LGB, they are the sheep users. Likewise, users like 12, 18, 29, 34, 48, 50 are goats in LGB's results but, they are sheep users in MLP's results. We can claim that there are classifier specific effects on users' categorization. So, the users getting rejections from one classifier may get accepted when we deploy another classifier to do the matching process for the templates.

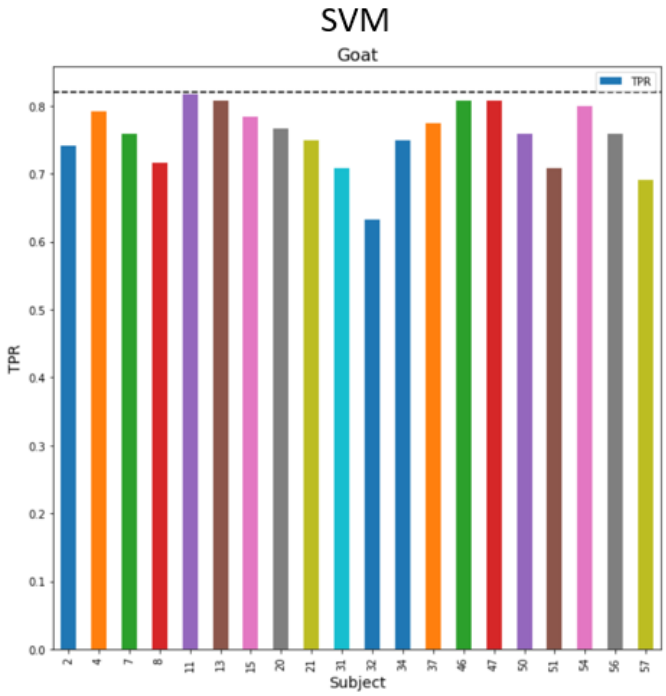

Fig. 6.3.3: Goat users' plotting with TPR and sheep threshold for SVM

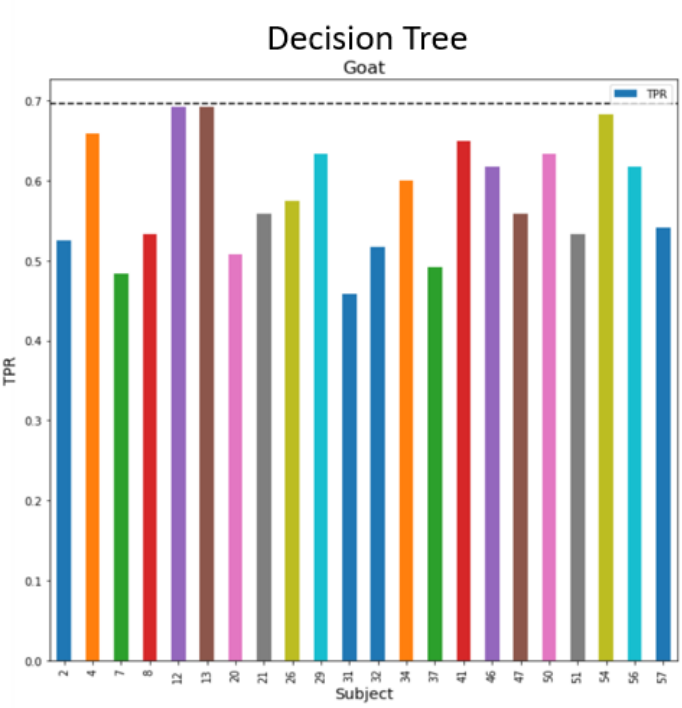

Fig. 6.3.4: Goat users' plotting with TPR and sheep threshold for Decision Tree

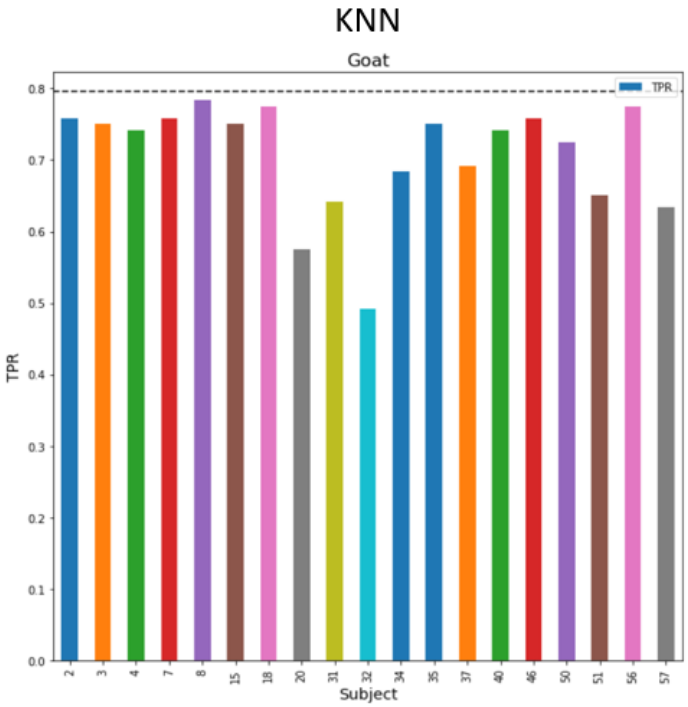

Fig. 6.3.5: Goat users' plotting with TPR and sheep threshold for KNN

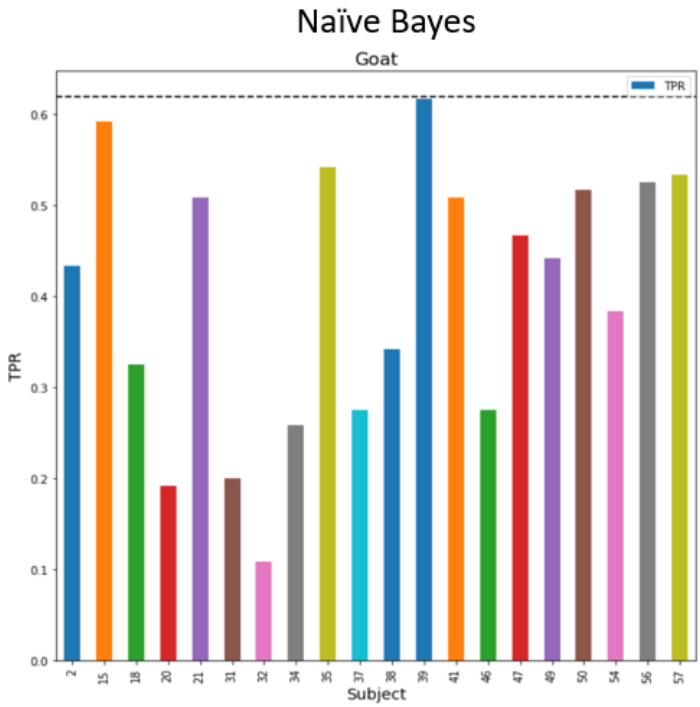

Fig. 6.3.6: Goat users' plotting with TPR and sheep threshold for Naive Bayes

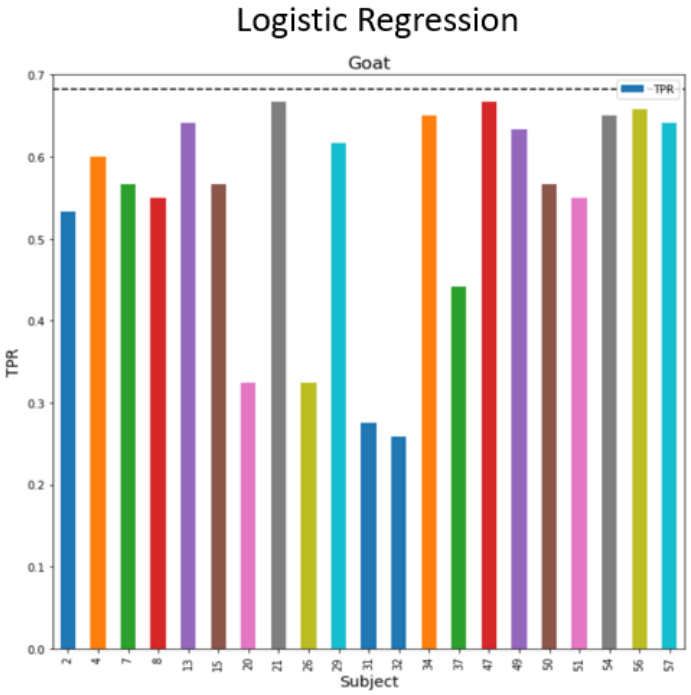

Fig. 6.3.7: Goat users' plotting with TPR and sheep threshold for Logistic Regression

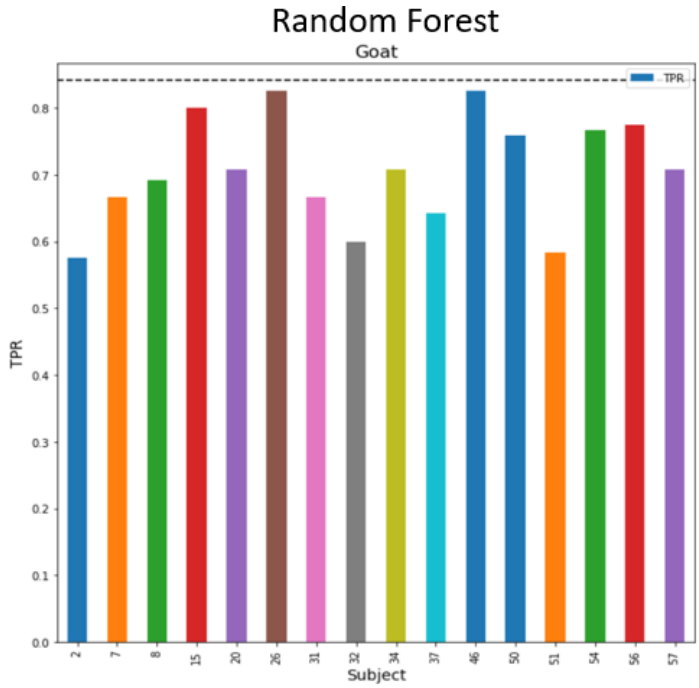

Fig. 6.3.8: Goat users' plotting with TPR and sheep threshold for Random Forest

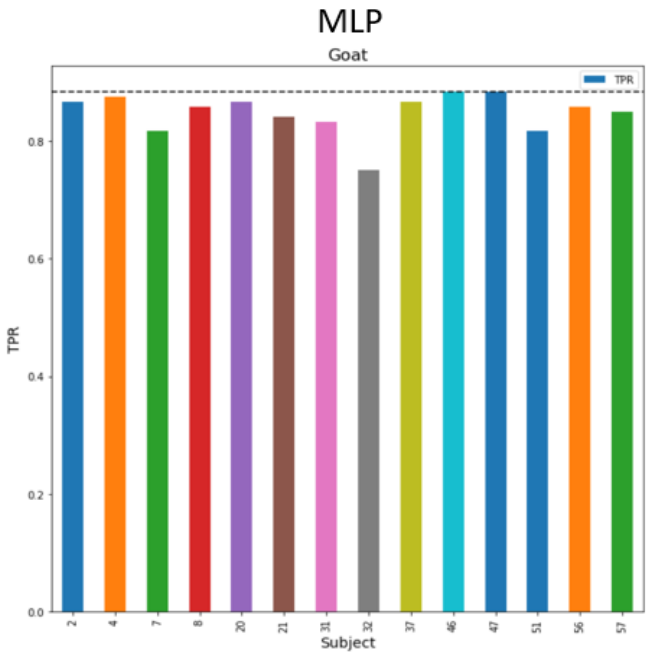

Fig. 6.3.9: Goat users' plotting with TPR and sheep threshold for MLP

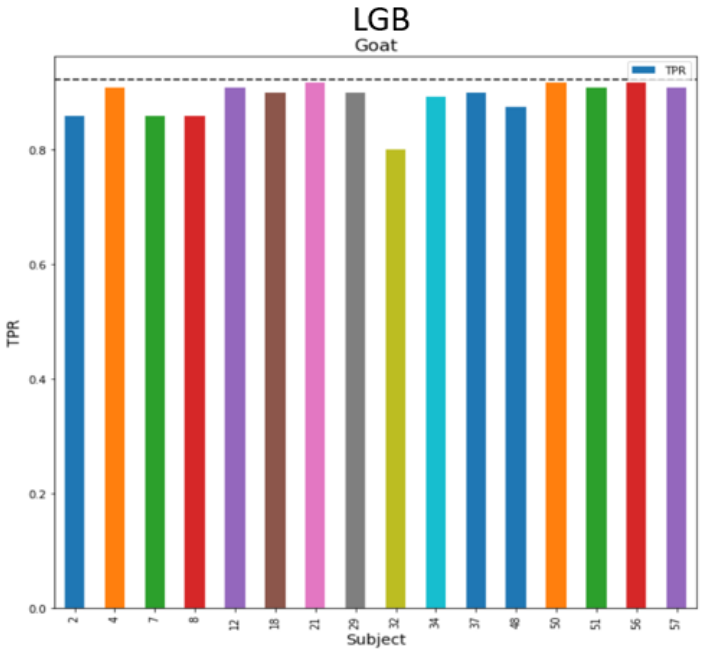

Fig. 6.3.10: Goat users' plotting with TPR and sheep threshold for LGB

6.3.2.6 Analysis Of Goatish Behavior

To analyze the goatish behavior of a user, we plotted the users with minimum false rejection rate (sheep) and Maximum false rejection rate (goat) for each classifier with their hold times for typing password '.tie5Roanl' to interpret the typing pattern for Goat users. Also, just to visualize more clearly, we plotted with only four characters '.tie'.

