Behavioral Biometrics-based Continuous User Authentication

Sanket Vilas Salunke, *The University of Western Ontario*

Supervisor: Ouda, Abdelkader, *The University of Western Ontario*

Abstract

The field of cybersecurity is exploring new ways to defend against cyber-attacks, including a technique called continuous user authentication. This method uses keystroke (typing) data to continuously match the user's typing pattern with patterns previously recorded using artificial intelligence (AI) to identify the user. While this approach has the potential to improve security, it also has some challenges, including the time it takes to register a user, the performance of machine learning algorithms on real-world data, and latency within the system. In this study, the researchers proposed solutions to these issues by using transfer learning to reduce user registration time, testing machine learning algorithms on real-world data, and developing a universal benchmarking framework to evaluate databases in practical situations. The results of the experiments supported the researchers' observations and suggestions for improving continuous user authentication.

Keywords

Transfer learning, Behavioral biometrics, Cybersecurity, Continuous authentication, Ensemble learning, Keystroke data, XGBoost, TabNet, LightGBM, Database benchmarking, PostgreSQL, Mysql.

Summary for Lay Audience

Modern systems require robust cybersecurity solutions. Traditional authentication methods like passwords, fingerprints, authorization cards, etc. authenticate the user at the beginning of the session but there is no validation during the session, which makes the system vulnerable. Continuous authentication is the solution to this challenge. In continuous authentication, keystroke data is used to extract the behavior patterns of the user. The data is then applied to train the machine learning (ML) classification algorithms to identify the unique behavioral patterns of each user and classify them accordingly. However, using continuous authentication comes with different challenges. First, it required a long registration time because ML algorithms require a lot of data to find the user's behavioral pattern, and plenty of time is required to gather the data which extends the start of continuously authenticating the new user. Therefore, the transfer learning technique was used for a feed-forward neural network model to overcome this issue for new users. Besides this, the performance of the ML classification algorithm is key in continuous user authentication, and it requires diverse and comprehensive data to be effective in the production environment. In many cases, the ML algorithm is trained on the datasets collected in a controlled lab environment and the model fails or does not perform as expected in the production environment. For example, China’s facial recognition system recognized the face on a bus advertisement as a jaywalker because the model was not trained on real-world data. To overcome this problem, this study uses the real-world data of 48 financial organizations’ employees to compare the performance of advanced ML algorithms and ensembles of algorithms. Next, data latency is critical in continuous authentication as millions of records are required to be managed by the database and its performance has a great influence on the continuous authentication process. Hence it is necessary to identify the leading database for a continuous authentication system. Therefore, to evaluate different databases a universal database benchmarking tool is developed, and the performance of MySQL and PostgreSQL is evaluated in production-like scenarios to determine the best-suited database for a continuous authentication system.

Acknowledgments

I thank my supervisor Dr. Abdelkader Ouda for his guidance, support, and mentorship through thick and thin times over the course of the program. Also, appreciate his management and leadership skills that helped me set the research vision. Moreover, this research was conducted in collaboration with our industry partner F8th Inc and Scrawlr Development Inc. I thank our co-workers from F8th Inc and Scrawlr Development Inc who provided resources, expertise, and insights that greatly helped this research work.

Table of Contents

List of Tables

List of Figures

List of Acronyms

Abbreviation	**Meaning**
ML	Machine Learning
AI	Artificial Intelligence
IoT	Internet of Things
SVM	Support Vector Machine
FRR	False rejection Rate
FAR	False Acceptance Rate
IQR	Interquartile range

Chapter 1

1 Introduction

The issue of cybersecurity is growing exponentially with the increasing number of devices. These devices have become an inseparable part of our lives, hence, making us more vulnerable to cyberattacks. Around 100 billion dollars are lost every year to cybercrime, and it is estimated to reach 10.5 trillion by 2025 [1].

The most important aspect of cybersecurity is to authenticate the legitimate user. As per Netsec News, 67% of breaches are caused by credential theft [2]. One of the recent events that happened was with outdoor retailer The North Face, and their customers' accounts were hacked to steal their data like credit card details, phone numbers, etc. Therefore, it is crucial to restrict access to authorized users only. Different methods like biometric authentication using fingerprint/face, password, and authorization cards, etc. are currently used to authenticate the user but it is not sufficient to use two-factor or multi-factor authentication as it only provides static user authentication. The potential solution for this is continuous authentication, which is a method to verify users' identities on an ongoing basis by using behavioral biometrics. Every user has unique behavioral patterns, like the way a user handles a mouse, keyboard, and touchscreen device. These behavioral patterns can be used to build a user profile and continuously authenticate them by using machine learning algorithms. This will not only provide continuous user authentication but also allow users to continue their work without disruption.

1.1 Research Motivation

Technology affects almost every aspect of 21st-century life, from socialization and healthcare to access to food and transport efficiency. Technology and the internet revolution have enabled global communities to be connected and share resources more easily. Most people on average have at least 2 to 3 devices connected to the internet. It is estimated that the total number of connected devices will rise to more than 75 billion by the end of 2025 [3]. However, this revolution has made the world more vulnerable to cyberattacks and cybercrimes. Many of the cyberattacks are motivated by financial gain, with hackers not only attacking public/private organizations and corporations but also individuals. Therefore, it is critical to have security tools for protection against cyberattacks.

Cybersecurity is the application of technologies, processes, and controls to protect devices, networks, and data from cyberattacks like malware, ransomware, identity theft, etc. The most important aspect of cybersecurity is authenticating legitimate users. Most recently, a widespread report of data leaks showed how close to 8.5 billion password entries were leaked on an underground hacker forum [4] .Therefore, it is crucial to restrict access to authorized users only.

Authentication is a method for verifying the identity of the user. Different methods like passwords, fingerprints, facial recognition, authorization cards, passcodes, etc. are used for user verification. All these methods are used to authenticate the user only once to unlock the session. After the user is successfully authenticated, there is no further validation to check the user's identity. This creates ample opportunity for hackers to hijack the session and steal the data. Additionally, the pandemic has forced employees to work from home, and therefore, remote workforces need additional security as they are not using their companies' secured networks. Since the beginning of the pandemic, cyberattacks have increased, especially in the banking sector [5] .Therefore, it is vital to implement robust security that provides more than static/one-time user authentication.

We believe that continuous user authentication can solve this problem. Continuous authentication is a method for verifying the identity of the user on an ongoing basis until the session expires. This method passively authenticates users without interrupting their workflow. Continuous authentication operates by analyzing multiple unique user behaviors. For instance, the way users handle the keyboard and mouse when typing can be examined to determine unique behavioral patterns. Mobile sensor data can be similarly analyzed for the same purpose. These different behavioral patterns are continuously monitored to verify specific users and to confirm or block their access to an ongoing session.

Continuous authentication using keystroke (typing) data continuously matches the user's typing pattern with patterns previously recorded using artificial intelligence (AI) to identify the user. Continuous authentication verify the user for every keystroke, if any unusual behavior is observed, the user can be locked out and must reconfirm their identity. Different methods like one-time passwords, passcodes, etc. can be used to reconfirm identities.

Government and defense institutions and banking and finance industries with high regulations need additional layers of security to ensure only authorized persons can access their information. These institutions can establish high-security standards using a continuous authentication system.

However, continuous authentication technology is in the development phase and has many research gaps. One of the drawbacks of continues authentication with keystrokes has been the enrollment process. AI algorithms require a high amount of data to identify unique user behavior patterns. This causes a delay in the registration process of the user and hence the start of continues authentication. Another issue is most of the previous studies used synthetic data, hence ML algorithms fail or do not perform as expected in the production environment. Moreover, it is critical to reducing latency in the continuous authentication architecture/system to improve overall performance. The major bottleneck could be the database that is used to insert the user data and retrieve it for making predictions. The continuous authentication system generates around 50 datapoints per second for every user. Hence the database should be able to operate efficiently on millions of records with the lowest latency. This research work focuses on solving the above-mentioned problem using different machine learning techniques and benchmarking frameworks.

1.2 Goal

The goal of the research is to reduce the enrollment time for continuous user authentication, improve the overall accuracy of the authentication process and identify the database with the lowest data latency.

1.3 Research objectives

To achieve the above goal, we have the following research objectives.

A. Transfer learning to accelerate the enrollment process for new users.

B. Ensemble learning to enhance continuous user authentication for real-world environments.

C. Database benchmarking to identify the data latency for different databases.

1.4 Methodologies

In statistics, classification is the problem of identifying what set of categories an observation belongs to. Examples of this, are assigning a given email to the spam or not spam class and assigning a diagnosis to a given patient based on observed characteristics (Sex, blood pressure, presence, or absence of certain symptoms, etc.).

In continuous authentication, the classification algorithms can be used to classify the users based on their keystroke data. Figure 1.1 shows the continuous authentication flow. The classification algorithm is trained using keystroke data, where the user's new keystroke data is given as input to the trained algorithm to classify it. If the classified user matches the input user, then authentication is successful. If the user classified by the algorithm does not match the input user, then that user is blocked.

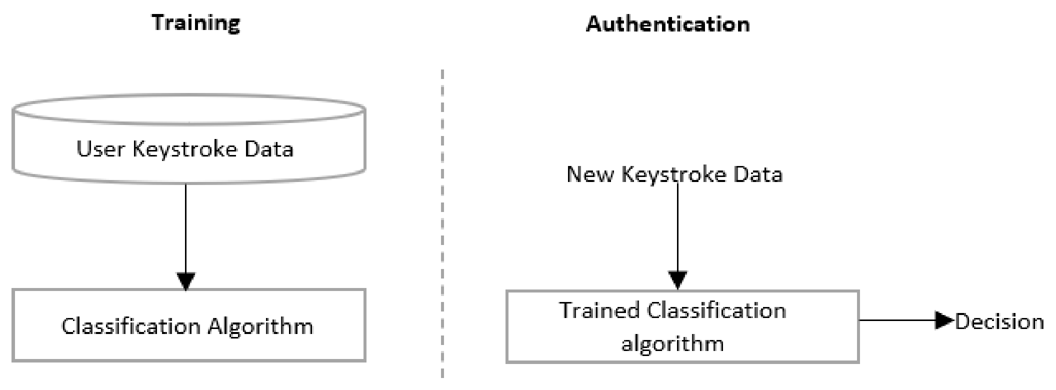


Figure 1.1 Continuous Authentication Process Flow

For example, if the classification algorithm is trained using the keystroke data of the number of users which is collected while they work on the computer, the classification algorithm will treat each user as a class and use keystroke data to identify unique behavioral patterns. When new data comes into the system, it will be classified using the algorithm. If the user is categorized for the right class, then that user would authenticate successfully but if the user data is categorized as a different class, then the user should be blocked. However, the issue is classification algorithm requires a lot of data to find users' behavioral patterns, and plenty of time is required to gather the data which extends the start of continuously authenticating the new user. To overcome this issue for new users, a transfer learning technique (technique that involves using the knowledge and experience gained from solving one problem to help solve a related problem) was used for the feed-forward neural network model. Experiments were done using only one behavioral pattern with a set of 5 users to find the difference in accuracy for the model trained with transfer learning and model trained without any previous learning. The results showed that the model using transfer learning had more accuracy than the model trained from scratch. This implies that using transfer learning improves the accuracy with a small amount of data which will help to speed up the onboarding process for new users. This work generates new knowledge which will allow the researchers to implement various machine-learning techniques with multiple behavioral patterns, thereby providing the best model performance for transfer learning.

Moreover, data quality is important to solving continuous authentication problems. Therefore, this study uses the real-world data of 48 financial organization employees to compare the performance of advanced ML algorithms, including Light GBM, XGboost, TabNet, Neural Network, and 1D Convolutional Neural Network (CNN).Moreover . ensemble learning was used to combine the prediction ability of all the models, which increased cumulative accuracy.

Lastly, identifying data latency is critical to reducing the time required for continuous user authentication and improving overall performance. Hence, a universal database benchmarking framework is developed to determine, evaluate, and report the data latency of the databases for different operations in diverse operating conditions.

1.5 Thesis Outline

The thesis structure is ordered as follows. Chapter 2 shows a literature review of relevant techniques in continuous user authentication. Chapter 3 describes the feature extraction and data-cleaning process. Chapter 4 details the use of transfer learning to reduce registration time. Chapter 5 delineates the evaluation of different ML algorithms and the results of ensemble learning. Chapter 6 includes the database benchmarking framework developed to evaluate databases in production-like scenarios of continuous user authentication.

Chapter 2

2 Literature Review

This chapter reviews current works related to our relevant subjects. Many research contributions have been made around behavior biometrics for user authentication. A few are briefly discussed below.

2.1 Academic Research Review

Identification of User Behavioural Biometrics for Authentication using Keystroke Dynamics and Machine Learning

Krishnamoorthy, Sowndarya proposed work on the use of user keystrokes dynamics for static user authentication [6] .She gathered the data of 94 users over five days. In the experiment, users had to type a short passcode (the passcode was '.tie5Roanl') every day on the iProfile android app. In total, 155 features were obtained and mRMR(Minimum Redundancy Maximum relevance) was used for feature selection. Users were then classified using different classification algorithms and the SVM linear model had the highest accuracy rate of 0.9727. Accuracy is defined as the number of correct predictions divided by the total number of predictions.

Notably, the same virtual keyboard was used to gather all the data, therefore, using different devices/keyboards will affect the accuracy of the model. Furthermore, the fixed short text was used to build the dataset, thus using free text will have an impact on the performance of the classifier. Evaluations, datasets size

A Comparison of Machine Learning Algorithms in Keystroke Dynamics

There has also been some work done by a researcher from North Carolina University on the comparison of machine learning algorithms in keystroke dynamics, they collected data from 23 volunteers in the same setting across two days. They used three different predefined texts, a strong short password, one sentence, and two sentences to produce three different datasets (the three important features of the dataset are Hold time, Downtime, and Uptime). 80% of each dataset was used to train random forest, neural network, decision tree, and SVM. The remaining 20% was used to test the models. The accuracy of the models was low (between 35% to 68%) for the strong short passwords dataset and it was high in the case of two sentence dataset for the random forest model (100%), where 20 samples of each user were used [7].

The same devices were used to enter the fixed text in this study as well. There are very few numbers of features in each dataset, consequently, the increased number of users might reduce the accuracy of the model, as the values in each dataset will repeat and cause difficulty to distinguish between users.

Machine Learning Algorithm on Keystroke Dynamics Pattern

In this research, the researcher used a controlled environment to create their dataset using 1000 volunteers at the University of Mauritius. A total of 30000 samples were generated during this experiment. Different types of passwords like, .tie5Roalnb, .aeihoz246@, .nzkla29zah.#, and aeR5t.ilnb were used to measure the variation between the distances of keys on a keyboard. The datasets were categorized into three types, the first user used both hands, the second only used their dominant hand and the third only used their weak hand. Then the datasets were normalized using the Z-score normalization technique. These datasets were used to train and evaluate different classifiers like Chaotic neural network, SVM, and Neuro Evolution of the Augmenting Topology. The Neuro Evolution of the Augmenting Topology gave the highest recognition rate of 99.1, lowest FRR of 0.25, and lowest FAR of 0.15 for database calculating distance between the key [8].

Though this study has shown a higher recognition rate, the length of the text used is short and it can only be used for static authentication and not continuous authentication. Since the datasets were generated in a controlled environment so it is more consistent but for real-world data, the algorithm might not perform as well as it did.

Multi-Modal Biometric-Based Implicit Authentication of Wearable Device Users

In 2019, Sudip Vhaduri and Christian Poellabauer implemented biometric and behavioral authentication for wearable IOT devices (Fit Bit) using Net Health data. Behavioral data of 400 students were gathered over 17 months with 20 hours of valid data per person per day. The Fitbit device collected heart rate, calorie burn, metabolic equivalent of a task (MET), physical activity level/intensity, step count, sleep status, and self-recorded activity labels. This data was divided into three biometric groups: behavioral (e.g., step counts, activity level/intensity), physiological (e.g., heart rate), and hybrid (e.g., calorie burn, MET) biometrics, where hybrid biometrics are derived from both behavioral and physiological biometrics. As real word data was used, it was cleaned to remove invalid periods of activity and segmented into a five-minute non-overlapping window. Multiple feature selection techniques like Kolmogorov-Smirnov test-based approach, Pearson Correlation Coefficient (PC)-based approach, and Standard Deviation (SD)-based feature selection approach were used to obtain different datasets. These datasets are evaluated using Quadratic Support Vector Machine and the unary Gaussian Support Vector Machine classification techniques.

For each feature set with N subjects/users, N separate models were built, one for each subject. A binary q-SVM classifier was trained with a positive class consisting of one subject's data and a negative class consisting of data from the rest of the N−1 subjects and a g-SVM classifier model was trained using a subject's data with a certain percentage of data being considered as outliers. Then models were tested on the 25% data from a particular subject as positive class and 25% data from the rest of N−1 subjects as negative class. Binary classifier trained with Kolmogorov-Smirnov feature set gave the maximum accuracy of 93% [9] .

The important thing to address is that the behavior data (step count) did not play a significant role in authenticating the user. Further, the user is authenticated every 5 minutes, so the intruder gets an opportunity to hack the device. Lastly, the physical level attack can be detected using this method however it cannot detect the communication protocol attacks.

PersonaIA: A Lightweight Implicit Authentication System Based on Customized User Behavior Selection

This research is focused on the development of a behavior-based user authentication system using partially labeled Dirichlet allocation and minimizing battery usage required for behavioral user authentication. To achieve it, the researcher-developed a new layer called the W layer which helps continuous user authentication even when a mobile device is not connected to a network, also this layer helps reduce battery consumption [10].

PLDA algorithm was tested for a dataset containing features like GPS, accelerometer, SMS messages, call logs, Bluetooth device logs, app installation data, app running data, and battery usage information. It gave an accuracy of 93.3% with a battery consumption of 14.5% of the device's total battery.

Performance Analysis of Multi-Motion Sensor Behavior for Active Smartphone Authentication

In this paper, the experiment was carried out to analyze the reliability and applicability of multi-motion sensor behavior for continuous smartphone authentication in various surroundings. Researchers used an accelerometer, gyroscope, magnetometer, and orientation sensors to collect data of 102 subjects. A total of 192 features were extracted from the data to characterize the input action. These features were characterized into two categories, first is descriptive features defining the motion patterns of touch action and intensive features indicate the complexity and intensity of the touch action. Then these features were filtered based on how well the feature can discriminate between the users. Next, the dataset was used to train the one-class classifier, Hidden Markov model, SVM, and neural network. The hidden Markov model gives the lowest FRR(False Rejection Rate), FAR(False Acceptance Rate), and EER among all the classifiers. FAR is computed as the ratio of the number of false acceptances and the number of test samples from impostors, FRR is computed as the ratio of the number of false rejections and the number of test samples from legitimate users and equal-error rate (EER) at the sensitivity of the classifier where FAR = FRR [11].

During the authentication process, legitimate users' behavioral profile is built using sensor data, and is compared with the current users' behavior after every N touch actions window. Depending upon the value of N intruders might get time to hack the mobile, to steal the data whereas, if the value of N is smaller to avoid intrusion, it will lower the accuracy as a very small amount of data will be available for authentication.

On Continuous User Authentication via Typing Behavior

The researchers proposed a novel biometric authentication method using a computer vision algorithm to identify users, based on typing behavior. To implement this system, they conducted 2 phase data collection including 63 computer subjects who type static text or free text in multiple sessions. In phase 1, under the monitoring of the researcher, the subject performed typing with the same chair, fixed keyboard position, lighting, and a similar computing environment. And in phase 2, the authors used a shared lab where the participant could come multiple times for five months period to write free and static text. Using this data, they extracted different features like the shape and motion of the hand, color, and texture of hands. Next, these datasets were used to train BoW(Bag of Words), BoP(Bag of Phrases) and BoMP(Bags of Multi-Dimensional phrases). BoMP had the highest accuracy of 0.9996 and True Positive Rate (TPR) of 67.8% and False Positive Rate (FPR) of 0.73, for the phase 2 sample dataset [12].

Though this is a novel authentication method, it requires extra hardware(camera) and higher computing power. Also, it is not a robust way of authentication because features extracted from the video are highly dependent on the camera angle. Therefore, a lot of research is required to be done to implement the proposed system in real-world scenarios.

