TECHNO INTERNATIONAL NEW TOWN

(Formerly known as Techno India College of Technology)

Newtown, Megacity, Kolkata -700156

Department of Information Technology

Submission for the partial fulfillment of B.Tech Final Year Project-PROJ-CS881

PROJECT REPORT

On

Comparative study on Intrusion Detection System using classification algorithms

Prepared by

Rouprita Bhattacharjee (Roll No- 18700218017)

Purbita Mitra (Roll No- 18700218021)

Keshav Tiwari (Roll No- 18700218026)

Ayushi Keshri (Roll No- 18700218035)

Under The Guidance of

Prof. Sanjukta Bhattacharya

For

Batch:- 2018-2022 *Semester : 8th* *Year : Jan 2022 – June 2022*

Stream:- Information Technology *Year of Study: 4th*

Affiliated to

MAULANA ABUL KALAM AZAD UNIVERSITY OF TECHNOLOGY, WEST BENGAL

MAULANA ABUL KALAM AZAD UNIVERSITY OF TECHNOLOGY, WESTBENGAL

(FORMERLY KNOWN AS WEST BENGAL UNIVERSITY OF TECHNOLOGY)

ACKNOWLEDGEMENT

We would like to express our sincere gratitude to Prof. Sanjukta Bhattacharya of the department of Information Technology, whose role as project guide was invaluable for the project. We are extremely thankful for the keen interest she took in advising us, for the books and reference materials provided for the moral support extended to us.

Last but not the least we convey our gratitude to all the teachers for providing us the technical skill that will always remain as our asset and to all non-teaching staff for the cordial support they offered.

Place: Techno International New Town, Kolkata

Date: _______________________

Rouprita Bhattacharya
(Roll No: -18700218017)

____

Purbita Mitra
(Roll No: -18700218021)

____

Keshav Tiwari
(Roll No: -18700218026)

____

Ayushi Keshri
(Roll No: - 18700218035)

____

Department of Information Technology,
Techno International New Town, Newtown Megacity
Kolkata – 700156
West Bengal, India.

Approval

This is to certify that the project report entitled " ***Comparative study on Intrusion Detection System using classification algorithms*** " prepared under my supervision by Rouprita Bhattacharya (18700218017), Purbita Mitra (18700218021), Keshav Tiwari (18700218026) and Ayushi Keshri (18700218035) be accepted in partial fulfilment for the degree of Bachelor of Technology in Information Technology which is affiliated to Maulana Abul Kalam Azad University of Technology, West Bengal (Formerly known as West Bengal University of Technology).

It is to be understood that by this approval, the undersigned does not necessarily endorse or approve any statement made, opinion expressed or conclusion drawn thereof, but approves the report only for the purpose for which it has been submitted.

Project Mentor:

…………………………………………

Prof. Sanjukta Bhattacharya
Dept. of Information Technology
Techno International New Town

External Examiner:

……………………………………….

………………………………………….

Prof. Shiladitya Munshi,
HOD, Dept. of Information Technology
Techno International Newtown

Abstract

There are a variety of tools available to detect attacks and exploits and take steps to block or stop cyber attacks. Things like firewalls to prevent unauthorized traffic from entering the network, spam filters to reject unwanted email messages, and antimalware tools to protect endpoints from malware are universal across just about every organization, regardless of size or industry. Another valuable security tool that is almost as ubiquitous is a network IDS—or intrusion detection system. Let's take a closer look at what a network IDS does and how it can help you protect your network and secure your data. No firewall is foolproof, and no network is impenetrable. Attackers continuously develop new exploits and attack techniques designed to circumvent your defences. Many attacks leverage other malware or social engineering to obtain user credentials that grant them access to your network and data
This project is a comparative study on how effective different classification algorithms are in filtering out intrusive data samples from a combination of both intrusive and normal data samples. The classification algorithms used in our project are

1) SVM algorithm
2) Random Forest algorithm
3) Decision Tree algorithm

For our experiment, we have used subset of UNSW-NB15 dataset and then the above mentioned three algorithms are applied using an available Intrusion detection coding model. The results thus obtained are compared on the parameters of accuracy and confusion matrix to determine the best algorithm.

Contents

Page No.

LIST OF FIGURES

Figures **Page No**

CHAPTER 1: INTRODUCTION

1.1 INTRODUCTION

As businesses shift to distributed environments, the threat landscape gets broader, and hackers are now shifting their focus to attacking the systems of remote workers. In this context, the importance of an intrusion detection system or IDS becomes more important than ever, in protecting endpoint devices and enterprise networks from sophisticated attacks.

The earliest preliminary IDS concept was delineated in 1980 by James Anderson at the National Security Agency and consisted of a set of tools intended to help administrators review audit trails. User access logs, file access logs, and system event logs are examples of audit trails.

Fred Cohen noted in 1987 that it is impossible to detect an intrusion in every case, and that the resources needed to detect intrusions grow with the amount of usage.

Dorothy E. Denning, assisted by Peter G. Neumann, published a model of an Intrusion Detection System in 1986 that formed the basis for many systems today. Her model used statistics for anomaly_detection, and resulted in an early IDS at SRI International named the Intrusion Detection Expert System (IDES), which ran on Sun workstations and could consider both user and network level data. IDE'S had a dual approach with a rule-based Expert System to detect known types of intrusions plus a statistical anomaly detection component based on profiles of users, host systems, and target systems. The author of "IDES: An Intelligent System for Detecting Intruders," Teresa F. Lunt, proposed adding an Artificial neural network as a third component. She said all three components could then report to a resolver. SRI followed IDES in 1993 with the Next-generation Intrusion Detection Expert System

1.2 INTRUSION DETECTION SYSTEM (I.D.S)

An **Intrusion Detection System** (IDS) is a network security technology originally built for detecting vulnerability exploits against a target application or computer. It is a device or software application that monitors a network for malicious activity or policy violations. Any malicious activity or violation is typically reported or collected centrally using a security information and event management system. Some IDS's are capable of responding to detected intrusion upon discovery.

IDS needs only to detect threats and as such is placed out-of-band on the network infrastructure, meaning that it is not in the true real-time communication path between the sender and receiver of information. Rather, IDS solutions will often take advantage of a TAP or SPAN port to analyze a copy of the inline traffic stream (and thus ensuring that IDS does not impact inline network performance).

IDS was originally developed this way because at the time the depth of analysis required for intrusion detection could not be performed at a speed that could keep pace with components on the direct communications path of the network infrastructure.

As explained, the IDS is also a listen-only device. The IDS monitors traffic and report its results to an administrator, but cannot automatically take action to prevent a detected exploit from taking over the system. Attackers are capable of exploiting vulnerabilities very quickly once they enter the network, rendering the IDS an inadequate deployment for prevention device.

1.3 TYPES OF IDS:

An intrusion detection system is broadly categorized based on where the IDS sensors are placed: network or host.

Network Intrusion Detection System

A network-based intrusion detection system (NIDS) monitors and analyzes network traffic for suspicious behaviour and real threats with the help of NIDS sensors. It scrutinizes the content and header information of all packets moving across the network. The NIDS sensors are placed at crucial points in the network to inspect traffic from all devices on the network. For instance, NIDS sensors are installed on the subnet where firewalls are located to detect Denial of Service (DoS) and other such attacks.

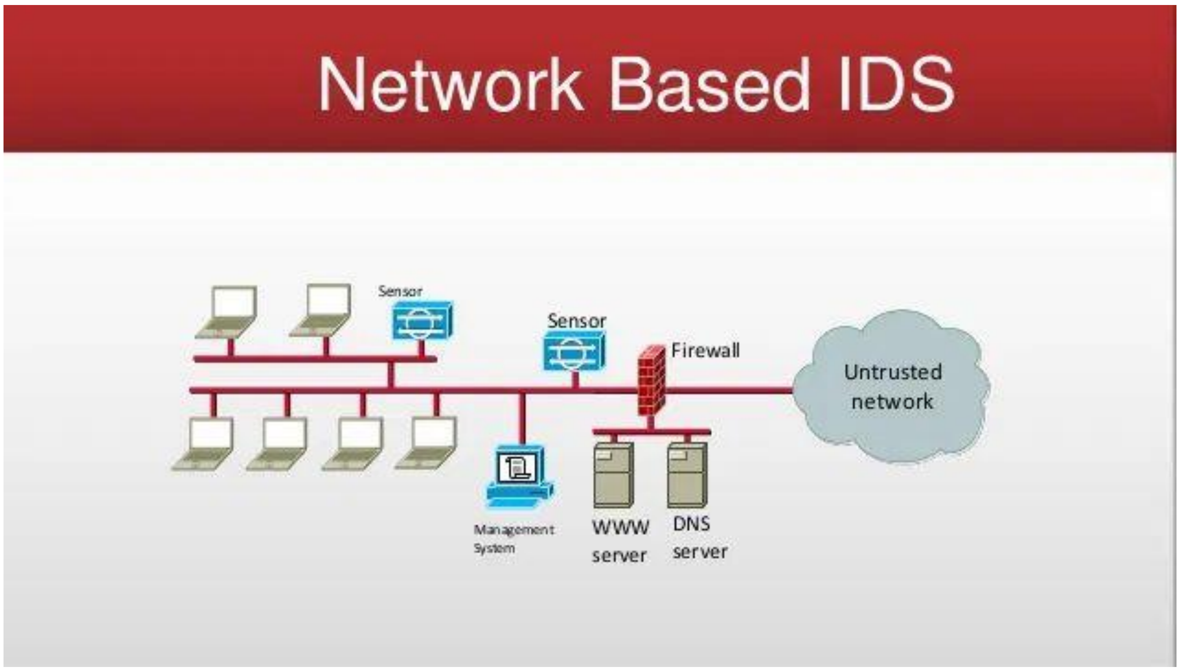


Figure 1 : Diagram representing Network based Intrusion Detection System

Host Intrusion Detection System

A host-based intrusion detection system (HIDS) monitors and analyzes system configuration and application activity for devices running on the enterprise network. The HIDS sensors can be installed on any device, regardless of whether it's a desktop PC or a server. HIDS sensors essentially take a snapshot of existing system files and compare them with previous snapshots. They look for unexpected changes, such as overwriting, deletion and access to certain ports.

Consequently, alerts are sent to administrators to investigate activities that seem iffy. They are a highly effective tool against insider threats. HIDS can identify file permission changes and unusual client-server requests, which generally tends to be a perfect concoction for internal attacks. That's why it should come as no surprise that HIDS is often used for mission-critical machines that are not expected to change.

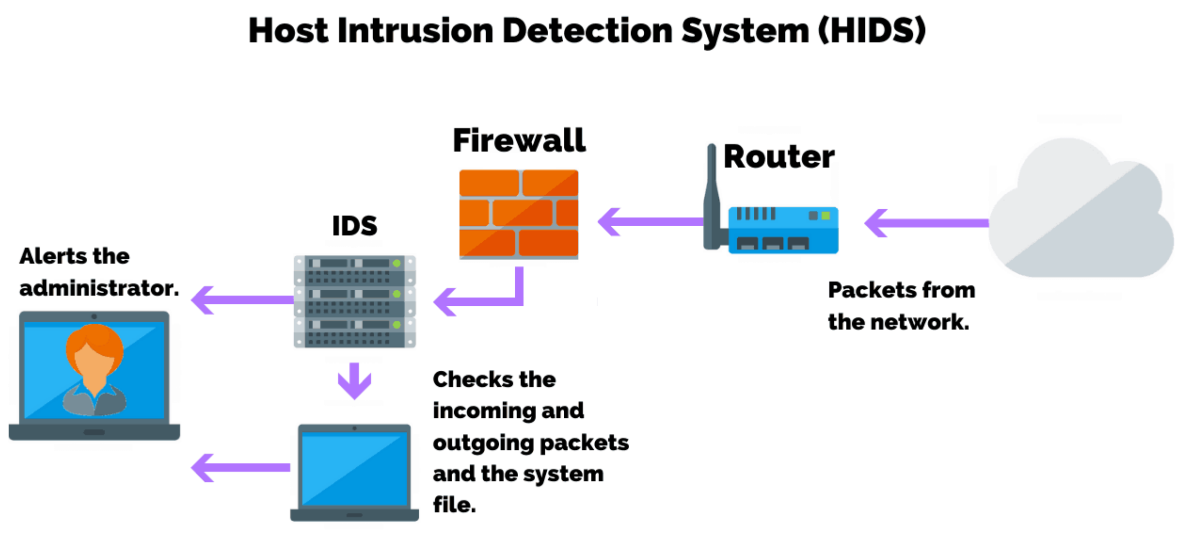

Host Intrusion Detection System (HIDS)
Firewall
Router
IDS
Alerts the administrator.
Packets from the network.
Checks the incoming and outgoing packets and the system file.