From the plots it is observed that the goat users have randomness in their typing behavior. It also demonstrates that the sheep users have very uniform typing patterns comparatively. For example, according to SVM's results the user with the maximum false rejection rate (i.e, goat) is s032 and the user with minimum false rejection rate (i.e, sheep) is s036. It can be seen from the user s032's results that there are major variances in their typing behavior. The spikes in the plot shows the inconsistency in the typing behavior. If we compare it with the user s036's typing behavior we can see that it fluctuates in a small range which is fine, it still exhibits consistent behavior. So, from our analysis, it can be said that the goat users are goats because of their

inconsistent typing behaviors.

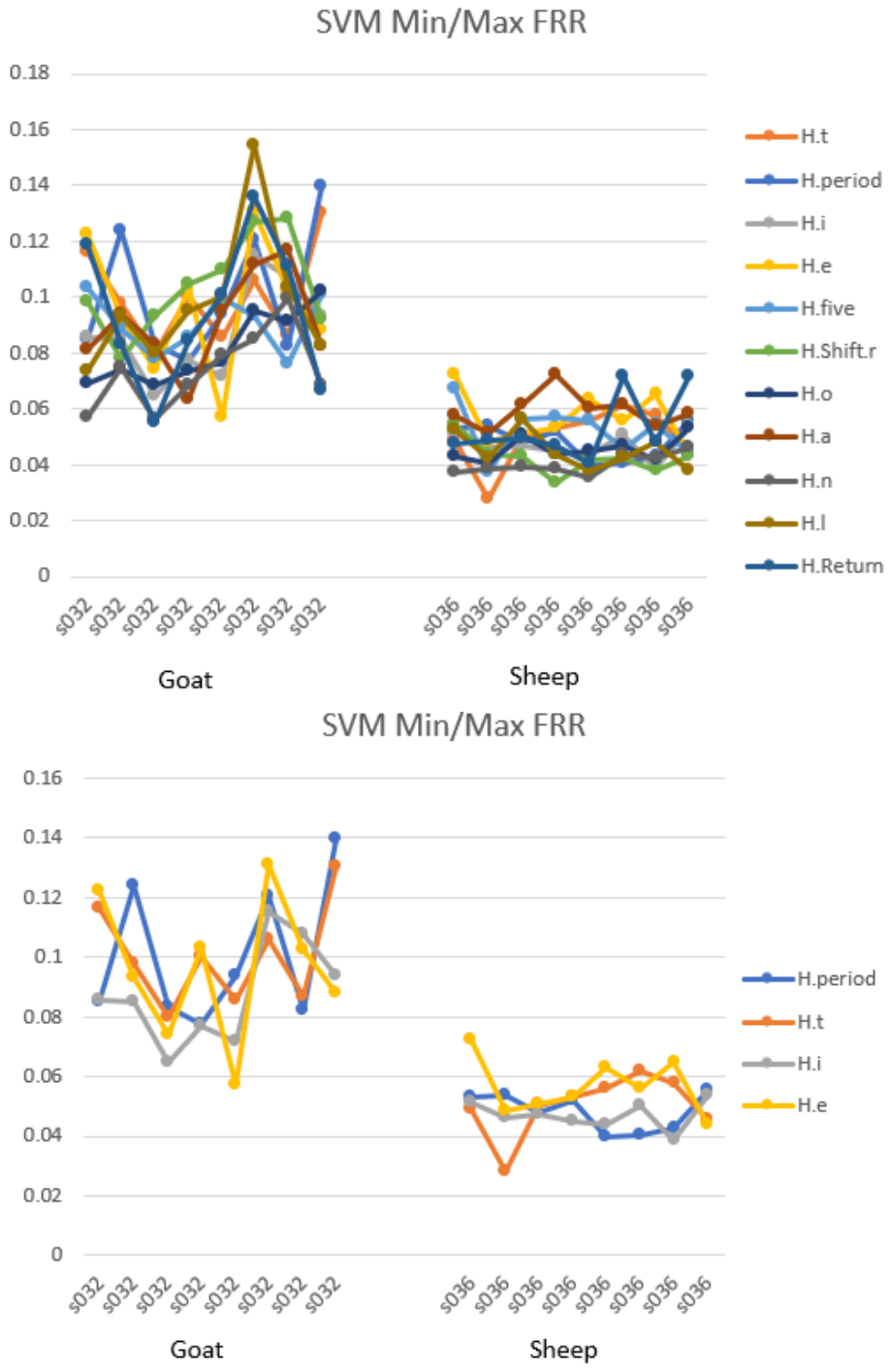

Fig. 6.3.11: SVM plot for Goat and Sheep user hold times for typing password '.tie5Roanl' and '.tie'

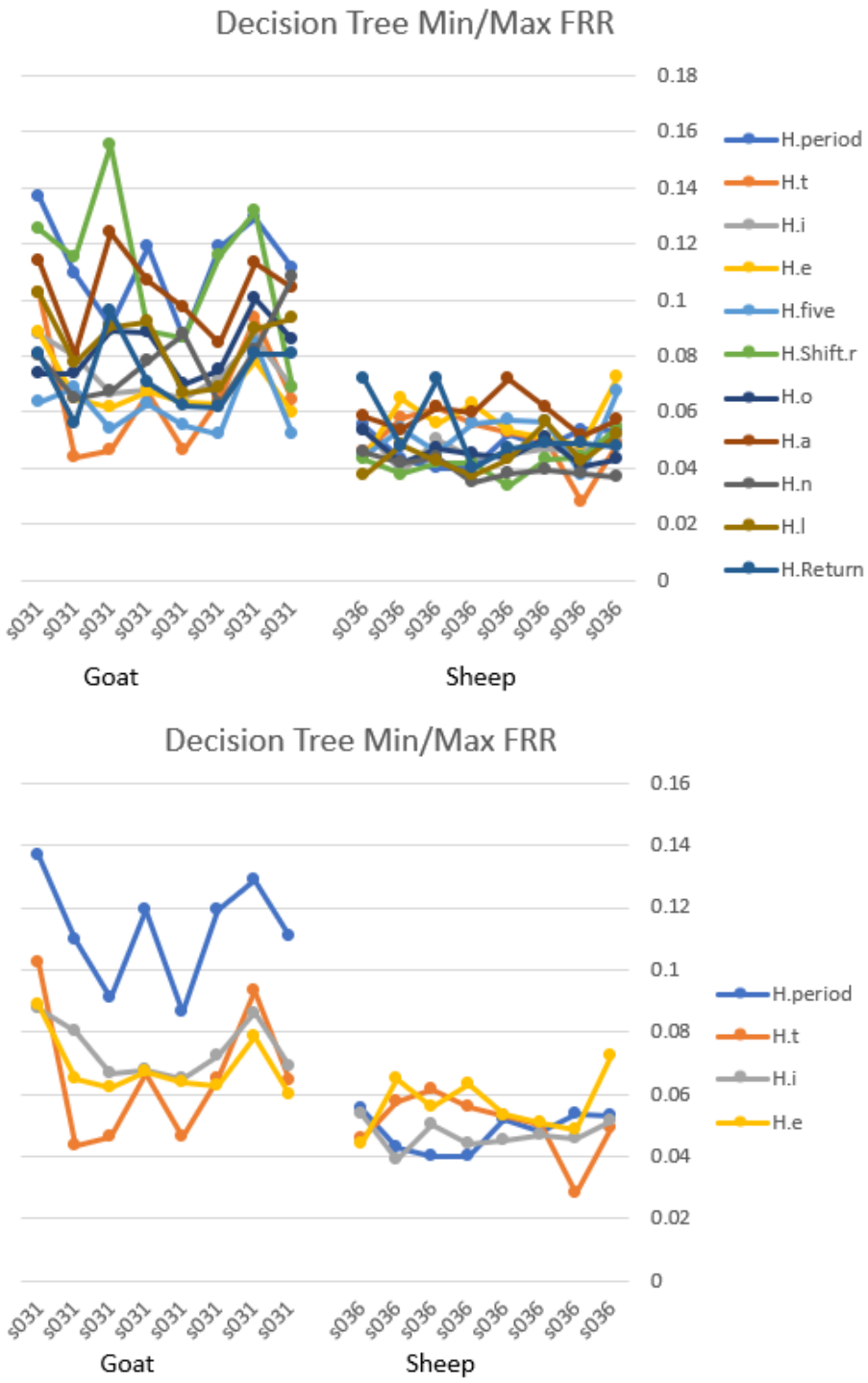

Fig. 6.3.12: Decision Tree plot for Goat and Sheep user hold times for typing password '.tie5Roanl' and '.tie'

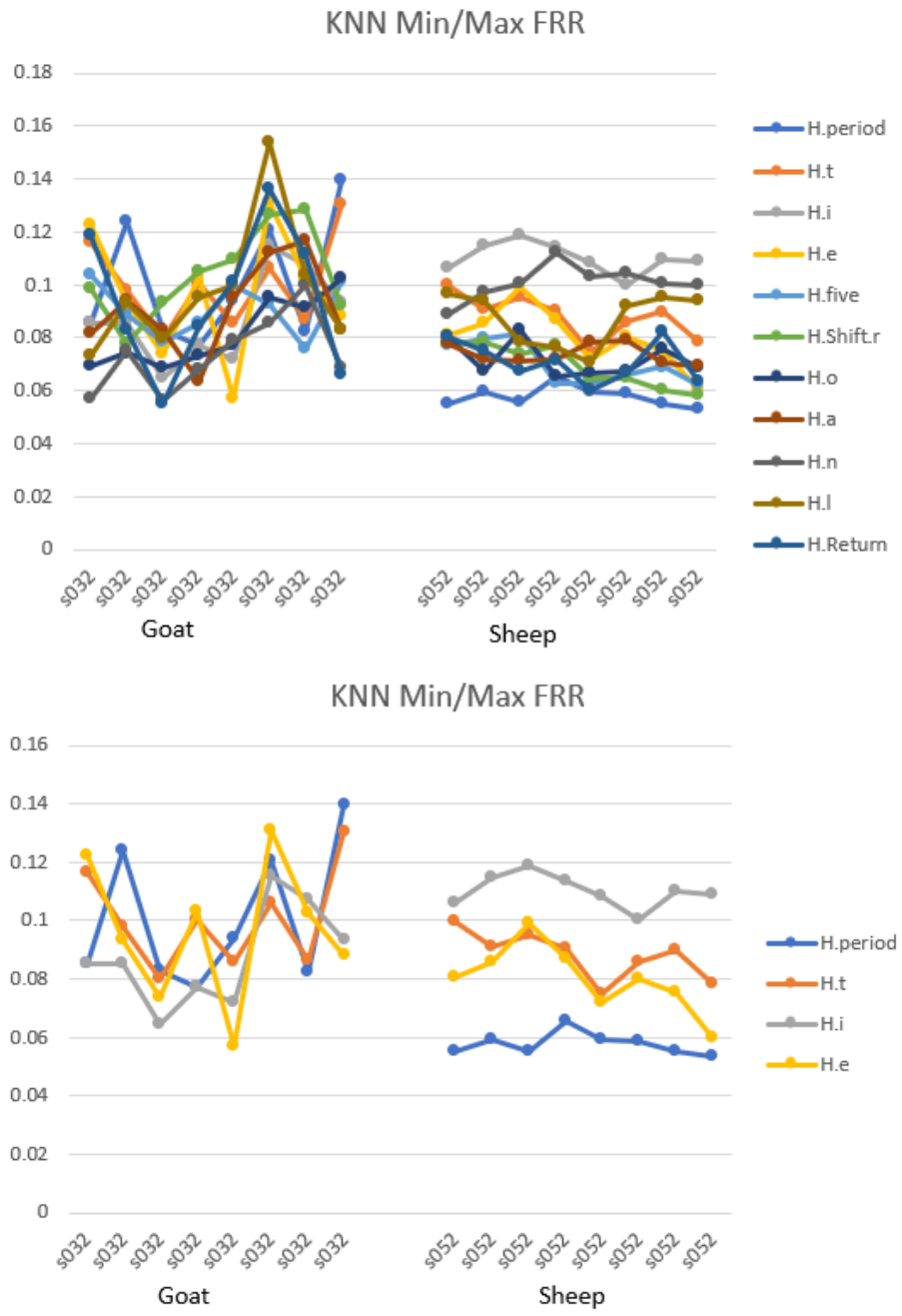

Fig. 6.3.13: KNN plot for Goat and Sheep user hold times for typing password '.tie5Roanl' and '.tie'