Pattern-Growth Based Mining Mouse-Interaction Behavior for an Active User Authentication System

In this paper, the experiment was carried out to analyze mouse-interaction behaviors for identifying computer users. This study divided the mouse-interaction behavior patterns into two categories, micro-habitual patterns, and task-intended patterns to extract frequent mouse behavior patterns from holistic behavior. To implement this system, researchers recruited 159 students and faculty, subjects were asked to use their own devices for data collection. And the collected data were periodically sent to the remote server, along with the subject ID. Then the features were extracted from the data, which were organized into a vector to represent behavior patterns and to construct the feature vectors to train ML models. Researchers used KNN, Neural Networks and one class SVM classifier machine learning models to perform user authentication. These models were trained using one of the subjects as the legitimate user, and the rest as impostors. Later the models were tested to identify the legitimate user and the imposters. One class SVM classifier performed the best out of the three trained models and had the lowest FAR of 0.09 percent with an FRR of 1 percent [13].

To test this system in a real-world scenario they used an observation window to make an authentication decision, which contained a sequence of N mouse operations. For five mouse operations, the EER was approximately 15 percent, but the authentication decision was made in 8.77 seconds (on average). As the number of operations increased error rates started to reduce. For the operation length of 20, the best EER dropped to 0.75 percent, but the corresponding time increased to 47.11 seconds.

In this study, researchers train the model with predefined imposter data but in the real world that will not be available, so the model's performance might get hampered.

Continuous authentication on mobile devices using behavioral biometrics

The authors present two different continuous authentication techniques for mobile devices. First is the model-based approach, here the ML models are built, trained, and tested for all the users. Second is a template-based approach where the similarity score is calculated for the individual user.

The data of 5 users were gathered around 200 minutes. For the model-based approach, the Convolution neural network (CNN) and Multilayer Perceptron (MLP) algorithms were trained. CNN outperforms MLP and reaches the Equal Error Rate (ERR) of 8% for a 20sec sample size. Whereas the template-based approach uses a Siamese network and archives ERR of 10% [14].

In this study, federated learning can be used with a template-based approach, it can assist to reduce authentication time and improve overall performance. Additionally, one class classifier can be used on the device for everyone.

Secure System of Continuous User Authentication Using Mouse Dynamics

In this research, authors collected mouse dynamics data and trained a One-class support vector machine (SVM) to identify the user. The data from 23 participants were collected over 4 weeks however data from only eight participants were used for the experiments. Later this data was used to create their profile and train 1-class SVM with five-fold cross-validation. Afterward, all 8 models were tested, and the training accuracy for all the participants is around 90% [15].

This study uses data from only eight users also, the approach to using 1-class SVM is not practical because the number of models increases with the number of users, and it would be difficult to maintain and periodically retrain individual models. Moreover, the test accuracy of the models is not mentioned in the result.

In addition to academic research, some market products were reviewed as well. The details are described below.

2.2 Market Products Review

There are a few companies that are trying or have already developed an authentication service using behavioral biometrics. These organizations are explored below:

TypingDNA : This New York-based company provides static keystroke authentication which works for both desktop and mobile applications. Their service can be used through an API or the TypingDNA application. TypingDNA is not continuously authenticating the users, instead, they are using behavioral biometrics as an additional layer for authenticating the user and providing multi-factor authentication at the time of login [16].

However, they have not mentioned their customer roster on the company website, so it is difficult to determine the efficiency of their product.

PluriLock: It is a North American organization that claims to have developed a continuous and static authentication system using machine learning algorithms. They have listed two products on their website [17].

Plurilock ADAPT: It performs static user authentication, by using information like typing biometrics, geolocation & travel, time of day, network & environmental context, device id, and fingerprint.

Plurilock DEFEND: This is used for continuous authentication, using keystrokes, pointer movements, and machine learning algorithms, it can predict intruders. If the movement seems unusual and of medium risk then it is logged whereas, if the risk threshold is crossed, it notifies the security staff.

Plurilock is currently providing its service to regional American banks.

Lastly, to compare all the research the below parameters are used.

Devices used to collect data: The type of device used to collect the data for the experiment is important because the user's behavior might change with the device. Also, there can be a difference in the data collected using different devices. Devices like keyboards, mouse, IoT devices, and smartphones can be used to collect the data.

Source of Data: The performance of the model depends on the size of the dataset as statistics of the dataset change with the size. For example, Variance decreases as dataset size increases. Therefore, it is important to choose the right dataset size. Behavioral features are extracted from the inputs; therefore, it is very important to analyze input patterns, which can be short text, free text, touch actions, or IoT device sensor data. These inputs are used to build the user's behavioral profile.

Features Extraction: Feature extraction is an important part because the models used for prediction are trained using these features thus, the performance of the model varies with features.

Proposed Model: Multiple machine learning models can be used to solve a single problem. After comparing the performance of multiple models, the best model is selected.

Evaluation Matrix of the Model: The performance of the model can be evaluated using different parameters like accuracy, precision, recall, etc. It is crucial to select the right parameters for evaluation as only using a few of the parameters can be misleading.

Environmental Condition: User’s behavior changes with surrounding environmental conditions. For example, a person talking with someone and typing using a keyboard will have a different behavioral pattern compared to the same person typing without any distractions.

Settings/Utilization: This time is a critical evaluation criterion for continuous authentication because if the system authenticates the user after a window of a certain time, then the intruders get a chance to attack.

The Table 1 shows the comparison of literature review using the above parameters.

Table 1 Literature Comparison

Source Study	Devices used to collect data	Source of data		Features Extraction	Proposed Model	Evaluation Matrix	Environmental Condition	Settings/ Utilizations
		No of User	Type of Data					
Identification of User Behavioural Biometrics for Authentication using Keystroke Dynamics and Machine Learning	Virtual Keyboard	94	Short and fixed text (E.g: '.tie5Roanl')	Minimum Redundancy Maximum relevance technique used to fetch 155 features	SVM Linear Model	Accuracy: 0.9727, F1 score: 0.9699	Variable Env conditions as virtual keyboard was used	Static/one time Authentication
A Comparison of Machine Learning Algorithms in Keystroke Dynamics	Computer keyboard	23	Strong short text, one sentence, two sentence	Only 3 features were used	Random Forest	Accuracy: 100	Controlled Environmental Lab setup used	Static/one time Authentication
Machine Learning Algorithm on Keystroke Dynamics Pattern	Same Device (Keyboard)	1000	Strong short password	Z-score normalization technique used for feature normalization	Neuro Evolution of the augmenting topology	FAR:0.15, FRR:0.25, and Recognition Rate:99.1	Controlled Environmental Lab setup used	Static/one time Authentication
Multi-Modal Biometric-Based Implicit Authentication of Wearable Device Users	Same Device (Smartwatch)	400	Health Data like Calories burnt, heart rate etc.	Kolmogorov-Smirnov, Pearson Correlation Coefficient, Standard Deviation (SD)-based feature selection used	Quadratic Support Vector Machine	Accuracy:93%, FPR:0.10, FNR:0.04	Variable Env as smartwatches were used to collect the of users for 17 months	Continue Authentication with the window of Five Minute
PersonaIA: A Lightweight Implicit Authentication System Based on Customized User Behavior Selection	Mobile device	23	Mobile device information (Battery Usage, GPS etc)	Topic modeling	Partially labeled Dirichlet allocation (PLDA)	Accuracy:98.6, Precision:93.3	Variable Env	Continues Authentication
Performance Analysis of Multi-Motion Sensor Behavior for Active Smartphone Authentication	Android mobiles	102	Device information as well as users' behavioral pattern (Angle at mobile is hold and Touch actions etc.)	Discriminating power	Hidden Markov Model	FRR:5.03%, FAR3.98%, EER:4.71%	Variable Env	Continue Authentication after every N touch windows
On Continuous User Authentication via Typing Behavior	Camera and Keyboard	63	Video captured to analyse Shape, texture of hand	Feature extraction steps: (a) original frames from multiple subjects, (b) foreground segmentation with hand separation, (c) shape context extraction. The top-left image shows four patches used in the linear regression	BoMP(Bags of Multi-Dimensional phrases)	Accuracy: 0.9996 . True Positive Rate (TPR) : 67.8% and False Positive Rate (FPR): 0.73	Step 1: Controlled Environmental Lab setup used Step 2: Same devices but surrounding conditions were not controlled	Continues Authentication
Pattern-Growth Based Mining Mouse-Interaction Behavior for an Active User Authentication System	Mouse	159	Mouse Clicks and Movements	Features selected based on Feature Stability in Behavior Pattern, Feature Discriminability in Behavior Pattern, Statistical Dispersion of Features	One class SVM classifier	FAR of 0.09 percent and FRR of 1 percent	Variable Env	Continues Authentication after N mouse actions

Chapter 3

3 Feature Extraction and Data Preprocessing

Continuous authentication uses behavioral information to authenticate the users. However, the raw data collected from users cannot be directly used for the authentication process as it doesn't provide any details of user behaviors. Therefore, it is needed to extract user behavioral features/data from the raw data. This chapter explains the behavioral feature extraction and data cleaning process of the raw user data. Later the cleaned data is used for experiments described in Chapters 4 and 5. The details of data processing are explained below.

The raw data of the employees from the financial institution was collected during a period while they were working. It is crucial to maintain data privacy while collecting data from employees. Currently, there is no ISO data privacy standard for continuous authentication. However, the sensitive data was identified for the organization and was encrypted for data privacy.

The raw data included features such as the event that occurred, for example, a key press, mouse click, mouse scroll, etc. It also included the x and y coordinates of the mouse pointer, the time stamp of the event, and lastly the user Id. Table 2 provides columns for all the raw data.

Table 2 Raw Dataset Description

Column Name	Details
Timestamp	Timestamp when any event happened
Event	Captures the event like mouse up, mouse down, mouse move, key up, and key down
X	X coordinates of the mouse pointer
Y	Y coordinates of the mouse pointer
Type	Describes the type of event that occurred such as mouse scroll, left click, right-click, key press information
User Id	Unique Id for each user

Using this raw data, two behavioral patterns were extracted for the purposes of the experiment: First, the mouse click length, the time difference between mouse key press and release, and second, the screen location, the area on the screen where the mouse click occurred. To extract the screen location, the screen was divided into a grid of 16 using the x and y coordinates of the mouse pointer. Figure 3.1 shows the method for dividing the screen to determine the screen location based on the x and y coordinates of a mouse click. Screen location feature was extracted specifically for the application used by the financial organization.

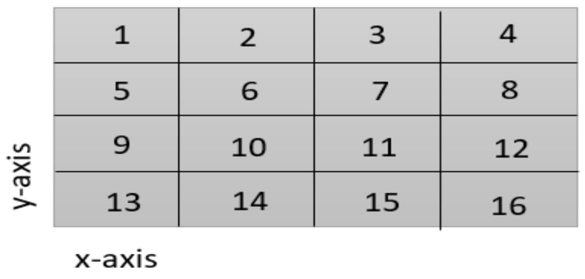


Figure 3.1 Screen Partitioning

Table 3 illustrate the details of the behavioral information dataset extracted from the raw data.

Table 3 Behavioral Dataset Description

Column Name	Details
Click Button	What key has been pressed, for example, mouse right or left click
Click Length	Time difference between key press and release
Screen location	Part of the screen where the mouse click happened
User Id	Unique Id for each user

3.1 Data cleaning for outlier removal

The behavioral dataset was then cleaned to identify and correct the errors in the dataset caused because of different factors like physical or other disturbances to the user. Such disturbances can create outliers that are different than the actual behavior of the user.

These outliers can affect the way machine learning models identify the unique behaviors of the user. To prevent this, the Interquartile range (IQR) technique was used to clean the data and remove the outliers.

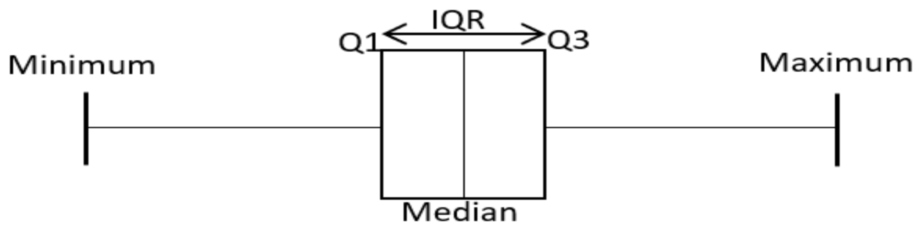


Figure 3.2 IQR Diagram

IQR is a statistical method to calculate the upper bound and lower bound of the dataset to set the decision range. Any data point outside of the decision range is considered an outlier. Figure 3.2 shows the distribution of the data. To set the decision range, it is necessary to find the minimum and the maximum values of the dataset and calculate Q1, i.e., the first quartile of the data (25% of the data between minimum and Q1), Q3, i.e., the third quartile of the data and lastly the median value (second quartile). These values are used to calculate IQR using the below formula.

$$\boldsymbol{IQR = Q3 - Q1} \qquad \textbf{(1)}$$

The value of IQR is used to calculate the lower and upper bounds of the decision range using the below formula:

$$\textbf{\textit{Lower bound: Q1 - 1.5 * IQR}} \qquad \textbf{(2)}$$

$$\textbf{\textit{Upper bound: Q3 + 1.5 * IQR}} \qquad \textbf{(3)}$$

Using this technique, mouse click length data were cleaned to remove outliers. Figure 3.3 shows the box plot for click length data before applying IQR, with a few outliers/ data points not indicative of the normal behavior of the user. By comparison, Figure 3.4 shows the box plot of the dataset after removing outliers using the IQR technique. Removing the outliers will help in determining the real behavioral patterns for all the users.

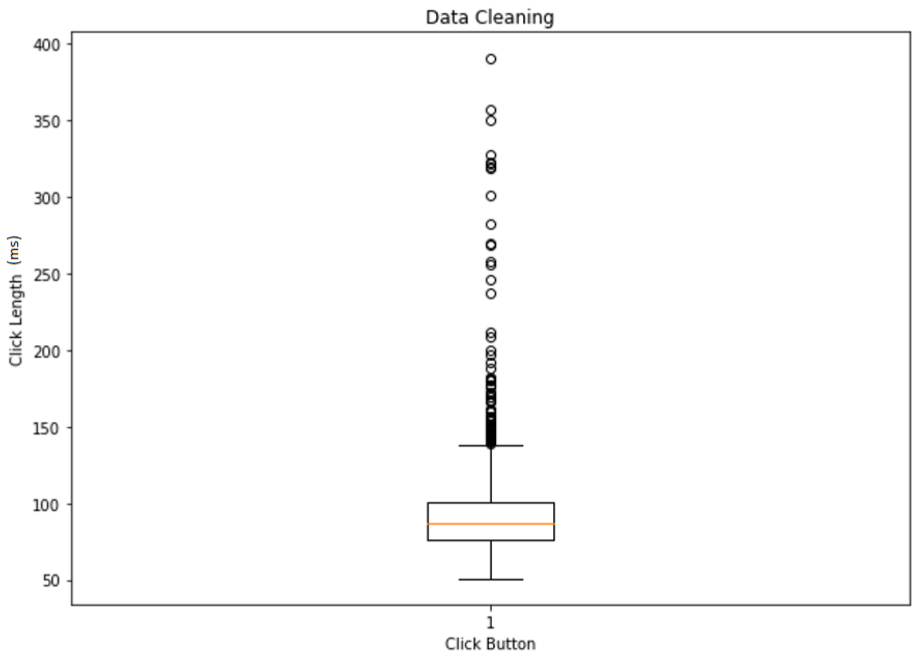


Figure 3.3 Data Distribution before IQR

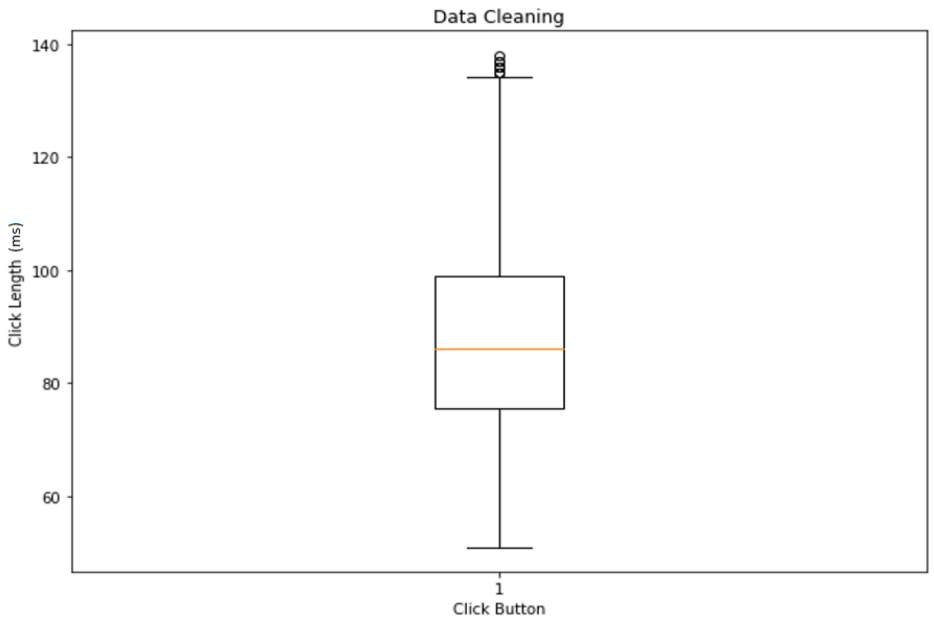


Figure 3.4 Data Distribution after IQR

The Figure 3.5 displays the data distribution of the click length for all the 48 users. Almost 80% of the click length values are between 50ms to 180ms for all the users which shows that it is overlapping data and makes it hard to find unique behavioral patterns for the users

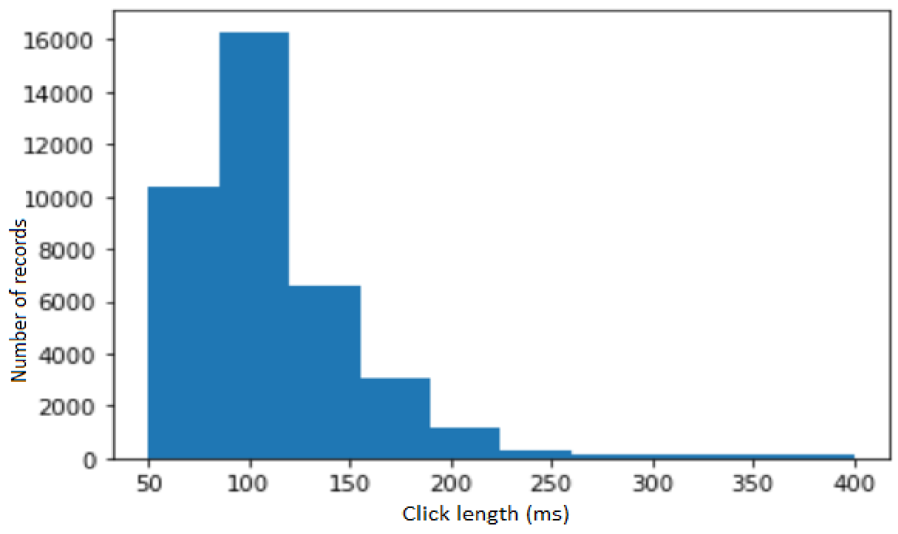


Figure 3.5 Click length data distribution for all the users

Chapter 4

4 Transfer Learning to Reduce Enrollment Time

Continuous authentication requires a lot of data to find the user's behavioral pattern and plenty of time is required to gather the data which extends the start of continuously authenticating the new user. In this study, the transfer learning technique was used for a feed-forward neural network model to overcome this issue for new users. Experiments were done using only one behavioral pattern with a set of 5 users to find the difference in accuracy between the model trained with transfer learning and the model trained without any previous learning. The results showed that the model using transfer learning had 9.76% more accuracy than the model trained from scratch. This implies that using transfer learning improves the accuracy with a small amount of data which will help to speed up the onboarding process for new users. This work generates new knowledge which will allow the researchers to implement various machine learning techniques with multiple behavioral patterns, thereby providing the best model performance for transfer learning. The details of the experiments are explained below.

4.1 Transfer Learning

The human brain can transfer the knowledge of one task to solve another similar task. The more related tasks, the easier it is to transfer knowledge. For the related tasks, humans don’t learn everything from scratch, they transfer past knowledge to learn new tasks.

In deep learning, models need a lot of labeled data to solve complex problems. It is hard to give a large amount of data to the model for training because the data-gathering process is costly, also, the model requires high-end computational resources to work on a large amount of data. Data gathering and model training are time-consuming processes. For example, continuous authentication requires a lot of data to recognize the unique behavioral pattern and collecting that much data is time-consuming and costly. To overcome this concept of transfer learning can be used in deep learning.

