MARMARA UNIVERSITY
INSTITUTE FOR GRADUATE STUDIES
IN PURE AND APPLIED SCIENCES

TRAINING AND MODELLING WITH PRIVACY IN NETWORK DATA USING MACHINE LEARNING

ONAT CAN BABA

MASTER THESIS

Department of Electrical and Electronics Engineering

Thesis Supervisor
Doç. Dr. Engin MAŞAZADE

ISTANBUL, 2024

MARMARA UNIVERSITY
INSTITUTE FOR GRADUATE STUDIES
IN PURE AND APPLIED SCIENCES

TRAINING AND MODELLING WITH PRIVACY IN NETWORK DATA USING MACHINE LEARNING

ONAT CAN BABA

(525021002)

MASTER THESIS

Department of Electrical and Electronics Engineering

Thesis Supervisor

Doç. Dr. Engin MAŞAZADE

ISTANBUL, 2024

ACKNOWLEDGEMENT

I would like to express my deepest gratitude to my advisor, Associate Professor Engin MAŞAZADE, for their unwavering support, guidance, and encouragement throughout the course of my research and the writing of this thesis. His extensive knowledge and expertise have been invaluable to my work. I am grateful for his patience and willingness to discuss ideas, provide direction, and offer continuous support even during the most challenging phases of my study.

TABLE OF CONTENTS

ACKNOWLEDGEMENT i

TABLE OF CONTENTS ii

ÖZET.....iv

ABSTRACT.....v

SYMBOLS…….…….vi

ABBREVIATIONS vii

LIST OF FIGURES.....viii

LIST OF TABLES.....x

1. INTRODUCTION.....1

2. METHODS.....4

2.1 IoTID20 Dataset.....4

2.2. Data Encoding.....8

2.2.1 Label Encoder …….....8

2.2.2 Target Encoder.....9

2.3 Feature Selection.....10

2.4 Data Normalization.....13

2.5 Gaussian Naïve Bayes Classifier......14

2.5 Logistic Regression Classifier 14

2.5 Decision Tree Classifier.....15

2.6 Differential Privacy.....15

3. RESULTS.....18

3. 1 Selected Features…….....18

3.2 Differential Privacy Effect to the Model Accuracy.....19

3.2.1 Gaussian Naïve Bayes Accuracy…….....20

3.2.2 Logistic Regression Accuracy…….....21

3.2.3 Decision Tree Accuracy…….....22

3.3 Differential Privacy Effect to the Data.....23

3.3.1 Differential Privacy Effect to the Data in Gaussian Naïve Bayes.....23

3.3.1.1 Histograms of Mean and Variances When ϵ is 1.....24

3.3.1.2 Histograms of Mean and Variances When ϵ is 0.1......27

3.3.1.3 Histograms of Mean and Variances When ϵ is 0.01 30

3.3.1.4 Accuracy comparison of GNB and GNB_DP for ϵ values 0.01, 0.1 and 1…….33

3.3.2 Differential Privacy Effect to the Data in Logistic Regression……...33

3.3.2.1 Histograms of Coefficients When ϵ is 1.....34

3.3.2.2 Histograms of Coefficients When ϵ is 0.1…….....35

3.3.2.3 Histograms of Coefficients When ϵ is 0.01.....36

3.3.2.4 Accuracy Comparison of LR and LR_DP for ϵ values 0.01, 0.1 and 1.....36

3.3.2.5 Prediction Comparison of LR and LR_DP for ϵ values 0.01 and 1.....37

4. DISCUSSION.....40

5. CONCLUSIONS.....43

REFERENCES 44

CURRICULUM VITAE 46

ÖZET

TRAINING AND MODELLING WITH PRIVACY IN NETWORK DATA USING MACHINE LEARNING

Günümüzün veri odaklı ortamında, özellikle de Nesnelerin İnterneti cihazlarının yaygın olarak benimsenmesiyle birlikte gizlilik kaygıları çok önemli hale gelmiştir. Bu çalışmada, bu endişeleri gidermek için makine öğrenimi algoritmalarının, özellikle de Gaussian Naive Bayes, Logistic Regression ve Decision Tree Diferansiyel Gizlilik gibi gizliliği koruyan tekniklerle entegrasyonu araştırılmıştır. Bu yenilikçi yaklaşımın ilkeleri, yöntemleri ve sonuçları derinlemesine incelenmiş olup; Diferansiyel Gizlilik kullanılmayan makine öğrenmesi algoritması ile doğruluk karşılaştırması yapılmıştır. 83 ağ özelliği, 3 etiket sütunu ve 9 farklı saldırı türü ile nesnelerin interneti ağ trafiğini temsil eden bir veri kümesi kullanılarak yapılan bu çalışma Diferansiyel Gizliliğe sahip çeşitli makine öğrenmesi algoritmalarının, bireysel gizliliği korurken verileri nasıl etkili bir şekilde sınıflandırabildiğini göstermiştir. Gizlilik korumasının, sınıflandırma doğruluğunda karşılık gelen bir düşüşe rağmen, daha düşük ϵ değerlerinde bile doğruluğu gözlemlenmiştir. Bu, veri hassasiyetinin çok önemli olduğu IoT uygulamalarında Diferansiyel Gizliliğin güçlü gizlilik koruması sunma konusundaki potansiyeline işaret etmektedir.

EYLÜL, 2024 **ONAT CAN BABA**

ABSTRACT

TRAINING AND MODELLING WITH PRIVACY IN NETWORK DATA USING MACHINE LEARNING

Privacy concerns have become paramount in today's data-driven landscape, particularly with the widespread adoption of Internet of Things (IoT) devices. This work explores the integration of machine learning algorithms, specifically Gaussian Naive Bayes, Logistic Regression and Decision Tree, with privacy-preserving techniques like Differential Privacy to address these concerns. We delve into this innovative approach's principles, methods, and implications. Using a dataset representative of IoT network traffic, with 83 network features and 3 label column and 9 various attack types. We demonstrate how three various machine learning algorithms with Differential Privacy can effectively classify data while safeguarding individual privacy, highlighting its significance in pervasive connectivity and data-driven decision-making. Privacy protection is robust even at lower ϵ values, albeit with a corresponding decrease in classification accuracy. This underscores the potential of differential privacy to offer strong privacy protection in IoT applications where data sensitivity is a paramount concern.

SEPTEMBER, 2024 **ONAT CAN BABA**

SYMBOLS

$\boldsymbol{i}$: Category representation

$\boldsymbol{j}$: Class representation

$\boldsymbol{Y}$: Target representation

$\boldsymbol{S_{ij}}$: The encoding for category i and class j

$\boldsymbol{n_{iy_j}}$: The number of observations with target Y equal to j for category i

$\boldsymbol{n_{Y_j}}$: The number of observations with target Y equal to j.

$\boldsymbol{\lambda_i}$: Shrinkage factor

$\boldsymbol{x}$: Data Point

$\boldsymbol{\mu_A}$: Mean of the distribution A

$\boldsymbol{\sigma_A}$: Standard deviation of distribution A

$\boldsymbol{\mu_B}$: Mean of the distribution B

$\boldsymbol{\sigma_B}$: Standard deviation of distribution B

$\boldsymbol{\epsilon}$: Privacy parameter

$\boldsymbol{\sigma(z)}$: Sigmoid function of z

$\boldsymbol{z}$: Linear combination of inputs and weights

$\boldsymbol{w}$: Weights learned during training

$\boldsymbol{b}$: Bias term

ABBREVATION

IoT **:** Internet of Things

RFID **:** Radio-frequency identification

IP **:** Internet Protocol

DoS **:** Denial of Service

TCP **:** Transmission Control Protocol

UDP **:** User Datagram Protocol

HTTP **:** Hyper Text Transfer Protocol

MITM **:** Man in the Middle

ARP **:** Address Resolution Protocol

GNB **:** Gaussian Navie Bayes without Differential Privacy

GNB_DP **:** Gaussian Navie Bayes with Differential Privacy

LR **:** Logistic Regression without Differential Privacy

LR_DP **:** Logistic Regression with Differential Privacy

DCT **:** Decision Tree without Differential Privacy

DCT_DP **:** Decision Tree with Differential Privacy

LIST OF FIGURES

Figure 2.1. Cumulative Explained Variance vs Number of Principle Components…..11
Figure 2.2. Test accuracy vs. Number of features selected……...……....12
Figure 2.3. Sequential Feature Selection Performance……..13
Figure 2.4. Example of a general decision tree……..16
Figure 3.1. Cross-validation accuracy of the sequential feature selection while the number of the features are incrementing……...19
Figure 3.2. Average accuracy of the Gaussian Naïve Bayes and Gaussian Naïve Bayes with Differential Privacy Comparison for various ϵ values…….20
Figure 3.3. Average accuracy of the Logistic Regression and Logistic Regression with Differential Privacy Comparison for various ϵ values……21
Figure 3.4. Average accuracy of the Decision Tree and Decision Tree with Differential Privacy Comparison for various ϵ values……23
Figure 3.5. Mean and Variance histograms of Src_Port feature for each classes in ϵ 1. ……..24
Figure 3.6. Mean and Variance histograms of Dst_Port feature for each classes in ϵ 1…….... 24
Figure 3.7. Mean and Variance histograms of Flow_IAT_Max feature for each classes in ϵ 1……25
Figure 3.8. Mean and Variance histograms of Pkt_Len_Var feature for each classes in ϵ 1……25
Figure 3.9. Mean and Variance histograms of Down/Up_Ratio feature for each classes in ϵ 1……26
Figure 3.10. Mean and Variance histograms of Timestamp_Encoded feature for each classes in ϵ 1……...…….26
Figure 3.11. Mean and Variance histograms of Src_Port feature for each classes in ϵ 0.1…….27
Figure 3.12. Mean and Variance histograms of Dst_Port feature for each classes in ϵ 0.1…….27
Figure 3.13. Mean and Variance histograms of Flow_IAT_Max feature for each classes in ϵ 0.1…….28
Figure 3.14. Mean and Variance histograms of Pkt_Len_Var feature for each classes in ϵ 0.1…….28
Figure 3.15. Mean and Variance histograms of Down/Up_Ratio feature for each classes in ϵ 0.1…….29
Figure 3.16. Mean and Variance histograms of Timestamp_Encoded feature for each classes in ϵ 0.1…….29
Figure 3.17. Mean and Variance histograms of Src_Port feature for each classes in ϵ 0.01……...30
Figure 3.18. Mean and Variance histograms of Dst_Port feature for each classes in ϵ 0.01……...30

Figure 3.19. Mean and Variance histograms of Flow_IAT_Max feature for each classes in ϵ 0.01……...31
Figure 3.20. Mean and Variance histograms of Pkt_Len_Var feature for each classes in ϵ 0.01……...31
Figure 3.21. Mean and Variance histograms of Down/Up_Ratio feature for each classes in ϵ 0.01……...32
Figure 3.22. Mean and Variance histograms of Timestamp_Encoded feature for each classes in ϵ 0.01……...32
Figure 3.23. Accuracy comparison of the Gaussian Naïve Bayes and Gaussian Naïve Bayes with Differential Privacy for ϵ values 0.01, 0.1 and 1……...33
Figure 3.24. Coefficient histograms of each feature in ϵ 1……34
Figure 3.25. Coefficient histograms of each feature in ϵ 0.1……35
Figure 3.26. Coefficient histograms of each feature in ϵ 0.01……...36
Figure 3.27. Accuracy comparison of the Logistic Regression and Logistic Regression with Differential Privacy for ϵ values 0.01, 0.1 and 1……..37
Figure 3.28. Prediction comparison of the Logistic Regression and Logistic Regression with Differential Privacy for ϵ value 0.01……38
Figure 3.29. Prediction comparison of the Logistic Regression and Logistic Regression with Differential Privacy for ϵ value 1……39

LIST OF TABLES

Table 2.1. List of feature categories…….…….....5

Table 2.2. List of classes in the dataset with explanations.……..……7

Table 2.3. Example of label encoded classes.....8

Table 2.4. Example of target encoded features……..……..…..…....10

Table 3.1. List of the selected features and explanations……..…….……....18

1. INTRODUCTION

In today's data-driven landscape, privacy concerns loom large, especially with the widespread adoption of Internet of Things (IoT) devices. While enhancing convenience, these devices also raise questions about the security of personal data. In response, integrating machine learning algorithms like Gaussian Naive Bayes with privacy-preserving techniques such as Differential Privacy has gained significant attention. This intersection holds promise for addressing privacy challenges without compromising the utility of data-driven applications. By ensuring the confidentially of sensitive information while extracting valuable insights, this approach offers a pathway to address privacy concerns without sacrificing the utility of data-driven applications.

Furthermore, the potential applications of privacy-preserving machine learning span various domains, from healthcare data analysis to secure financial transactions and personalized recommendations. In this article, we explore the principles, methods, and implications of this innovative approach, shedding light on the potential of data-driven decision-making while prioritizing individual privacy.

Kevon Ashton described the Internet of Things term for the first time as connecting the physical world to the Internet by sensors in 1999. IoT starts with Radio-frequency identification (RFID) tags in supply chains to count and track goods without employees. Although the IoT concept is relatively new, machine-to-machine communication has been used since the 1990s for monitoring and operations for industry-specific reasons. Therefore, the idea of IoT has been a popular topic for many decades due to its benefits, such as low-cost connectivity, widespread adoption of Internet Protocol (IP) networks, miniaturization, advances in data analytics, and cloud computing for easy management [1].

With the expansion of IoT networks, traffic classification has become a challenge in the networking world. Shahid et al. [2] created an experimental smart home network with IoT devices and selected features from their activities. Training and testing with various machine learning algorithms, they achieved 99.9% accuracy in analysing network traffic.

Sivanathan et al. [3] addresses the challenge of managing IoT devices in smart environments. The authors collected smart-campus data and characterized the traffic based on activity patterns. Using the random forest algorithm, they achieved over %95 accuracy in identifying IoT devices.

While IoT has begun to take an essential place in our lives, major privacy issues have emerged. Since all the IoT devices gather human life data, an intruder can easily reach them in a server where the data is stored. According to Weber et al. [4] privacy is not only the protection of personal information but also the ability to control what happens with that information. Concealment of personal information is a fundamental human right that people may not be aware of. An individual may not know how the information created by a device can be followed and reveal daily routines and activities.

Chanal et al. [5] mention that an individual endpoint or a device wouldn't cause any privacy issues. However, due to the interoperability of things, it can make inferences about personal data when the separate information from several end devices is assembled, categorized, and analysed. It is a person's right to control the information to share with others. It is hard to control privacy when a person is included in an IoT scheme. As a result, the owners of IoT devices need to deal with who is using their information.

Altam et al. [6] find profiling as a threat to data privacy. Profiling is a method used to classify an individual's activities and actions based on the extracted features from collected and processed data over a long time. If the information is collected without willingness from the user and assembled with other personal information, it can create a more complete profile.

Shi et al. [7] explain the challenge of privacy leakage with an example. An adversary can easily speculate whether the house is vacant by reading the electricity or water usage. Similarly, with the network traffic information, one can make inferences about the house and the time people leave there.

Dwork [8] described a privacy standard in order to address privacy issues: differential privacy. It quantifies the risk to one's privacy from participating in a database. The idea is to ensure that the presence or absence of a single individual's data

does not significantly affect the outcome of any analysis, thereby providing deniability to participants.

McSherry et al. [9] investigate how privacy-preserving algorithms can be used in mechanism design, specifically those ensuring differential privacy. They demonstrate that mechanisms with differential privacy can limit participants' influence on outcomes. This results in mechanisms that are approximate dominant strategies, resilient to coalitions, and easily repeatable.

Our work represents a significant enhancement in addressing privacy concerns within the IoT landscape. By integrating machine learning algorithms with differential privacy, our work is a novel approach to ensuring the confidentiality of sensitive IoT data while extracting valuable insights. Implementing Differential Privacy mechanisms while training machine learning models on IoT data contributes to the burgeoning field of privacy-preserving machine learning. It offers practical solutions to the pressing privacy challenges inherent in IoT environments. This methodology not only enhances the security of personal data gathered by IoT devices but also underscores the ethical imperative of prioritizing individual privacy rights in the era of pervasive connectivity.

To the best of our knowledge, this work is among the first to successfully combine differential privacy techniques with machine learning models specifically tailored for IoT data. While previous studies have explored differential privacy in various domains, the unique challenges posed by the vast, heterogeneous, and sensitive nature of IoT data have not been comprehensively addressed until now.