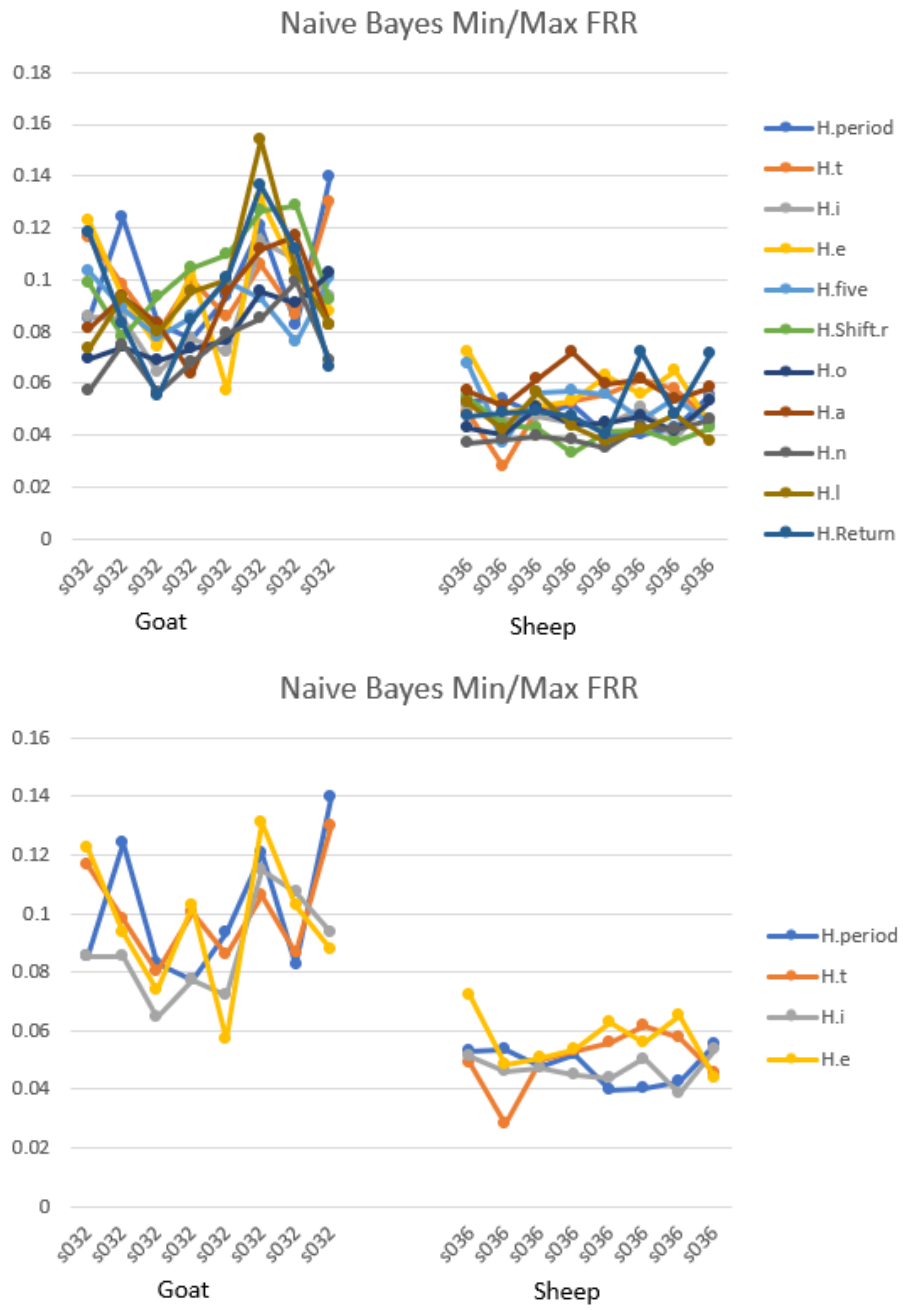

Fig. 6.3.14: Naive Bayes plot for Goat and Sheep user hold times for typing password '.tie5Roanl' and '.tie'

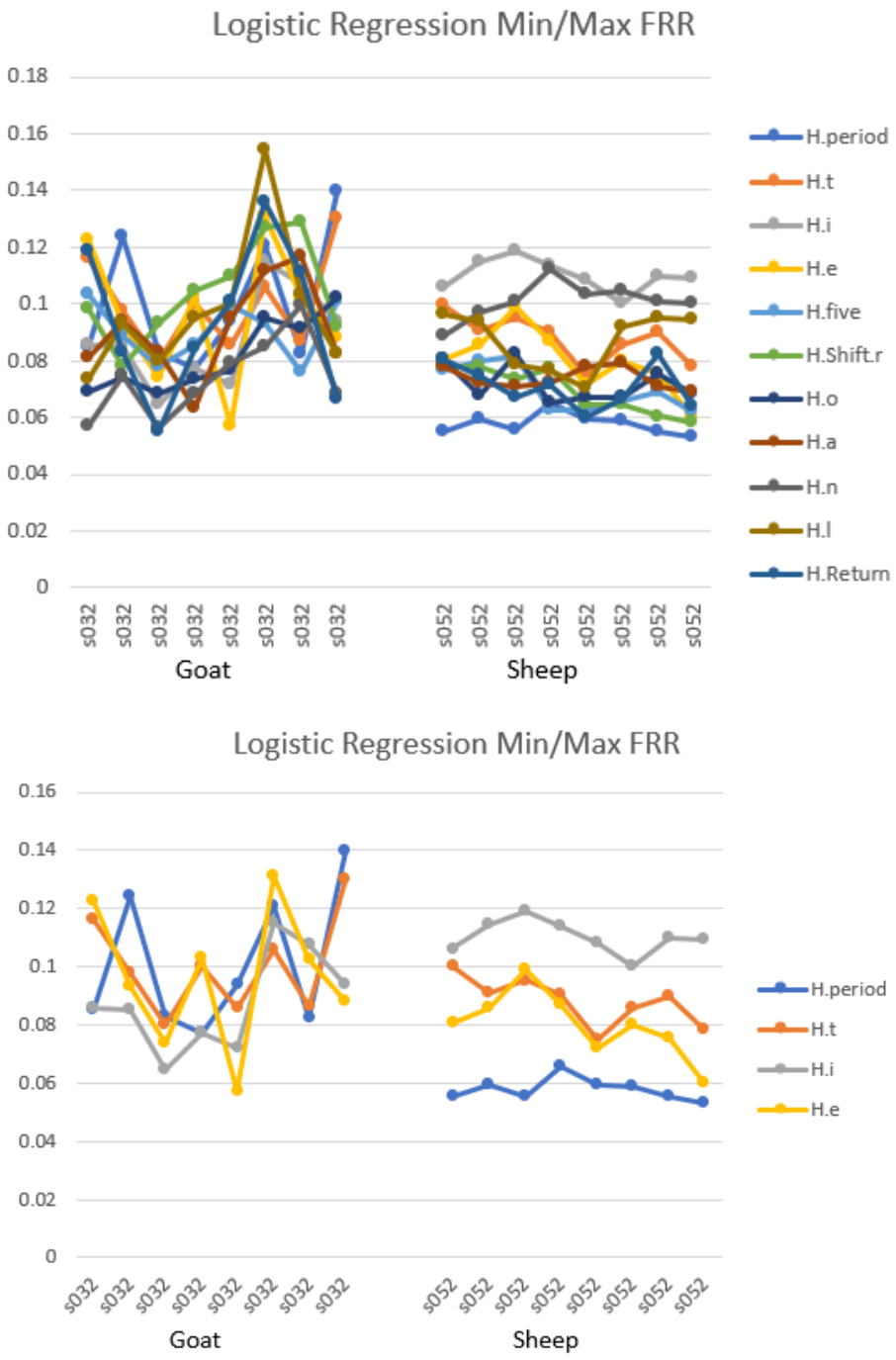

Fig. 6.3.15: Logistic Regression plot for Goat and Sheep user hold times for typing password '.tie5Roanl' and '.tie'

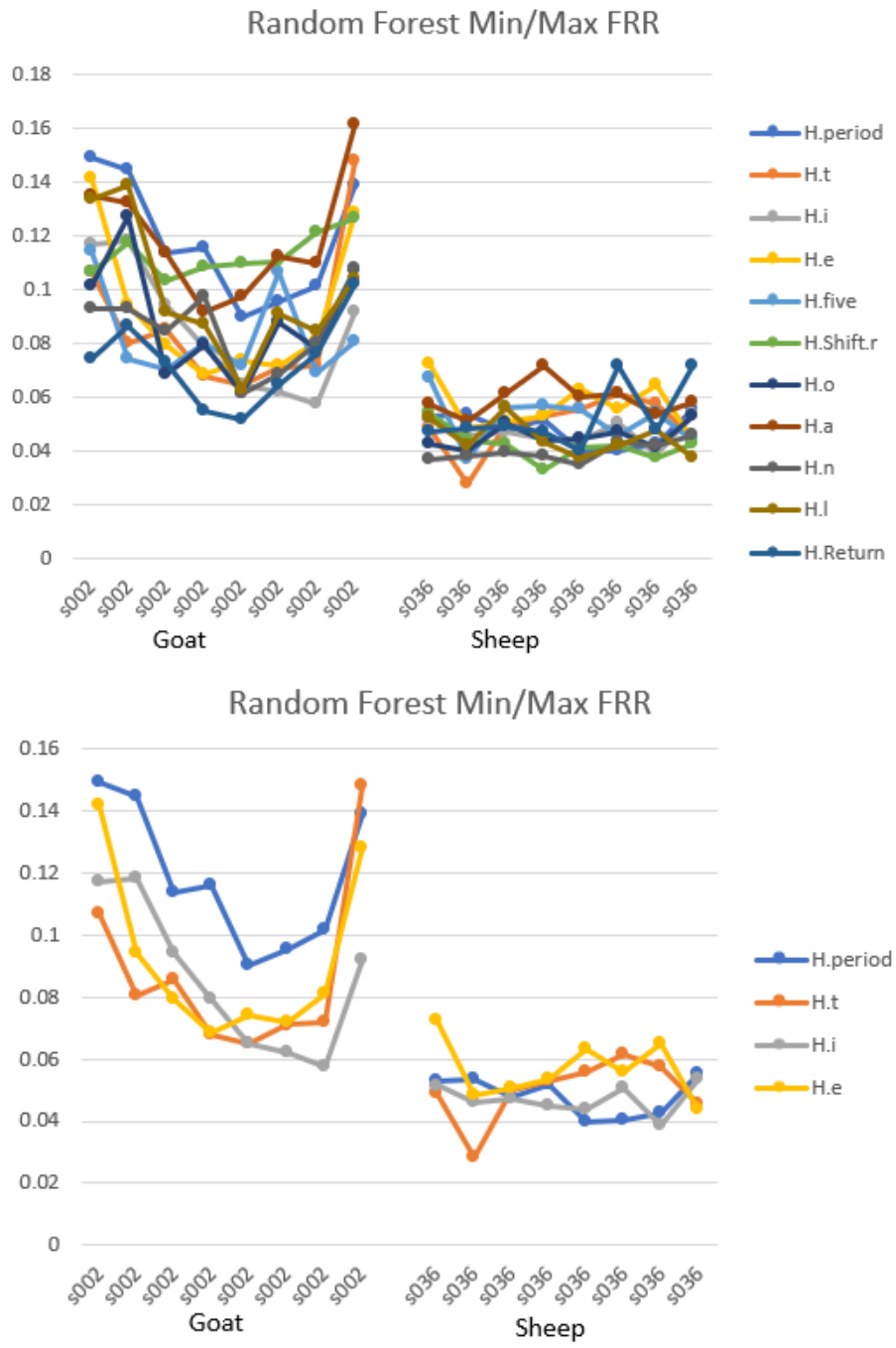

Fig. 6.3.16: Random Forest plot for Goat and Sheep user hold times for typing password '.tie5Roanl' and '.tie'