Figure 2: Diagram representing Host Intrusion Detection System

NIDS vs. HIDS: What's the Difference?

Each of these intrusion detection systems come with their own strengths. NIDS works in real-time, which means it tracks live data and flags issues as they happen. On the other hand, HIDS examines historical data to catch savvy hackers that use non-conventional methods that might be difficult to detect in real-time. The ideal scenario is to incorporate both HIDS and NIDS since they complement each other. NIDS offers faster response time while HIDS can identify malicious data packets that originate from inside the enterprise network.

1.4 INTRUSION DETECTION TECHNIQUES

Primarily there are two techniques to implement intrusion detection system, discussed as follows-

a) Anomaly-based intrusion detection techniques

Also called behaviour-based, these solutions track activity within the specific scope, looking for instances of malicious behaviour — at least, as they define it, which is a difficult job, and sometimes leads to false positives. For instance, outbound URLs of Web activity might be considered, and sites involving certain domains or URL length/contents might automatically be blocked, even though it's a human being trying to go there, and that user has a business-legitimate reason.

Anomaly-based IDS was introduced to detect unknown malware attacks as new malware are developed rapidly. In anomaly-based IDS there is use of machine learning to create a trustful activity model and anything coming is compared with that model and it is declared suspicious if it is not found in model. Machine learning-based method has a better-generalized property in comparison to signature-based IDS as these models can be trained according to the applications and hardware configurations.

b) Signature-based intrusion detection techniques

This approach, also known as knowledge-based, involves looking for specific signatures — byte combinations — that when they occur, almost invariably imply bad news. These solutions generate fewer false positives than anomaly solutions because the search criterion is so specific, but they also only cover signatures that are already in the search database (which means truly novel attacks have good odds of success).

Signature-based IDS detects the attacks on the basis of the specific patterns such as number of bytes or number of 1's or number of 0's in the network traffic. It also detects on the basis of the already known malicious instruction sequence that is used by the malware. The detected patterns in the IDS are known as signatures.

Signature-based IDS can easily detect the attacks whose pattern (signature) already exists in system but it is quite difficult to detect the new malware attacks as their pattern (signature) is not known.

In simple words we can conclude that signature-based IDS monitors the packets traversing the network, it compares these packets to the database of known IOCs or attack signatures to flag any suspicious behaviour.

On the other hand, anomaly-based intrusion detection systems can alert you to suspicious behaviour that is unknown.

Following is a diagram showing difference between the two techniques.

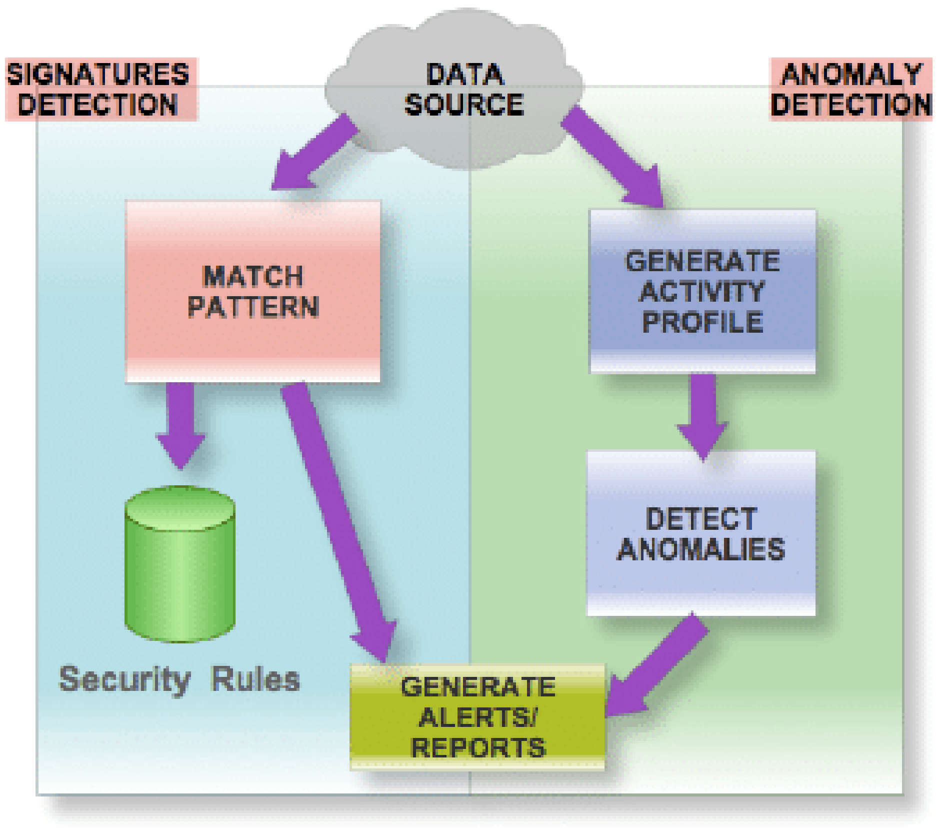


Figure 3: Diagrammatic comparison of Signature based and Anomaly based methods of IDS

1.5 Goals of Intrusion Detection System:

Following are seven major holistic goals of IDS

1. Enables SecOps

We talk about SecOps a lot, but it's not just lip service. We practice what we preach and employ a SecOps mentality — in which development, operations, and security work in sync at all times, rather than as separate teams or disciplines. When you choose an intrusion detection platform, it should enable security and DevOps to integrate their practices and thereby save time, increase efficiency, and improve your overall security posture.

2. Supports Complex Environments

For a variety of reasons, organizations do not always exist entirely in the cloud. Many find themselves with infrastructures that include cloud, multi-cloud, hybrid, on-premise, and containerized environments. To achieve security (and compliance), you need visibility, regardless of where your data resides. Visibility is the only way to make sure you know when a security event takes place so you can remediate it as quickly as possible. So when you go to choose an IDP, make sure you select one that offers visibility across complex environments, not just in the cloud.

3. Detects in Several Modes

Without a network perimeter to monitor, you need a variety of detection types to catch all the threats that will come your way. This includes the ability to detect at the levels of: behavior on the host, cloud configuration auditing, vulnerabilities, file integrity monitoring, and threat intelligence. A good IDP can do all of this and then some.

4. Identifies All Attacks

In addition to several types of detection, the ideal intrusion detection platform also needs the ability to detect multiple types and points of attack. This includes spotting both internal and external threats (which can look very different) and being able to detect an attack at many different stages, from initial reconnaissance, to exploitation, to vector-hopping. This is the best way to ensure that you catch any and all attacks, not just the ones you already know how to spot.

5. Alerts on Anomalous Behaviour

Anomalous behaviour is the best way to identify a threat in action (often before it even becomes a real threat). Strong IDP solutions should be able to baseline what is normal for your environment (which of course changes over time) so that it is possible to detect anomalous behaviour in real-time. From there, it should be a straightforward process for you to investigate the anomalous behaviour and determine whether it is harmless or indicative of an evolving threat.

6. Provides Unified Data

Your incident response depends heavily on your data collection capability. If you use multiple point solutions, you will have fragmented data points, and this will increase Mean Time to Resolution (MTTR). The goal is to keep MTTR low, and to do this, we need to invest in an IDP that unifies your data and provides a comprehensive picture of your infrastructure and all activity within it.

7. Maintains Compliance

Last but not least, we’d be remiss not to mention that a strong IDP will support continuous compliance with major standards like PCI DSS, HIPAA, SOC 2, and more. While many platforms support security, some do not also provide the controls, monitoring, and auditability necessary to maintain compliance with these complex and necessary frameworks. Do yourself a favour and invest in an IDP that will make life easier when it comes to compliance.

CHAPTER 2: LITERATURE SURVEY

Two most significant motives to launch attacks are, either to force a network to stop some service(s) that it is providing or to steal some information stored in a network. An intrusion detection system must be able to detect such anomalous activities. However, what is normal and what is anomalous is not defined, an event may be considered normal with respect to some criteria, but the same may be labelled anomalous when this criterion is changed. applies to values inside the interval, i.e., all will be viewed as nor-malto the same degree. Unfortunately, this causes an abrupt separation between normality and anomaly. With the fuzzy input sets defined, the next step is to write the rules to, identify each type of attack. A collection of fuzzy rules with the same input and output variables is called a fuzzy system. We believe the security administrators can use their expert knowledge to help create a set of rules for each attack. The rules are created using the fuzzy system editor contained in the MATLAB Fuzzy Toolbox. This tool contains a graphical user interface that allows the rule designer to create the member functions for each input or output variable, create the inference relationships between the various member functions and to examine the control surface for the resulting fuzzy system. It is not expected, that the rule designer utterly relies on intuition to create the rules. Visual data mining can assist the rule designer in knowing which data features are most appropriate and relevant in detecting different kinds of attacks. The goal for using ANNS for intrusion detection is to be able to generalize from incomplete data and to be able to classify data as being normal or intrusive. An ANN consists of a collection of processing elements that are highly interconnected. Given a set of inputs and a set of desired outputs, the transformation from input to output is determined by the weights associated with the inter-connections among processing elements. By modifying these interconnections, the network is able to adapt to desired outputs. The ability of high tolerance for learning-by-example makes neural networks flexible and powerful in IDS.

In the literary of CAPTCHAS, most schemes were aimed at the Turing test that embeds characters in an image. However, illustrated that computer vision techniques by optical character recognition, have over 90% accuracy to recognize the character in an image. To improve the strength of a character image against to a program, tries to add more noise and distortion, but this will be harder for a human to recognize the characters too. Thus, adding too much noise and distortion will make the characters image to be unusable. Furthermore, proposed alternative image question CAPTCHAS which does not have the above issue and provided a combination of character and image CAPTCHA which possesses both of the above properties and users have to do simple mathematical computation in order to answer the question Two approaches to intrusion detection are currently used. The first one, called misuse detection, is based on attack signatures, i.e. on a detailed description of the sequence of actions performed by the attacker. This approach allows the detection of intrusions matching perfectly the signatures, so that, new attacks performed by slight modification of known at tacks cannot be detected.

CHAPTER 3:
PROBLEM STATEMENT

In this project, we have taken up UNSW-NB15 dataset, which is a combination of intrusive sample and normal sample.
Then we have performed intrusion detection on this dataset. For doing so, we have pre-processed the dataset, used feature extraction method and used three classification algorithms -
1) Random Forest algorithm
2) Decision Tree algorithm
3) SVM algorithm
The three results thus obtained are compared with each other to decide the best performing classification algorithm based on accuracy and confusion matrix.

Rationale for procurement of UNSW-NB15 dataset :

The dataset used is UNSW-NB15 dataset. This dataset is an improved modern alternative of the classic KDD cup dataset, as KDD cup dataset has become outdated and had a lot of anomalies. This dataset is class wise more balanced as compared to other datasets available for intrusion detection. The UNSW-NB15 dataset although less in size as compared to others, but even less redundant, is sufficient to train a model with high accuracy. This dataset has a total of 45 features in the dataset which helps in accurately classifying the targets.

CHAPTER 4:

SOFTWARE REQUIREMENT SPECIFICATION

We have used the following software tools in our project:

1. Python
2. Google Colaboratory / Jupyter Notebook
3. UNSW-NB15 dataset
4. Pandas
5. System Architecture

4.1 PYTHON

Python is often used as a support language for software developers, for build control and management, testing, and in many other ways. The following points summarize why python is a favourable option and well suited for machine learning.

1. Python is easy and simple

The most important feature of Python for machine learning is that it does not need any hardcore programmer to put effort into it.
Many pieces of research verify that the semantics of Python have correspondence to numerous mathematical ideas. As a result, it does not take much to learn to apply those mathematical ideas in this informal language. It is humble with an effortlessly comprehensible syntax.
The straightforwardness of Python lets the developer focus on actually solving the ML complications, rather than setting the bugs.

2. Python is efficient

Python for machine learning is supremely efficient. The code of Python is comfortably sensible by individuals. It performs the bulky task using fewer line of code, doing this one can save the memory space.

