MITIGATING INSIDER THREAT BY PROFILING USERS BASED ON MOUSE USAGE PATTERN

by

Metehan Yıldırım

B.S., Electrical & Electronics Engineering, Boğaziçi University, 2019

Submitted to the Institute for Graduate Studies in
Science and Engineering in partial fulfillment of
the requirements for the degree of
Master of Science

Graduate Program in Electrical Electronics Engineering

Boğaziçi University

2021

ACKNOWLEDGEMENTS

This work is supported by the Scientific and Technological Research Council of Turkey (TUBITAK), under Cloud-Based Privileged Access Management System Project, Project No. 117R030 and 2210-C Priority Areas Graduate Scholarship Program.

I would like to thank my thesis advisor, Prof. Emin Anarım for his patience and contributions during this thesis. He was always been a great mentor. I also would like to thank Prof. Mutlu Koca for his great mentorship.

Çağatay Ateş, Arjen Aykan Kılıç, Süleyman Özdel, Begün Ünal have great contributions to this thesis. I appreciate their friendship and assistance.

Finally, I would like to thank my father Sebahattin Yıldırım and my mother Emine Sever Yıldırım for encouraging me with their best wishes.

ABSTRACT

MITIGATING INSIDER THREAT BY PROFILING USERS BASED ON MOUSE USAGE PATTERN

Authentication is the process of proving who a user or program is when accessing a system. Three-factor authentication with something you know (password), something you are (biometric), and something you own (token) is the best way to prevent unauthorized access. One of these steps, biometric authentication, performs this verification by checking distinctive biological or behavioral characteristics. Behavioral biometrics is the field of study concerned with the measurement of uniquely descriptive and measurable patterns in human activity. Behaviors that can be used in this study area are processes that are subconsciously controlled by measurable patterns such as keyboard use, gait, mouse use. A well-perceived pattern allows us to constantly check the user's authentication.

In this study, the detection of the mouse usage patterns of the users for the company and the inhibition of the insider attack vectors are studied. The studies on the described biometric model are carried out on the Balabit Mouse Dynamics and Bogazici Mouse Dynamics datasets. This area has been contributed by feature studies in the frequency domain, the use of machine learning methods based on use cases, and the procedures of fusing information from various detectors. We presented our findings as a security layer by designing how the proposed procedures could work as a whole as a service.

ÖZET

KULLANICILARIN FARE KULLANIM MODELİNE GÖRE PROFİLİNİ OLUŞTURARAK İÇSEL TEHDİTLERİ AZALTMA

Kimlik doğrulama, bir kullanıcının veya programın bir sisteme erişirken kim olduğunu kanıtlama işlemidir. Bildiğiniz bir şey (şifre), olduğunuz bir şey (biyometrik) ve sahip olduğunuz bir şey (belirteç) içeren üç faktörlü kimlik doğrulama, yetkisiz erişimi önlemenin en iyi yoludur. Bu basamaklardan biyometrik kimlik doğrulama ayırt edici biyolojik veya davranışsal özellikleri kullanır. Davranışsal biyometri ise, insan faaliyetlerinde benzersiz bir şekilde tanımlayıcı ve ölçülebilir örüntülerin ölçümü ile ilgili çalışma alanıdır. Bu çalışma alanında kullanılabilecek davranışlar klavye kullanımı, yürüyüş biçimi, fare kullanımı gibi bilinçaltında ölçülebilir örüntüler ile kontrol edilen işlemlerdir. Taklit etmesi zor olduğu için, iyi algılanmış bir örüntü kullanıcının kimlik doğrulamasını sağlamaktadır.

Kullanıcıların fare kullanım örüntü tespiti ile içsel atak vektörlerinin engellenmesi üzerine çalışılmıştır. Tez çıktıları Balabit ve Bogazici Mouse Dynamics veri kümeleri üzerinde gerçeklenmiştir. Bu alana, frekans uzayında öznitelik çalışmaları, farklı yapay öğrenme yöntemlerinin kullanılması ve tespit edicilerden gelen bilgileri füzyon etme yordamları ile katkı verilmiştir. Yordamların bir bütün halinde servis olarak nasıl çalışabileceğini tasarlayarak, bir güvenlik katmanı olarak bulgularımızı sunduk.

TABLE OF CONTENTS

ACKNOWLEDGEMENTS . . . iii
ABSTRACT . . . iv
ÖZET . . . v
LIST OF FIGURES . . . ix
LIST OF TABLES . . . xi
LIST OF ACRONYMS/ABBREVIATIONS . . . xii
1. INTRODUCTION . . . 1
1.1. Authentication . . . 1
1.1.1. Two-factor Authentication . . . 1
1.2. Biometric Systems . . . 2
1.3. Behavioral Biometry . . . 4
1.4. Literature View . . . 5
2. BIOMETRIC SYSTEM MODEL . . . 9
2.1. The Bogazici Mouse Dynamics dataset . . . 10
2.1.1. Experimental Design, Materials and Methods . . . 12
2.2. Balabit Mouse Dynamics dataset . . . 14
2.3. Prevented Attack Models and Use Cases . . . 15
2.3.1. Machine Learning Model Types to Build Biometric Models . . . 16
2.3.2. Insider Threat . . . 17
2.3.2.1. Focused Insider Threat Scenarios . . . 18
2.3.2.2. Insider Threat Modelling . . . 20
3. FEATURE EXTRACTION . . . 22
3.1. Data Preprocessing . . . 23
3.1.1. Action Segmentation . . . 24
3.1.1.1. Movement Actions . . . 24
3.1.1.2. Click Actions . . . 25
3.2. Feature Signals . . . 26
3.2.1. Angle of movement . . . 26

3.2.2. Curvature-Based Feature Signals . . . 27
3.2.3. Velocity . . . 28
3.2.4. Acceleration . . . 28
3.2.5. Jerk . . . 29
3.2.6. Angular Velocity . . . 29
3.3. Fixed-sized Vectorization of Feature Signals . . . 29
3.3.1. Time Domain Statistical Analysis . . . 30
3.3.2. Frequency Domain Statistical Analysis . . . 30
3.4. Eigenspace Transform with Kernels . . . 33
3.4.1. Kernel Principal Component Analysis . . . 34
3.4.1.1. Sigmoid . . . 35
3.4.1.2. Polynomial . . . 36
3.4.1.3. Radial basis function (RBF) . . . 36
4. PROPOSED VERIFICATION MODEL . . . 38
4.1. Learning . . . 38
4.1.1. Supervised Learning . . . 38
4.1.1.1. Gradient Boosting Machines . . . 39
4.1.1.2. Avoiding Overfitting . . . 42
4.1.2. Self-supervised Learning . . . 43
4.1.2.1. One Class Support Vector Machines (OCSVM) . . . 44
4.1.2.2. Distance-Based Methods . . . 45
4.2. Decision Making . . . 46
4.2.1. Weight Selection . . . 48
5. EXPERIMENTS AND RESULTS . . . 51
5.1. Experiment Setup . . . 51
5.1.1. Evaluation Metrics . . . 52
5.2. Comparison of Proposed Methods . . . 52
5.2.1. Supervised Learning . . . 52
5.2.1.1. Comparison with the simple averaging aggregation . . . 54
5.2.1.2. Contribution of new features . . . 55
5.2.2. Self-supervised Learning . . . 55

5.2.2.1. Performance of the classifiers 57
5.2.2.2. Contribution of the Kernel PCA 58
5.3. Discussion 58
6. CONCLUSION 61
REFERENCES 63
APPENDIX A: Area Under ROC Calculation 68

LIST OF FIGURES

Figure 2.1. Biometric System Model. 10

Figure 2.2. Supervised Biometry Model. 18

Figure 2.3. Self-supervised Biometry Model. 20

Figure 3.1. Segmentated MM action. 23

Figure 3.2. Correlation Matrix of Time and Frequecy Features. 33

Figure 4.1. Gradient Boosting Algorithm. 39

Figure 4.2. Supervised Learning Scheme. 42

Figure 4.3. Verification Scheme. 50

Figure 5.1. Contribution of new features on performance in terms of AUC. . . 54

Figure 5.2. Contribution of new features on performance in terms of EER. . . 55

Figure 5.3. Contribution of the Kernel PCA on performance in terms of AUC. 57

Figure A.1. ROC with 10 Partition. 68

Figure A.2. ROC with 200 Partition. 69

Figure A.3. The trapezoidal calculation of Area Under ROC with 10 Partition. 71

Figure A.4. The trapezoidal calculation of Area Under ROC with 10 Partition. 71

LIST OF TABLES

Table 1.1. Existing works. 5

Table 2.1. Bogazici dataset mouse usage data description. 11

Table 2.2. Balabit dataset mouse usage data description. 14

Table 3.1. A partial raw mouse usage data. 22

Table 3.2. Mouse feature signals used in methodology. 25

Table 3.3. Features are used to represent a movement action. 29

Table 4.1. Aggregation Types and Descriptions. 48

Table 5.1. Performances of aggregation methods according to verification time on the Balabit dataset. 53

Table 5.2. Performances of aggregation methods according to verification time on the Bogazici dataset. 53

Table 5.3. Performances of the methods on the Balabit dataset. 56

Table 5.4. Performances of the methods on the Bogazici dataset. 57

LIST OF ACRONYMS/ABBREVIATIONS

1D-CNN	One Dimensional Convolutional Neural Network Model
2D-CNN	Two Dimensional Convolutional Neural Network Model
AUC	Area under the ROC Curve
FAR	False Acceptance Rate
FFT	Fast Fourier Transform
FRR	False Rejection Rate
GBM	Gradient Boosting Machines
KNN	K-Nearest Neighbors
KPCA	Kernel Principal Component Analysis
LOF	Local Outlier Factor
LSTM	Long short-term memory
MLP	Multi Layer Perceptron
OCSVM	One Class Support Vector Machines
PCA	Principal Component Analysis
PSD	Power Spectral Density
RBF	Radial Basis Function
RDP	Remote Desktop Connection
ROC	Receiver Operating Characteristic
SVM	Support Vector Machines

1. INTRODUCTION

1.1. Authentication

Authentication is the process of proving who a user or program is when accessing a system. Authentication technology provides access control for systems by checking whether a user's credentials match the credentials of authorized users in the database or a data authentication server. In today's systems, users usually use the passwords they set when registering in the system for authentication. Many companies use authentication to verify users who log into their websites. Without the right security measures, user data that should be confidential, such as credit and debit card numbers, can fall into the hands of hackers.

1.1.1. Two-factor Authentication

The security system that we continue to use in many of our accounts only with username and password is included in the literature as single-factor authentication. In this one-step verification system, no matter how unpredictable the password is, it can be easily obtained with software such as viruses, worms, keyloggers, spyware, trojans developed by hackers. In two-step authentication, this malicious software are inadequate. The patent [1] is obtained for the purpose of generating a device that would generate unpredictable codes at that time and includes a computational mechanism for comparing an unpredictable first code with an unpredictable second code. Today, besides these devices, which are still called "tokens", they can now be evaluated in this context on smartphones.

Bank accounts are constantly stolen by hackers in the country until then, and bank customers are suffering financial damage and falling into the situation. Within the scope of the published communique, a two-step identity verification system was implemented with the requirement of "One-Time Password" when logging into inter-

net banking. Now, users can perform more secure transactions with the second step password or biometric trait. This becomes an industry standard. Still, these security cautions are subject to steal or misuse.

1.2. Biometric Systems

Biometric systems; stand out as technologies that make sense of individuals by analyzing their unique biological, physical, and behavioral data. Biometric readers using options such as DNA data, fingerprint identification, retina scanning, or face recognition; It has replaced traditional applications such as personnel identification cards, keys, passwords, magnetic cards, and so on. These systems, which have advanced technology behind them, have started to be preferred by many businesses with the development of internet technologies and data-based company understanding. These technologies also called biometric security systems; also provide a great advantage in terms of the security of personal data. Biometric security infrastructure is primarily used in modern buildings, workplaces, public areas requiring high security, and similar locations. Biometric encryption solutions, which provide a great advantage over key, password, and card systems, which are disadvantageous compared to biometric systems in terms of application and security, are easily integrated with the security systems of the buildings. In this way, high safety standards are provided in the area of use. The working principle of biometric security systems is based on scanning and reading data on the face, voice, fingerprint, retina, palm, vein, which are biometric and personally evaluated. During the system setup phase, each user's biometric data, also called enrollment, is taken to be defined in the database. These data received from individuals are authorized to be used in different parts of the system. Thanks to these authorizations, people have access to the regions within their jurisdiction over the biometric system. This application is preferred in areas requiring high security; It also has high-level integration options with other infrastructures such as security camera systems and perimeter security systems. A biometric security system stands out as an infrastructure with versatile usage options and is structured with advanced technological options. In this direction, the system consists of parts that have more complex

features and technologies. Each of the mentioned parts strengthens the structure of the system with its own functionality.

Biometric systems are divided into two basic groups as physical and behavioral. Physical biometrics include iris, face, fingerprint, retina, hand recognition. Special technologies used within the physical biometrics system can be listed as follows:

Face Recognition Systems. This technology, also called face recognition access control system, stands out as one of the technologies that users are most interested in in the title of biometric security applications. Face recognition systems, thanks to advanced sensors and built-in cameras, allow the faces defined in the database to be authorized. In this way, people authorized in the desired area and high-security areas have the initiative to act flexibly and to pass.

Eye and Retina Scanning Systems. Thanks to special software technologies used in eye and retina scanning systems, it enables the retina found in each person to be scanned and matched with the information in the database. Within the scope of this process, authorized persons within the limits of their authority after scanning; have the opportunity of fast and flexible movement within the scope of the passage, control, and security.

Fingerprint Recognition Devices. Fingerprint identification devices are known as one of the fastest solutions within the scope of biometric access control infrastructures and also one of the most technological solutions. Fingerprint technologies, which are used extensively in high-security areas, are preferred especially in detective and military security standards within the scope of criminal security activities. This system, which has usage options integrated with technologies such as eye scanning systems and face recognition devices, maximizes the safety standards in the field of use. By integrating with perimeter security infrastructure and security camera systems, it promises a high-

level protection experience within the desired area boundaries.

In the following, the details about behavioral biometry are discussed.

1.3. Behavioral Biometry

Behavioral biometry is based on the behavioral characteristics of the individual while dealing with systems; usage patterns, voice and handwriting, wet signature, and so on. There are several advantages to using behavioral features. A strong argument for behavioral biometrics is that it can aid in authentication without requiring additional hardware. As a result, besides the fact that behavioral authentication is less expensive than using physiological biometrics, the user-side access system is easier to use because it does not involve the use of additional hardware. Passive verification is very convenient in the system where behavioral biometric features are used. Active authentication requires dealing with a device and entering one or more pieces of valid information or some other kind of interaction. Since users use so many applications or services these days, it can be considered tedious and frustrating that this type of authentication is required for each application/service. As a result, users may prefer less secure options for system access. Passive, also known as continuous verification, work in the system background without sending a direct input request to the user. After the user logs into the system, it starts to keep track of the features defined in the system. When enough data is collected, a user model is created using statistical methods or machine learning, and the model can be updated at certain time intervals or when certain conditions are met. While the user is operating in the system, data is continuously collected and the resulting behavior model is compared with the user model. When abnormal behavior is detected, the security procedure defined in the system is applied. While high-quality input is required to collect physical traits, this sensitivity is not essential for behavioral traits. Because the way of expressing changes over time is followed rather than a static measurement. For example, images are taken from security cameras, especially those used for surveillance rather than access control, are not of sufficient quality for facial recognition authentication. However, it may be

sufficient to obtain the silhouette of the human body movement parts in determining the gait profile.

In this manner, this thesis is focusing on behavioral traits while continuous authentication. The user behavior selected is the mouse usage behavior that allows the user to interact with the computer. Thus, we open the way for users to be authorized in online environments. We are developing this authorization module as an additional security module to the 2-factor authentication algorithm. In the literature, systems in which a biometric feature is questioned are called "single model biometric systems", while systems in which more than one biometric feature is questioned are called "multiple model biometric systems". For example; A system where voice and face recognition analysis is performed together is defined as a multi-model biometric system. In other words, biometric models can be combined to have a greater effect even if they do not have enough performance to use along. Therefore, authentication with mouse usage behavior will work as the second factor in 2-factor authentication systems or assist with 2-factor authorization systems.

1.4. Literature View

Table 1.1. Existing works.

Paper	**Features**	**Results**	**Remarks**
Antal and El [2]	Click, Movement and Drag	EER: 18.1, AUC: 92.2%%	20 actions (210 seconds)
Chong et al. [3]	Movement	EER: 10%, AUC: 96%	390 Events (+54 seconds)
Chong et al. [4]	Movement	EER: 10%, AUC: 96%	390 Events (+54 seconds)
Feher et al. [5]	Click, Movement and Drag	EER: 8.53%	30 actions (280 seconds)
Zheng and Wang. [6]	Click and Movement	EER: 1.3%	20 clicks (37.73 min)
Zheng et al. [7]	Click and Movement	EER: 1.3%	20 clicks (37.73 min)
Ahmed and Traore [8]	Click and Movement and Drag	EER: 2.46%	2000 actions (13.55 min)

In the field of biometric authentication, important studies based on the mouse device started in 2003. The work by [9] provides a new behavioral biometric approach based on human-computer interaction in this research. They created a system that employs a pointing device to collect human interaction and then uses that data to

validate an individual's identification. They built a sequential classifier that evaluates user interaction using statistical pattern recognition techniques, according to which the user's identity is regarded authentic if a preset accuracy threshold is met, and the person is labeled as an imposter otherwise. Their findings reveal that when a person interacts with a computer using a pointing device, behavioral information with discriminating power is generated, which may be used to verify identification. Their findings are also dependent on the preset tasks, like solving a puzzle or dragging the mouse to the preset locations.