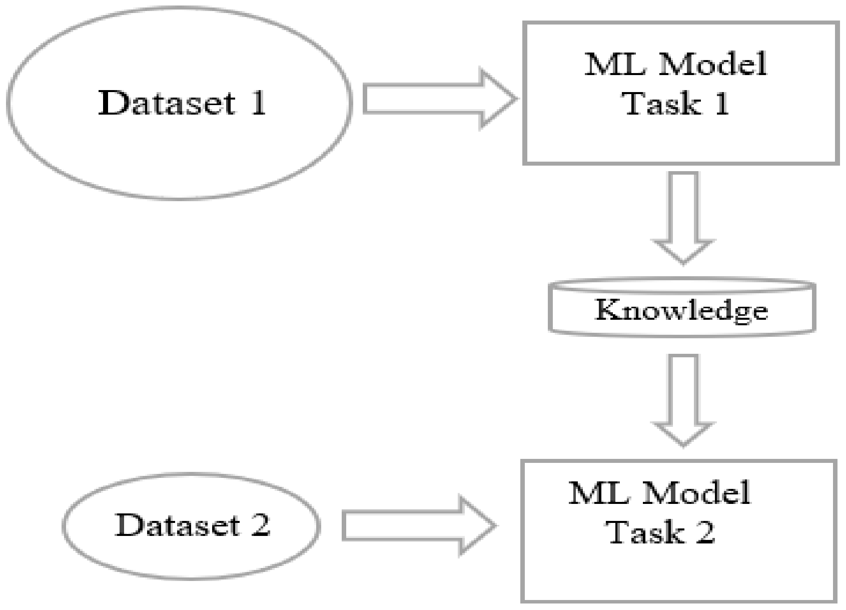


Figure 4.1 Transfer Learning

Figure 4.1 shows the workflow of transfer learning. The deep learning model learns one task and adjusts the network parameters to get the optimal result. These network parameters can be transferred for the model to learn the task from the same domain. As the model optimize parameter for other tasks, it doesn’t start learning from scratch, but it uses previous knowledge and use that to learn a new task [18] .This helps the network learn faster with a small amount of data.

The same technique was used, where the network was trained using 48 users with 48000 data points (1000 for each user) to have a base model. The dataset was divided into 80% for training and 20% for testing. And network parameters were optimized to find the best model. This was the base model for the experiments.

The base model was then used for a new task where it classifies 5 new users from the same dataset with 245 records each. The output layer of the base model was changed because different users were classified, and all the hidden layers were frozen to use the same parameters. Parameters were not fine-tunned because the number of users is scarce and to avoid overfitting same network parameters were used by freezing the hidden layers.

4.2 Neural Networks

A neural network consists of an artificial network of functions, called parameters, which allows the computer to learn, and fine-tune itself, by analyzing new data. Each parameter, sometimes also referred to as neurons, is a function that produces an output, after receiving one or multiple inputs. Those outputs are then passed to the next layer of neurons, which use them as inputs for their function and produce further outputs. Those outputs are then passed on to the next layer of neurons, and so it continues until every layer of neurons has been considered, and the terminal neurons have received their input. Those terminal neurons then output the result for the model.

Neural networks can detect the complex nonlinear relationship between the variables which can help to solve our classification problem. Also, great computational power is available to us which is required for neural networks [19].

Neural networks may have three layers input layer, an output layer, hidden layer.

Input layer: It depends on the shape of the data, and the number of neurons in the input layer is equal to the number of features in the data.

Output layer: The number of neurons in this layer depends on the activation function used. If using the SoftMax function, then one neuron per class label is required and for other activation functions like Sigmoid, we can use one output node.

Hidden layer: Hidden layer is a layer between the input layer and the output layer in which the function applies weight to inputs and directs them through an activation function as the output. It does the nonlinear transform of the inputs entered into the network.

Figure 4.2 shows the neural network architecture used for the experiments. It has an input layer, three hidden layers, and an output layer. The input layer has 3 neurons, one for each attribute. The output layer has a neuron for each user (class) and several neurons in the hidden layer are tuned to get optimized results.

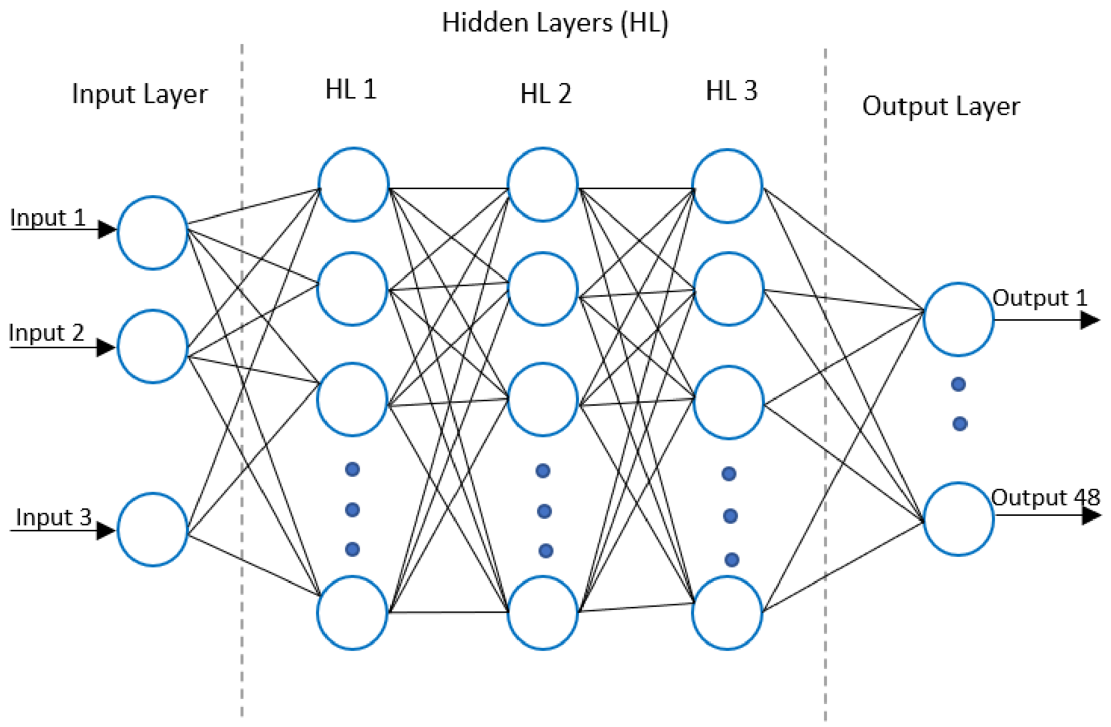


Figure 4.2 Neural Network Architecture

4.3 Hyperparameter tunning

In machine learning, a hyperparameter is a parameter whose value is used to control the learning process and find out the value of model parameters. Hyperparameter tuning is the process of choosing a set of optimal hyperparameters for a learning algorithm [20]. All the machine learning models require tuning the hyperparameters to get optimal results for different types of problems.

There are three commonly used hyperparameter tunning methods:

Grid Search: This is the basic method to tune the hyperparameter. In this method, the grid of hyperparameters is provided and the algorithm is trained using all the possible combinations of it to find the best values combination among the given hyperparameter. This method is very inefficient as the number of models to train increases exponentially with an increase in the number of hyperparameters.

Random Search: In this method, it is not required to provide a set of values instead it takes the statistical distribution/ range of each hyperparameter. Then the values are sampled from the given range of parameters. Unlike grid search, this method doesn’t try all possible combinations, but it can be specified how many combinations to try.

Bayesian Search: In both above methods all the experiments are independent of each other. However, the Bayesian optimizer is a sequential model-based optimization technique that uses previous results of the model training and decides the next hyperparameters candidates. This method chooses the hyperparameters in an informed manner and can find the best parameters in less time.

After comparing all the methods Bayesian search was used for tuning the neural network. A total of four parameters were tuned including the number of neurons, learning rate, activation function, and loss function.

The below set of parameters was used for the tuning process.

1. Number of neurons: 256, 1500
2. Learning Rate:0.001, 0.01, 0.1
3. Activation function: SoftMax, Sigmoid, Relu
4. Loss Function: sparse categorical cross entropy, hinge

4.4 Experiments And Results

This section describes the results of hyperparameters optimization, classification accuracies, and a comparison of the transfer learning technique to the regular method.

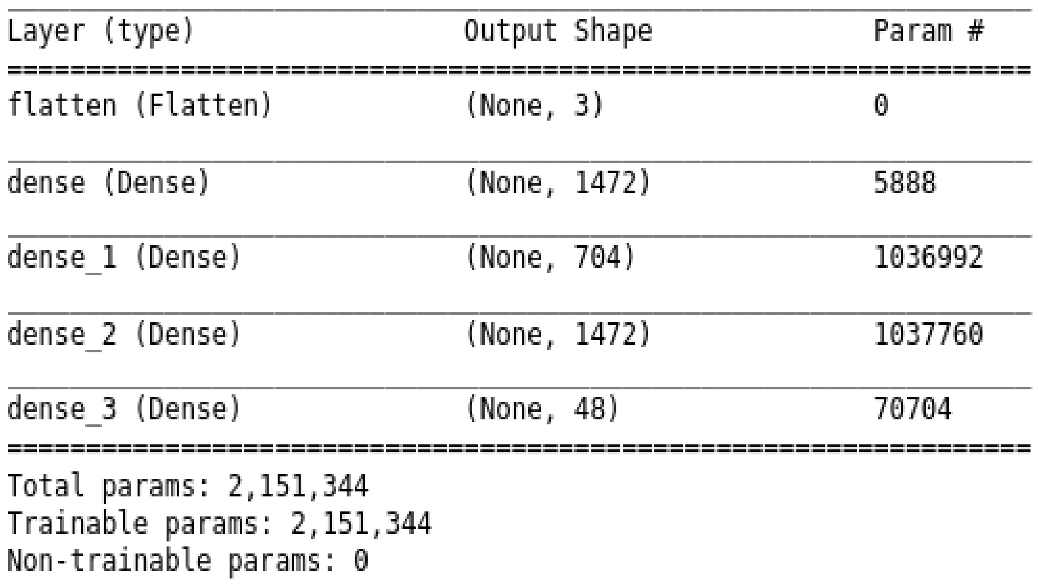
4.4.1 Hyperparameter optimization

The feed-forward neural network was tuned using a set of hyperparameters mentioned in chapter 4.2. Table 4 shows the best values of the parameters.

Table 4 Hyperparameter Details

Hyperparameter	Value
Number of neurons	Hidden layer 1: 1472 Hidden layer 2: 704 Hidden layer 3: 1472
Activation Function	Hidden layers: Relu Output Layer: Sigmoid
Learning rate	0.001
Loss Function	sparse categorical cross-entropy

Using these hyperparameters neural network was trained on the data of 48 users. This helped to create the base model which is trained on a large amount of data and a greater number of users. Figure 4.3 shows the model summary, all the model parameters are transferable



```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
flatten (Flatten)            (None, 3)                 0
_________________________________________________________________
dense (Dense)                (None, 1472)              5888
_________________________________________________________________
dense_1 (Dense)              (None, 704)               1036992
_________________________________________________________________
dense_2 (Dense)              (None, 1472)              1037760
_________________________________________________________________
dense_3 (Dense)              (None, 48)                70704
=================================================================
Total params: 2,151,344
Trainable params: 2,151,344
Non-trainable params: 0
```

Figure 4.3 Optimized Neural Network Model Summery

4.4.2 Model Training

The learning from the base model was then transferred to train the model to perform the new task to classify 5 different users. All the layers shown in Figure 4.2 were frozen except the output layer. In addition, the model was trained using the click lengths of 5 new users. A total of 1225 records were used, 245 for each user, 80% was used to train, and the remaining 20% for testing the model. Figure 4.4 shows the accuracy of the model during the training process. The training accuracy of the model was 54.55%.

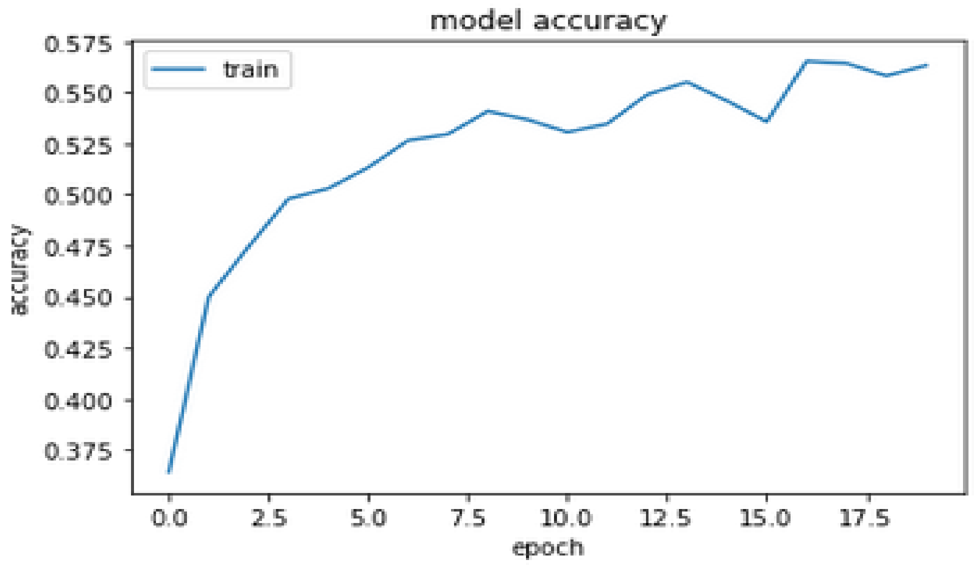


Figure 4.4 Model Training with Transfer Learning

To compare the performance of transfer learning, the neural network model was trained from scratch using the same 5 user data and hyperparameters represented in Table 5. Figure 4.5 shows the model training process. The training accuracy of the model was 42.61%.

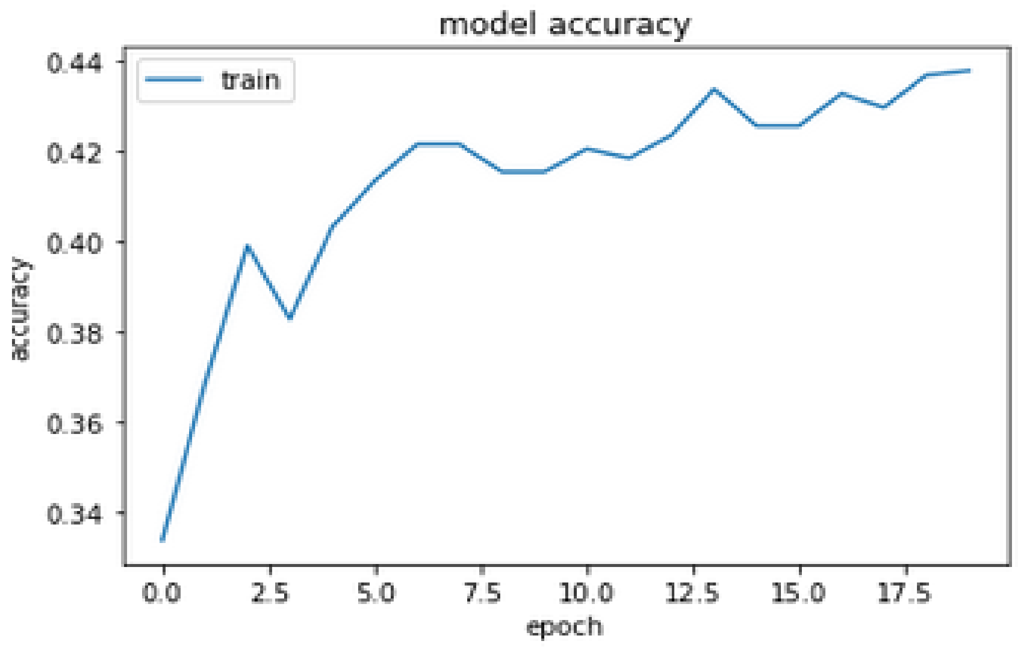


Figure 4.5 Model Training from Scratch

For the fair evaluation, the neural network hyperparameters were optimized for the new task as well, where 5 users were classified. The same range of hyperparameters was used from Section 4.3 with a Bayesian search to tune the hyperparameters. Table 5 shows the hyperparameters after the tunning process and Figure 4.6 shows the model training history. The training accuracy of the model was 40.56%.

Table 5 Optimized Model Hyperparameters for a new task

Hyperparameter	Value
Number of neurons	Hidden layer 1: 640 Hidden layer 2: 1120 Hidden layer 3: 480
Activation Function	Hidden layer1: Relu Hidden layer 2&3: Sigmoid Output Layer: SoftMax
Learning rate	0.001
Loss Function	sparse categorical cross-entropy

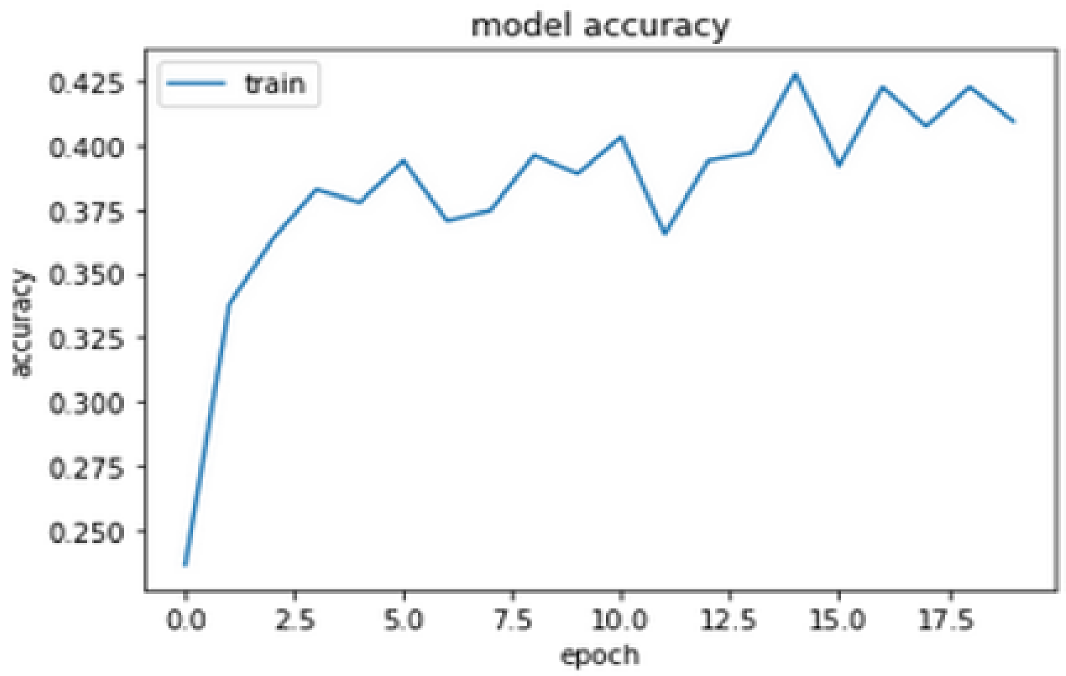


Figure 4.6 Model Training with optimized hyperparameter

4.4.3 Model Testing

The 20% data which was held out to test the models gave the below results. The model with transferred learning had 9.76% more accuracy than the model trained from the scratch.

Table 6 Results Comparison

Model	Test accuracy	F1 Score
Transfer Learning model	50.61%	0.5031
Model with hyperparameter optimization	40.82%	0.3869
Model trained with same hyperparameters	39.59%	0.3693

4.5 Conclusion

Similar studies (Chapter 2) conducted were not focused on reducing the time required for data collection as it can be a very long and time-consuming process. Whereas this research presents a novel approach to reducing the data collection time and hence the registration time. This will make the system ready to be used early when compared to others. It can help in commercialization of the continuous authentication. Once the base model is ready then it will be easy to onboard new clients/organizations in less amount of time. Transferring the learning from the base model will help to get better accuracy with a small amount of data gathered in less time duration.

For the experiment purpose, only one behavioral pattern was used to train the models. The results with it show that using transfer learning for a new set of users gives better accuracy than the model trained from scratch. Also, when the same technique is used with multiple patterns combined, it is expected that accuracy would increase exponentially. This is because the model will have more features to learn under laying behavioral patterns. Also, this would help to get better classification accuracy for a small amount of data. Hence, the time required for data gathering would reduce and a continuous authentication system could be implemented in less amount of time for new users.