2. METHODS

2.1. IoTID20 Dataset

IoT systems recently started to play a more significant role in our daily lives. IoT is the concept where physical devices are connected to the Internet to help with daily activities such as setting the A/C, running the oven, and presenting security cameras. For instance, when your wristwatch collects your exercise data or when it knows at what time you usually sleep at night, it can tell the exact time you are in deep sleep. Or when you return to your home after work, your mobile phone is automatically connecting to your Wi-Fi, another data collection starts. When a technology accompanies our lives this close, it may become a risk due to the data created by the nature of the IoT devices. This data may contain the private habits of the home members, such as the time that no one is at home, the time that security cameras start to work, or the network traffic where the IoT devices authenticate with username and password. Thus, IoT devices have a significant risk of being subject to cyber-attacks, which may cause the shutting down of the smart systems.

IoTID20 [10] dataset contains a combination of IoT devices that are part of a smart home and their networks in its testbed. While a set of IoT devices are the victims of a cyber-attack, another set of IoT devices are the attacking devices. The researchers aim to address the lack of publicly available and representative datasets that can be used for training and evaluating machine learning models for IoT security. This dataset includes both benign and malicious activities to reflect real-world scenarios. It includes smart lights, smart locks, and temperature sensors, all connected through a common gateway. The diversity in devices and their interactions is crucial for capturing a wide range of normal and abnormal behaviours.

The dataset contains 83 network features to be categorized in 12 titles as shown in the Table 2.1. When sorting the features in the dataset, some fall under "Basic Information" like Flow_ID, Src_Port, Dst_IP, or Timestamp. Some relate to "Packet Lengths" such as Pkt_Len_Min, Pkt_Len_Max, and Pkt_Len_Mean. Additionally, the dataset encompasses other features in terms of "Packet Rate," "Inter-Arrival Times," "Flags," "Flow Size and Rates," and "Activity Time" as well.

Table 2.1. List of feature categories.

Category	Feature
Basic Information	Flow_ID Src_IP Src_Port Dst_IP Dst_Port Protocol Timestamp
Flow Duration and Packet Counts	Flow Duration Total Forward Packets Total Backward Packets
Packet Lengths	TotalLenght Forward Packets TotalLenght Backward Packets Forward Packet Lenght Max Forward Packet Lenght Min Forward Packet Lenght Mean Forward Packet Lenght Std Backward Packet Lenght Max Backward Packet Lenght Min Backward Packet Lenght Mean Backward Packet Lenght Std Packet Lenght Min Packet Lenght Max Packet Lenght Mean Packet Lenght Std Packet Lenght Var Forward Seg Size Avg Backward Seg Size Avg Packet Size Avg
Packet Rate	Flow Byts/s Flow Packets/s Forward Packets/s Backward Packets/s
Inter-Arrival Times	Flow InterArrivalTime Mean Flow InterArrivalTime Std Flow InterArrivalTime Max Flow InterArrivalTime Min Forward InterArrivalTime Tot Forward InterArrivalTime Mean Forward InterArrivalTime Std Forward InterArrivalTime Max Forward InterArrivalTime Min Backward InterArrivalTime Tot Backward InterArrivalTime Mean

	Backward InterArrivalTime Std Backward InterArrivalTime Max Backward InterArrivalTime Min
Flags	Forward PSH Flags Backward PSH Flags Forward URG Flags Backward URG Flags FIN Flag Count SYN Flag Count RST Flag Count PSH Flag Count ACK Flag Count URG Flag Count CWE Flag Count ECE Flag Count
Flow Direction	Down/Up_Ratio
Connection Characteristics	Init_Fwd_Win_Byts Init_Bwd_Win_Byts
Segment Characteristics	Fwd_Seg_Size_Min
Flow Size and Rates	Fwd_Byts/b_Avg Fwd_Pkts/b_Avg Fwd_Blk_Rate_Avg Bwd_Byts/b_Avg Bwd_Pkts/b_Avg Bwd_Blk_Rate_Avg
Subflow Statistics	Subflow_Fwd_Pkts Subflow_Fwd_Byts Subflow_Bwd_Pkts Subflow_Bwd_Byts
Activity Time	Active_Mean Active_Std Active_Max Active_Min Idle_Mean Idle_Std Idle_Max Idle_Min

There are 3 label columns with several informative features for machine learning algorithms. The Sub_Cat column has 9 classes as explained in Table 2.2. Since the Sub_Cat column has more class variation comparing the other two label columns, we utilized Sub_Cat labels in our work.

The dataset contains multiple attack types, such as denial of service, man-in-the-middle, scanning, and injection attacks.

Table 2.2. List of classes in the dataset with explanations.

Attack Type	Class	Explanation
Normal	Normal	Expected network activity that doesn't contain any malicious behaviour.
DoS	DoS-Synflooding	Exploiting the TCP (Transmission Control Protocol) handshake process by sending a large number of SYN requests.
Mirai	Mirai-Hostbruteforce	The Mirai botnet attempts to gain unauthorized access to devices by trying various username/password combinations.
	Mirai-HTTP Flooding	The Mirai malware flooding to the victim's network with HTTP requests, aiming to exhaust its resources and disrupt its services.
	Mirai-UDP Flooding	The Mirai malware flooding to generate a large volume of User Datagram Protocol (UDP) packets targeting a victim's network, overwhelming its resources.
	Mirai-Ackflooding	The Mirai botnet flooding to the target network with TCP ACK packets, consuming its resources.
MITM	MITM ARP Spoofing	Man-in-the-Middle (MITM) Attack using Address Resolution Protocol (ARP) Spoofing, intercepting communication between two parties by impersonating one of the parties.
Scan	Scan Hostport	Scanning for open ports specifically on the target hosts.
	Scan Port OS	Scanning for open ports on target systems and attempting to identify the

		operating system running on those systems.

2.2. Data Encoding

With the categorical features in the dataset, it is hard to consider these features for training a machine learning algorithm. Consequently, encoding becomes a crucial step in data processing in our work. By converting categorical variables into a numerical format, we enable the machine learning model to interpret and learn from the data efficiently. The choice of encoding method can significantly impact the performance and accuracy of the model, making it essential to carefully analyse the characteristics of the categorical data before deciding on the appropriate encoding strategy. There are two encoding mechanisms used in this work: label encoder and target encoder.

2.2.1. Label Encoder

Label Encoder mechanism of sklearn in the preprocessing library is a competent algorithm for converting labels into numerical values [11]. The columns "Sub_Cat", "Cat" and "Label" are encoded by using this technique and turned into the numerical labels starting from 0 to the number of the unique labels minus 1, respectively.

While “Label” column has only two kinds of labels as Anomaly and Normal; “Cat” column has five kinds of labels as Normal, DoS, Mirai, MITM and Scan as shown in the Table 2.3. Since the class variation of Sub_Cat better represents our goal, we are only interested in the "Sub_Cat" labels. The columns "Cat" and "Label" are dropped from our dataset.

Table 2.3. Example of label encoded classes.

Label	**Cat**	**Sub_Cat**	**Encoded Sub_Cat**
Anomaly	Mirai	Mirai-Ackflooding	2
Anomaly	DoS	DoS-Synflooding	0
Anomaly	Scan	Scan Port OS	8

Anomaly	Mirai	Mirai-Hostbruteforce	4
Anomaly	Mirai	Mirai-Hostbruteforce	4

2.2.2. Target Encoder

As well as the label columns, the dataset consists of four feature columns that contain categorical values which are Flow_ID that has a unique id for each flow; Src_IP that shows where the flow started; Dst_IP that shows where the IP packet is destined for a flow and lastly Timestamp that has the time information of the flows. However, the LabelEncoder technique is unsuitable for these columns due to the nature of the machine-learning algorithms. When the LabelEncoder is used for a categorical feature, the training of the machine learning algorithm may yield poor accuracy because of the incorrect correlations of the encoded categorical features. In order to tackle this problem, we used target encoding. Target encoding is a technique that groups the categories in a column and calculates the mean target value for each category. Then, it maps the computed mean target values to the corresponding categories in the specified column [12].

In order to understand the Target Encoder working mechanism, process is explained below:

1. Calculate the mean for each class.
2. Apply Smoothing if necessary.
3. Replace categories with encoded values.

To do so, multiclass target encoding formula where S_{ij} is is the encoding for category i and class j; n_{iy_j} is the number of observations with target Y equal to j for category i; n_{Y_j} is the number of observations with target Y equal to j:

$$S_{ij} = \lambda_i \frac{n_{iy_j}}{n_i} + (1 - \lambda_i) \frac{n_{Y_j}}{n} \quad (2.1)$$

λ_i is shrinkage factor as:

$$\lambda_i = \frac{n_i}{m+n_i} \quad (2.2)$$

Here, m is the smoothing parameter.

Table 2.4. Example of target encoded features.

Flow_ID_Encoded	Src_IP_Encoded	Dst_IP_Encoded	Timestamp_Encoded
3.33	4.778975	4.742246	3.70
0.00	0	3.588726	0.00
8.00	5.297811	3.588726	8.00
4.00	5.297811	3.588726	4.00
4.00	4.734851	3.730638	4.00

Table 2.4 shows the encoded values for each categorical feature in our dataset, calculated by the formula 2.1.

2.3. Feature Selection

The presence of 83 features in the dataset brings computational challenges for machine learning algorithms. On one hand, the high dimensionality of the dataset increases the complexity of model training, requiring more computational resources and time. Algorithms such as Gaussian Naive Bayes may struggle to handle many features and potentially inaccurate performance. On the other hand, the presence of irrelevant or redundant features can further these challenges, as the model may waste computational resources trying to learn from noisy or irrelevant data.

The issues outlined previously are the reasons for employing a feature selection mechanism. In this work we tried several feature selection algorithms. First, we reduced the training data using Principal Component Analysis (PCA). PCA is a technique that analyses data to extract the most important information. It reduces the size of dataset by keeping only the important information [13]. In our work, the number of principal components needed to explain 95% of the variance was 8 as shown in the Figure 2.1. However, it is decided not to continue with PCA since it is usually utilized for dimension reduction.

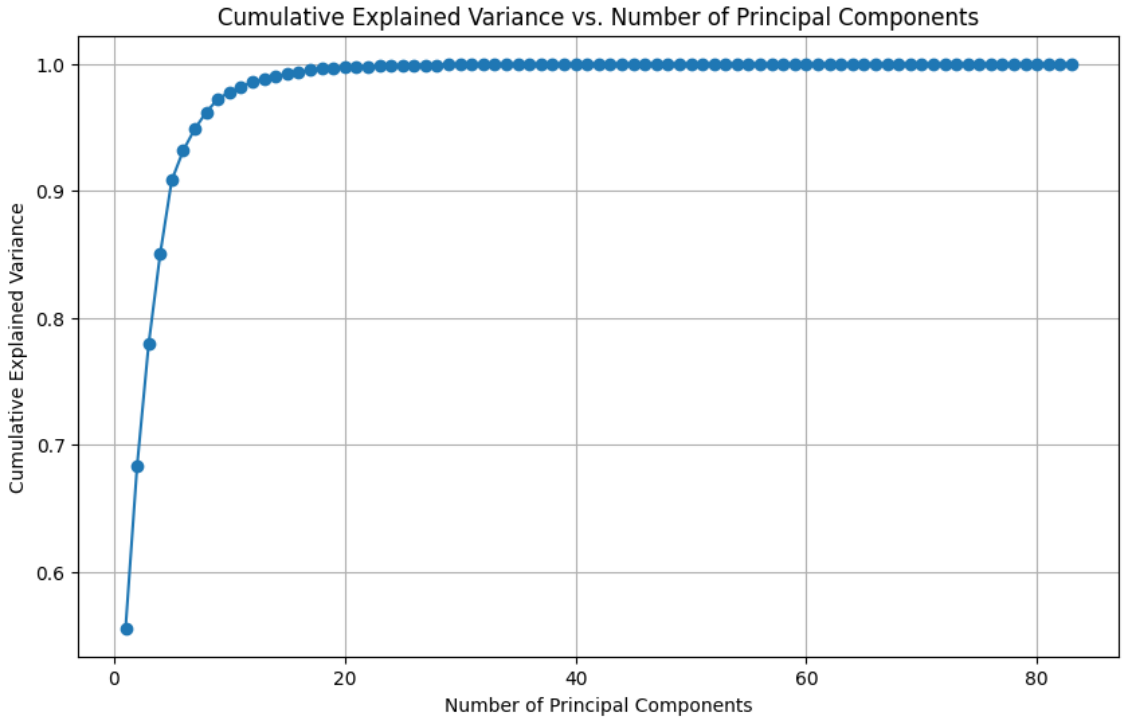


Figure 2.1. Cumulative Explained Variance vs Number of Principle Components

Second, we employed Best Individual Feature algorithm which is a method that selects the best features based on univariate statistical tests. SelectKBest removes all the features but the k highest scoring features. This method estimates the degree of linear dependency between two random variables [14]. The algorithm calculated the best test accuracy as 0.8844 and best k = 3 which gives the number of the most informative features as shown in the Figure 2.2. However, it is decided not to continue with SelectKBest since there are more powerful feature selection algorithms.

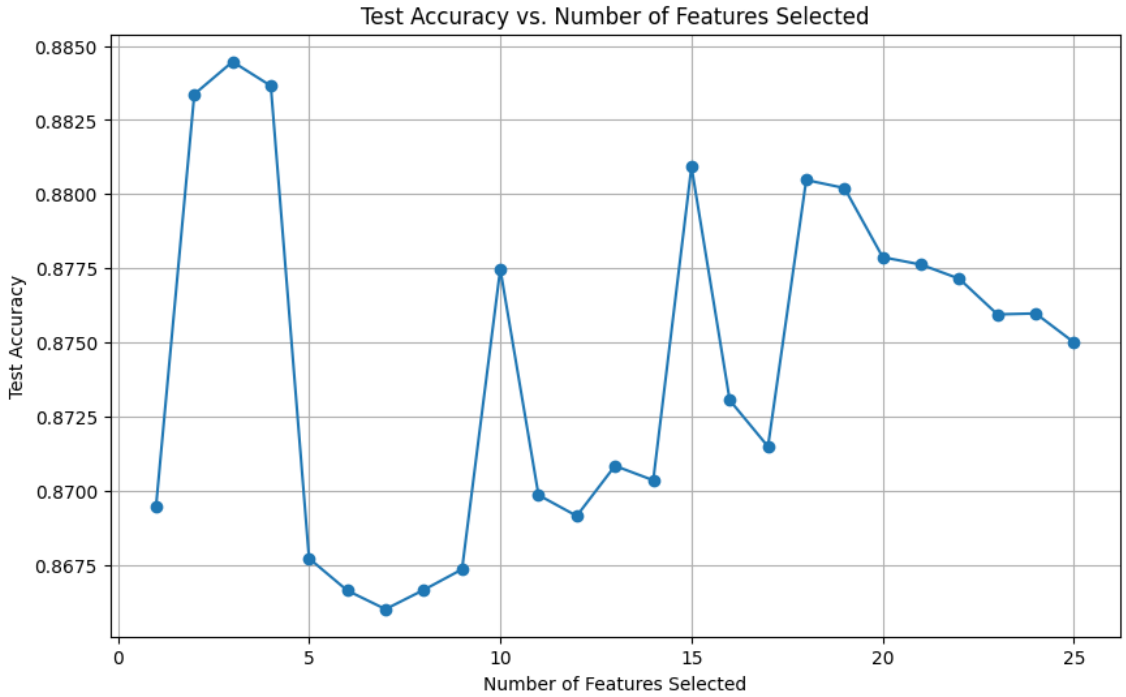


Figure 2.2. Test accuracy vs. Number of features selected.

Lastly, we employed Sequential Feature Selection in our research. Sequential feature selection is a technique that selects the features successively. The method starts with the most informative feature and, by sequentially adding the features that contribute most to the classification performance, reaches a set of the best features [15].

We used the Gaussian Naive Bayes classifier in our sequential forward feature selection study. We took the accuracy result as the metric for selecting features based on cross-validated accuracy with 5 folds. The initial performance of Sequential feature selector is shown in Figure 2.3.

