Performance Analysis of Intrusion Detection Systems Using the PyCaret Machine Learning Library on the UNSW-NB15 Dataset

by

Abdullah
16101305
Faisal Bin Iqbal
17101339
Srijon Biswas
21141053
Rubabatul Urba
17101426

A thesis submitted to the Department of Computer Science and Engineering
in partial fulfillment of the requirements for the degree of
B.Sc. in Computer Science

Department of Computer Science and Engineering
Brac University
June 2021

Declaration

It is hereby declared that

1. The thesis submitted is my/our own original work while completing degree at Brac University.
2. The thesis does not contain material previously published or written by a third party, except where this is appropriately cited through full and accurate referencing.
3. The thesis does not contain material which has been accepted, or submitted, for any other degree or diploma at a university or other institution.
4. I/We have acknowledged all main sources of help.

Student's Full Name & Signature:

Abdullah

Abdullah
16101305

faisal

Faisal Bin Iqbal
17101339

Srijon

Srijon Biswas
21141053

Rurba

Rubabatul Urba
17101426

Approval

The thesis/project titled "Performance Analysis of Intrusion Detection Systems Using the PyCaret Machine Learning Library on the UNSW-NB15 Dataset" submitted by

1. Abdullah (16101305)
2. Faisal Bin Iqbal (17101339)
3. Rubabatul Urba (17101426)
4. Srijon Biswas (21141053)

Of Spring, 2021 has been accepted as satisfactory in partial fulfillment of the requirement for the degree of B.Sc. in Computer Science on June 6, 2021.

Examining Committee:

Supervisor:
(Member)

[illegible]

Dr. Amitabha Chakrabarty
Associate Professor
Department of Computer Science and Engineering
Brac University

Head of Department:
(Chair)

Sadia Hamid Kazi
Chairperson
Department of Computer Science and Engineering
Brac University

Abstract

As one of the fastest growing technologies on earth, the Internet of Things (IoT) is being embraced almost everywhere. From smart home technology to industrial automation, IoT is revolutionizing almost everything around us. It has enabled humans and organizations to do more with less, both in terms of time, as well as finances. This feat of the Internet of Things, however, has also led to an alarming rise in attacks on IoT networks. Among these attacks, botnet intrusions are perhaps the most worrying ones. And with the advancement of time and technology, attackers are getting more creative. Hence, it is important to use better and more efficient machine learning technologies to identify these attacks and detect these intrusions before they can paralyze the system. This research aims to identify a more efficient machine learning approach for detecting botnets in IoT networks by utilizing the PyCaret machine learning library and analyzing its overall performance. The research will encompass different classifiers and analyze the different performance metrics for each of them. It will also shed light on the feasibility of using the PyCaret library and how well suited it is for such usage.

Keywords: IoT (Internet of Things), Machine Learning, Anomaly detection, Intrusion detection, PyCaret, UNSW-NB15.

Acknowledgement

Firstly, we would like to express our heartfelt gratitude to the Almighty for giving us strength, and enabling us to successfully complete our thesis during the troubling times of a global pandemic.
Secondly, to our advisor, Dr. Amitabha Chakrabarty sir (Associate Professor, Department of Computer Science and Engineering, Brac University) for his timely and insightful support. He was always there to support, encourage, and motivate us. He played a vital role in the completion of our research, and for that, we will eternally be grateful to him.
Thirdly, to Dr. Md. Golam Robiul Alam sir (Associate Professor, Department of Computer Science and Engineering, Brac University) for providing us with all the necessary guidelines and instructions regarding thesis, and helping us with all related queries.
Fourthly, to Ayesha Abed Library for providing us with adequate thesis help and plagiarism report.
Then to the PyCaret developer's community on GitHub for assisting us with all relevant queries.
And finally to our parents and loved ones without whom this would not have been possible. Their constant support, both mentally as well as financially, has helped us achieve a great deal in life, and we will forever be grateful to them.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 What is IoT?

The Internet of Things – abbreviated as IoT – is a technology that supports the concept of establishing an inter-connection of devices within a set network. As the name suggests, IoT teases the idea of forming an 'internet' like network where instead of people communicating with one-another, there is a plethora of interconnected devices. These connections can be wired as well as wireless. However, given the complexity of these networks, more priority is being given to wireless networks over wired ones when it comes to establishing IoT systems [18].

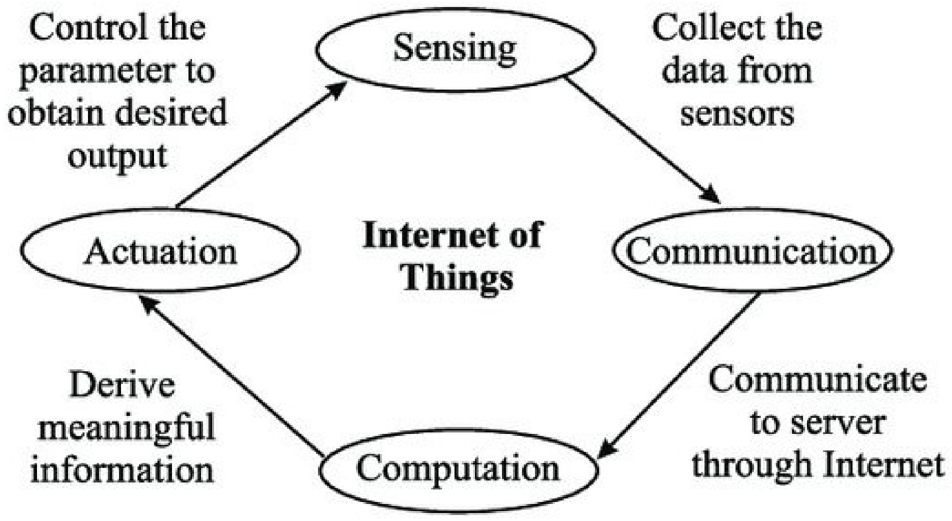
Control the parameter to obtain desired output
Sensing
Collect the data from sensors
Actuation
Internet of Things
Communication
Derive meaningful information
Communicate to server through Internet
Computation

Figure 1.1: *Working Principle of an IoT System [37]*

The aim of IoT is to introduce a virtual footprint of all available electronic devices, and the people who are connected to these devices [12]. In a more generalized context, the internet of things aims to encompass everything that can be connected to a network, and utilize them in a faster and more efficient way.

1.1.1 How does Iot work?

A typical IoT-based system consists of a central processing unit, one or more sensors, and actuators. Alongside all these, there are different communication protocols [18] involved in the passing of instructions and data between the various nodes, which consists of sensors and actuators, and the processing unit. And instead of using a local processing unit, the IoT system can utilize cloud-based processing systems as well.
The sensors are used to perceive real world information and deliver that data to the processing unit. The processing unit, in turn, analyzes this data and authenticates necessary actions. These actions are then carried out by the actuators. In some systems, the actions are carried out by the processing unit itself.

1.1.2 Applications of IoT

When the concept of the Internet of Things was initially proposed by the MIT Auto-ID center in 1999 [3], no one, in their wildest imaginations, thought that by 2019, it would turn into an industry worth 250 billion US dollars [40]. And by 2027, it is expected that the IoT industry will be worth almost 1500 billion US dollars [40].

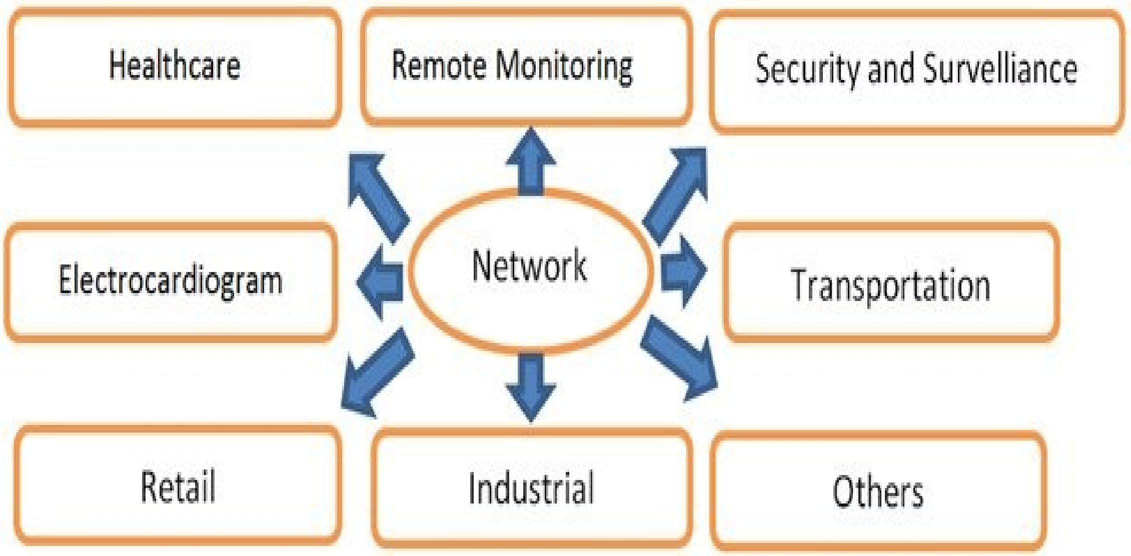

Figure 1.2: *A Simplified Diagram Showing the Various Applications of the Internet of Things [17]*

IoT technology is now being utilized almost everywhere. The healthcare sector is seen working their way around different wearable IoT (WIoT) devices to monitor their patients' health remotely [5]. Mass production and utilization of such wearable devices will mean that doctors or healthcare professionals no longer need to be present on-spot in order to monitor the patient's health condition.
A similar technology is utilized in the agricultural sector, where an IoT infrastructure is established to monitor the farmlands and livestock, and gather vital information about them constantly [2]. Apart from all this, the Internet of Things is being utilized in various other sectors as well, including different businesses, factories,

transportation systems, power and energy sector, and so on. With time, we will be introduced to more diverse uses of IoT and will have seen advances made in its existing sectors.

1.1.3 Challenges in IoT

Like any other technology, the Internet of Things is also prone to different challenges and problems. As far as technical challenges go, IoT faces six major obstacles - issues related to security and integrity, connectivity related problems, longevity and compatibility, proper standards, and intelligent analysis and actions [13]. Apart from technical challenges, the Internet of Things is also subjected to challenges posed by its usage in different business and social purposes [13]
Among all these challenges, issues related to network and device security and privacy are always prioritized over everything else.

1.1.4 Security threats in Iot

Potential Botnet attack configuration, DDoS attacks, data manipulation over an unencrypted way of data transmission from system to cloud, malware infections are all the threats IoT still needs to tend to. Primarily security threats are the biggest concerns of the growing IoT network.

1.2 What is a Botnet?

A botnet can be best described as the concept of a computer network manipulated for a targeted task. A device's security is breached leading the hacker to manipulate other connected devices through the infected device to further infect more devices leading to a massive interconnection controlled by the hacker without alarming the owner of the devices. In short, botnets carry out activities automatically and make them appear legitimate [1].

1.2.1 How does a Botnet attack?

The development of a botnet can be explained in as a two phase [1] procedure. The first of these two phases is the creation phase, where an attacker or individual creates a botnet or extends an existing one by writing lines of code. The second stage involves the propagation of these botnets, mostly through backdoors and exploiting possible network vulnerabilities.

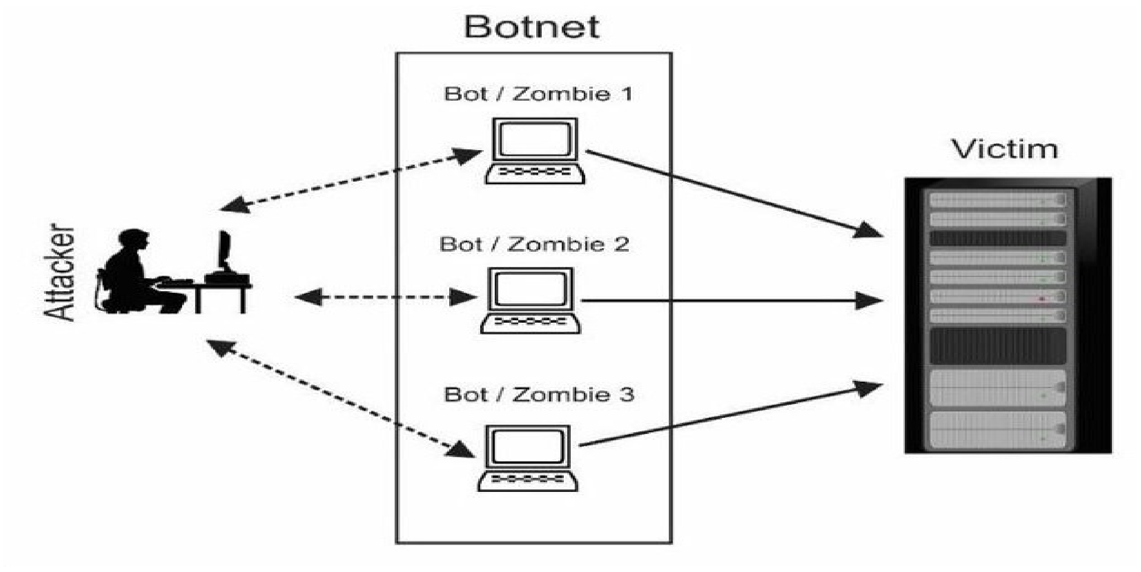

Figure 1.3: *A Typical Botnet Attack Structure [11]*

1.2.2 Effects of Botnet attack

A botnet attack can cripple entire IT infrastructures. They can take control of end devices as well as entire networks. Botnets can launch DDoS attacks, steal personal information, generate spam emails, and so on. The destructive feats of botnets have made it into a cyber-weapon that can be used in cyber-warfare [41].

1.3 Intrusion Detection Systems (IDS)

An intrusion detection system (IDS) is a monitoring system over a suspicious intervention of traffic. It can be a software or a device that detects abnormal intrusions or anomalies using different methods (host/network based). In both host (working device) and network (current operating network) devices, an IDS uses detection methods like Signature based, like using fingerprints of known threats and matching them with incoming ones, anomaly based, like using a pattern or defining a standard for “normal” behavior for the particular device, or even a hybrid one. An IDS is an extra line of defense tackling the threats that have slipped passed the initial line of defense.

1.4 Introduction to the PyCaret Machine Learning Library

Built around the Python programming language, PyCaret is an open-source machine learning library [30]. It is a low-code library, meaning that a user will require significantly less lines of code to run a machine learning program. It eliminates the need to produce hundreds of lines of code through its low-code alternatives.

PyCaret utilizes several machine learning libraries and frameworks including scikit-learn, XGBoost, and CatBoost. Here, the PyCaret library is technically working as a wrapper library consisting of these popular machine learning libraries.
One of the main aims of the PyCaret library is to establish the idea of 'democratizing machine learning", which means that it enables business analysts, data scientists, and everyone else, who may not have a computer science or tech background, to fully utilize these technologies for their personal as well as professional needs. Using PyCaret allows these users to create ML models more easily and with less effort. A single line of code is all it takes for the library to provide its users with different analyses of the data through the variety of models and modules under its scope.
Another thing that makes PyCaret suitable for a large user base is its ability to handle different parameters. Unlike the traditional way of fitting any ML model, PyCaret does everything for its users. It passes a list of parameters during the fitting by utilizing its internal methods. That way, users need only to evaluate the model, which is also made easy by this library, and compare the results with traditional methods.