Chapter 5

5 Ensemble Learning to Enhance Continuous User Authentication for Real World Environments

The performance of the ML classification algorithm is key in continuous user authentication, and it requires diverse and comprehensive data to be effective in the production environment. In many cases, the ML algorithm is trained on the datasets collected in a controlled lab environment and the model fails or does not perform as expected in the production environment. For example, China's facial recognition system recognized the face on a bus ad as a jaywalker because the model was not trained on real-world data. To overcome this problem, the real-world data of 48 of a financial organization's employees was used to compare the performance of advanced ML algorithms, including Light GBM, XGboost, TabNet, Neural Network, and 1D CNN. Among all the individual models, LightGBM performed best with an accuracy of 23.58%. However, some ML models were better at predicting particular sets of users than others, hence ensemble learning was used to combine the prediction ability of all the models, which increased cumulative accuracy to 24.03%. These results suggest that the boosting algorithm is effective at classifying users. Additionally, the prediction performance can be improved using ensemble learning techniques.

5.1 Background

Data quality is vital to solving continuous authentication problems. Data is deemed of high quality if it correctly represents the real-world construct to which it refers. To properly train a predictive model, data must meet exceptionally broad and high-quality standards. The previous research in continuous authentication has used the keystroke data collected in a controlled environment like computer labs, to train and test the Machine Learning (ML) algorithms [21] [22] [23] [24] [25]. However, because these models function differently on real-world datasets where the user has no restrictions, it is crucial to assess the performance of ML algorithms on real-world data. In this study, the ML models are evaluated using the core features explained below.

- The performance of ML models was assessed using the data collected from a financial organization’s employees (real-world data). (Chapter 3).

- Each model was good at predicting a certain set of users, so ensemble learning was used to improve accuracy by merging the prediction potential of the models. (Chapter 5.3).
- Only two behavioral patterns out of hundreds were used for the experiments, and they showed good accuracy. Using more patterns would exponentially raise the authentication accuracy (Chapter 3).

The clean data was used to train different classification models and an ensemble of them. The details of the algorithm are described below.

5.1.1 LightGBM

Light GBM is a high-performance gradient boosting framework based on a decision tree algorithm. Boosting models are built sequentially by minimizing errors from previous models while increasing the influence of high-performing models. It uses boosting to convert weak learners to strong learners by growing vertically i.e., it grows leaf-wise (as shown in Figure 5.1). It chooses the leaf with a large loss to grow and therefore reduces loss more than a level-wise algorithm when growing the same leaf. The figure shows the leaf-wise growth in LightGBM [26].

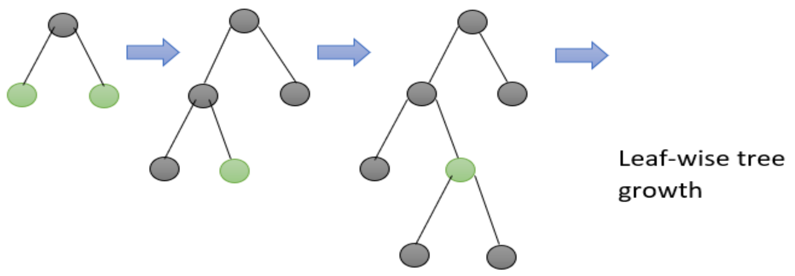


Figure 5.1 LighGBM Tree Growth

LightGBM is a fast, distributed machine learning algorithm used for classification, ranking, and other tasks. It requires less computational power to deal with a large amount of data and gives faster results. LightGBM has more than 100 parameters, but it is not required to tune all of them. The parameters that more profoundly impact algorithms' performance are the max depth of the tree, minimum data in the leaf, feature fraction, bagging fraction, early stopping round, lambda (regularization parameter), and minimum gain to split.

5.1.2 XGBoost

XGBoost is a decision-tree-based ensemble machine learning algorithm that uses a gradient-boosting framework. Gradient boosting is a supervised learning algorithm, which attempts to accurately predict a target variable by combining the estimates of a set of simpler, weaker models. It grows horizontally (level-wise) to reduce loss as shown in Figure 5.2.

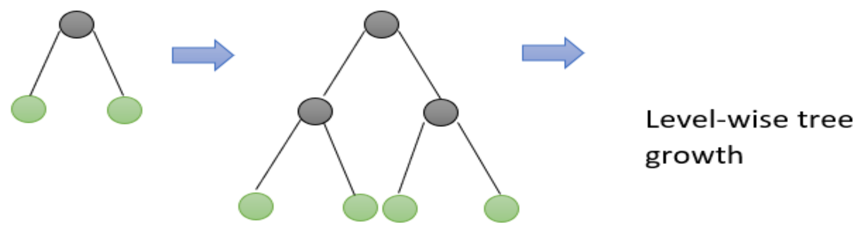


Figure 5.2 XGBoost Tree Growth

XGBoost performs well for the well-structured dataset. Also, XGBoost uses parallel processing, which makes it faster. It can handle missing values and uses regularization to avoid overfitting [27].

5.1.3 Neural Networks

Neural networks are the representation of the human brain, i.e., neurons interconnected to other neurons to form a network. They have three layers: the first layer is the input layer in which inputs are entered, the next layer is a hidden layer with multiple internal layers that perform mathematical operations, and lastly, the output layer gives output. If a hidden layer has multiple layers, then it is called a deep learning/ deep neural network [28].

Each neuron connects to another and has an associated weight and bias. If the output of any individual neuron is above the specified threshold value, that node is activated, sending data to the next layer of the network. Otherwise, no data is passed along to the next layer of the network. The threshold for each neuron is obtained using the activation function.

Neural networks are powerful tools in machine learning, once they are fine-tuned for accuracy because they can be used for classifying and clustering the data at a high velocity.

5.1.4 TabNet

TabNet was developed at Google to be used specifically for tabular/ structured data. It uses a machine learning technique called sequential attention to select which model features to reason from at each step in the model. Each step has a block of components like an attention transformer, mask, feature transformer, activation function, etc. Each step has its vote in the final classification and these votes are equally weighted. The number of steps is a hyperparameter option and increasing it will increase learning capacity but will increase training time, the chance of overfitting, and memory usage as well. TabNet was developed in 2019 and has shown good performance for structured data. Therefore, it is used in this research [29].

5.1.5 1D CNN

CNN (Convolution Neural Network) is a type of neural network mainly used for image classification. It typically has three layers: a convolution layer, a pulling layer, and a fully connected layer. Recently, a 1D-convolution neural network (CNN) achieved the best single model performance in a Kaggle competition with tabular data. In this model, a fully connected layer is used to create a larger set of features with locality characteristics, and it is followed by several 1D-Conv layers with shortcut-like connections [30].

5.1.6 Ensemble learning

Ensemble learning is an impressive machine learning technique that has shown advantages in many applications. The ensemble method uses multiple learning algorithms working in parallel and their outputs are combined using different strategies to achieve better prediction results for the given problem. The core idea of ensemble learning is based on the principle that the generalization ability of an ensemble is better than the single machine learning model. Ensemble learning methods are mainly used because the dataset cannot give sufficient information to choose the best machine learning model, or the search process of the algorithm is not perfect [31].

The ensemble method has member learners or component learners who form the group to make the prediction. The diversity of component learners is a very important factor in the performance of the ensemble. The diversity can be enhanced either by choosing different machine learning algorithms or using different parameters for the same machine learning algorithm.

After selecting diverse member learners, it is important to select the right decision fusion strategy. There are three strategies used for decision fusion: hard voting, soft voting, and stacking.

- Hard Voting: The prediction made by component learners for each class label is added and the class with the highest number of votes is the prediction of the ensemble. For example, there are three component learners and two labels/ classes 0 and 1. Suppose, two-component members predict class 1 as the output, and one predicts class 0, then the ensemble's output will be class 1 given a majority of votes [32].

- Soft Voting: This technique uses component learners' prediction probability of each predicted class. All the prediction probabilities are summed and the class with the highest prediction probability is the prediction of the ensemble learning.

- Stacking: This technique works by adding another layer of a machine learning model. This layer will learn and train on the prediction of the component learners with the real label from the original data to produce the final predictions [33].

5.2 Evaluation matrix

The below methods are used to evaluate the model performance.

- FAR (False Acceptance Rate) FAR is the proportion of times a system grants access to an unauthorized person.

FAR = FP/ (FP + TN) **(4)**

Where, FP = False Positive, TN = Total Negative

- FRR (False Rejection Rate) is the proportion of times a biometric system fails to grant access to an authorized person.

FRR = FN/ (FN + TP) **(5)**

Where, FN = False Negative, TP = Total Positive

- The accuracy of the model is calculated as the number of correct predictions divided by the total number of predictions.

Accuracy = TP + TN/ (TP + FP + FN +TN) **(6)**

- Prediction delay is the time required by trained ML algorithms to make predictions. Generally, ML algorithms do mathematical operations to give predictions and it varies depending upon the complexity of the mathematical operations.

  The prediction delay is calculated by measuring the time difference between the start and end of the prediction function.

5.3 Experiments and Results

All the experiments are carried out on the laptop with 12 GB memory, 512 GB SDD, windows 10 OS, and an intel i5 processor. Additionally, the tools such as Jupyter Notebook and MySQL work bench are utilized along with all the latest versions of ML algorithm libraries required for the experiments. The details of the experiments are explained below.

The raw data from 48 users were captured to extract two behavioral features, mouse click length, and screen location. This data was then used to train the machine learning models described in section III. The dataset was balanced and has a total of 48,000 records, 1000 records for each user, 800 of them used for training and the remaining 200 for testing the model i.e., 80% for training and 20% for testing the machine learning model.

Table 7 Individual ML Model Results

Model Name	Evaluation Matrix			Prediction Delay (sec)
	Accuracy	FAR	FRR	
Light GBM	23.58%	1.62	76.41	34.42
XGBoost	22.92%	1.63	77.08	1.67
Neural Network	15.04%	1.80	84.96	4.20
TabNet	15.45%	1.79	84.54	1.59
1D-CNN	12.71%	1.84	87.29	6.78

Table 7 shows the overall accuracy of the models; moreover, the accuracy of the individual user was calculated for each model i.e., out of 200 test records how many were predicted correctly for each user. Also, the prediction delay for a total of 9600 records (48 users with 200 testing records for each) was measured for each model. Through this analysis, it was found that some models were better at predicting a set of users than other models and vice versa. For example, Figure 5.3 shows the accuracy of the first 25 users for TabNet and XGBoost, and here TabNet was better at predicting some users i.e., it had higher accuracy for them than XGBoost, and similarly, XGBoost was superior in identifying other users. Therefore, to consolidate the prediction power of all the models and improve the overall accuracy of the prediction, experiments with different ensemble techniques were performed. Additionally, employing ensemble learning will have an impact on resource consumption and prediction time will assist to measure it in terms of time.

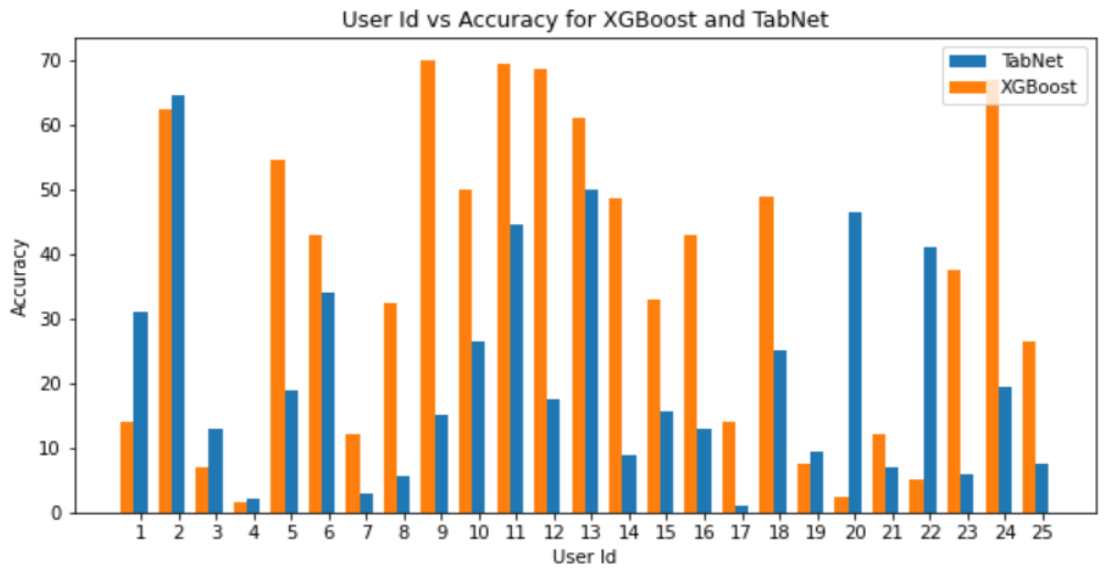


Figure 5.3 Accuracy of users for XGBoost and TabNet Models

Table 8 illustrates the performance of the four voting ensemble models to identify the user based on their accuracy. The ensemble of the top four models i.e., Light GBM, XGBoost, Neural network, and TabNet, had the highest accuracy of 24.03% with soft voting methods. By comparison, the ensemble of the top three (XGB, LGB, and TabNet) and gradient boost models (XGB, LGB) gave almost identical results with accuracies of 23.95% and 23.94% respectively. However, the prediction time required for the ensemble of the top four models is 43.51 sec which is the highest across the board.

Table 8 Voting Ensemble Result

Voting Ensemble Name	**Accuracy**		**Prediction Delay(sec)**
	Hard Voting	**Soft Voting**	
Ensemble of top four models (based on accuracy)	23.28%	24.03%	43.51
Ensemble of gradient boost models	23.58%	23.94%	37.85
Ensemble of deep learning models	15.45%	15.67%	3.03
Ensemble of top three models	23.12%	23.95%	38.16

The results of the stacking ensemble are shown in Table 9; none of the models was able to beat the performance of the best individual model (Light GBM). Also, the prediction time is high.

Table 9 Staking Ensemble Results

Stacking Ensemble Name	**Accuracy**	**Prediction Delay (sec)**
Ensemble of top three models and the top model at the last layer	23.28%	77.58
Ensemble of top three models and second best model at the last layer	23.58%	167.3
Ensemble of top two models and the top model at the last layer	21.47%	14.27

In addition, an ensemble of three XGBoost models with the top three hyperparameter sets determined during hyperparameter tunning was evaluated. XGBoost was selected because of the lowest prediction time and higher side accuracy. However, the accuracy of this ensemble was 23.040%, that’s lower compared to the soft voting ensemble and the prediction time was 3.88 sec.

To summarize, a voting ensemble of the models does improve the performance, especially with the soft voting. Although the prediction time can be a potential problem, it can be solved with high-end infrastructure like multiple GPUs and processors that can do parallel processing which will assist to reduce the prediction delay. Additionally, the accuracy of the models is lower compared to previous research (discussed in chapter 2) which shows that it is difficult to find unique behavioral patterns from real-world data because it is messy compared to the synthetic datasets or the data collected in the labs or controlled environments. The real-world dataset has less variance for individual user, whereas data values are overlapping when considering all the users (shown in Figure 3.5). This makes it difficult to find unique behavioral patterns of the users. Therefore, the highest accuracy of 24.03% is very practical. Moreover, only two behavioral patterns were used to train the models, an increasing number of behavioral patterns will assist models to find more unique behaviors about users and hence improve the overall accuracy.

5.4 Conclusion

In this study, the real-world data of office employees was used to compare multiple machine learning models on the classifications of users based on their behavioral information. Since previous research used controlled environments to collect keystroke data, this might have led to ML model failure for a real-world dataset.

The two behavioral patterns, click length and screen location, were extracted from the raw data and cleaned using the Interquartile range technique. Later, the behavioral data was used to train different machine learning algorithms. However, it was found from the algorithm predictions that each of the models performed differently for different users. To achieve optimal performance for all the users, the prediction power of all the models was combined using ensemble learning.

Five advanced ML algorithms, TabNet, Neural Networks, 1D- CNN, Light GBM, and XGBoost, were trained using the behavioral data of the user. Light GBM had the best prediction accuracy of 23.58% but other models were better at predicting a set of users. Therefore, to combine these predictions, three different ensemble methods, hard voting, soft voting, and stacking, were used. These ensembles were able to increase prediction performance, with the soft voting ensemble of the top four models having the best accuracy of 24.03%. This illustrates the benefits of ensemble learning for continuous user authentication. However, the prediction time required for the ensemble is higher compared to others, but it can be reduced with high end infrastructure.

Chapter 6

6 Database benchmarking to identify the data latency for different databases.

In the digital world, latency is the new outage. In simple words latency means delay. In technology terms, it is the time required to perform any action/ operation. For example, when the user searches on google, the search engine takes time to showcase all the related results. The time difference between entering the query and getting the result is called latency. It is essential to study the latency of the system as it has a major impact on performance.

In the case of continuous user authentication reducing latency is critical as the users are authenticated on an ongoing basis thus any delay can create an opportunity for hackers. Also, as a rule of thumb, the lower the latency, the higher the speed and performance. Therefore, it is crucial to identify and reduce delays. One such major factor causing lag is the performance of the database. It is also called data latency; it is the time taken to store and retrieve the data from the database. In continuous authentication, users' raw data is collected and saved in the database (as shown in Figure 6.1). Here the insert operation is performed and later this data is fetched for feature extraction.

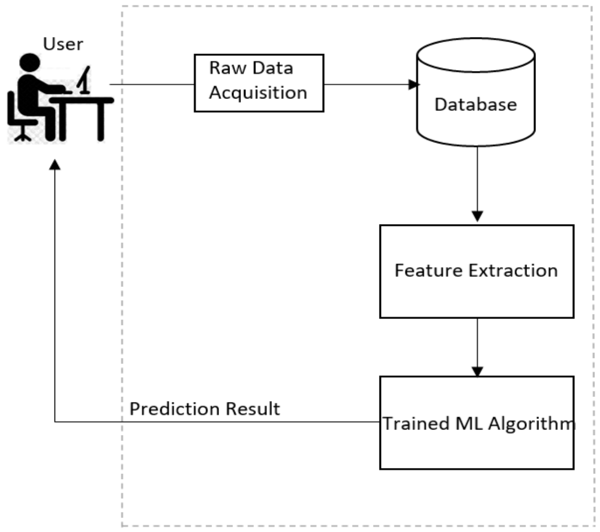


Figure 6.1 Continuous Authentication Architecture

Data latency is key especially when a large amount of data is added or fetched such as continuous user authentication wherein around 50 new raw behavioral data records are generated and stored in the database every second for each user. Considering there are a few hundred users, this would generate millions of records every day. Consequently, it would impact database performance. Hence it is essential to select the appropriate database.

There are many research contributions on database benchmarking, however, all of them are generic i.e., databases are benchmarked for comparison whereas, this study focuses on benchmarking databases, especially on the production-like scenarios of continuous user authentication.

To decide on database selection, a python framework is developed to benchmark databases based on different operations. PostgreSQL and MySQL databases were chosen for comparison based on the nature of the data collected, as it is structured and tabular data. The framework is used to identify the best fit for continuous authentication. Moreover, the framework can be used to compare any database with small changes in the code. The details of the framework and experiments are explained in this chapter.

6.1 Benchmarking Framework

The benchmarking framework is built to simplify the development environment and avoid coding repetitive functions and libraries. This framework can be used to benchmark different databases with little modification in a few functions and configuration variables. Figure 6.2 gives a brief overview of the framework.

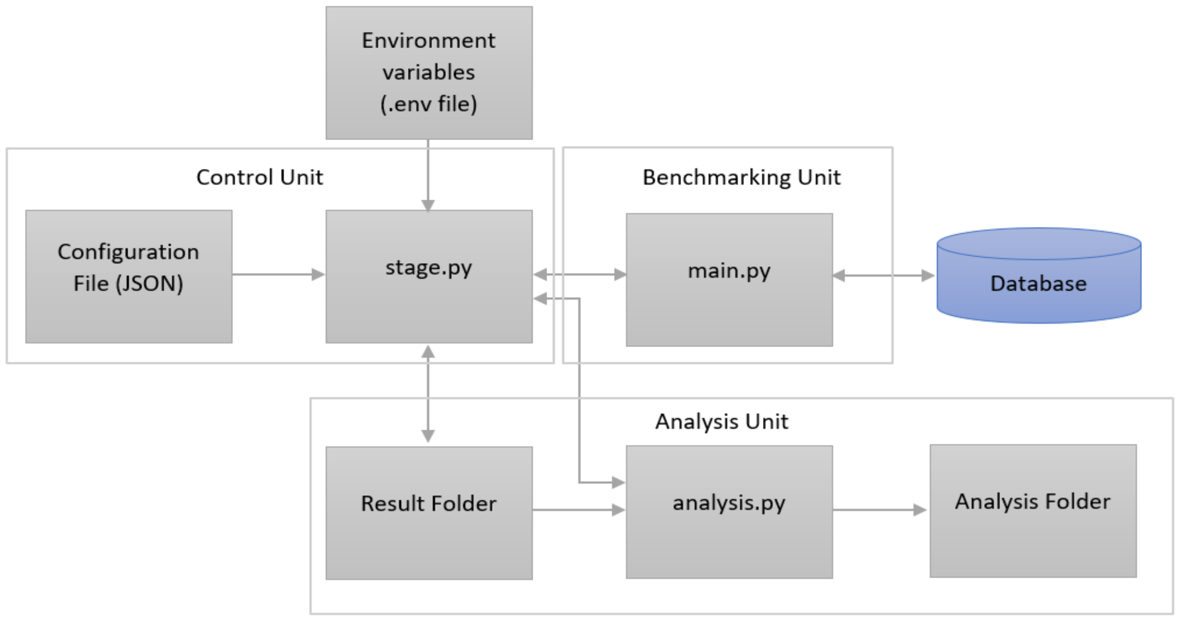


Figure 6.2 Benchmarking framework block diagram

A detailed description of each component is given below.