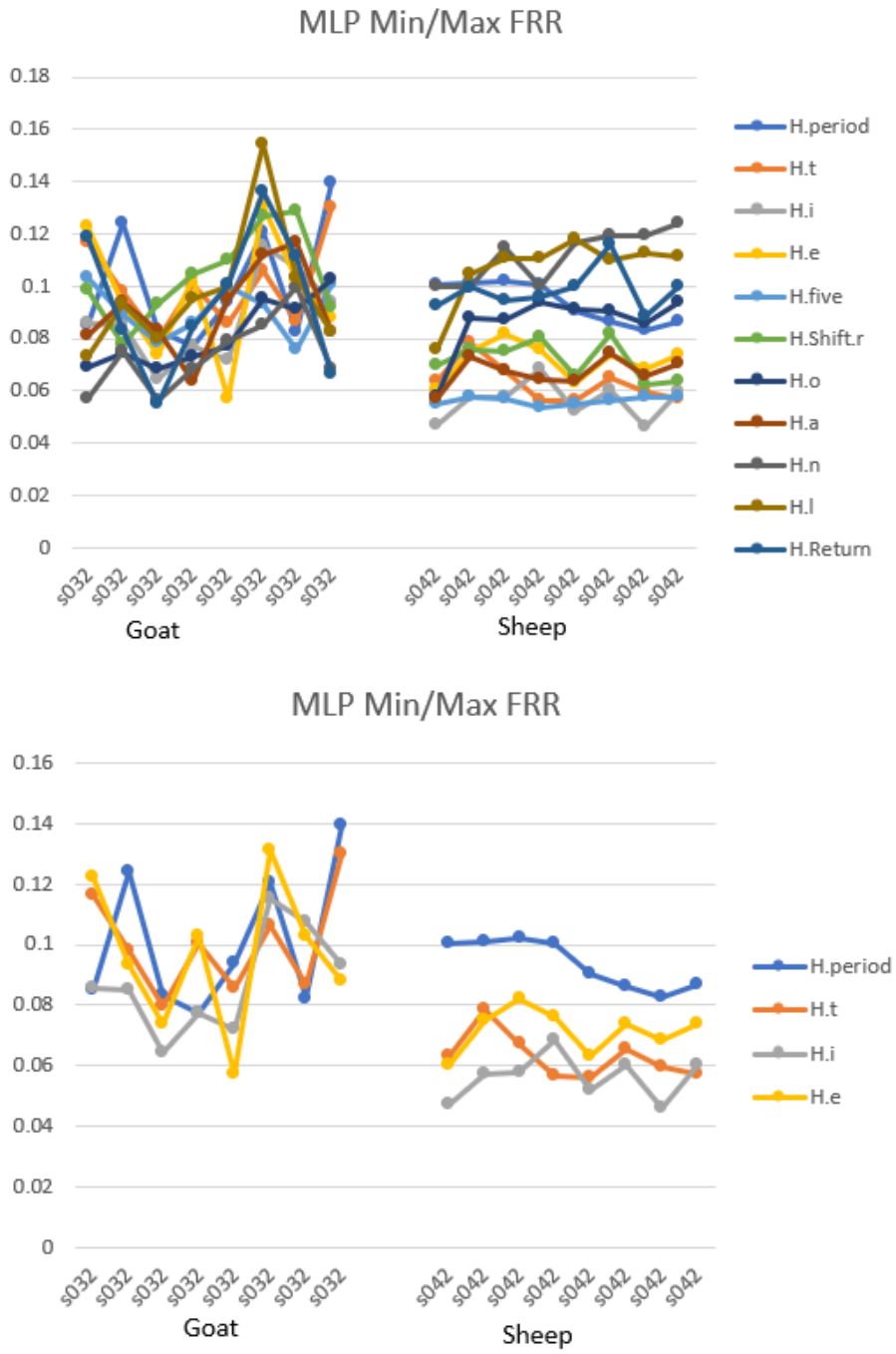

Fig. 6.3.17: MLP plot for Goat and Sheep user hold times for typing password '.tie5Roanl' and '.tie'

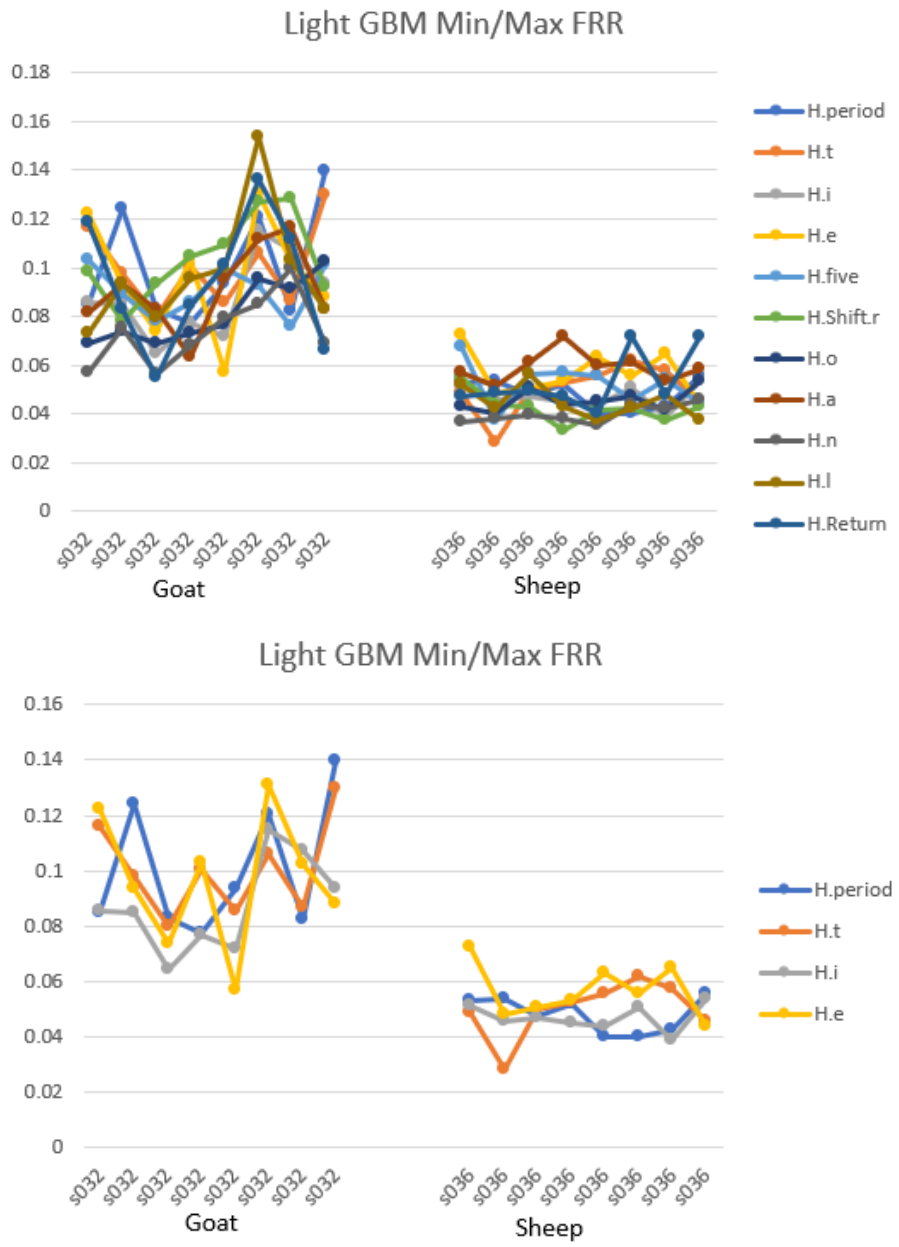

Fig. 6.3.18: LGB plot for Goat and Sheep user hold times for typing password '.tie5Roanl' and '.tie'

6.3.3 Part – II : System Generated Errors

In this section we perform experiments to find the errors generated by the system itself. The results show us that the system tries to predict unseen users as one of the other available user classes. We call this the system's tendency to generate errors

by itself. We call it a system generated artificial Wolf. Firstly, the system generates lambs and predicts unseen user (wolf) as one of those lambs. The results in the next section demonstrate the same.

6.3.3.1 Artificial Wolf Results

The wolves derived from the experiments are shown in the following tables. The users in the wolf category are either goat (G) or a sheep (S) from our Part-I's results. So, in the regular functioning of the system they are actually not a wolf but, in this experiment while we keep these goat and sheep users as held out users they are converted into the wolf by the same system. The system by itself convert some of the internal users into the lambs and categorize the held out user as one of those lambs. So, we call it the system's error which tries to falsely classify the unknown user into one of the available categories. For example, table 6.3.10 shows the system error results for the support vector machine in the results 2, 4, 7, 8, 15, 20, 21, 31, 32, 34, 37, 46, 47 and, 51 are goats, while, users 5, 26 and 35 are sheeps from the part-I's results. Additionally, the resultant tables shows that there are variation in number of the wolf users for each classifiers. For example in table 6.3.10 the number of wolves are 17, while, for KNN it reduces to 12 wolf users. So, this shows that for wolf results as well there are classification specific effects. By looking at the experiment and its results, we can claim that the machine learning based keystroke access control systems will not be able to provide correct predictions when it sees unknown samples. It will try to categorize the given unknown sample as one of the available categories. Which shows the algorithm's capacity to create errors in the system by itself. This shows that the systems is not reliable.

Table 6.3.10: Artificial Wolf Results for Support Vector Machine

Support Vector Machine (SVM) (Accuracy: 85.9%)		
User Type	**Number of Users**	**Users**
Wolf	17	31 (G), 32 (G), 8 (G), 15 (G), 37 (G), 51 (G), 21 (G), 35 (S), 47 (G), 2 (G), 4 (G), 5 (S), 7 (G), 20 (G), 26 (S), 34 (G), 46 (G)

Table 6.3.11: Artificial Wolf Results for Decision Tree

Decision Tree (DT) (Accuracy: 73.0 %)		
User Type	**Number of Users**	**Users**
Wolf	26	2 (G), 4 (G), 7 (G), 20 (G), 26 (G), 31 (G), 32 (G), 34 (G), 37 (G), 47 (G), 48 (G), 51 (G), 56(G), 8 (G), 15 (S), 25 (S), 29 (G), 30 (S) 46 (G) , 3 (S), 41 (G), 50 (G), 21 (G), 18 (S), 54 (G), 57(G)

Table 6.3.12: Artificial Wolf Results for K - Nearest Neighbors

K- Nearest Neighbors (KNN) (Accuracy: 82.5 %)		
User Type	**Number of Users**	**Users**
Wolf	12	2 (G), 4 (G), 31 (G), 32 (G), 25 (S), 48 (S), 47 (G), 7 (G), 8 (G), 29 (S), 30 (S), 34 (G), 35(G)

Table 6.3.13: Artificial Wolf Results for Naive Bayes

Naive Bayes (NB) (Accuracy: 66.7%)		
User Type	**Number of Users**	**Users**
Wolf	26	2 (G), 4 (S), 7 (S), 20 (G), 26 (S), 31 (G), 32 (G), 34 (G), 37 (G), 47 (G), 48 (S), 51 (S), 56(G), 8 (S), 15 (G), 25 (S), 29 (S), 30 (S) 46 (G) , 3 (S), 41 (G), 50 (G), 21 (G), 18 (G), 54 (G), 57(G)

Table 6.3.14: Artificial Wolf Results for Logistic Regression

Logistic Regression (LR) (Accuracy: 71.2 %)		
User Type	**Number of Users**	**Users**
Wolf	19	2 (G), 31 (G), 32 (G), 8 (G), 15 (G), 57 (G), 11 (S), 18 (S), 13 (G), 16 (S), 47 (G), 51 (G), 54 (G), 39 (S), 53 (S), 4 (G), 7 (G), 20 (G), 26 (G)

Table 6.3.15: Artificial Wolf Results for Random Forest

Random Forest (RF) (Accuracy: 87.3%)		
User Type	**Number of Users**	**Users**
Wolf	13	31 (G), 32 (G), 37 (G), 35 (S), 15(G), 18 (S), 2 (G), 4 (S), 16 (S), 26 (G), 46 (G), 50 (G), 21 (S)

Table 6.3.16: Artificial Wolf Results for Multi-layer Perceptron

Multi-layer Perceptron (MLP) (Accuracy: 91.7 %)		
User Type	**Number of Users**	**Users**
Wolf	10	31 (G), 32 (G), 2 (G), 3 (S), 15 (S), 21 (G), 26 (S), 37 (G), 54 (S), 56 (G)

Table 6.3.17: Artificial Wolf Results for LightGBM

LightGBM (LGB) (Accuracy: 94.7%)		
User Type	**Number of Users**	**Users**
Wolf	9	32 (G), 48 (G), 7 (G), 31 (S), 34 (G), 37 (G), 41 (S), 50 (G), 57 (G)

6.3.4 Summary Of The Experiments' Findings

- Through the common classification metrics we are easily able to find the sheep and the goat but not the lamb and the Wolf.
- Goats are goats because of their inconsistent typing behavior.
- Some goats are sheep, but due to category constraint and also due to different classifier behavior variances, they are classified in the goat category in one classifier while in others, they are a sheep. In general terms, it can be said that there are classifier effects on the results.
- Second experiment shows that there are system generated wolves which exist in each system so if there can be a system generated wolf then there can also be an artificial lamb in the system. In general terms, we can say that the machine learning based keystroke access control systems will not be able to

provide correct predictions when it sees unknown samples. It will categorize the given unknown sample as one of the available categories. Which shows the algorithm's capacity to create errors in the system by itself.

- Thus, by looking at the findings, it can be said that the multi-class machine learning methods are not enough unless you come up with the threshold technique to categorize the user as genuine or an imposter or in the classification you must have a way to distinguish between the legitimate and illegitimate users.

6.4 Anomaly Detection

The multi-class classification methods doesn't seem promising while we think of using it for the keystroke based access control systems. Some current state of the art methods suggest use of anomaly detection based approaches to verify the user profiles for the keystroke based biometrics system. This motivated us to use anomaly detection next for our research.

In this section we show experiments and results for the anomaly detection techniques and demonstrate how they work when used for keystroke based access control system. As mentioned in previous sections all three benchmark datasets are used for these experiments. Additionally, the section investigates the effects of feature selection and normalization on the detectors' performance. A test is presented to show the need for continuous update of the users' profiles in keystroke based access control system. Also, the experiments and results for the two proposed approaches to periodically update the user profiles are illustrated.

6.4.1 Data Preprocessing

As a part of data preprocessing we removed qualitative features like 'id', 'user_id' from both the android keystroke datasets. Also for the andorid keystroke dataset - II we performed class balancing. In the dataset majority of the classes have 51 sample entries, while there are some classes containing more than 51 samples. So, we reduced

the samples for those classes to 51 to keep the balance between the classes. We also re-labelled the features in both the datasets like the normal personal computer based keystroke dataset to maintain the consistency.