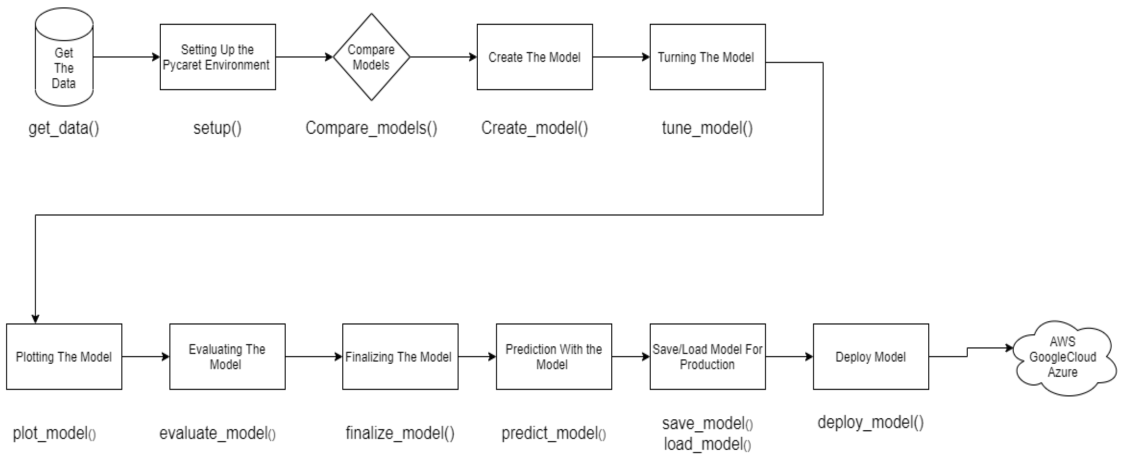

Figure 1.4: *Workflow diagram of PyCaret*

Figure 1.4 demonstrates the step-by-step workflow of PyCaret. Here, The setup() function does all of the data preparation. This all done in single function instead of spreading the work into a number of different, hard to remember function calls. The compare_models() function runs a benchmark against all of the applicable algorithms and returns performance data. Functions such as evaluate_model() and interpret_model() return easy to use interfaces for developing a deeper understanding of your model. Finally, The deploy_model() function allows the user a process for deploying models in AWS, Google Cloud or Azure[38].
The main reason our research is centered on this library is to see its effectiveness. It is to see how well it performs and compares to existing researches conducted on the same set of data but with a more traditional approach. We are eager to understand the capabilities as well as the limitations of this library and whether or not is self-sufficient for real-life deployment on various IoT based cloud platforms.

Chapter 2

Aims and Objectives

Our research aims to carry out a performance analysis on the capabilities of the PyCaret machine learning library – an open source and low-code ML library based on the Python programming language. It aims to do so by running binary classifications using eight different classification models, and additional blending (ensemble) techniques, on the "UNSW-NB15 dataset". The machine learning program is to function as an Intrusion Detection System (IDS) that detects possible intrusions or botnet attacks within an IoT-based network.
The objectives of this research are discussed below in brief –
• To explore the aforementioned dataset.
• To explore the PyCaret library.
• To understand the functional capabilities of the PyCaret machine learning library.
• To utilize the PyCaret library and detect intrusions in the dataset using multiple classification models.
• To retrieve the detection/classification results and analysis the overall performance of the PyCaret library on the dataset.
• To understand whether or not the PyCaret library is efficient in providing security to IoT networks against potential threats while allowing users to prepare the model for deployment with ease.
• To observe how efficient a low-code machine learning actually is in delivering an effective solution to real life security threats that an IoT system might encounter.

Chapter 3

Literature Review

3.1 The Internet of Things

The Internet of Things has revolutionized the way we connect with inanimate objects. IoT has eliminated the need for a physical interface between humans and electronic devices or other inanimate objects, and replaced it with sensors and live data feed. Physical storage devices are also being replaced with cloud-based systems and storage facilities. By 2023, 85-90% of IoT data will be stored in the cloud [29]. On top of that, it is being predicted that by 2025, there will be over 75 billion connected devices all over the world [29].

Thanks to its scalability, and availability of sensors, IoT technology is being deployed almost everywhere one can look. At present, it has widespread use, and covers various diverse fields of our everyday lives.

The Internet of Things has had a huge impact on healthcare. Thanks to IoT based implementations, there has been a lot of developments in the field of smart healthcare, and more opportunities are being explored [14]. Over the years, research efforts have been put into the development of IoT based wearable devices for patients using sensors to monitor parameters in order to analyse the health condition [36]. Remote health monitoring system that uses deep learning in order to diagnose heart diseases has been proposed [34]. In addition, effective and feasible approaches to manage hospital data using IoT in order to make it useful for proper health analysis has been studied [21].

Another field that has seen significant increase in the use of IoT is the industry of smart homes. Smart homes can grant the user control to devices and appliances in their homes in order to provide convenience and boost security using the internet. IoT-based low-cost smart home systems and prototypes that can provide users security and convenience while also ensuring reduced energy consumption has already been proposed [24].

Given its vast areas of implementations, it is important to secure IoT networks and devices. Hence, security is of great concern when it comes to IoT technology.

Security challenges in IoT can be divided into two groups- technological challenges, and security challenges [7]. Technological challenges are mostly related to the IoT devices and their natures. Security challenges, on the other hand, are mostly related to the network and different working principles of entire system, as well as intervention from foreign entities outside the network's user base [7].

3.2 Botnets

Botnets have the capability to cripple large networks and systems. They also pose a great threat to IoT-based systems. Over the past decade, cyber-attacks on large infrastructures by IoT botnets have made it clear that no IoT or network-based system is fully safe and secure from these intrusions [15]. Botnets have the ability to launch Distributed Denial-of-Service (DDoS) attacks which can overwhelm IoT infrastructures. It can also be used to create a backdoor to the network in order to provide access to sensitive data and devices in the network. The only way to tackle these attacks is by monitoring these networks 24x7, identifying the attacks, and keeping them away from interfering with the network's operations.

3.3 Machine Learning and Intrusion Detection

Over the years, researchers have looked into the possibilities of detecting and isolating botnets using machine learning technologies [19]. There have been numerous efforts to retrofit existing machine learning models for identifying botnets. Efforts have also been taken to develop models from scratch where data packets and requests are assigned threshold values to check whether an access request to the network is legitimate or an attack [19].
Researchers have been more inclined towards supervised machine learning models in detecting intrusions [6]. The aim is to train the models into identifying certain traits and signs, and classify them as threats or botnets. These papers reflect on both binary classification, where the model is able to detect the nature of the packet and declare whether it is a normal packet request or an anomaly, as well as multi-classification, where the models are not only able to declare the nature of the request, but also able to classify the type of attack, if that is the case.
Despite the numerous researches conducted in the past, it is important to carry on with the on-going efforts of developing newer and faster machine learning techniques that are not only more effective, but can also be developed and deployed with ease.

3.4 PyCaret Machine Learning Library

The PyCaret library is a low-code autoML framework that can be used for classification as well as prediction purposes. It is an open-source machine learning library made publicly available in April of 2020. PyCaret requires significantly less lines of code to run machine learning models that can then be easily deployed to cloud services. It comes with its own modules much of which use existing wrappers and frameworks like Scikit-learn, and is able to provide users with an interactive dashboard for a clearer understanding of the analysis and results [39].
Given that it is comparatively new, it remains to be seen how the PyCaret library will perform in terms of botnet detection.

Chapter 4

Dataset Description

4.1 Dataset Description

The dataset to be used for this research is the "UNSW-NB 15 dataset" by the University of New South Wales [10]. It consists of real-life modern normal scenarios as well as contemporary synthesized attack activities of network traffic [10].

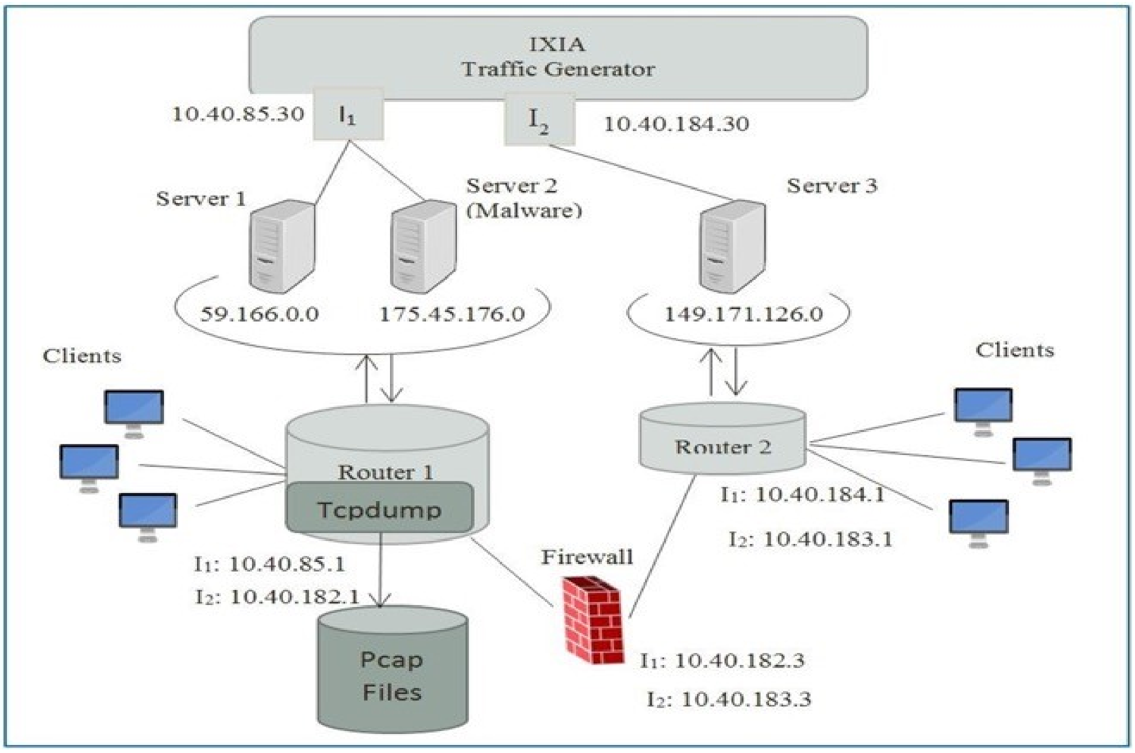

Figure 4.1: *A Brief Overview on how the "UNSW-NB 15 dataset" was generated using the "IXIA PerfectStorm tool" [10]*

The dataset contains raw packets that were originally created by the "IXIA PerfectStorm tool" at the "Cyber Range Lab of the Australian Center for Cyber Security (ACCS)" [10]. It has over two million data records and 49 features which also include the 2 class labels.

The features included in the dataset are grouped into 6 different categories [8] namely "flow features", "basic features", "content features", "time features", "additional generated features", and "labelled features".

The class labels in the dataset are attack category, and label. The attack category is again of 9 types [8]. There is an additional type referred to as 'Normal' which represents a safe or normal scenario within the network. Label is categorized into 2 types – 0, representing a normal scenario within the network, and 1, representing an intrusion or attack [8].

For our research, out of the 2 million records, we will be using a total of 257,673 records divided into training and testing sets in a distribution of 2:1. Out of the 257,673 records, 175,341 fall in the training set, and the other 82,332 records fall in the testing set. It is to be noted that the CSV versions of these training and testing sets, as well as the original two million records, feature lists, and other source files have been made readily available for public use at this link [9].

And while there are other exiting datasets that can be used for intrusion detections systems and related analyses, the UNSW-NB 15 is a more modern and up-to-date dataset that overcomes the limitations of the older datasets [10].

4.2 Exploratory Data Analysis

Out of the total number of records in the training set, 119,341 are categorized as label 1, while the rest of the 56,000 are categorized as label 0. For the test set, there are 45,332 records labeled 1, while the rest of the 37,000 are labeled as 0.

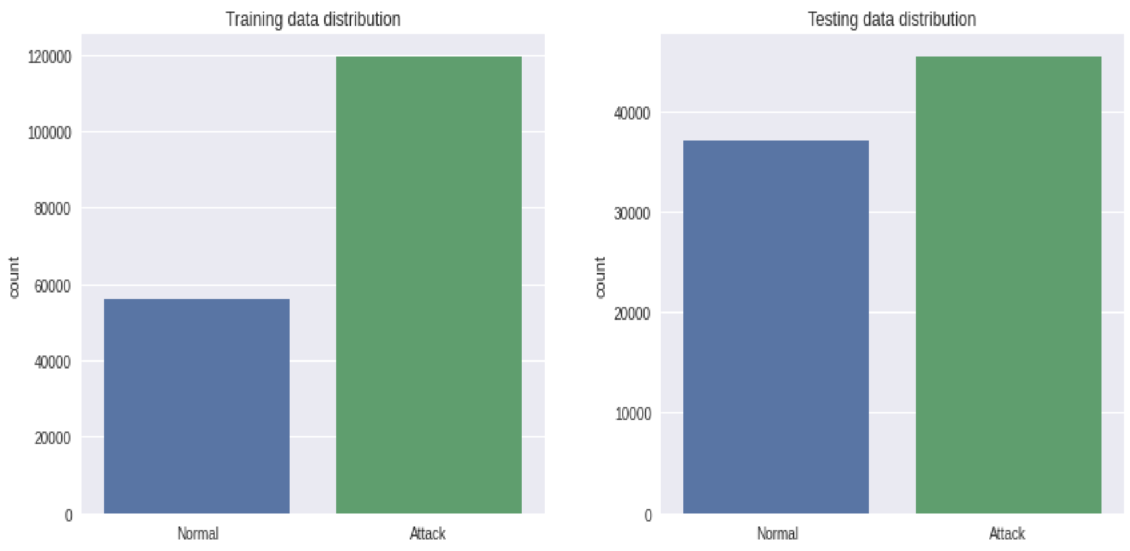

Figure 4.2: *Label Distribution in Training and Testing Datasets*

For the attack_cat feature, the testing set contains 32% records labeled normal, 23%

labeled generic, and the rest cover the remaining 45%. The training set, on the other hand, has 45% labeled as normal, 23% generic, and the rest covers the remaining 32%.

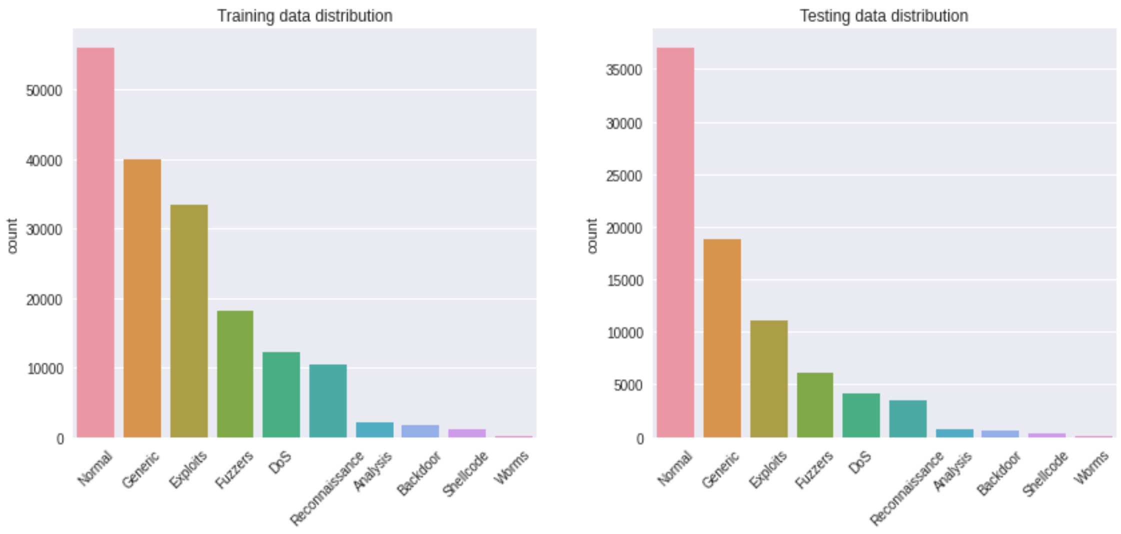

Figure 4.3: *Attack Category Distribution in Training and Testing Datasets*

Table 4.1 breaks down the different types of attack categories mentioned in the attack_cat feature, and their overall distribution in the training and testing subsets.

Class	**Training Set**	**Testing Set**
Normal	56,000	37,000
Analysis	2,000	677
Backdoor	1,746	586
DoS	12,264	4,089
Exploits	33,393	11,132
Fuzzers	18,184	6,062
Generic	40,000	18,871
Reconnaissance	10,491	3,496
Shellcode	1,133	378
Worms	130	44
Total	175,341	82,332

Table 4.1: Different Classes of Attacks and Their Distribution in Training and Testing Subsets

Upon evaluating the correlations, the following cases, described in table 4.2, were observed for a threshold of 0.9.