6.1.1 Control Unit

This unit has the configuration, environment, and stage file necessary to begin the benchmarking experiments. Details of each file are described below.

Configuration file: This is a JSON file where different configuration variables can be set based on the experiment. It has a total of 23 properties, which are of String, array, and Boolean type. The file’s details are shown below.

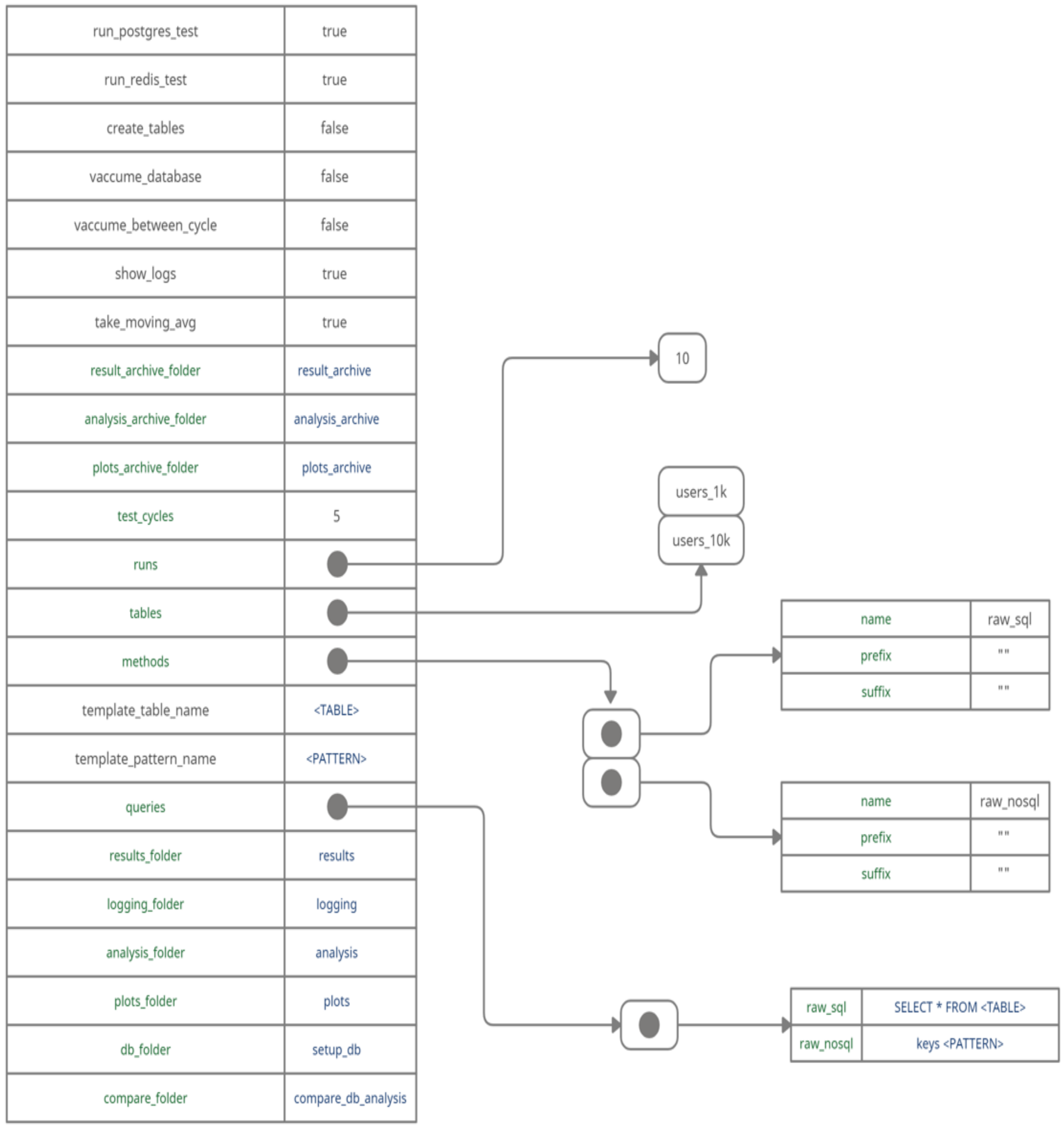


Figure 6.3 Configuration File Variables

The configuration file has more than 40 keys, but some import keys are explained below.

runs: This is an array that determines how many times the query should run, as for instance, if the value of a run is an array of values 10 and 20, then the query would run 10 times and then 20 times.

test_cycles: This key has an integer value that defines how many times the test should be executed for the runs. For example, if the value of runs is 10 and test_cycles is 5, then there will be five cycles with 10 runs in each cycle.

methods: This key has a nested array of an object that defines the name of the method, suffix, and prefix. The suffix and prefix can be used to build the query for the method.

queries: This is a nested JSON object that has queries used for benchmarking the database. Here the method name is used to identify the query.

Environmental variables (.env file): The file has environment variables required to connect to the database such as the database server IP address, port number, username, and password. Each database has an individual environment file to store this data, which is then used to make the database connection.

Stage file: This is a python file that fetches all the configuration and environment variables from respective files and stores them for use in the next phases. Then, based on the configuration variable, all the required operations are performed by calling functions from benchmarking and analysis units.

6.1.2 Benchmarking Unit

The unit has a main.py file, which contains all the python operations required to perform the benchmarking, including functions to connect to the database and perform and save the benchmarking results. In addition, the system information, such as RAM, processor, database version, CPU threads, etc, is saved in a JSON file for future analysis. All these functions are used by the stage.py file for benchmarking.

6.1.3 Analysis Unit

This unit contains all the components required to perform an analysis of the results. The analysis.py file fetches all the results to perform different statistical and graphical analyses as well as compare the results from both databases.

6.2 Benchmarking process flow

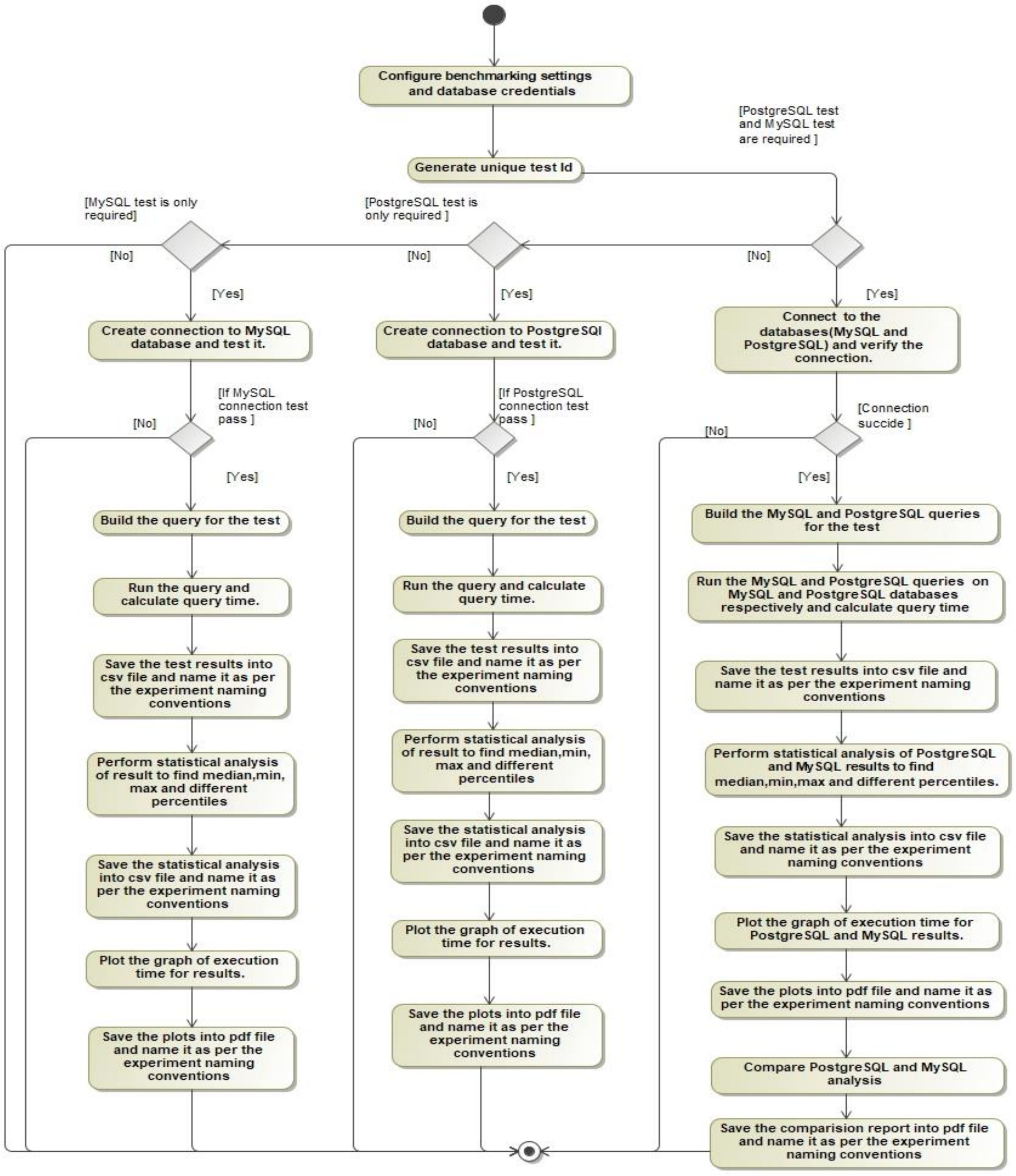

Configure benchmarking settings and database credentials
Generate unique test Id
[PostgreSQL test and MySQL test are required]
[MySQL test is only required]
[PostgreSQL test is only required]
[No]
[No]
[No]
[Yes]
[Yes]
[Yes]
Create connection to MySQL database and test it.
Create connection to PostgreSQl database and test it.
Connect to the databases(MySQL and PostgreSQL) and verify the connection.
[If MySQL connection test pass]
[If PostgreSQL connection test pass]
[Connection succide]
[No]
[No]
[No]
[Yes]
[Yes]
[Yes]
Build the query for the test
Build the query for the test
Build the MySQL and PostgreSQL queries for the test
Run the query and calculate query time.
Run the query and calculate query time.
Run the MySQL and PostgreSQL queries on MySQL and PostgreSQL databases respectively and calculate query time
Save the test results into csv file and name it as per the experiment naming conventions
Save the test results into csv file and name it as per the experiment naming conventions
Save the test results into csv file and name it as per the experiment naming conventions
Perform statistical analysis of result to find median,min, max and different percentiles
Perform statistical analysis of result to find median,min, max and different percentiles
Perform statistical analysis of PostgreSQL and MySQL results to find median,min,max and different percentiles.
Save the statistical analysis into csv file and name it as per the experiment naming conventions
Save the statistical analysis into csv file and name it as per the experiment naming conventions
Save the statistical analysis into csv file and name it as per the experiment naming conventions
Plot the graph of execution time for results.
Plot the graph of execution time for results.
Plot the graph of execution time for PostgreSQL and MySQL results.
Save the plots into pdf file and name it as per the experiment naming conventions
Save the plots into pdf file and name it as per the experiment naming conventions
Save the plots into pdf file and name it as per the experiment naming conventions
Compare PostgreSQL and MySQL analysis
Save the comparision report into pdf file and name it as per the experiment naming conventions

Figure 6.4 Database Benchmarking Activity Diagram

The activity diagram defines the dynamic behavior of the modeled system and assists in understanding program flow at a high level. Figure 6.4 shows the behavior of the database benchmarking framework.

Initially, the configuration variables are set based on the nature of the benchmarking test. Along with configuration variables, environment variables are required for the database. Once this information is provided, the program fetches and stores it until the end of the test and uses the data at different stages. Next, it creates a unique test ID to be used to identify the test. Later, the configuration variables are scanned to determine if both PostgreSQL and MySQL tests are required or not. If they are required, then to run the test, first, a database connection is made and verified, as failure to connect to the database would terminate the test. However, if the connection to both databases succeeds, then again, the configuration variables are used to build the MySQL and PostgreSQL queries for benchmarking. Afterward, the queries are executed on the respective databases. The number of times a query should be executed depends on the configuration variables ‘runs’ and ‘test_cycles’. For example, if the value of ‘runs’ is 100 and ‘test_cycles’ is 5, then the query will be executed 100 times for 5 cycles i.e., a total of 500 times a query will be executed. Adding cycles assists in determining the database performance patterns after executing a set of queries. Simultaneously, the query time/latency (time required by the database to return the results i.e., the time difference between query triggered and query execution completed) is calculated and the results are saved in the CSV file, which is named based on the experiment naming conventions (explained in section 6.1.3). Thereafter, these results are used to perform different types of analyses. First, statistical analysis is used to find the median, min, max, and different percentile values in the results. Second, the graphical analysis plots the graph of query time vs the runs for the complete test and each cycle. Then, the analyses are saved in JSON and pdf files respectively according to experiment naming conventions. These analyses assist in finding the patterns in individual MySQL and PostgreSQL results. However, it is vital to compare the results and analyses from both to understand and compare the patterns throughout the test. The succeeding step creates the comparison pdf, which has plots for both MySQL and PostgreSQL query time and moving average plots to obtain smoother patterns from the test. After this, the test is completed.

On the other hand, if both PostgreSQL and MySQL tests are not required, then the program checks if the PostgreSQL test is required. If it is required, then the connection with the PostgreSQL database is created and tested. If the connection fails, the test is terminated, but otherwise, a query is built using the variables from the configuration file. Following that, the query is executed to calculate the query time and the results are then saved in the CSV file. Subsequently, these results are used for statistical and graphical analyses, which are then saved in JSON and pdf files respectively, marking the end of the test.

Furthermore, if both the PostgreSQL and MySQL tests and PostgreSQL tests are not required, the program checks if the MySQL test is required. If it is required, then the program performs the same steps mentioned for the PostgreSQL test. And if it is not required, then the test will be terminated.

6.2.1 Experiment naming conventions

Result file: Query time/latency results are saved in a CSV file with a name determined according to experiment naming conventions i.e., database name _ table name _ query type_ number of runs _ test Id.

For example, the name “postgresql_users_select_r_100_t_1666812060_1.csv” explains that the database PostgreSQL was tested for the users table by running a select query 100 times.

The below format is used to save the test results.

Table 10 Result Format

Column Name	Date	Table	Query	Execution Time (Sec)	Cycle No
Details/example	Unix Time Stamp	users	Select * from users;	0.005880188000446651	1

Statistical and graphical analysis files: Result data are used to find statistical information like median, max, min, etc. This information is then saved in the JSON with a name such as “analysis_postgresql_users_select_r_100_t_1666812060_1.json” which has a similar meaning to the result file except it has an analysis at the beginning to indicate that it is the analysis file. In the case of graphs, the plots are saved in a pdf file with a name. For example, “plots_postgresql_users_select_r_100_t_1666812060_1.pdf”.

6.3 Experiments and Results

In continuous authentication, the two important database operations are select and insert queries. The system needs to insert the new records coming from the user and fetch the records from the database for user authentication. Hence, it is important to perform benchmarks on these operations in different conditions to identify the database with the lowest latency in a production environment. To achieve these outcomes, the below data seed is used for the experiment.

6.3.1 Data seed for the experiment

For faultless evaluation of databases in production-like cases, the key is to use a dataset similar to production data for benchmarking. Therefore, a table with the below columns was created that replicates the table from the production environment.

Table 11 Dataset details

Field	**Type**
SessionID	bigint
timestamp	int
type	tinyint(1)
x	int
y	int
Event	int
userId	varchar(255)

The table has nine columns, seven of which are of type int or similar and the other two are of varchar and DateTime type. To create data similar to production, a SQL procedure was developed that would insert the records based on the requirement. For example, if the input to the procedure is 1000, then it would add one thousand new records to the table.

6.3.2 Experiment setup

To replicate the production scenario, select and insert queries are executed with different conditions. The experiments are divided into two categories, primary experiments, and complex experiments. In primary experiments, simple production scenarios are evaluated, whereas in complex experiments, production scenarios are intricate. The below table explains the details of the same.

6.3.2.1 Primary Experiments

In production, a query is not executed only once but multiple times, and hence in the experiments, queries are executed multiple times for large datasets to analyze the query latency. For primary experiments, select and insert queries are assessed individually. Table 12 provides the details of the experiment conditions, number of queries executed, and total records in the table.

Table 12 Primary Experiment Details

Experiment Condition	No of times query executed (runs)	Number of records in the table
Select query to fetch all the records from the table	100	1 million
Select query with the condition to fetch record for one user only	100	1 million
Insert new records in the table	100	1 million

6.3.2.2 Complex Experiments

In production, databases do not perform only one operation like select or insert but have to do multiple operations simultaneously, which can degrade their performance. Therefore, it is critical to analyze each database in certain conditions while executing multiple operations simultaneously. For instance, in continuous authentication, the database has to insert new records as well as answer any fetch requests. Hence, to study the database under these conditions, the below experiments were performed.

Table 13 Complex Experiment Details

Experiment Condition	No of times query executed (runs)	Number of records in the table
Select query to fetch all the records from the table was evaluated while the database performs insert operations simultaneously	100	1 million
Select query with the condition to fetch record for one user only while database performs insert operation parallelly	100	1 million
Insert new records in the table while database performs select operation simultaneously	100	1 million

In complex experiments, the select and insert query will be evaluated while the database also performs other operations simultaneously. For example, in the first experiment, data latency for the hundred select queries was calculated whilst the database executed the insert operation simultaneously.

6.3.2.3 Hardware/Software Details

The experiments were performed on the computer with the below system details.

Table 14 System Details

Database Type	MySQL and PostgreSQL
Database Kind	SQL
Database Version MySQL	8.0.20
Database Version PostgreSQL	PostgreSQL 14.6
Operating System	Windows-10-10.0.19041-SP0
System Memory	11.650901794433594
CPU Type	Intel64 Family 6 Model 140 Stepping 1, GenuineIntel
Total Cores	8
Total Threads	1

6.3.4 Results

A total of six different experiments (Table 12 and 13) was performed to evaluate the databases in different production-like scenarios. The results of both primary experiments and complex experiments are discussed below.

6.3.4.1 Primary Experiments results

1. **Select query to fetch all the records from the table**

To begin, a select query to fetch all the records was executed 100 times and the performance was recorded on both PostgreSQL and MySQL databases.

Figure 6.5 shows the execution time required to fetch 1 million records one hundred times on the MySQL database. It can be observed that the execution time i.e., query latency varies between 9ms to 12ms, whereas Figure 6.6 shows query latency for select queries on PostgreSQL. The execution time was between 0.6ms to 0.8ms, which is very low compared to MySQL.