The work by [8] develop a novel kind of behavioral biometrics based on mouse dynamics in their 2007 paper. They create a method for employing artificial neural networks to simulate behavioral features derived from collected data. They also propose a detector design and implementation that covers all aspects of the biometric data flow, including the detection process. They offer a new method of feature extraction that combines sequential cursor movement into higher-level mouse activities while still relying on predefined tasks. As a result, the signal processing work in the field has begun.

The approach by [10] breaks raw data into mouse curves; length, curvature, and inflection points of the curve are employed as major features; a reference signature is generated by producing histograms from the curve characteristics of multiple curvatures. The Euclidean distance between the reference signature and the mouse activity recorded at the moment of authentication is used to verify the authenticity of the signature.

The work by [11] critiqued [10]'s and [8]'s methods and noted some of their shortcomings. However, it's not apparent if the system is looking for variations in mouse behavior or variations in working conditions, such as tasks being completed.

Another research that takes advantage of mouse movements is [12]. In this study, the usage-independent (without preset tasks) features are discovered and used. To

solve this problem, they must find characteristics that can accurately reflect the user's behavior patterns independent of the job or environment in which she/he is working. They concentrated on the features of the curves created from successive mouse positions during typical mouse movements and achieved a poorer performance than the previously stated works.

From the perspective of signal processing, the most mature study is [5]. They demonstrate an innovative method for continuously validating users based on their mouse interaction characteristics. They have three main contributions: first, user verification is determined from the categorization results of individual mouse actions, rather than from aggregated mouse activity. Second, they suggest a hierarchical classification of mouse activities that may be used to infer attributes. Thirdly, they offer new characteristics for defining mouse activity that integrate previously presented qualities. This work also is not including a public dataset.

In this field, for repeatability, the Balabit Data Set acquired in [13] is used in many studies, including this thesis scope. The Balabit data set was released in 2016 for a data science competition, and being one of the few publicly available ones, it may be regarded as the first adequate one. The work by [2] has published research evaluating the performance of this data set for imposter detection. Due to the presence of extremely brief test sessions, this data set presents a challenge. After segmenting the raw data into mouse motion, point and click, and drag and drop mouse operations, many characteristics were recovered. Evaluations were conducted using both action- and set-based criteria. On the test section of the data set, the collection of actions-based evaluations obtains an AUC of 0.92. However, when the same sort of evaluation was performed on the training portion of the data set, the maximum AUC of 1 was obtained with only 13 activities. This work has quite similar to [5]' work. The main difference is the dataset. Then, a lot more works come after this dataset.

With emerging big data analytics and machine learning concepts, many methods are proposed for mouse dynamics. The work in [3] obtains a 96.1 percent AUC and

a 10.1 percent EER using 2D-Convolutional Neural Networks (2D- CNN) and Long Short-Term Memory (LSTM) approaches to encode and categorize motions as pictures. The work by [4] is a continuation of a previous paper. Both take into account solely mouse movement and drag-and-drop motions. Other activities, such as mouse button clicks and scrolling with the mouse, are ignored. The 1D-CNN, LSTM, and CNN-LSTM approaches utilized in this work all need the collection and evaluation of a predetermined number of 130 events. This study determined the performance rates for a single picture by creating one after every 130 mouse movement occurrences. However, it is demonstrated that by integrating these pictures sequentially, a superior performance may be attained. While each image comprises an average of 18 seconds of mouse movement data, calculating the total time required for verification is challenging owing to the methods employed to separate the data for shaping the images and a lack of information on the average gap durations between formed images. As a result, the aggregation performances listed in Table 1.1 are quite optimistic in terms of time. Additionally, the document makes a point of highlighting the target user's screen resolution. While performing session-based estimations using 2D-CNN and SVM, the paper flags the whole test session as illegal if any out-of-resolution sequence from the same session is encountered. This appears to be an overfit to the Balabit Mouse Dynamics Challenge Dataset. Table 1.1 summaries the papers with dynamic authentication.

2. BIOMETRIC SYSTEM MODEL

Biometrics is the science that studies the physical and behavioral characteristics that can distinguish people from each other. Due to its distinctive feature, it is frequently preferred in verification applications. Biometric systems; classic password control, card pass, etc. are very different from the techniques in terms of concept. Because biometric features are of a nature that cannot be changed by the person or transferred to another person. From this point on, biometrics is a technique that greatly increases the level of security. Biometric systems can be designed in different ways in accordance with the general lines of the application. However, a biometric system basically consists of five subsystems. [14] These are: data collection, transmission, feature extraction, modeling and matching, respectively. Data collection is the recording of the relevant biometric data of the users to be included in the system. The next stage is the transmission, in which biometric data, mouse usage records, are digitally transferred to the system by human-computer interaction. Then the transmitted data will be processed according to the algorithms we propose. At this stage, digital signals obtained by data acquisition and transmission are reduced to their feature vectors using various signal processing techniques. Thus, the input mouse data has been simplified and unnecessary parts have been removed. In the modeling phase, attributes of different people are stored in the system by creating different biometric models. This part, also defined as the storage stage, is the stage where the basic database of system users is created. Finally, the biometric data collected at the time of request of the person requesting authorization to the system is compared with the models in the database. In case of matching, the authorization request will be accepted, in case of a mismatch, the action determined according to the type of application will be performed. The summary of the specified biometric model is shown in Figure 2.1.

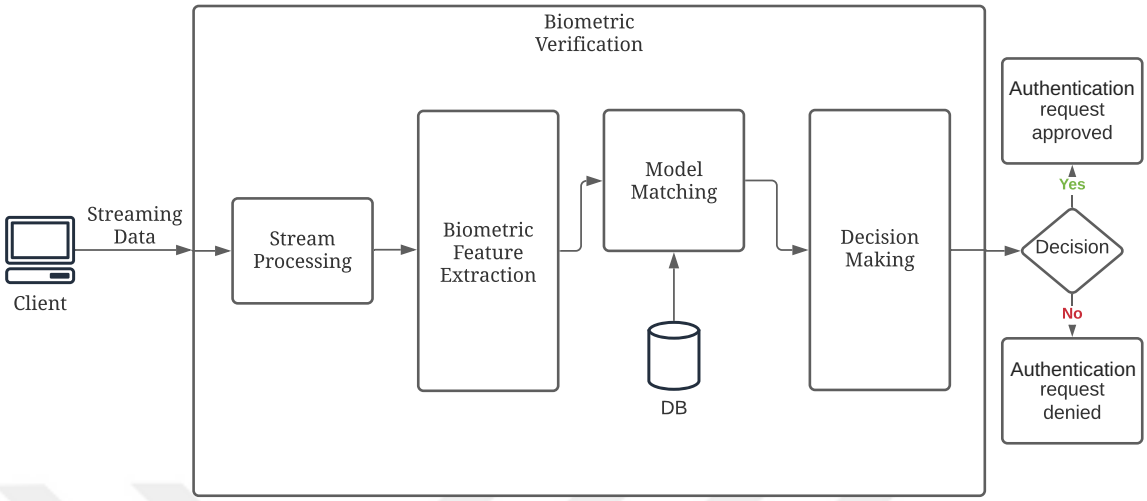


Figure 2.1. Biometric System Model.

2.1. The Bogazici Mouse Dynamics dataset

We contribute to this area, which is growing in importance as big data capabilities improve, by providing a data set that is currently unavailable in the field. This data article [15] discusses a large dataset containing freely downloadable mouse dynamics data. This dataset contains seven variables and data from 24 users totaling 2550 hours of active usage. Each user has data for training, internal attack simulations, and external assault simulations. A Python application is created that continually listens for mouse movements and clicks and logs them to a file with a timestamp, the primary window's name, and the specifics of the mouse action. We designated five individuals with the least quantity of data as external users among the 24 unique users since their data would be insufficient for training purposes and we could test external threat detection. Each user receives training data consisting of ten days of the most frequent mouse activity, while reminder days of data are utilized to conduct internal attack tests. This dataset is ideal for evaluating the vulnerability of under-development procedures to insider threats, remote unauthorized access, and physical access.

This dataset may be beneficial to students, engineers, and, in particular, researchers interested in behavioral biometrics. Similar datasets contain limited quantities of data and have a small number of users, however, our dataset has 24 unique

users and 157.377 distinct sessions. The Bogazici Mouse Dynamics dataset contains data on mouse behavior obtained from 24 distinct subjects. The dataset is divided into three categories: training, external testing, and internal testing, totaling 157.377 sessions. These categories are discussed in further depth in the next chapter. There is a file entitled "labels.csv" in the dataset directory that has two columns: the session name and the illegality of the session (0 if the session is legal, 1 if the session is illegal). Each user has their own directory dedicated to session files. There are three folders in each user's directory: one for training data, one for external test data, and one for internal test data, which contains session files. The collected data is divided into seven columns: action type, timestamp, mouse pointer X and Y locations, clicked button (or None if there was no click), click state (Move, Pressed, or Released), and the primary program window name. Dataset columns are listed with detailed descriptions in Table 2.1.

Table 2.1. Bogazici dataset mouse usage data description.

Variable	**Type**	**Description**
Action Type	Categorical	Movement or click
Timestamp	Numeric	Floating point number expressed in seconds since the epoch in UTC
X	Numeric	X location of the mouse cursor
Y	Numeric	Y location of the mouse cursor
Button	Categorical	Indicates which button is clicked (Left or Right). None if it is a movement.
State	Categorical	Button state for clicks (Press or Release). Move if it is a movement.
Window Name	Categorical	Foremost application window name

For all users, surfing (Google Chrome, Mozilla Firefox, Safari, etc.) is by far the most time-consuming activity, accounting for 51% of total time spent. Development applications (VMWare, Docker, PyCharm, etc.) come in second place with 19%, Office

applications (MS Word, MS Excel, LibreOffice, etc.) come in third place with 17%, filesystem applications (System Preferences, OS programs, setup files, etc.) come in fourth place with 11%, and entertainment applications (Spotify, Discord, video games, etc.) come in fifth place with 2%.

2.1.1. Experimental Design, Materials and Methods

Experimental users were chosen from a variety of jobs inside a software firm, allowing them to experiment with a variety of applications and tools in an office setting. Each user that takes part in the experiment is loaded with an application that is launched at boot. This application is intended to capture the user's mouse movements independently of any particular job. This program is based on the fact that on OS systems, listener callbacks are invoked directly from an operating thread. Thus, users are not impeded from conducting daily tasks, since just a little amount of computing power and memory is required to capture mouse movements. Each user runs this application for one month on their computer, and the mouse movements are saved in ".csv" files. The raw data records are encrypted using the user-supplied password and safely sent to the platform for preprocessing.

The raw data records include the time of the recording, the x and y coordinates, the button, the state, and the process name of the window that is open throughout the motion's execution. The initial stage of preprocessing is to categorize the users' process names by activity type. We classified all processes into five categories: file system, browsing, development, office, and entertainment. This is accomplished by manually marking all processes in an Excel file with their associated names. Under the file system label, operating system-generated sounds and unintelligible process names were gathered. We've established that unintelligible process names are a result of the library we're using having difficulty capturing "explorer.exe" in the Windows operating system.

After tagging the raw data, it is separated into sessions. To guarantee that training and test data for all experiments are kept distinct, it is chosen to segregate training and test data into different files. In general, each user has their own training sessions: "internal threat" sessions for evaluating performance against internal threats and "external threat" sessions for evaluating performance against external threats.

The preparation of training sessions begins with the selection of the target user's ten most active days. These are the days with the most session data. Each day's data is provided in a separate session in the training folder. Thus, ten training sessions are obtained for each user. Sessions are prepared with 30 to 100 activities picked from each user's remaining days before being divided into folders for performance monitoring against internal and external threats. The action description is critical in this case. We classified user activity by five distinct acts. There are several types of mouse movements: free mouse movement (MM), drag and drop (DD), left single click (LC), right single click (RC), and double click (DC). To interpret lower-level occurrences into these higher-level actions, rules are developed.

The attack model is not motivated by malevolent behavior. All data reflects work that was completed during authorized remote connections that were not hijacked. To mimic unauthorized use of a user's account, data from other users is intentionally blended with the user's test data. Due to the fact that all users do the same nonspecific administrative duties, the data collected from the fictional attackers do not show criminal activity.

Within the scope of the thesis, the collected data are used to simulate the "Client" block for insider threat shown in Figure 2.1. The sessions of the data contain a certain amount of data to authenticate each user. Raw user data is processed before being transmitted to other blocks. In the areas covered by the thesis, this process is designed to be performed offline for model training and online for evaluation.

Table 2.2. Balabit dataset mouse usage data description.

Variable	**Type**	**Description**
record timestamp	Numeric	Floating point number expressed in seconds since the epoch in UTC
client timestamp	Numeric	Floating point number expressed in seconds since the epoch in UTC
X	Numeric	X location of the mouse cursor
Y	Numeric	Y location of the mouse cursor
Button	Categorical	Indicates which button is clicked (Left or Right). None if it is a movement.
State	Categorical	Button state for clicks (Press or Release). Move if it is a movement.

2.2. Balabit Mouse Dynamics dataset

Experiments are also conducted using the Balabit Mouse Dynamics Challenge dataset [13]. The Balabit Mouse Dynamics Challenge data set contains information about users' mouse usage characteristics gleaned through an unidentified administrative task sequence. Users are not required to do any specific task during data acquisition. These users typically log in to remote servers via their remote desktop client over the course of their work. Between the client and the remote computer, a network monitoring device is installed that inspects all traffic according to the RDP protocol's specifications. This includes the user's mouse interactions that are communicated between the client and server during a remote session. The mouse log data comprises information on the date and time of the action, as well as the cursor's coordinates.

Each session file's fields are translated as in Table 2.2. The data structure is based on the conventional train-test set split. Each user's train set consists of legal sessions relating to its usage, while the test set comprises legal sessions relating to that user and illegal sessions relating to an unknown user. Producing an illegal session for a

user is as simple as gathering legal sessions from other users to imitate attacks carried out by an unknown attacker.

The attack model is not motivated by malicious behavior. All data reflects work that was completed during authenticated remote sessions that were not hijacked. To simulate illicit use of a user's account, data from other users is intentionally blended with the user's test data. Due to the fact that all users perform the same nonspecific administrative chores, the data collected from the fictional attackers do not show malicious activity.

The data structure of the Balabit Mouse Dynamics dataset and the data structure of the Bogazici Mouse Dynamics dataset are exactly the same in terms of the features used in the scope of the thesis. It is very important in terms of comparison of the studies [2–4] conducted using the Balabit Mouse Dynamics dataset and the methods we propose. Since it is almost impossible to re-code previous studies to produce the same results with the same parameters, this publicly available dataset, which is also used by other studies, is used in all our evaluations.

2.3. Prevented Attack Models and Use Cases

In the past, cybersecurity attacks caused by outsider vulnerabilities were more commonly reported in the mass media and were the subject of a slew of movies and television shows. According to several surveys, including a Vanson Bourne poll [16], the overwhelming majority (74 percent) of cybersecurity attacks come from inside the company. This means that a cyber-attack by the staff (or anyone with access to data) is almost three times more likely than a hacker from the outside. Insider attacks will go undetected for a long time when the user who is responsible already has access to the data and seems to be behaving naturally on the surface. In this thesis, we develop several use cases to prevent insider threats. To continue on this subject, we will briefly discuss these models and designs to prevent such threats.

2.3.1. Machine Learning Model Types to Build Biometric Models

Machine learning is the general name of computer algorithms that can learn structurally and make meaningful predictions on data. These algorithms basically work by creating models from sample data. Today, it is not possible to manually process and analyze a huge amount of data. Therefore, in order to solve a given problem, it is aimed to reach a solution by training the data obtained from the environment of the problem with machine learning algorithms. Machine Learning algorithms are basically divided into 3 groups:

(i) **Supervised Learning**

In the supervised learning method, training data contains "label" information. In other words, data with known results are used in developing a model for a solution. In this way, it is aimed to predict the results of the data without labeling information in the data set based on the model created.

(ii) **Unsupervised Learning**

Unsupervised learning, unlike supervised learning, is the learning of the relationships and structures that exist within the data without labeling the data as cause-effect or input-output. It provides inferences about data by using distances, neighborhood relationships, and densities of data samples. Two important approaches to unsupervised learning are dimension reduction and clustering.

(iii) **Self-supervised Learning**

A key feature of self-supervised learning is that it derives supervisory signals from the data itself, which is frequently made possible by the data's underlying structure. Predicting any unseen or hidden component (or property) of the input from any observed or unhidden part (or property) of the input is the goal of self-supervised learning in its most comprehensive form.

We have applied two of these 3 methods, supervised learning and self-supervised learning, to our problems. While defining the behavioral biometrics problem, we used these tools by defining two separate problems. The information of users with privileged ac-

cess is always the first target of malicious people, as it provides access to very important resources. This can cause companies to lose financially or in terms of brand image, and to suffer great material and moral damages. In addition, the privileged access right, which can be assigned to artificial intelligence-powered non-human users, can also be assigned to common accounts used to manage a company's social media profiles. The main purpose of our problem definitions here is to prevent privileged accounts from being used by someone else from within the company and to prevent another outside user from acquiring authority by using a vulnerability. In Section 2.3.2, we will discuss the potential use cases and threats in greater detail.

2.3.2. Insider Threat

Although external threats such as cyber-attacks, malware, or natural disasters are among the major causes that disrupt business continuity, threats that undermine business continuity are mostly caused by user errors. Recent research [16] indicates that the biggest threats companies overlook are internal threats, and companies should tailor their strategies to employee errors and internal threats in order to protect their data and business continuity. An insider threat is a malicious threat to an organization that comes from insiders, such as staff, former employees, contractors, or business associates, who have inside knowledge about the organization's security procedures. Fraud, theft of sensitive or commercially valuable information, theft of intellectual property, or sabotage of computer systems are all potential risks. Insiders could have accounts that grant them legal access to information systems, access that was granted to them to assist them in the execution of their duties; however, these rights may be misused to damage the company. Insiders are also intimately acquainted with the company's records and intellectual property, as well as the safeguards in order to protect them. Insiders do not need to break into the corporate network from the outside perimeter by circumventing firewalls since they are still within the premises, often with easy access to the organization's internal network. Since the insider now has valid access to the organization's information and properties, insider threats are more difficult to guard against than external attacks.