Features	**Correlation Value**
dpkts, dloss	0.98
sbytes, sloss	1.0
dbytes, dloss	1.0
sinpkt, is_sm_ips_ports	0.94
tcprtt, synack	0.95
tcprtt, ackdat	0.94
ct_srv_src, ct_dst_src_ltm	0.97
ct_srv_src, ct_srv_dst	0.98
ct_dst_ltm, ct_src_dport_ltm	0.96
ct_src_dport_ltm, ct_dst_sport_ltm	0.91
ct_src_dport_ltm, ct_src_ltm	0.9
ct_dst_src_ltm, ct_srv_dst	0.97

Table 4.2: Correlation between different features

4.3 Data Pre-Processing

As a library that is constantly ready for deployment purposes, the PyCaret library forms a pipeline consisting of all necessary blocks of functions or modules that can simplify the model training process. This includes the data pre-processing phase as well. Data pre-processing as well as data preparation related functions are a major component of the library. As a result of this, the PyCaret library is able to handle these functions automatically. All the user needs to do is set things up by calling the 'setup' function.
For dealing with values that are missing in the dataset, PyCaret, by default, utilizes the mean value of the feature in the case of numeric features. A median value can also be selected by the user manually if needed. For categorical features, a default 'not_available' feature is set for missing cases. The other option is the mode value. For normalization, PyCaret, by default, uses 'zscore'. And for transformation, it uses 'yeo-johnson' method by default.

Chapter 5

Research Methodology

5.1 Supervised Learning

In order to train the models, the research takes a supervised learning approach. The models are trained on the training subset of the dataset. The prediction capabilities of the model are then analysed on the testing subset, which contains records previously unseen by the models.
To analyze the performance, the research takes into account the model's training and testing accuracy, time, confusion matrix, precision, F1 score, and recall. This is then followed by a comparative analysis with relevant research that have followed a similar approach, or have presented results that deal with these metrics and datasets.

5.2 Classification Models

5.2.1 Logistic Regression

This model here is a mathematical one that is used for describing the chances of an event occurring, and hence the result is binary, 1 indicating true or happening or 0 meaning the event does not occur. However the model uses a sigmoid function to operate and give out the decisions. The sigmoid function can be calculated by the following formula:

$$Y = \frac{\epsilon^{b_0+b_1\times X}}{1+\epsilon^{b_0+b_1\times X}} \tag{5.1}$$

Here, b_1, b_0 are variables, called weights trained from the dataset, b_0 represents the bias, or intercept, and b_1 is the coefficient. The product of this formula will produce a percentage, or probability, that will be mapped over discrete classes [23].

To calculate logistic regression, we use the following formula:

$$log\,\frac{p}{1-p} \tag{5.2}$$

Where p is the probability of an event occurring.

The confusion matrix and classification report obtained through our research for the above model are provided below.

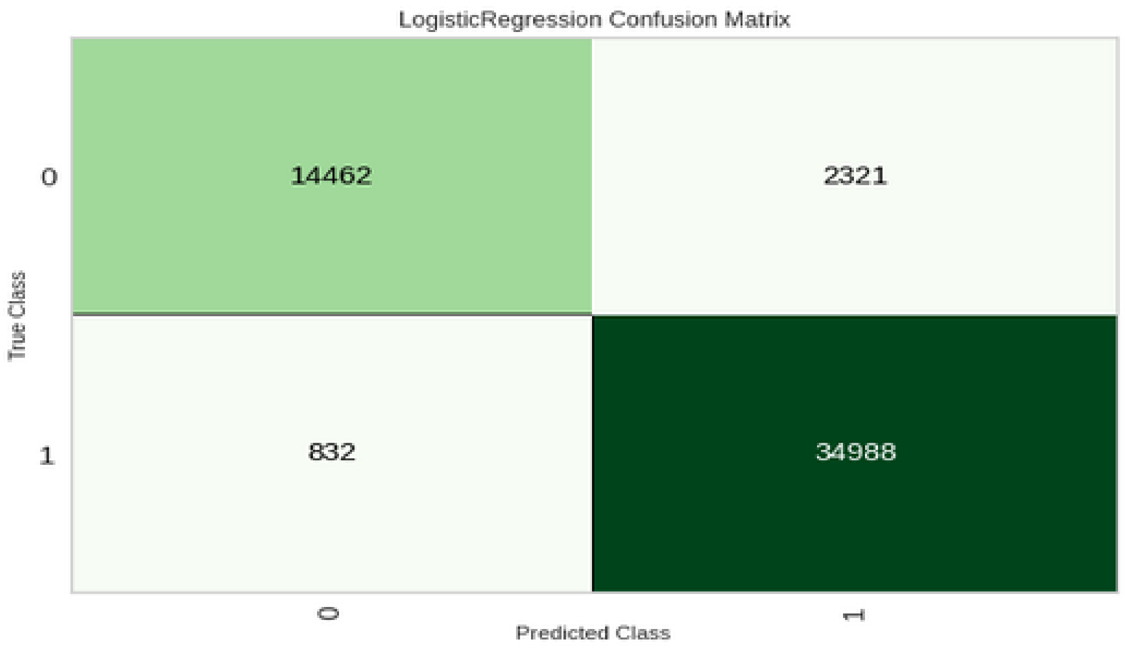

Figure 5.1: *Confusion Matrix for Logistic Regression*

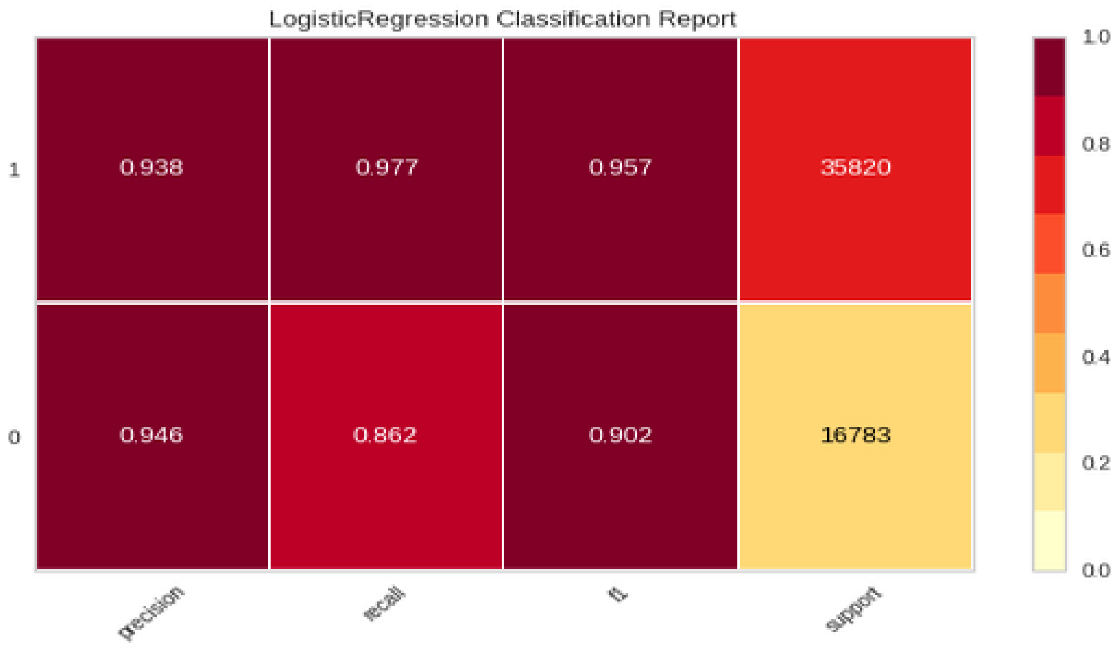

Figure 5.2: *Classification report for Logistic Regression*

5.2.2 K-Nearest Neighbor

KNN is another supervised machine learning algorithm used for classification or regression models or for search. This algorithm uses the concept of proximity to pass out decisions. Similar things exist in near proximity is what KNN promotes. It depends on labelled data taken as input to learn a function and then produces an appropriate output when a new unlabeled data is given. With different values of K, we choose the K that reduces the number of errors we encounter while maintaining the algorithm's ability to accurately make predictions when it is given data it has not seen before. It is again a non-parametric machine learning algorithm [25].

The confusion matrix and classification report obtained through our research for the above model are provided below.

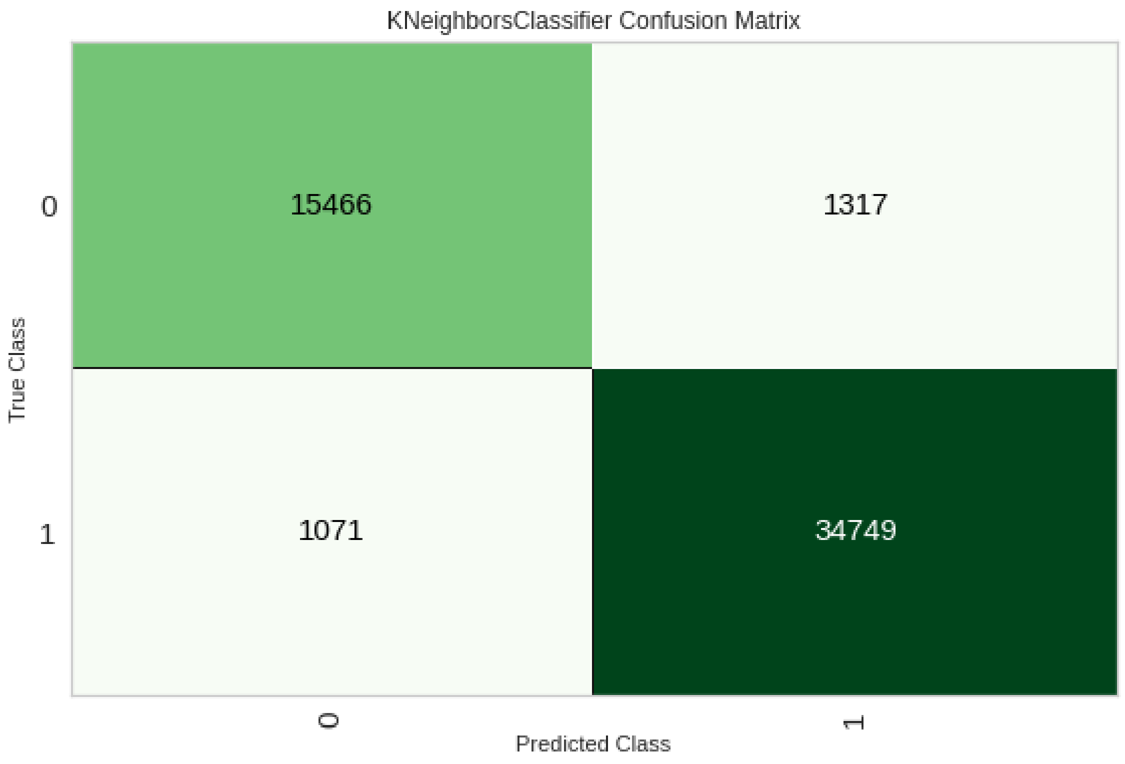

Figure 5.3: *Confusion Matrix for KNN*

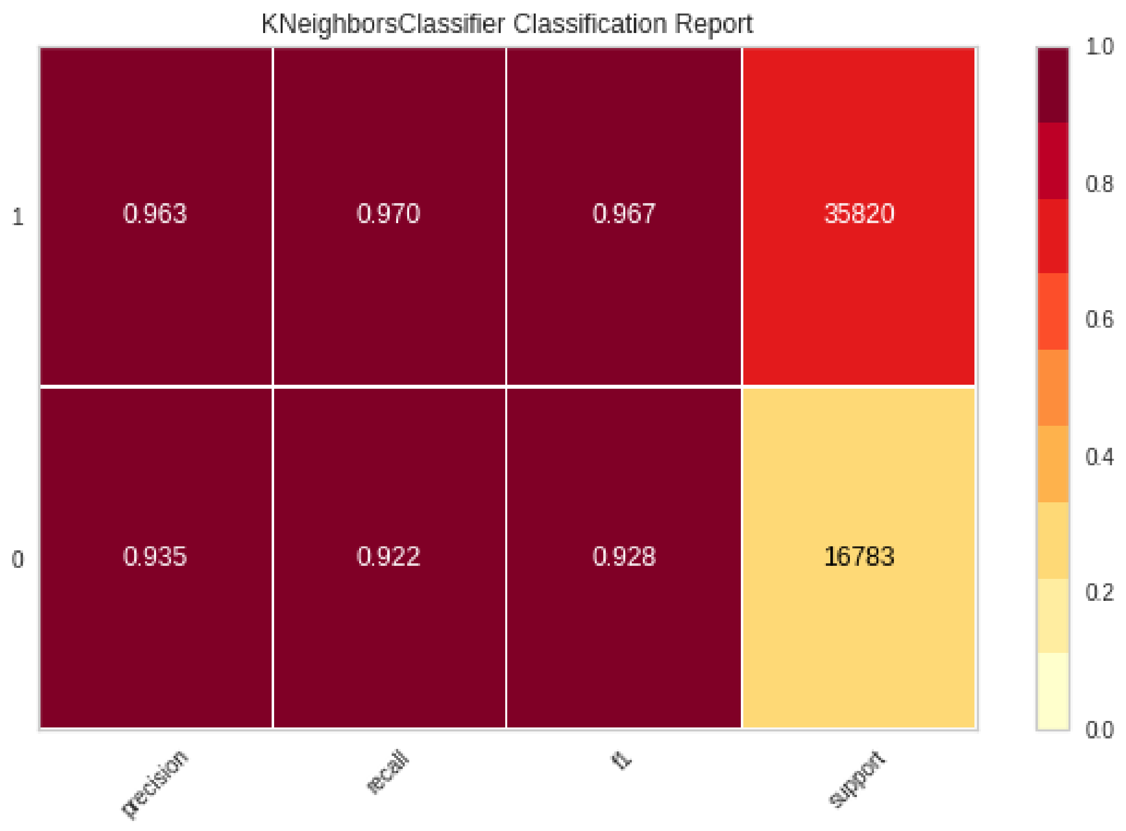

Figure 5.4: *Classification report for KNN*

5.2.3 Decision Tree

Another popular machine learning algorithm for classification and regression is this decision tree model which operates by creating trees based on decisions. To create a decision tree, we choose the Decision Tree template and enter information in the knowledge article fields. The goal is to create a model that predicts the value of a target variable by learning simple decision rules inferred from the data features. Considering the entire data as root it then on particular condition, starts splitting by means of branches or internal nodes and makes a decision until it produces the outcome as a leaf. The entropy in this model would be the impurity present in the data and is calculated by the following equation:

$$Entropy(S) = \sum_{i=1}^{c} -p(i)logp(i) \tag{5.3}$$

Here, c is the number of classes of an attribute. 'p(i)' is the fraction of examples of the class 'i'. The entropy is almost zero when the sample attains homogeneity but is one when it is equally divided. To lower this entropy another parameter has to increase known as 'information gain'. The following formula is used to calculate gain:

$$Gain(S, A) = Entropy(S) - Entropy(A) \tag{5.4}$$

Here S is the parent or root while A is an attribute that we want to split [31].

The confusion matrix and classification report obtained through our research for the above model are provided below.