6.4.2 Experiments Of Anomaly Detection With 70 - 30 Train/Test Ratio

In this section we present experiments for anomaly detection using 70% data as training set and 30% data as a test set. The experiments and results are divided into three subsections for each dataset. The results for all three dataset demonstrate that the FAR is very high compared to the FRR. Generally, the false rejection rate is something which is related to the system's accuracy. The higher the rejections the lower the accuracy of the system. False acceptance rate demonstrates the successful attacks against a particular user. That is the part of the performance evaluation but not the system's accuracy. experiments are performed in the section 6.4.3 to see if we can reduce the false acceptance rates.

6.4.2.1 Personal Computer Keyboard Based Keystroke Dataset

To obtain the false rejection rate, 280 samples for individual user is used for training and 120 samples from the same user is considered for testing. On the trained model, total of 120 samples randomly from other users are used to get the false acceptance rate. The same procedure is repeated for all 51 users of the dataset.

Table 6.4.1: Anomaly Detection with 70-30 ratio results for Personal Computer Keyboard based Keystroke Dataset

Anomaly Detector	**FRR**	**FAR**	**Accuracy**
KNN	9.96%	61.5%	90.04%
AVG-KNN	10.4%	59.8%	89.6%
IForest	11.1%	29.1%	88.9%
One-Class SVM	10.1%	68.1%	89.9%

6.4.2.2 Android Keystroke Dataset - I

To obtain the false rejection rate, 42 samples for individual user is used for training and 18 samples from the same user is considered for testing. On the trained model, total of 18 samples randomly from other users are used to get the false acceptance rate. The same process is repeated for all 54 users of the dataset.

Table 6.4.2: Anomaly Detection with 70-30 ratio results for Android Keystroke Dataset - I

Anomaly Detector	**FRR**	**FAR**	**Accuracy**
KNN	10%	73.5%	90%
AVG-KNN	9.4%	71.8%	90.6%
IForest	13%	23%	87%
One-Class SVM	12%	71%	88%

6.4.2.3 Android Keystroke Dataset - II

To obtain the false rejection rate, 36 samples for individual user is used for training and 15 samples from the same user is considered for testing. On the trained model, total of 15 samples randomly from other users are used to get the false acceptance

rate. The same process is repeated for all 42 users of the dataset.

Table 6.4.3: Anomaly Detection with 70-30 ratio results for Android Keystroke Dataset - II

Anomaly Detector	**FRR**	**FAR**	**Accuracy**
KNN	12.5%	78.5%	87.5%
AVG-KNN	11.6%	76.6%	88.4%
IForest	11.7%	20.7%	88.3%
One-Class SVM	14%	70%	86%

6.4.3 Effects Of Feature Selection And Normalization

In this section we apply six feature selection techniques on the three datasets and get the best feature set by removing inessential features from the individual dataset. Following the feature selection we perform feature normalization. Once the feature set is ready we again perform the anomaly detection using 70-30 train/test ratio on the new feature sets and record the differences in the outcomes.

6.4.3.1 Feature selection results

- **Personal Computer Keyboard Based Keystroke Dataset**

  The results are displayed in table 6.4.4 the features marked as "True" are the important features according to particular feature selection technique. For example; the first four features are marked important by all the feature selection techniques. 6th feature is marked important by "Pearson Correlation", "Chi-Squared", "Recursive Feature Elimination (RFE)", "Logistics", "Random Forest" but according to LightGBM it is not important ("False"). We have removed those features which are not marked "True" by any of the feature selectors (Total = 0). Those are the features from 25 to 32. The total column shows the total "True" count the greater the number, the important the feature.

So, first 4 features are most important features because they have total "True" count = 6 that is every feature selector marks them as important features.

- **Android Keystroke Dataset-I**

  Same as the table 6.4.4 we derived the important features from the android dataset. We removed those features which have total "true" count = 0. So, there are total 7 features marked unimportant by all the feature selectors: UD.e.123, UD.abc.Shift, DD.period.t, DD.n.l, DD.e.123, DD.abc.Shift and DD.123.5.

- **Android Keystroke Dataset-II**

  We applied all six feature selection techniques on the second android keystroke dataset and the results suggested to remove 9 features from the feature set. The features eliminated from the final feature set are: UD.period.t, UD.e.123, UD.abc.Shift, UD.R.Shift, DD.period.t, DD.e.123, DD.abc.Shift, DD.R.Shift and DD.123.5.

6.4.3.2 Feature Selection Effects On The Performance Of Anomaly Detectors

It is evident from the resultant tables that the false acceptance rate is decreased significantly while we applied the feature selection and normalization techniques. To show the improvements in FAR from the previous results we have mentioned the old values in the red color. For example, refer the table 6.4.5 for the PC based keystroke dataset the false rejection rate for KNN reduced from 61.5% to 10.4%, for the AVG-KNN it is 10.6% which was at 59.8% previously. Similarly, for IForest and one class SVM it reduced to 10.81% and 11.5% from 29.1% and 68.1%, respectively. From the experiments outcome, it is evident that the feature selection and normalization helps in reducing the FAR and improves the overall performance of the system.

- **Personal Computer Keyboard Based Keystroke Dataset**

Table 6.4.4: Feature selection results for Personal Computer Keyboard based Keystroke Dataset

	Feature	**Pearson**	**Chi2**	**RFE**	**Logistics**	**Random Forest**	**Light GBM**	**Total**
1	H.t	True	True	True	True	True	True	6
2	H.period	True	True	True	True	True	True	6
3	H.i	True	True	True	True	True	True	6
4	H.Shift.r	True	True	True	True	True	True	6
5	UD.Shift.r.o	True	True	True	True	True	False	5
6	H.n	True	True	True	True	False	True	5
7	H.l	False	True	True	True	True	True	5
8	H.five	True	True	True	True	False	True	5
9	H.e	False	True	True	True	True	True	5
10	H.o	True	False	True	True	False	True	4
11	H.a	True	False	False	True	True	True	4
12	DD.n.l	False	True	False	True	True	True	4
13	UD.n.l	False	False	False	True	True	True	3
14	UD.a.n	True	False	True	True	False	False	3
15	DD.Shift.r.o	True	True	True	False	False	False	3
16	UD.t.i	False	False	False	False	True	True	2
17	UD.l.Return	False	True	False	False	True	False	2
18	UD.e.five	False	False	True	False	False	True	2
19	DD.l.Return	False	True	False	False	True	False	2
20	DD.e.five	True	True	False	False	False	False	2
21	UD.period.t	True	False	False	False	False	False	1
22	UD.o.a	True	False	False	False	False	False	1
23	DD.t.i	False	False	False	False	True	False	1
24	DD.period.t	True	False	False	False	False	False	1
25	sessionIndex	False	False	False	False	False	False	0
26	rep	False	False	False	False	False	False	0
27	UD.i.e	False	False	False	False	False	False	0
28	UD.five.Shift.r	False	False	False	False	False	False	0
29	DD.o.a	False	False	False	False	False	False	0
30	DD.i.e	False	False	False	False	False	False	0
31	DD.five.Shift.r	False	False	False	False	False	False	0
32	DD.a.n	False	False	False	False	False	False	0

Table 6.4.5: Feature selection effects on the performance of Personal Computer Based Keystroke Dataset

Anomaly Detector	**FRR**	**FAR**	**Accuracy**
KNN	10.37% (9.96%)	10.4% (61.5%)	89.63%
AVG-KNN	10.08% (10.4%)	10.06% (59.8%)	89.9%
IForest	11.06% (11.1%)	10.81% (29.1%)	88.9%
One-Class SVM	10.86% (10.1%)	11.5% (68.1%)	89.14%

- **Feature Selection Effects On The Android Keystroke Dataset-I**

Table 6.4.6: Feature selection effects on the Android Keystroke Dataset-I

Anomaly Detector	**FRR**	**FAR**	**Accuracy**
KNN	12.2% (10%)	16.25% (73.5%)	87.7%
AVG-KNN	11.9% (9.4%)	14.9% (71.8%)	88%
IForest	13.7% (13%)	14.4% (23%)	86.2%
One-Class SVM	15.2% (12%)	6.79% (71%)	84.8%

- **Feature Selection Effects On The Android Keystroke Dataset-II**

Table 6.4.7: Feature selection effects on the Android Keystroke Dataset-II

Anomaly Detector	**FRR**	**FAR**	**Accuracy**
KNN	14.1% (12.5%)	6.19% (78.5%)	85.86%
AVG-KNN	11.3% (11.6%)	7.14% (76.6%)	88.69%
IForest	13.09% (11.7%)	9.84% (20.7%)	86.9%
One-Class SVM	17.1% (14%)	9.52% (70%)	82.9%

6.4.4 Feature Selection With Less Data

In this section we present the experiments and results for the feature selection on the subset containing only 10 samples per user. According to the results' analysis, for different feature set sizes the results of feature selection techniques vary. The features once marked important may become unimportant while the data size provided to the feature selectors is small. The results demonstrates that the feature selection results are varying when we change the size of the data. So it is suggested not to use feature selection with less amount of samples.

6.4.4.1 Personal Computer Keyboard Based Keystroke Dataset

In the result of previous feature selection on the entire keystroke dataset (table 6.4.4) there are total 7 features which are marked unimportant by all the feature selectors: sessionIndex, rep, UD.i.e, UD.five.Shift.r, DD.o.a, DD.i.e, DD.five.Shift.r and DD.a.n. The entire dataset has total 20,400 samples.

On the other hand, when we performed the same 6 feature selections on the subset containing 51 (total no. of users) X 10 = 510 samples the features marked unimportant by all the feature selectors reduces to 4 features which are: UD.o.a, UD.n.l, DD.n.l and DD.a.n. Additionally, the features marked unimportant in both the categories are not the same.

6.4.4.2 Android Keystroke Dataset - I

In the previous feature selection results, there are total 7 features marked unimportant by all the feature selectors: UD.e.123, UD.abc.Shift, DD.period.t, DD.n.l, DD.e.123, DD.abc.Shift and DD.123.5. The dataset contains 3,240 samples in total.

We performed the same feature selection on the subset containing 54 (total no. of users) X 10 = 540 samples. Consequently, no features are marked as unimportant by all the feature selectors. This means all features are important for that particular subset.

6.4.4.3 Android Keystroke Dataset - II

The previous feature selection result suggested to remove 9 unimportant features from the dataset. The features eliminated from the final feature set are: UD.period.t, UD.e.123, UD.abc.Shift, UD.R.Shift, DD.period.t, DD.e.123, DD.abc.Shift, DD.R.Shift and DD.123.5. The dataset contains total 2,142 samples.

After feature selection on the subset of 42 (total no. of users) X 10 = 420 samples, the results marked only 3 features as unimportant which are FA.a, FA.shift and FA.123 which were marked important while we performed feature selections on the entire dataset.

6.4.5 Experiment Without Updating The User Profile

We performed and presented the results for the normal personal computer keyboard based keystroke dataset where in we have 400 samples per user. So, we update the test set with 10-10 samples per iteration, starting with 10, 20, 30, ... ,390 samples from the same user to test for the FRR. To get the FAR we considered random 10 samples from the other users (except the current user). The following are the results for the user 's002' by applying the anomaly detection with the said method.

As we can see except a few iterations in the beginning, the detectors are giving very high or almost 100% false rejection rate (FRR). It gives us an insight into the system that if we don't update the user's profile, the user has higher chances of facing

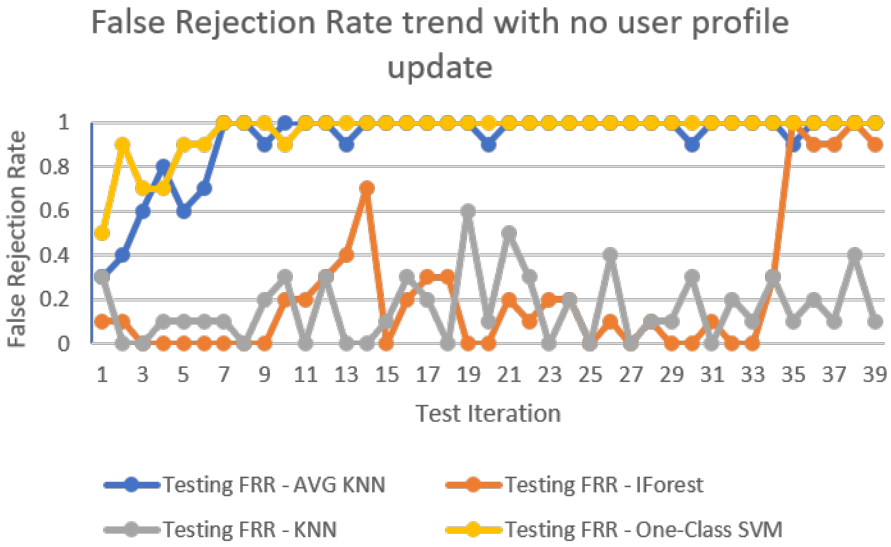

Fig. 6.4.1: Impact of No Profile Update on FRR for user 's002'

rejections from the system after sometime of user's enrollment. Thus, to maintain the system's reliability, it should get updated in some way. Following sections discuss the experiments and results for the two proposed approaches to update the user profiles.

6.4.6 Methods To Update User Profile

The section demonstrates the experiments and results for both the proposed methodology to update the user's profile. The results suggest that the sliding window approach catches more user behavior variations and is able to perform better than the batch mode approach that we experimented. Thus, we suggest using sliding window approach to update the user's profile time to time when using keystroke biometrics based access control systems.