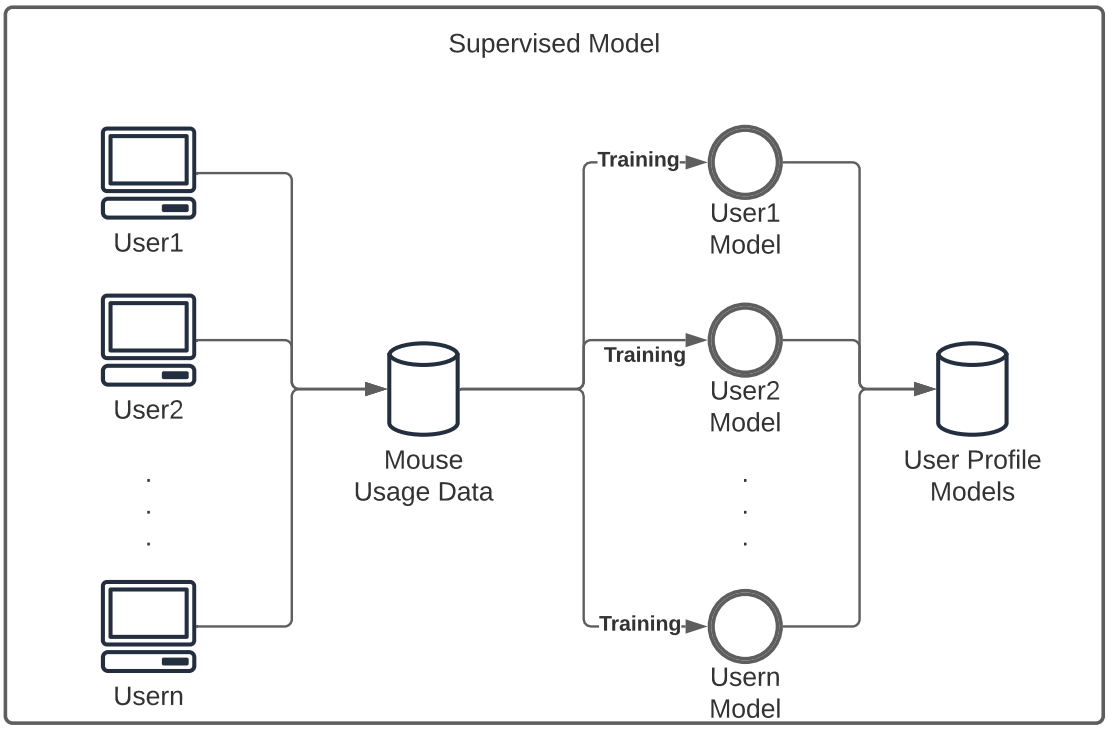


Figure 2.2. Supervised Biometry Model.

2.3.2.1. Focused Insider Threat Scenarios. Insider threats are often discussed as a broad group. This section focuses on the malicious side of insider threats, ignoring the terms "human error" and "accident." We will briefly describe the risks associated with employee activities by grouping them into 6 categories and discuss the potential risks that we try to prevent. The following scenarios are discussed in this scope:

(i) **Operations have been sabotaged by an insider (or a recent employee):** Almost every case of sabotage involved a disgruntled employee who knew he or she was about to be fired and took action to prevent it. For post-termination sabotage, access was not correctly cut-off, or the insider managed to backdoor their access before their last day. Many of the tasks were straightforward administrative in nature: delete this, disable that, change a password. The deployment of malware was one of the cases. In this scenario, the malicious user is using her/his credentials provided legally. Thus, this type of action can be prevented by ISO27001 processes [17] that regulate what should be done after the work exit

act is declared. This type of action is not in our scope since the malicious user is using the legally provided credentials.

(ii) **An insider has profited from confidential information:**
These situations necessitated some sort of profit scheme that would not have been feasible without data access. When the malicious user has enough motivation, he can try to extract data from the system using the credentials legally provided. Depending on the magnitude of the motivation, it can be quite damaging to the company. In the past, such an attack has happened to Amazon. Internal sales data and reviewers' email addresses, as well as a service to erase negative reviews and recover banned Amazon accounts, are being offered by brokers for Amazon employees in Shenzhen, according to the Journal. Payments range from $80 to more than $2,000. Since these attacks occur with legally assigned credentials, the biometric system suggested in the thesis does not prevent such attacks.

(iii) **An insider has gained access to another employee's credentials:**
In this part, the sensitive data or information that is tried to be accessed may be at a level that the malicious employee cannot reach with his / her authority. With enough motivation, the malicious employee who reaches the credentials of his colleague can try to access the information with them. He/She may try to do this by basically stealing or guessing his friend's information. The malicious employee, who uses these credentials where his/her authority is not normally sufficient, can connect to the sessions assigned to him/her by the privileged access management system with the RDP protocol. The KRON SingleConnect product is created to prevent this by asking users for a piece of second information before granting authorization over the RDP protocol, this can be a SIM card or an SMS service. However, after passing the second layer, the attack is completed for the malicious user. In the biometric system proposed in the thesis, a continuous scan is performed in the virtual machine that the malicious user connects with the RDP protocol. The malicious user is detected in this way and an alarm is generated in the privileged access management system. As can be seen, the proposed biometric system can monitor the interaction of the user with the device on the fly, detect the malicious user with a security layer that cannot be imitated,

and generate an alarm for the security manager.

In this manner, the biometric structure proposed by the thesis has been developed to prevent the third problem mentioned. The studies carried out to ensure that the structure itself is reliable against attacks will be explained later. How we model the threat is given in detail in Section 2.3.2.2.

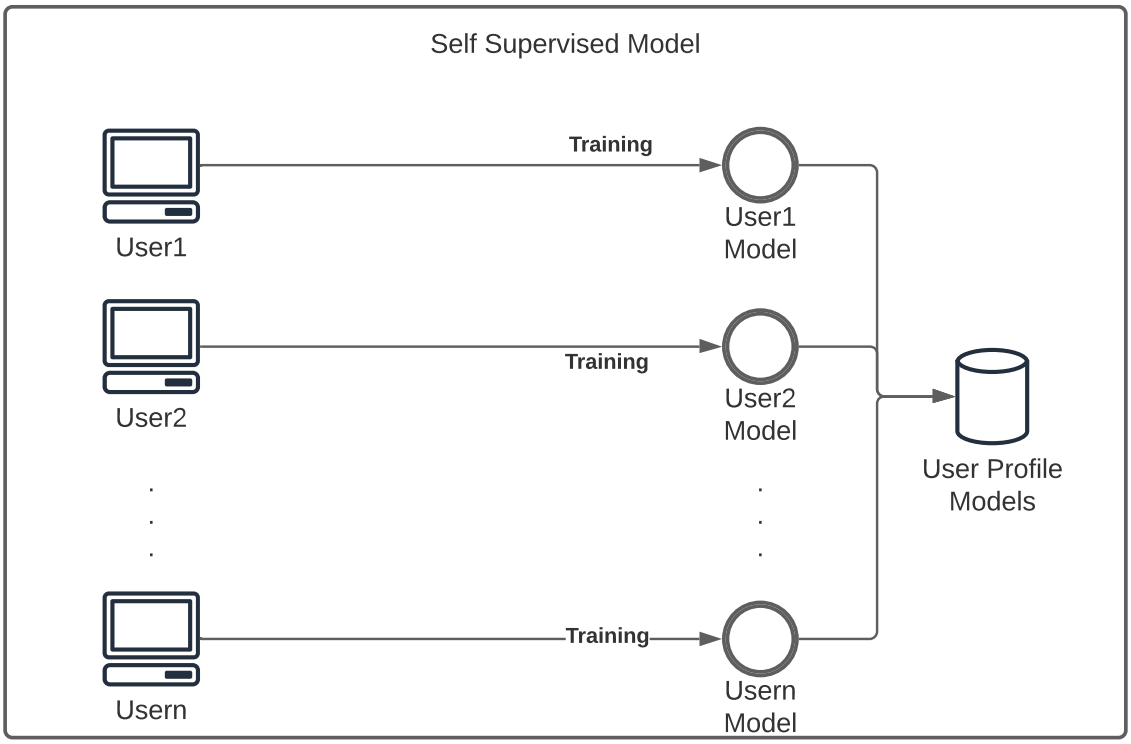


Figure 2.3. Self-supervised Biometry Model.

2.3.2.2. Insider Threat Modelling. The scenario of the third problem we mentioned identified is as follows: Within a company, data from the devices used by the employees is continuously collected by auxiliary software. Here, each user has their device and it is assumed not to change it. This user must be registered in authority access systems within the company. Since the usage data of the registered user is different from other company employees, modeling is made to ensure that the roles assigned to individuals within the company are not used by someone else. This model uses supervised learning. Basically, learning is done in order to obtain a model that learns the difference between the target user and other users. In this way, a model is obtained that can detect the

use of privileged accounts by someone within the company. The data flow scenario of the first problem defined is as seen in Figure 2.2. Here, it is assumed that the data received from the users are collected in a center. It is known from which user the data collected from in this center. With data from the central dataset, each user's own data and a randomly selected portion of other users' data are trained in binary classifiers. Here, the usage difference of the target user from other users within the company is learned.

In some use cases, users may not want to keep sensitive user data in a central data warehouse, this may be due to data protection laws or the desire of internal employees not to share this data. For this reason, it is necessary to learn only with the data belonging to the user. In the literature, such problems are called novelty detection [18] and this problem is solved with self-supervised learning. The contribution of the thesis to this field will be very important. The defined problem can be applied to many places with different biometric data sources. The general structure is given in Figure 2.3.

3. FEATURE EXTRACTION

Raw mouse data is very large and useless data without processing, where it is possible to express a single activity with different sizes and amounts of data, which differ according to each activity. An exemplary raw mouse usage data fragment is shown in Table 3.1. The focus of the thesis at this point is to express the daily operations performed by the user without any activity restriction with a fixed-length vector. This fixed-length vector is used in model education as a feature vector. Before obtaining the feature vector, the raw data is passed through several stages. The first step begins by defining each row of raw mouse data as an event and grouping those rows into an action. This section is examined under the heading of data preprocessing. The grouped events are then converted into attribute signals. At this stage, the constant feature vector is still not obtained. The last step, time and frequency-based statistical analysis, ends by defining the available raw data, each action as a vector of fixed length. Briefly, data pre-processing, extraction of the feature signal, and conversion of this feature signal to a fixed-length vector by analysis are completed.

Table 3.1. A partial raw mouse usage data.

Recording Time Stamp (milliseconds)	**Button**	**State**	**x (Pixels)**	**y (Pixels)**
23.150	Left	Press	177	354
23.229	Left	Release	177	354
23.399	NoButton	Movement	177	353
23.524	NoButton	Movement	159	348
23.524	NoButton	Movement	156	346
23.758	NoButton	Movement	153	347
23.962	NoButton	Movement	153	355
24.071	NoButton	Movement	153	343
24.227	NoButton	Movement	155	336
24.477	NoButton	Movement	155	338
24.727	Left	Press	155	338
24.977	Left	Release	155	338

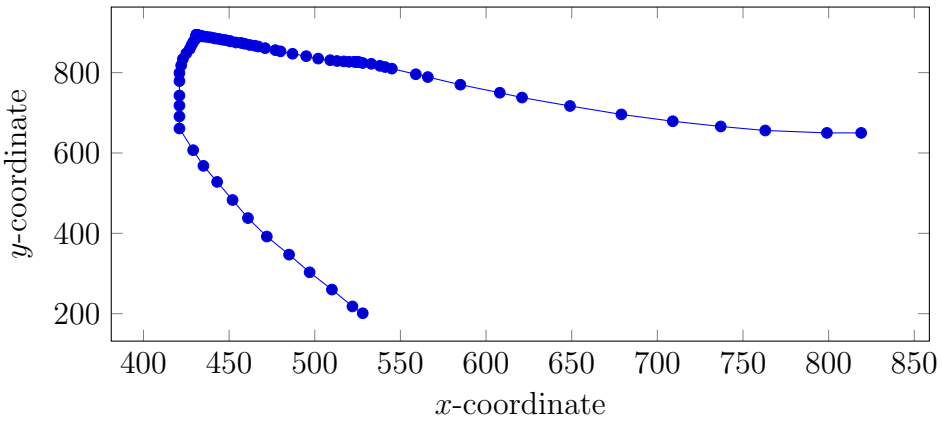


Figure 3.1. Segmentated MM action.

3.1. Data Preprocessing

The Bogazici Mouse Dynamics dataset is used for experiments. The Bogazici Mouse Dynamics data includes mouse usage characteristics of users that were collected from an unspecified administrative task sequence. There is no particular task that users are obligated to perform during data acquisition. A mouse usage monitoring software is used for each users' data collection process as discussed in Section xx. The mouse log data contains information on time, action type, and coordinates of the mouse cursor.

Every session file's fields are to be interpreted as follows:

- register timestamp: the amount of time that has passed since the session began (in seconds),
- button: the status of the mouse buttons at the time,
- state: supplementary information about the mouse's present state
- x: the cursor's x coordinate on the screen, -y: the cursor's y coordinate on the screen.

The data is organized using a traditional train and test set split. Each user's train set is made up of legal sessions of their use, while the test set is made up of legal sessions from that user and illegal sessions from an undisclosed user. Creating illegal sessions for a user is as simple as collecting legal sessions from other users to simulate attacks carried

out by an unknown attacker. There is no malicious behavior in the attack model. All of the data recorded reflects work done during authenticated remote sessions that were not hijacked. The attacks are spoofs. Data from other users is artificially blended into the test data of each user to simulate unauthorized use of his or her account. The data from the artificial attackers do not represent malicious activities because all users perform the same unspecified administrative tasks.

3.1.1. Action Segmentation

Coordinates, keys, states, and a timestamp are used to display all mouse operations. Mouse-move Event *mm*, Drag Event *dd*, Left Button Pressed Event *lp*, Right Button Pressed Event *rp*, Left Button Released Event *lr*, and Right Button Released Event *rr* are the different types of events accessible. Aggregating events into actions is as simple as dividing a whole session into actions and is based on composing data instances. To interpret lower-level events into higher-level actions, such as Mouse-move Action, Drag&Drop Action, Left Click Action, Right Click Action, and Double Click Action, several thresholds are defined. Movement Actions and Click Actions are the two types of actions available.

3.1.1.1. Movement Actions. Movement actions contain Mouse-move Action (MM) and Drag Action (DA). These actions are collections of *mm* events for MM and *dd* events for DA. MM action is defined as

$$\mathrm{MM}_{t_1}^{t_k} = \langle \mathrm{mm}_{t_1}, \mathrm{mm}_{t_2}, \cdots, \mathrm{mm}_{t_k} | t_{n+1} - t_n \leq \eta_{\mathrm{MM}} \rangle \tag{3.1}$$

where $\forall n : 1 \leq n \leq k$ and η_{MM} is silence threshold. Silence threshold is used to separate inactive moments of users. Figure 3 is an example for this action. DA action is defined as that it starts with either *lp* or *rp* and finishes either *lr* or *rr* as

$$\mathrm{DA}_{t_1}^{t_k} = \langle \mathrm{dd}_{t_1}, \mathrm{dd}_{t_2}, \cdots, \mathrm{dd}_{t_k} | t_{n+1} - t_n \leq \eta_{\mathrm{DA}} \rangle \tag{3.2}$$

where $\forall n : 1 \leq n \leq k$ and η_{DD} is silence threshold for DA action.

Table 3.2. Mouse feature signals used in methodology.

Feature Name	**Definition**
Angle of Movement	$\theta_i = \arctan(\frac{\delta y_1}{\delta x_1}) + \sum_{j=1}^{i} \delta\theta_j$
Curvature	$\vec{c} = \delta\theta/\delta s$
Curvature Change	$\vec{\Delta c} = \delta c/\delta t$
Horizontal Velocity	$\vec{\nu_x} = \delta x/\delta t$
Vertical Velocity	$\vec{\nu_y} = \delta y/\delta t$
Velocity	$\vec{\nu} = \sqrt{\nu_x^2 + \nu_y^2}$
Acceleration	$\dot{\nu} = \delta\nu/\delta t$
Jerk	$\vec{\ddot{\nu}} = \delta\dot{\nu}/\delta t$
Angular Velocity	$\vec{w} = \delta\theta_t/\delta t$

3.1.1.2. Click Actions. Click Actions contain Left Click Action (LC), Right Click Action (RC) and Double Click Action(DC). These actions are defined with thresholds as

$$\text{LC}_{t_1}^{t_2} = \langle \text{lp}_{t_1}, \text{lr}_{t_2} | t_2 - t_1 \leq \eta_{\text{LC}} \rangle \tag{3.3}$$

where η_{LC} is silence threshold for LC and

$$\text{RC}_{t_1}^{t_2} = \langle \text{rp}_{t_1}, \text{rr}_{t_2} | t_2 - t_1 \leq \eta_{\text{RC}} \rangle \tag{3.4}$$

where η_{RC} is silence threshold for RC. DC action is defined as the concentration of two consecutive LC actions as

$$\text{DC}_{t_1}^{t_4} = \langle \text{LC}_{t_1}^{t_2} \cdot \text{LC}_{t_3}^{t_4} | t_3 - t_2 \leq \eta_{\text{DC}} \rangle \tag{3.5}$$

where η_{DC} is silence threshold for double click activity.