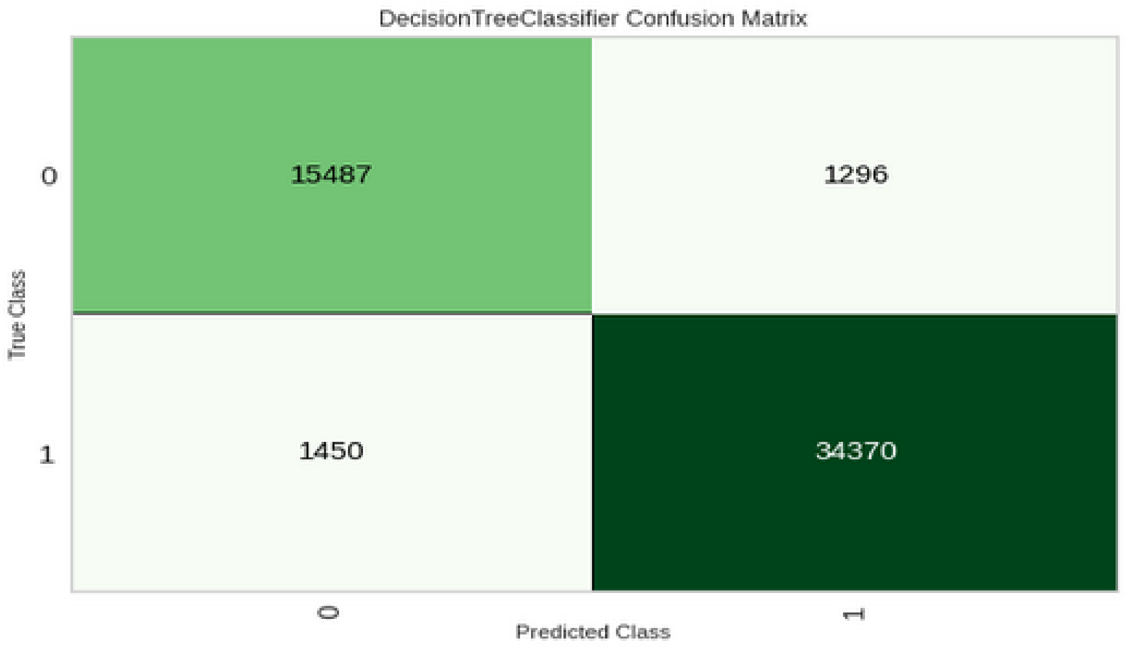

Figure 5.5: *Confusion Matrix for Decision Tree*

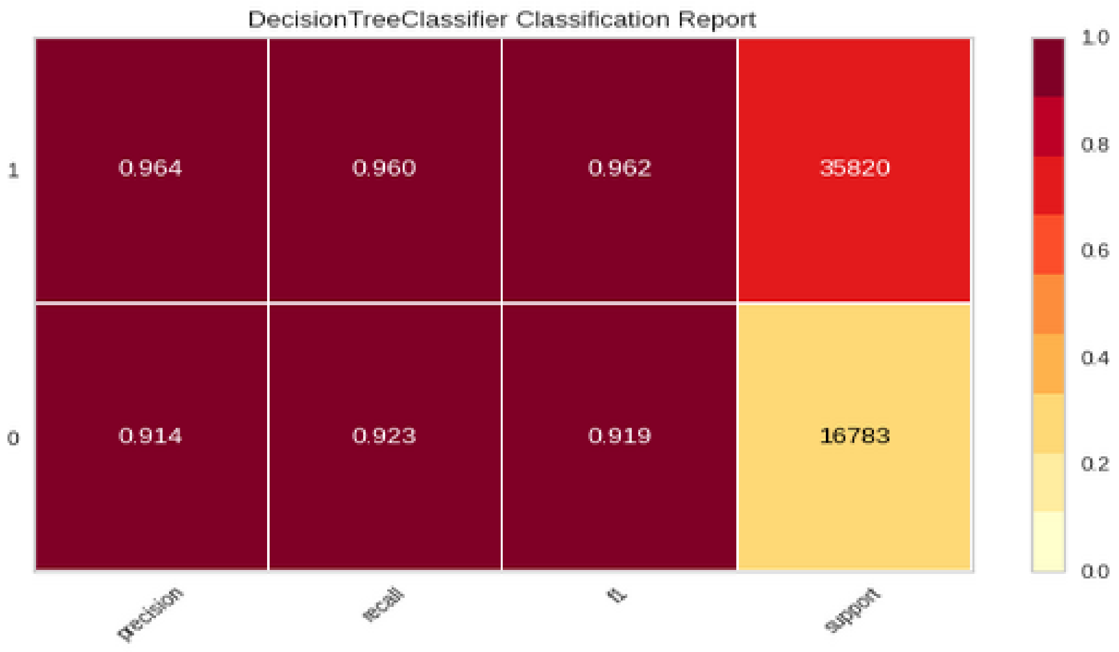

Figure 5.6: *Classification report for Decision Tree*

5.2.4 Random Forest

Another powerful supervised machine learning algorithm for classification, regression operates through building an ensemble of decision trees. Random forest builds multiple decision trees and merges them together to get a more accurate and stable prediction. It can be said that a random forest is a collection of multiple decision trees [28].
When using the Random Forest Algorithm to solve regression problems, the following formula is considered:

$$R_f = \frac{1}{N}\sum_{i=1}^{N}(fi - yi)^2 \tag{5.5}$$

Here, n is the number of data points, fi= value returned by the model and yi = actual value for data point i. While performing Random Forests based on classification data:

$$Gini = 1 - \sum_{i=1}^{C}(Pi)^2 \tag{5.6}$$

Most of the time Gini index is used to calculate the classification model, where Pi represents the relative frequency of the class we are observing in the dataset and c represents the number of classes [27].

The confusion matrix and classification report obtained through our research for the above model are provided below.

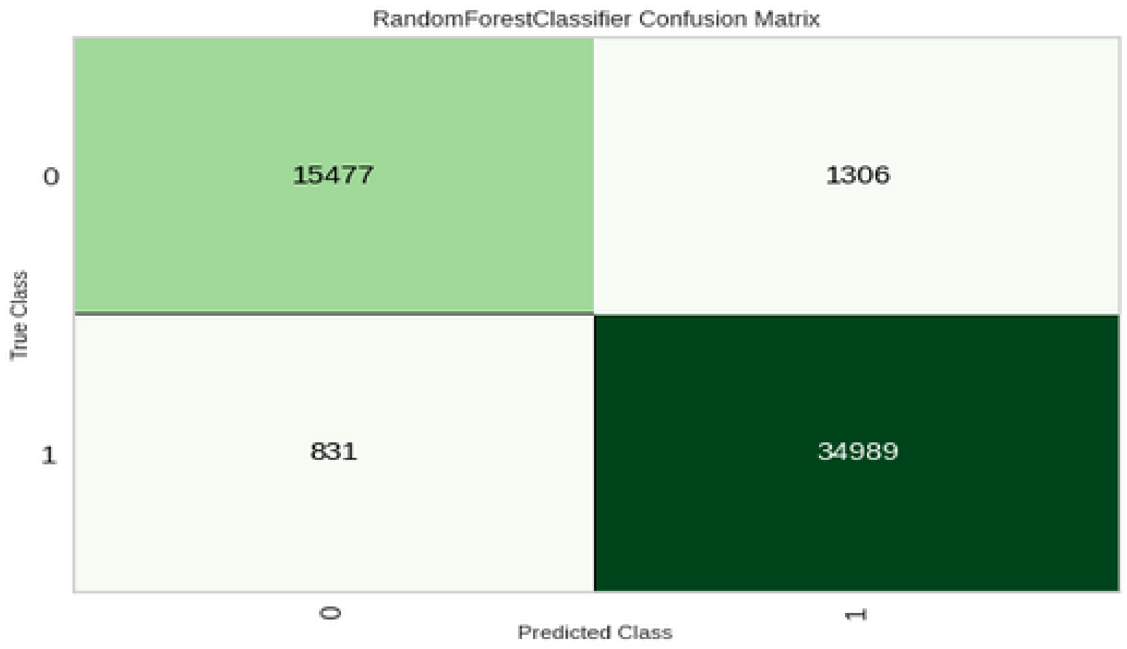

Figure 5.7: *Confusion Matrix for Random Forest*

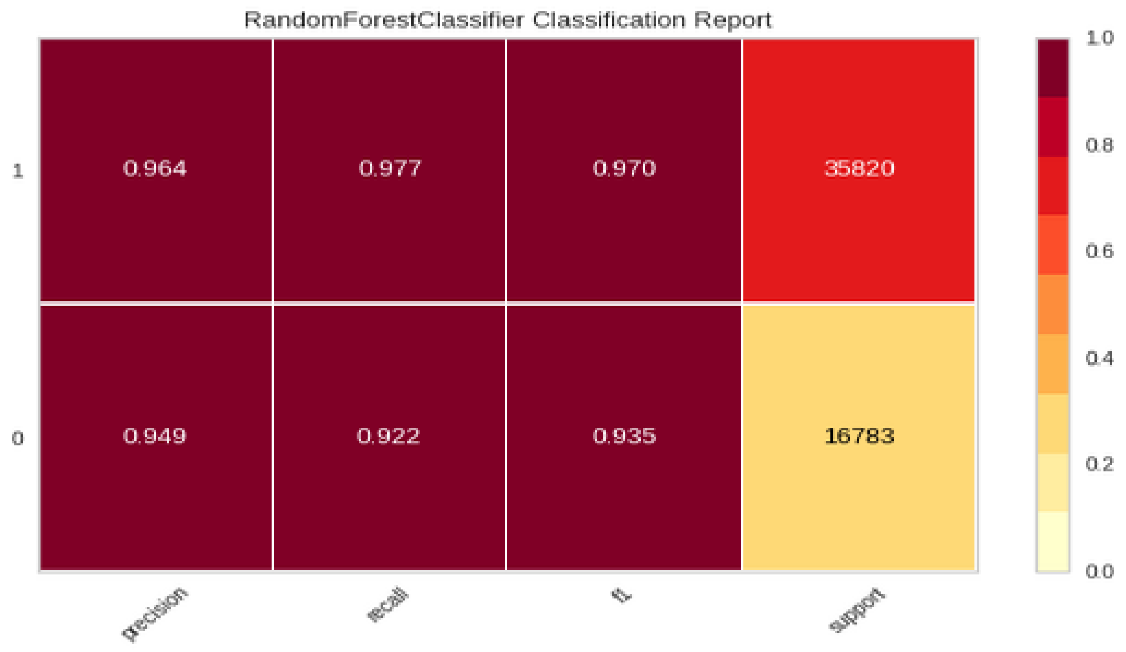

Figure 5.8: *Classification report for Random Forest*

5.2.5 Gradient Boosting Classifier

Another machine learning algorithm which accumulates multiple low predicting learning models resulting in a strong prediction. It is used for both classification and regression models and can work on complex datasets. The key idea is to set the target outcomes for the next model in order to minimize the error. It relies on the intuition that the best possible next model, when combined with previous models, minimizes the overall prediction error. The target outcomes of each case is based on the gradient of errors with respect to the prediction.
The gradient boosting algorithm (gbc) depends upon another function known as the loss function. The loss function can be represented as:

$$\psi(y, f(x)) \tag{5.7}$$

GBC works through minimizing the loss function over a given dataset. Our estimate function:

$$\hat{f}(x) = \arg \min_{f(x)} \psi(y, f(x)) \tag{5.8}$$

This function asks for a minimized loss function for increased functionality [4].

The confusion matrix and classification report obtained through our research for the above model are provided below.

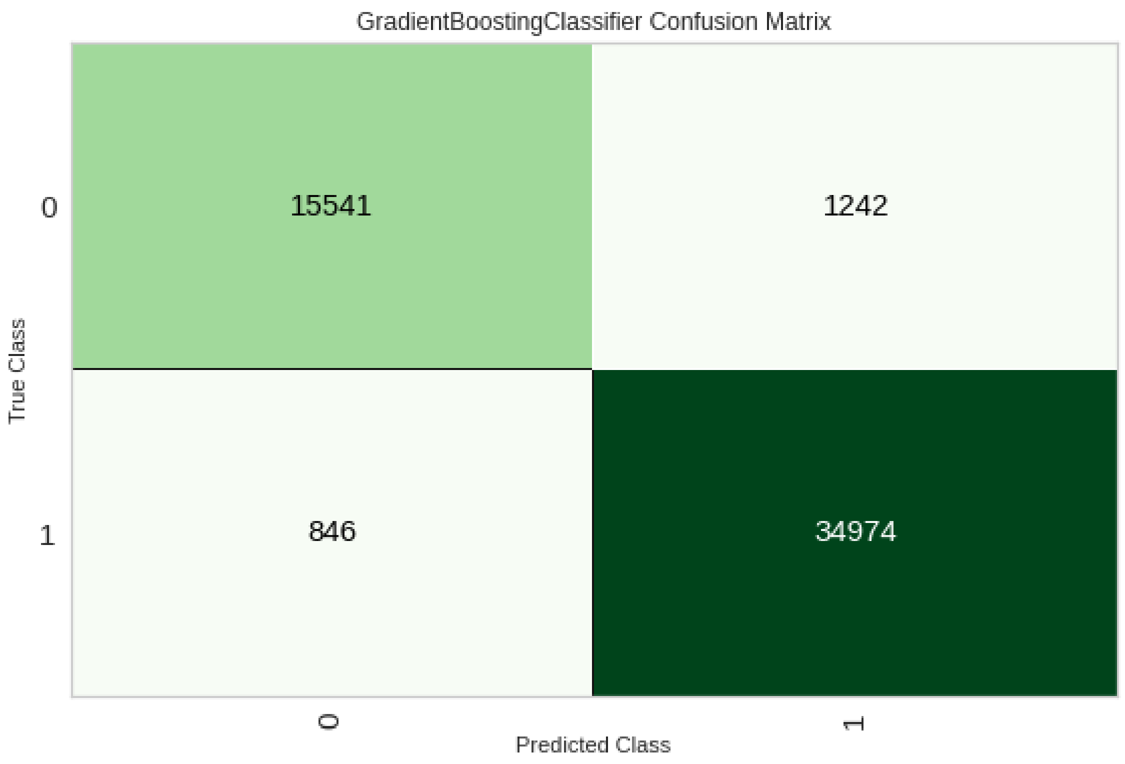

Figure 5.9: *Confusion Matrix for GBC*

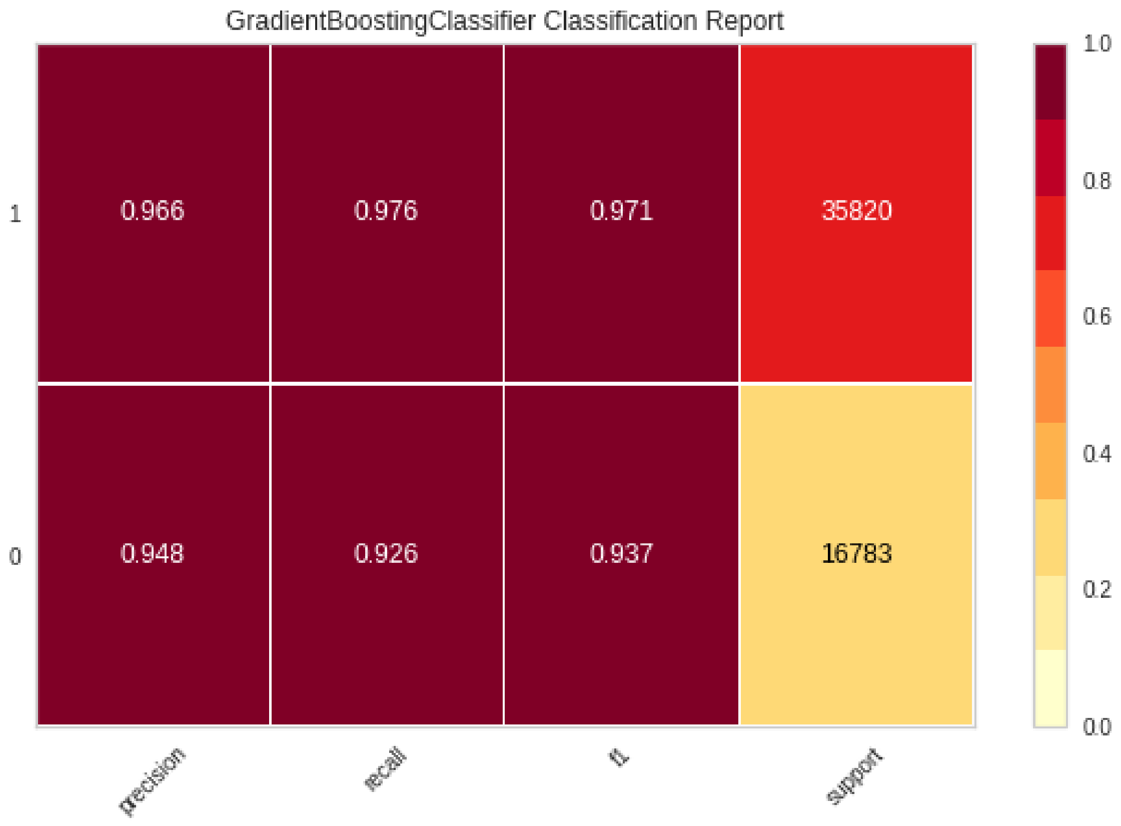

Figure 5.10: *Classification report for GBC*

5.2.6 Naive Bayes

A classification algorithm using Bayes theorem to predict the independence of an event. Bayes theorem:

$$P(\frac{c}{E}) = \frac{P(\frac{E}{c}) \times P(c)}{P(E)} \tag{5.9}$$

On a class or total event of E, the probability of c independently is what can be calculated by the above formula. The complexity of the algorithm grows with the increase of features. Such massive rise in features in a dataset is addressed by this classifying algorithm where a naive assumption is made, where all features are considered naively, as independents. The naive bayes classifier:

$$P(\frac{y}{X}) = \frac{P(y)\prod_{i=1}^{N} P(\frac{X_i}{y})}{P(X)}[16] \tag{5.10}$$

The confusion matrix and classification report obtained through our research for the above model are provided below.