3. Python has diverse libraries and framework

Python for machine learning is an abundant source of libraries and frameworks. It comes with plenty of integral libraries. It comprises of some special tools that are extremely useful while working with the Machine learning system.
These libraries are Keras (which particularly throw light on experimentation with deep neural networks), TensorFlow (which is applicable for many Machine Learning applications), etc. as the list keeps going and never ends.

4. Python is versatile

Though Python has crossed more than two decades, it is still flattering extra versatile over time. Considering today’s situation, one can implement Python on almost all software, actions, infrastructure, etc.
Incredible data handling tool is also served by Python, making Python can deal with a huge amount of data.

5. Python has a vast community

Since Python’s release, the community is largely supporting Python learners. They help learners improve their Machine Learning knowledge and never fell abandoned to cope-up with sudden changes.

6. Python is portable and extensible

Python programs can work on any other platforms without making any significant changes. Many developers prefer Graphics Processing Units (GPU) to train their ML models on their machines, hence giving this feature height in the sky.
Python can be extended to supplementary languages, i.e., one can write Python code in any other languages as well.

7. Python is flexible

Python Developers have the right to choose between OOPs and scripting. Developers can choose the programming style they are used to, else they can combine these styles to crack different snags in a well-organized manner.

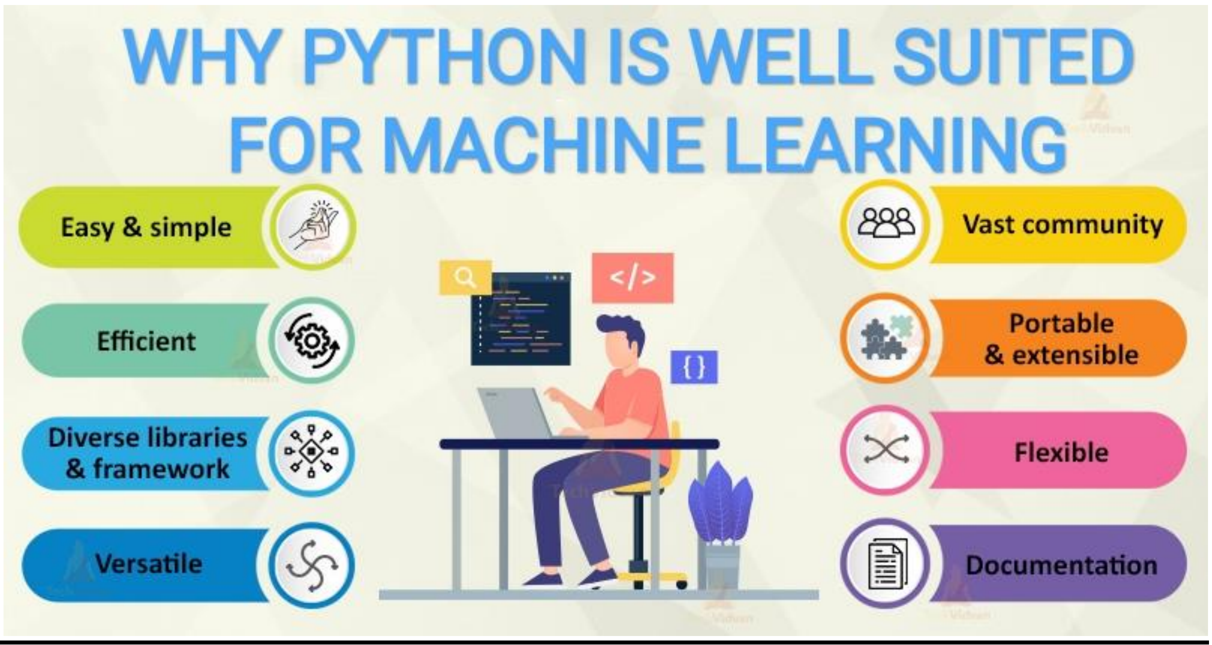


Figure 4: Suitability of python for machine learning

4.2 Google Colaboratory

Colaboratory, or Colab for short, is a Google Research product, which allows developers to write and execute Python code through their browser. Google Colab is an excellent tool for deep learning tasks. It is a hosted Jupyter notebook that requires no setup and has an excellent free version, which gives free access to Google computing resources such as GPUs and TPUs.

Colaboratory, or "Colab" for short, is a product from Google Research. Colab allows anybody to write and execute arbitrary python code through the browser, and is especially well suited to machine learning, data analysis and education.

There are several reasons to opt to use Google Colab :

(a)Pre-Installed Libraries

Anaconda distribution of Jupyter Notebook shipped with several pre-installed data libraries, such as Pandas, NumPy, Matplotlib, which is awesome. Google Colab, on the other hand, provides even more pre-installed machine learning libraries such as Keras, TensorFlow, and PyTorch.

(b)Saved on the Cloud

When we opt to use a plain Jupyter notebook as your development environment, everything is saved in our local machine. If we are cautious about privacy, this may be a preferred feature for us. However, if we want your notebooks to be accessible to us from any device with a simple Google log-in, then Google Colab is the way to go. All of our Google Colab notebooks are saved under our Google Drive account, just like our Google Docs and Google Sheets files.

(c) Collaboration

Another great feature that Google Colab offers is the collaboration feature. If you are working with multiple developers on a project, it is great to use Google Colab notebook. Just like collaborating on a Google Docs document, you can co-code with multiple developers using a Google Colab notebook. Besides, you can also share your completed work with other developers.

(d) Free GPU and TPU Use

I think this is an absolute no brainer to choose Google Colab instead of a local Jupyter notebook. Google Research lets us use their dedicated GPUs and TPUs for our personal machine learning projects.

For some projects, the GPU and TPU acceleration make a huge difference even for some small projects. This is one of the main reasons to code educational projects on Google Colab. Besides, since it uses Google resources, the neural network optimization operations do not mess with my processors, and my cooling fan doesn't go crazy.

In, nutshell we can say that Google Colab is just a specialized version of the Jupyter Notebook, which runs on the cloud and offers free computing resources. The relationship between iPython, Jupyter Notebook, and Google Colab

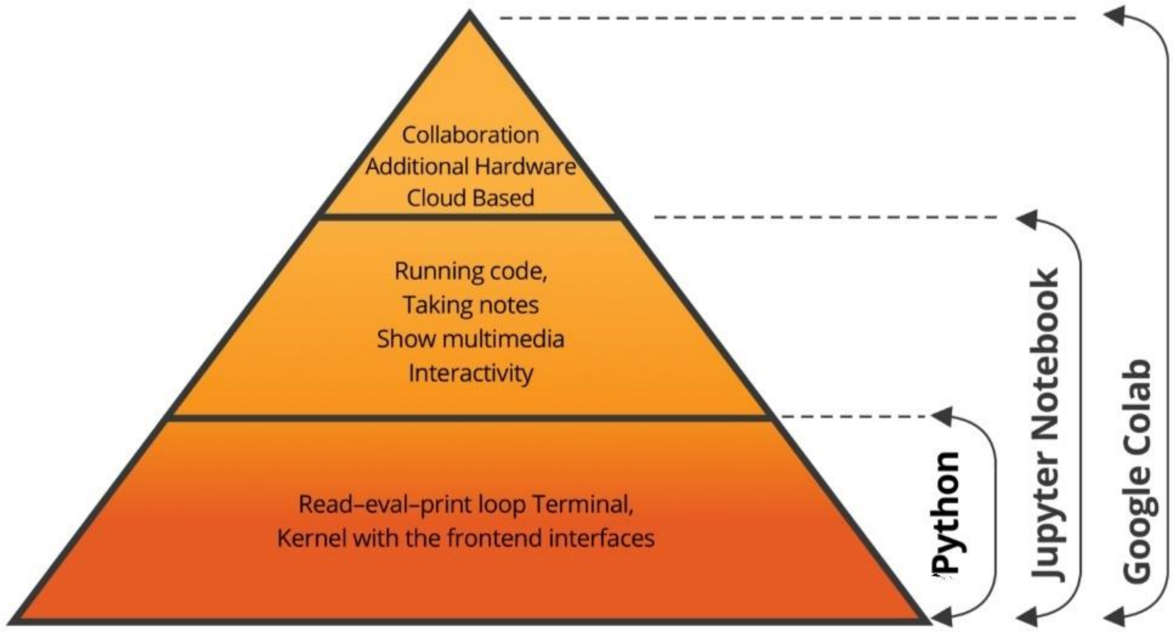


Figure 5: The relation between Python, Jupyter Notebook and Google Colab

4.3 UNSW Dataset

UNSW-NB15 is a network intrusion dataset. It contains nine different attacks, includes DoS, worms, Backdoors, and Fuzzers. The dataset contains raw network packets. The number of records in the training set is 175,341 records and the testing set is 82,332 records from the different types, attack and normal. We have used a subset of this dataset.

The existing datasets do not represent the comprehensive representation of the modern orientation of network traffic and attack scenarios. These reasons have instigated a serious challenge for the cyber security research group at the Australian Centre for Cyber Security (ACCS) and other researchers of this domain around the globe. The raw network packets of the UNSW-NB15 dataset were created by the IXIA PerfectStorm tool in the Cyber Range Lab of ACCS for generating a hybrid of real modern normal activities and synthetic contemporary attack behaviours.

4.4 PANDAS

Pandas is an open-source python package built on top of Numpy developed by Wes McKinney**.** It is used as one of the most important data cleaning and analysis tool. It provides fast, flexible, and expressive data structures.

Data analysis requires lots of processing, such as restructuring, cleaning or merging, etc. There are different tools are available for fast data processing, such as Numpy, Scipy, Cython, and Panda**.** But we prefer Pandas because working with Pandas is fast, simple and more expressive than other tools.

Key Features of Pandas

- Fast and efficient DataFrame object with default and customized indexing.
- Tools for loading data into in-memory data objects from different file formats.
- Data alignment and integrated handling of missing data.
- Reshaping and pivoting of date sets.
- Label-based slicing, indexing and subsetting of large data sets.
- Columns from a data structure can be deleted or inserted.
- Group by data for aggregation and transformations.
- High performance merging and joining of data.
- Time Series functionality.

4.5 SYSTEM ARCHITECTURE

System Architecture is abstract, conceptualization-oriented, global, and focused to achieve the mission and life cycle concepts of the system. It also focuses on high-level structure in systems and system elements. It addresses the architectural principles, concepts, properties, and characteristics of the system-of-interest. It may also be applied to more than one system, in some cases forming the common structure, pattern, and set of requirements for classes or families of similar or related systems.

The purpose of the System Architecture process is to generate system architecture alternatives, to select one or more alternative(s) that frame stakeholder concerns and meet system requirements, and to express this in a set of consistent views.

It should be noted that the architecture activities below overlap with both system definition and concept definition activities. In particular, key aspects of the operational and business context, and hence certain stakeholder needs, strongly influence the approach taken to architecture development and description. Also, the architecture activities will drive the selection of, and fit within, whatever approach to solution synthesis has been selected.

CHAPTER 5: CLASSIFIERS

5.1 Description

Machine learning algorithms are helpful to automate tasks that previously had to be done manually. They can save huge amounts of time and money and make businesses more efficient.
A classifier in machine learning is an algorithm that automatically orders or categorizes data into one or more of a set of “classes.” One of the most common examples is an email classifier that scans emails to filter them by class label: Spam or Not Spam.

Intrusion detection can be thought of as a classification problem where each audit record can be classified into one of a discrete set of possible categories, normal or a particular
kind of intrusion. Intrusion detection using data mining have attracted more and more interests in recent years. As an important application of data mining, they aim to meliorate the great burden of analyzing huge volumes of audit data and realizing performance optimization of detection rules.

The three classifiers that we have used in out project are as follows-

1) Support Vector Machine
2) Random Forest Classifier
3) Decision Tree Classifier

5.2 Decision Tree Classifier

Decision Trees are a type of Supervised Machine Learning (that is you explain what the input is and what the corresponding output is in the training data) where the data is continuously split according to a certain parameter. The tree can be explained by two entities, namely decision nodes and leaves. The leaves are the decisions or the final outcomes. And the decision nodes are where the data is split. A decision tree is a supervised machine learning classification algorithm used to build models like the structure of a tree. It classifies data into finer and finer categories: from “tree trunk,” to “branches,” to “leaves.” It uses the if-then rule of mathematics to create sub-categories that fit into broader categories and allows for precise, organic categorization.