3.2. Feature Signals

After the two types of action are determined, it is seen that the grouped movement actions are still not of fixed length. As mentioned, there is one more process that needs to be done before reducing these different length vectors to a single length. MM and DA actions are treated as same in terms of extracted features. Each mouse movement action is associated with time information as $\left\{t_i\right\}_{i=1}^{n}$ and coordinate information as $\left\{x_i\right\}_{i=1}^{n}$ and $\left\{y_i\right\}_{i=1}^{n}$. With these, the methodology of [9] is used to extract spatial and temporal information from each mouse movement. The trajectory length of each mouse movement is defined as

$$S_i = \sum_{k=1}^{i-1} \sqrt{\delta x_k^2 + \delta y_k^2} \tag{3.6}$$

where $\delta x_k = x_{k+1} - x_k$ and $\delta y_k = y_{k+1} - y_k$. The proposed feature signals are given in Table 3.2.

3.2.1. Angle of movement

This feature signal represents the angle of the path tangent with the x-axis as a vector.

$$\theta_i = \arctan\left(\frac{y_1}{dx_1}\right) + \sum_{j=1}^{l} \delta\theta_j \tag{3.7}$$

where

$$\delta\theta_j = \delta \arctan\left(\frac{\delta y_y}{dx_j}\right) \tag{3.8}$$

is the minimal angle change given in $[-\pi, \pi]$ between the j_{th} $(j+1)_{th}$ points.

3.2.2. Curvature-Based Feature Signals

The rate of change of the tangential angle with respect to the arc length θ [19] is known as the curvature of a continuously differentiable curve in mathematics; these angles are independent of the curve's direction, orientation, and size. Curvature is defined as

$$\vec{c} = \frac{\delta\theta}{\delta s}. \tag{3.9}$$

The curvature of mouse curves is difficult to handle because they are piecewise linear, with zero curvature within line segments and ill-defined curvature at the corners. Approximating the rough mouse curve with another smooth curve is one way to solve this problem which is B spline [10]. A parametric approximation of a mouse curve is generated by using the mouse curve coordinates as B -spline control points. The cubic B-spline blending functions can now be used to determine the curve's curvature properties. The B-spline curvature is one of the properties used in mouse curve classification in this scheme. The general shape of the curve can be measured using conveniently computed metrics by analyzing the curvature. Due to the parametric nature of B-spline curves, the standard extrinsic curvature for a 2D parametric curve is used as

$$\vec{c} = \frac{\frac{\delta x}{\delta t}\frac{\delta y^2}{\delta^2 t} - \frac{\delta y}{\delta t}\frac{\delta x^2}{\delta^2 t}}{\left((\frac{\delta x}{\delta t})^2 + (\frac{\delta y}{\delta t})^2\right)^{\frac{3}{2}}} \tag{3.10}$$

where $\delta t_k = t_{k+1} - t_k$ and used this function as a approximation of $\frac{\delta\theta}{\delta s}$. After that, we define curvature change as

$$\vec{\Delta} c = \delta c / \delta t. \tag{3.11}$$

3.2.3. Velocity

The velocity stands for mouse usage speed in terms of pixel/seconds. Since x and y coordinates are available with occurrence time, we used 3 types of velocities to generate feature signals.

(i) Horizontal Velocity:

$$\vec{\nu_x} = \delta\vec{x}/\delta\vec{t} \tag{3.12}$$

(ii) Vertical Velocity:

$$\vec{\nu_y} = \delta\vec{y}/\delta\vec{t} \tag{3.13}$$

(iii) Velocity:

$$\vec{\nu} = \sqrt{\vec{\nu_x}^2 + \vec{\nu_y}^2} \tag{3.14}$$

3.2.4. Acceleration

The rate of change of velocity $\vec{\nu}$ with respect to time is referred to as acceleration. We defined acceleration as

$$\dot{\vec{\nu}} = \delta\vec{\nu}/\delta\vec{t}. \tag{3.15}$$

3.2.5. Jerk

The rate at which an object's acceleration varies with respect to time is referred to as jerk or jolt. We defined jerk as

$$\vec{\ddot{\nu}} = \delta\vec{\dot{\nu}}/\delta\vec{t}. \tag{3.16}$$

3.2.6. Angular Velocity

The angular velocity metric, also known as the angular frequency vector, is a vector measure of rotation rate in relation to time that describes how quickly an object rotates or spins in relation to another point. We defined angular velocity as

$$\vec{w} = \delta\vec{\theta}/\delta\vec{t}. \tag{3.17}$$

3.3. Fixed-sized Vectorization of Feature Signals

Table 3.3. Features are used to represent a movement action.

Feature Name	**Definition**	**Number of Features**
Minimum, Maximum, Mean, Variance, Skewness, Kurtosis	$\theta, c, \Delta c, \nu_x, \nu_y, \nu, \dot{\nu}, \ddot{\nu}, w$	54
Mean Frequency, Mean Power, Peak Powers, Peak Power Frequencies	$c, \Delta c, \nu, \dot{\nu}, \ddot{\nu}, w$	48
Duration of Movement, Traveled Distance	t_n, S_n,	2
Straightness	$\frac{\sqrt{(x_n - x_1)^2 + (y_n - y_1)^2}}{S_n}$	1

In this section, the vectorization i.e feature extraction is described. Since the mouse usage curves varies in length, we used a couple of methods to make them fit into a learning scheme. These methods are grouped into time domain statistical analysis and frequency domain statistical analysis. All methods are summarized in Table 3.3.

3.3.1. Time Domain Statistical Analysis

It's really interesting to figure out what the fundamental mathematical properties of function signals are. These function signals are statistically evaluated, and the corresponding minimum, limit, mean, standard deviation, skewness, and kurtosis are determined for each of them, as suggested by [9]. In addition, several scalar features such as Duration of Movement, Traveled Distance, and Straightness is determined from these function signals, as seen in Table 3.3.

3.3.2. Frequency Domain Statistical Analysis

The power spectral density of these feature signals is a strong candidate for feature extraction since any feature signal of the movement sequences has a period associated with it. However, Fourier Power Spectral Density (PSD) does not refer to data that has been sampled unevenly, as we have. For detecting periodicity in irregularly sampled function signals, the well-known Lomb-Scargle periodogram suggested by [20] and [21] is used.

Astrophysicists created the Lomb-Scargle periodogram protocol because they often deal with data that isn't uniformly sampled. However, the algorithm's usefulness isn't limited to data that isn't uniformly distributed. There are also advantages of having data that is uniformly sampled. While this algorithm is not a conventional Fourier transform and would not exactly replicate FFT results, it generates results that are almost equivalent to an FFT. Using a variable offset term in the PSD specification, the algorithm effectively specifies the second sequence of abscissa (time) values. This yields an algorithm that is equivalent to fitting sine curves (at given frequencies) to the data using the least-squares method. With the exception of oversampling and spectral length choices, this choice is close to the Fourier Spectrum option. Data with void values can also be processed using this spectral protocol. These are simply omitted from the data stream, resulting in data that is sampled unevenly.

Consider N data point in a feature signal $[e_1, \cdots, e_N]$, sampled at t_j where $j = 1, \cdots, N$. The Lomb-Scargle method proposes that

$$P(\omega) = \frac{1}{2\sigma^2} \left(\frac{\left[\sum_{j=1}^{N} (e_j - \overline{e}) \cos \omega(t_j - \tau)\right]^2}{\sum_{j=1}^{N} \cos^2 \omega(t_j - \tau)} + \frac{\left[\sum_{j=1}^{N} (e_j - \overline{e}) \sin \omega(t_j - \tau)\right]^2}{\sum_{j=1}^{N} \sin^2 \omega(t_j - \tau)} \right) \tag{3.18}$$

where

$$\overline{e} = \frac{1}{N} \sum_{j=1}^{N} e_j \tag{3.19}$$

and

$$\sigma^2 = \frac{1}{N-1} \sum_{j=1}^{N} (e_j - \overline{e})^2 \tag{3.20}$$

are mean and variance of the feature signal. τ is the time-offset which satisfies orthogonality of the model and makes $P(\omega)$ time-invariant. τ is chosen as

$$\tan 2\omega\tau = \frac{\sum_{j=1}^{N} \sin 2\omega t_j}{\sum_{j=1}^{N} \cos 2\omega t_j}. \tag{3.21}$$

Thus, unevenly sampled feature signals can be analyzed in frequency space . Since the Lomb–Scargle Method calculates $P(\omega)$ for individual $\omega = 2\pi f$ frequencies, The boundary frequencies and frequency spacing δf are determined between limits.

A logical approach to determine limits is considering the feature signal characteristics and computational complexity. The Nyquist frequency is derived from properties of regularly-sampled data and it can not be applied directly into our case. Feature signal with range $T = t_{max} - t_{min}$ contains N data point has a high frequency component

multiple of

$$\hat{f}_{Ny} = \frac{N}{2T} \tag{3.22}$$

where $\hat{f}_{Ny}$ is "average" Nyquist frequency mentioned in [22]. While the upper limit is determined by feature signal itself as $\hat{f}_{Ny}$, the lower limit is chosen to be one δf away from 0 Hz.

In order to capture enough resolution within boundary, δf is chosen as

$$\delta f = (5T)^{-1} \tag{3.23}$$

with oversampling factor 5. This allows us to reduce complexity as

$$N_{freq} \sim O\left[\frac{\hat{f}_{Ny}}{\delta f}\right] \sim O\left[\frac{N/(2T)}{1/T}\right] \sim O[N] \tag{3.24}$$

as suggested by [23].

After the Lomb–Scargle periodogram estimation, the estimation is used to extract following features:

- Mean frequency:

$$\text{MF} = \frac{\sum_{i=1}^{M} f_i P_i}{\sum_{i=1}^{M} P_i} \tag{3.25}$$

  where $M = \hat{f}_{Ny}/\delta f$.

- Mean power:

$$\text{MP} = \frac{1}{M} \sum_{i=1}^{M} P_i \tag{3.26}$$

- Peak Powers (PP) and Corresponding Frequencies (PF): We find three local maxima (PP) by a simple comparison of neighboring power values and corresponding frequencies (PF).

Thus, additional 8 features are extracted for c, Δc, ν, $\dot{\nu}$, $\ddot{\nu}$, w. Time and frequency domain feature analysis are summarized in Table 3.3.

3.4. Eigenspace Transform with Kernels

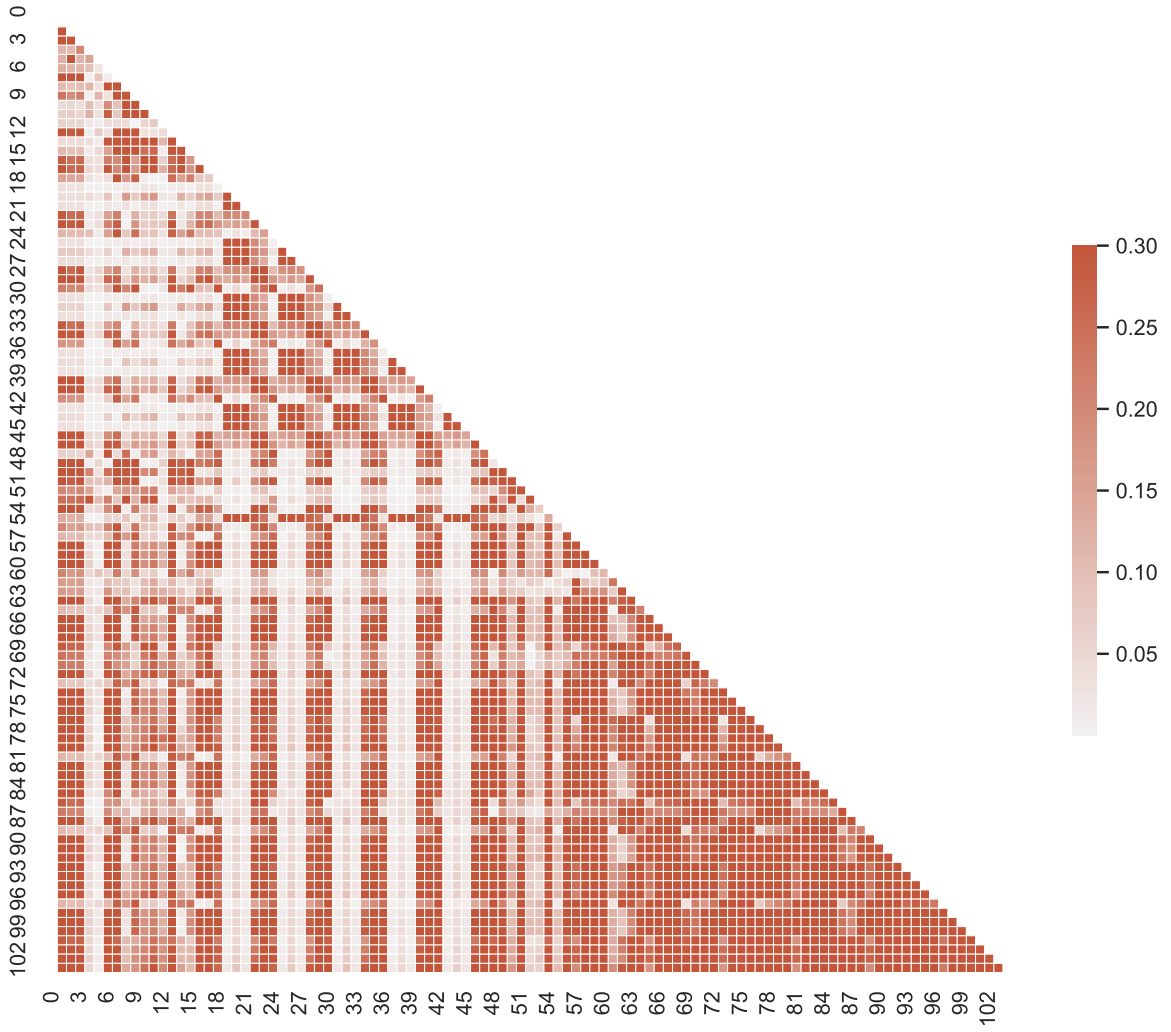


Figure 3.2. Correlation Matrix of Time and Frequecy Features.

Principal Components Analysis (PCA) is a well-known unsupervised dimensionality reduction technique that creates relevant features/variables by combining the original variables in linear (linear PCA) or non-linear (kernel PCA) ways (features).

So, it is an orthogonal transformation of data into a series of uncorrelated data in the reduced PCA space, with the first component explaining the most variation and each subsequent component explaining less.

We use eigenspace transform for self-supervised learning methods only. The state of the extracted features is highly correlated as can be seen in Figure 3.2. However, the used supervised method is correlation-proof in a computational sense. For movement features described in Section 3.3, an eigenspace transformation is fitted on training instances after the standardization process and principal components are extracted. With these principal components, the test instances are also projected into the same space and given as input for action classifiers. In the following parts, the details about the operations are given.

3.4.1. Kernel Principal Component Analysis

Kernel PCA uses the kernel trick to perform PCA in a richer feature space. Since we try to reach a richer space, we push all data points $\{\mathbf{x}_n, (n = 1, \cdots, N)\}$ to $f(x)$ where F is higher dimensional feature space. Covariance matrix is

$$\Sigma_f = \frac{1}{N} \sum_{n=1}^{N} f(\mathbf{x}_n) f(\mathbf{x}_n)^T \tag{3.27}$$

and intuitively, we plug this equation into eigenvalue finding equation

$$\Sigma_f \phi_i = \lambda_i \phi_i \tag{3.28}$$

and we get

$$[\frac{1}{N} \sum_{n=1}^{N} f(\mathbf{x}_n) f(\mathbf{x}_n)^T] \phi_i = \sum_{n=1}^{N} (f(\mathbf{x}_n) \cdot \phi_i) f(\mathbf{x}_n) = \lambda_i \phi_i \tag{3.29}$$

where $\cdot$ denotes dot product. To express eigenvector ϕ_i as a linear combination of N mapped data points and reach adjustable kernel expression, let's rewrite the expression as

$$\phi_i = \frac{1}{\lambda_i N} \sum_{n=1}^{N} (f(\mathbf{x}_n) \cdot \phi_i) f(\mathbf{x}_n) \tag{3.30}$$

and left multiply both sides with $f(\mathbf{x}_m)^T$

$$(f(\mathbf{x}_m) \cdot \phi_i) = \frac{1}{\lambda_i N} \sum_{n=1}^{N} (f(\mathbf{x}_n) \cdot \phi_i)(f(\mathbf{x}_m) \cdot f(\mathbf{x}_n)). \tag{3.31}$$

In order to use kernel trick, we focus $(f(\mathbf{x}_m) \cdot f(\mathbf{x}_n))$ part and define kernel as

$$k(\mathbf{x}_m, \mathbf{x}_n) = (f(\mathbf{x}_m) \cdot f(\mathbf{x}_n)), \quad (m, n = 1, \cdots, N) \tag{3.32}$$

and we reach final expression by substituting kernel expression into Equation (3.31)

$$(f(\mathbf{x}_m) \cdot \phi_i) = \frac{1}{\lambda_i N} \sum_{n=1}^{N} (f(\mathbf{x}_n) \cdot \phi_i) k(\mathbf{x}_m, \mathbf{x}_n). \tag{3.33}$$

For vectorized computation and more detailed derivations, Schölkopf's work [24] can be visited.

There are some kernels that widely used by scientists and three of them are under our concern in addition to linear PCA.

3.4.1.1. Sigmoid. The Sigmoid Kernel and the Multilayer Perceptron (MLP) kernel are other names for the Hyperbolic Tangent Kernel. The Sigmoid Kernel is derived from the field of Neural Networks, where the bipolar sigmoid function is frequently used as an artificial neuron activation function. The kernel is defined as

$$k(\mathbf{x}_m, \mathbf{x}_n) = \tanh((\mathbf{x}_m \cdot \mathbf{x}_n) + \theta). \tag{3.34}$$

The slope alpha and the intercept constant c are two customizable parameters in the sigmoid kernel. 1/N, where N is the data dimension, is a common value for alpha. The works of Hsuan-Tien and Chih-Jen [25] provide a more in-depth look at sigmoid kernels.