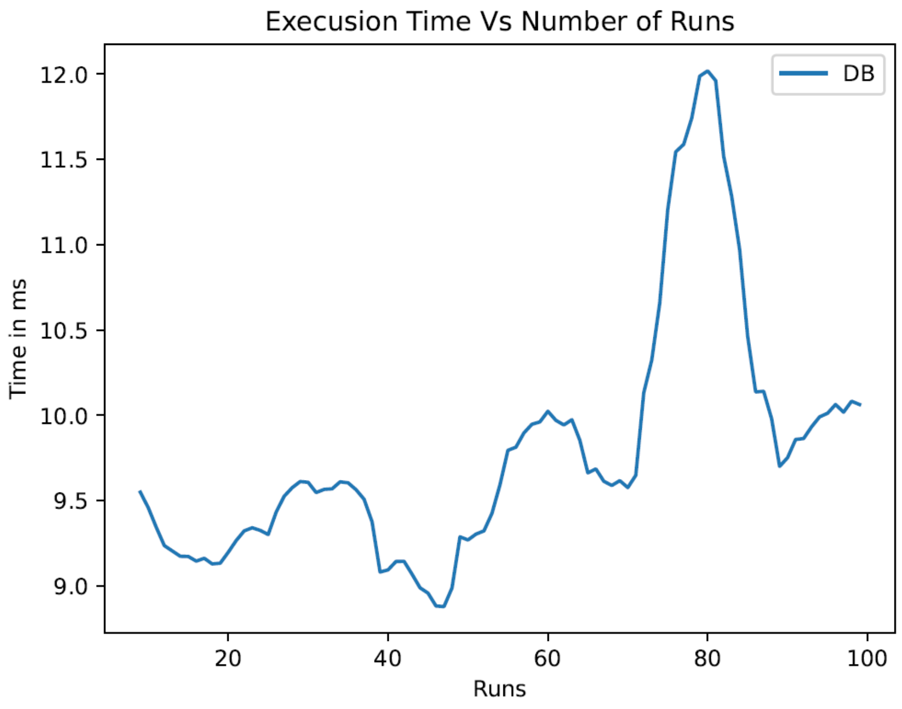


Figure 6.5 Select Query Execution Time MySQL for Primary Experiment One

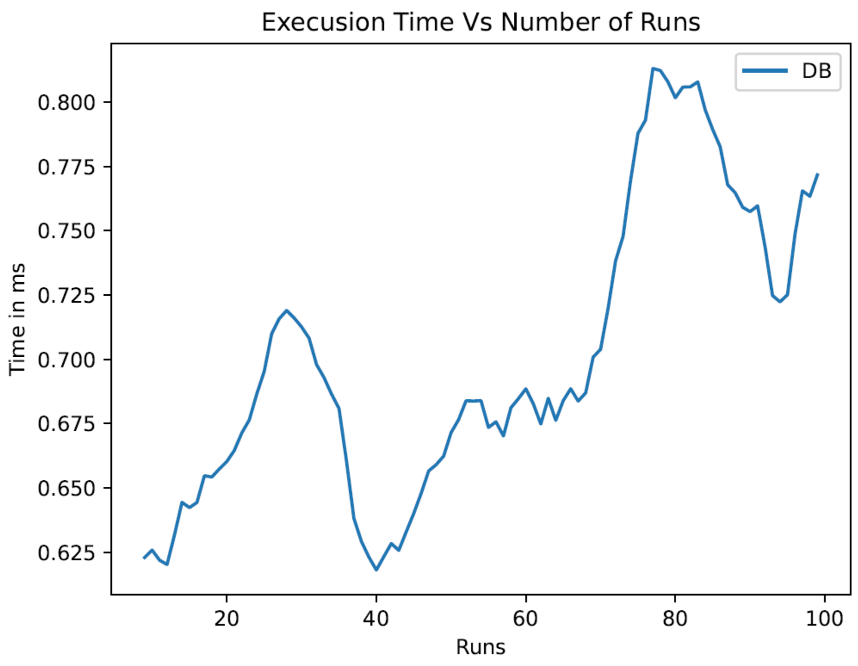


Figure 6.6 Select Query Execution Time PostgreSQL for Primary Experiment One

Figure 6.7 compares the execution times to retrieve 1 million records from PostgreSQL and MySQL. Clearly, PostgreSQL performs way better than MySQL in fetching all the records from the table.

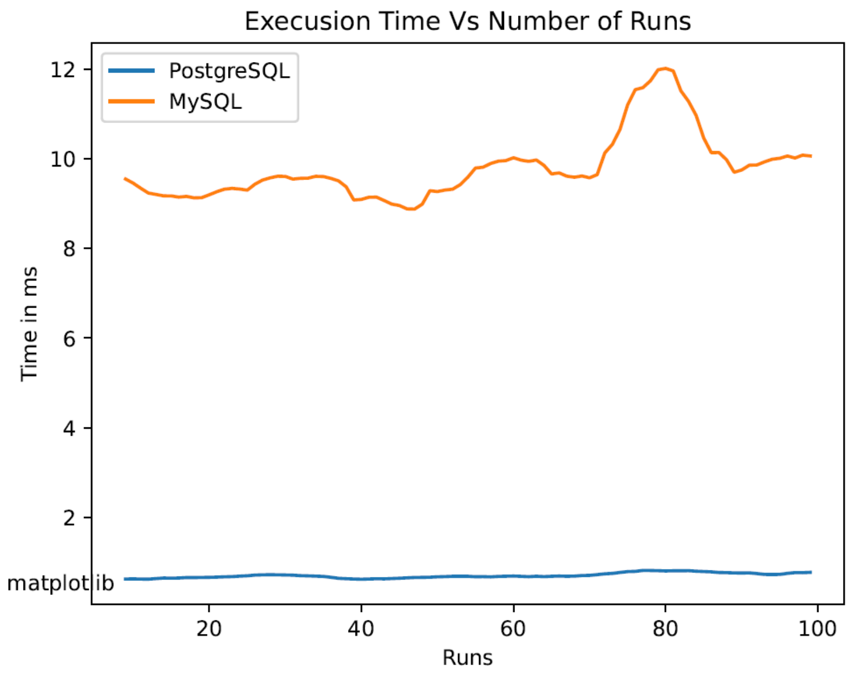


Figure 6.7 Select Query Comparison of MySQL and PostgreSQL for Primary Experiment One

The Table 15 shows the different statistics for the execution times of both databases. The table has the median, maximum, minimum, and percentile values of the results. These statistics will assist to evaluate the performance of the database.

For the first experiment, the difference between the median value of MySQL and PostgreSQL is huge and clearly, PostgreSQL outperforms MySQL in all the select query stats. The performance of PostgreSQL is 13 times better than MySQL.

Table 15 Statistics MySQL and PostgreSQL for Primary Experiment One

Execution Time Stats	MySQL(ms)	PostgreSQL(ms)
Median	9.610416149999999	0.6922907000000014
Max	14.653671499999971	0.9570205999999928
Min	6.749867999999992	0.4895766000000003
Percentile 25%	9.338115	0.644354
Percentile 50%	9.610416	0.692291
Percentile 75%	9.962929	0.743065

2. **Select query with the condition to fetch records for one user only**

The second experiment was to test the performance of a select statement with a where clause because, in production, it is often necessary to fetch data for a specific user based on different criteria. Therefore, to benchmark the databases on this condition, the data was fetched using a select statement with username in the where condition. For example, ‘select * from data where uname =’clair’.

Using the query above, around ten thousand records were fetched from 1 million records for the username ‘clair’. To fetch the data, MySQL took between 0.9ms to 1ms while PostgreSQL required around 0.09ms to 0.13ms. Again, MySQL does not perform as well as PostgreSQL.

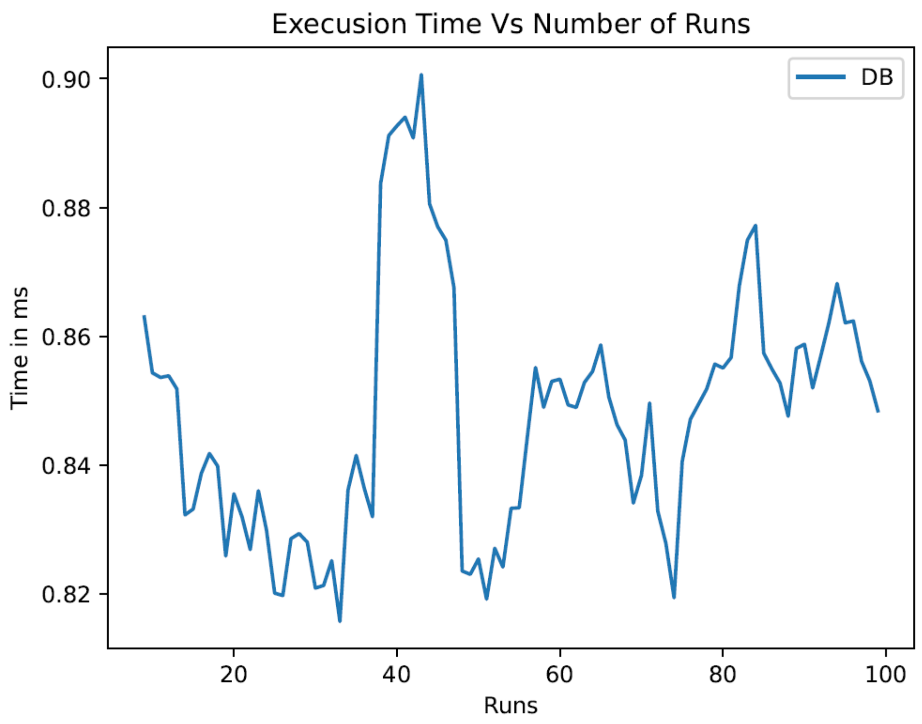


Figure 6.8 Select with Where Condition Query Execution Time MySQL for Primary Experiment Two

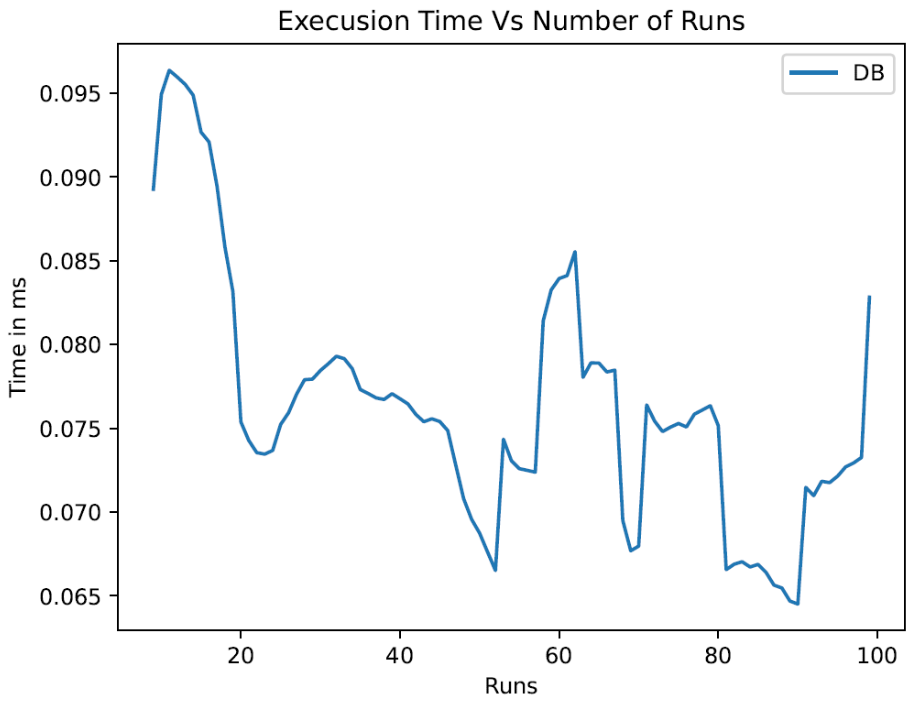


Figure 6.9 Select with Where Condition Query Execution Time PostgreSQL for Primary Experiment Two

Figure 6.10 compares the MySQL and PostgreSQL results for select query with where clause, the performance of MySQL is poor compared to PostgreSQL. MySQL is not even close to PostgreSQL.

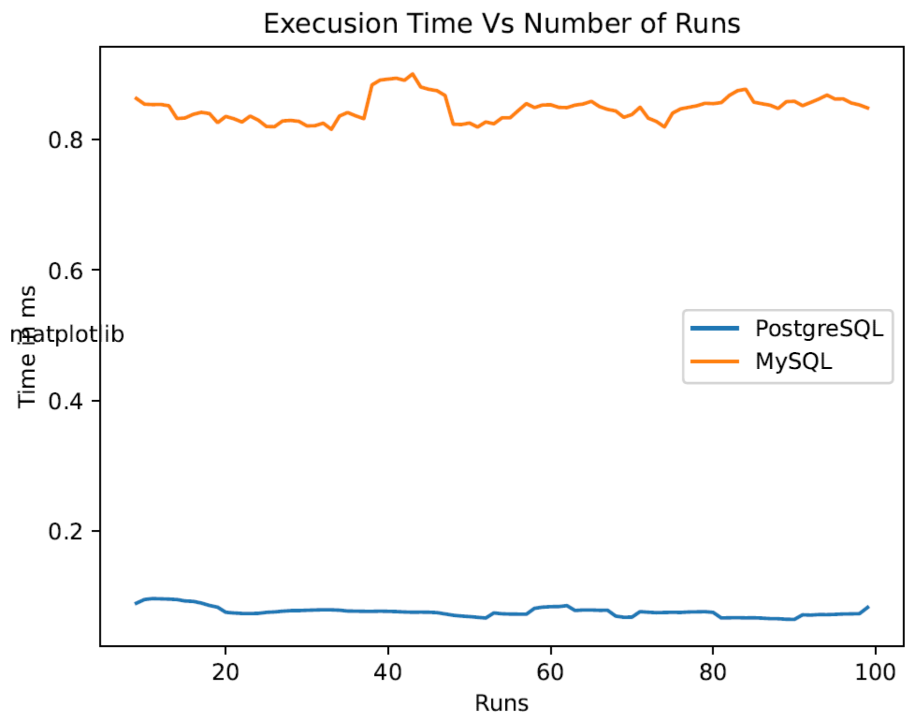


Figure 6.10 Select with Where Condition Query Comparison of MySQL and PostgreSQL for Primary Experiment Two

The Table 16 shows the stats for the experiment. Undoubtedly, PostgreSQL beats MySQL at every stage. The median execution time for PostgreSQL is 0.0726ms, whereas for MySQL it is 0.8428ms. Moreover, the difference between the minimum and maximum value for PostgreSQL is 0.091 and for MySQL, it is 0.66. It shows that the performance of both databases does not change drastically throughout the experiment.

Table 16 Statistical Comparison for Primary Experiment Two

Execution Time Stats	MySQL(ms)	PostgreSQL(ms)
Median	0.8438375000000011	0.07268409999999956
Max	1.3509363000000008	0.1564174999999998
Min	0.6974038000000036,	0.0596166
Percentile 25%	0.80234	0.066272
Percentile 50%	0.843838	0.072684
Percentile 75%	0.869998	0.078394

3. Insert new records in the table

The next important operation is data insertion; in continuous authentication, a large amount of data is generated every second. Thus, to assess the performance of the databases, the experiment with insert query was conducted where the query was executed 100 times on both databases.

Figures 6.11 and 6.12 show the results of this experiment, where there is not much difference between the performance of MySQL and PostgreSQL. Both databases took around the same time to insert the record on the database. The execution time varies between 0.0010ms to 0.0030ms for MySQL and 0.0007ms to 0.0014ms for PostgreSQL. Additionally, the execution time for the first few records was high for both databases. However, after that the performance was stable, and databases took around a similar time to add new records.

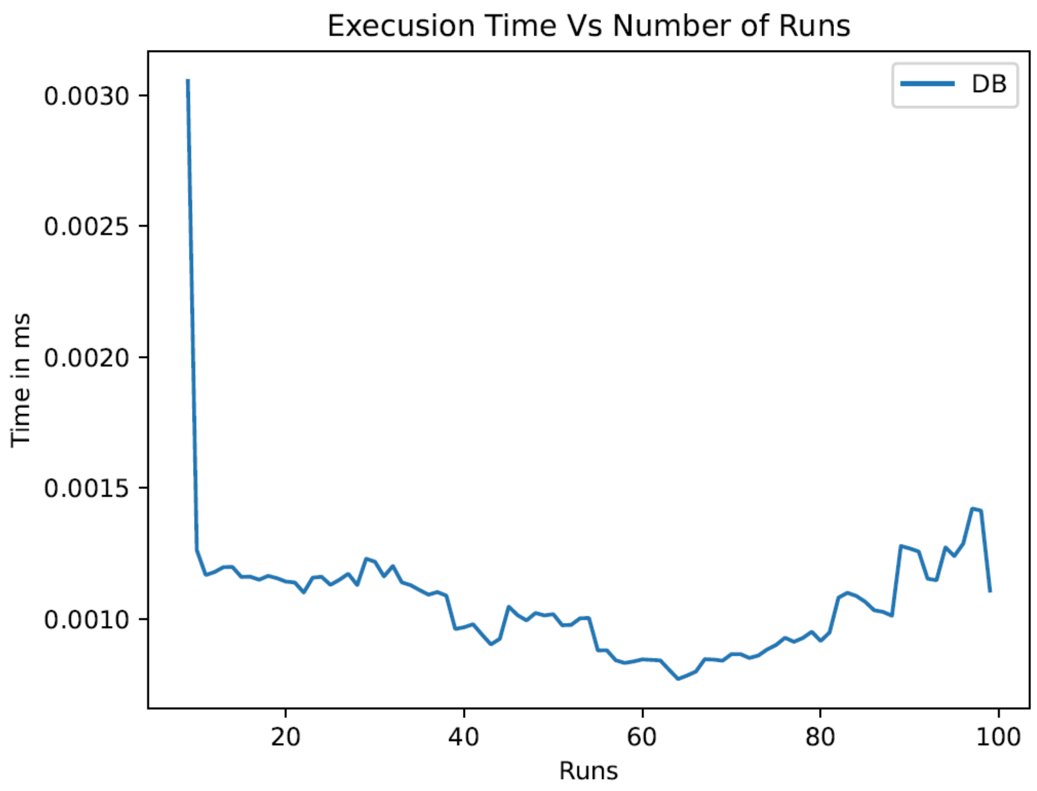


Figure 6.11 Insert Query Execution Time MySQL for Primary Experiment Three

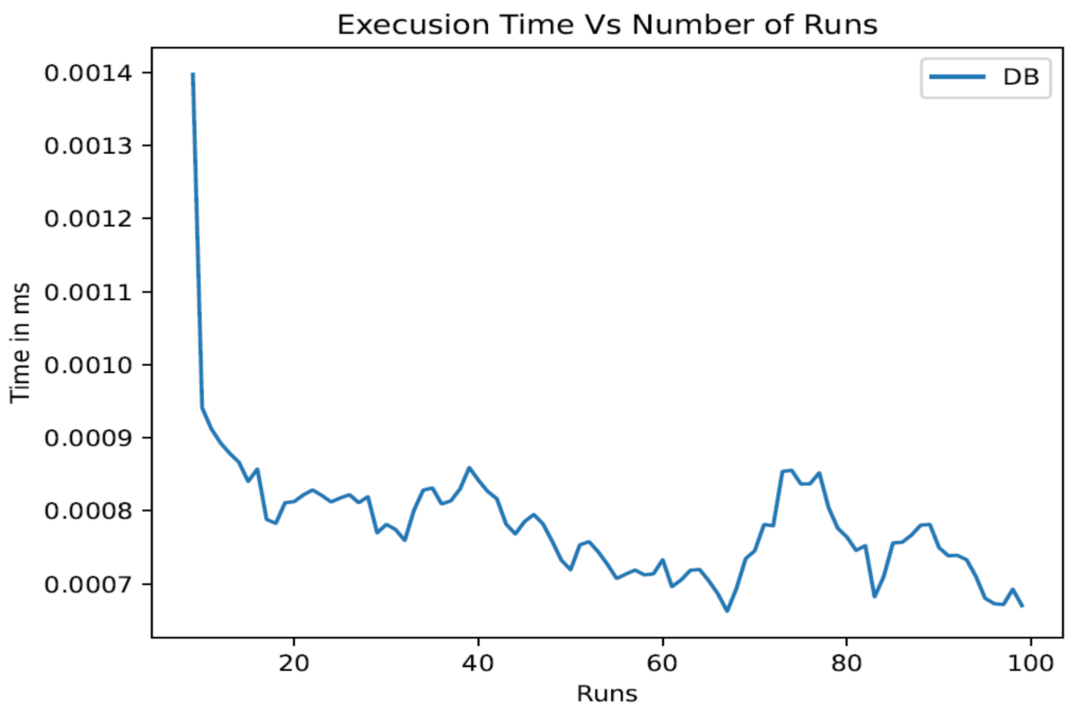


Figure 6.12 Insert Query Execution Time PostgreSQL for Primary Experiment Three

Figure 6.13 shows the execution time graphs for both databases. The orange line was for MySQL and the blue line was for PostgreSQL. The performance of the databases was comparable, but PostgreSQL performs slightly better than MySQL.