The batch mode and sliding window results in terms of anomaly detectors' performance suggests that both the KNN anomaly detectors i.e, KNN as well as AVG-KNN are giving best results with least false rejection, false acceptance and equal error rates in the overall results.

6.4.6.1 Batch Mode Experiments And Results

In this section we present the experiments for the batch mode approach on the three data sets. The results are shown in terms of tables containing the average FRR, FAR and the accuracy of the anomaly detectors.

- **Personal Computer Based Keystroke Dataset**

  The dataset contains 400 samples per user. We divide the 400 samples into 40 subsets of 10 samples each. Next, we perform batch mode using the methodology from the section 5.2.8.1. All the steps described in the methodology are repeated until the 40 subsets are used for training for the particular user and the whole process is repeated for all 51 users of the dataset.

  The overall results for each anomaly detector is shown in the table 6.4.8. To give insight for user vise performance of the anomaly detector, results for first two users' of the dataset is also provided in the table 6.4.9.

Table 6.4.8: Batch mode overall results of anomaly detectors for PC keystroke dataset

Anomaly Detector	**FRR**	**FAR**	**ERR**	**Accuracy**
KNN	22.89%	5.97%	16.92%	77.10%
AVG-KNN	23.10%	5.50%	17.6%	76.89%
IForest	24.93%	10.44%	14.49%	75.06%
One-Class SVM	43.20%	3.20%	40%	56.79%

Table 6.4.9: Batch mode results for subjects 'S002' and 'S003' from PC keystroke dataset

Subject	**Anomaly Detector**	**FRR**	**FAR**	**Accuracy**
S002	KNN	21%	4.87%	78.9%
	AVG-KNN	20.51%	8.97%	79.48%
	IForest	25.6%	13.8%	74.3%
	One-Class SVM	48.4%	7.43%	51.5%
S003	KNN	20.25%	9.74%	79.7%
	AVG-KNN	21.5%	12.56%	78.4%
	IForest	23.58%	9.23%	76.4%
	One-Class SVM	35.38%	2%	64.6%

- **Android Keystroke Dataset-I**

  The dataset contains 60 samples per user. We divide the 60 samples into 6 subsets of 10 samples each. Next, we perform batch mode using the methodology from the section 5.2.8.1. All the steps described in the methodology are repeated are until the 6 subsets are used for training for the particular user and the whole process is repeated for all 54 users of the dataset.

  The overall results for each anomaly detector is shown in the table 6.4.10. To give insight for user vise performance of the anomaly detector, results for first two users' of the dataset is also provided in the table 6.4.11.

Table 6.4.10: Batch mode overall results of anomaly detectors for android keystroke dataset-I

Anomaly Detector	**FRR**	**FAR**	**ERR**	**Accuracy**
KNN	26.89%	10.70%	16.19%	73.11%
AVG-KNN	27.18%	7.55%	19.63%	72.81%
IForest	38.29%	9.70%	28.59%	61.70%
One-Class SVM	56.55%	2.48%	54.07%	43.44%

Table 6.4.11: Batch mode results for users '600' and '601' from android keystroke dataset-I

user_id	**Anomaly Detector**	**FRR**	**FAR**	**Accuracy**
600	KNN	20%	8%	80%
	AVG-KNN	40%	2%	60%
	IForest	43.99%	10%	55.9%
	One-Class SVM	26%	4%	74%
601	KNN	18%	16%	82%
	AVG-KNN	13.99%	2%	86%
	IForest	30%	15.99%	70%
	One-Class SVM	34%	2%	66%

- **Android Keystroke Dataset-II**

  The dataset contains 51 samples per user. We divide the 51 samples into four subsets of 10 samples and one subset containing 11 samples. Next, we perform batch mode using the methodology from the section 5.2.8.1. All the steps described in the methodology are repeated until all the 5 subsets are used for training for the particular user and the whole process is repeated for all 42 users of the dataset.

  The overall results for each anomaly detector is shown in the table 6.4.12. To

give insight for user vise performance of the anomaly detector, results for first two users' of the dataset is also provided in the table 6.4.13.

Table 6.4.12: Batch mode overall results of each anomaly detectors for android keystroke dataset-II

Anomaly Detector	**FRR**	**FAR**	**ERR**	**Accuracy**
KNN	16.67%	9.52%	7.15%	83.33%
AVG-KNN	16.33%	8.76%	7.57%	83.67%
IForest	25.95%	8.61%	17.34%	74.04%
One-Class SVM	63.9%	5.71%	58.19%	36.09%

Table 6.4.13: Batch mode results for users '1' and '2' from android keystroke dataset-II

user_id	**Anomaly Detector**	**FRR**	**FAR**	**Accuracy**
1	KNN	8%	0%	91.9%
	AVG-KNN	8%	0%	91.9%
	IForest	22%	0%	78%
	One-Class SVM	22%	20%	78%
2	KNN	8%	4%	91.9%
	AVG-KNN	6%	6%	94%
	IForest	4%	4%	96%
	One-Class SVM	40%	6%	60%

6.4.6.2 Sliding Window Experiments And Results

The experiments are performed following the methodology discussed in the section 5.2.8.2. The results are shown in terms of tables containing the average FRR, FAR and the accuracy of the anomaly detectors.

- **Personal Computer Based Keystroke Dataset**

  The training and testing process is the same as mentioned in the methodology, the process is repeated until all the 400 samples are used for training for the particular user and the whole process is repeated for all 51 users of the dataset.

  The overall results for each anomaly detector is shown in the table 6.4.14. To give insight for user vise performance of the anomaly detector, results for first two users' of the dataset is also provided in the table 6.4.15.

Table 6.4.14: Sliding window overall results of anomaly detectors for PC keystroke dataset

Anomaly Detector	**FRR**	**FAR**	**ERR**	**Accuracy**
KNN	17.69%	7.86%	9.83%	82.30%
AVG-KNN	17.65%	6.54%	11.11%	82.34%
IForest	17.85%	10.65%	7.2%	82.14%
One-Class SVM	34.32%	3.46%	30.86%	65.67%

Table 6.4.15: Sliding window results for subjects 'S002' and 'S003' from PC keystroke dataset

Subject	**Anomaly Detector**	**FRR**	**FAR**	**Accuracy**
S002	KNN	17.4%	6.38%	82.56%
	AVG-KNN	18.7%	5.66%	81.28%
	IForest	19.2%	11.7%	80.7%
	One-Class SVM	38.9%	7.1%	61%
S003	KNN	17.1%	6.46%	82.8%
	AVG-KNN	17.4%	7.92%	82.56%
	IForest	13%	11.20%	86.9%
	One-Class SVM	28.9%	4.25%	71%

- **Android Keystroke Dataset-I**

  The training and testing process is the same as mentioned in the methodology, all the steps are repeated until all the 60 samples are used for training for the particular user and the whole process is repeated for all 54 users of the dataset.

  The overall results for each anomaly detector is shown in the table 6.4.16. To give insight for user vise performance of the anomaly detector, results for first two users' of the dataset is also provided in the table 6.4.17.

Table 6.4.16: Sliding window overall results of anomaly detectors android keystroke dataset-I

Anomaly Detector	**FRR**	**FAR**	**ERR**	**Accuracy**
KNN	18.44%	10.22%	8.22%	81.56%
AVG-KNN	24.18%	15.31%	8.87%	75.81%
IForest	24.18%	15.31%	8.87%	75.81%
One-Class SVM	47.56%	1.84%	45.72%	52.44%

Table 6.4.17: Sliding window results for subjects '600' and '601' from android keystroke dataset-I

user_id	**Anomaly Detector**	**FRR**	**FAR**	**Accuracy**
600	KNN	24%	6.8%	76%
	AVG-KNN	32%	0.8%	68%
	IForest	28%	4.2%	72%
	One-Class SVM	32%	8%	68%
601	KNN	16%	3.4%	84%
	AVG-KNN	14%	0.8%	86%
	IForest	24%	17.3%	76%
	One-Class SVM	36%	1%	64%

- **Android Keystroke Dataset-II**

The training and testing process is the same as mentioned in the methodology, all the steps are repeated until the 50 samples are used for training for the particular user and the whole process is repeated for all 42 users of the dataset.

The overall results for each anomaly detector is shown in the table 6.4.18. To give insight for user vise performance of the anomaly detector, results for first two users' of the dataset is also provided in the table 6.4.19.

Table 6.4.18: Sliding window overall results of anomaly detectors android keystroke dataset-II

Anomaly Detector	**FRR**	**FAR**	**ERR**	**Accuracy**
KNN	17.02%	8.49%	8.53%	82.97%
AVG-KNN	16.42%	8.55%	7.87%	83.57%
IForest	26.9%	8.54%	18.36%	73.09%
One-Class SVM	45.65%	4.12%	41.53%	54.34%

Table 6.4.19: Sliding window results for users '1' and '2' from android keystroke dataset-II

user_id	**Anomaly Detector**	**FRR**	**FAR**	**Accuracy**
1	KNN	12.5%	0.5%	87.5%
	AVG-KNN	20%	0.25%	80%
	IForest	30%	0.75%	70%
	One-Class SVM	25%	8.25%	75%
2	KNN	10%	6%	90%
	AVG-KNN	7.5%	4.7%	92.5%
	IForest	20%	5.25%	80%
	One-Class SVM	52.5%	9.25%	47.5%

6.5 Comparisons And Discussions

In this section we present the comparison between the batch mode and the sliding window approach and deduce which one should be used in practice. We also discuss the differences or similarities in our 70-30 approach and the database literature approaches. Finally, the comparison is presented for the same dataset literature works and the proposed approaches for the user profile update.

6.5.1 Comparison Of Batch Mode And Sliding Window Approach

The comparison of both the approaches is shown through the plots of the EER and accuracy of the overall results mentioned in tables 6.4.14, 6.4.16 and 6.4.18. The plots shows that almost in all the scenarios the sliding window approach is performing better than the batch mode approach. The application of both the approaches is discussed in the end.

6.5.1.1 Personal Computer Based Keystroke Dataset

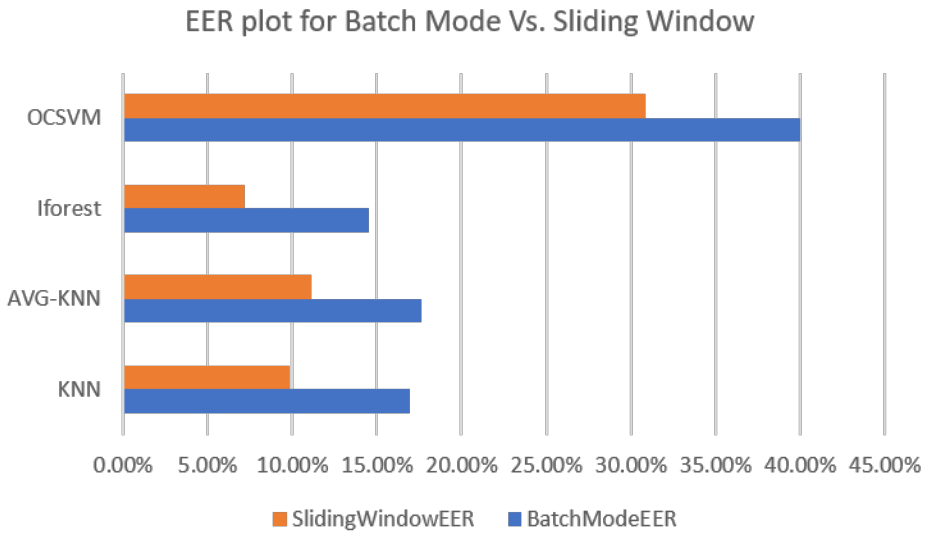

Fig. 6.5.1: PC keystroke dataset EER plot for batch mode vs. sliding window

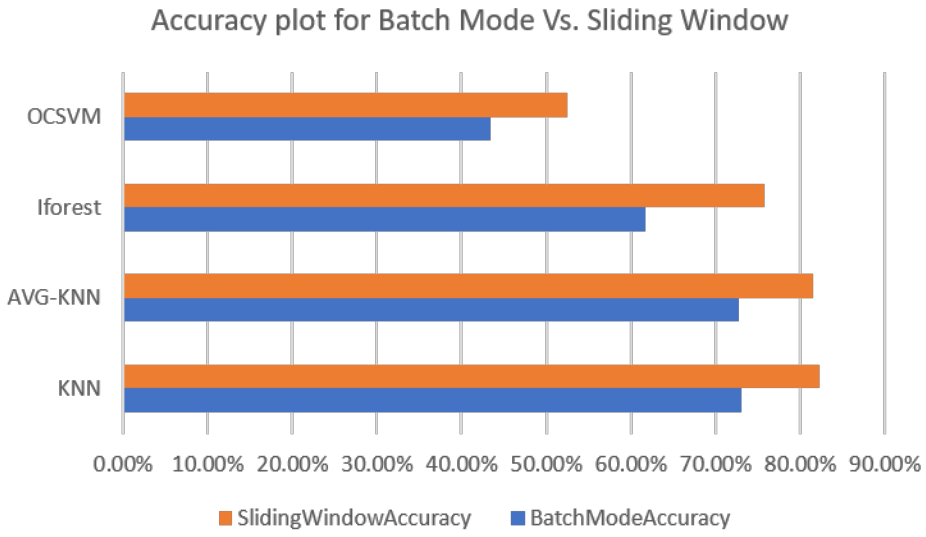

Fig. 6.5.2: PC keystroke dataset Accuracy plot for batch mode vs. sliding window

6.5.1.2 Android Keystroke Dataset-I

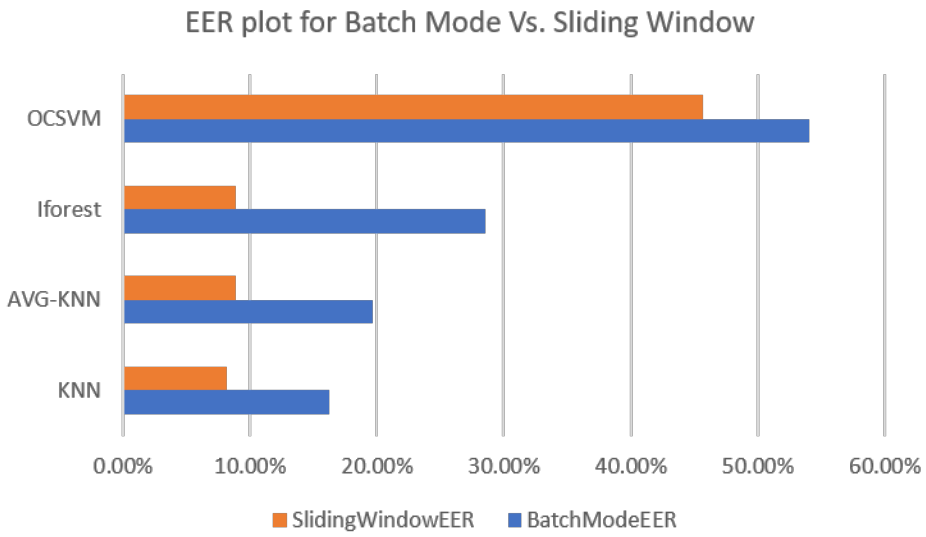

Fig. 6.5.3: Android keystroke dataset-I EER plot for batch mode vs. sliding window