For example, this is how a decision tree would categorize individual sports, as shown on the next page:

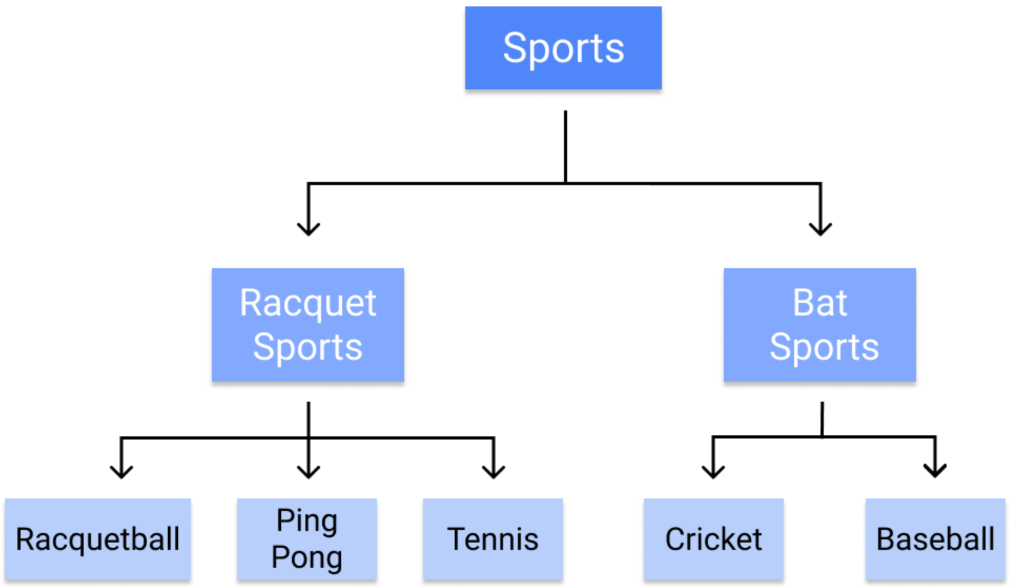


Figure 6: Example of Decision Tree classification algorithm

As the rules are learned sequentially, from trunk to leaf, a decision tree requires high quality, clean data from the outset of training, or the branches may become over-fitted or skewed.

5.3 Random Forest Classifier

Random Forest is a classifier that contains a number of decision trees on various subsets of the given dataset and takes the average to improve the predictive accuracy of that dataset." Instead of relying on one decision tree, the random forest takes the prediction from each tree and based on the majority votes of predictions, and it predicts the final output. The greater number of trees in the forest leads to higher accuracy and prevents the problem of overfitting.

Since the random forest combines multiple trees to predict the class of the dataset, it is possible that some decision trees may predict the correct output, while others may not. But together, all the trees predict the correct output. Therefore, below are two assumptions for a better Random forest classifier:

- There should be some actual values in the feature variable of the dataset so that the classifier can predict accurate results rather than a guessed result.
- The predictions from each tree must have very low correlations.

The main benefits of Random Forest classifier are as noted below:

- Random Forest is capable of performing both Classification and Regression tasks.
- It is capable of handling large datasets with high dimensionality.
- It enhances the accuracy of the model and prevents the overfitting issue.

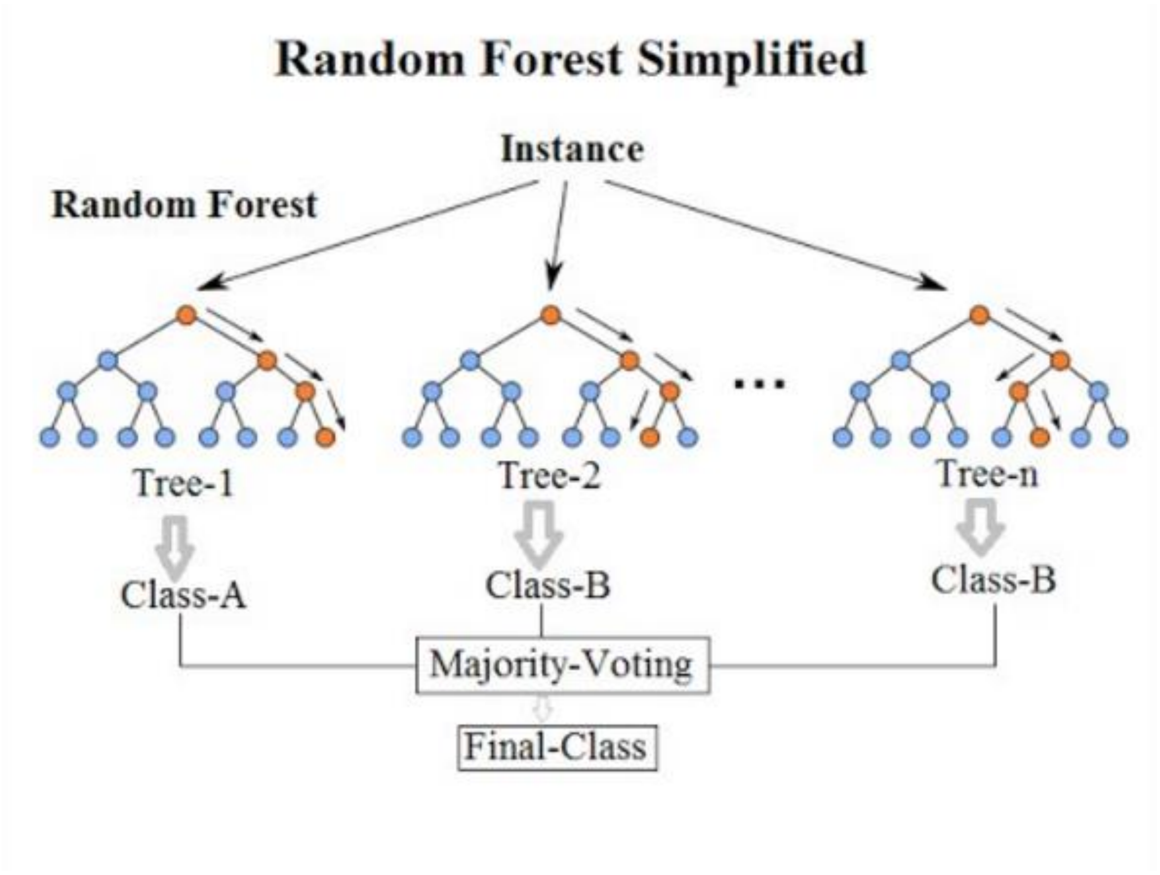


Figure 7: Diagram for representation of Random Forest Classifier

Implementation of Random Forest Algorithm in python includes the following steps:

- Data Pre-processing step
- Fitting the Random forest algorithm to the Training set
- Predicting the test result
- Test accuracy of the result (Creation of Confusion matrix)
- Visualizing the test set result.

5.4 Support Vector Machine

Support vector machines are a set of supervised learning methods used for classification, regression, and outliers detection. All of these are common tasks in machine learning. SVMs are different from other classification algorithms because of the way they choose the decision boundary that maximizes the distance from the nearest data points of all the classes. The decision boundary created by SVMs is called the maximum margin classifier or the maximum margin hyper plane.

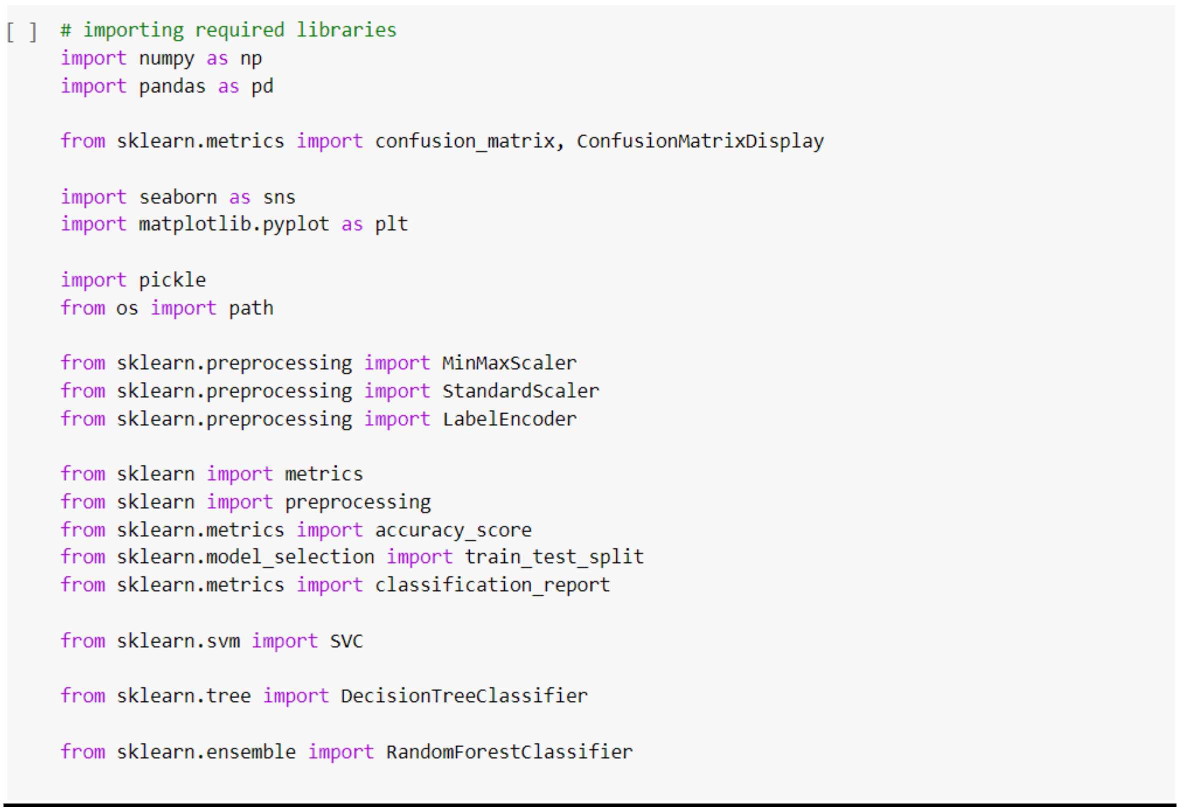
SVMs are used in applications like handwriting recognition, intrusion detection, face detection, email classification, gene classification, and in web pages. This is one of the reasons we use SVMs in machine learning. It can handle both classification and regression on linear and non-linear data. Another reason we use SVMs is because they can find complex relationships between your data without you needing to do a lot of transformations on your own. It's a great option when you are working with smaller datasets that have tens to hundreds of thousands of features. They typically find more accurate results when compared to other algorithms because of their ability to handle small, complex datasets.

CHAPTER 6: PRE PROCESSING

6.1 Importing Modules

Here we have imported these following modules:
1. Numpy and Pandas
2. Confusion matrix
3. Pickle
4. Min Max scaler, Standard scaler and Label encoder
5. SVC (svm), Random forest and Decision tree

A visual representation pf the same is shown below

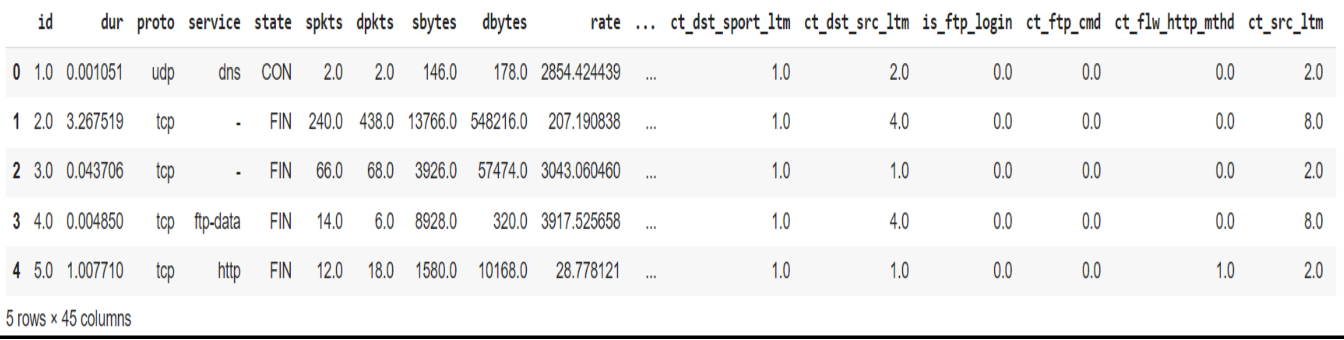


```
[ ]  # importing required libraries
     import numpy as np
     import pandas as pd

     from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay

     import seaborn as sns
     import matplotlib.pyplot as plt

     import pickle
     from os import path

     from sklearn.preprocessing import MinMaxScaler
     from sklearn.preprocessing import StandardScaler
     from sklearn.preprocessing import LabelEncoder

     from sklearn import metrics
     from sklearn import preprocessing
     from sklearn.metrics import accuracy_score
     from sklearn.model_selection import train_test_split
     from sklearn.metrics import classification_report

     from sklearn.svm import SVC

     from sklearn.tree import DecisionTreeClassifier

     from sklearn.ensemble import RandomForestClassifier
```

Figure 8: Visual representation of importing of modules

6.2 Importing Dataset

As we have used UNSW-NB15 dataset, the following figure 9 shows the data before the dataset has been pre-processed and trimmed.