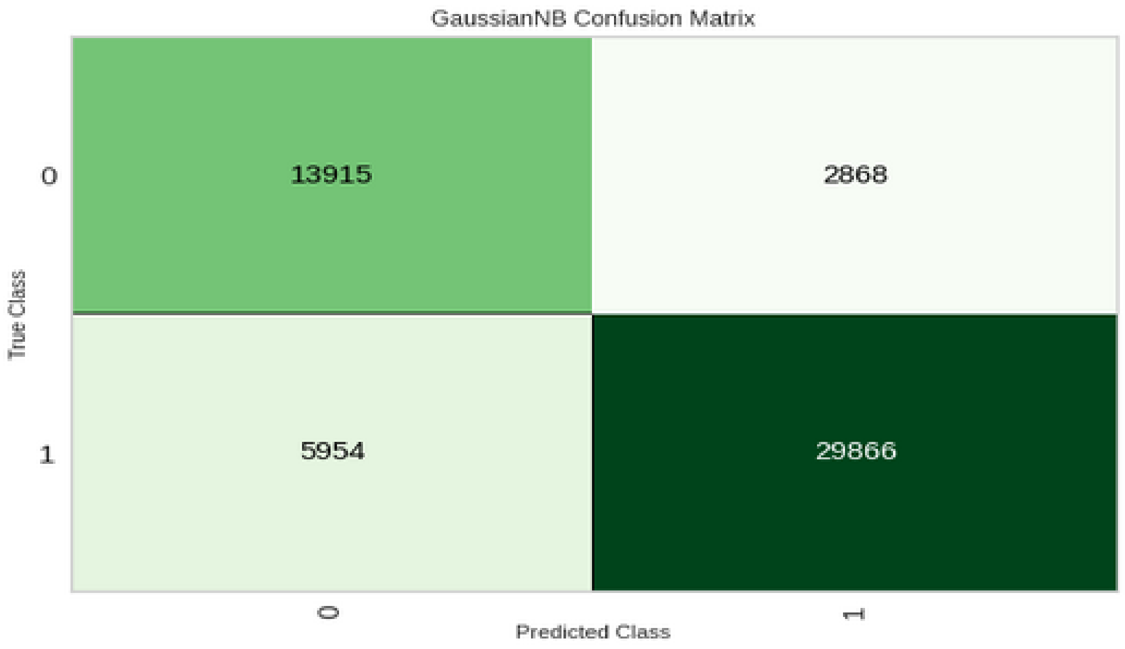

Figure 5.11: *Confusion Matrix for Naive Bayes*

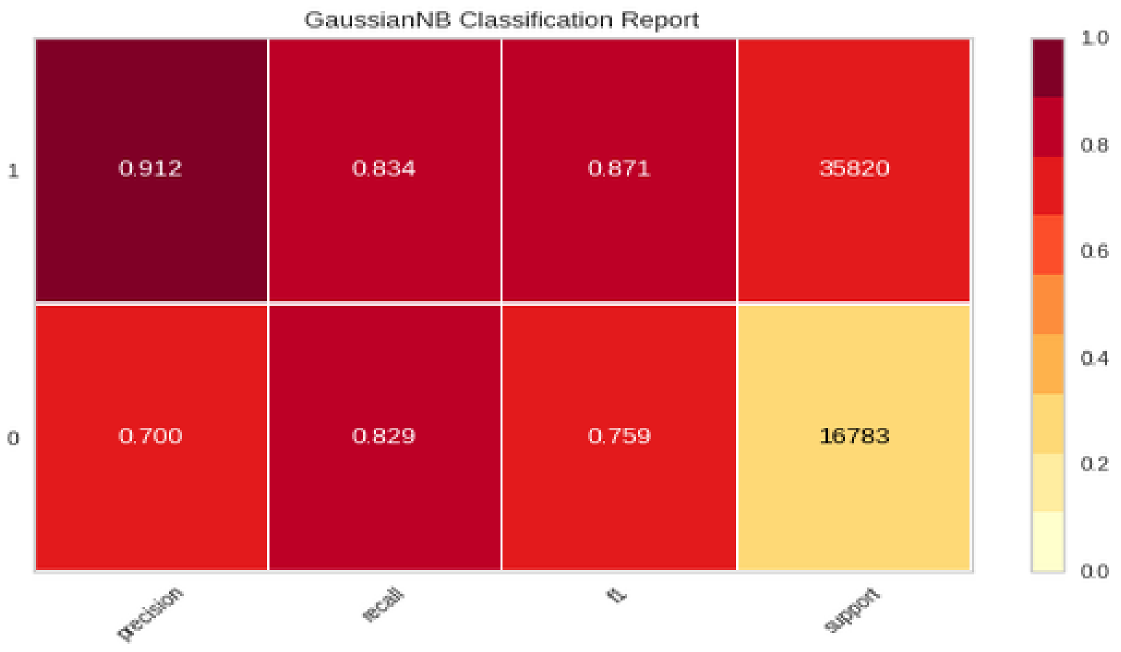

Figure 5.12: *Classification report for Naive Bayes*

5.2.7 Adaptive Boost classifier

Adaptive boosting or ada is another classifying algorithm similar to GBC in terms of operation. It is a boosting technique that is used as an Ensemble Method in machine learning. For an 'n' number of iterations on a given dataset, this algorithm produces weak predictions (i.e trees), the record which is incorrectly classified during the first model is given more priority. This record is then sent as an input for the next model. 'Weighted errors' are put in the regions where the previous classifiers performed poorly.
To assign some sample weight, the formula used is:

$$W = \frac{1}{N} \tag{5.11}$$

Here N is the number of records or training samples.
ADA classifiers only make a node with two leaves called stumps. Stumps are weak learners and they are calculated by the given formula:

$$\alpha = \frac{1}{2} ln \frac{1 - \epsilon}{\epsilon} \tag{5.12}$$

Here, is the total error, which is the sum of all errors accumulated. ADA can be used for regression calculation also [22].

The confusion matrix and classification report obtained through our research for the above model are provided below.

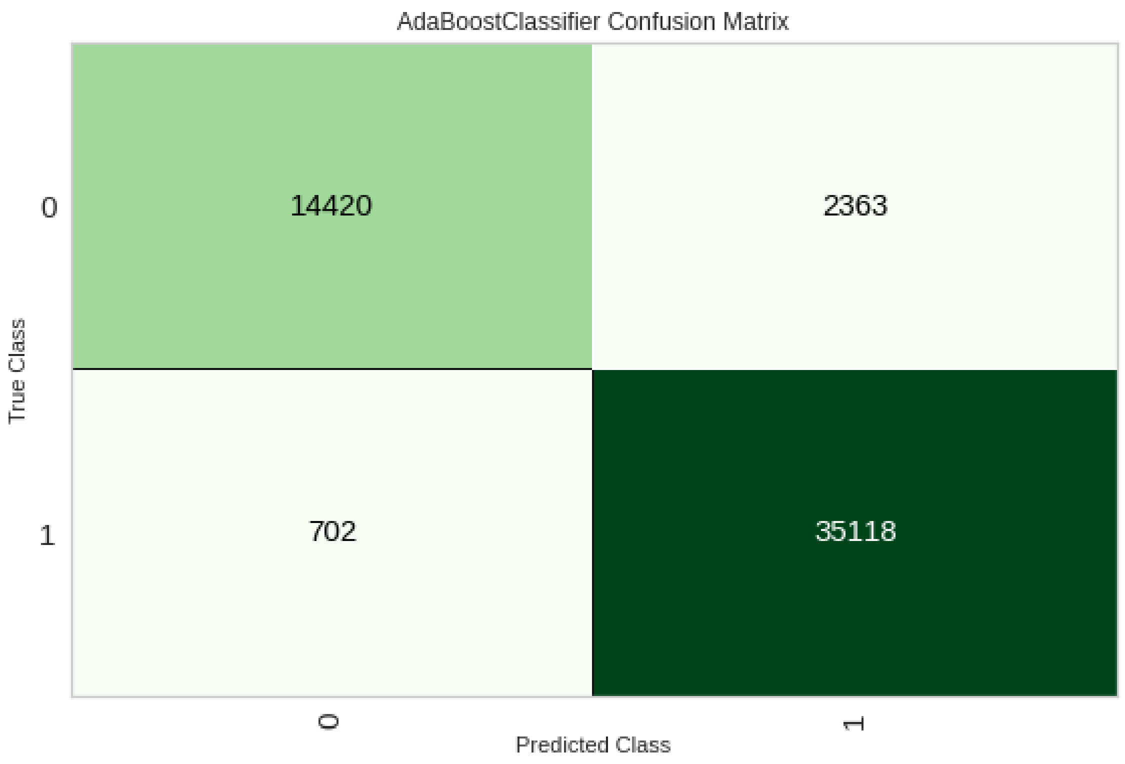

Figure 5.13: *Confusion Matrix for AdaBoost*

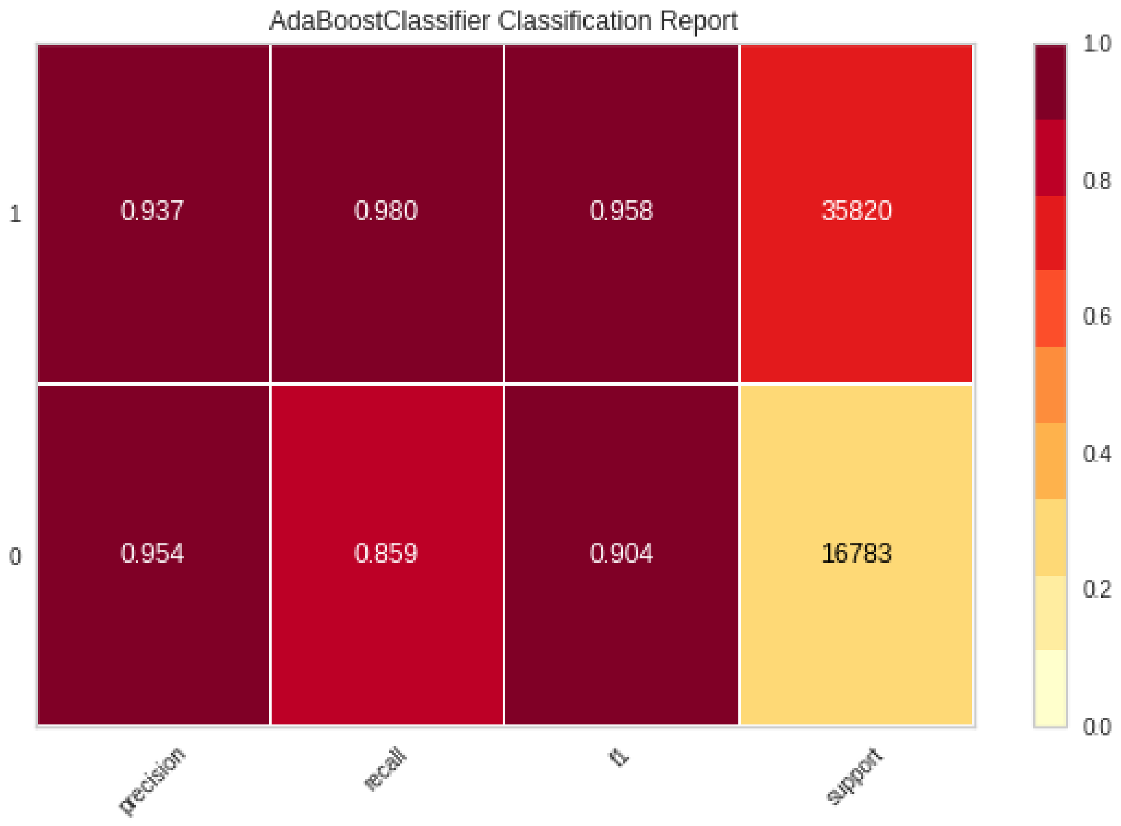

Figure 5.14: *Classification report for AdaBoost*

5.2.8 Blending

This research also uses the blending technique of ensemble learning to generate higher accuracies by taking the performance of multiple classifiers into accounts. For our research, we proceeded with two attempts at blending - one with the top three high performing classifiers, and the other with all the classifiers that met a certain threshold value for accuracy.
For our research we prepared two models with the ensemble blending technique. The first one, Blend 1, incorporates the data obtained from models DT, RF, KNN, and GBC. The second one, Blend 2, incorporates data from RF, DT, GBC, ADA, KNN, LR, and NB.
The confusion matrix and classification report obtained through our research for the model Blend 1 are provided below.