It's worth noting that a sigmoid kernel function SVM model is similar to a two-layer perceptron neural network. Because of its neural network theory origins, this kernel was very common for support vector machines. It has also been found to perform well in practice in Boughorbel et al. [26], despite being only conditionally positive definite. Thus, it is a strong candidate for our feature space.

3.4.1.2. Polynomial. The Polynomial kernel is a non-stationary kernel. Polynomial kernels are ideal for problems in which all of the training data has been normalized. Since we use normalization on movement features, it is suitable for our proposed self-supervised methods. The kernel is defined as

$$k(\mathbf{x}_m, \mathbf{x}_n) = (\mathbf{x}_m \cdot \mathbf{x}_n)^k. \tag{3.35}$$

3.4.1.3. Radial basis function (RBF). The Gaussian kernel is an example of a radial basis function kernel. We prefer Gaussian kernel and this kernel is defined as

$$k(\mathbf{x}_m, \mathbf{x}_n) = exp(-\frac{||\mathbf{x}_m - \mathbf{x}_n||}{2\sigma^2}) \tag{3.36}$$

where σ is adjustable. The adjustable parameter sigma has a significant impact on the kernel's performance and should be fine-tuned to the specific problem at hand. When the exponential is overestimated, it behaves almost linearly, and the higher-dimensional projection loses its non-linear power. On the other hand, if the function is undervalued, it will lack regularization, and the decision boundary will be extremely sensitive to noise in the training data.

While we are tuning parameters for better performance, one of four kernels that best expressed the Movement Features was chosen based on performance. After the KPCA is applied and the principal component is extracted, the resulting new representation for the test set generated by transformation with principal components is input to the action classifiers.

4. PROPOSED VERIFICATION MODEL

The overall verification process is described in the following sections, which are divided into how to learn legality scores for individual acts and how to aggregate these scores for decision-making. The learning part considers two types of models, which are supervised learning-based verification and self-supervised learning-based. The details are discussed about the learning schemes and the individual models.

4.1. Learning

The proposed approach involves calculating the likelihood of each action belonging to a target consumer and combining these possibilities in decision-making. The trained classifiers for each action are mentioned in this section. For each type of action such as movement actions and click actions, we provide brief explanations on usages.

4.1.1. Supervised Learning

The classifiers are trained separately for each action type, and each action is given a legality rating. The learning process is structured as a binary classification problem with these classifiers. In Figure 4.2, the supervised learning scheme is given. The data collection part consists of the target user and other users in the company. The agent used for the data collection on user devices is given as a Mouse data collector block. Then, the usage data is processed into features and written to the database as given. Lastly, the features are read in terms of *userid*s, and binary classification problem conditions are met by defining positive data as the target user data and negative data as the other users' data. Lastly, each action classifier is trained with these data and written into the corresponding database table. Gradient Boosting Machines are used as classifiers to boost classification performance. In the next part, the gradient boosting machines and the methods that prevent the models' overfitting are discussed.

Gradient Boosting Algorithm:

Input: training set $\{(x_i, y_i)\}_{i=1}^{n}$, a differentiable loss function $L(y, F(x))$, number of iterations M.

1. Initialize model with a constant value:

$$F_0(x) = \arg\min_{\gamma} \sum_{i=1}^{n} L\left(y_i, \gamma\right).$$

2. For $m = 1$ to M :

2a. Compute so-called pseudo-residuals:

$$r_{im} = -\left[\frac{\partial L\left(y_i, F\left(x_i\right)\right)}{\partial F\left(x_i\right)}\right]_{F(x)=F_{m-1}(x)} \quad \text{for } i = 1, \ldots, n.$$

2b. Fit a decision tree $h_m(x)$ to pseudo-residuals, i.e. train it using the training set $\{(x_i, r_{im})\}_{i=1}^{n}$.

2c. Compute multiplier γ_m by solving the following one-dimensional optimization problem:

$$\gamma_m = \arg\min_{\gamma} \sum_{i=1}^{n} L\left(y_i, F_{m-1}\left(x_i\right) + \gamma h_m\left(x_i\right)\right)$$

2d. Update the model:

$$F_m(x) = F_{m-1}(x) + \gamma_m h_m(x)$$

3. Output $F_M(x)$.

Figure 4.1. Gradient Boosting Algorithm.

4.1.1.1. Gradient Boosting Machines. It is extremely important to understand the internal mechanism of the gradient boosting machines. We use this algorithm to generalize our findings on representative Bogazici Mouse Dynamics Dataset and we alter

the optimization steps to shape the algorithm for making it suitable for real-world use cases. In Figure 4.1, there are 3 steps that make GBM (Gradient Boosting Machines) converge. The algorithm starts with a constant model $F_0 = y$ suits the real y-values well. Going from a high bias to a low bias model is where the concept of boosting comes into play. We start with a constant function (no other function can have more bias than a constant function unless the dataset is a constant model) and work our way down to a function with a reasonable bias. For a constant function, the loss function is as follows:

$$\sum_{i=1}^{n} L\left(y_i, \gamma\right) \tag{4.1}$$

As a result, $\gamma_{optimal}$ is decided by solving the optimization problem as

$$F_0(x) = \gamma_{\text{optimal}} = \min_{\gamma} \sum_{i=1}^{n} L\left(y_i, \gamma\right). \tag{4.2}$$

At this point $F_M(x) = F_0(x)$. $F_M(x)$ is updated until all of the M base learners have been added with appropriate weights. In Step 2, from $m = 1$ to $m = M$, it repeats the substeps for each base learner i.e decision tree. It's worth noting that gradient boosting only adds one model at a time, one by one. In Step 2a, for each i_{th} training example, pseudo-residuals are calculated as

$$r_{im} = -\left.\frac{\partial L\left(y, F_{m-1}(x)\right)}{\partial F_{m-1}(x)}\right|_{x=x_i, y=y_i} \text{ for all } i = 1, 2, \ldots, n. \tag{4.3}$$

In this manner, pseudo-residuals are origined from residual error calculated by taking a derivative of square loss function as

$$\begin{aligned} L &= \left(y_i - y_p\right)^2 \\ \tfrac{\partial L}{\partial y_p} &= -2\left(y_i - y_p\right) \\ -\tfrac{\partial L}{\partial y_p} &= 2\left(y_i - y_p\right) \end{aligned} \tag{4.4}$$

where $F(x_i) = y_p$ and negative of derivative of squared loss with respect to hypothesis function gives the residual for i_{th} sample. $F_{m-1}(x)$ is the model obtained by multiplying the initial constant function by $m-1$ weighted base learners. The base learner m_{th} has yet to be added. The weighted sum of base learners from 1 to $m-1$ and the initial constant function are used in each residual calculation r_{im} for i_{th} training example corresponding to the current base learner m. In Step 2b, the base learners are fit on the pseudo residuals. A new dataset is created from the provided dataset as

$$D_{\text{modified}} = \{(x_i, r_{im}) : i = 1, 2, \ldots, n\}. \tag{4.5}$$

This means adding a r_{im} column to the dataset (removing y_i).A decision tree $h_m(x)$ is trained and fitted on this dataset. r_{im}s are now terminal leafs in the decision tree. Then,

$$F_m(x) = F_{m-1}(x) + \gamma h_m(x) \tag{4.6}$$

calculated to determine $F_m(x)$. γ is the hyperparameter for the learning rate. In Step 2c, the optimum γ is calculated. The original dataset is taken and optimum γ is selected as the value where the loss is minumum. Thus,

$$\gamma_{\text{optimum}} = \underset{\gamma}{\operatorname{argmin}} \sum_{i=1}^{n} L\left(y_i, F_m\left(x_i\right)\right) = \operatorname{argmin} \sum_{i=1}^{n} L\left(y_i, F_{m-1}\left(x_i\right) + \gamma h_m\left(x_i\right)\right) \tag{4.7}$$

calculated optimum γ. After reaching optimum γ, the $F_m(x)$ is updated. In Step 2d, the update is occured as

$$F_m(x) = F_{m-1}(x) + \gamma_{\text{optimum}}\, h_m(x). \tag{4.8}$$

Lastly, these substeps are repeated for each $h_m(x)$ to update overall model to reach $F_M(x)$. In this algorithm, the dataset used is always the same and no feature selection is done. This algorithm is really open to overfitting in this way. Also, the individual decision trees may result in an overfitting problem.

To learn user actions in comparison to others, ensemble learning approaches are used. Gradient boosting machines (GBM) outperform other bagging algorithms and hyperplane separation algorithms like SVM in many areas in small dataset size cases, according to Wainer et. al [27]. Gradient Boosting classifiers, according to Friedman et. al [28], minimize variance and prejudice in classification and improve accuracy as compared to traditional models.

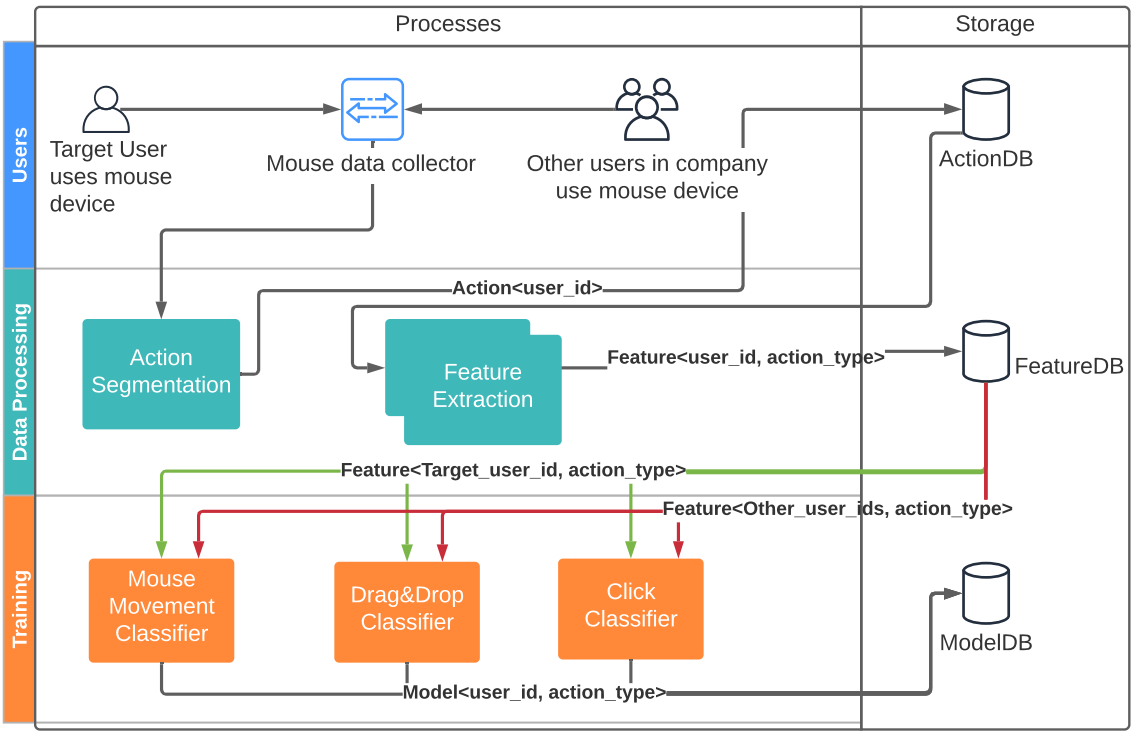


Figure 4.2. Supervised Learning Scheme.

Boosting algorithms are built on the basis of additive modeling. The concept is straightforward: Create a complex function by combining a number of simpler terms. Gradient Boosting combines a variety of simpler models to create a more complex final model. Gradient boosting learns a model by taking the weighted total of a suitable number of base learners.

4.1.1.2. Avoiding Overfitting. Our techniques for resolving the overfitting issue brought on by large data sets are divided into four categories. Some of these include trimming, sampling and early stopping. L2 regularization applies a constant to the second

derivative (Hessian) of the loss function for calculating gain and weight (prediction). L1 regularization employs a constant that is removed from the gradient of the loss function during growth and weight measurements. Like the L2 regularization, it has an impact on the weight scale and the choice of division points for trees. It is also possible to customize the rate at which new information is being absorbed. This parameter multiplies the tree variables by a certain quantity in order to make the model fit more slowly and avoid overfitting (less than one). When a tree is pruned, its entire depth is used. In reality, this is a way of controlling the degree of complexity in each tree, not a pruning technique. Because shallower trees are poorer learners, they are less prone to overfitting. Samples are used to reduce the correlation between our models and the feature filtering effects. Due to their correlation, data and feature sampling become increasingly important. By utilizing the framework we developed, we were able to subsample rows from the training data prior to fitting a novel estimator and create a proportion of features for subsampling at various times during the tree-building process. As soon as the region under ROC on holdout dataset is set to 20% of training data, the ensemble is stopped from being created. We may stop training the model if it becomes overfit.

4.1.2. Self-supervised Learning

The suggested technique enables us to track and verify users' motions when using the mouse, but only using data from the user's usual use which is shown in Figure 2.3. So, self-supervised anomaly detection techniques are utilized, and only a limited quantity of anomalous data acquired from other users is used for parameter tuning. The challenge is referred to as novelty or anomaly detection [18], as well as one-class categorization [29]. We employ these techniques to construct a probability of being an inlier for each action type, and in the following, this idea is referred to as categorization.

Anomaly detection techniques construct a zone within which normal data is constrained and identify everything that does not fall inside that region as an anomaly. Determining the bounds of normal data can provide several difficulties. Because gath-

ered normal data cannot accurately reflect all normal behaviors in the actual world, normal and anomalous data do not share a clear border, impairing classifier performance. Normal data, on the other hand, might alter over time as a result of regular usage and emotional elements. This issue may be addressed by diversifying the data gathering procedure and increasing the amount of normal data collected. Additionally, when data has a high level of noise, it becomes difficult to discriminate between noisy examples and abnormalities.

Novelty detection algorithms that are domain-based (One-Class Support Vector Machines [30]) and distance-based (k-nearest neighbor [31] and Local Outlier Factor [32]) are employed; further information on novelty detection algorithms can be found in [18]. This section delves into the specifics of each action classification classifier. A small amount of data from common and other users is used to determine the threshold for all models. In this way, the performance of each classifier is also measured for the decision making.

4.1.2.1. One Class Support Vector Machines (OCSVM). OCSVM divides data points from the origin into a richer F space via kernel trickery and optimizes the distance between this hyperplane and the origin. The minimization function for quadratic programming is nearly identical to the original suggested SVM [33] as

$$\min_{w,\ \xi_n,\ \rho} \frac{1}{2}\|w\|^2 + \frac{1}{\nu N}\sum_{n=1}^{N}\xi_n - \rho \text{ subject to:} \tag{4.9}$$

$$(w \cdot f(x_n)) \geq \rho - \xi_n \qquad \text{for all } n = 1, \ldots, N \tag{4.10}$$

$$\xi_n \geq 0 \qquad \text{for all } n = 1, \ldots, N \tag{4.11}$$

$$\tag{4.12}$$

and slack variables ξ_n are used to strengthen the classifier's resistance to overfitting with noisy data, and the ν parameter provides an upper constraint on the proportion

of outliers where w and ρ establish a hyperplane with the greatest distance from the origin of F space. The final decision function becomes

$$S(x) = \text{sgn}((w \cdot f(x_n)) - \rho) = \text{sgn}(\sum_{n=1}^{N} \alpha_n K(x, x_n) - \rho) \tag{4.13}$$

when Lagrange methods are used and where α_ns are the Lagrange multipliers.

4.1.2.2. Distance-Based Methods. The distance between two data points is computed using well-defined distance measures. Distance measures are also included in the parameter adjustment for these models. First, we define more general Minkowski distance metric as

$$d_{\text{Minkowski}} = \sqrt[p]{\sum_{i=1}^{N} |x_{ni} - x_{mi}|^p} \tag{4.14}$$

where x_n and $x_m \in \mathbb{R}^N$. Distance metrics used in distance-based methods are $p = 1$, $p = 2$ and $p = \infty$ as Manhattan, Euclidean and Chebyshev distances which are widely use in field.

k-nearest neighbour(k-NN). The k-NN strategy is predicated on the premise that, whereas normal data is anticipated to be close to its neighbors and hence dense, new data is predicted to be remote from its neighbors. The parameters that can be tuned for the k-NN approach are the distance metric, the number of neighbors k to consider when running the algorithm, and the method used to convert k distances to a scalar while searching for the closest neighbors, such as averaging, median, or selecting maximum distances between neighbors.

Local Outlier Factor(LOF). Calculates the local deviation of a data instance's local density in relation to its neighbors. The anomaly score is local, based on the object's isolation from its immediate surroundings. The locality is specified more specifically by k nearest neighbors, which are used to estimate local density. When comparing a data instance's local density to that of its neighbors, outliers have a lower density

than their neighbors. The number of neighbors used to estimate density and distance metrics is two parameters that may be modified for LOF.

4.2. Decision Making

The process of evaluating each test session is distinct from the process of learning. To train behavior classifiers, all data relevant to acts is analyzed as a whole in the learning phase. However, as shown in Figure 4.3, each session and the activities within it are evaluated sequentially and aggregated into a decision-making block in this section. We only display MM, DA, and DC classifiers in this scheme.

This section discusses integrating several probabilities expected from individual actions. Each action taken by a target consumer is given a likelihood score indicating whether or not it is legal, and the final decision is made based on the information gathered from each action. This step is critical because it determines how much of the classifier's knowledge is introduced and how well the target user's legality is determined. The simple average, also known as a linear opinion pool and used by [2] and [5], is the most popular method. We also look at a few more probability aggregation techniques.