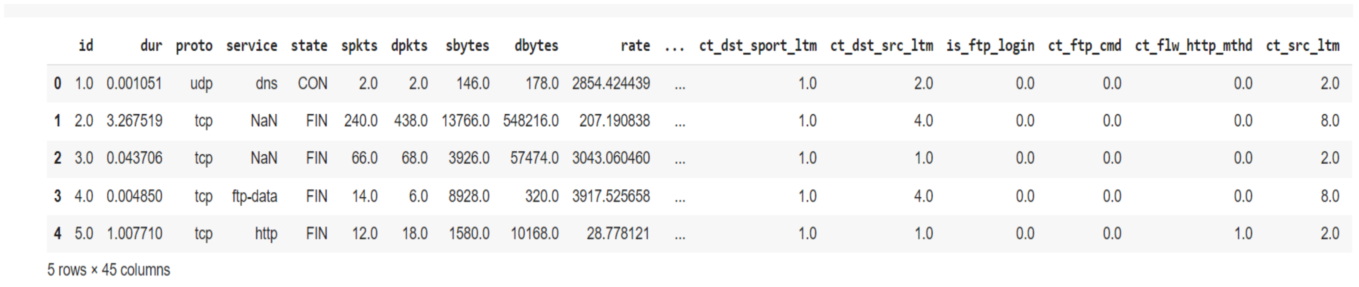


	id	dur	proto	service	state	spkts	dpkts	sbytes	dbytes	rate	...	ct_dst_sport_ltm	ct_dst_src_ltm	is_ftp_login	ct_ftp_cmd	ct_flw_http_mthd	ct_src_ltm
0	1.0	0.001051	udp	dns	CON	2.0	2.0	146.0	178.0	2854.424439	...	1.0	2.0	0.0	0.0	0.0	2.0
1	2.0	3.267519	tcp	-	FIN	240.0	438.0	13766.0	548216.0	207.190838	...	1.0	4.0	0.0	0.0	0.0	8.0
2	3.0	0.043706	tcp	-	FIN	66.0	68.0	3926.0	57474.0	3043.060460	...	1.0	1.0	0.0	0.0	0.0	2.0
3	4.0	0.004850	tcp	ftp-data	FIN	14.0	6.0	8928.0	320.0	3917.525658	...	1.0	4.0	0.0	0.0	0.0	8.0
4	5.0	1.007710	tcp	http	FIN	12.0	18.0	1580.0	10168.0	28.778121	...	1.0	1.0	0.0	0.0	1.0	2.0

5 rows × 45 columns

Figure 9: Dataset before being pre-processed and trimmed

The following figure 10 shows the result of replacing the Hyphen '-' (No value) in service feature with NaN (Not a Number)



	id	dur	proto	service	state	spkts	dpkts	sbytes	dbytes	rate	...	ct_dst_sport_ltm	ct_dst_src_ltm	is_ftp_login	ct_ftp_cmd	ct_flw_http_mthd	ct_src_ltm
0	1.0	0.001051	udp	dns	CON	2.0	2.0	146.0	178.0	2854.424439	...	1.0	2.0	0.0	0.0	0.0	2.0
1	2.0	3.267519	tcp	NaN	FIN	240.0	438.0	13766.0	548216.0	207.190838	...	1.0	4.0	0.0	0.0	0.0	8.0
2	3.0	0.043706	tcp	NaN	FIN	66.0	68.0	3926.0	57474.0	3043.060460	...	1.0	1.0	0.0	0.0	0.0	2.0
3	4.0	0.004850	tcp	ftp-data	FIN	14.0	6.0	8928.0	320.0	3917.525658	...	1.0	4.0	0.0	0.0	0.0	8.0
4	5.0	1.007710	tcp	http	FIN	12.0	18.0	1580.0	10168.0	28.778121	...	1.0	1.0	0.0	0.0	1.0	2.0

5 rows × 45 columns

Figure 10: Result of replacing the Hyphen (No value) in service feature with NaN (Not a Number)

6.3 Data Processing

Data in its raw form is not useful to any organization. Data processing is the method of collecting raw data and translating it into usable information. It is usually performed in a step-by-step process by a team of data scientists and data engineers in an organization. The raw data is collected, filtered, sorted, processed, analyzed, stored, and then presented in a readable format.

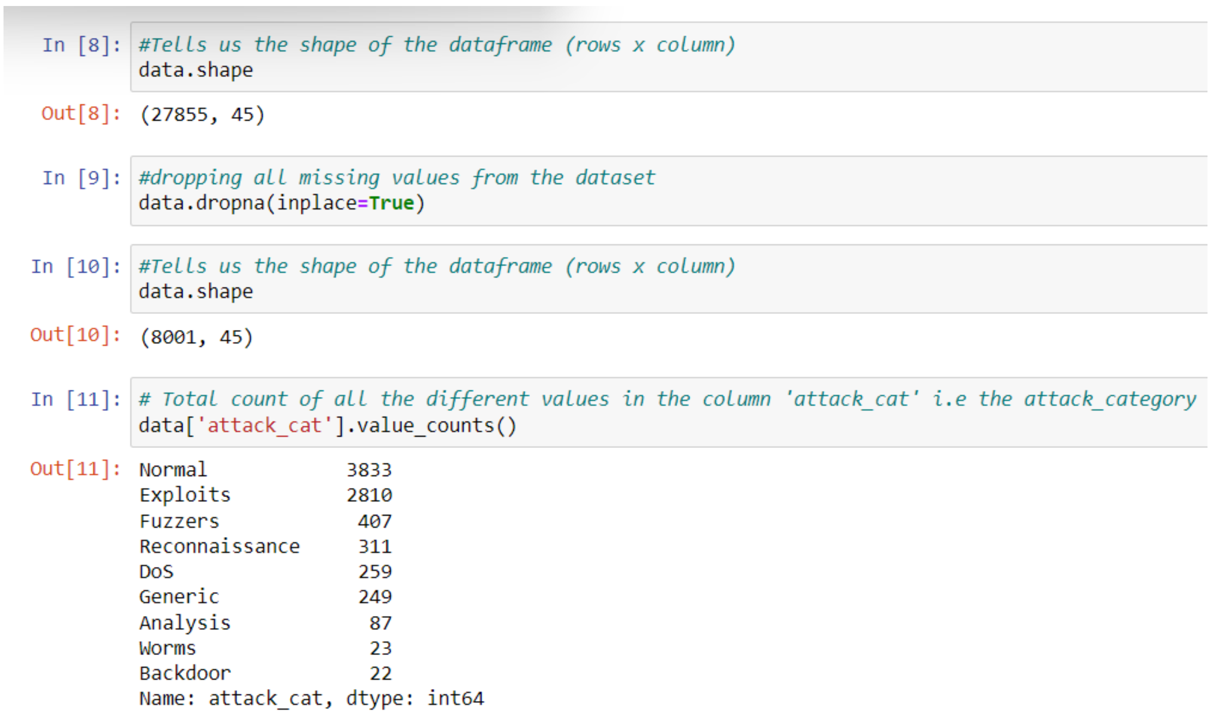
The data processing cycle consists of a series of steps where raw data (input) is fed into a process (CPU) to produce actionable insights (output). Each step is taken in a specific order, but the entire process is repeated in a cyclic manner. The first data processing cycle's output can be stored and fed as the input for the next cycle.

Data processing cycle

- It refers to the sequence of activities involved in data transformation from its row form to information. it is often referred to as cycle because the output obtained can be stored after processing and may be used in future as input.
- The four main stages of data processing cycle are:
  - Data collection
  - Data input
  - Data processing
  - Data output

1. **Data collection**

- Also referred to as data gathering or fact finding, it involves looking for crucial facts needed for processing.
- Includes interviews, use of questionnaires, observation, etc. In most cases the data is collected after sampling



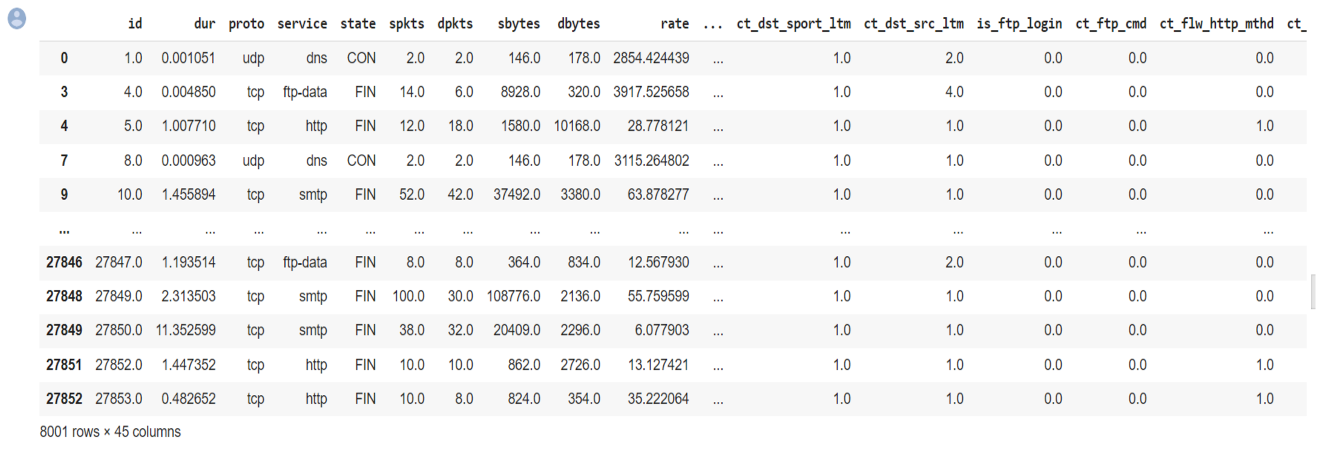
```
In [8]:  #Tells us the shape of the dataframe (rows x column)
         data.shape

Out[8]:  (27855, 45)

In [9]:  #dropping all missing values from the dataset
         data.dropna(inplace=True)

In [10]: #Tells us the shape of the dataframe (rows x column)
         data.shape

Out[10]: (8001, 45)

In [11]: # Total count of all the different values in the column 'attack_cat' i.e the attack_category
         data['attack_cat'].value_counts()

Out[11]: Normal            3833
         Exploits          2810
         Fuzzers            407
         Reconnaissance     311
         DoS                259
         Generic            249
         Analysis            87
         Worms               23
         Backdoor            22
         Name: attack_cat, dtype: int64
```

Figure 11: Dropping null values and observing the change in shape of the dataframe

2.Data Input- The Data Input Plan will identify the data to be included and the means for inputting such data (direct entry or otherwise), the specific fields of information to be included, the past time periods for which information is to be included, the deadlines for inputting the data, and the responsibility for the input of the data.