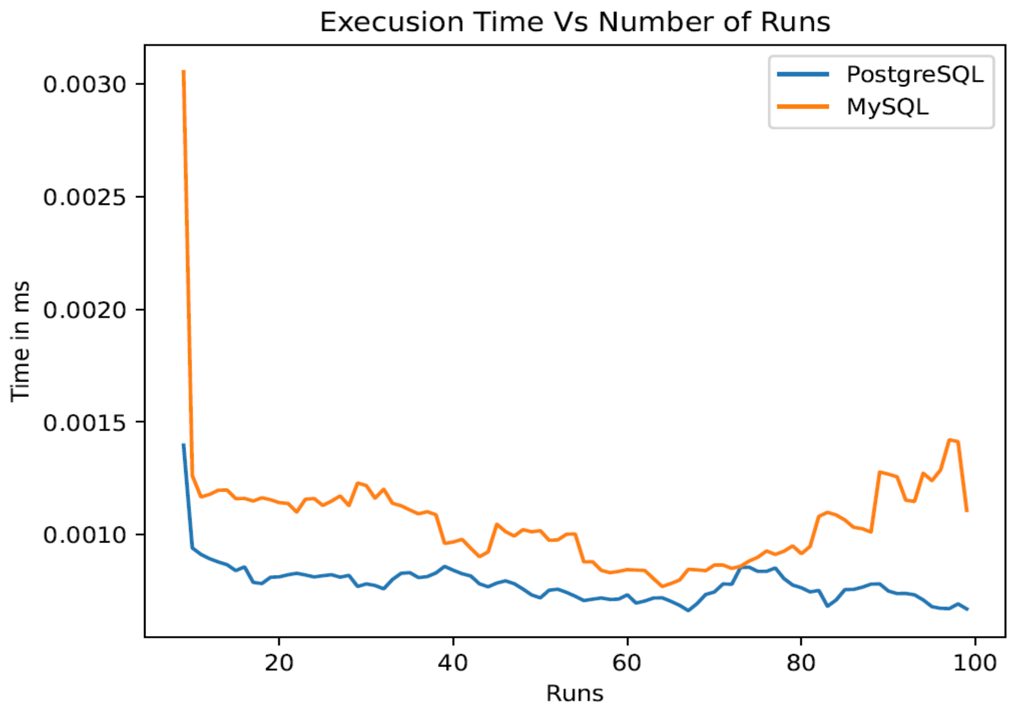


Figure 6.13 Insert Query Comparison of MySQL & PostgreSQL for Primary Experiment Three

The statistic from Table 17 confirms that the MySQL and PostgreSQL performance overlapped. The difference between all the statistical values is very small. Also, the performance of MySQL is better compared to the first two experiments i.e., fetching all the data and selecting the data based on the where clause.

Table 17 Statistical Comparison for Primary Experiment Three

Execution Time Stats	MySQL(ms)	PostgreSQL(ms)
Median	0.0002600000000001	0.00012070000000005
Max	0.0030774	0.0005792
Min	0.0001705999999999	8.740000000018178e-05
Percentile 25%	0.000224	0.000107
Percentile 50%	0.00026	0.000121
Percentile 75%	0.000305	0.000146

6.3.4.2 Complex Experiments Results

In production, databases are required to handle multiple connections and requests. Therefore, to simulate a similar environment, the experiments were performed with the below scenarios.

1. **Select query to fetch all the records from the table was evaluated while the database performs insert operations simultaneously**

In continuous authentication, insert and select operations are critical. The user's data is continuously added to the databases and at the same time data is fetched for feature extraction. The database needs to perform both operations every few seconds. Hence this scenario was added to evaluate the select operation performance when insert operations were executed simultaneously.

Figure 6.14 shows the performance of the MySQL select operation while insert queries were executed simultaneously. The execution time varied between 7ms to 13ms, whereas, for PostgreSQL, it ranged from 0.7ms to 0.9ms. The execution time for both databases is increasing after every execution because the new data has been added simultaneously, therefore, the databases had to fetch more records every time.

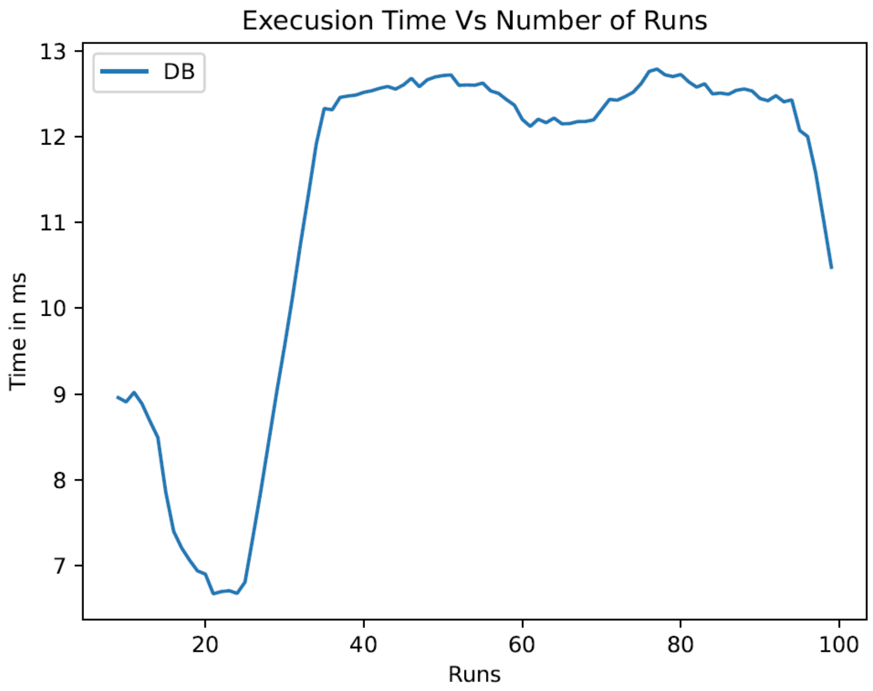


Figure 6.14 Select Query Execution Time MySQL with Insert Operation in Parallel

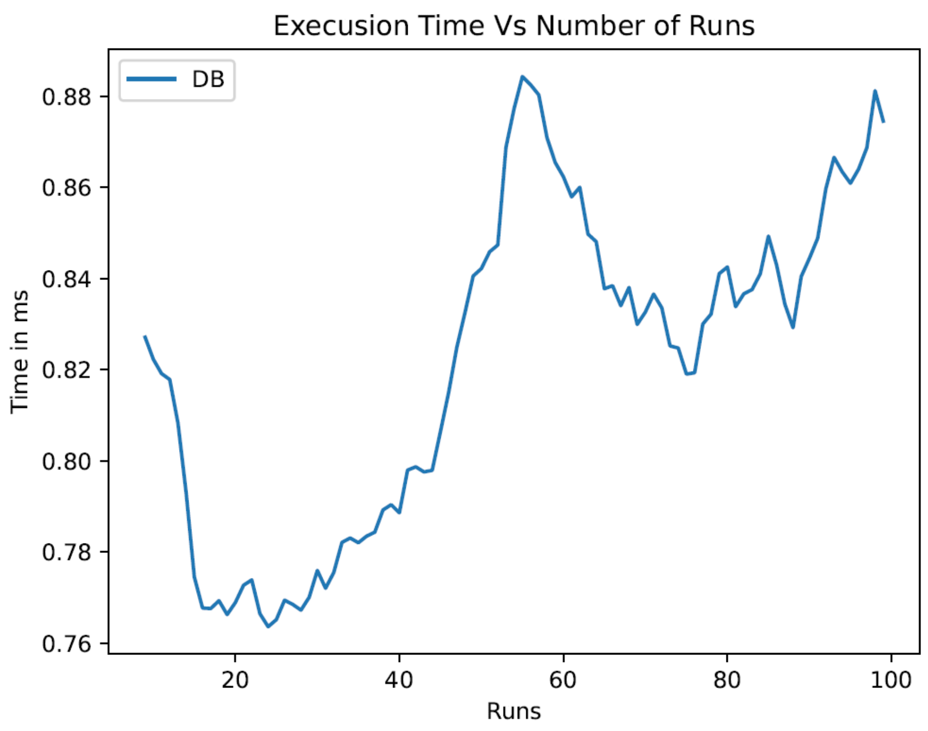


Figure 6.15 Select Query Execution Time PostgreSQL with Insert Operation in Parallel

Figure 6.16shows the comparison of MySQL and PostgreSQL for the select operation while the insert queries were executed simultaneously. The performance of PostgreSQL had very little impact of the increasing number of records, whereas MySQL performance was highly impacted because of the increasing number of records and simultaneous operations.

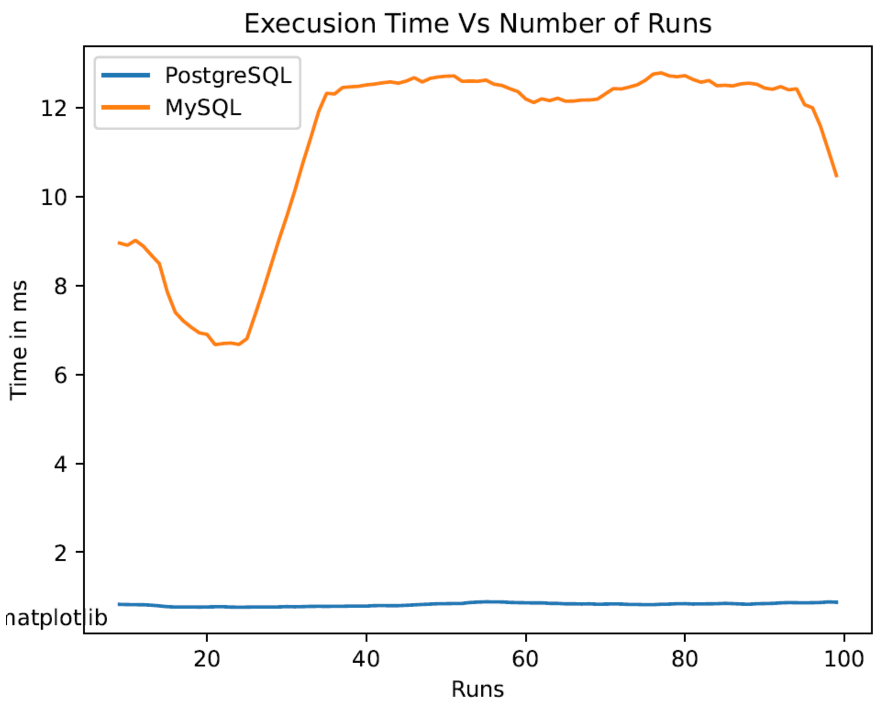


Figure 6.16 Select Query Comparison of MySQL & PostgreSQL with Insert Operation in Parallel

The Table 18 shows the statical comparison of PostgreSQL and MySQL. The difference between minimum and maximum is very high for MySQL when compared with PostgreSQL. This indicates that the MySQL performance would degrade with an increasing number of records, whereas the performance of PostgreSQL was stable with all the changes.

Table 18 Statistical Comparison for Complex Experiment One

Execution Time Stats	MySQL(ms)	PostgreSQL(ms)
Median	12.228753049999938	0.8172035000000051
Max	13.367786599999988	1.0093484000000004
Min	6.454262799999924	0.7369057999999953
Percentile 25%	8.864012	0.782431
Percentile 50%	12.228753	0.817204
Percentile 75%	12.642323	0.858551

2. **Select query with the condition to fetch records for one user only while database performs insert operation parallelly**

In the second case, the select operation with a where the condition was executed with insert queries running in parallel. The select query used for the experiment was “select * from data where uname =’clair’” but no additional records for username ‘clair’ was inserted with insert query because in the previous experiment the performance of the select operation was already analyzed for the increasing number of records. Therefore, this experiment evaluates the performance of select with the where clause while the insert operation is performed parallelly. However, the number of records for select operations was the same.

Figures 6.17 and 6.18 show the performance of MySQL and PostgreSQL respectively. PostgreSQL beats MySQL and performs better in fetching records.

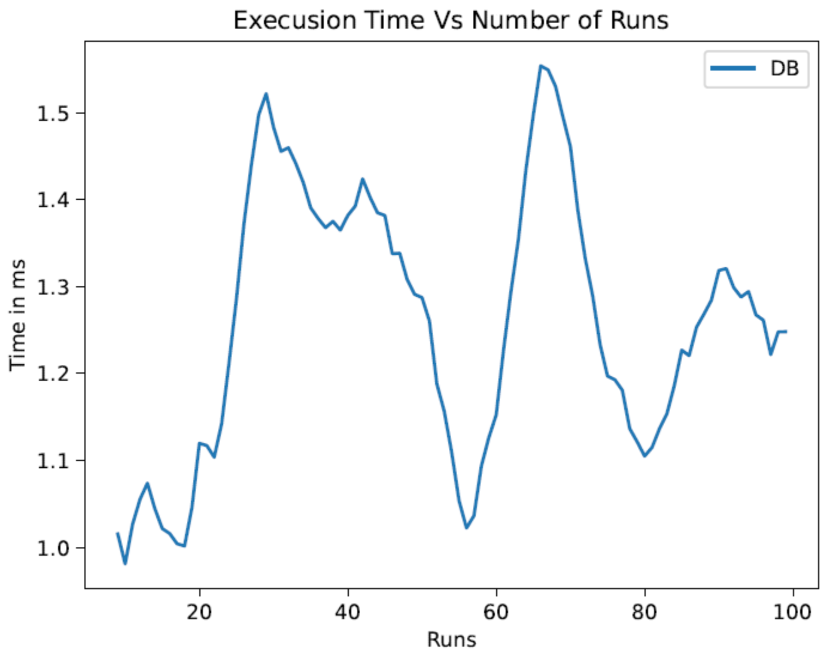


Figure 6.17 Select with Where Query Execution Time MySQL with Insert Operation in Parallel

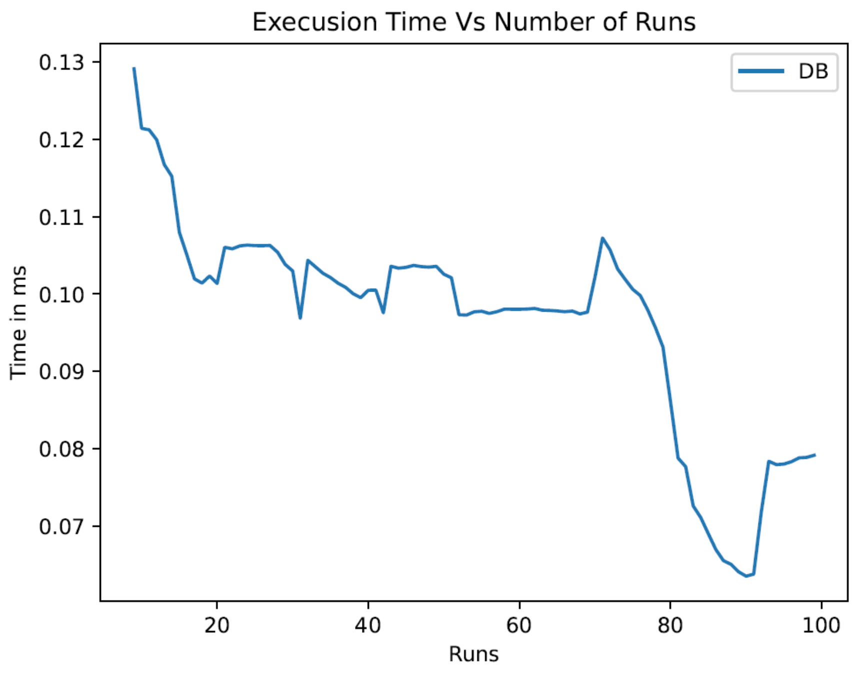


Figure 6.18 Select with Where Query Execution Time PostgreSQL with Insert Operation in Parallel

Figure 6.19 shows the performance of both databases in terms of execution time. The execution time for MySQL varies between 1 ms to 1.5 ms, whereas for PostgreSQL it is between 0.07 ms to 0.13 ms. Additionally, The performance of MySQL is degraded compared with the performance of the second primary experiment in which only select with where clause was evaluated. The median time for the primary experiment was 0.84ms but for this experiment, it soared to 1.25ms. Whereas for PostgreSQL the median time for the primary experiment was 0.072ms and in this experiment, it rose to 0.09ms. This shows that the performance of MySQL degrades faster compared to PostgreSQL when select and insert operations are performed parallelly.

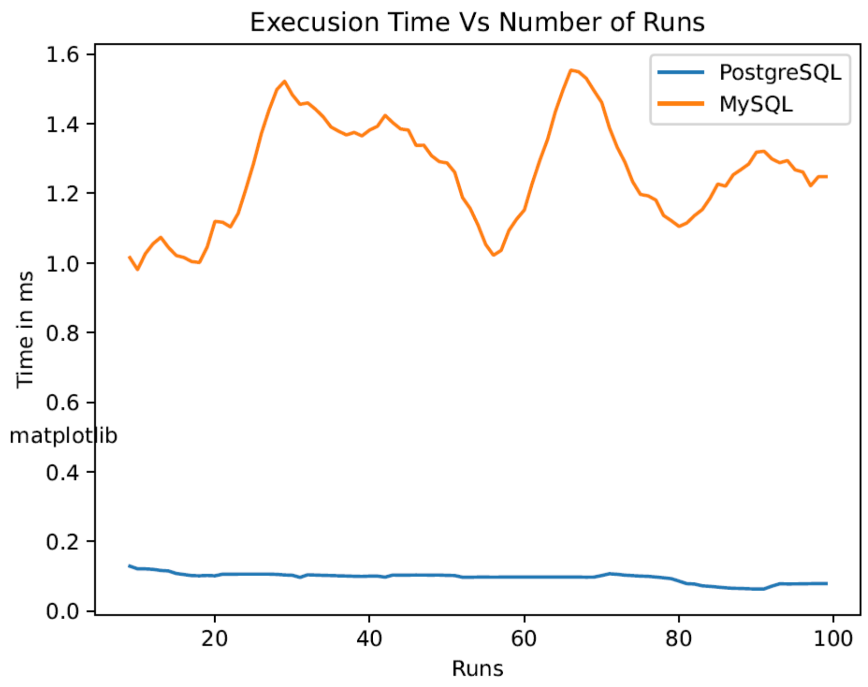


Figure 6.19 Select with Where Query Comparison of MySQL & PostgreSQL with Insert Operation in Parallel

The table below displays the statistics for MySQL and PostgreSQL. Again, PostgreSQL performs around 9 times better than MySQL.

Table 19 Statistical Comparison for Complex Experiment Two

Execution Time Stats	MySQL(ms)	PostgreSQL(ms)
Median	1.253879149999996	0.0929318000000001
Max	1.7836954999999932	0.1886225999999999
Min	0.7546789999999994	0.0586152000000002
Percentile 25%	1.106501	0.086717
Percentile 50%	1.253879	0.092932
Percentile 75%	1.439175	0.10246

3. Insert new records in the table while database performs select operation simultaneously

The last scenario evaluated the performance of the insert operation when select queries were executed parallelly. Figures 6.20 and 6.21 show the performances of both databases. The performance of MySQL varies between 0.00020ms to 0.00043ms, whereas for PostgreSQL it changes between 0.00010ms to 0.00017ms. Hence, distinctly PostgreSQL performs better to insert new data while select queries were executed at the same time.