We assess a class of pooling functions that use geometric averaging rather than linear averaging. Furthermore, we employ an aggregator that is derived from a log-odds statistical model of the data, which is based on an idealization of the problem and, as a result, an oversimplification. In [34], a related method for averaging expert forecasts to compensate for under- or overconfidence is suggested. Defining

$$\text{logit}(p) = \log\left(\frac{p}{1-p}\right) \tag{4.15}$$

which converts a probability $pin[0,1]$ into a real-valued log-odds. We presume that each classifier predicts legality at the same level of expertise for the sake of derivation simplicity. Let's describe $Y_i = \text{logit}(p_i)$ for $i = 1, \cdots, K$ using each classifier prediction p_i. We also use a Normal distribution to model the log-odds, with the true label of the

prediction log-odd as the center.This makes

$$Y_i = \log\left(\frac{p}{1-p}\right) + \epsilon_i \tag{4.16}$$

where p is the approximate "true likelihood" and $\epsilon_i \sim \mathcal{N}(0, \sigma^2)$ is a random deviation from fact. On σ^2, we don't make any assumptions. In a couple step,

$$\begin{aligned} &\log\left(\tfrac{p_i}{1-p_i}\right) \overset{\text{i.i.d.}}{\sim} \text{Normal}\left(\log\left(\tfrac{p}{1-p}\right)^{1/a}, \sigma^2\right) \\ \Leftrightarrow\ &\tfrac{p_i}{1-p_i} \overset{\text{i.i.d.}}{\sim} \text{Log-Normal}\left(\log\left(\tfrac{p}{1-p}\right)^{1/a}, \sigma^2\right) \\ \Leftrightarrow\ &p_i \overset{\text{i.i.d.}}{\sim} \text{Logit-Normal}\left(\log\left(\tfrac{p}{1-p}\right)^{1/a}, \sigma^2\right). \end{aligned} \tag{4.17}$$

Maximum likelihood estimator of p in Equation (4.17) becomes

$$\hat{p} = \frac{e^{\overline{Y}}}{1 + e^{\overline{Y}}} \tag{4.18}$$

and converts to geometric mean of the odds as

$$\hat{p} = \frac{\left[\prod_{i=1}^{K}\left(\frac{p_i}{1-p_i}\right)^{1/K}\right]}{1 + \left[\prod_{i=1}^{K}\left(\frac{p_i}{1-p_i}\right)^{1/K}\right]}. \tag{4.19}$$

As previously mentioned, this aggregation approach assumes that each classifier performs similarly in terms of prediction. Simply replace and $1/k$ with a weight term w_i and normalize to convert $\hat{p}$ into a more convenient aggregator. The aggregation methods are given in Table 4.1.

Final decision for a test session is done by a threshold λ as

$$\text{Decision} = \begin{cases} \text{Legal}, & \text{if } \hat{p} \geq \lambda \\ \text{Illegal}, & \text{otherwise.} \end{cases} \tag{4.20}$$

When assigning a likelihood score to a test session, the number of actions assumed to be aggregated has a great impact on success. Since we combine the first K actions, the time it takes to complete each verification process increases as K increases. For selecting actions to aggregate in a test session, we limit K with t_{th} denoting a hard time threshold. If t_{th} is 10 seconds, we obtain and analyze the action within 10 seconds. As a consequence, K varies depending on the test session. Thus, one of the goals of this research is to achieve the most reliable verification as soon as possible.

Table 4.1. Aggregation Types and Descriptions.

Aggregation Type	**Description**	**Remarks**
Median		Non-linear combining
Averaging	$\hat{p} = \frac{1}{K}\sum_{i=1}^{K} p_i$	Linear pooling
Weighted Averaging	$\hat{p} = \frac{\sum_{i=1}^{K} w_i p_i}{\sum_{i=1}^{K} w_i}$	Linear pooling
Geometric Mean of Odds	$\hat{p} = \frac{\left[\prod_{i=1}^{K}\left(\frac{p_i}{1-p_i}\right)^{1/K}\right]}{1+\left[\prod_{i=1}^{K}\left(\frac{p_i}{1-p_i}\right)^{1/K}\right]}$	Geometric pooling
Weighted Geometric Mean of Odds	$\hat{p} = \frac{\left[\prod_{i=1}^{K}\left(\frac{p_i}{1-p_i}\right)^{1/w_i}\right]^{1/\sum_{i=1}^{K} w_i}}{1+\left[\prod_{i=1}^{K}\left(\frac{p_i}{1-p_i}\right)^{1/w_i}\right]^{1/\sum_{i=1}^{K} w_i}}$	Geometric pooling

4.2.1. Weight Selection

Voting ensembles are made up of multiple machine learning models, with the predictions from each model being directly averaged. In the case of classification, this could entail determining the statistical mode (most prevalent class label) or a similar voting mechanism, or summing the probabilities predicted for each class and selecting the class with the highest summed probability. The voting ensemble technique has a flaw in that it assumes that all models in the ensemble are equally effective. This may

not be the case because some models may outperform others, particularly if different machine learning techniques are employed to train each model ensemble member. An alternative to voting is to presume that ensemble members are not all equally capable, and that certain models are superior to others and should be given more votes or seats when making a forecast. This explains why the weighted sum or weighted average ensemble method is used.

Weighted arithmetic is comparable to ordinary arithmetic (the most frequent type of average), except that some data points contribute more than others to the final average. Weighted ensembles are a machine learning ensemble strategy that aggregates predictions from several models, with each model's contribution weighted proportionally to its capability or competence. Because each action type has its own classifier and their reliability varies, we employ weighted ensembles with their performances as weights. Furthermore, when calculating the weighted sum for prediction, the voting ensemble for classification may include both hard voting for combining crisp class labels and soft voting for combining class probabilities. In this context, we employ soft voting mechanism.

The receiver operating characteristic (ROC) curve is a popular tool for assessing the performance of binary classification systems [35]. While visualizing a classifier's ROC curve is valuable, in many cases we can reduce this information to a single statistic - the AUC. AUC is an abbreviation for the area under the (ROC) curve. The greater the AUC value, the better a classifier performs for the given task. Area under ROC (AUC) measures how the true positive rate (recall) and false positive rate trade off, therefore it already measures something other than accuracy. More crucially, we do not have to select a particular threshold for evaluating how discriminate a classifier is. It is a classifier evaluation where the threshold varies across all potential values. It is, in some ways, a broader metric, measuring the quality of the internal value generated by the classifier [36]. It is not a test of the quality of a certain threshold decision. This make AUC metric a good candidate on weighting scores while weighting according to performance. In Appendix A, we give a detailed explanation how it is calculated.

In this case, the performance of each ensemble model on the training dataset will be used as the model's relative weighting when making predictions. Area under the ROC Curve as an aggregate measure of performance across all possible classification criteria is used to calculate performance. As mentioned in Sections 4.1.1.2 and 4.1.2, each action model is first fit to the training set and then tested on the validation set. The model weighting is based on the area under the ROC performance on the validation set.

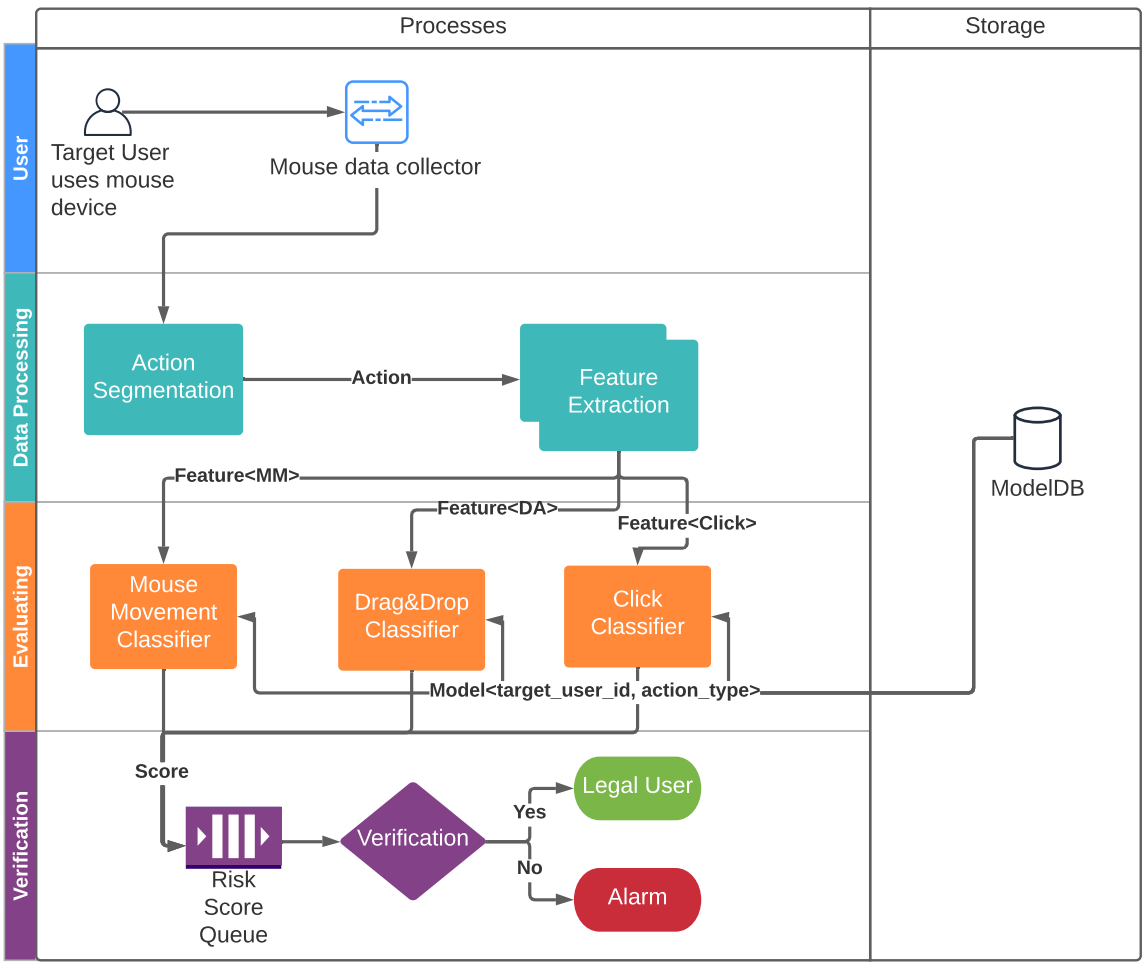


Figure 4.3. Verification Scheme.

5. EXPERIMENTS AND RESULTS

In this chapter, a general biometric system will be tested for insider attack, and the options that may affect the success will be discussed in this chapter. As mentioned, supervised and self-supervised machine learning methods solve different optimization problems. While the first tries to reveal the usage differences between a user and others by creating legal and illegal data within the users, the second tries to understand the usage pattern of a user and checks the similarity of the new data with the one at hand. The first case is a more studied and resolved issue in the field of machine learning. In this respect, the model obtained with supervised learning is expected to be more successful than the self-supervised method.

5.1. Experiment Setup

To evaluate both types of training models in a sufficiently transparent manner, studies were carried out with both the Bogazici Mouse Dynamics dataset and the Balabit Mouse Dynamics dataset. Two types of training models are evaluated under separate experiments. Under each experiment, the approaches carried out to reach the highest performance within the training methods and the effects of these approaches on the performance are shown for insider attacks.

The Bogazici Mouse Dynamics dataset and the Balabit Mouse Dynamics dataset contain 24 and 10 users, respectively. Each user has an average of 5 and 80 hours of train sessions, an average of 40 and 3000 test sessions, respectively. Each user has their classifiers, which are independently trained after the action segmentation and feature extraction parts. Each user has an average of 30500 and 420000 actions in training sets, respectively. Within the training part, the best estimators for each action type are selected and their parameters are tuned. We use the created validation set to tune and select the best estimator and used AUC as a performance indicator. These models are saved as shown in Figure 4.2. Then, the verification part in Figure

4.3 is handled. First, actions are segmented and features are extracted from the test sessions. Then, the trained models are used for predicting the legality score of each action in a test session. Then, each aggregator in Section 4.2 is used to produce a single risk score for a single test session. Since the evaluation time of a session is also considered, the actions used in the aggregation are determined by t_{th}, which is the evaluation time of a session. t_{th} is the main parameter that presents the trade-off between evaluation time and performance. Since each test session is labeled legal or illegal, we try to distinguish these classes by the risk scores that the aggregators produce. The performance of determining a test session as legal or illegal, the AUC performance is also used. We present our findings in terms of t_{th} and aggregator type for each dataset. The performance of each biometric modeling belonging to each user is analyzed separately, and then the total performance is determined as the average of their performance numbers.

All thresholds used in action segmentation in Section 3.1.1 are determined experimentally and chosen as 500 ms except η_{DC} which is chosen as 300 ms.

5.1.1. Evaluation Metrics

The evaluation findings are reported as area under the receiver operating characteristic curve (AUC) and Equal Error Rate (EER).

5.2. Comparison of Proposed Methods

In this section, we discuss self-supervised learning and supervised learning models separately.

5.2.1. Supervised Learning

The suggested technique is assessed using the data by doing the following experiments:

(i) Given verification time, comparison between the proposed action aggregation approaches in Section 4.2 to the *simple averaging* proposed by [5] and [2]

(ii) Contribution of the frequency domain feature analysis introduced in Section 3.3.2 to the verification accuracy.

Table 5.1. Performances of aggregation methods according to verification time on the Balabit dataset.

	t_{th}=5 secs		t_{th}=20 secs		t_{th}=60 secs		t_{th}=120 secs		t_{th}=240 secs	
Aggregation Methods	AUC	EER	AUC	EER	AUC	EER	AUC	EER	AUC	EER
Weighted Averaging	82.04%	25.28%	83.18%	22.39%	90.14%	17.06%	94.21%	8.11%	96.16%	7.05%
Weighted Geometric Mean of Odds	82.93%	22.77%	84.25%	22.12%	91.19%	13.75%	94.83%	9.81%	96.47%	7.462%
Averaging	82.04%	25.28%	83.15%	22.49%	90.11%	16.88%	94.21%	8.11%	96.19%	7.056%
Median	82.10%	23.78%	83.33%	23.89%	89.07%	15.97%	92.80%	12.03%	94.38%	10.32%
Geometric Mean of Odds	82.65%	24.77%	84.16%	22.90%	91.15%	13.53%	94.81%	9.633%	96.50%	8.063%

Table 5.2. Performances of aggregation methods according to verification time on the Bogazici dataset.

	t_{th}=5 secs		t_{th}=20 secs		t_{th}=60 secs		t_{th}=120 secs		t_{th}=240 secs	
Aggregation Methods	AUC	EER	AUC	EER	AUC	EER	AUC	EER	AUC	EER
Weighted Averaging	80.21%	27.63%	82.73%	25.57%	89.72%	19.98%	91.65%	9.81%	92.28%	11.26%
Weighted Geometric Mean of Odds	82.49%	23.82%	85.09%	22.95%	90.14%	15.94%	92.89%	10.89%	95.65%	7.56%
Averaging	79.77%	27.2%	81.84%	23.82%	89.41%	18.38%	92.5%	9.02%	95.12%	8.52%
Median	80.49%	26.49%	81.49%	25.36%	87.77%	16.99%	91.17%	12.29%	93.0%	11.36%
Geometric Mean of Odds	81.8%	26.99%	82.22%	24.24%	89.75%	15.76%	94.64%	11.33%	95.65%	9.14%

As it is mentioned, the Balabit Mouse Dynamics dataset and Bogazici Mouse Dynamics dataset is used in this simulation. For the Balabit Mouse Dynamics dataset, the public data set comprises 10 users; during the training phase, we label each user as an "internal" user. In other words, each user is stressed as a 1-target-user-9-illegal-user. In total, ten models are trained and assessed. For the Bogazici Mouse Dynamics

dataset, the dataset comprises 24 users, during the training phase, 24 models are trained as 1-target-user-23-illegal-user and assessed for each user. Testing is done with internal attack test data for each user in the dataset.

5.2.1.1. Comparison with the simple averaging aggregation. The works by [5] and [2] provide ways that utilized individual action categorization and advocated simple averaging to aggregate the legality probability of each activity. The experiments on various aggregating strategies are described in Section 4.2. The AUC and EER values in Table 5.1 and 5.2 are calculated using the t_{th} threshold. When compared to geometric pooling and weighting, just averaging the probability estimates leads to worse performance. Utilizing the classifier's performance for weighting and log-odds statistics improves the performance of the whole model. At best, the achieved performance for the Balabit Mouse Dynamics dataset is EER with a value of 7.46% and AUC with a value of 96.47%, for the Bogazici Mouse Dynamic dataset is EER with a value of 7.56% and AUC with a value of 95.65%.