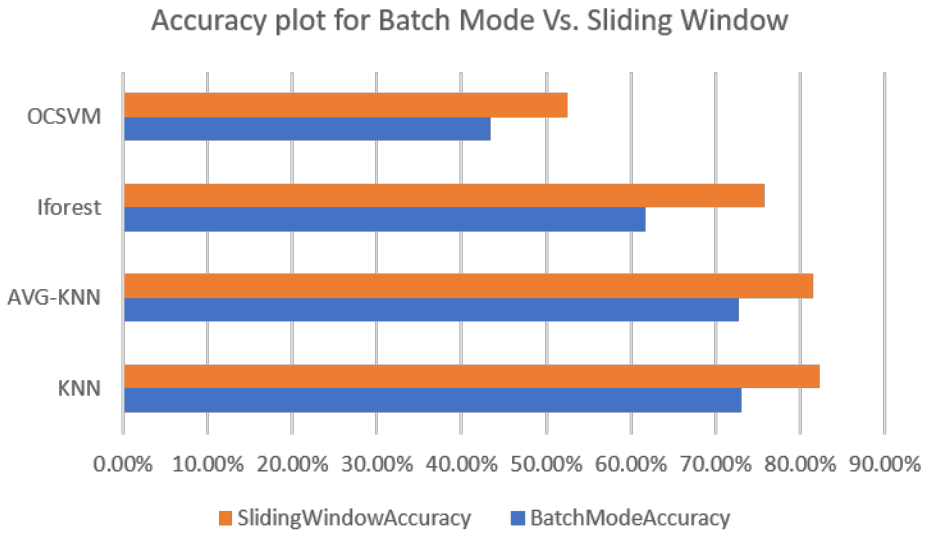

Fig. 6.5.4: Android keystroke dataset-I accuracy plot for batch mode vs. sliding window

6.5.1.3 Android Keystroke Dataset-II

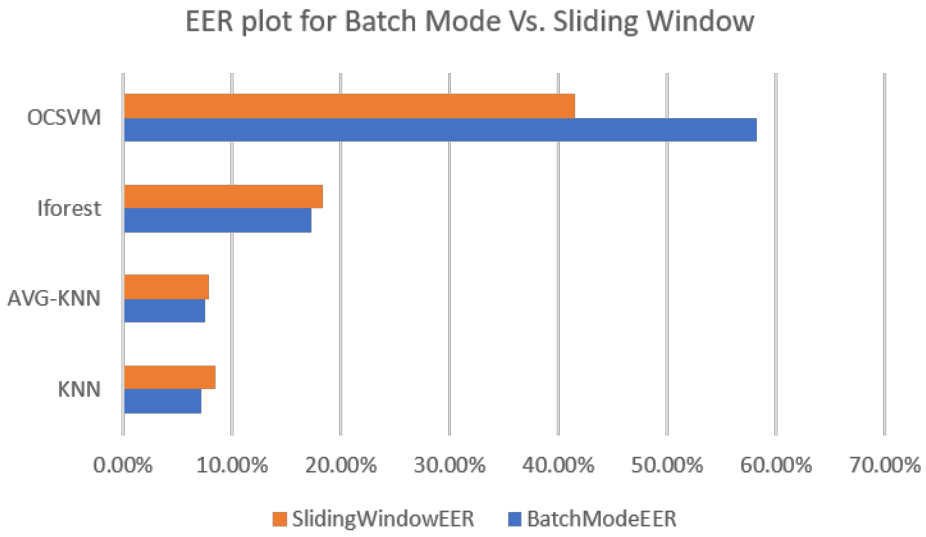

Fig. 6.5.5: Android keystroke dataset-II EER plot for batch mode vs. sliding window

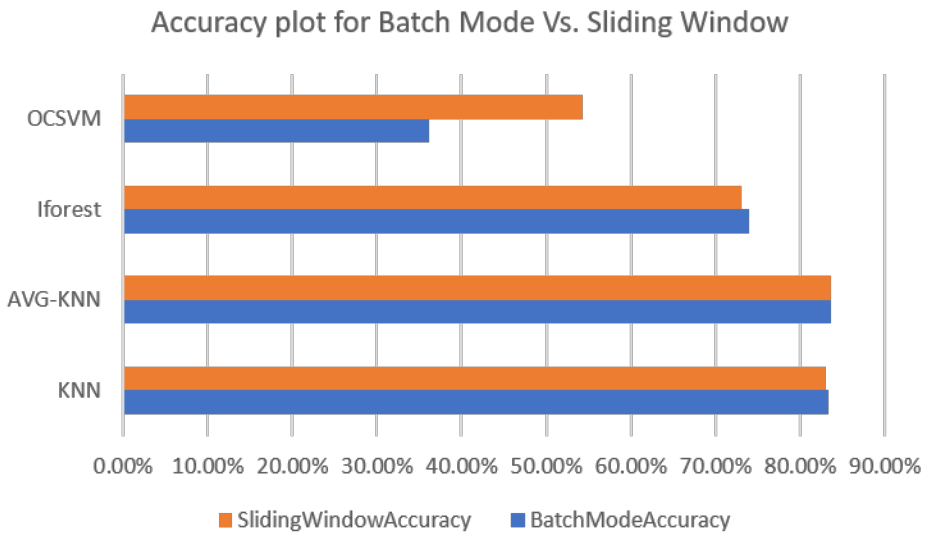

Fig. 6.5.6: Android keystroke dataset-II accuracy plot for batch mode vs. sliding window

- Batch mode update is useful only when the user is using the particular account regularly. If the user is using the account for say 2 times a month then to use batch mode, system has to wait for at least 6 months to update the user's profile. Because as per the given methodology the system should have new successfully logged in 10 samples to update the previous batch of 10 samples. In this case, the user's pattern of typing is likely to change before update and for that reason the knowledge of context affects the behavior of update strategy.

- Consider the same scenario for the sliding window approach, we are updating the user profile each time the user successfully logins. But, we are not entirely replacing the previously entered user's samples, we are just adding newly introduced sample at the end and remove the oldest sample present in the training set which enhances the system's behavior. Also, in this approach system is not required to wait for the 10 new successful samples. The approach is versatile in a sense that one can use it in any context no matter what it will keep updating the users' profiles.

- The batch mode simply replaces the previously entered 10 samples with the

newer 10 samples which may fail to catch the typing patterns correctly. For example; A user tries to login and the system trained on batch mode rejects the user, but there was a chance that the sample had a match with the previously entered samples which are replaced by the new batch.

- The sliding window moves one sample at a time which helps it to capture behaviour differences in a better way.

Based on results and the comparison we suggest the use of sliding window approach in practice to update the user profiles continuously.

6.5.2 Comparison Of Literature And 70-30 Train/Test Approach

In this section we demonstrate the comparisons of the three data set literature results and our 70-30 train/test approach's result for the three dataset.

6.5.2.1 Personal Computer Based Keystroke Dataset

Through our 70/30 approach we achieved lowest equal error rate of 0.02% and 0.03% from AVG-KNN and KNN respectively. The lowest EER mentioned in the data set literature [19] was 9.62% from Manhattan (scaled) anomaly detector.

Table 6.5.1: PC based keystroke dataset EER 70/30 train/test ratio

Anomaly Detector	**EER 70/30 Approach**
KNN	0.03%
AVG-KNN	0.02%
IForest	0.25%
One-class SVM	0.64%

6.5.2.2 Android Keystroke Dataset - I

We attained 0.7% EER through Isolation Forest anomaly detector. While, in the dataset literature [5] the lowest EER recorded was 1.31% from the Kmeans anomaly detector.

Table 6.5.2: Android Keystroke Dataset - I EER 70/30 train/test ratio

Anomaly Detector	**EER 70/30 Approach**
KNN	4.05%
AVG-KNN	3%
IForest	0.7%
One-class SVM	8.41%

6.5.2.3 Android Keystroke Dataset - II

We obtained lowest equal error rate of 3.25% from IForest anomaly detector. The lowest EER mentioned in the data set literature [4] was 12.9% from the Manhattan anomaly detector.

Table 6.5.3: Android Keystroke Dataset - II EER 70/30 train/test ratio

Anomaly Detector	**EER 70/30 Approach**
KNN	7.91%
AVG-KNN	4.16%
IForest	3.25%
One-class SVM	7.58%

6.5.3 Comparison Of Literature And Proposed Approach

The section shows the comparison of the three data set literature results with the proposed batch mode and sliding window approaches. The results as shown in the plot 6.5.10 and table 6.5.4 demonstrate that, even with the user profile update the sliding window approach is giving quite comparable results with the literature works done. For the PC based and android keystroke datset-II it outperform the literature results. The batch mode is also giving good results in case of the android keystroke dataset -II.

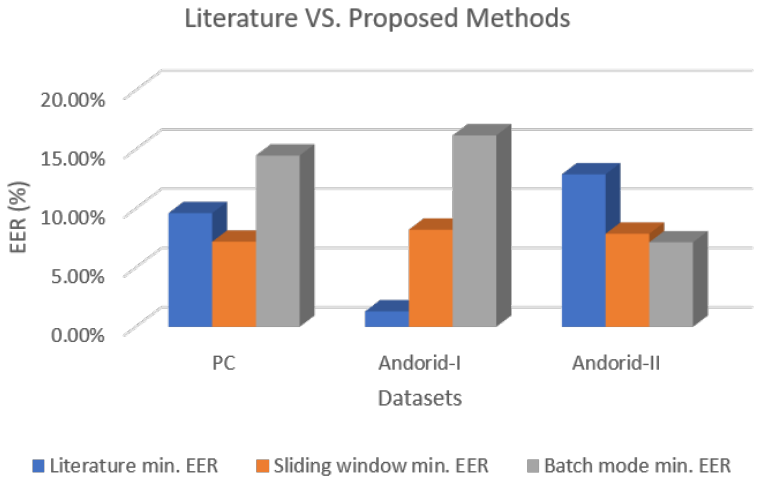

Fig. 6.5.7: Minimum EER comparison plot for each dataset

Table 6.5.4: Comparison of literature and proposed approach

Dataset	**Literature average EER**	**Sliding window average EER**	**Batch mode average EER**
PC based keystroke dataset	9.62%	**7.2%**	14.49%
Android keystroke dataset-I	1.31%	**8.22%**	16.19%
Android keystroke dataset-II	12.9%	**7.87%**	7.15%

CHAPTER 7

Conclusion and Future Work

7.1 Conclusion

In this thesis, we presented a systematic approach to design and build a reliable keystroke biometrics system. After debugging the performance of eight different multi-class classifiers using the biometrics zoos theory, we show that the conventional multi-class classifiers are not suitable for building a keystroke based access control system. Furthermore, we investigated the anomaly detection techniques in our work. The effects of feature selection and normalization on the anomaly detection performance are evaluated, demonstrating that it is beneficial to apply both feature selection and normalization to reduce the overall false acceptance rates. On the other hand, we prove that the feature selection is ineffective for the small subset of the data because the feature set and the feature importance vary when we change the size of the data. Additionally, we discuss the differences between production and research-based settings. Although there are several effective keystroke detection techniques proposed in literature works, which are said to be effective but they are not perfect for production-based settings. We have introduced two practical approaches that require only a few user samples to train the anomaly detectors. We also claim that the keystroke dynamics of a user changes over time, so there is a need for continuous update of the user profile to maintain the system's reliability. This thesis implements two approaches, namely, batch mode and sliding window, to continuously update the user's profile. Both approaches use a few samples from the user to process. From the two proposed methods sliding window is the most effective method because of its

dynamic update strategy. The batch mode replaces the previously entered batch of samples with the newer batch of samples, which may fail to catch the typing pattern differences correctly. The sliding window moves one sample at a time, which helps it capture behaviour differences better.