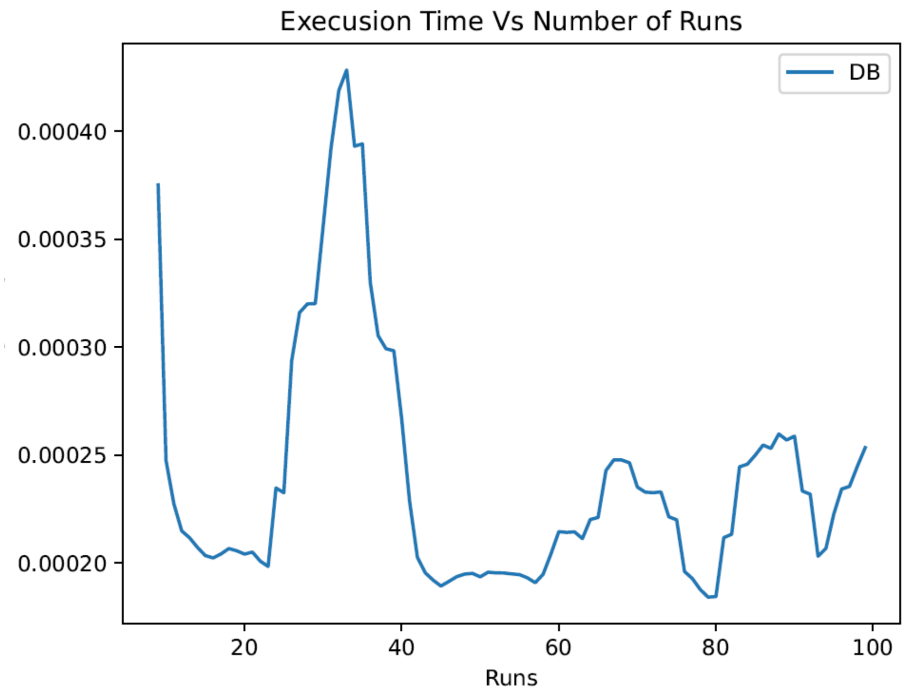


Figure 6.20 Insert Query Execution Time MySQL with Select Operation in Parallel

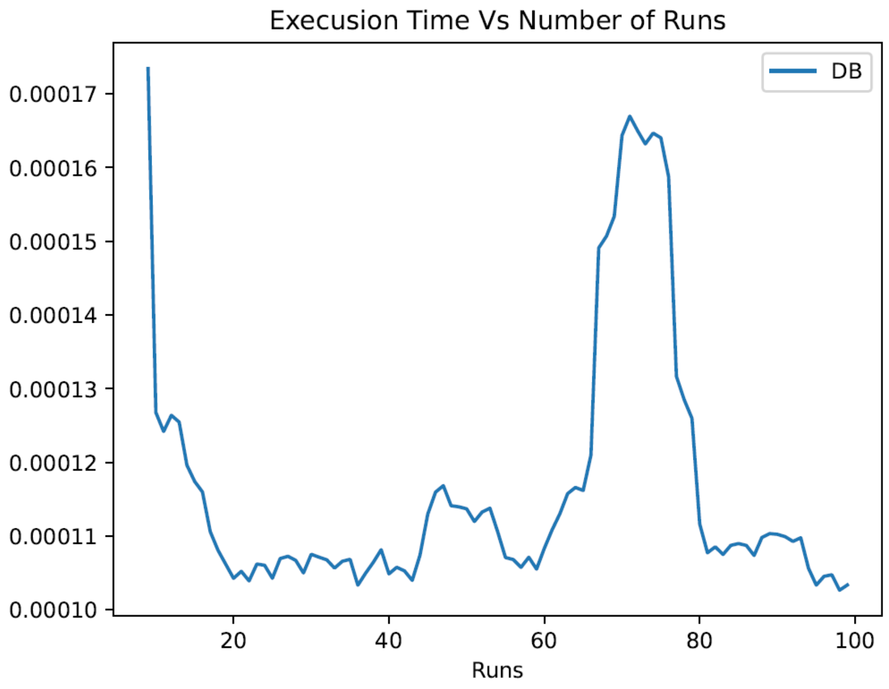


Figure 6.21 Insert Query Execution Time PostgreSQL with Select Operation in Parallel

Figure 6.22 compares the execution time results of both databases. Evidently, PostgreSQL performed better. Additionally, if compared with the third primary experiment where only the insert operation was analyzed and the results were very similar for both the databases, the results of this experiment were different and there was a clear winner. This shows that the performance of PostgreSQL is not changed much but for MySQL, it was lower.

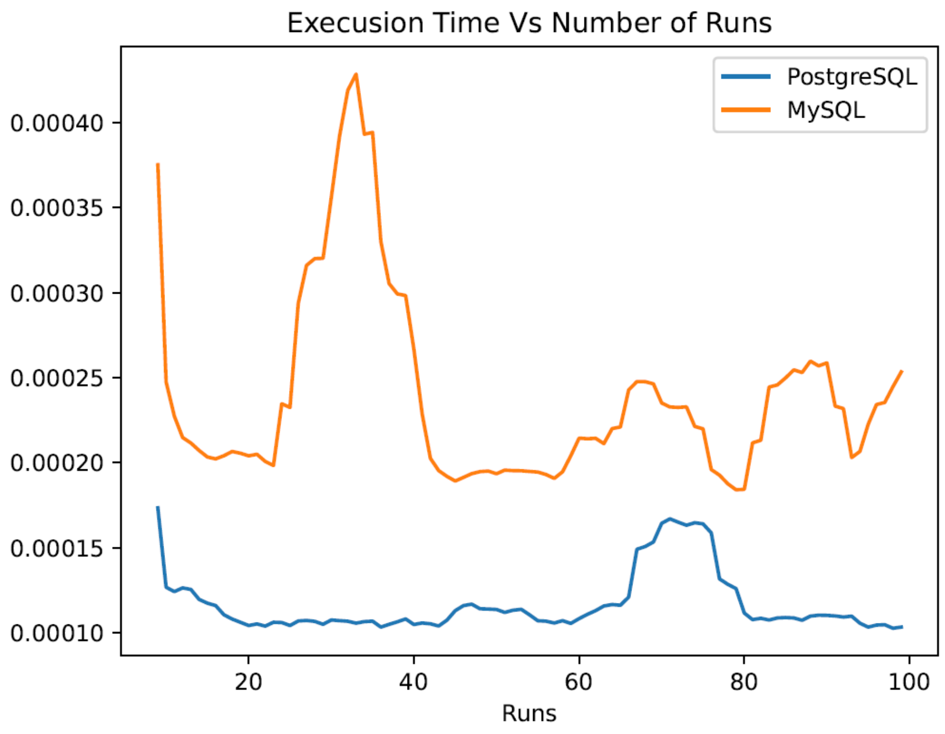


Figure 6.22 Insert Query Comparison of MySQL & PostgreSQL with Select Operation in Parallel

Table 20 shows the statistics for both MySQL and PostgreSQL. The median value for MySQL in this experiment was 0.00020ms, whereas for PostgreSQL it was 0.00010ms. PostgreSQL performs twice better as that as MySQL. Also, the execution time variation was more for MySQL compared to PostgreSQL.

Table 20 Statistical Comparison for Complex Experiment Three

Execution Time Stats	MySQL(ms)	PostgreSQL(ms)
Median	0.00020265000000005	0.00010899999999995001
Max	0.001465	0.0005941999999999
Min	0.0001616000000002	9.090000000000488e-05
Percentile 25%	0.000187	9.9e-05
Percentile 50%	0.000203	0.000109
Percentile 75%	0.000238	0.000121

6.4 Conclusion

To summarise, database latency is very critical in the overall performance of continuous authentication because if the database takes a long time to inset or fetch the data, then the time required to authenticate the user will increase and create an opportunity for the hacker. Therefore, a universal benchmarking framework has been developed to evaluate the performance of the databases. PostgreSQL and MySQL databases were selected because the data generated for continuous authentication is tabular and hence the two databases are best suited for it. These databases are then evaluated using a similar type of data and production-like scenario to identify the best database for the system.

The experiments were carried out for six different cases/operations that are frequently used by databases in continuous authentication. To begin, a select all operation was evaluated in fetching all the records from the database. Next, the select operation with where clause was tested, and lastly the insert operation was investigated. In the later stage, more complex operations were performed to evaluate the performance. All the above-mentioned operations were executed in parallel with other database operations to replicate production scenarios.

The results of the experiments show that the performance of PostgreSQL is around 9 times better than MySQL for select operations both with and without a where clause. However, the insert operation results were very similar for both databases. Additionally, the performance of PostgreSQL is more stable when multiple operations are executed at the same time. Whereas the performance of MySQL lowers drastically when evaluated for complex experiment conditions. Therefore, based on the results, PostgreSQL is better suited for continuous authentication as it has low data latency.

Chapter 7

7 Conclusion and Future Work

This chapter discusses the research conclusion and future work, including proposals to try new but different methods.

7.1 Conclusion

In this day and age, most data are stored and accessed online whether social media, banking or education. Additionally, COVID has been a catalyst expediting the digital transformation. Most businesses have accelerated the digitalization of their customer and supply-chain interactions and of their internal operations by three to four years. On the other side, consumers have also moved dramatically towards online channels. And the largest changes are also the most likely to stick in the long term. However, rushed implementation and lack of due diligence will almost certainly expose vulnerabilities in systems that were put in place to adapt to remote work. Sectors such as telecom, banks, and government are especially at risk as they collect large volumes of customer data. For instance, in October 2022 hackers targeted a communications platform in Australia, which handles Department of Defence data, in a ransomware attack. The government believes hackers breached sensitive government data in this attack. Many such instances are occurring around the world and so it is critical to have robust cyber security to protect against cyber attacks.

For any organization, the user identity review is important as it is a critical component of Identity and Access management. Only legitimate users should have access to the systems and applications. Therefore, companies use authentication methods, such as passwords, passcodes, access cards, fingerprints etc. All of these are static authentication methods i.e., the identity of the user is verified at the beginning of the session, but there is not validation throughout the session. If the user credentials are breaches, then hackers can access all the data that the user has access to. To prevent such breaches, an advanced solution is to integrate continuous authentication. Continuous authentications utilize users' behavioural information to confirm their identities on an ongoing basis. However, continuous authentication is a new technology and has some gaps. In this study, we have proposed potential solutions for the gaps which will help in the commercialization of continuous authentication technology.

To begin, the raw behavioural data of 48 employees of a financial organization were collected. This raw data cannot be used directly with classification algorithms as they include no specific user behavioural information. Therefore, an algorithm was developed to extract two behavioural data points, mouse click length, which is the time difference between the mouse key press and release, and screen location, i.e., in what part of the screen the click was made. However, both these behavioural pattern data points had some irregularities which could have a negative impact on the ML classification algorithms. Therefore, the data was cleaned using the Inter-quartile range technique to remove the outliers. This cleaned data was then used for all the experiments.

In this research, a novel approach for reducing the data collection time and hence the registration time was proposed. The transfer learning technique was effectively used to improve the accuracy of the model for small amounts of data. To do so, the base model was trained and optimized using 48,000 records. Afterwards, all the learning from the base model was transferred to the new model and the model was trained for five new users. The model gave 9.76% higher accuracy than the model trained from scratch without transfer learning. This increase will make the system ready for earlier use than other systems. It can also help in the commercialization of continuous authentication.

In the second stage of the research, a real-world dataset was used to evaluate different machine learning models. In previous research, synthetic data or data collected in control environments was used, which cannot give correct evaluation models because the performance of ML models changes with the nature of the data. Therefore, it is important to evaluate ML models on real-world datasets. Therefore, in this study, different ML algorithms were analyzed on real-world datasets. Through this analysis. it was found that each model was better at predicting a set of users; therefore, to merge the prediction capability of all the models, ensemble learning was used, which improved the accuracy of the prediction. However, using ensemble learning requires higher processing power; hence high infrastructure is required to utilize ensemble learning.

In the last part of this study, a universal database benchmarking framework is developed to analyze the performance of the databases and choose the best performing database for continuous authentication systems. PostgreSQL and MySQL databases were selected for the evaluation based on the nature of the data generated in continuous authentication systems. These two databases were evaluated for different production-like scenarios of continuous authentication systems. The results indicate that PostgreSQL is multiple times better than MySQL at handling different production-like scenarios of continuous authentication.

7.2 Future Work

Two behavioral patterns were used in this study, but in the future, more behavioural patterns can be extracted from the raw data and used to train the models. Increases in the number of behavioural patterns will assist models in learning more precise behaviours of the user and hence increase the predictive accuracy. Additionally, using ensemble learning will help combine the prediction power of different ML algorithms to further enhance accuracy.

Additionally, to reduce overall latency and improve the performance, federated learning can be used. Federated learning is a machine learning technique that trains an algorithm across multiple decentralized edge devices holding local samples without exchanging them. In many cases, only one user is supposed to access the device, as for example in an organization, a laptop or desktop is used by only one employee and no one else should access it. Similarly, a personal mobile device is handled only by the owner. In such scenarios, federated learning can be used in which the data are collected and processed locally on the device and later used to train ML algorithms such as one class classifier or anomaly detection. Both these models can be trained on a single user’s behavioural data and any other behaviour would be marked as intruder/ attacker. Applying this technique would extraordinarily reduce the latency as all the processes will execute on the same device. Additionally, it can solve the data privacy issue as the user data is not shared with any other servers, applications, or systems.

References

[1] "Cybercrime To Cost The World $10.5 Trillion Annually By 2025," Newswire, 13 November 2020. [Online]. Available: https://www.prnewswire.com/news-releases/cybercrime-to-cost-the-world-10-5-trillion-annually-by-2025--301172786.html.

[2] "67 Percent of Breaches Caused by Credential Theft, User Error, and Social Attacks," NetSec.news , [Online]. Available: https://www.netsec.news/67-percent-of-breaches-caused-by-credential-theft-user-error-and-social-attacks/#:~:text=The%20report%20revealed%20that%20the,and.

[3] L. S. Vailshery, " Number of Internet of Things (IoT) connected devices worldwide from 2019 to 2021, with forecasts from 2022 to 2030," 2022.

[4] E. Sayegh, "www.forbes.com," Forbes, 1 10 2021. [Online]. Available: https://www.forbes.com/sites/emilsayegh/2021/10/01/reviewing-the-mother-of-all-leaks/?sh=6e13d30992c7.. [Accessed 2022].

[5] ANI, 21 May 2021. [Online]. Available: https://ca.finance.yahoo.com/finance/news/cyber-crime-growing-risk-bank-051922015.html. [Accessed 2022].

[6] S. Krishnamoorthy, "Identification of User Behavioural Biometrics for Authentication," University of Winsor, Windsor , 2018.

[7] J. G. Y. Y. T. W. J. C. a. T. A. K. Elliot, "A Comparison of Machine Learning Algorithms in Keystroke Dynamics," in *International Conference on Computational Science and Computational Intelligence (CSCI)*, Las Vegas, NV, USA, 2019.

[8] K. M. S. S. a. M. H.-M. K. P. Baynath, "Machine Learning Algorithm on Keystroke Dynamics Pattern," in *IEEE Conference on Systems, Process and Control (ICSPC)*, Melaka, Malaysia, 2018.

[9] S. V. a. C. Poellabauer, "Multi-Modal Biometric-Based Implicit Authentication of Wearable Device Users," *IEEE Transactions on Information Forensics and Security,* Vols. vol. 14, no. 12 , doi: 10.1109/TIFS.2019.2911170. , pp. pp. 3116-3125, Dec. 2019.

[10] J. S. a. L. G. Y. Yang, "PersonaIA: A Lightweight Implicit Authentication System Based on Customized User Behavior Selection," *IEEE Transactions on Dependable and Secure Computing,* Vols. vol. 16, no. 1, doi:

10.1109/TDSC.2016.2645, pp. pp. 113-126, 1 Jan.-Feb. 2019.

[11] Y. L. Y. C. X. G. a. R. A. M. C. Shen, "Performance Analysis of Multi-Motion Sensor Behavior for Active Smartphone Authentication," *IEEE Transactions on Information Forensics and Security,*, Vols. vol. 13, no. 1, no. doi: 10.1109/, pp. pp. 48-62, Jan. 2018.

[12] X. L. a. D. M. J. Roth, "On Continuous User Authentication via Typing Behavior," *IEEE Transactions on Image Processing,* Vols. vol. 23, no. 10, no. doi: 10.1109/TIP.2014.2348802., pp. pp. 4611-4624, Oct. 2014.

[13] Y. C. X. G. a. R. A. M. C. Shen, "Pattern-Growth Based Mining Mouse-Interaction Behavior for an Active User Authentication System," *IEEE Transactions on Dependable and Secure Computing* , Vols. vol. 17, no. 2, no. doi: 1, pp. pp. 335-349, 1 March-April 2020.

[14] J. D. a. P. Nawrocki, "Continuous authentication on mobile devices using behavioral biometrics," in *2022 22nd IEEE International Symposium on Cluster, Cloud and Internet Computing (CCGrid)*, Taormina, Italy, 2022.

[15] S. J. Q. a. S. S. Bedi, "Secure System of Continuous User Authentication Using Mouse Dynamics," in *2022 3rd International Conference on Intelligent Engineering and Management (ICIEM)*, London, UK, 2022.

[16] "Typing DNA," [Online]. Available: https://www.typingdna.com. [Accessed 2021].

[17] "plurilock.com," Plurilock, [Online]. Available: https://www.plurilock.com. [Accessed 1 2021].

[18] Wikipedia, [Online]. Available: https://en.wikipedia.org/wiki/Transfer_learning. [Accessed 2021].

[19] J. V. Tu, "Advantages and disadvantages of using artificial neural networks versus logistic regression for predicting medical outcomes," *Journal of Clinical Epidemiology,* Vols. vol. 49, no. 11, pp. 1225-1231, 1996. .

[20] Wikipedia, " https://en.wikipedia.org/wiki/Hyperparameter_optimization.," [Online]. [Accessed 2022].

[21] K. G. C. D. N. V. D. T. a. G. U. G. H. M. C. K. B. Herath, "Continuous User Authentication using Keystroke Dynamics for Touch Devices," in *2nd International Conference on Image Processing and Robotics (ICIPRob)*, 2022.

[22] A. a. S. Singh, in *8th International Conference on Cloud Computing, Data Science & Engineering (Confluence)*, Keystroke Dynamics for Continuous Authentication, 2018 .

[23] J. H. D. H. E. Vural, " International Joint Conference on Biometrics, IJCB 2014," in *Stephanie Schuckers,Shared Research Dataset to Support Development of KeystrokeAuthentication*, Clearwater,Florida, USA,, September 29-

October 2 2014.

[24] D. a. A. S. Namin, "On Accuracy of Classification-Based Keystroke Dynamics for Continuous User Authentication," in *2015 International Conference on Cyberworlds (CW)*, 2015.

[25] H. C. a. S. U. Y. Sun, "Shared keystroke dataset for continuous authentication," in *Proc. IEEE Int. Workshop Inf. Forensics Secur. (WIFS)*, Dec. 2016.

[26] G. K. e. al, "LightGBM: A highly efficient gradient boosting decision tree," in *Advances in Neural Information Processing Systems*, 2017.

[27] T. C. a. C. Guestrin, "XGBoost: A scalable tree boosting system," in *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2016.

[28] B. G. R. K. a. Q. W. L. DeSilets, "A neural network model for cell suppression of tabular data," in *IJCNN International Joint Conference on Neural Networks*, [Proceedings 1992].

[29] S. A. a. T. Pfister, "TabNet: Attentive Interpretable Tabular Learning," 2021.

[30] I. A. a. M. D. M. Startsev, "1d cnn with blstm for automated classification of fixations saccades and smooth pursuits," in *Behavior Research Methods*.

[31] G. X. a. R. X. F. Huang, "Research on Ensemble Learning," in *International Conference on Artificial Intelligence and Computational Intelligence*, 2009.

[32] W. S. a. C. P.-i. Rojarath, "Improved ensemble learning for classification techniques based on majority voting," in *7th IEEE International Conference on Software Engineering and Service Science (ICSESS)*, 2016.

[33] D. H. Wolpert, "Stacked generalization," in *Neural Netw., vol. 5, no. 2, pp. 241-259*, 1992.

[34] Wikipedia, [Online]. Available: https://en.wikipedia.org/wiki/Transfer_learning. [Accessed 2021].

[35] NetSec.news, "67 Percent of Breaches Caused by Credential Theft, User Error, and Social Attacks," pp. https://www.netsec.news/67-percent-of-breaches-caused-by-credential-theft-user-error-and-social-attacks/#:~:text=The%20report%20revealed%20that%20the,and%2077%25%20involved%20credential%20theft., 22 May 2020.

Curriculum Vitae

Name:	Sanket Salunke
Post-secondary Education and Degrees:	University of Pune Pune, Maharashtra, India 2011-2015 B.A. The University of Western Ontario London, Ontario, Canada 2019-2020 Meng. The University of Western Ontario London, Ontario, Canada 2021-2022 MESc.
Related Work Experience	Teaching Assistant The University of Western Ontario 2021-2022 Data Scientist Intern F8th Inc 2021-2022 Research Intern Scrawlr Development Inc 2022

Publications:

S. Salunke *et al.*, "Comparison of Machine Learning Techniques for Activities of Daily Living Classification with Electromyographic Data," *2022 International Conference on Rehabilitation Robotics (ICORR)*, 2022, pp. 1-6, doi: 10.1109/ICORR55369.2022.9896565.

S. Salunke, A. Ouda and J. Gagne, "Transfer Learning for Behavioral Biometrics-based Continuous User Authentication," *2022 International Symposium on Networks, Computers and Communications (ISNCC)*, 2022, pp. 1-6, doi: 10.1109/ISNCC55209.2022.9851764.