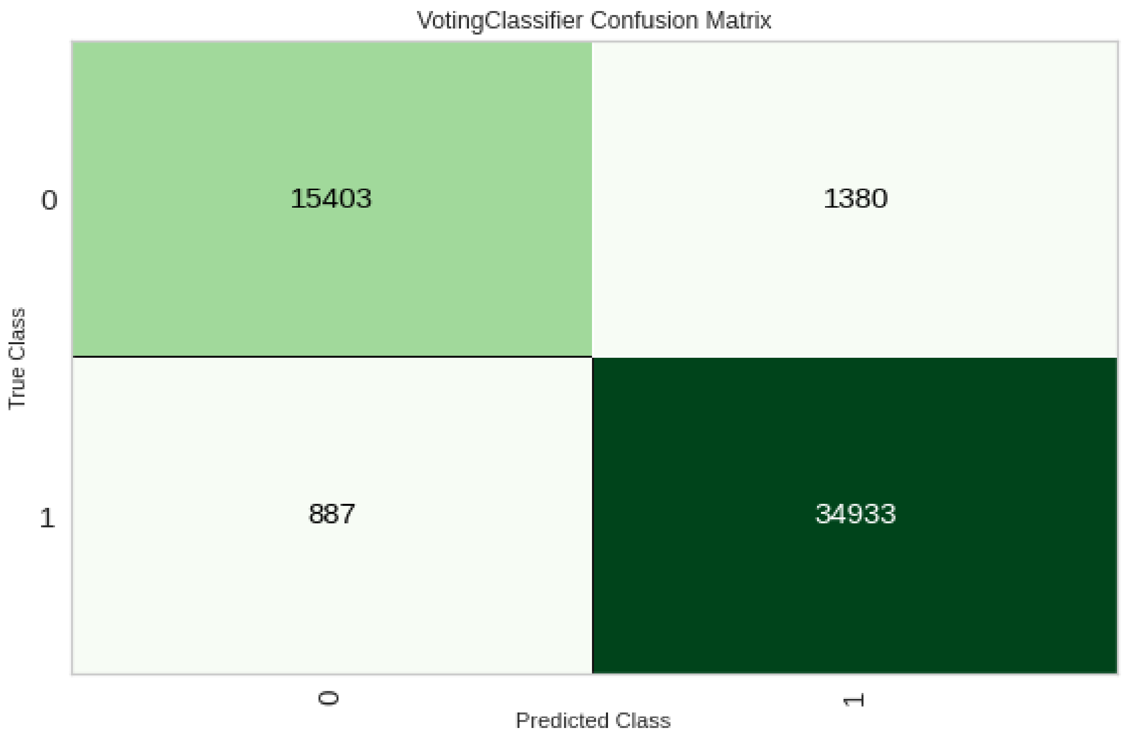

Figure 5.15: *Confusion Matrix for Blend 1*

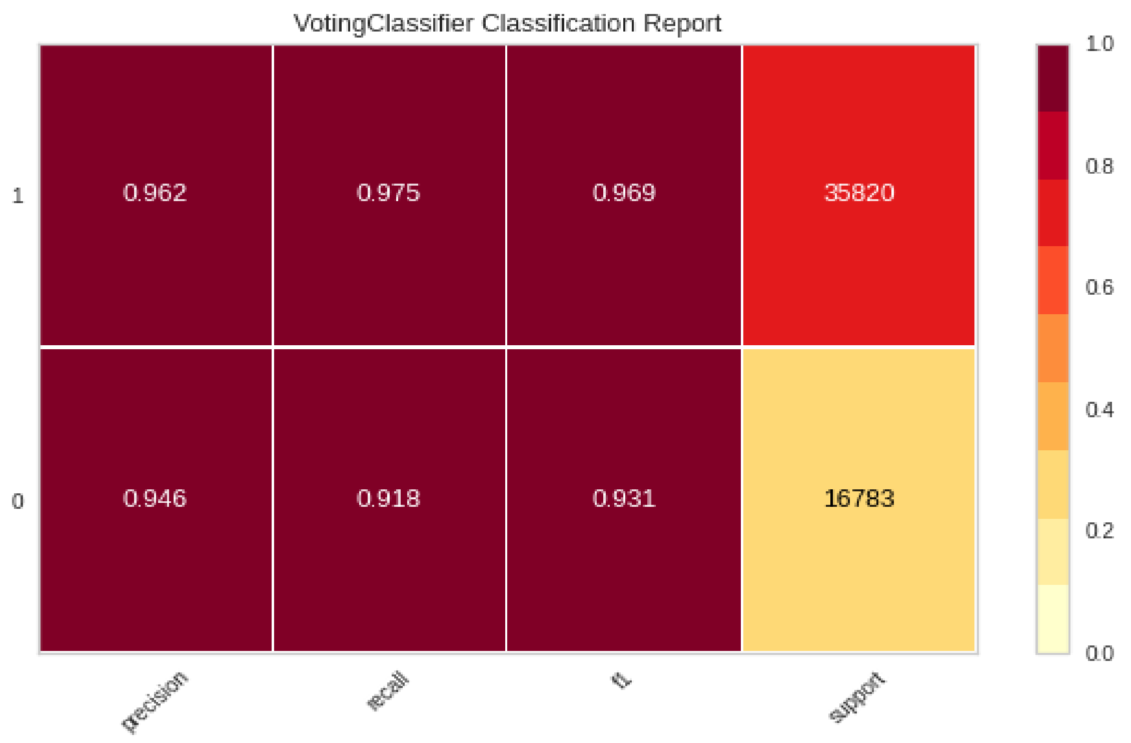

Figure 5.16: *Classification report for Blend 1*

The confusion matrix and classification report obtained through our research for the model Blend 2 are provided below.

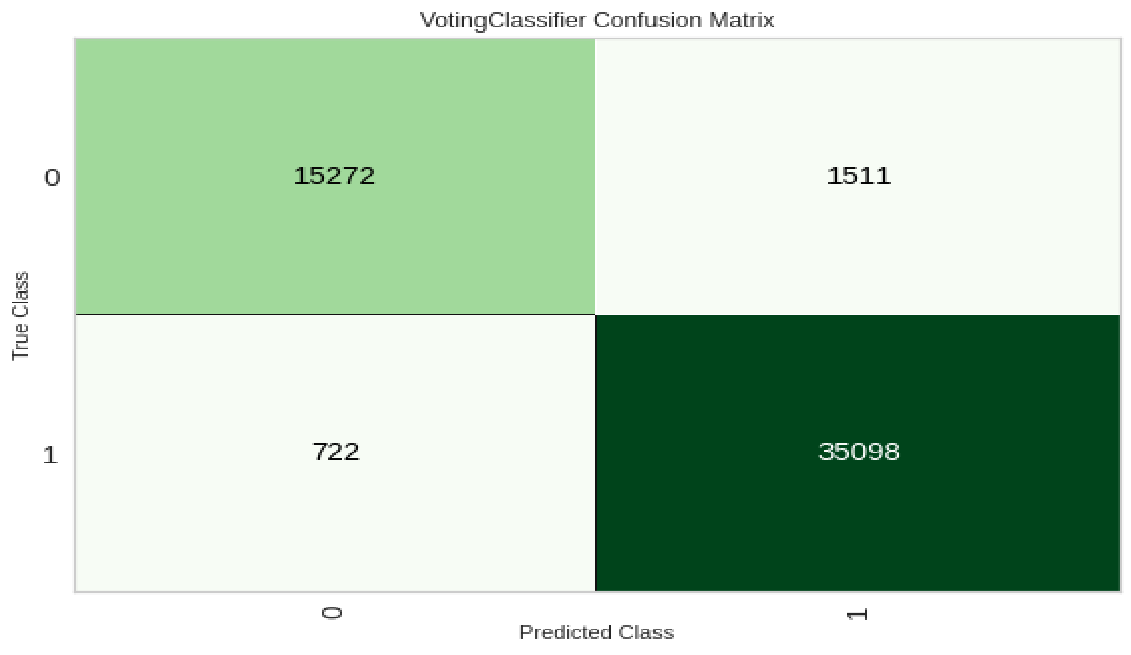

Figure 5.17: *Confusion Matrix for Blend 2*

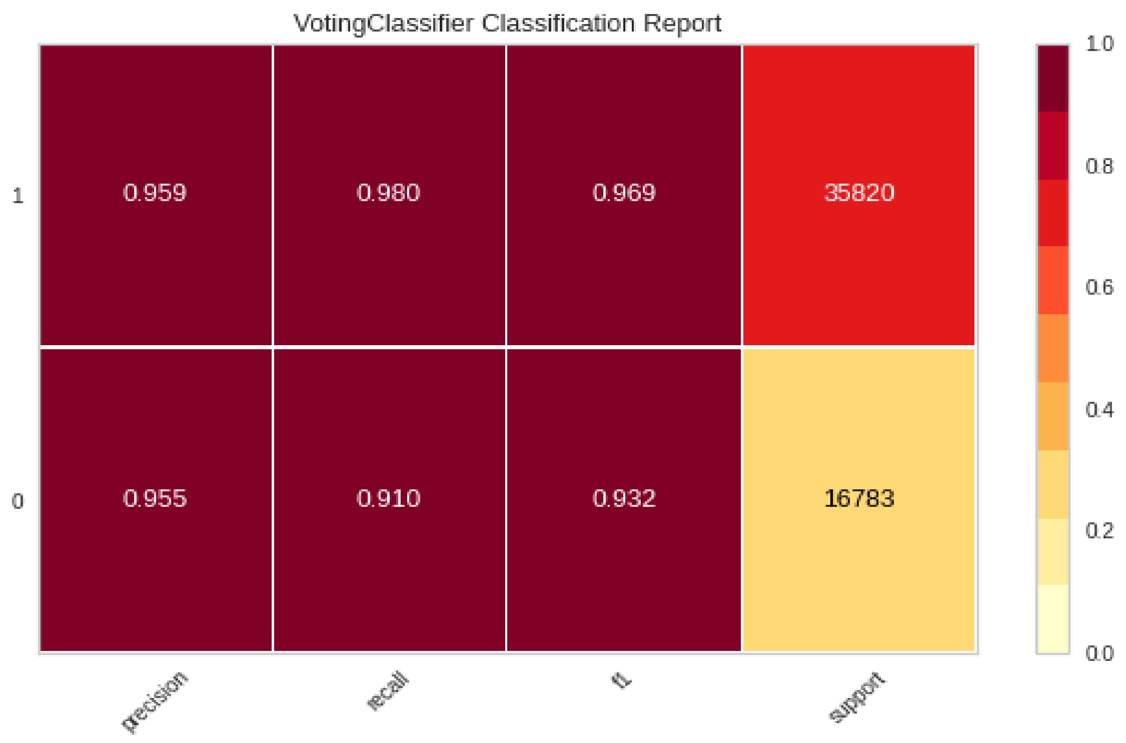

Figure 5.18: *Classification report for Blend 2*

Chapter 6

Result Analysis

The classifiers discussed in the previous section were used to train the models with a multi-collinearity threshold of 0.95. It is to be noted here that multi-collinearity is not an issue in case of classifiers like Decision Tree or Random Forest. That means that even without dropping similar columns from the dataset, these classifiers are able to perform at their fullest, as they automatically take care of such similar columns. However, setting the threshold led to an improvement in performance for the remaining models.
The environment used to run the program was "Google Colaboratory's Pro Version" that provides users with 25 GB of Virtual Memory, and the Nvidia Tesla T4 and P100 GPUs. Despite being a paid version and having high-end configurations, the Pro version still has its limitations as confirmed by Google itself.
The performance for each model was analyzed using the following metrics – accuracy, AUC, recall, precision, F1 score, kappa, MCC, and time taken to train.

Model	**Accuracy**	**AUC**	**Recall**	**Prec.**	**F1**	**Kappa**	**MCC**	**TT (Sec)**
Random Forest Classifier (RF)	0.9575	0.9932	0.9758	0.9622	0.969	0.9014	0.9017	5.3
Decision Tree Classifier(DT)	0.9471	0.9411	0.9597	0.9624	0.9611	0.8785	0.8785	0.589
K Neighbors Classifier(KNN)	0.9524	0.989	0.9701	0.9604	0.9652	0.89	0.8901	5.833
Gradient Boosting Classifier(GBC)	0.9584	0.9937	0.9769	0.9625	0.9696	0.9034	0.9037	17.311
Ada Boost Classifier(ADA)	0.9397	0.9879	0.9714	0.9419	0.9564	0.8589	0.86	3.779
Logistic Regression(LR)	0.9392	0.9861	0.9784	0.9353	0.9563	0.8565	0.8588	4.943
Naive Bayes (NB)	0.8322	0.9343	0.8363	0.9098	0.8715	0.631	0.6356	0.063

Table 6.1: Performance Score for Each Trained Model

Figures 6.1 through 6.9 show the class prediction error for each classifier.