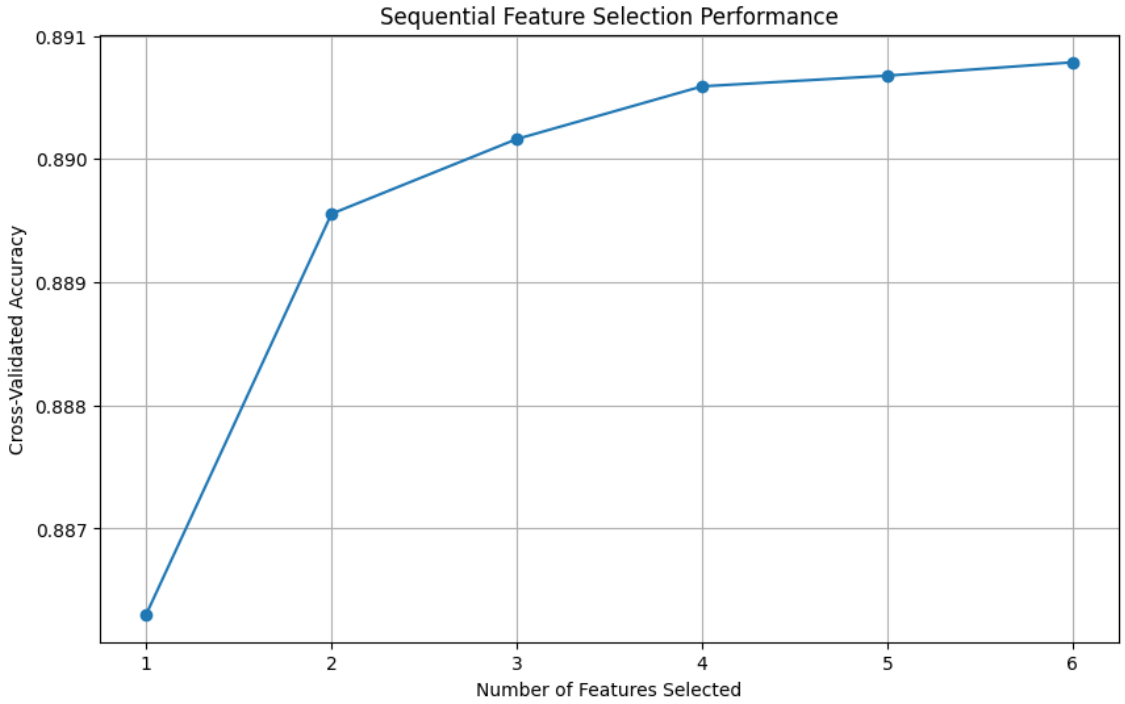


Figure 2.3. Sequential Feature Selection Performance

2.4. Data Normalization

Data normalization is a data preprocessing method for numerical features to be on a similar scale. MixMaxScaler of sklearn transforms features by scaling each feature to a given range. The range of [0-1] was initially tried in our work where the largest occurring data point corresponds to the maximum value and the smallest one corresponds to the minimum value [11].

On the other hand, StandardScaler normalization is a common technique used in machine learning to standardize features by removing the mean and scaling to unit variance [11]. It is useful when features have different scales like in our work. StandardScaler helps to ensure that all features are treated equally. Standardized data can help the optimization algorithm converge faster, as it eliminates the bias caused by features with varying magnitudes. In this technique, each feature is centered by subtracting its mean. This ensures that the mean of the transformed data is zero. After centering, each feature is scaled by dividing by its standard deviation. This makes the variance of each feature equal to one [16].

2.5. Gaussian Naïve Bayes Classifier

Machine learning algorithms are widely used for classification problems such as spam email detection, sentiment analysis, medical diagnosis, and more [17]. These algorithms make classifications based on the relationships of the values in the data. Therefore, while their performance depends on the quality of the provided features, selecting the correct machine-learning algorithm is also crucial for classification problems.

Gaussian Naive Bayes is a commonly used machine learning algorithm. Gaussian Naive Bayes is an algorithm that utilizes mean and variance values of the features in the dataset for each class to classify the target values. Gaussian Naive Bayes assumes that the features are conditionally independent given the class, making it particularly efficient for datasets with many features [18]. The Gaussian Naive Bayes makes predictions starting with the prior probabilities, then calculates the likelihoods for each feature distribution to compute the final scores for each class. Calculates p(x|A) the probability of observing x, if x came from the class A distribution and p(x|B) the probability of observing x, if x came from the class B distribution. Then it calculates z-score distance x from class A and z-score distance of x from class by the formulas below:

$$(x - \mu_A)/\sigma_A \tag{2.3}$$

$$(x - \mu_B)/\sigma_B \tag{2.4}$$

2.6. Logistic Regression Classifier

Logistic Regression is another popular algorithm used in machine learning. It applies sigmoid function to model probabilities [19]. Linear regression starts by creating a linear combination of input features:

$$z = w_1x_1 + w_2x_2 + \cdots + w_nx_n + b \tag{2.5}$$

- $x_1, x_2, \ldots, x_n$ are the input features.
- $w_1, w_2, \ldots, w_n$ are the weights learned during training.
- b is the bias term.

Logistic regression then applies the sigmoid function to convert the linear output z into a probability:

$$\sigma(z) = \frac{1}{1+e^{-z}} \quad (2.6)$$

The sigmoid function squashes the linear value z into a range between 0 and 1, making it interpretable as a probability.

- If z is a large positive number, $\sigma(z)$ approaches 1.
- If z is a large negative number, $\sigma(z)$ approaches 0.

Once the probability is computed, logistic regression applies a threshold to make a classification decision.

- If $\sigma(z) \geq 0.5$, the output is classified as 1 which is positive class.
- If $\sigma(z) < 0.5$, the output is classified as 0 which is negative class.

Since our dataset contains more than 2 classes, we applied a multiclass classification method called one-vs-rest. In this strategy, the multiclass classification problem is broken down into multiple binary classification problems, where each class is treated as a "positive class" against "rest" the classes [20]. For a dataset with k classes, the algorithm trains $k$separate logistic regression classifiers. Each classifier is responsible for distinguishing whether a sample belongs to a specific class or not. During prediction, all k classifiers output a probability for their respective class. The class with the highest predicted probability is chosen as the final prediction. In scikit-learn, the LogisticRegression model automatically supports multiclass classification and can be used with One-vs-Rest [11].

2.7. Decision Tree Classifier

Decision Tree Classifiers break down complex decision-making processes into simpler, interpretable decisions by recursively partitioning the input space based on feature values [21]. This structure allows decision tree classifiers to handle both categorical and numerical data. The primary strength of DTCs lies in their ability to efficiently handle high-dimensional datasets by selecting relevant features at each decision node. However, several challenges accompany DTC design, such as the risk of overfitting, especially when tree structures become too deep. This overfitting often

results in poor generalization to unseen data. Additionally, the hierarchical nature of DTCs can lead to error propagation, where misclassifications in higher levels impact decisions further down the tree. Decision Tree Classifiers are popular in areas like radar signal classification, medical diagnosis, and speech recognition. Their major advantage lies in simplifying complex decision-making processes into interpretable steps [22].

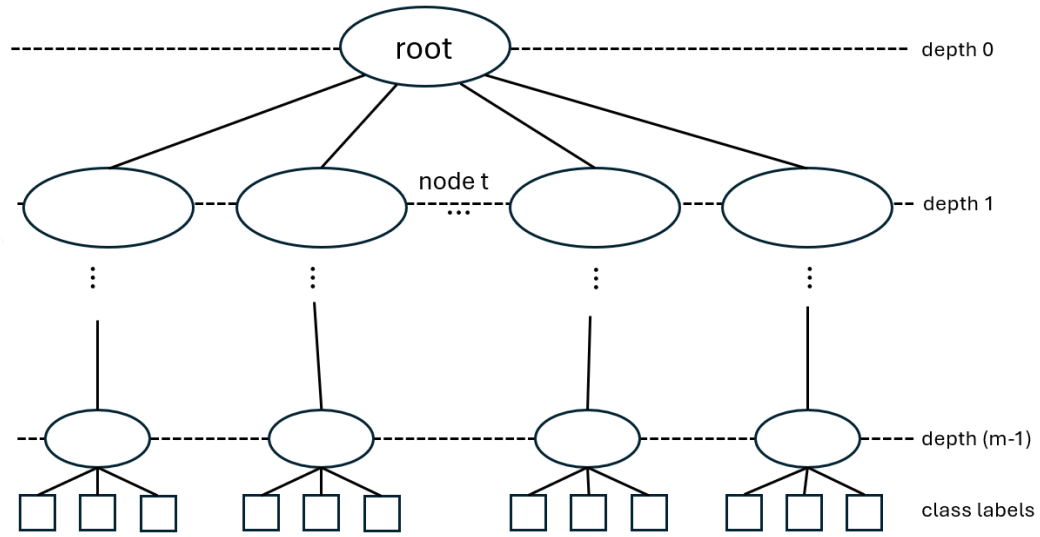


Figure 2.4. Example of a general decision tree

2.8. Differential Privacy

Dalenius, in 1977, asserted that statistical databases should avoid any personal information being reachable from the database [23]. However, it has been understood that reaching this security grade is impossible, even for individuals not in the database [8]. This led to differential privacy, a concept that can quantify the increased privacy risk.

A privacy-preserving statistical database aims to reveal population characteristics while preserving individual privacy within the sample. This leads to the concept of differential privacy, which ensures that participating in a statistical database does not significantly increase privacy risks.

An adversary with auxiliary information in the statistical database can cause a privacy breach for an individual not even in the database. For example, assume that the database results in the average weights of men from different cities. When an adversary has access to the statistical database and auxiliary information such as "Luna Bosh is 5

kilograms lighter than the average man who lives in İstanbul." learns Luna Bosh's exact weight. To address this problem, Dwork [8] shifted from absolute privacy guarantees to relative ones: any result is equally likely whether or not the individual is part of the database, within a small margin. While a disclosure might still occur, our assurance ensures it won't be caused by the presence of the individual's data, and the individual's actions or inaction couldn't prevent it.

The mechanism works by adding random noise to the answer yielded by the database. In order to describe the privacy mechanism in differential privacy, ϵ must be defined first. Privacy parameter or ϵ determines "how much" of an individual's privacy that an analysis may utilize. A smaller ϵ value means better protection, while a larger value means worse privacy protection [24].

IBM's Differential Privacy Library, also known as diffprivlib, provides a set of mechanisms that helps to add noise to protect the privacy of the data. Like well-known machine learning libraries such as Scikit-Learn, diffprivlib models are also easy to train and fit [25].

3. RESULT

3.1. Selected Features

The utilization of a sequential feature selector is one of the crucial implementation decisions in our work. While the Gaussian Naive Bayes is used to obtain the optimal performance of the feature selector mechanism, cross-validation is also necessary to avoid overfitting and obtain more reliable accuracy scores. As the result of the sequential feature selection, 6 of them are selected out of 83 features shown in the Table 3.1.

Table 3.1. List of the selected features and explanations.

Selected Feature	Description
Src_Port	The source port number used to identify different types of traffic on a network for specific services and applications.
Dst_Port	Similar to Src_Port, it represents the destination port number that indicates the port to which the traffic is being sent.
Flow_IAT_Max	Represents the maximum time between the arrival of two consecutive packets within a flow.
Pkt_Len_Var	Represents the packet length (size of the data contained in each packet) variation within a flow that measures the variance or variability in packet lengths within a flow, which can indicate changes in the type or volume of data being transmitted.
Down/Up_Ratio	Represent the ratio of downstream traffic to upstream traffic. It indicates whether the traffic is flowing in one direction or

	balanced.
Timestamp_Encoded	Represents the categorical timestamp of when the observation was made or when the traffic occurred. It analyses network traffic over time, identifies patterns, and detects anomalies. This feature is encoded by the target encoder in our working dataset.

Fig. 3.1 shows the cross-validation accuracy of the sequential feature selection while the number of the features are incrementing. In this selection, we used Gaussian Naïve Bayes accuracy score, where the accuracy has a very short range between 0.886 and 0.891.

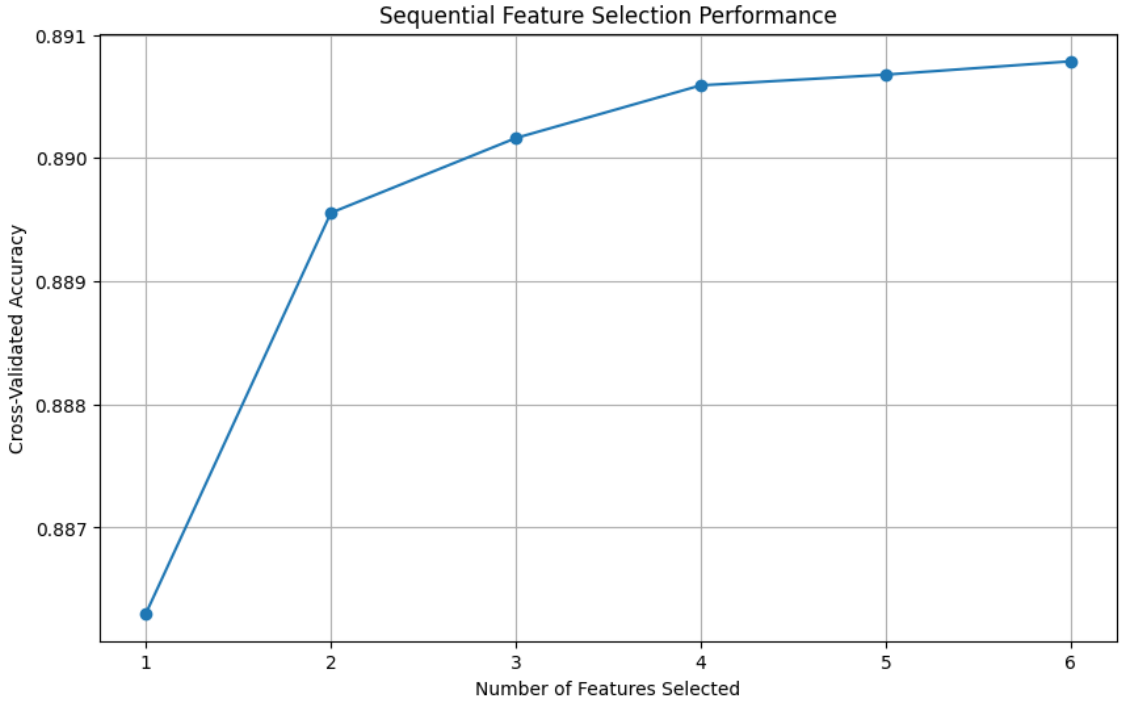


Figure 3.1. Cross-validation accuracy of the sequential feature selection while the number of the features are incrementing.

3.2. Differential Privacy Effect to the Model Accuracy

3.2.1. Gaussian Naïve Bayes Accuracy

Observing the differential privacy effects is slightly tricky regarding data protection. Thus, our work ran Gaussian Naive Bayes algorithm several times with various ϵ values to compare the accuracy. We picked 10 different ϵ values between -2 and 2, and we ran both Gaussian Naive Bayes without differential privacy (GNB) and Gaussian Naive Bayes with differential privacy (GNB_DP) 15 times to take the average values to avoid undesired incorrect results. As shown in Figure 3.2, the average accuracy of the GNB is 0.88, while the average accuracy of the GNB_DP initiates at 0.48 with the least ϵ value in our work -2. In the same graph, it can be seen that as the ϵ value increases, the GNB_DP accuracy also increases. Finally, when the ϵ value is set to 2, the GNB and the GNB_DP accuracy values are met. The figure clearly shows that when the ϵ value is small, the data is heavily protected however consequently classification accuracy is poor. While the ϵ value is increasing which indicates less privacy; classification accuracy is resulting better. At the end of the graph when we set ϵ to 2, two classification algorithms are met in the accuracy of 0.88.