3.Data processing- Concept of Data processing is collecting and manipulating data into a usable and appropriate form. The automatic processing of data in a predetermined sequence of operations is the manipulation of data. The processing nowadays is automatically done by using computers, which is faster and gives accurate results

After initial processing all the data, the result obtained is as shown in figure 12.



```
          id        dur proto  service state spkts dpkts   sbytes   dbytes        rate ... ct_dst_sport_ltm ct_dst_src_ltm is_ftp_login ct_ftp_cmd ct_flw_http_mthd ct_
    0      1.0  0.001051   udp      dns   CON   2.0   2.0    146.0    178.0 2854.424439 ...              1.0            2.0          0.0        0.0              0.0
    3      4.0  0.004850   tcp ftp-data   FIN  14.0   6.0   8928.0    320.0 3917.525658 ...              1.0            4.0          0.0        0.0              0.0
    4      5.0  1.007710   tcp     http   FIN  12.0  18.0   1580.0  10168.0   28.778121 ...              1.0            1.0          0.0        0.0              1.0
    7      8.0  0.000963   udp      dns   CON   2.0   2.0    146.0    178.0 3115.264802 ...              1.0            1.0          0.0        0.0              0.0
    9     10.0  1.455894   tcp     smtp   FIN  52.0  42.0  37492.0   3380.0   63.878277 ...              1.0            1.0          0.0        0.0              0.0
  ...      ...       ...   ...      ...   ...   ...   ...      ...      ...         ... ...              ...            ...          ...        ...              ...
27846  27847.0  1.193514   tcp ftp-data   FIN   8.0   8.0    364.0    834.0   12.567930 ...              1.0            2.0          0.0        0.0              0.0
27848  27849.0  2.313503   tcp     smtp   FIN 100.0  30.0 108776.0   2136.0   55.759599 ...              1.0            1.0          0.0        0.0              0.0
27849  27850.0 11.352599   tcp     smtp   FIN  38.0  32.0  20409.0   2296.0    6.077903 ...              1.0            1.0          0.0        0.0              0.0
27851  27852.0  1.447352   tcp     http   FIN  10.0  10.0    862.0   2726.0   13.127421 ...              1.0            1.0          0.0        0.0              1.0
27852  27853.0  0.482652   tcp     http   FIN  10.0   8.0    824.0    354.0   35.222064 ...              1.0            1.0          0.0        0.0              1.0
8001 rows × 45 columns
```

Figure 12: Result screen after initial Data Processing

4. Data Output- Data Output means the maps, reports or other information that you generate by analyzing or processing Data Products, including geocode coordinates or cluster segmentation assignments appended to your database records. “Derived Data” means the result generated by combining, or performing mathematical calculations on, variables or fields of Data Products, extracting subsets of Data Products, and combining Data Products or portions of them with your data or third party data.

6.4 Data Visualization

Data visualization is the practice of translating information into a visual context, such as a map or graph, to make data easier for the human brain to understand and pull insights from. The main goal of data visualization is to make it easier to identify patterns, trends and outliers in large data sets.

Our eyes are drawn to colours and patterns. We can quickly identify red from blue, square from circle. Our culture is visual, including everything from art and advertisements to TV and movies. Data visualization is another form of visual art that grabs our interest and keeps our eyes on the message. When we see a chart, we quickly see trends and outliers. If we can see something, we internalize it quickly. It’s storytelling with a purpose.

Uses of data visualization: Data Visualization helps users to analyze and interpret the data easily. It makes complex data understandable and usable. Various Charts in Data Visualization helps to show relationships in the data for one or more variables.

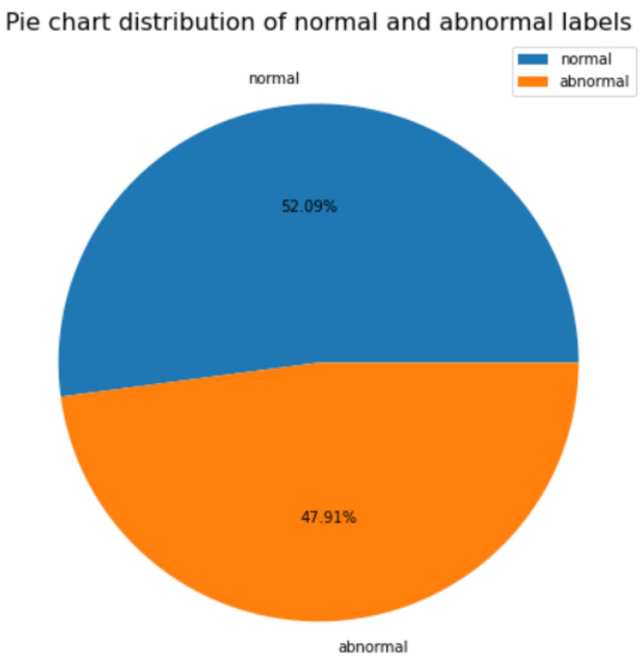


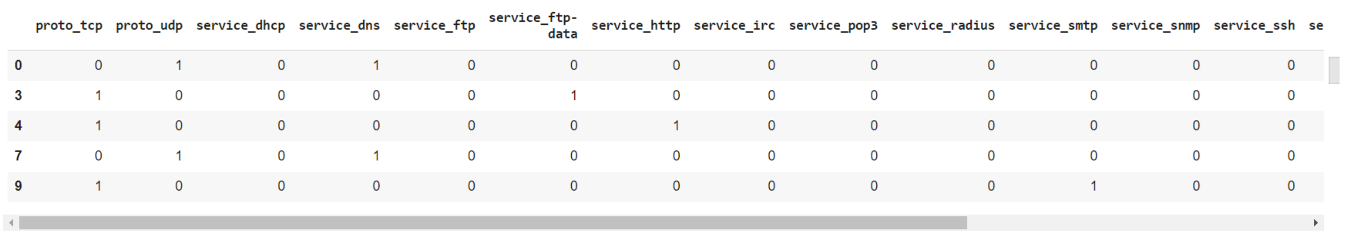
Figure 13: Pie chart distribution of normal and abnormal labels

6.5 One Hot Encoding

For categorical variables where no such ordinal relationship exists, the integer encoding is not enough.

In fact, using this encoding and allowing the model to assume a natural ordering between categories may result in poor performance or unexpected results (predictions halfway between categories).

In this case, a one-hot encoding can be applied to the integer representation. This is where the integer encoded variable is removed and a new binary variable is added for each unique integer value.



	proto_tcp	proto_udp	service_dhcp	service_dns	service_ftp	service_ftp-data	service_http	service_irc	service_pop3	service_radius	service_smtp	service_snmp	service_ssh	se
0	0	1	0	1	0	0	0	0	0	0	0	0	0	
3	1	0	0	0	0	1	0	0	0	0	0	0	0	
4	1	0	0	0	0	0	1	0	0	0	0	0	0	
7	0	1	0	1	0	0	0	0	0	0	0	0	0	
9	1	0	0	0	0	0	0	0	0	0	1	0	0	

Figure 14: Data in One Hot Encoding

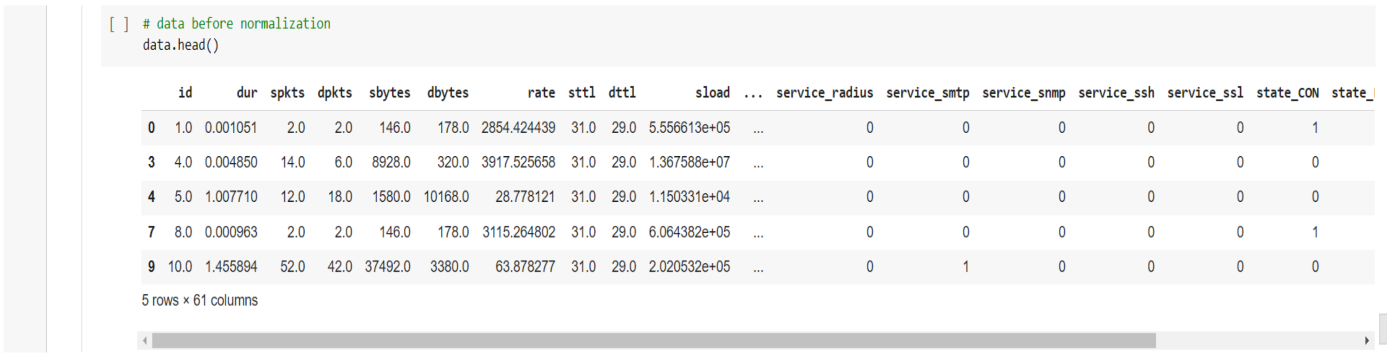
6.6 Data Normalization

Data normalization is the process of reorganizing data within a database so that users can utilize it for further queries and analysis. Simply put, it is the process of developing clean data. This includes eliminating redundant and unstructured data and making the data appear similar across all records and fields.

As data becomes more and more valuable to any type of business, data normalization is more than just reorganizing the data in a database. Here are some of its major benefits:

- Reduces redundant data
- Provides data consistency within the database
- More flexible database design
- Higher database security
- Better and quicker execution

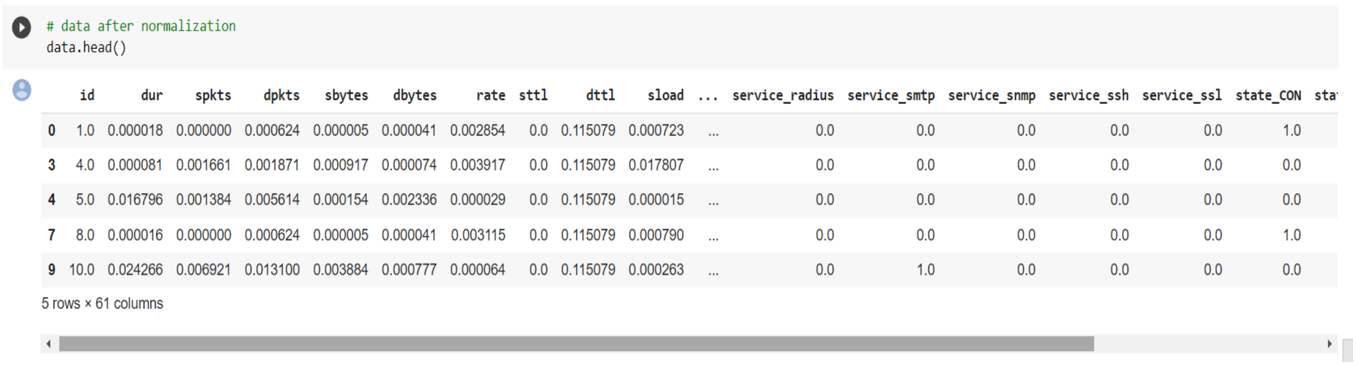
Following Figure 15 and Figure 16 show the results of dataset before and after data normalization respectively



```
[ ] # data before normalization
    data.head()
```

```
     id       dur  spkts  dpkts   sbytes   dbytes         rate  sttl  dttl         sload  ...  service_radius  service_smtp  service_snmp  service_ssh  service_ssl  state_CON  state_
0   1.0  0.001051    2.0    2.0    146.0    178.0  2854.424439  31.0  29.0  5.556613e+05  ...               0             0             0            0            0          1
3   4.0  0.004850   14.0    6.0   8928.0    320.0  3917.525658  31.0  29.0  1.367588e+07  ...               0             0             0            0            0          0
4   5.0  1.007710   12.0   18.0   1580.0  10168.0    28.778121  31.0  29.0  1.150331e+04  ...               0             0             0            0            0          0
7   8.0  0.000963    2.0    2.0    146.0    178.0  3115.264802  31.0  29.0  6.064382e+05  ...               0             0             0            0            0          1
9  10.0  1.455894   52.0   42.0  37492.0   3380.0    63.878277  31.0  29.0  2.020532e+05  ...               0             1             0            0            0          0

5 rows × 61 columns
```

Figure 15: Data set before data normalization



```
# data after normalization
data.head()
```

```
     id       dur     spkts     dpkts    sbytes    dbytes      rate  sttl      dttl     sload  ...  service_radius  service_smtp  service_snmp  service_ssh  service_ssl  state_CON  sta
0   1.0  0.000018  0.000000  0.000624  0.000005  0.000041  0.002854   0.0  0.115079  0.000723  ...             0.0           0.0           0.0          0.0          0.0        1.0
3   4.0  0.000081  0.001661  0.001871  0.000917  0.000074  0.003917   0.0  0.115079  0.017807  ...             0.0           0.0           0.0          0.0          0.0        0.0
4   5.0  0.016796  0.001384  0.005614  0.000154  0.002336  0.000029   0.0  0.115079  0.000015  ...             0.0           0.0           0.0          0.0          0.0        0.0
7   8.0  0.000016  0.000000  0.000624  0.000005  0.000041  0.003115   0.0  0.115079  0.000790  ...             0.0           0.0           0.0          0.0          0.0        1.0
9  10.0  0.024266  0.006921  0.013100  0.003884  0.000777  0.000064   0.0  0.115079  0.000263  ...             0.0           1.0           0.0          0.0          0.0        0.0

5 rows × 61 columns
```