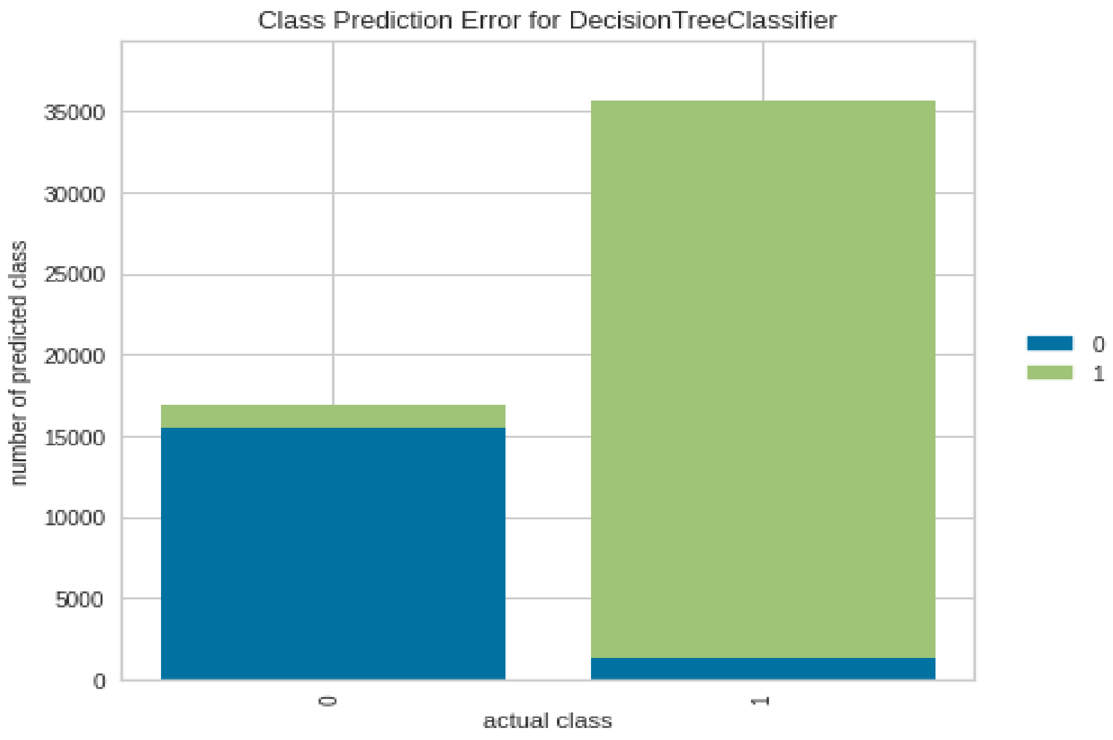

Figure 6.1: *Class Prediction Error for DT.*

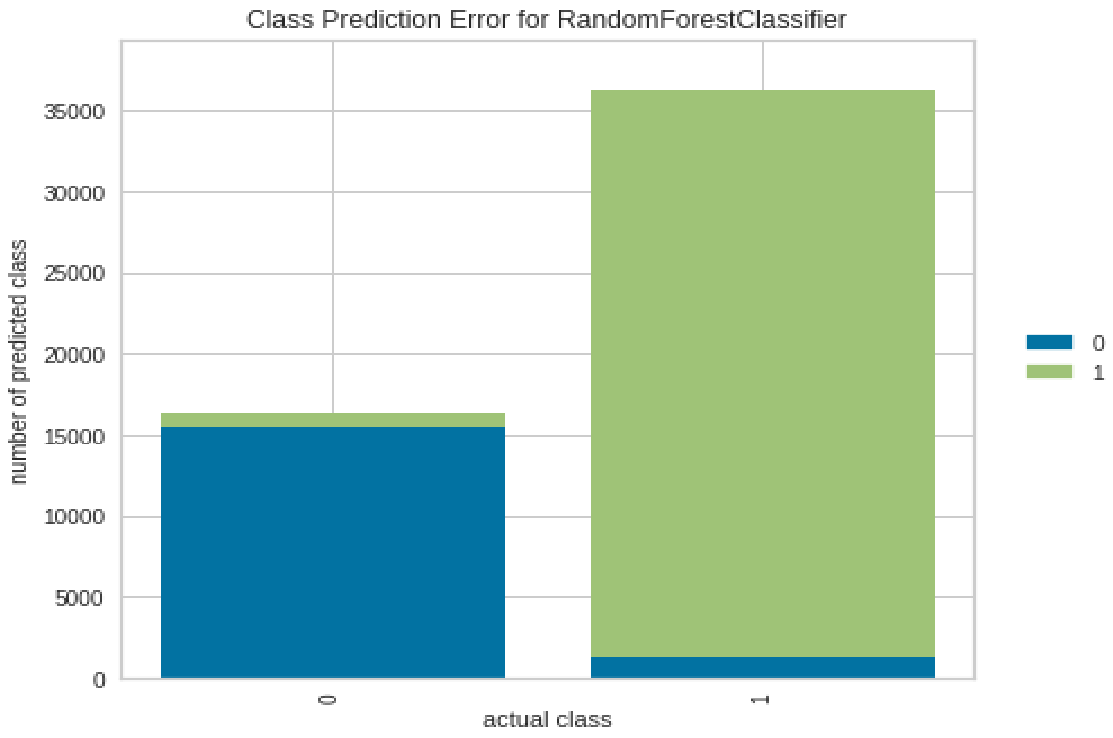

Figure 6.2: *Class Prediction Error for RF.*

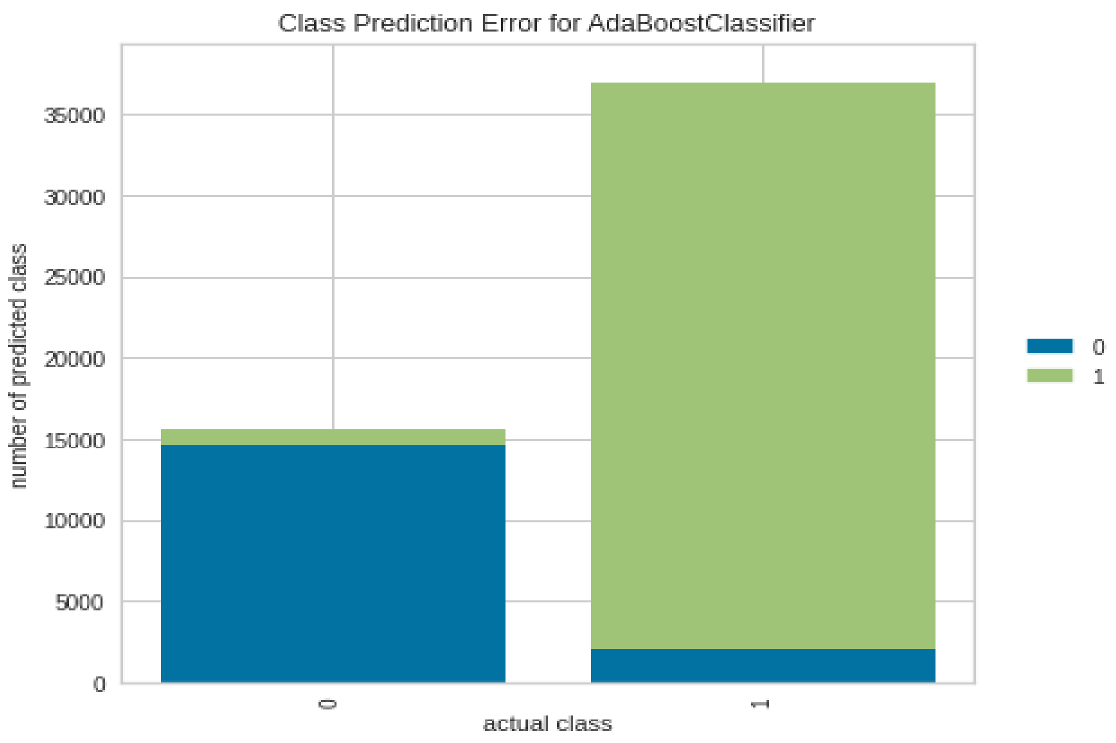

Figure 6.3: *Class Prediction Error for ADA.*

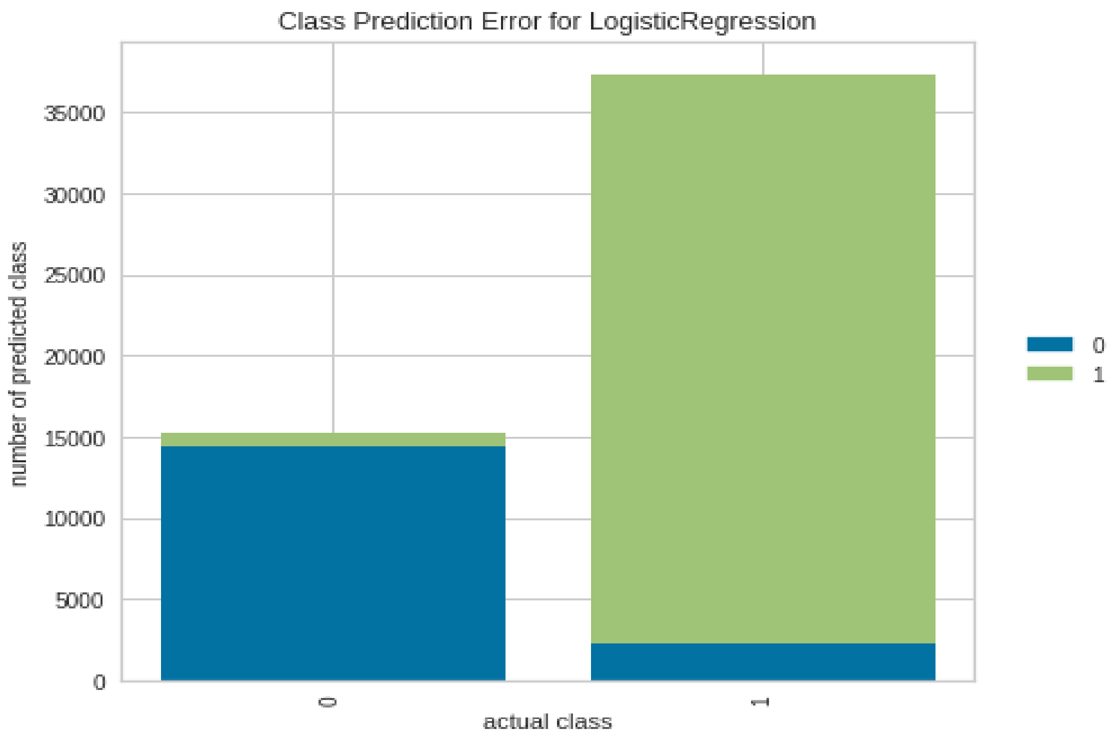

Figure 6.4: *Class Prediction Error for LR.*

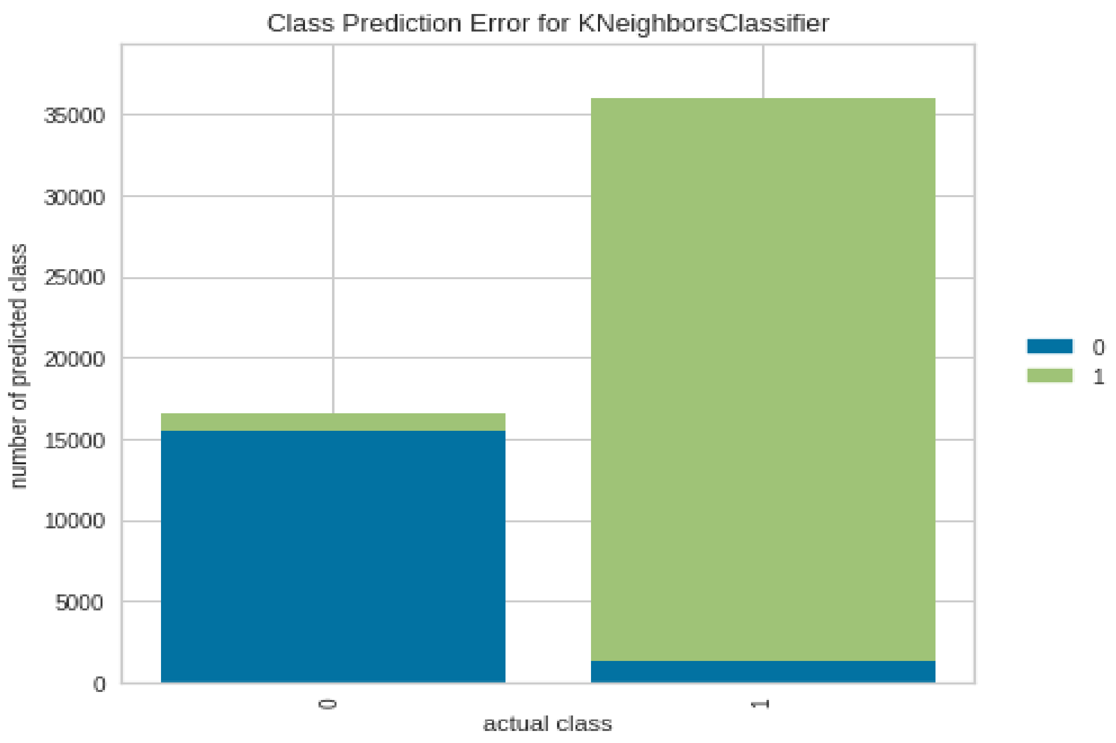

Figure 6.5: *Class Prediction Error for KNN.*

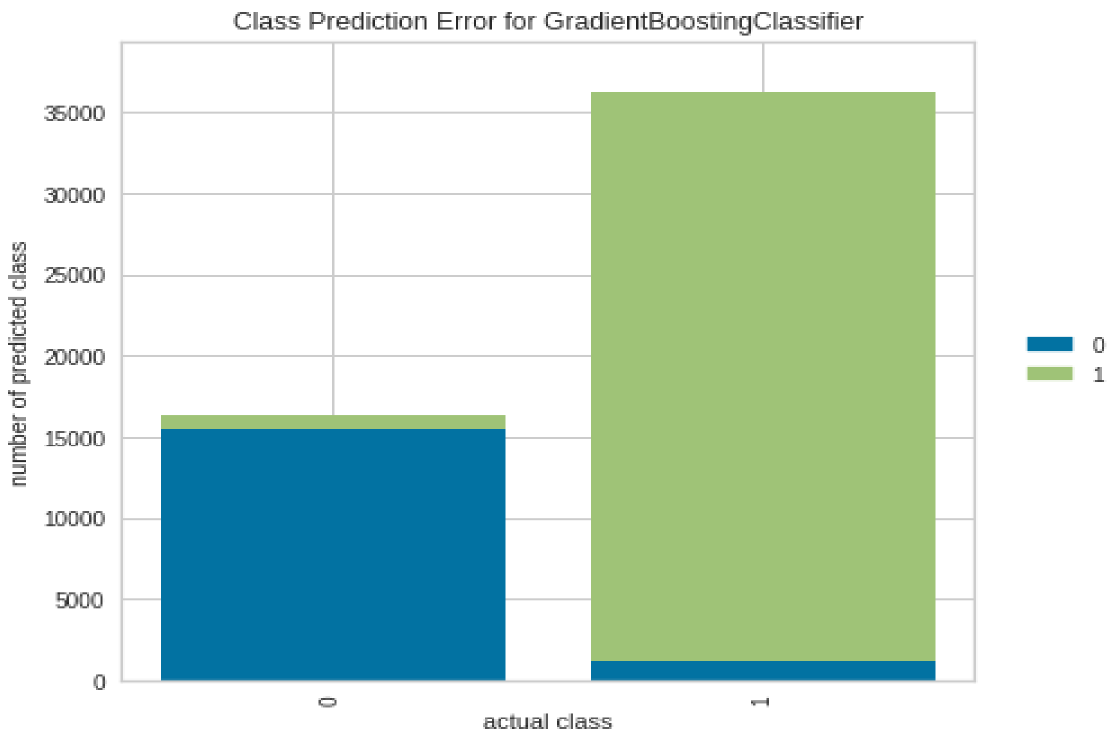

Figure 6.6: *Class Prediction Error for GBC.*

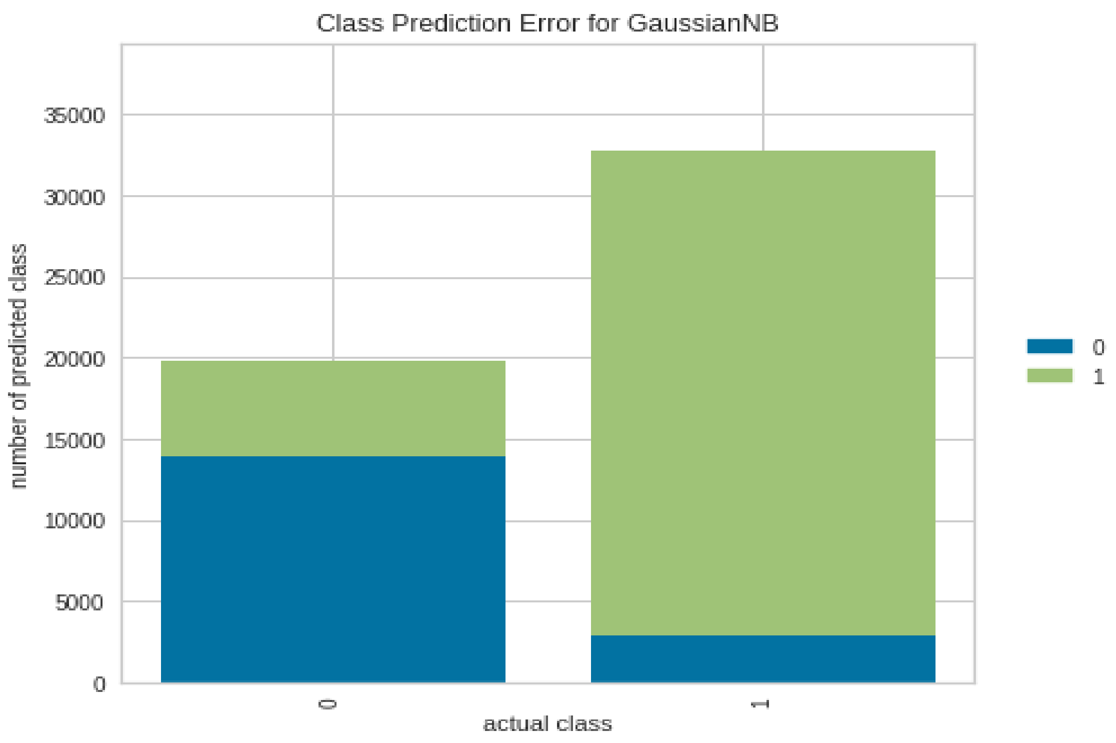

Figure 6.7: *Class Prediction Error for NB.*

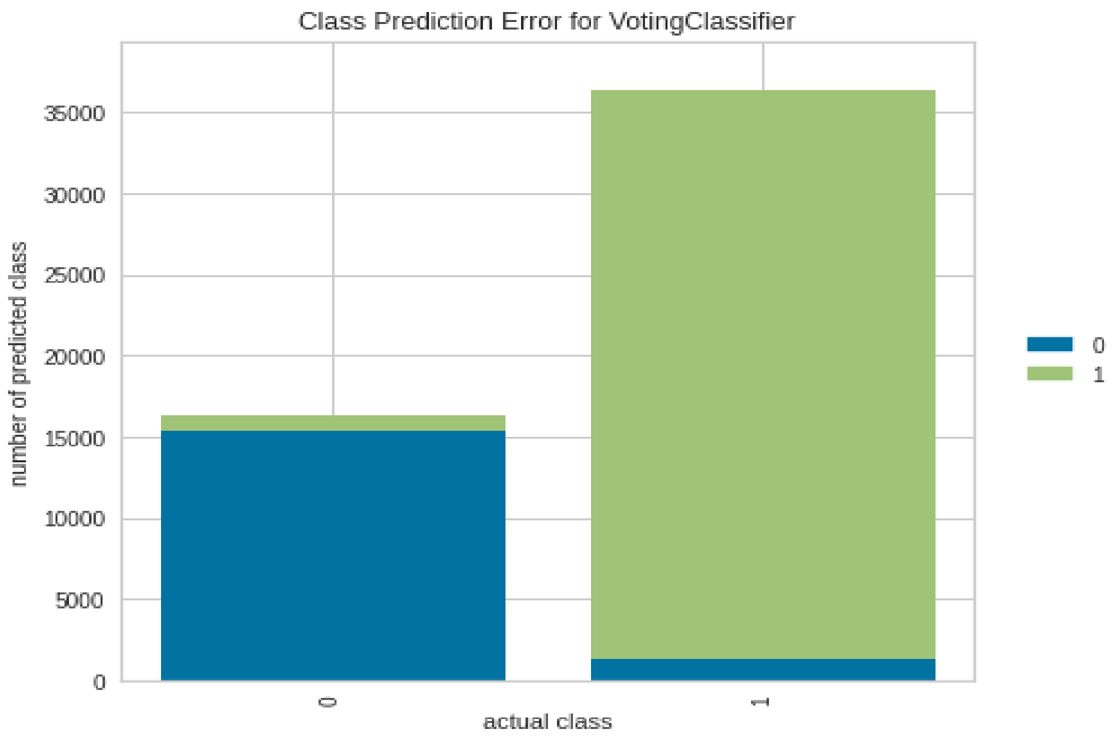

Figure 6.8: *Class Prediction Error for Blend 1.*

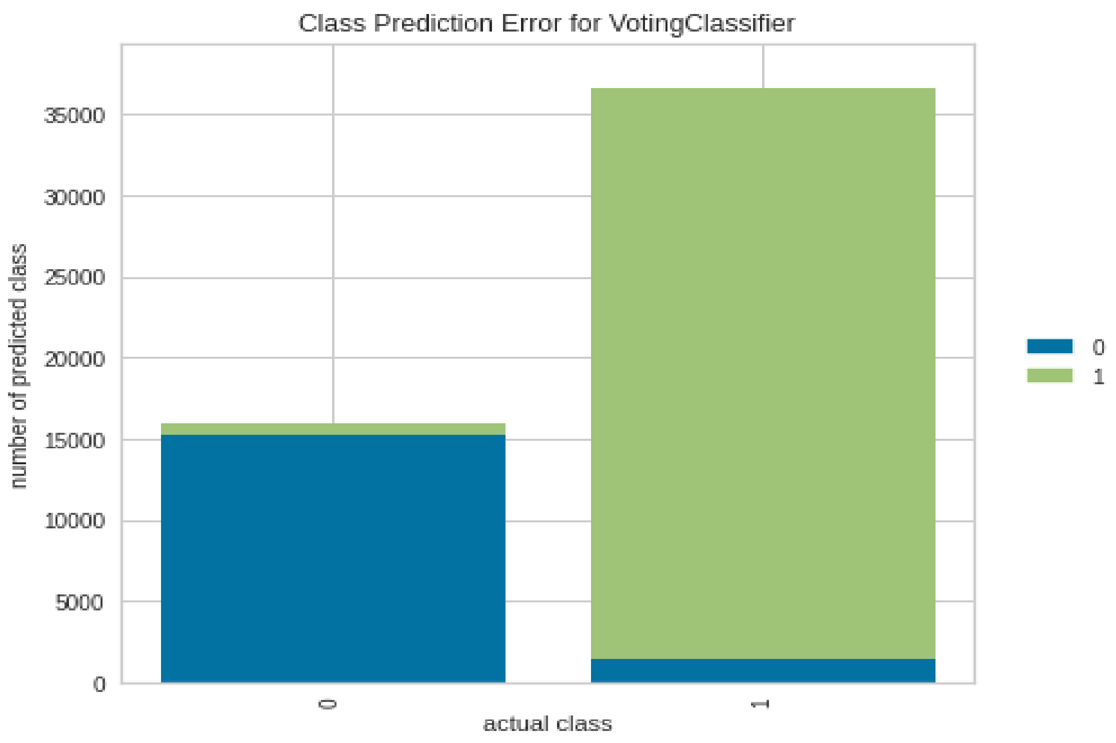

Figure 6.9: *Class Prediction Error for Blend 2.*

After testing the models on the unseen data subset, we get the results as summarized in table 6.2.