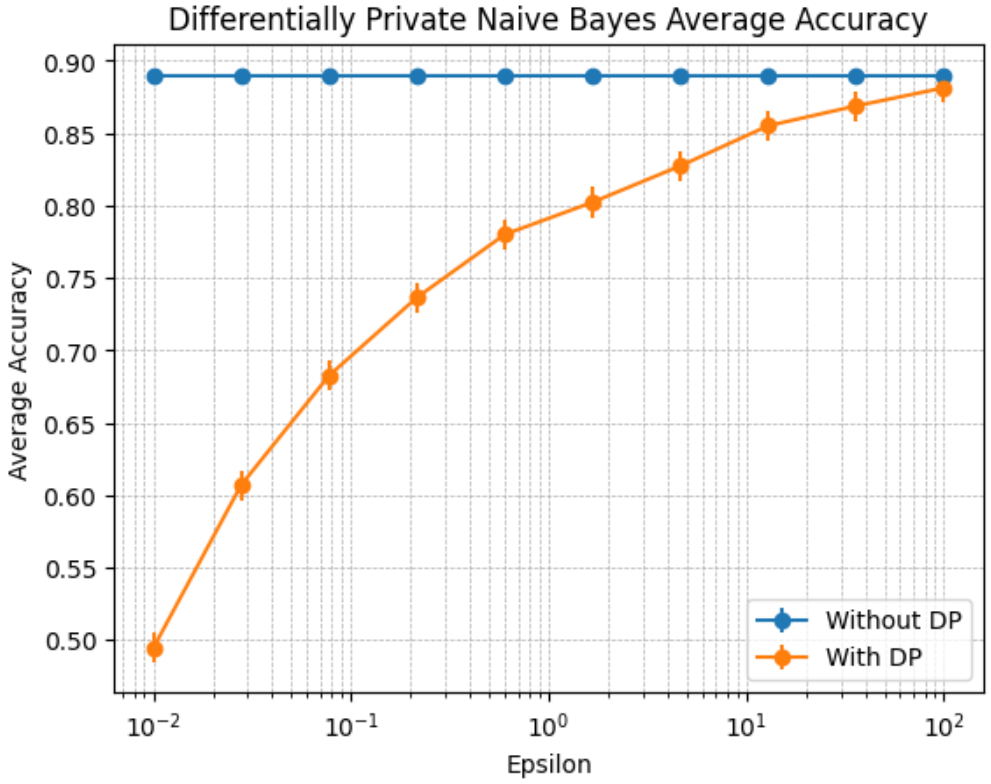


Figure 3.2. Average accuracy of the Gaussian Naïve Bayes and Gaussian Naïve Bayes with Differential Privacy Comparison for various ϵ values.

3.2.2. Logistic Regression Accuracy

In order to observe the differential privacy effects, Logistic Regression Classifier algorithm is ran several times with various ϵ values to compare the accuracy. We picked 10 different ϵ values between 10^{-2} and 10^{2}, and we ran both Logistic Regression without differential privacy (LR) and Logistic Regression with differential privacy (LR_DP) 15 times to take the average values to avoid undesired incorrect results. As shown in Figure 3.3, the average accuracy of the LR is 0.70, while the average accuracy of the LR_DP initiates at 0.1 with the least ϵ value in our work 10^{-2}. In the same graph, it can be seen that as the ϵ value increases, the LR_DP accuracy also increases. Unlike GNB_DP, the accuracy of LR_DP approaches to LR when ϵ is $10^{0.66}$. Finally, when the ϵ value is set to $10^{1.1}$, the LR and the LR_DP accuracy values are met. The figure clearly shows that when the ϵ value is small, the data is very heavily protected, however, consequently classification accuracy is poor starting around 0.30. While the ϵ value is increasing which indicates less privacy; classification accuracy is resulting better. At the end of the graph when we set ϵ to $10^{1.55}$, LR_DP accuracy even exceed the LR accuracy with 0.71.

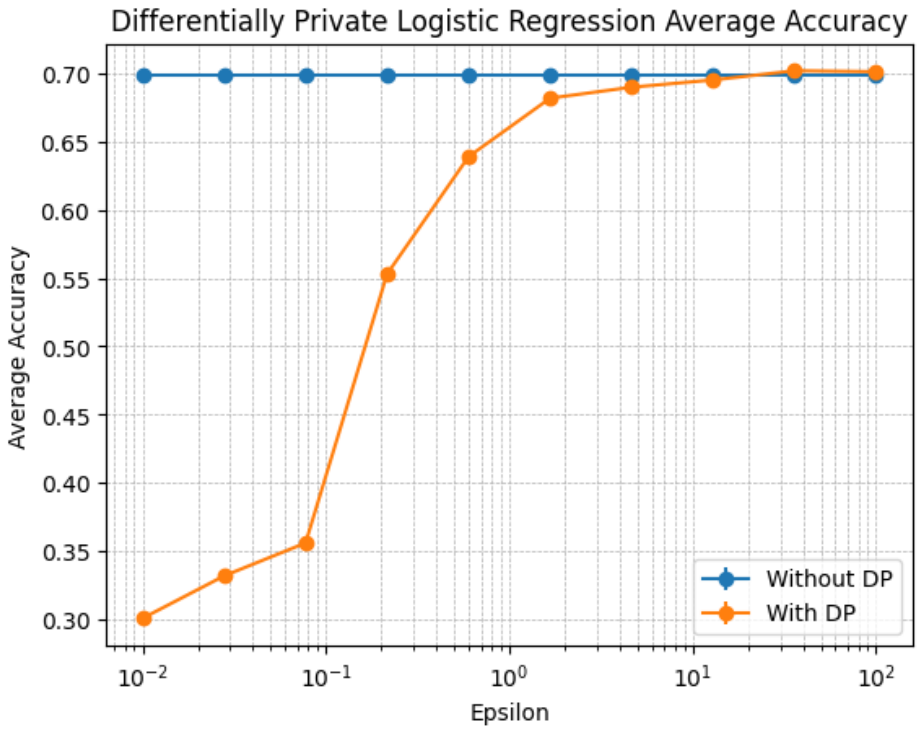


Figure 3.3. Average accuracy of the Logistic Regression and Logistic Regression with Differential Privacy Comparison for various ϵ values.

3.2.3. Decision Tree Accuracy

In order to observe the differential privacy effects, Decision Tree Classifier algorithm is ran several times with various ϵ values to compare the accuracy. We picked 3 different ϵ values between 10^{-2} and 10^{2}, and we ran both Decision Tree without differential privacy (DCT) and Decision Tree with differential privacy (DCT_DP) 15 times to take the average values to avoid undesired incorrect results. As shown in Figure 3.4, the average accuracy of the DCT is 0.88, while the average accuracy of the DCT_DP initiates at 0.745 with the least ϵ value in our work 10^{-2}. In the same graph, it can be seen that as the ϵ value increases, the DCT_DP accuracy also increases. The accuracy of DCT_DP approaches to DCT when ϵ is 10^{2}. However, when the ϵ value is set to 10^{2}, the DCT and the DCT_DP accuracy values couldn't meet unlike the other two models. The figure clearly shows that when the ϵ value is small, the data is very heavily protected, however this time, with the Decision Tree introduction, classification accuracy isn't poor, it starts around 0.745 which is significantly higher when we compare DCT_DP with GNB_DP and LR_DP. While the other two models show poor classification performances with a small ϵ value, DCT_DP shows way better classification performance. Although with ϵ value of 10^{2}, DCT and DCT_DP can't meet, we still observe better classification accuracies which indicates less privacy in the data.

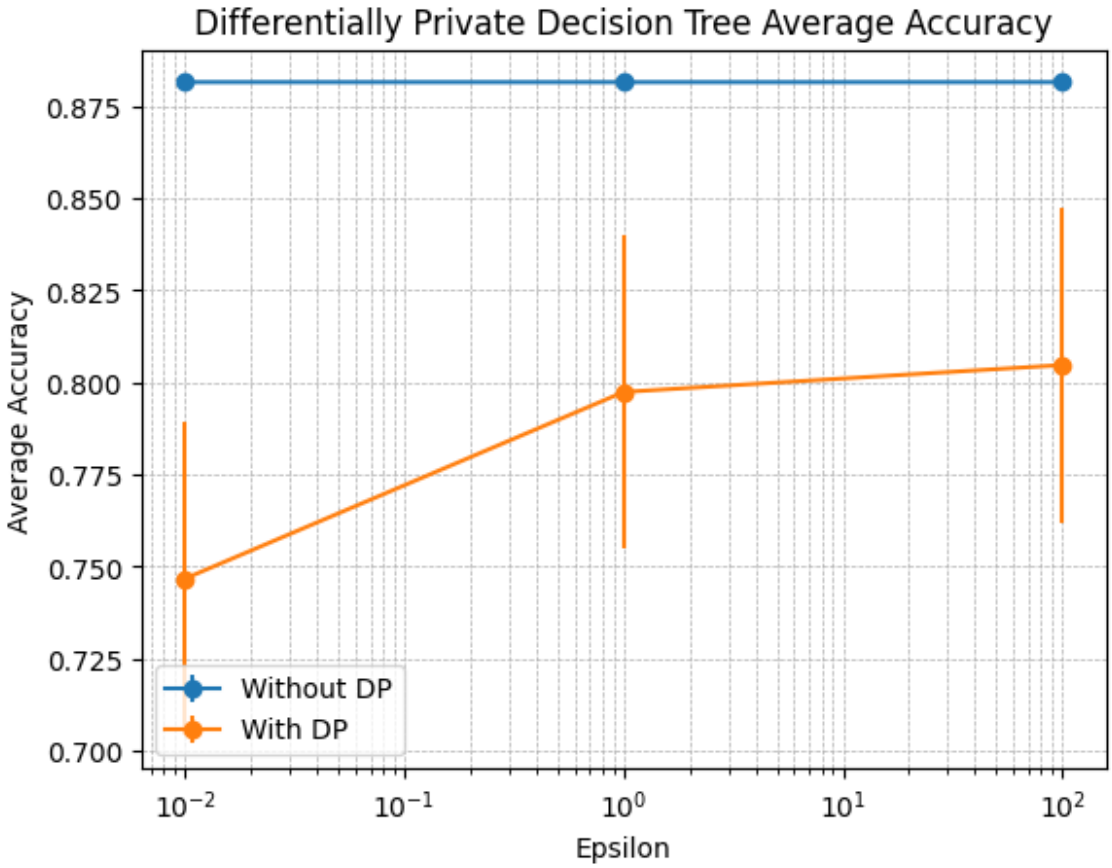


Figure 3.4. Average accuracy of the Decision Tree and Decision Tree with Differential Privacy Comparison for various ϵ values.

3.3. Differential Privacy Effect to the Data

3.3.1. Differential Privacy Effect to the Data in Gaussian Naïve Bayes

The Gaussian Naive Bayes algorithm in diffprivlab injects Laplacian noise during the model training to provide data privacy. Due to the working mechanism of the Gaussian Naive Bayes, the features' mean, and variance values are computed for each class independently. The random noise injection is applied to the computed mean and variance values where privacy is guaranteed. The ϵ value is the key to the protection level in differential privacy. It is also known as the privacy parameter; a smaller ϵ parameter results in stronger privacy but less accuracy. We examined how the data histograms are changed when Laplacian noise is injected for the ϵ values: 1, 0.1, and 0.01.

3.3.1.1. Histograms of Mean and Variances When ϵ is 1

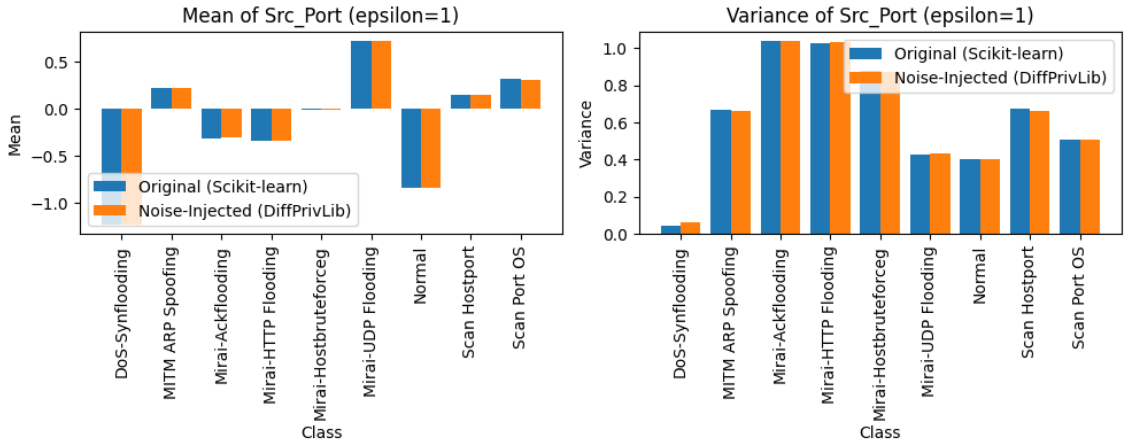


Figure 3.5. Mean and Variance histograms of Src_Port feature for each classes in ϵ 1.

As shown in Figure 3.5, when the ϵ is set to 1, the effect of the Laplacian noise to the Mean and the Variance of the Src_Port is not strong. The difference between the original and the noise-injected means and variances are nearly zero.

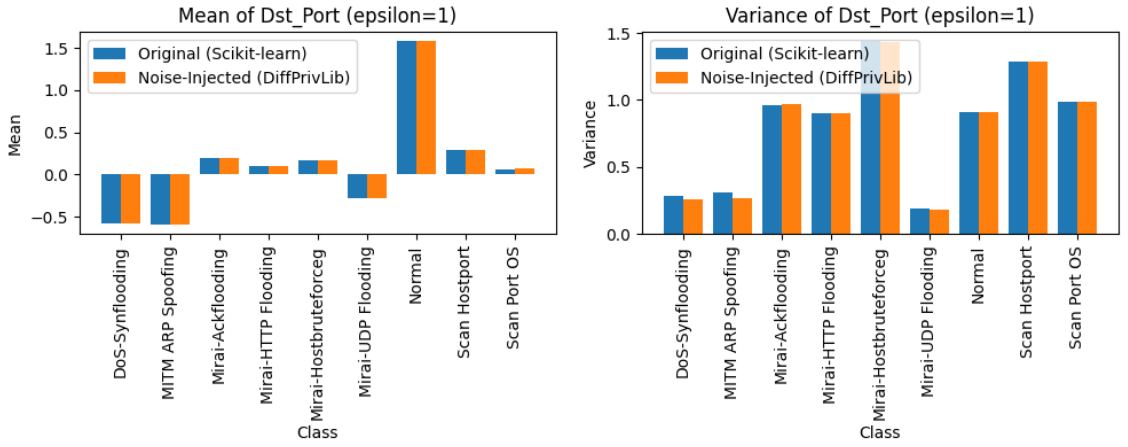


Figure 3.6. Mean and Variance histograms of Dst_Port feature for each classes in ϵ 1.

In Figure 3.6, ϵ is set to 1, and the effect of the Laplacian noise to the Mean and the Variance of the Dst_Port is similar to the Src_Port where the difference between the original and the noise-injected means and variances are very low.

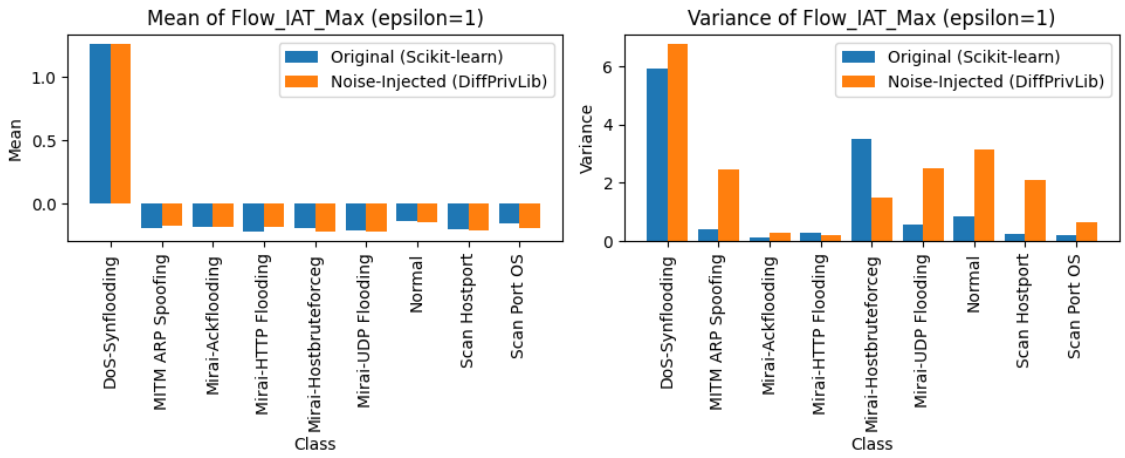


Figure 3.7. Mean and Variance histograms of Flow_IAT_Max feature for each classes in ϵ 1.