7.2 Future Work

Keystroke dynamics is a type of behavioral biometrics. As the keystroke changes over time other behavioral biometrics also evolve and change with time. In the future, we plan to carry on investigating other behavioral biometrics for their pattern changes and would like to apply our proposed methods to see if it is effective for other behavioral biometrics as well. During our experiments for anomaly detectors we have only considered four anomaly detectors. In future, it would be interesting to experiment with other types of anomaly detection techniques. Another interesting future work could be to develop a real time application which follows our suggested methodology. We also want to test our approaches against various malicious attacks to check for any kind of vulnerabilities. This may help us to enhance the quality of the system. Mhenni et al in [25] supports similar kind of thought as we do that we can not ask the user to enter the password 100 times to build his profile. They ask the user to enter a few samples like 10 and then use a genetic algorithm to reproduce synthetic data (90 samples) that looks similar to the 10 real samples they collected from the users to build the user's profile. We also aim to verify this methodology's effectiveness on our proposed approaches for keystroke dynamics as well as other behavioral biometrics.

REFERENCES

[1] Abilash R (Accessed: 2018-07-31). Applying random forest (classification) — machine learning algorithm from scratch with real datasets. `https://medium.com/@ar.ingenious/applying-random-forest-classification-machine-learning-algorithm-from-scratch-with-real-24ff198a1c57`.

[2] Al-Jarrah, M. M. (2012). An anomaly detector for keystroke dynamics based on medians vector proximity.

[3] Alghamdi, S. J. and Elrefaei, L. A. (2018). Dynamic authentication of smartphone users based on touchscreen gestures. *Arabian Journal for Science and Engineering*, 43:789–810.

[4] Antal, M. and Nemes, L. (2016). The mobikey keystroke dynamics password database: Benchmark results. In *Software Engineering Perspectives and Application in Intelligent Systems*, pages 35–46, Cham. Springer International Publishing.

[5] Antal, M., Szabo, L., and László, I. (2014). Keystroke dynamics on android platform.

[6] Ayush Pant (Accessed: 2019-01-22). Introduction to logistic regression. `https://towardsdatascience.com/introduction-to-logistic-regression-66248243c148`.

[7] Bashar, M. K., Chiaki, I., and Yoshida, H. (2016). Human identification from brain eeg signals using advanced machine learning method eeg-based biometrics. In

2016 IEEE EMBS Conference on Biomedical Engineering and Sciences (IECBES), pages 475–479.

[8] Bo, C., Zhang, L., Jung, T., Han, J., Li, X., and Wang, Y. (2014). Continuous user identification via touch and movement behavioral biometrics. In *2014 IEEE 33rd International Performance Computing and Communications Conference (IPCCC)*, pages 1–8.

[9] Boles, A. and Rad, P. (2017). Voice biometrics: Deep learning-based voiceprint authentication system. In *2017 12th System of Systems Engineering Conference (SoSE)*, pages 1–6.

[10] Clear IT Security (2020). Access control systems near me. `https://clearitsecurity.com/access-control-systems-near-me`.

[11] De Marsico, M., Petrosino, A., and Ricciardi, S. (2016). Iris recognition through machine learning techniques: A survey. *Pattern Recognition Letters*, 82:106 – 115. An insight on eye biometrics.

[12] DeCann, B. and Ross, A. (2013). Relating roc and cmc curves via the biometric menagerie. In *2013 IEEE Sixth International Conference on Biometrics: Theory, Applications and Systems (BTAS)*, pages 1–8.

[13] Doddington, G. R., Liggett, W., Martin, A. F., Przybocki, M. A., and Reynolds, D. A. (1998). Sheep, goats, lambs and wolves: a statistical analysis of speaker performance in the nist 1998 speaker recognition evaluation. In *ICSLP*.

[14] Expert System Team (Accessed: 2020-5-30). What is machine learning? a definition. `https://expertsystem.com/machine-learning-definition/`.

[15] Gowthamy Vaseekaran (Accessed: 2018-9-28). Machine learning: Supervised learning vs unsupervised learning. `https://medium.com/@gowthamy/machine-learning-supervised-learning-vs-unsupervised-learning-f1658e12a780`.

[16] He Zheng, Liao Ni, Ran Xian, Shilei Liu, and Wenxin Li (2015). Bmdt: An optimized method for biometric menagerie detection. In *2015 IEEE 7th International Conference on Biometrics Theory, Applications and Systems (BTAS)*, pages 1–8.

[17] Howard, J. J. and Etter, D. (2013). The effect of ethnicity, gender, eye color and wavelength on the biometric menagerie. In *2013 IEEE International Conference on Technologies for Homeland Security (HST)*, pages 627–632.

[18] Ivannikova, E., David, G., and Hämäläinen, T. (2017). Anomaly detection approach to keystroke dynamics based user authentication. In *2017 IEEE Symposium on Computers and Communications (ISCC)*, pages 885–889.

[19] Killourhy, K. and Maxion, R. (2009). Comparing anomaly-detection algorithms for keystroke dynamics. pages 125 – 134.

[20] Killourhy, K. and Maxion, R. (2010). Why did my detector do that?! In Jha, S., Sommer, R., and Kreibich, C., editors, *Recent Advances in Intrusion Detection*, pages 256–276, Berlin, Heidelberg. Springer Berlin Heidelberg.

[21] Kirchgasser, S. and Uhl, A. (2016). Biometric menagerie in time-span separated fingerprint data. *2016 International Conference of the Biometrics Special Interest Group (BIOSIG)*, pages 1–7.

[22] Krishnamoorthy, S., Rueda, L., Saad, S., and Elmiligi, H. (2018). Identification of user behavioral biometrics for authentication using keystroke dynamics and machine learning. In *ICBEA '18*.

[23] Krüger, F. (2016). *Activity, Context, and Plan Recognition with Computational Causal Behaviour Models*. PhD thesis.

[24] LAKSHAY ARORA (Accessed: 2019-02-14). An awesome tutorial to learn outlier detection in python usig pyod library. `https://www.analyticsvidhya.com/blog/2019/02/outlier-detection-python-pyod/`.

[25] Mhenni, A., Cherrier, E., Rosenberger, C., and ESSOUKRI BEN AMARA, N. (2018a). Adaptive biometric strategy using doddington zoo classification of user's keystroke dynamics.

[26] Mhenni, A., Cherrier, E., Rosenberger, C., and ESSOUKRI BEN AMARA, N. (2018b). User dependent template update for keystroke dynamics recognition.

[27] Mike Chapple, J. M. and Gibson, D. (2018). *CISSP: Certified Information Systems Security Professional*. Sybex, A Wiley Brand, United states of America.

[28] Miluzzo, E., Varshavsky, A., Balakrishnan, S., and Choudhury, R. R. (2012). Tapprints: your finger taps have fingerprints. In *MobiSys '12*.

[29] mlxtend (2014). Neural network - multilayer perceptron. `http://rasbt.github.io/mlxtend/user_guide/classifier/MultiLayerPerceptron/`.

[30] Monaco, J. V. (2016). Robust keystroke biometric anomaly detection. *CoRR*, abs/1606.09075.

[31] Neal, T. J. and Woodard, D. L. (2019). You are not acting like yourself: A study on soft biometric classification, person identification, and mobile device use. *IEEE Transactions on Biometrics, Behavior, and Identity Science*, 1(2):109–122.

[32] Nicholas M. Orlans John D. Woodward Jr., P. (2002). *Biometrics*. Osborne Networking Series. Mcgraw-hill.

[33] Paone, J. and Flynn, P. J. (2011). On the consistency of the biometric menagerie for irises and iris matchers. In *Proceedings of the 2011 IEEE International Workshop on Information Forensics and Security*, WIFS '11, page 1–6, USA. IEEE Computer Society.

[34] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

[35] plurilock (2019). Behavioral biometrics. https://www.plurilock.com/behavioral-biometrics-guide/1-what-is-behavioral-biometrics/.

[36] Poh, N., Bengio, S., and Ross, A. (2006). Revisiting doddington's zoo: A systematic method to assess user-dependent variabilities.

[37] Popescu-Bodorin, N., Balas, V., and Motoc, I. (2012). The biometric menagerie - a fuzzy and inconsistent concept. *5 th Int. Conf. on Soft Computing and Applications (Szeged, HU), 22-24 Aug 2012*.

[38] Prateek Sharma (Accessed: 2019-07-21). Decoding the confusion matrix. https://towardsdatascience.com/decoding-the-confusion-matrix-bb4801decbb.

[39] Pushkar Mandot (Accessed: 2017-08-17). What is lightgbm, how to implement it? how to fine tune the parameters? https://medium.com/@pushkarmandot/https-medium-com-pushkarmandot-what-is-lightgbm-how-to-implement-it-how-to-fine-tune-the-parameters-60347819b7fc.

[40] Ramu, T., Suthendran, K., and Arivoli, T. (2019). Machine learning based soft biometrics for enhanced keystroke recognition system. *Multimedia Tools and Applications*, pages 1–17.

[41] Rohith Gandhi (Accessed: 2018-06-18). Support vector machine — introduction to machine learning algorithms. https://towardsdatascience.com/support-vector-machine-introduction-to-machine-learning-algorithms-934a444fca47.

[42] Ross, A., Rattani, A., and Tistarelli, M. (2009). Exploiting the doddington zoo effect in biometric fusion. pages 1 – 7.

[43] Schnitzer, D., Flexer, A., and Schlüter, J. (2013). The relation of hubs to the doddington zoo in speaker verification. In *21st European Signal Processing Conference (EUSIPCO 2013)*, pages 1–5.

[44] scikit-learn (2011). Decision trees. `https://scikit-learn.org/stable/modules/tree.html`.

[45] Sharifah, M., Asma, S., Admad Faudzi, M., and Md. Anwar, R. (2010). Analysis of 'goat' within user population of an offline signature biometrics. In *10th International Conference on Information Sciences, Signal Processing and their Applications, ISSPA 2010*, 10th International Conference on Information Sciences, Signal Processing and their Applications, ISSPA 2010, pages 765–769. 10th International Conference on Information Sciences, Signal Processing and their Applications, ISSPA 2010 ; Conference date: 10-05-2010 Through 13-05-2010.

[46] Spotfire Blogging Team (Accessed: 2020-03-16). Top 10 methods for outlier detection. `https://www.tibco.com/blog/2020/03/16/top-10-methods-for-outlier-detection-in-spotfire/`.

[47] Sundararajan, K., Neal, T. J., and Woodard, D. L. (2018). Style signatures to combat biometric menagerie in stylometry. *2018 International Conference on Biometrics (ICB)*, pages 263–269.

[48] Sundararajan, K. and Woodard, D. L. (2018). Deep learning for biometrics: A survey. *ACM Comput. Surv.*, 51:65:1–65:34.

[49] Teli, M., Givens, G. H., Phillips, P., Draper, B. A., Beveridge, J., and Bolme, D. S. (2011). Biometric zoos: Theory and experimental evidence. In *Biometrics, International Joint Conference on*, pages 1–8, Los Alamitos, CA, USA. IEEE Computer Society.

[50] tutorialspoint (2019). Knn algorithm - finding nearest neighbors. `https://www.tutorialspoint.com/machine_learning_with_python/machine_learning_with_python_knn_algorithm_finding_nearest_neighbors.htm`.

[51] Wang, Z., Serwadda, A., Balagani, K. S., and Phoha, V. V. (2012). Transforming animals in a cyber-behavioral biometric menagerie with frog-boiling attacks. In

2012 IEEE Fifth International Conference on Biometrics: Theory, Applications and Systems (BTAS), pages 289–296.

[52] Will Koehrsen (Accessed: 2018-03-18). Beyond accuracy: Precision and recall. `https://towardsdatascience.com/beyond-accuracy-precision-and-recall-3da06bea9f6c`.

[53] Yager, N. and Dunstone, T. (2007). Worms, chameleons, phantoms and doves: New additions to the biometric menagerie. In *2007 IEEE Workshop on Automatic Identification Advanced Technologies*, pages 1–6.

[54] Yager, N. and Dunstone, T. (2010). The biometric menagerie. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(2):220–230.

[55] Yang, W., Wang, S., Hu, J., Zheng, G., Chaudhry, J., Adi, E., and Valli, C. (2018). Securing mobile healthcare data: A smart card based cancelable finger-vein bio-cryptosystem. *IEEE Access*, 6:36939–36947.

[56] Yaser Sakkaf (Accessed: 2020-03-31). Decision trees: Id3 algorithm explained. `https://towardsdatascience.com/decision-trees-for-classification-id3-algorithm-explained-89df76e72df1`.

VITA AUCTORIS

NAME:	Anjali Shah
PLACE OF BIRTH:	Ahmedabad, Gujarat, India
YEAR OF BIRTH:	1994
EDUCATION:	Gujarat Technological University, Bachelor's of Engineering in Computer Engineering, Gandhinagar, Gujarat, 2015
	University of Windsor, M.Sc in Computer Science, Windsor, Ontario, 2020