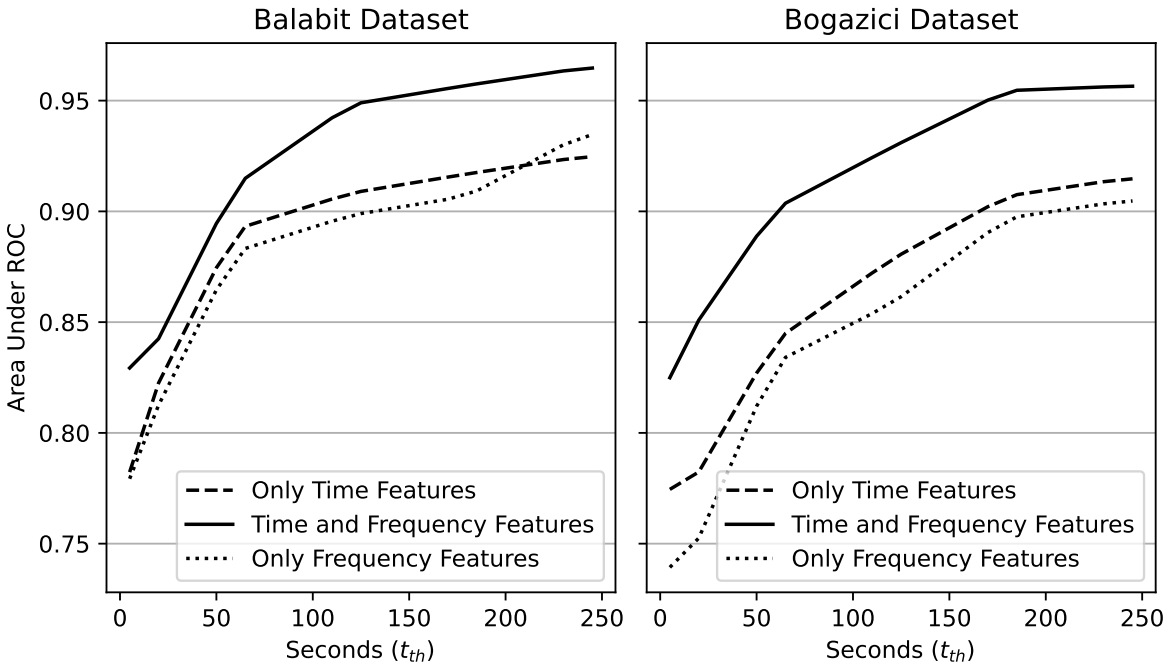


Figure 5.1. Contribution of new features on performance in terms of AUC.

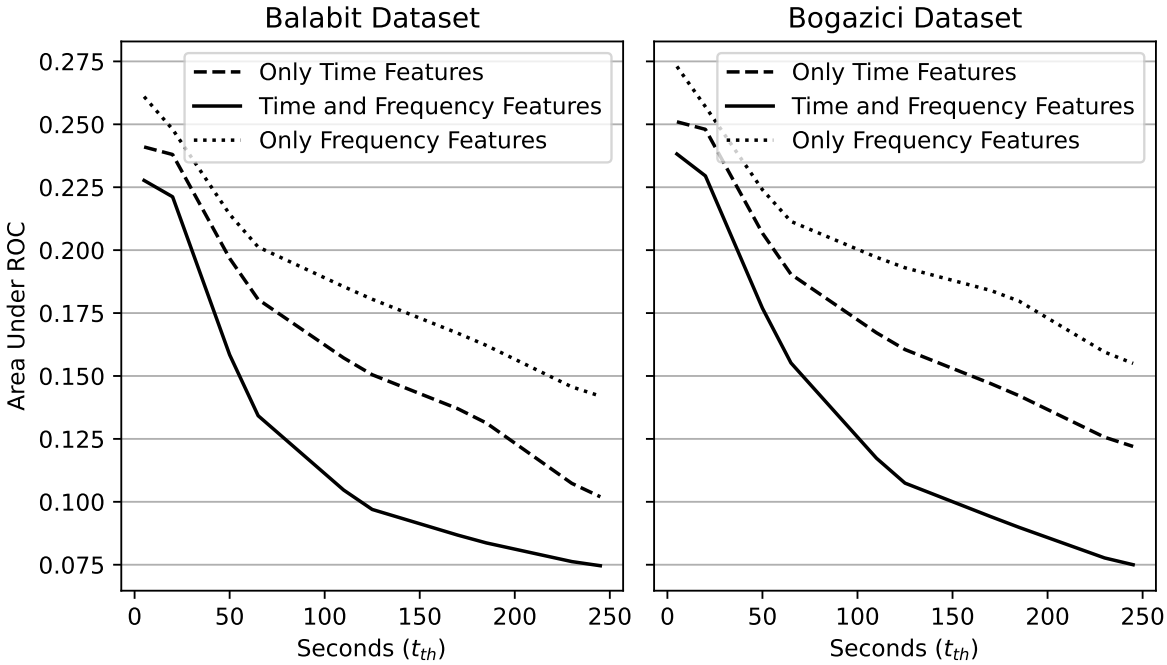


Figure 5.2. Contribution of new features on performance in terms of EER.

5.2.1.2. Contribution of new features. The frequency-domain analysis technique simplifies the process of deciphering the pattern of user behavior. By definition, it enables the interpretation of periodic motions and the incorporation of this information into the feature vector. As a result, it results in a more precise understanding of user behavior. The Figures 5.1 and 5.2 illustrate the frequency domain analysis alone, the time domain statistical analysis alone, and the combined experiments using the weighted geometric mean of odds as the aggregation type. As discussed in Section 4.1.1, it is noticed that safely combining two different types of feature extraction methods improves performance. Additionally, it is demonstrated that the proposed frequency domain analysis produces acceptable results on its own. Within each t_{th} threshold, a significant improvement is seen in the performance of the suggested model.

5.2.2. Self-supervised Learning

The suggested technique is assessed using the data by doing the following experiments:

(i) Given verification time, the performance comparison between novelty detection algorithms.

(ii) Contribution of the Kernel PCA analysis introduced in Section 3.31 to the verification accuracy.

In this section, the contribution of the feature domain feature analysis is not presented due to avoiding duplication. The performance indicators belonging to the core models are investigated instead. In the following experiments, the weighted geometric mean of odds as the aggregation type and frequency and time domain feature analysis is used as a feature extractor. The Balabit Mouse Dynamics dataset and Bogazici Mouse Dynamics dataset is used in this simulation. For the Balabit Mouse Dynamics dataset, the public data set comprises 10 users; during the training phase, we label each user as an internal user. In the experiments, unlike the supervised learning scheme, each user's behavior model was trained only with their data. Training models based on probability estimation obtained with the amount of similarity between the learned pattern and the data whose class is tried to be determined are used. In this respect, 10 models were created by training 10 users with their data and defining the thresholds with the data reserved for validation. For the Bogazici Mouse Dynamics dataset, the dataset comprises 24 users, during the training phase, 24 models are trained as the Balabit Mouse Dynamics dataset and assessed for each user. Testing is done with internal attack test data for each user in the dataset.

Table 5.3. Performances of the methods on the Balabit dataset.

	t_{th}=5 secs		t_{th}=20 secs		t_{th}=60 secs		t_{th}=120 secs		t_{th}=240 secs	
Training Methods	AUC	EER	AUC	EER	AUC	EER	AUC	EER	AUC	EER
OCSVM	62.01%	37.18%	67.16%	34.43%	69.06%	33.91%	70.59%	31.06%	75.97%	28.88%
KNN	59.64%	38.34%	65.02%	35.82%	68.97%	34.46%	69.99%	32.61%	71.75%	30.96%
LOF	63.34%	35.29%	68.38%	30.37%	72.21%	28.75%	75.86%	26.32%	82.9%	22.98%

Table 5.4. Performances of the methods on the Bogazici dataset.

	t_{th}=5 secs		t_{th}=20 secs		t_{th}=60 secs		t_{th}=120 secs		t_{th}=240 secs	
Training Methods	AUC	EER	AUC	EER	AUC	EER	AUC	EER	AUC	EER
OCSVM	64.23%	35.28%	69.55%	32.77%	71.71%	33.35%	73.19%	30.16%	77.07%	28.87%
KNN	61.44%	36.77%	65.12%	33.97%	70.81%	33.77%	71.75%	32.19%	72.15%	30.13%
LOF	65.34%	32.99%	70.21%	29.22%	74.11%	26.15%	78.23%	24.13%	83.77%	23.15%

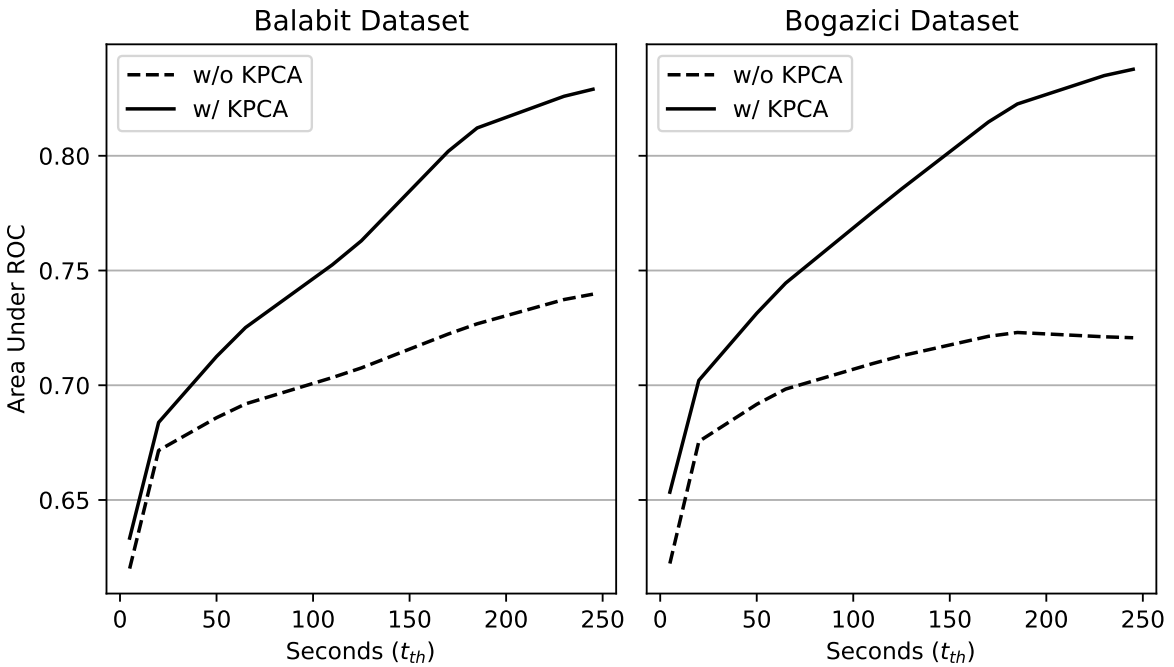


Figure 5.3. Contribution of the Kernel PCA on performance in terms of AUC.

5.2.2.1. Performance of the classifiers. In this part, we evaluate 3 novelty detection algorithms as mentioned in Section 4.1.2. OCSVM, KNN, and LOF were used in experiments using Balabit Mouse Dynamics dataset and Bogazici Mouse Dynamics dataset. Each model is trained to get the best parameters in itself. For OCSVM, ν parameter in Equation (4.9) is set. For KNN, the best distance metric and k number were selected. For LOF, the best distance metric, number of neighbors, and leaf size were selected. The AUC and EER values in Table 5.3 and 5.4 are calculated using the t_{th} threshold. At best, the achieved performance for the Balabit Mouse Dynamics dataset is EER with a value of 23.3% and AUC with a value of 82.9%, for the Bogazici

Mouse Dynamic dataset is EER with a value of 23.15% and AUC with a value of 83.77%. This experiment states that the Local Outlier Factor classifier outperforms the others.

5.2.2.2. Contribution of the Kernel PCA. By encoding data in a richer space and utilizing principal components, the kernel PCA approach is utilized to accommodate user behavior variability. In experiments, KPCA's parameters like kernel type, and individual parameters belonging to parameters are searched via grid search. The best results come with the RBF kernel. Readers should be careful about OCSVM kernels and Kernel PCA. Even if they have the same purposes which are projecting the feature space into a richer one, using the Kernel PCA before OCSVM with the kernel is still useful since it reduces the dimension of the feature set and makes the search for a distinctive hyperbole easy. As illustrated in Figure 5.3, Kernel PCA strengthens our model's resistance to user behavior variability and outperforms models trained without Kernel PCA projection.

5.3. Discussion

In order for users to have a safer working environment, we have completed our work on a passive security layer that does not directly affect the user. These studies basically enable us to keep security problems such as stealing passwords and phishing under control, without reducing the quality of the end user's usage experience. In this respect, behavioral biometrics plays an important role in designing a robust security layer by using subconscious mouse usage patterns that cannot be imitated even if desired. The aim of the thesis is to investigate a security layer that does not affect usage performance in the fastest and safest way and to show its feasibility. Contributions to this research area have been successfully completed by demonstrating a time and performance that is ahead of previous studies. Within the framework of companies and changing data protection laws, different types of training were tried and their achievements were shared. Self-supervised methods are also included in the thesis in order to overcome the security cost of collecting the user's sensitive data in a central

environment and providing training. Thus, the pros and cons of the trainer are shown without the need to collect only the user's own data into a central data repository, albeit with lower performance. The supervised learning method, which is higher performing and reduces attack vectors for the company, has achieved high safety performance results for most users, along with centralized data storage. However, it has been clearly observed in the obtained success rates that this authentication method can only help two-factor authentication methods. In this respect, with the development of data analysis methods and feature extraction methods in non-table data, it is inevitable for most large companies to use this technology.

Our contributions to this field are an improved feature extraction flow, the use of different machine learning methods for different usage stories within the framework of data protection laws and the sharing of their performance, and the use of logit analytics, which has not been used in this field before, as a performance enhancer to combine the information of multiple detecters. The AUC above 95% and EER below 8%, which we achieved in the supervised models, is the best performance achieved in this area. It is especially important that we achieve these success rates faster than other studies. Our comparison with other studies includes other studies in the literature that used the Balabit Mouse Dynamics dataset. Detailed information on this subject can be found in Table 1.1. With the development of big data processing technologies in the last 15 years, the increase in success rates can be seen in other studies.

Among other studies using the Balabit Mouse Dynamics dataset, our success rates are very close to those using curve visualization and deep learning (2D-CNN, etc.) methods. The number of these studies will increase in the coming years. Nonetheless, deep learning models capable of establishing higher-level associations, such as the sequence of emergence of subsequent types of movements, can be learned. This was not the case in our investigation. Although it is frequently impractical to collect huge amounts of data for each user in real life, the frequency and interval of succeeding action types in training sets can be added in the visualization with a few additional data points. While gradient boosting decision trees perform well on small data sets, it

is worth noting that, like deep learning approaches, their performance improves as the volume of data rises.

On the other hand, the solution we suggest is easily accessible at the product level and can be used in conjunction with Apache Spark to establish a unified pipeline. As a result, the latency introduced by visualization-based deep learning techniques is decreased. If the computations are performed on the local system, the time between the user completing the movement and generating a score is less than 2 milliseconds. The computations are performed using the Intel Core i7-8550U processor's single core. This is five times faster than techniques based on deep learning. Additionally, prior research on this subject has reported verification times that are multiples of the average duration of the actions. However, in real-world systems, the assumption of constant user activity is incorrect. The user might begin an activity and then pause for a bit before resuming the activity. To this end, we developed a verification time benchmark that can be used for real-world systems described in the literature.

6. CONCLUSION

Banks and commercial establishments, whose numbers are increasing day by day, identify customers in order to ensure information security with the data created by collecting small and sensitive physical movements such as users' touching electronic devices, swiping the screen, using the mouse, through computer programs. This method, which is called behavioral biometrics, it is tried to prevent cyber theft, such as the seizure of passwords and personal information, which is experienced almost every day. The hypothesis that users have unique use of the mouse device and that this interaction can be used for security purposes forms the basis of this thesis.

The presented thesis contributes to the field of behavioral biometrics with signal processing, frequency domain analysis, and advanced data processing techniques. Together with the defined insider attack vectors, the largest publicly available dataset has been prepared to test these attacks. Examining the user's periodic behavior in the frequency domain is one of the contributions to the performance. Next, two different types of use cases are investigated in order to profile the mouse usage pattern of the users. The use cases are implemented using the self-supervised training model, which is trained using only the user's own data, and the supervised training model, which mutually trains the data of the employees. From a technical point of view, it has been observed that the performance rates of the self-supervised methods by using only the user's own data remain low compared to the supervised learning methods. This result shows that supervised training models solve the more robustly defined problem. Also, in contrast to prior research that used statistical aggregation and simple averaging methods, we increased our performance by looking at the legality estimations in a different space. As a consequence, we are able to reach equivalent authentication performance in a shorter time window than the other works.

The proposed biometric system models and data analysis techniques suitable for these models were examined and their individual performances were measured. Under

these conditions, a high rate of success was achieved against the insider threat. In this way, it has been shown that the proposed system can work in integration with an extra layer of security and Privileged Access Management products.

REFERENCES

1. Grim, E. T., "Two-factor Authentication Systems and Methods", U.S. Patent US8578454B2, November 2013, available: https://patents.google.com/patent/US8578454B2.

2. Antal, M. and E. Egyed-Zsigmond, "Intrusion Detection Using Mouse Dynamics", *IET Biometrics*, Vol. 8, No. 5, pp. 285–294, 2019.

3. Chong, P., Y. X. M. Tan, J. Guarnizo, Y. Elovici and A. Binder, "Mouse Authentication Without the Temporal Aspect – What Does a 2D-CNN Learn?", *2018 IEEE Security and Privacy Workshops (SPW)*, pp. 15–21, 2018.

4. Chong, P., Y. Elovici and A. Binder, "User Authentication Based on Mouse Dynamics Using Deep Neural Networks: A Comprehensive Study", *IEEE Transactions on Information Forensics and Security*, Vol. 15, pp. 1086–1101, 2019.

5. Feher, C., Y. Elovici, R. Moskovitch, L. Rokach and A. Schclar, "User Identity Verification via Mouse Dynamics", *Information Sciences*, Vol. 201, pp. 19–36, 2012.

6. Zheng, N., A. Paloski and H. Wang, "An Efficient User Verification System via Mouse Movements", *Proceedings of the 18th ACM Conference on Computer and Communications Security*, pp. 139–150, 2011.

7. Zheng, N., A. Paloski and H. Wang, "An Efficient User Verification System Using Angle-based Mouse Movement Biometrics", *ACM Transactions on Information and System Security (TISSEC)*, Vol. 18, No. 3, pp. 1–27, 2016.

8. Ahmed, A. A. E. and I. Traore, "A New Biometric Technology Based on Mouse Dynamics", *IEEE Transactions on Dependable and Secure Computing*, Vol. 4, No. 3, pp. 165–179, 2007.