For the Flow_IAT_Max feature which represents the maximum time between the arrival of two consecutive packets, in Figure 3.7, ϵ is set to 1 again. The effect of the Laplacian noise to the Mean and the Variance of the Flow_IAT_Max is not zero this time but low, as the range of y-axis of the graph is short.

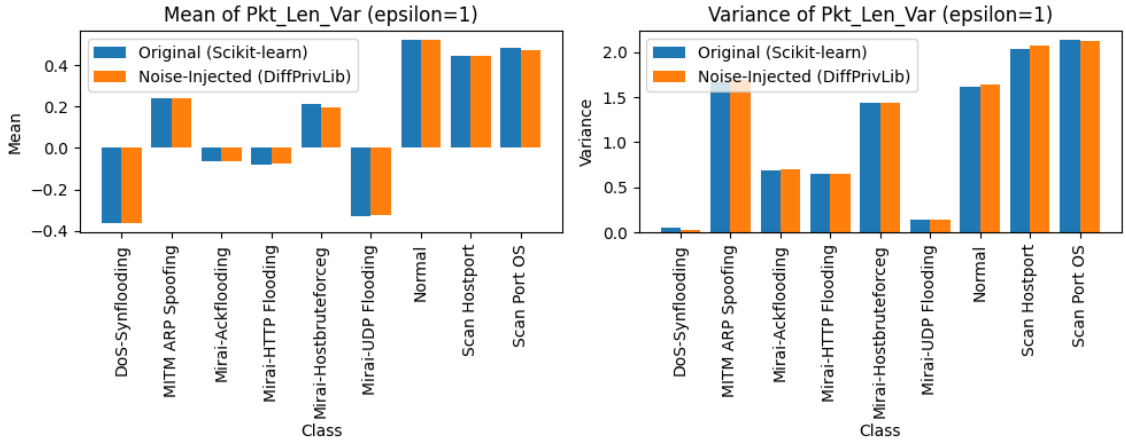


Figure 3.8. Mean and Variance histograms of Pkt_Len_Var feature for each classes in ϵ 1.

While the mean and variance values are nearly the same with differential privacy and without differential privacy for Pkt_Len_Var feature under ϵ 1, we can see a few short differences in the variance histogram for the “Normal” and “Scan Hostport” classes in Figure 3.8.

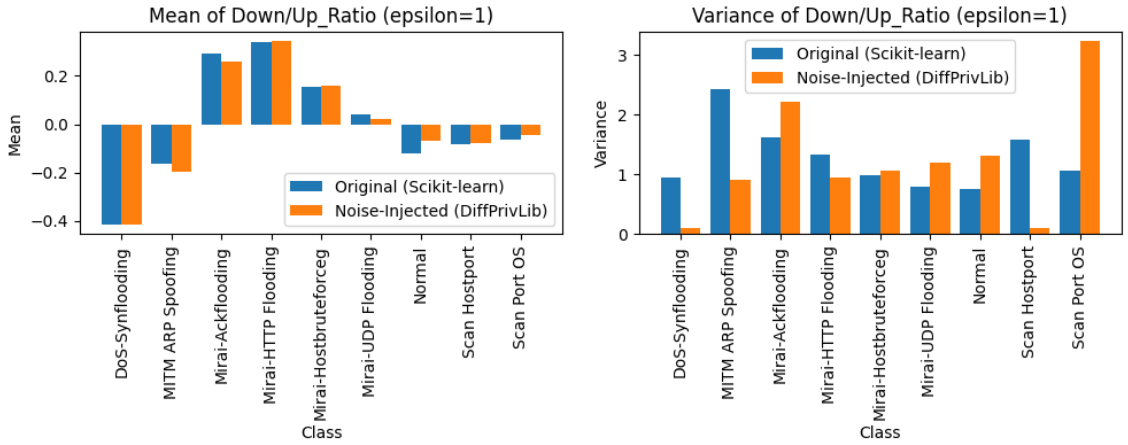


Figure 3.9. Mean and Variance histograms of Down/Up_Ratio feature for each classes in ϵ 1.

Down/Up_Ratio shows whether the traffic is flowing in one direction or balanced as an informative feature. When ϵ is set to 1, noisy and the original means have balanced histograms that indicates there isn't noise effect much. However, noisy and the original variances are slightly differed. Yet, we must note that the range of y-axis of the mean graph is short again as shown in Figure 3.9.

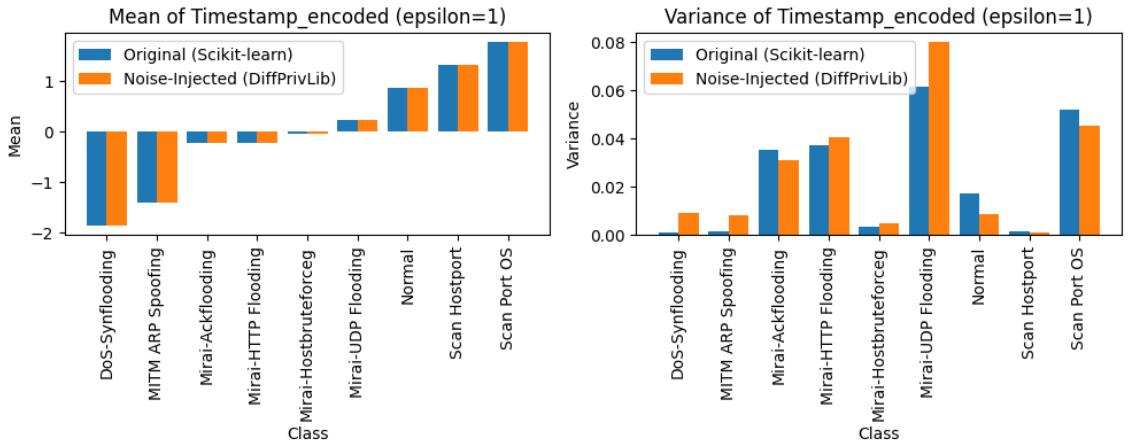


Figure 3.10. Mean and Variance histograms of Timestamp_Encoded feature for each classes in ϵ 1.

The last feature histogram is Timestamp_Encoded. When ϵ is set to 1, the difference of means is nearly zero, while the difference of variances isn't that straight forward. Although the range of y-axis of the variance graph is very short again, we can clearly see the effect of differential privacy in our data shown in Figure 3.10.

3.3.1.2. Histograms of Mean and Variances When ϵ is 0.1

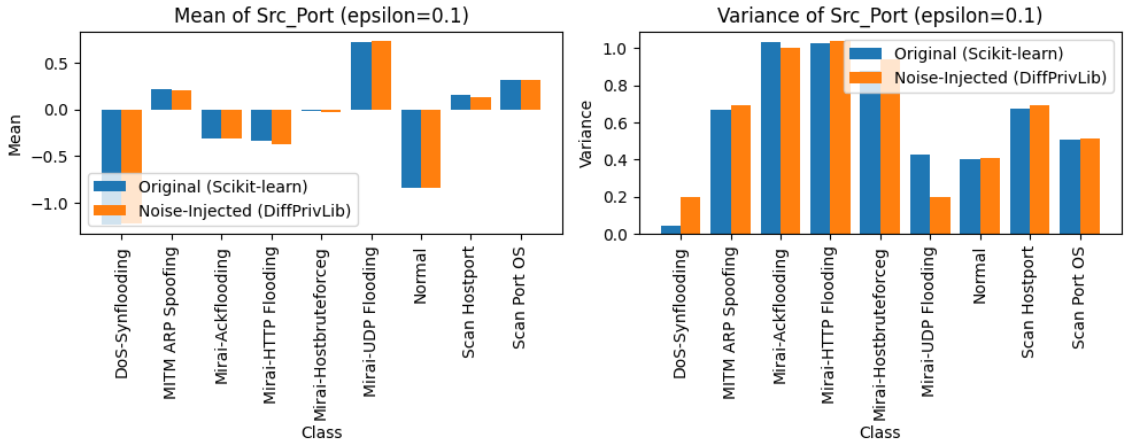


Figure 3.11. Mean and Variance histograms of Src_Port feature for each classes in ϵ 0.1.

When set ϵ is set to 0.1, we started to observe differences between the original data and the noise injected data for the Src_Port feature shown in Figure 3.11. Here, the difference of the means for the "MITM ARP Spoofing", "Mirai-HTTP Flooding", "Mirai-UDP Flooding" and "Scan Hostport" classes are clearly observable while the variance values are also affected by the injected noise as well.

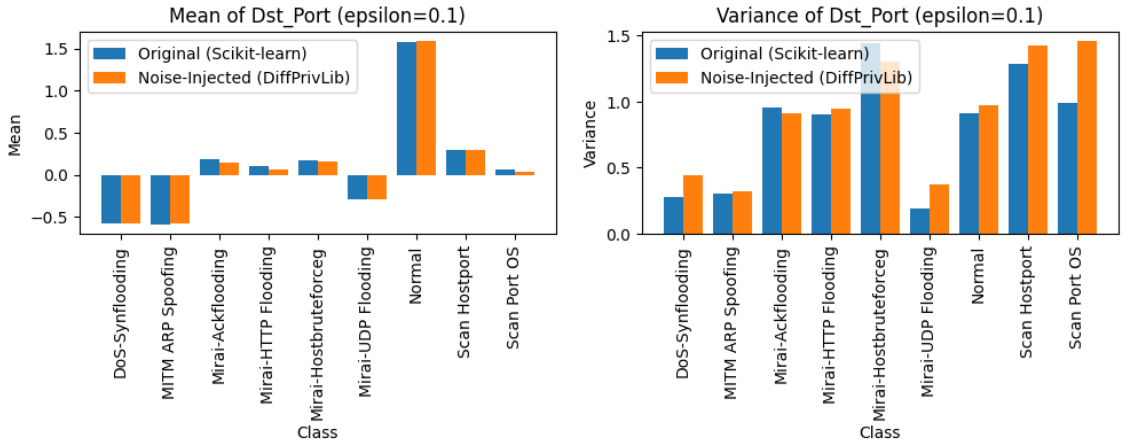


Figure 3.12. Mean and Variance histograms of Dst_Port feature for each classes in ϵ 0.1.

This time the difference between noisy and the original variance values in Figure 3.12 are significant for the Dst_Port feature when we set the ϵ to 0.1. Especially the

"Mirai-Ackflooding", "Mirai-HTTP Flooding" and, "Mirai-Hostbruteforce" classes are widely differed with the ϵ 0.1 value.

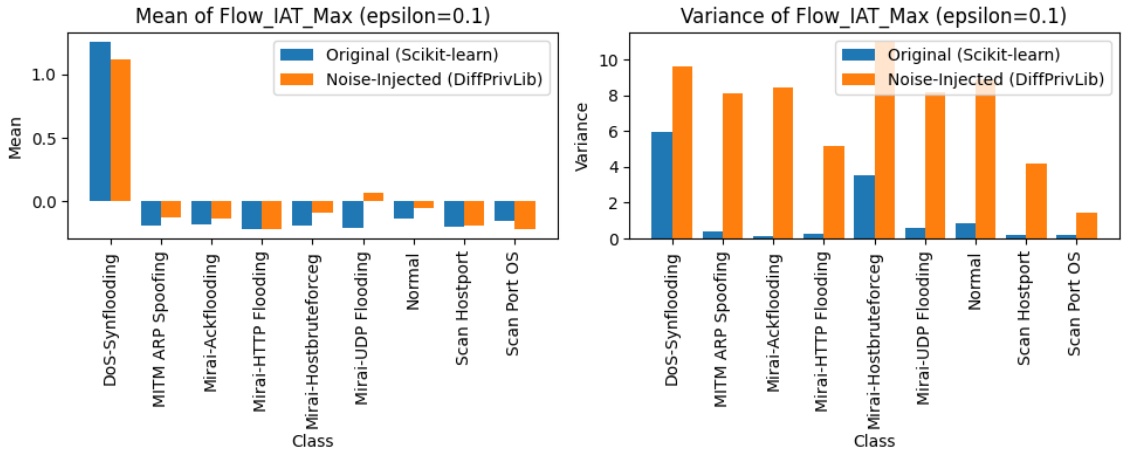


Figure 3.13. Mean and Variance histograms of Flow_IAT_Max feature for each classes in ϵ 0.1.

The effect of the Laplacian noise to the Mean and the Variance of the Flow_IAT_Max in Figure 3.13 is wider this time when the ϵ is set to 0.1. Laplacian noise effect is much more significant in this feature especially for the variance histograms.

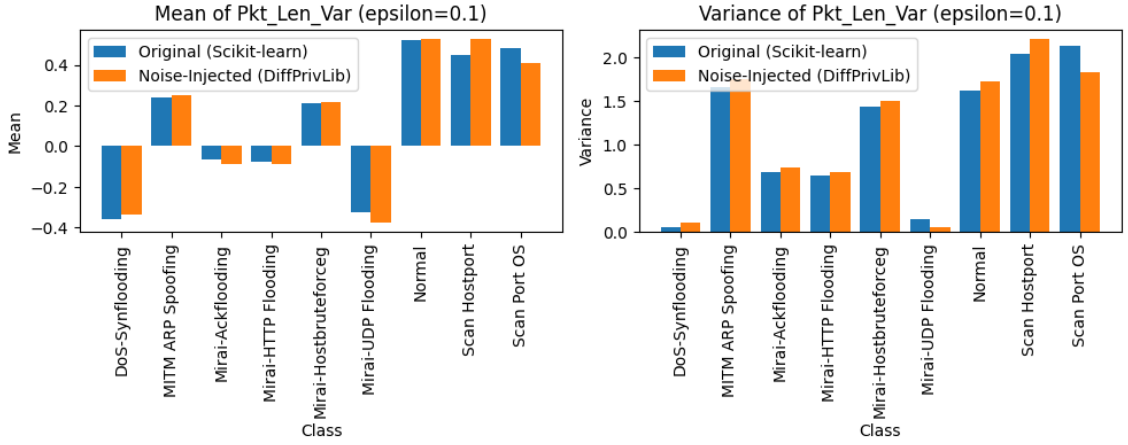


Figure 3.14. Mean and Variance histograms of Pkt_Len_Var feature for each classes in ϵ 0.1.

Pkt_Len_Var feature represents the packet length variation within a flow that measures the variance. When ϵ is 0.1, we see the effect of differential privacy is higher

compared to ϵ 1. Difference between noisy and the original values are increased in this feature for each classes as shown in Figure 3.14.

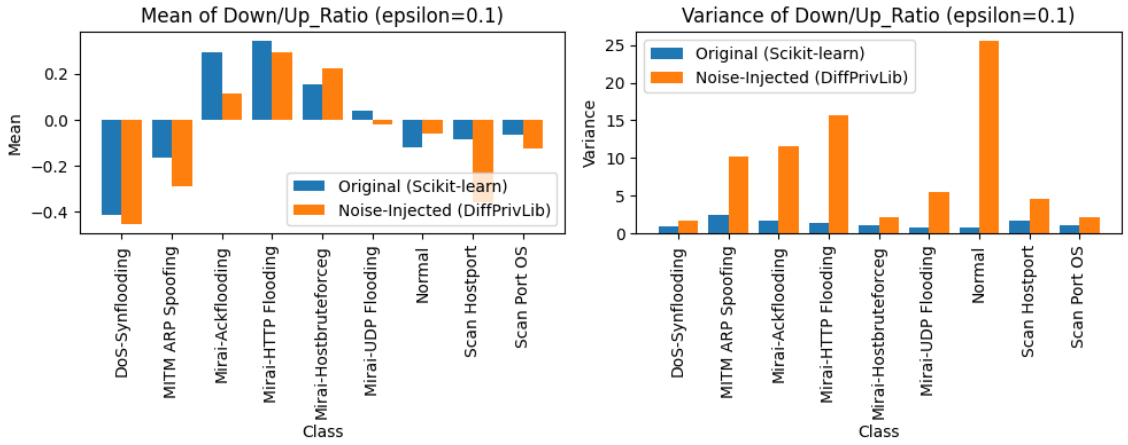


Figure 3.15. Mean and Variance histograms of Down/Up_Ratio feature for each classes in ϵ 0.1.