Figure 16: Data set after data normalization

CHAPTER 7: RESULTS

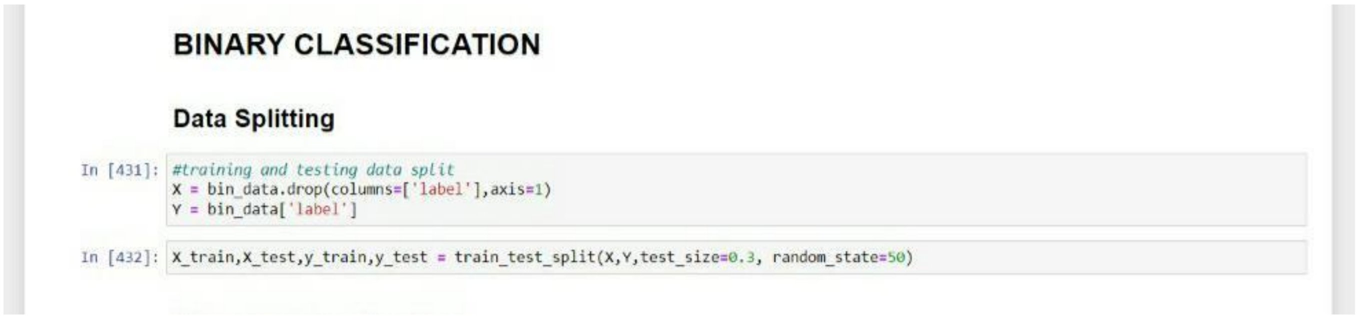
In this project, we have used binary classification method and, in this method, we have used three classifier algorithms.

Binary Classification

Binary classification is a form of classification. In binary classification the process of predicting categorical variables where the output is restricted to two classes.

Classification into one of two classes is a common machine learning problem. We might want to predict whether or not a customer is likely to make a purchase, whether or not a credit card transaction was fraudulent, whether deep space signals show evidence of a new planet, or a medical test evidence of a disease. These are all binary classification problems.

In your raw data, the classes might be represented by strings like "Yes" and "No", or "Dog" and "Cat". Before using this data we'll assign a class label: one class will be 0 and the other will be 1. Assigning numeric labels puts the data in a form a neural network can use.



BINARY CLASSIFICATION

Data Splitting

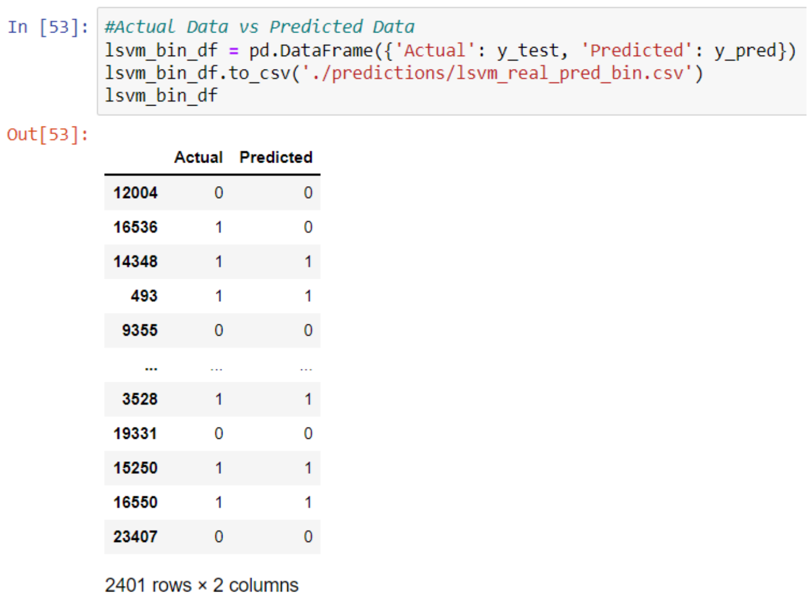
```
In [431]: #training and testing data split
          X = bin_data.drop(columns=['label'],axis=1)
          Y = bin_data['label']

In [432]: X_train,X_test,y_train,y_test = train_test_split(X,Y,test_size=0.3, random_state=50)
```

Figure 17: Data splitting in Binary classification

7.1 Result on SVM classifier

The Support Vector Machines algorithm is a great algorithm to learn. It offers many unique benefits, including high degrees of accuracy in classification problems. The algorithm can also be applied to many different use cases, including facial detection, classification of websites or emails, and handwriting recognition.

However, a key benefit of the algorithm is that it is intuitive. Being able to understand the mechanics behind an algorithm is important. This is true even when the math is a bit out of scope.



```
In [53]: #Actual Data vs Predicted Data
         lsvm_bin_df = pd.DataFrame({'Actual': y_test, 'Predicted': y_pred})
         lsvm_bin_df.to_csv('./predictions/lsvm_real_pred_bin.csv')
         lsvm_bin_df

Out[53]:
```

	Actual	Predicted
12004	0	0
16536	1	0
14348	1	1
493	1	1
9355	0	0
...	...	...
3528	1	1
19331	0	0
15250	1	1
16550	1	1
23407	0	0

2401 rows × 2 columns

Figure 18: Tabular comparison of actual and predicted data of SVM

Plot between Actual and Predicted Data

```
In [57]: #Plot
         plt.figure(figsize=(20,8))
         plt.plot(y_pred[:200], label="prediction", linewidth=2.0,color='blue')
         plt.plot(y_test[:200].values, label="real_values", linewidth=2.0,color='lightcoral')
         plt.legend(loc="best")
         plt.title("Linear SVM Binary Classification")
         plt.savefig('plots/lsvm_real_pred_bin.png')
         plt.show()
```

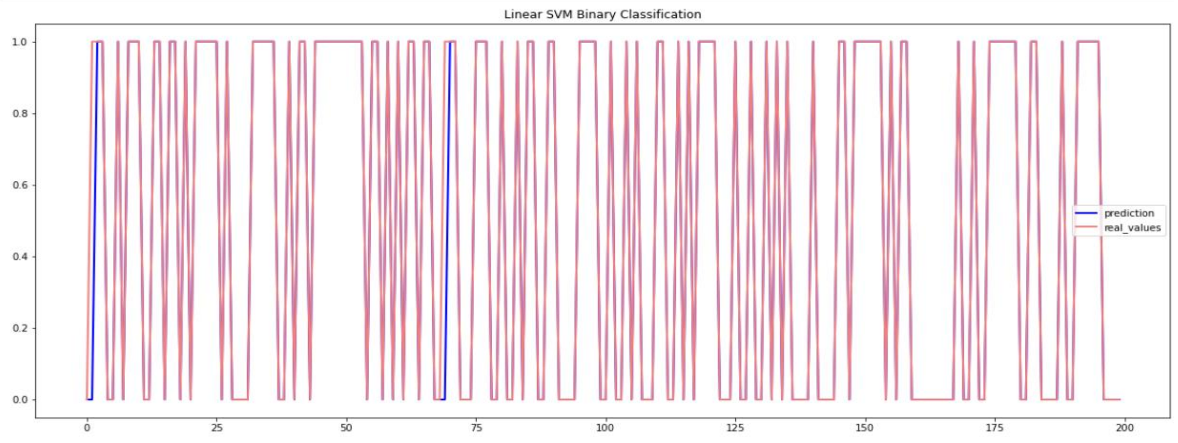


Figure 19: Graphical representation of actual and predicted data on SVM

Actual and Predicted Data ¶

```
In [56]: #Confusion matrix for svm
         predictions = lsvm_bin.predict(X_test)
         cm = confusion_matrix(y_test, predictions, labels=lsvm_bin.classes_)
         disp = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=lsvm_bin.classes_)
         disp.plot()
         plt.show()
```

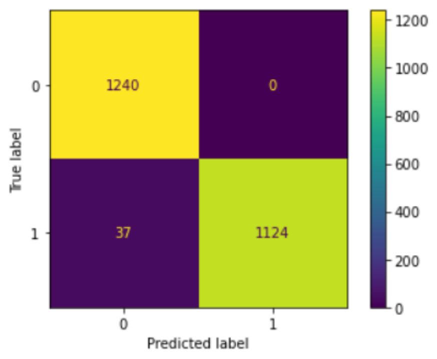


Figure 20: Confusion Matrix for SVM

```
In [54]: #Accuracy Score of SVM
         print("Mean Absolute Error - " , metrics.mean_absolute_error(y_test, y_pred))
         print("Mean Squared Error - " , metrics.mean_squared_error(y_test, y_pred))
         print("Root Mean Squared Error - " , np.sqrt(metrics.mean_squared_error(y_test, y_pred)))
         print("R2 Score - " , metrics.explained_variance_score(y_test, y_pred)*100)
         print("Accuracy - ",accuracy_score(y_test,y_pred)*100)

         Mean Absolute Error -  0.015410245730945439
         Mean Squared Error -  0.015410245730945439
         Root Mean Squared Error -  0.12413801082241264
         R2 Score -  93.92431441193632
         Accuracy -  98.45897542690545
```

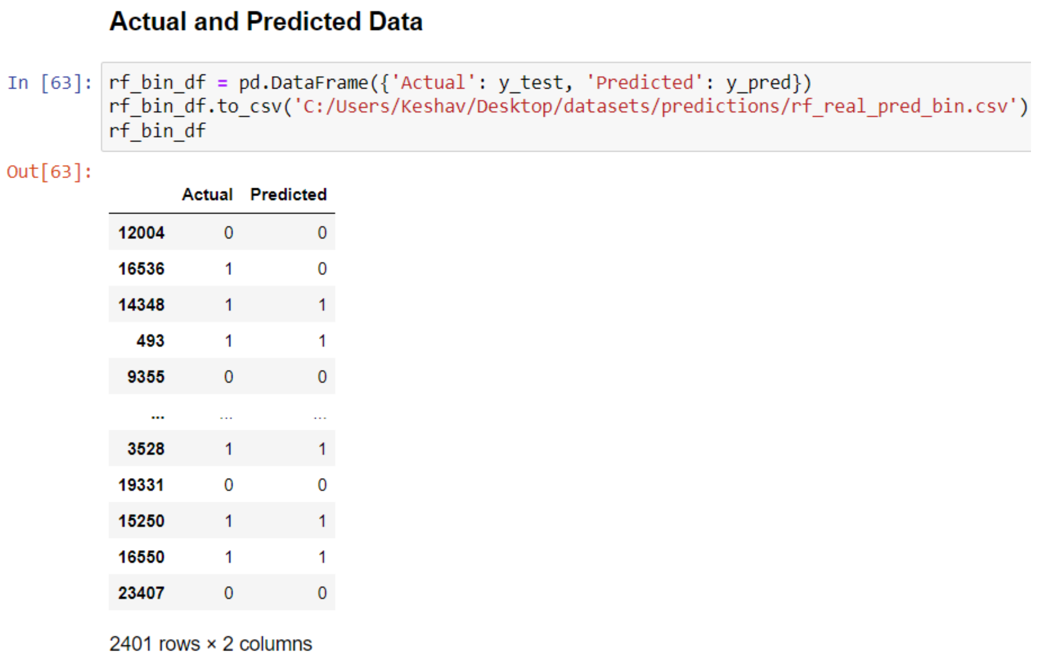
Figure 21: Accuracy of SVM Classifier

7.2 Results on Random Forest Classifier

Random forest is a Supervised Machine Learning Algorithm that is used widely in Classification and Regression problems. It builds decision trees on different samples and takes their majority vote for classification and average in case of regression.

One of the most important features of the Random Forest Algorithm is that it can handle the data set containing continuous variables as in the case of regression and categorical variables as in the case of classification. It performs better results for classification problems.

Random forest is a Supervised Machine Learning Algorithm that is used widely in Classification and Regression problems. It builds decision trees on different samples and takes their majority vote for classification and average in case of regression.