9. Gamboa, H. and A. Fred, "A Behavioral Biometric System Based on Human-computer Interaction", *Biometric Technology for Human Identification*, Vol. 5404, pp. 381–393, 2004.

10. Schulz, D. A., "Mouse Curve Biometrics", *Biometrics Symposium: Special Session on Research at the Biometric Consortium Conference*, pp. 1–6, 2006.

11. Zhong, Y. and Y. Deng, "A Survey on Keystroke Dynamics Biometrics: Approaches, Advances, And Evaluations", *Recent Advances in User Authentication Using Keystroke Dynamics Biometrics*, 1, pp. 1–22, Science Gate Publishing, 2015.

12. Hinbarji, Z., R. Albatal and C. Gurrin, "Dynamic User Authentication Based on Mouse Movements Curves", *International Conference on Multimedia Modeling*, pp. 111–122, 2015.

13. Fülöp, Á., L. Kovács, T. Kurics and E. Windhager-Pokol, *Balabit Mouse Dynamics Challenge Data Set*, 2016, `https://github.com/balabit/Mouse-Dynamics-Challenge`, accessed in December 2021.

14. Wayman, J. L., "Fundamentals of Biometric Authentication Technologies", *International Journal of Image and Graphics*, Vol. 1, No. 01, pp. 93–113, 2001.

15. Kılıç, A. A., M. Yıldırım and E. Anarım, "Bogazici Mouse Dynamics Dataset", *Data in Brief*, Vol. 36, p. 107094, 2021.

16. Izurieta, S., *Survey: Insider Threats Haunt Security Execs*, 2019, `https://bit.ly/3yLctxn`, accessed in December 2021.

17. Coles-Kemp, L. and M. Theoharidou, "Insider Threat and Information Security Management", *Insider Threats in Cyber Security*, pp. 45–71, Springer, 2010.

18. Pimentel, M. A., D. A. Clifton, L. Clifton and L. Tarassenko, "A Review of Novelty

Detection", *Signal Processing*, Vol. 99, pp. 215–249, 2014.

19. Casey, J., *Exploring Curvature*, Springer Science & Business Media, 2012.

20. Scargle, J. D., "Studies in Astronomical Time Series Analysis. II-Statistical Aspects of Spectral Analysis of Unevenly Spaced Data", *The Astrophysical Journal*, Vol. 263, pp. 835–853, 1982.

21. Lomb, N. R., "Least-squares Frequency Analysis of Unequally Spaced Data", *Astrophysics and Space Science*, Vol. 39, No. 2, pp. 447–462, 1976.

22. Press, W. H. and G. B. Rybicki, "Fast Algorithm for Spectral Analysis of Unevenly Sampled Data", *The Astrophysical Journal*, Vol. 338, pp. 277–280, 1989.

23. VanderPlas, J. T. and Ž. Ivezic, "Periodograms for Multiband Astronomical Time Series", *The Astrophysical Journal*, Vol. 812, No. 1, p. 18, 2015.

24. Schölkopf, B., A. Smola and K.-R. Müller, "Kernel Principal Component Analysis", *International Conference on Artificial Neural Networks*, pp. 583–588, 1997.

25. Lin, H.-T. and C.-J. Lin, "A Study on Sigmoid Kernels for SVM and the Training of Non-PSD Kernels by Smo-type Methods", *Submitted to Neural Computation*, Vol. 3, No. 1-32, p. 16, 2003.

26. Boughorbel, S., J.-P. Tarel and N. Boujemaa, "Conditionally Positive Definite Kernels for SVM Based Image Recognition", *IEEE International Conference on Multimedia and Expo*, pp. 113–116, 2005.

27. Wainer, J., "Comparison of 14 Different Families of Classification Algorithms on 115 Binary Datasets", *arXiv preprint arXiv:1606.00930*, 2016.

28. Friedman, J. H., "Stochastic Gradient Boosting", *Computational Statistics & Data Analysis*, Vol. 38, No. 4, pp. 367–378, 2002.

29. Tax, D. M. J., *One-class Classification: Concept Learning in the Absence of Counter-examples*, Ph.D. Thesis, Technische Universiteit Delft, 2001.

30. Schölkopf, B., R. C. Williamson, A. Smola, J. Shawe-Taylor and J. Platt, "Support Vector Method for Novelty Detection", *Advances in Neural Information Processing Systems*, Vol. 12, MIT Press, 2000.

31. Ramaswamy, S., R. Rastogi and K. Shim, "Efficient Algorithms for Mining Outliers from Large Data Sets", *Proceedings of the 2000 ACM Sigmod International Conference on Management of Data*, pp. 427–438, 2000.

32. Breunig, M. M., H.-P. Kriegel, R. T. Ng and J. Sander, "LOF: Identifying Density-Based Local Outliers", *SIGMOD Rec.*, Vol. 29, No. 2, p. 93–104, 2000.

33. Suykens, J. A. and J. Vandewalle, "Least Squares Support Vector Machine Classifiers", *Neural Processing Letters*, Vol. 9, No. 3, pp. 293–300, 1999.

34. Wang, G., S. Kulkarni, H. V. Poor and D. N. Osherson, "Improving aggregated forecasts of probability", *2011 45th Annual Conference on Information Sciences and Systems*, pp. 1–5, 2011.

35. Bradley, A. P., "The Use of the Area Under the Roc Curve in the Evaluation of Machine Learning Algorithms", *Pattern Recognition*, Vol. 30, No. 7, pp. 1145–1159, 1997.

36. Fawcett, T., "An Introduction to ROC Analysis", *Pattern Recognition Letters*, Vol. 27, No. 8, pp. 861–874, 2006.

37. Marzban, C., "The ROC Curve and the Area Under It as Performance Measures", *Weather and Forecasting*, Vol. 19, No. 6, pp. 1106–1114, 2004.

38. Candy, J. and E. Breitfeller, *Receiver Operating Characteristic (ROC) Curves: An Analysis Tool for Detection Performance*, Tech. rep., Lawrence Livermore National

Lab.(LLNL), Livermore, CA (United States), 2013.

39. Yeh, S., "Using a Trapezoidal Rule for the Area Under a Curve Calculation– SAS Advanced Tutorial", *Proceedings of the 27th Annual Conference of SAS Users Group International*, pp. 14–17, 2002.

APPENDIX A: Area Under ROC Calculation

The ROC curve is constructed by graphing the true positive rate (TPR) vs the false positive rate (FPR) at various threshold levels [37]. The true-positive rate is often referred to as sensitivity, recall, or detection probability. The false-positive rate, also known as the likelihood of false alarm, is computed as (1-specificity). It can also be viewed as a plot of the power as a function of the decision rule's Type I Error (when the performance is calculated from just a sample of the population, it can be thought of as estimators of these quantities). The ROC curve tries to explain how well (or poorly) a binary classifier is performing. There are a number of reasons why a simple confusion matrix is insufficient for testing the models. Nonetheless, the ROC representation addresses the following quite well: the ability to establish more than one threshold in a single image.

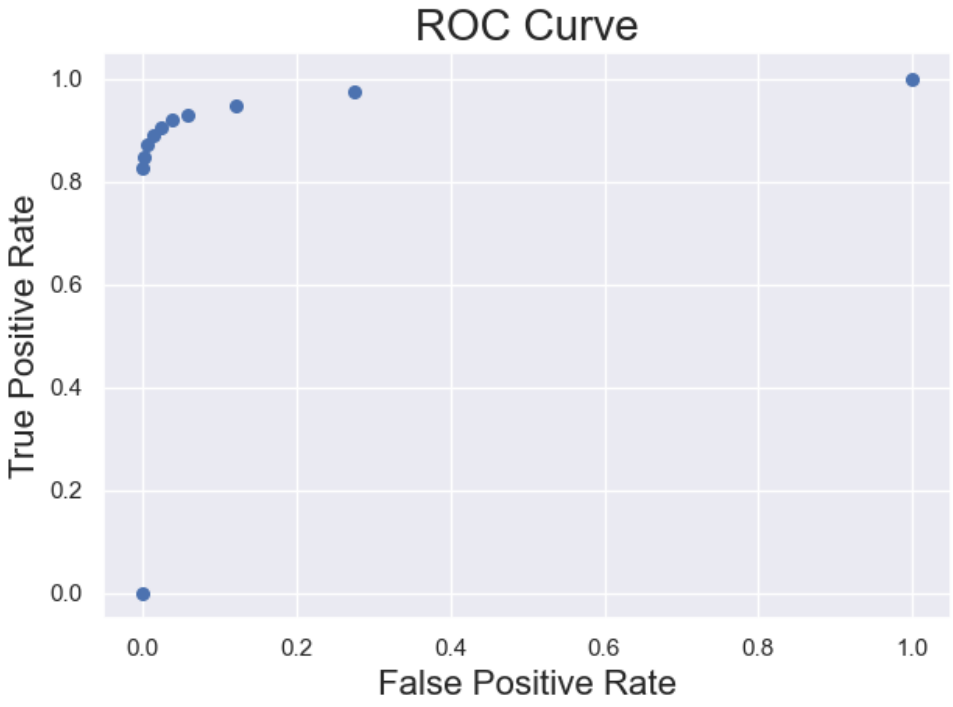


Figure A.1. ROC with 10 Partition.

We use brute force methodology explained in [38]. There are 2 main steps while plotting the ROC curve. For an example for the steps described, we use a target user from Balabit Dataset and its mouse-movement classifier. It has 2507 test datapoints i.e predictions.

Step 1. Choosing a Threshold. As we discussed earlier, the ROC curve's whole idea is to check out different thresholds. Since we conduct threshold checking in probability space, uniform spacing between 0 and 1 is a simple yet efficient method. The number of partition we select is directly affecting the approximation quality of work of the curve. This can be further optimized into sorting the predictions and use them as thresholds [36].

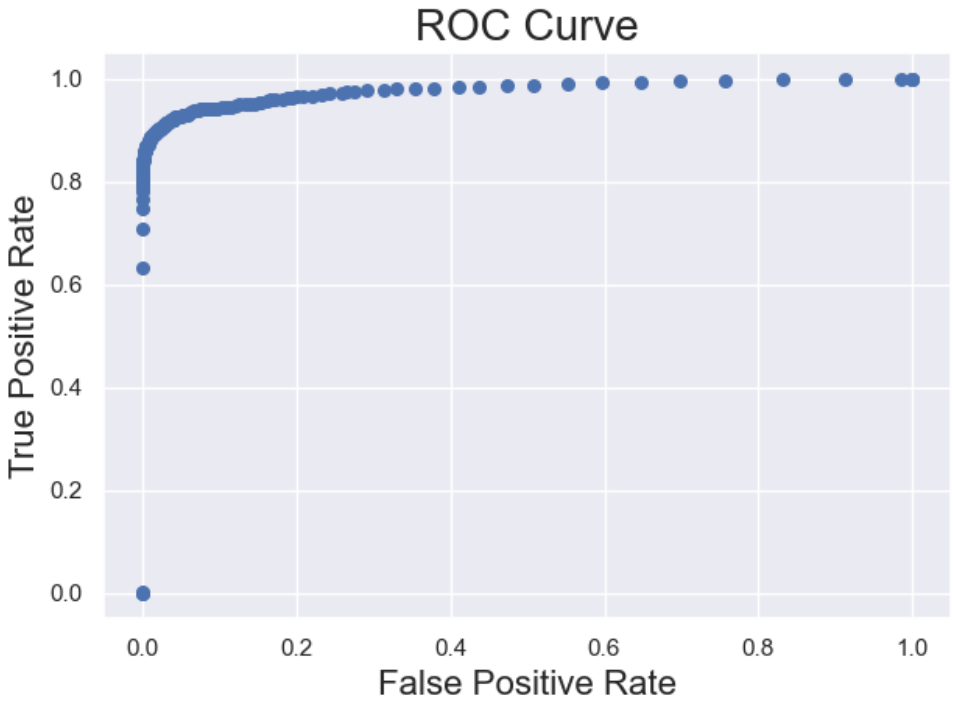


Figure A.2. ROC with 200 Partition.

Step 2. Calculating TPR and FPR. We calculate the TPR and FPR at every threshold on the predictions. In Figure A.1, 10 thresholds are evaluated in terms of TPR and FPR and plotted accordingly. Increasing the number of partition to 100

makes ROC curve estimation better as can be seen in Figure A.2. True Positive Rate (TPR) tells us what proportion of the positive class got correctly classified as

$$\text{TPR} = \frac{TP}{TP + FN}$$

and False Positive Rate (FPR) tells us what proportion of the negative class got incorrectly classified by the classifier as

$$FPR = \frac{FP}{TN + FP}$$

where TP is true positives, FP is false positives, FN is false negatives, TN is true negatives.

The area under the curve in the ROC graph is the primary metric to determine if the classifier is doing well. The higher the value, the higher the model performance. This metric's maximum theoric value is 1, but it's usually a little less than that. The AUC can be calculated for functions using the integral of the function between 0 and 1. But in this case, it's not that simple to create a function. Nonetheless, a good approximation is to calculate the area, separating it into smaller pieces using Trapezoidal rule [39]. The trapezoidal rule is a technique for approximating the definite integral in mathematics, and more specifically in numerical analysis. The trapezoidal rule works by approximating the region under the graph of the function $f(x)$ as a trapezoid and calculating its area. Since the grid spacing is non-uniform due to TPR/FPR results, the formula

$$\int_a^b f(x)dx \approx \sum_{k=1}^{N} \frac{f\left(x_{k-1}\right) + f\left(x_k\right)}{2} \Delta x_k \tag{A.1}$$

can be used. This becomes a linear algorithm in terms of time complexity. On the calculated ROC curve, the trapezoidal calculation with few partition can be seen on Figure A.4. When the number of partition is increased, the approximation becomes better.

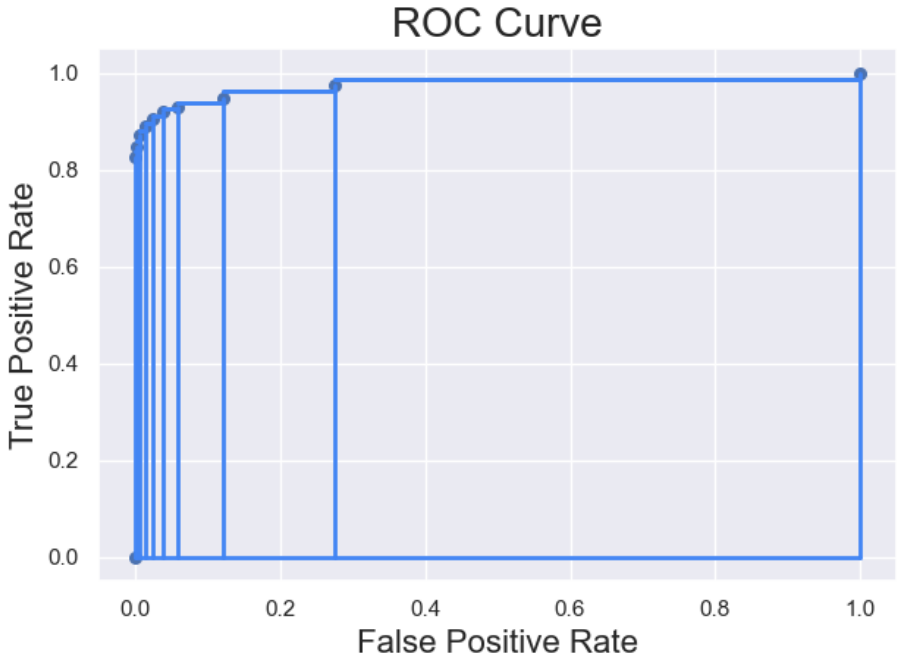


Figure A.3. The trapezoidal calculation of Area Under ROC with 10 Partition.

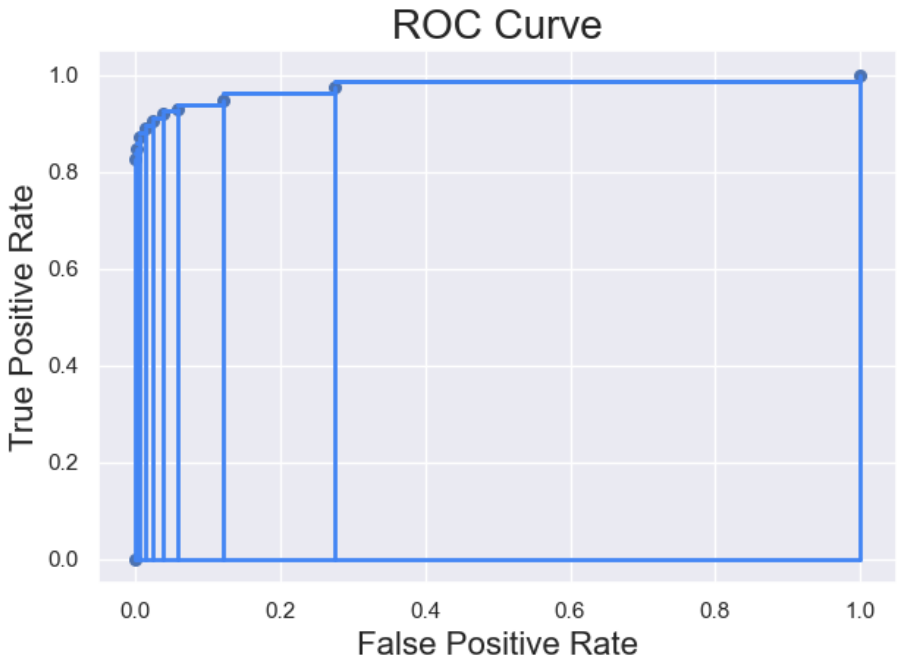


Figure A.4. The trapezoidal calculation of Area Under ROC with 10 Partition.