The effect of ϵ is much more significant this time for the Down/Up_Ratio feature shown in Figure 3.15. While the gap between noisy mean and the original mean is significantly higher for ϵ is 0.1; the gap between noisy variance and the original variance is much higher.

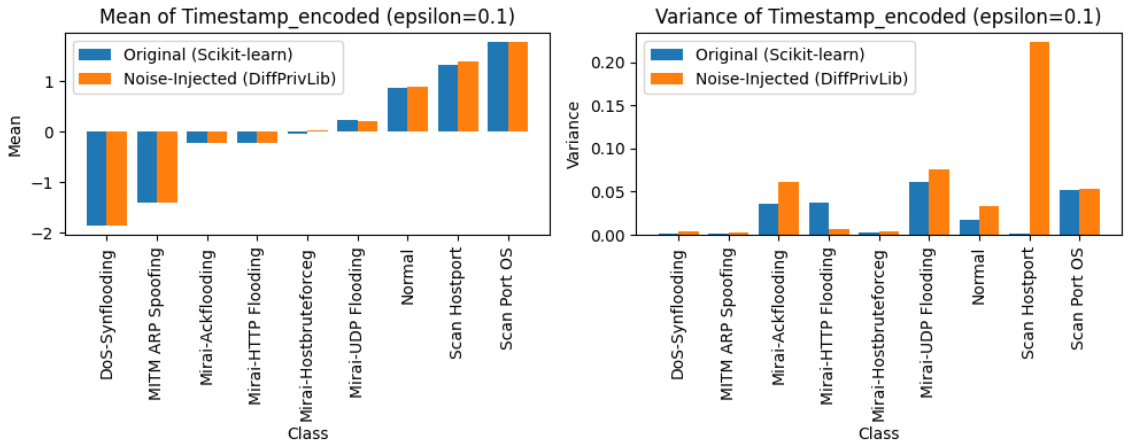


Figure 3.16. Mean and Variance histograms of Timestamp_Encoded feature for each classes in ϵ 0.1.

As shown in the Figure 3.16, Timestamp_Encoded feature which represents the categorical timestamp of when the observation was made or when the traffic occurred is viewed. When ϵ is set to 1, the difference of means is nearly zero, however with ϵ 0.1,

we can clearly see the effect of differential privacy in our data especially in the variance histograms.

3.3.1.3. Histograms of Mean and Variances When ϵ is 0.01

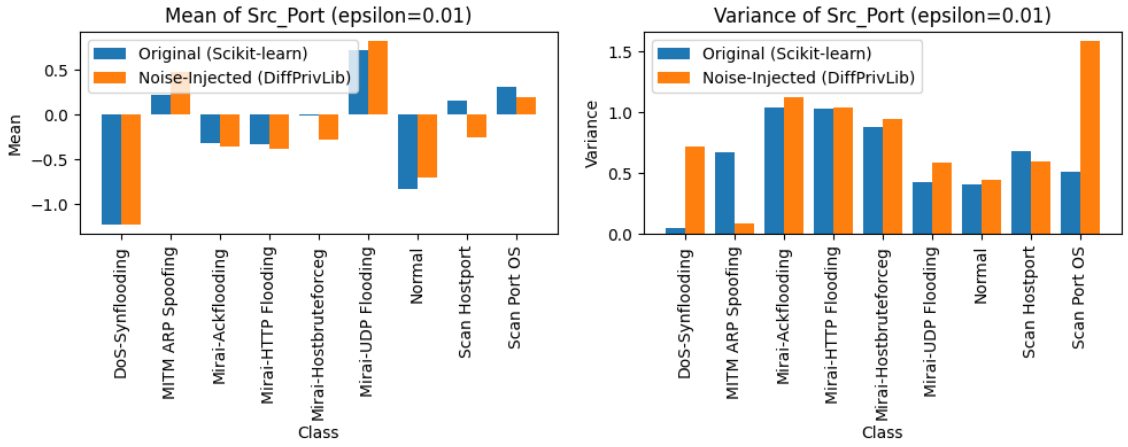


Figure 3.17. Mean and Variance histograms of Src_Port feature for each classes in ϵ 0.01.

Lastly, ϵ is set to 0.01, the gap between the original data and the noise injected data in the histograms for each class of the Src_Port feature is much higher. In Figure 3.17, the difference of the means for all the classes are clearly observable as well as the variance values due to the Laplacian noise injection.

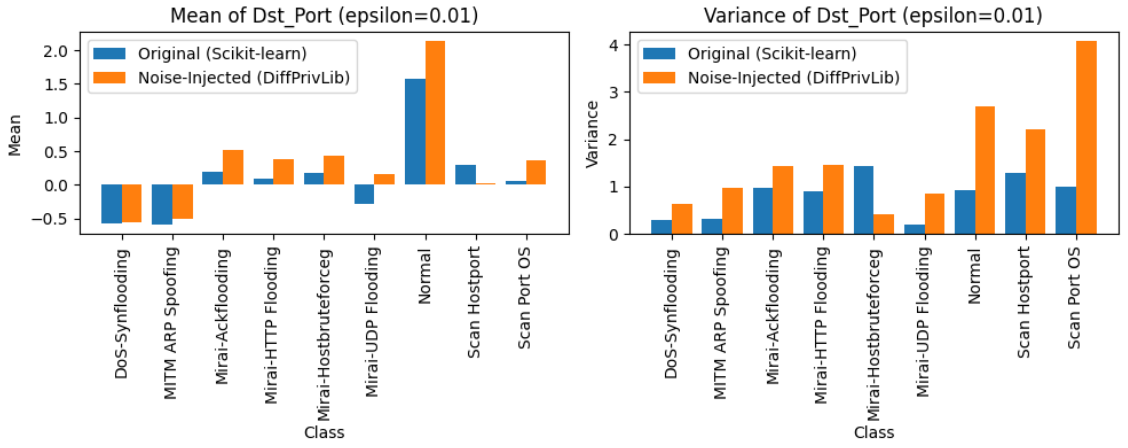


Figure 3.18. Mean and Variance histograms of Dst_Port feature for each classes in ϵ 0.01.

This time the difference between noisy and the original variance values are significant for the Dst_Port feature when we set the ϵ to 0.01. Especially "Normal", "Scan Hostport" and "Scan Port OS" classes are widely differed with the ϵ 0.01 value in Figure 3.18.

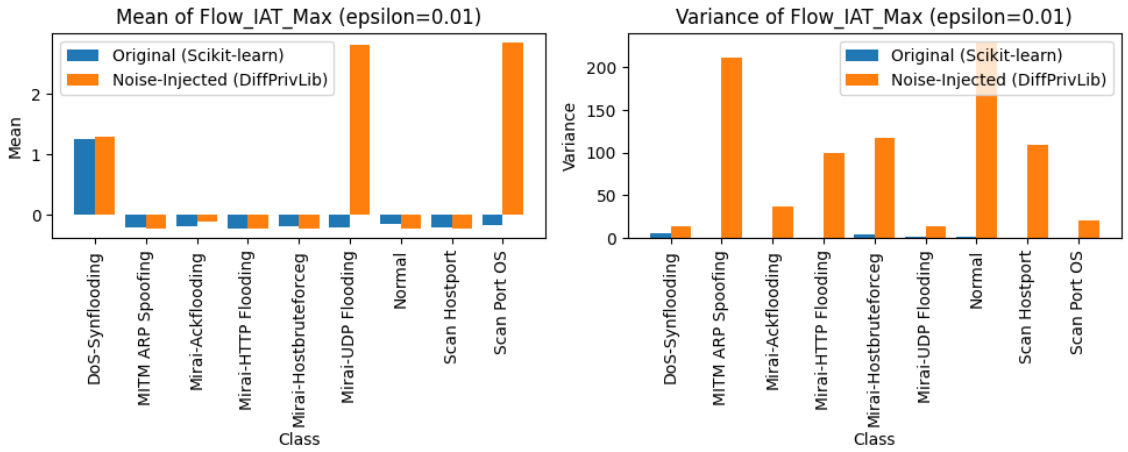


Figure 3.19. Mean and Variance histograms of Flow_IAT_Max feature for each classes in ϵ 0.01.

Figure 3.19 is another Flow_IAT_Max feature representation when ϵ is 0.01. As shown in the figure, noise is heavily affecting both the mean and the variance values of all classes. Thus, we see very limited representations of the original data histograms as the noise injected ones are widely high.

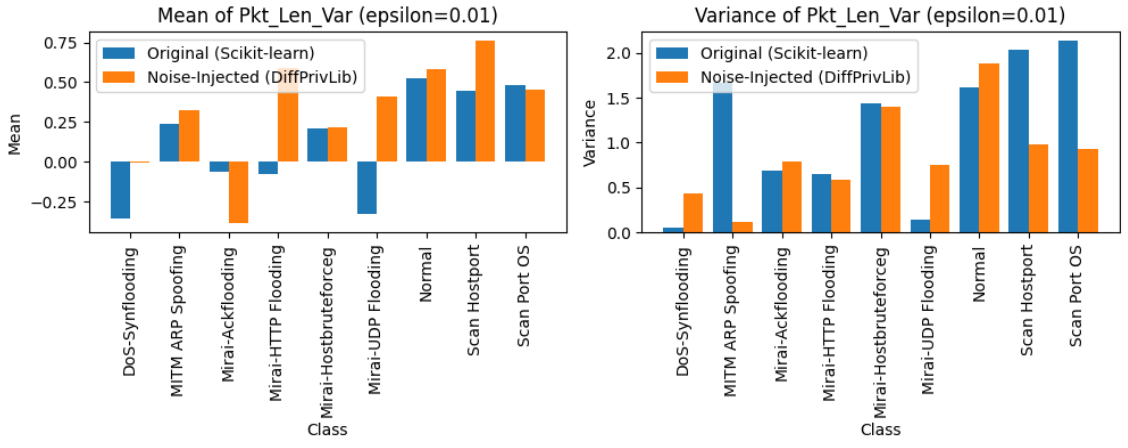


Figure 3.20. Mean and Variance histograms of Pkt_Len_Var feature for each classes in ϵ 0.01.

In the Figure 3.20, we see the mean and variance histograms of Pkt_Len_Var feature. Again, when we set ϵ to 0.01, the noise injection is widely observable for both mean and variance values in each class.

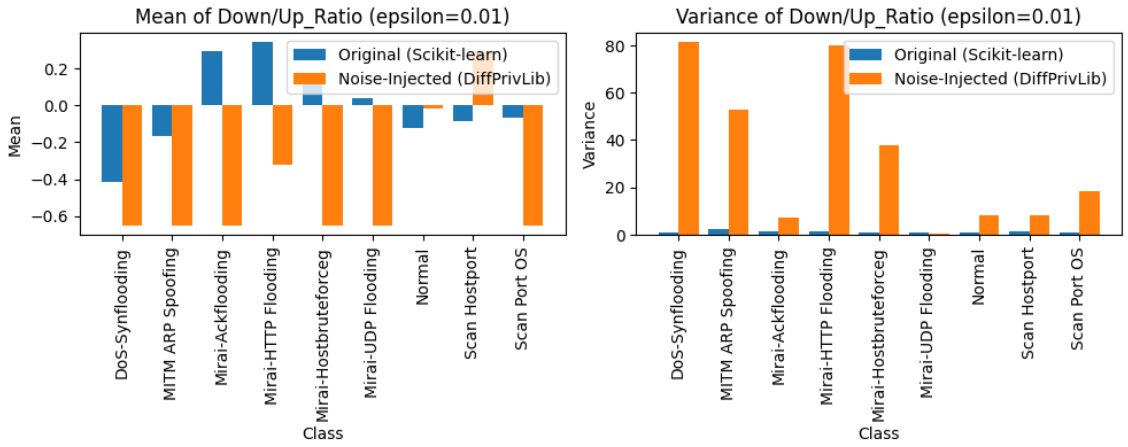


Figure 3.21. Mean and Variance histograms of Down/Up_Ratio feature for each classes in ϵ 0.01.

Down/Up_Ratio feature has already affected by ϵ 0.1 value however in order to see the actual difference between the ϵ values we can compare "DoS-Synflooding" class in all the ϵ values. While the original variance of "DoS-Synflooding" class is around 1 originally, when the ϵ is 1, it becomes 0.1 due to the noise as in Figure 3.8. When the ϵ is 0.1, variance of "DoS-Synflooding" class becomes 2 as shown in Figure 3.14, and lastly when the ϵ is set to 0.01, variance of "DoS-Synflooding" class becomes 80 as in Figure 3.21 which is much higher than the original variance.

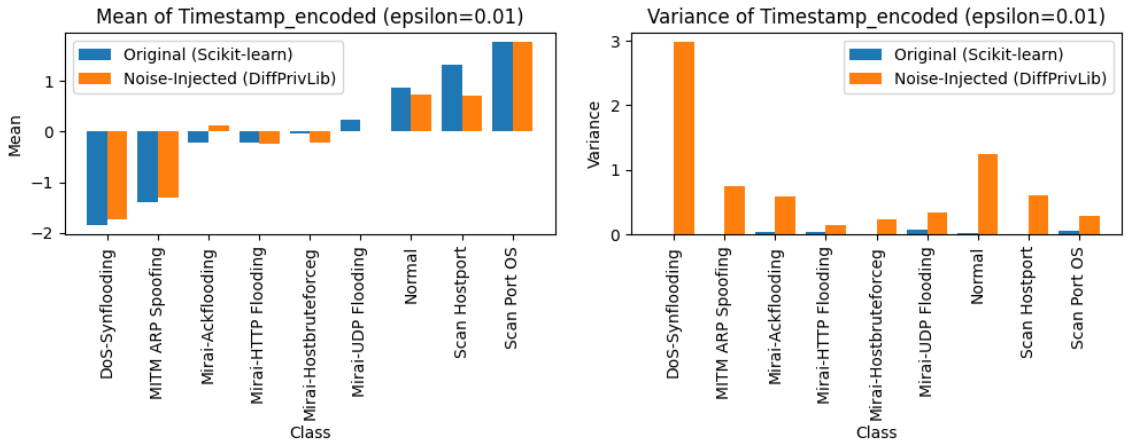


Figure 3.22. Mean and Variance histograms of Timestamp_Encoded feature for each classes in ϵ 0.01.

Lastly when we look at the Timastamp_Encoded feature under ϵ 0.01 noise injection shown in Figure 3.22, we can clearly see that both mean, and variance values are heavily changed due to the Laplacian noise.

3.3.1.4. Accuracy comparison of GNB and GNB_DP for ϵ values 0.01, 0.1 and 1.

While the noise injected feature representations are reviewed, Gaussian Naïve Bayes classification of each ϵ with both original and noise injected feature trained and accuracies of all 3 trainings are save in the Figure 3.23 below.

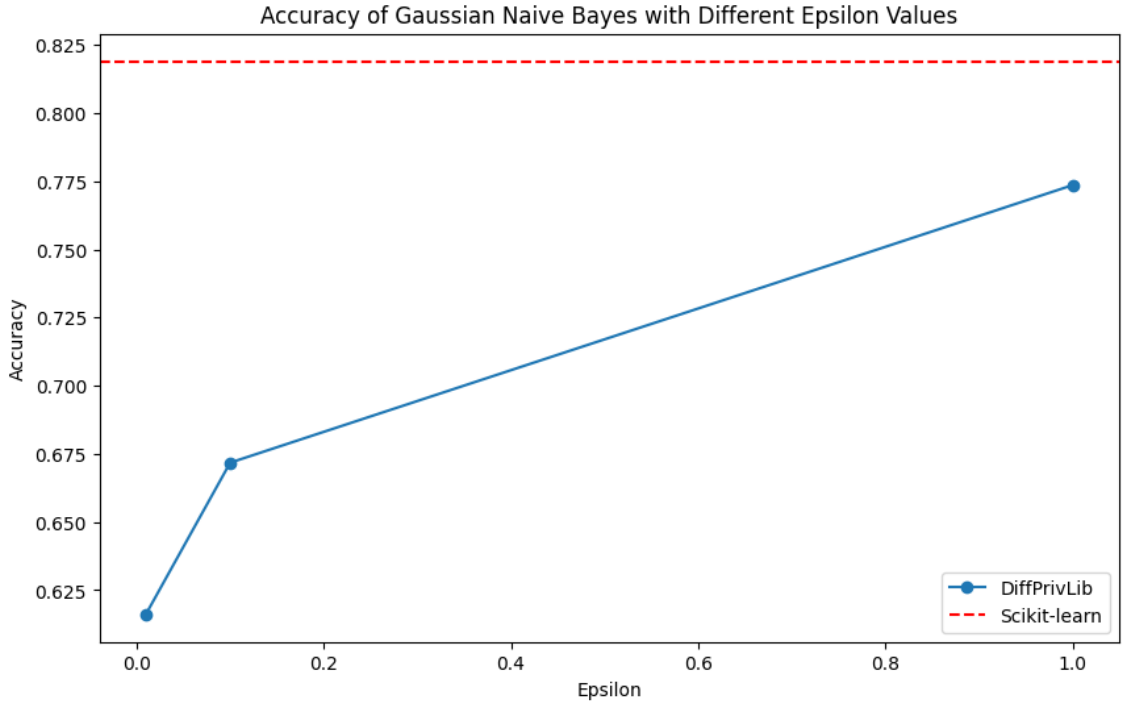


Figure 3.23. Accuracy comparison of the Gaussian Naïve Bayes and Gaussian Naïve Bayes with Differential Privacy for ϵ values 0.01, 0.1 and 1.