Model	**Train**	**Test**
Random Forest Classifier (RF)	0.9575	0.8722
Decision Tree Classifier (DT)	0.9471	0.8637
Gradient Boosting Classifier (GBC)	0.9584	0.8729
Ada Boost Classifier (ADA)	0.9397	0.8486
K Neighbors Classifier (KNN)	0.9524	0.8646
Logistic Regression (LR)	0.9392	0.8358
Naive Bayes (NB)	0.8322	0.7386

Table 6.2: Train-Test Results (Accuracy) for Each Classifier

As seen from the train-test results, RF gives a test accuracy of 87.22%, DT gives 86.37%, GBC gives 87.29%, ADA gives 84.86%, KNN gives 86.46%, LR gives 83.58%, and NB gives 73.86%.
KNN, RF, and NB provides a better accuracy score when compared to [35] that uses a wrapper-based decision tree for feature selection.
DT, KNN, and LR shows a comparatively better accuracy score when compared to [32] and its feature selection method that opted for only 19 of the available 42 features.
LR, NB, RF, KNN, and DT gives a comparatively better accuracy when compared against [33]. However, [33] does have a better precision value for NB and a greater recall score for LR.

RF, KNN, and NB also produced a greater accuracy score when compared against [20].
LR, and GBC achieved a much higher accuracy score when compared against [26].
It is to be noted here that while there are a few variations in the approach followed by this research and those hinted at above, this research required little effort and was able to achieve greater results in most cases. The PyCaret library enabled this research to be carried out within a short period and with less hassles when it came to feature selection and model training.
PyCaret's Tuning function adds to our research as well. It is through this function that the accuracy of classifiers – KNN, GBC, and NB – were improved upon. Especially in the case of Naive Bayes, which saw a jump from 45.9% accuracy, to an 83.22% accuracy.
The library's ensemble learning method was also applied. More specifically, we applied the blending technique, first incorporating the models obtained from DT, RF, KNN, and GBC, and then incorporating RF, DT, GBC, ADA, KNN, LR, and NB. The former gave a train accuracy of 95.47% and a test accuracy of 84.88%. The latter, on the other hand, gave a train accuracy of 95.51% and a test accuracy of 86.02%.

Chapter 7

Conclusion and Future Direction

7.1 Conclusion

To conclude, we will be revisiting some of the key objectives of this research and assess how far we have been able to progress with those objectives through our paper. Firstly, one of the primary objectives of our paper was to explore the PyCaret low-code machine learning library. Through this research, we were able to do so as we utilized some of the most important features and modules that the library has to offer. Out of the 18 available classifiers, we were able to utilize 8 to their full potential. We were also able to look into different functions that the library offers, including model training, evaluation, and ensembling. At the same time, we were able to assess how easy and flexible it is to use the library, and how feasible it is for quick deployment purposes.
Secondly, we were able to explore the UNSW-NB 15 dataset and understand its key features. We were able to analyze the dataset to find out information relevant to our line of work. Through the help of the classifiers, we were also able to check for important features that played a vital role in the final results of each algorithm used to train the models.
Finally, we were able to assess the performance of the PyCaret library for detecting intrusions in the UNSW-NB 15 dataset. Information from all 8 classifiers used for the research was successfully retrieved and presented in the preceding section.

7.2 Future Work

The PyCaret library, although shows great promise in terms of performance and ease of use, is in no way perfect. There are certain modules in this library that are yet to work properly. However, the developers behind it are continuously releasing updates and fixes that deal with such issues. For our future work, we would like to use those modules, which by then will hopefully be fixed, and do a more thorough analysis of how intrusion detection can be conducted in a more simplified and resourceful manner.
We also plan on carrying out a multi-classification on the UNSW-NB 15 dataset and conduct a performance analysis for that as well.
Alongside that, we would also like to work with more datasets of a similar or different nature. While doing so, we would like to work with the regression and anomaly

detection modules of the PyCaret library.

Bibliography

[1] M. T. Banday, J. Qadri, and N. Shah, "Study of botnets and their threats to internet security," Jan. 2009.

[2] J.-c. Zhao, J.-f. Zhang, Y. Feng, and J.-x. Guo, "The study and application of the iot technology in agriculture," in *2010 3rd International Conference on Computer Science and Information Technology*, vol. 2, 2010, pp. 462–465. DOI: 10.1109/ICCSIT.2010.5565120.

[3] T. Liu and D. Lu, "The application and development of iot," in *2012 International Symposium on Information Technologies in Medicine and Education*, vol. 2, 2012, pp. 991–994. DOI: 10.1109/ITiME.2012.6291468.

[4] A. Natekin and A. Knoll, "Gradient boosting machines, a tutorial," *Frontiers in neurorobotics*, vol. 7, p. 21, Dec. 2013. DOI: 10.3389/fnbot.2013.00021.

[5] S. Hiremath, G. Yang, and K. Mankodiya, "Wearable internet of things: Concept, architectural components and promises for person-centered healthcare," in *2014 4th International Conference on Wireless Mobile Communication and Healthcare - Transforming Healthcare Through Innovations in Mobile and Wireless Technologies (MOBIHEALTH)*, 2014, pp. 304–307. DOI: 10.1109/MOBIHEALTH.2014.7015971.

[6] M. Stevanovic and J. M. Pedersen, "An efficient flow-based botnet detection using supervised machine learning," in *2014 International Conference on Computing, Networking and Communications (ICNC)*, 2014, pp. 797–801. DOI: 10.1109/ICCNC.2014.6785439.

[7] R. Mahmoud, T. Yousuf, F. Aloul, and I. Zualkernan, "Internet of things (iot) security: Current status, challenges and prospective measures," in *2015 10th International Conference for Internet Technology and Secured Transactions (ICITST)*, 2015, pp. 336–341. DOI: 10.1109/ICITST.2015.7412116.

[8] N. Moustafa and J. Slay, "The significant features of the unsw-nb15 and the kdd99 data sets for network intrusion detection systems," in *2015 4th International Workshop on Building Analysis Datasets and Gathering Experience Returns for Security (BADGERS)*, 2015, pp. 25–31. DOI: 10.1109/BADGERS.2015.014.

[9] ——, *UNSW-NB15 Dataset*, UNSW, 2015. [Online]. Available: https://cloudstor.aarnet.edu.au/.

[10] ——, "Unsw-nb15: A comprehensive data set for network intrusion detection systems (unsw-nb15 network data set)," in *2015 Military Communications and Information Systems Conference (MilCIS)*, 2015, pp. 1–6. DOI: 10.1109/MilCIS.2015.7348942.

[11] E. Ogu, N. Vrakas, C. Ogu, and A.-I. B.M., "On the internal workings of botnets: A review," *International Journal of Computer Applications*, vol. 138, pp. 975–8887, Apr. 2016. DOI: 10.5120/ijca2016908797.

[12] G. K. Scope, "Review on iot technologies," 2016.

[13] Ahmed, "Three major challenges facing iot," *IEEE Internet of Things*, Mar. 2017. [Online]. Available: https://iot.ieee.org/newsletter/march-2017/three-major-challenges-facing-iot.html.

[14] S. B. Baker, W. Xiang, and I. Atkinson, "Internet of things for smart healthcare: Technologies, challenges, and opportunities," *IEEE Access*, vol. 5, pp. 26 521–26 544, 2017. DOI: 10.1109/ACCESS.2017.2775180.

[15] E. Bertino and N. Islam, "Botnets and internet of things security," *Computer*, vol. 50, no. 2, pp. 76–79, 2017. DOI: 10.1109/MC.2017.62.

[16] P. Kaviani and S. Dhotre, "Short survey on naive bayes algorithm," *International Journal of Advance Research in Computer Science and Management*, vol. 04, Nov. 2017.

[17] P. Parhana, M. Lakshmaiah, S. Allahudheen, S. Dastagiri, and V. Saritha, "Review on internet of things: Recent applications and its challenges," Nov. 2017.

[18] S. Al-Sarawi, M. Anbar, K. Alieyan, and M. Alzubaidi, "Internet of things (iot) communication protocols: Review," in *2017 8th International Conference on Information Technology (ICIT)*, 2017, pp. 685–690. DOI: 10.1109/ICITECH.2017.8079928.

[19] S. Haq and Y. Singh, "Botnet detection using machine learning," in *2018 Fifth International Conference on Parallel, Distributed and Grid Computing (PDGC)*, 2018, pp. 240–245. DOI: 10.1109/PDGC.2018.8745912.

[20] M. Suleiman and B. Issac, "Performance comparison of intrusion detection machine learning classifiers on benchmark and new datasets," English, 28th International Conference on Computer Theory and Applications, ICCTA 2018 ; Conference date: 30-10-2018 Through 01-11-2018, Oct. 2018. [Online]. Available: https://iccta.aast.edu/.

[21] C. Xie, P. Yang, and Y. Yang, "Open knowledge accessing method in iot-based hospital information system for medical record enrichment," *IEEE Access*, vol. 6, pp. 15 202–15 211, 2018. DOI: 10.1109/ACCESS.2018.2810837.

[22] *Boosting and AdaBoost clearly explained*, Medium, Feb. 2019. [Online]. Available: https://towardsdatascience.com/boosting-and-adaboost-clearly-explained-856e21152d3e.

[23] DeepAI, *Logistic regression*, May 2019. [Online]. Available: https://deepai.org/machine-learning-glossary-and-terms/logistic-regression.

[24] W. A. Jabbar, T. K. Kian, R. M. Ramli, S. N. Zubir, N. S. M. Zamrizaman, M. Balfaqih, V. Shepelev, and S. Alharbi, "Design and fabrication of smart home with internet of things enabled automation system," *IEEE Access*, vol. 7, pp. 144 059–144 074, 2019. DOI: 10.1109/ACCESS.2019.2942846.

[25] *K-Nearest Neighbor in Machine Learning*, Knowledgehut, Sep. 2019. [Online]. Available: https://https://www.knowledgehut.com/blog/data-science/knn-for-machine-learning.

[26] S. Meftah, T. Rachidi, and N. Assem, *Network based intrusion detection using the unsw-nb15 dataset*, Sep. 2019. [Online]. Available: https://journal.uob.edu.bh/handle/123456789/3580.

[27] *Random Forest Algorithm for Machine Learning*, Medium, Apr. 2019. [Online]. Available: https://medium.com/capital-one-tech/random-forest-algorithm-for-machine-learning-c4b2c8cc9feb.

[28] *Understanding Random Forest*, Medium, Jun. 2019. [Online]. Available: https://towardsdatascience.com/understanding-random-forest-58381e0602d2.

[29] T. Alam, *A reliable communication framework and its use in internet of things (iot)*, Aug. 2020. DOI: 10.36227/techrxiv.12657158.v1. [Online]. Available: https://www.techrxiv.org/articles/preprint/A_Reliable_Communication_Framework_and_Its_Use_in_Internet_of_Things_IoT_/12657158/1.

[30] M. Ali, *Pycaret: An open source, low-code machine learning library in python*, PyCaret version 2.3.1, Apr. 2020. [Online]. Available: https://www.pycaret.org.

[31] *Introduction to Decision Tree Algorithm - Explained with Examples*, GreatLearning Blog, Feb. 2020. [Online]. Available: https://www.mygreatlearning.com/blog/decision-tree-algorithm/.

[32] S. M. Kasongo and Y. Sun, *Performance analysis of intrusion detection systems using a feature selection method on the unsw-nb15 dataset*, Nov. 2020. [Online]. Available: https://journalofbigdata.springeropen.com/articles/10.1186/s40537-020-00379-6.

[33] G. Kocher and G. Kumar, *Performance analysis of machine learning classifiers for intrusion detection using unsw-nb15 dataset*, 2020. [Online]. Available: https://aircconline.com/csit/papers/vol10/csit102004.pdf.

[34] S. S. Sarmah, "An efficient iot-based patient monitoring and heart disease prediction system using deep learning modified neural network," *IEEE Access*, vol. 8, pp. 135 784–135 797, 2020. DOI: 10.1109/ACCESS.2020.3007561.

[35] M. A. Umar, C. Zhanfang, and Y. Liu, *Network intrusion detection using wrapper-based decision tree for feature selection*, Aug. 2020. [Online]. Available: https://arxiv.org/abs/2008.07405.

[36] Z. Zhou, H. Yu, and H. Shi, "Human activity recognition based on improved bayesian convolution network to analyze health care data using wearable iot device," *IEEE Access*, vol. 8, pp. 86 411–86 418, 2020. DOI: 10.1109/ACCESS.2020.2992584.

[37] L. Bhajantri and G. S S, "A comprehensive survey on resource management in internet of things," *Journal of Telecommunications and Information Technology*, vol. 4, pp. 27–43, Jan. 2021. DOI: 10.26636/jtit.2020.145220.

[38] Ericbrownaustin, *Machine learning with pycaret and the home price dataset*, Jan. 2021. [Online]. Available: https://ericonanalytics.com/pycaret-and-machine-learning-home-price-dataset/?fbclid=IwAR1CPGTsEp5gdkYdNv9w4_fLYfCDOnR3XkBl1CNYaXAASaN1wvKHc9WpXAw.

[39] U. Gain and V. Hotti, "Low-code AutoML-augmented data pipeline – a review and experiments," *Journal of Physics: Conference Series*, vol. 1828, no. 1, p. 012 015, Feb. 2021. DOI: 10.1088/1742-6596/1828/1/012015. [Online]. Available: https://doi.org/10.1088/1742-6596/1828/1/012015.

[40] *Global iot market to be worth usd 1,463.19 billion by 2027 at 24.9% cagr; demand for real-time insights to spur growth, says fortune business insights™*, Fortune Business Insights, Apr. 2021. [Online]. Available: https://www.globenewswire.com/en/news-release/2021/04/08/2206579/0/en/Global-IoT-Market-to-be-Worth-USD-1-463-19-Billion-by-2027-at-24-9-CAGR-Demand-for-Real-time-Insights-to-Spur-Growth-says-Fortune-Business-Insights.html.

[41] J. Butts and S. Shenoi, "Botnets as an instrument of warfare," *IFIP ACT 367*, 2011. [Online]. doi:https://link.springer.com/content/pdf/10.1007%2F978-3-642-24864-1_2*.pdf*.