As shown in the graph, when ϵ is 0.01, the classification accuracy is widely poor around 0.6 compared to the other ϵ values. When ϵ is set to 0.1, we observe a classification accuracy of 0.675. Lastly, when ϵ is set to 1, we had a better classification accuracy of 0.775.

3.3.2. Differential Privacy Effect to the Data in Logistic Regression

In Diffprivlib's implementation of logistic regression, Laplace noise is injected

during the model training process to ensure differential privacy. Specifically, the noise is added to the optimization procedure, where the gradient descent is used to minimize the loss function to learn the model's coefficients (weights). During each iteration of the gradient descent optimization, the logistic regression algorithm updates its coefficients by calculating the gradient of the loss function with respect to the weights. In Diffprivlib, this gradient is perturbed by adding random Laplace noise. This noise ensures that the model training process does not reveal specific details about any individual data point. The ϵ value is the key to the protection level in differential privacy. It is also known as the privacy parameter; a smaller ϵ parameter results in stronger privacy but less accuracy. We examined how the weight histograms are changed when Laplacian noise is injected for the ϵ values: 1, 0.1, and 0.01.

3.3.2.1. Histograms of Coefficients When ϵ is 1

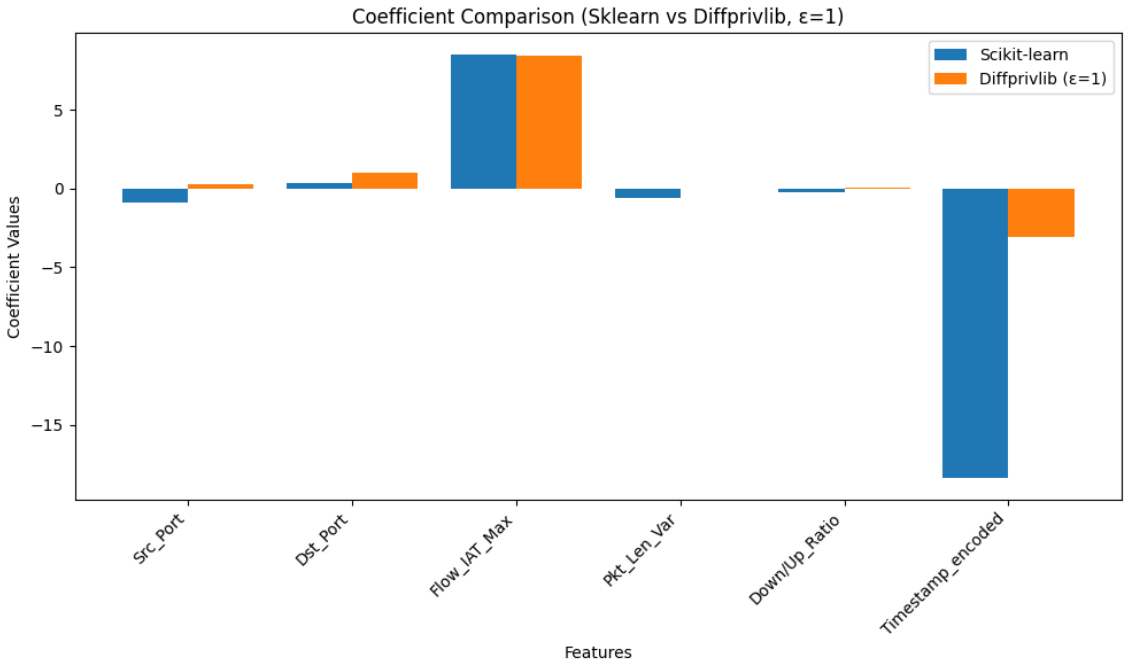


Figure 3.24. Coefficient histograms of each feature in ϵ 1.

Logistic Regression algorithm calculates the weights and updates them in each iteration. Our model is set to 1000 iterations, where the coefficient values in the graph are taken from the 1000th or the final iteration if it converges earlier. In each iteration, the Laplacian noise is injected into the calculated weight value and in the next iteration the algorithm utilizes that noise injected weight to calculate its weight for that

individual iteration. In Figure 3.24, we see the cumulative calculated coefficient by scikit-learn logistic regression in the final iteration of each independent feature in the dataset in blue. On the other hand, the orange values are the random Laplacian noise added coefficient values cumulatively calculated by the diffprivlib logistic regression. Since we set the ϵ to 1 at this point, we don't see very big effect of the differential privacy. Yet, there is still a significant difference between LR and LR_DP coefficients in the "Timestamp_encoded" feature.

3.3.2.2. Histograms of Coefficients When ϵ is 0.1

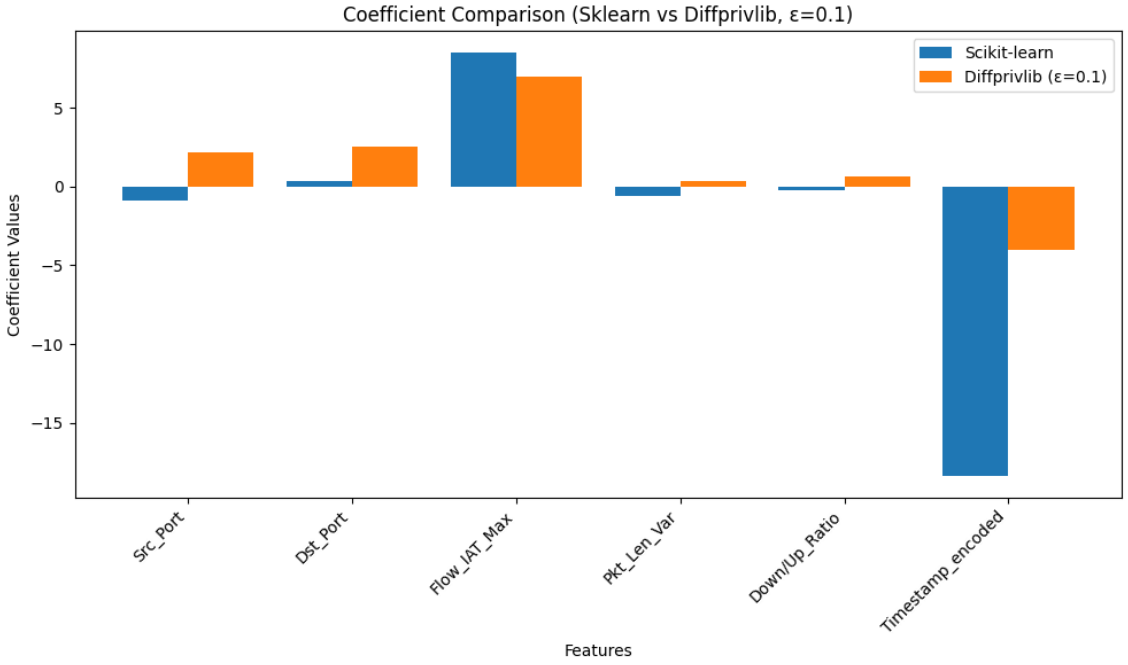


Figure 3.25. Coefficient histograms of each feature in ϵ 0.1.

In Figure 3.25, we see the cumulative calculated coefficient by scikit-learn logistic regression in the final iteration of each independent feature in the dataset in blue again. The calculated values by scikit-learn logistic regression are the same as the values when we set ϵ to 1 and 0.1. Because there is no effect of differential privacy in the scikit-learn library. On the other hand, the orange values are the random Laplacian noise added coefficient values cumulatively calculated by the diffprivlib logistic regression. Since we set ϵ to 0.1 at this examination, we started to see effects of the differential privacy significantly. Almost all the feature coefficients are differed between LR and LR_DP when we set ϵ to 0.1.

3.3.2.3. Histograms of Coefficients When ϵ is 0.1

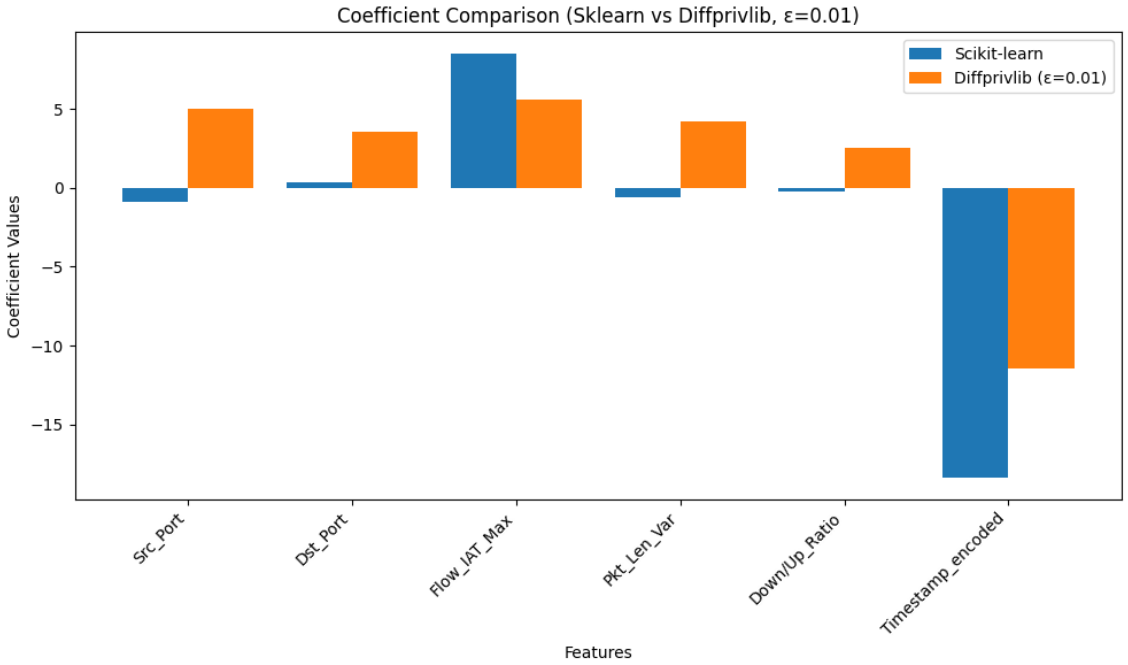


Figure 3.26. Coefficient histograms of each feature in ϵ 0.01.

Lastly in Figure 3.26, we see the cumulative calculated coefficient by scikit-learn logistic regression in the final iteration of each independent feature in the dataset in blue for the last time. The calculated values by scikit-learn logistic regression are the same as the values when we set ϵ to 1, 0.1 and 0.01. Because there is no effect of differential privacy in the scikit-learn library. On the other hand, the orange values are the random Laplacian noise added coefficient values cumulatively calculated by the diffprivlib logistic regression. Since we set ϵ to 0.01 in this examination, we see heavily significant effects of the differential privacy to the coefficients. All the feature coefficients are highly differed between LR and LR_DP when we set ϵ to 0.01. While this effect ensures the privacy of the data points, it causes the prediction accuracy to heavily drop.

3.3.2.4. Accuracy comparison of LR and LR_DP for ϵ values 0.01, 0.1 and 1.

While the noise injected feature coefficient representations are reviewed, Logistic Regression classification of each ϵ with both original and noise injected feature trained and accuracies of all 3 trainings are saved in the Figure 3.27 below.

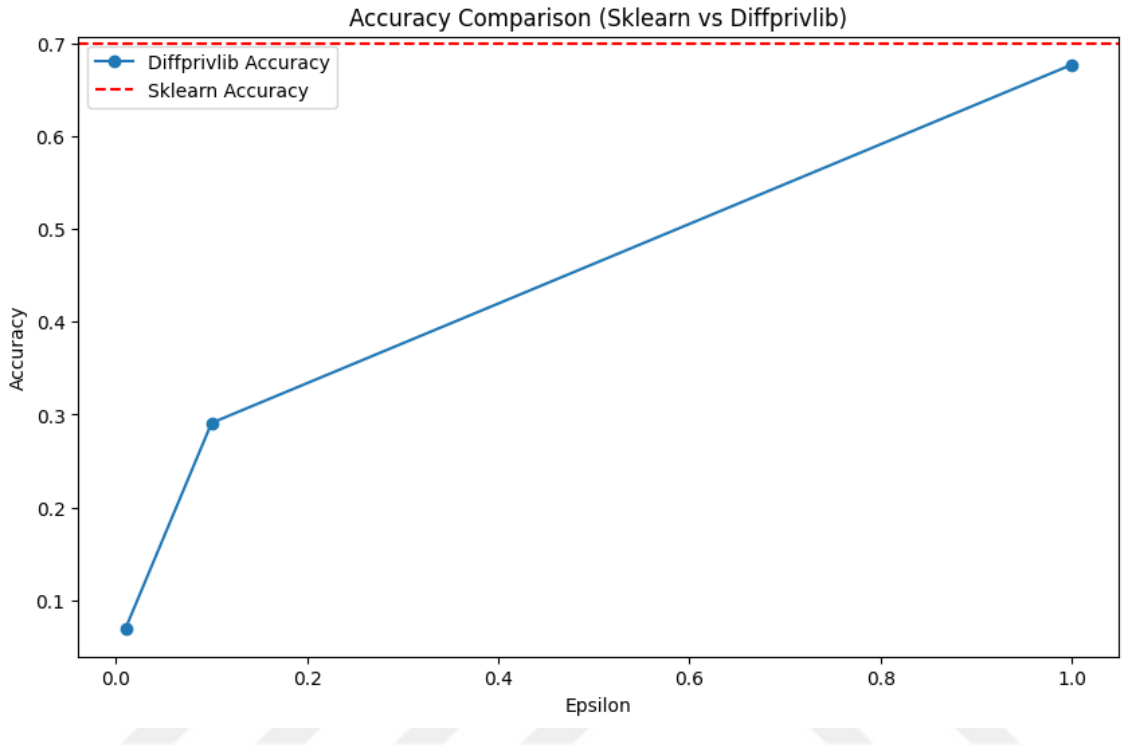


Figure 3.27. Accuracy comparison of the Logistic Regression and Logistic Regression with Differential Privacy for ϵ values 0.01, 0.1 and 1.

As shown in the graph, when ϵ is 0.01, the classification accuracy is widely poor less than 0.1 compared to the other ϵ values. When ϵ is set to 0.1, we observe a classification accuracy of 0.30. Lastly, when ϵ is set to 1, we had a significantly better classification accuracy of 0.69.

3.3.2.5. Prediction comparison of LR and LR_DP for ϵ values 0.01, and 1.

The graphs below aim to visually compare the predictions made by two models: LR and LR_DP for ϵ values 0.01, and 1. On the y-axis, there are each prediction classes such as "Mirai-HTTP Flooding," "Normal," etc. These represent the classes of our dataset for IoT intrusion detection. On the x-axis, the test sample index represents each instance from the test set. The idea is to show how the predictions vary across the test set.

- Blue Dots: Represent predictions made by the non-private (scikit-learn) logistic regression model.
- Orange Dots: Represent predictions made by the private (diffprivlib) logistic regression model with ε values of 0.01 and 1.

In the Figure 3.28, with ε = 0.01, the orange dots are slightly more scattered compared to the blue dots, indicating some deviation in predictions due to the noise introduced for privacy.

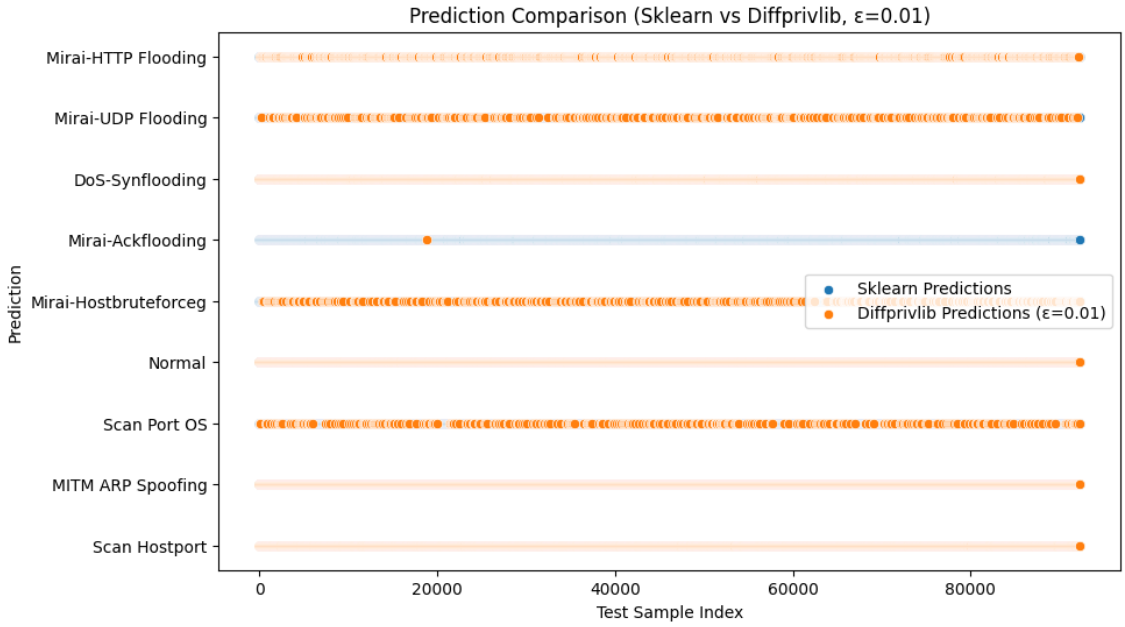


Figure 3.28. Prediction comparison of the Logistic Regression and Logistic Regression with Differential Privacy for ϵ value 0.01.

In the Figure 3.29, with ε = 1, the predictions between the two models (scikit-learn and diffprivlib) align more closely, as a higher epsilon value introduces less noise and provides weaker privacy guarantees.

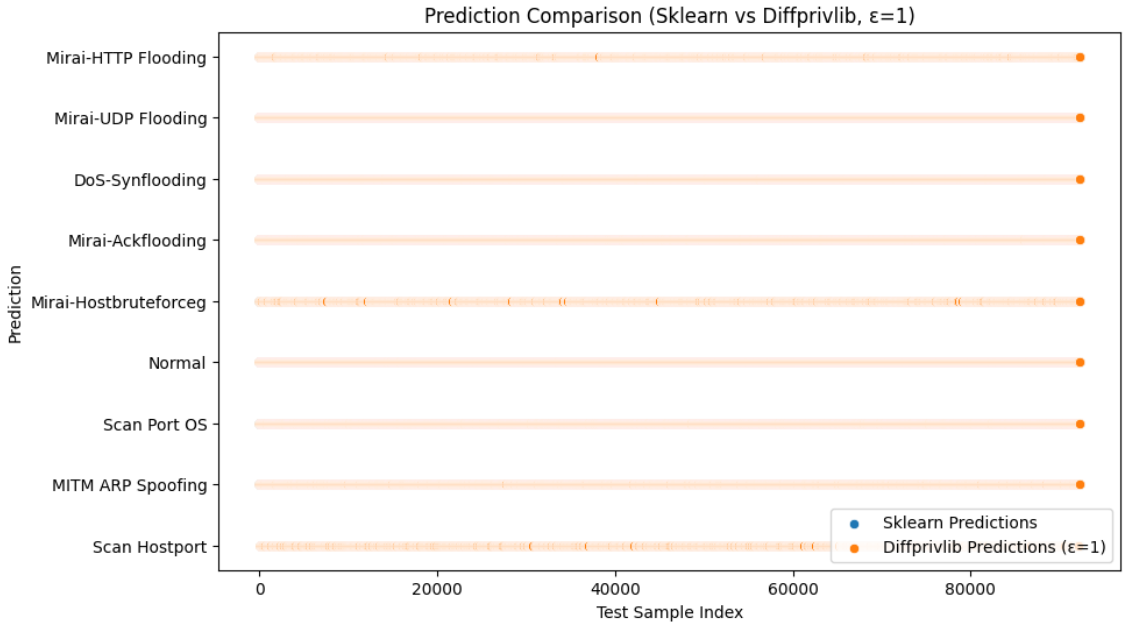


Figure 3.29. Prediction comparison of the Logistic Regression and Logistic Regression with Differential Privacy for ϵ value 1.

4. DISCUSSION

Our research demonstrates the effective integration of Gaussian Naive Bayes, Logistic Regression and Decision Tree Classifiers with Differential Privacy to address privacy concerns in IoT network data classification. We showed that all t three algorithms, when combined with differential privacy mechanisms, can classify IoT network traffic while preserving individual privacy. Our experiments with varying levels of the privacy parameter ϵ highlight the trade-off between privacy and accuracy, providing a clear framework for balancing these crucial aspects.

Performance of Gaussian Naïve Bayes with Differential Privacy (GNB_DP) indicate that the GNB_DP maintains a reasonable classification accuracy even at lower ϵ values, which offer stronger privacy guarantees. The accuracy increases as ϵ increases, aligning closely with the non-privacy-preserving GNB model at higher ϵ values. This suggests that GNB_DP can be fine-tuned to achieve an optimal balance between privacy and utility based on specific application requirements.

The performance of Logistic Regression with Differential Privacy (LR_DP) shows that it retains reasonable classification accuracy even with medium ϵ values. However, providing stronger privacy guarantees is more challenging compared to the GNB_DP model as ϵ decreases, prediction accuracy drastically drops. However, when ϵ increases, the accuracy of LR_DP improves and closely matches that of the non-privacy-preserving LR model at higher ϵ values. This indicates that LR_DP can be adjusted to strike an optimal balance between privacy and utility, depending on the specific needs of the application.

The Decision Tree with Differential Privacy (DCT_DP) model demonstrates strong classification accuracy even at lower ϵ values, which provide stronger privacy guarantees. As ϵ increases, the accuracy improves and approaches that of the scikit-learn GNB model at higher ϵ values. This suggests that with proper tuning of the DCT_DP parameters, the model can maintain good performance at small ϵ values, offering stronger data protection. This result of the DCT_DP makes it one of the strongest models in the library. However, Decision Tree Classifier has a major challenge which is the computational resource and compute time. Comparing the other

two models, both DCT and DCT_DP requires much more time and computational resources for training. Depending on the specific needs, other two models can be better options for a basic dataset.

The sequential feature selection process successfully reduced the dimensionality of the dataset to six key features, significantly improving computational efficiency without compromising model performance. This highlights the importance of feature selection in handling high-dimensional IoT data, ensuring that the most relevant features are utilized for classification.

The injection of Laplacian noise in the DP mechanism effectively obfuscates individual data points, thereby safeguarding privacy. As shown in our experiments, the privacy protection is robust even at lower ϵ values, albeit with a corresponding decrease in classification accuracy. This underscores the potential of differential privacy to offer strong privacy protection in IoT applications where data sensitivity is a paramount concern.

Our findings align with and extend the work of McSherry et al. [9] and Dwork [8] on the application of differential privacy in mechanism design and statistical databases. While previous studies have established the theoretical foundations of differential privacy, our research provides empirical evidence of its practical applicability in IoT network data classification. Moreover, our approach complements the findings of Shahid et al. [2] and Sivanathan et al. [3] by demonstrating the feasibility of maintaining high classification accuracy in IoT traffic analysis through privacy-preserving techniques.

The successful implementation of Gaussian Naïve Bayes, Logistic Regression and Decision Tree with differential privacy has significant implications for IoT security. By ensuring that sensitive IoT data remains confidential, our approach can enhance user trust and adoption of IoT technologies, particularly in privacy-sensitive domains such as healthcare, smart homes, and financial services. Our work contributes to the burgeoning field of privacy-preserving machine learning, offering a practical solution to the dual challenge of data utility and privacy protection. This advancement is particularly

relevant in the context of IoT, where the proliferation of connected devices necessitates robust privacy safeguards.

One of the primary limitations observed is the trade-off between privacy and accuracy. As the ϵ value decreases to enhance privacy, the classification accuracy of GNB_DP, LR_DP and DCT_DP diminishes. This trade-off needs careful consideration based on the specific privacy requirements and acceptable accuracy levels for different IoT applications.

While our approach has shown promising results on the IoTID20 dataset, further research is required to evaluate its scalability and performance across diverse IoT datasets with varying characteristics and complexities.

Future work will explore a comparative analysis of GNB_DP, LR_DP and DCT_DP with other datasets publicly available. This will provide a comprehensive understanding of the strengths and limitations of different approaches in IoT data privacy. Investigating advanced feature selection methods could further improve the models' performance and efficiency. Techniques such as deep learning-based feature selection or hybrid approaches combining multiple feature selection strategies may yield better results. Implementing and testing our approach in real-world IoT environments will be crucial to validate its practicality and effectiveness. This includes evaluating the impact of differential privacy on various IoT applications and user experiences.

5. CONCLUSIONS

Ensuring privacy in IoT networks is one of the most crucial aspects of today's world to protect information about people from being exposed. When the IoT network data is processed and classified, one can learn about the activities and habits of a person, which may cause future harm. We address the data leakage problem with differential privacy that can protect personal information while processing and classifying the network data of an IoT environment. By using the GNB_DP, LR_DP and DCT_DP, we observed that we can still make meaningful inferences to classify data while protecting the individual's privacy. Although there are still several models in diffprivlib, GNB_DP, LR_DP and DCT_DP are very good options within the diffprivlib library.

The integration of these three models with Differential Privacy presents a viable solution to the pressing privacy challenges in IoT networks. By balancing the trade-offs between privacy and accuracy, our approach ensures that sensitive information remains protected while enabling effective data classification. This work not only addresses current privacy concerns but also sets the stage for future advancements in privacy-preserving technologies within the IoT landscape.

This discussion highlights the potential and applicability of our approach, emphasizing its contributions and identifying areas for further research. By continuing to explore and refine these techniques, we can enhance the security and privacy of IoT systems, fostering greater trust and widespread adoption of these transformative technologies.

REFERENCES

[1] K. Rose, S. Eldridge, and L. Chapin, "The internet of things: An overview," *The internet society (ISOC)*, vol. 80, no. 15, pp. 1–53, 2015.

[2] M. R. Shahid, G. Blanc, Z. Zhang, and H. Debar, "IoT devices recognition through network traffic analysis," in *2018 IEEE international conference on big data (big data)*, 2018, pp. 5187–5192.

[3] A. Sivanathan *et al.*, "Characterizing and classifying IoT traffic in smart cities and campuses," in *2017 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, 2017, pp. 559–564.

[4] R. H. Weber, "Internet of Things–New security and privacy challenges," *Computer law & security review*, vol. 26, no. 1, pp. 23–30, 2010.

[5] P. M. Chanal and M. S. Kakkasageri, "Security and privacy in IoT: a survey," *Wirel Pers Commun*, vol. 115, no. 2, pp. 1667–1693, 2020.

[6] H. F. Atlam and G. B. Wills, "IoT security, privacy, safety and ethics," *Digital twin technologies and smart cities*, pp. 123–149, 2020.

[7] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, "Edge computing: Vision and challenges," *IEEE Internet Things J*, vol. 3, no. 5, pp. 637–646, 2016.

[8] C. Dwork, "Differential Privacy," in *Automata, Languages and Programming*, B. and S. V. and W. I. Bugliesi Michele and Preneel, Ed., Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 1–12.

[9] F. McSherry and K. Talwar, "Mechanism design via differential privacy," in *48th Annual IEEE Symposium on Foundations of Computer Science (FOCS'07)*, 2007, pp. 94–103.

[10] I. Ullah and Q. Mahmoud, "A Scheme for Generating a Dataset for Anomalous Activity Detection in IoT Networks," pp. 508–520, Jun. 2020, doi: 10.1007/978-3-030-47358-7_52.

[11] E. Bisong, "Introduction to Scikit-learn," in *Building Machine Learning and Deep Learning Models on Google Cloud Platform: A Comprehensive Guide for Beginners*, Berkeley, CA: Apress, 2019, pp. 215–229. doi: 10.1007/978-1-4842-4470-8_18.

[12] D. Micci-Barreca, "A preprocessing scheme for high-cardinality categorical attributes in classification and prediction problems," *ACM SIGKDD Explorations Newsletter*, vol. 3, no. 1, pp. 27–32, Jul. 2001, doi: 10.1145/507533.507538.

[13] H. Abdi and L. J. Williams, "Principal component analysis," *Wiley Interdiscip Rev Comput Stat*, vol. 2, no. 4, pp. 433–459, 2010.

[14] F. Pedregosa *et al.*, "Scikit-learn: Machine learning in Python," *the Journal of machine Learning research*, vol. 12, pp. 2825–2830, 2011.

[15] H. K. Ekenel and B. Sankur, "Feature selection in the independent component subspace for face recognition," *Pattern Recognit Lett*, vol. 25, no. 12, pp. 1377–1388, Sep. 2004, doi: 10.1016/j.patrec.2004.05.013.

[16] V. N. G. Raju, K. P. Lakshmi, V. M. Jain, A. Kalidindi, and V. Padma, "Study the Influence of Normalization/Transformation process on the Accuracy of Supervised Classification," in *2020 Third International Conference on Smart Systems and Inventive Technology (ICSSIT)*, 2020, pp. 729–735. doi: 10.1109/ICSSIT48917.2020.9214160.

[17] I. H. Sarker, "Machine Learning: Algorithms, Real-World Applications and

Research Directions," *SN Comput. Sci.*, vol. 2, no. 3, Mar. 2021, doi: 10.1007/s42979-021-00592-x.
[18] G. I. Webb, J. R. Boughton, and Z. Wang, "Not So Naive Bayes: Aggregating One-Dependence Estimators," *Mach Learn*, vol. 58, no. 1, pp. 5–24, Jan. 2005, doi: 10.1007/s10994-005-4258-6.
[19] K. P. Murphy, *Machine Learning: A Probabilistic Perspective*. The MIT Press, 2012.
[20] J. Friedman, T. Hastie, and R. Tibshirani, "Additive logistic regression: a statistical view of boosting (With discussion and a rejoinder by the authors)," *The Annals of Statistics*, vol. 28, no. 2, pp. 337 – 407, 2000, doi: 10.1214/aos/1016218223.
[21] Y.-Y. Song and Y. Lu, "Decision tree methods: applications for classification and prediction," *Shanghai Arch Psychiatry*, vol. 27, pp. 130–135, Sep. 2015, doi: 10.11919/j.issn.1002-0829.215044.
[22] S. R. Safavian and D. Landgrebe, "A survey of decision tree classifier methodology," *IEEE Trans Syst Man Cybern*, vol. 21, no. 3, pp. 660–674, 1991, doi: 10.1109/21.97458.
[23] T. Dalenius, "Towards a methodology for statistical disclosure control," *Statistik Tidskrift*, vol. 15, no. 429–444, pp. 1–2, 1977.
[24] A. Wood *et al.*, "Differential Privacy: A Primer for a Non-Technical Audience," *SSRN Electronic Journal*, 2018, doi: 10.2139/ssrn.3338027.
[25] N. Holohan, S. Braghin, M. A. Pól, and K. Levacher, "Diffprivlib: The IBM Differential Privacy Library," Jul. 2019. [Online]. Available: https://arxiv.org/abs/1907.02444

CURRICULUM VITAE

Onat Can Baba received the B.S. degree in Electronics and Communication Engineering from Yıldız Technical University, İstanbul, Turkey, in 2018. He is pursuing the M.S. degree in Electrical-Electronics Engineering at Marmara University, İstanbul, Turkey. He is currently working as Network Architect at Nokia, İstanbul, Turkey. His research interests are machine learning and network security.