Actual and Predicted Data

```
In [63]: rf_bin_df = pd.DataFrame({'Actual': y_test, 'Predicted': y_pred})
         rf_bin_df.to_csv('C:/Users/Keshav/Desktop/datasets/predictions/rf_real_pred_bin.csv')
         rf_bin_df
```

Out[63]:

	Actual	Predicted
12004	0	0
16536	1	0
14348	1	1
493	1	1
9355	0	0
...	...	...
3528	1	1
19331	0	0
15250	1	1
16550	1	1
23407	0	0

2401 rows × 2 columns

Figure 22: Tabular comparison between Actual and Predicted data of Random Forest Classifier

Plot between Actual and Predicted Data

```
In [65]: plt.figure(figsize=(20,8))
         plt.plot(y_pred[200:400], label="prediction", linewidth=2.0,color='blue')
         plt.plot(y_test[200:400].values, label="real_values", linewidth=2.0,color='lightcoral')
         plt.legend(loc="best")
         plt.title("Random Forest Binary Classification")
         plt.savefig('C:/Users/Keshav/Desktop/datasets/plots/rf_real_pred_bin.png')
         plt.show()
```

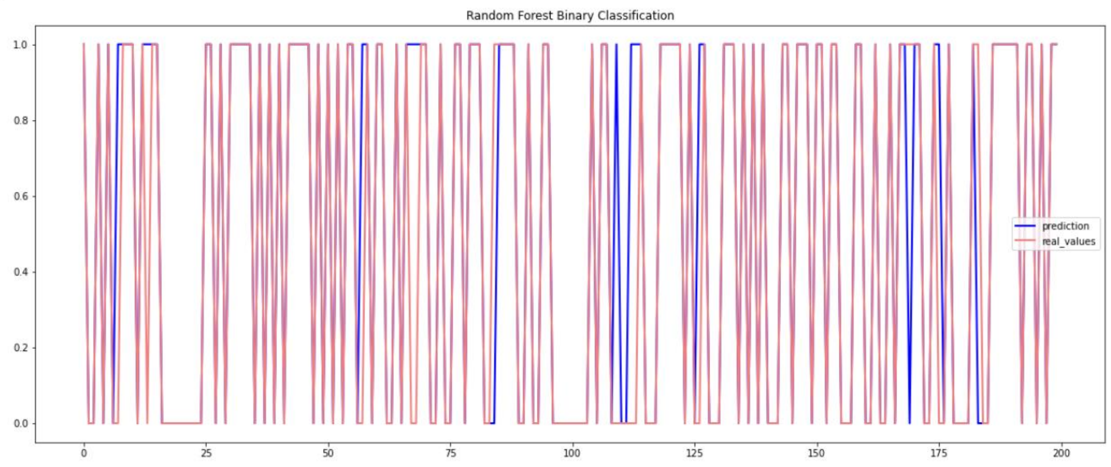


Figure 23: Graphical comparison between Actual and Predicted data of Random Forest Classifier

```
In [64]: # Confusion matrix for rf
         predictions = rf_bin.predict(X_test)
         cm = confusion_matrix(y_test, predictions, labels=rf_bin.classes_)
         disp = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=rf_bin.classes_)
         disp.plot()
         plt.show()
```

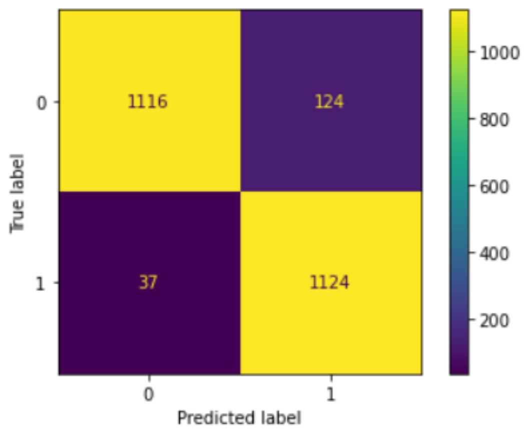
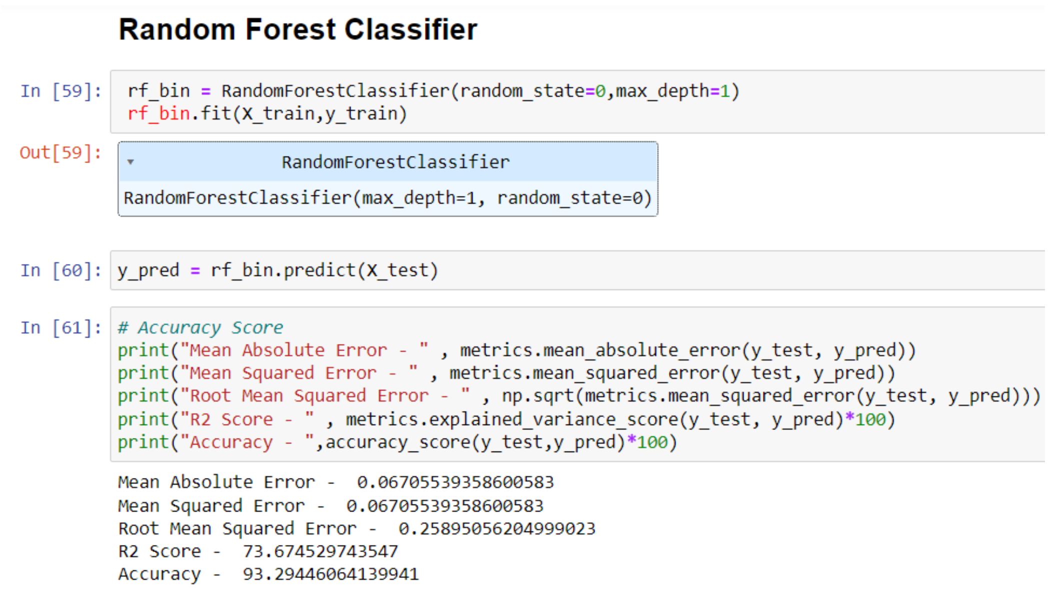


Figure 24: Confusion matrix of Random Forest Classifier



Random Forest Classifier

```
In [59]: rf_bin = RandomForestClassifier(random_state=0,max_depth=1)
         rf_bin.fit(X_train,y_train)

Out[59]: ▾          RandomForestClassifier
         RandomForestClassifier(max_depth=1, random_state=0)

In [60]: y_pred = rf_bin.predict(X_test)

In [61]: # Accuracy Score
         print("Mean Absolute Error - " , metrics.mean_absolute_error(y_test, y_pred))
         print("Mean Squared Error - " , metrics.mean_squared_error(y_test, y_pred))
         print("Root Mean Squared Error - " , np.sqrt(metrics.mean_squared_error(y_test, y_pred)))
         print("R2 Score - " , metrics.explained_variance_score(y_test, y_pred)*100)
         print("Accuracy - ",accuracy_score(y_test,y_pred)*100)

         Mean Absolute Error -  0.06705539358600583
         Mean Squared Error -  0.06705539358600583
         Root Mean Squared Error -  0.25895056204999023
         R2 Score -  73.674529743547
         Accuracy -  93.29446064139941
```

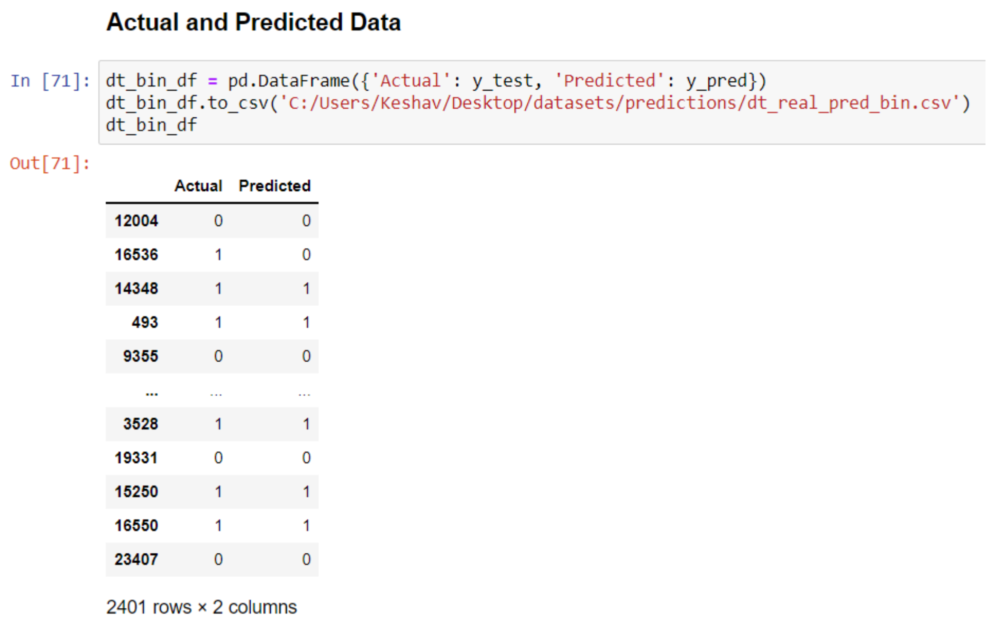
Figure 25: Accuracy of Random Forest Classifier

7.3 Results on Decision Tree Classifier

Decision Tree algorithm belongs to the family of supervised learning algorithms. Unlike other supervised learning algorithms, the decision tree algorithm can be used for solving regression and classification problems too.

The goal of using a Decision Tree is to create a training model that can use to predict the class or value of the target variable by learning simple decision rules inferred from prior data(training data).

In Decision Trees, for predicting a class label for a record we start from the root of the tree. We compare the values of the root attribute with the record's attribute. On the basis of comparison, we follow the branch corresponding to that value and jump to the next node.



Actual and Predicted Data

```
In [71]: dt_bin_df = pd.DataFrame({'Actual': y_test, 'Predicted': y_pred})
         dt_bin_df.to_csv('C:/Users/Keshav/Desktop/datasets/predictions/dt_real_pred_bin.csv')
         dt_bin_df
Out[71]:
```

	Actual	Predicted
12004	0	0
16536	1	0
14348	1	1
493	1	1
9355	0	0
...	...	...
3528	1	1
19331	0	0
15250	1	1
16550	1	1
23407	0	0

2401 rows × 2 columns

Figure 26: Tabular comparison between Actual data and Predicted data of Decision tree classifier

Plot between Actual and Predicted Data

```
In [73]: plt.figure(figsize=(20,8))
         plt.plot(y_pred[300:500], label="prediction", linewidth=2.0,color='blue')
         plt.plot(y_test[300:500].values, label="real_values", linewidth=2.0,color='lightcoral')
         plt.legend(loc="best")
         plt.title("Decision Tree Binary Classification")
         plt.savefig('C:/Users/Keshav/Desktop/datasets/plots/dt_real_pred_bin.png')
         plt.show()
```

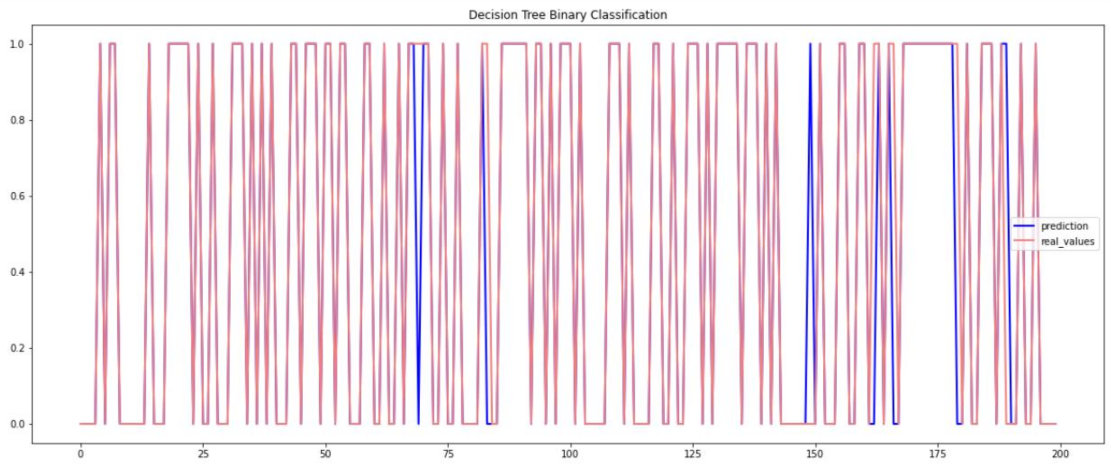


Figure 27: Graphical comparison between Actual data and Predicted data of Decision Tree classifier

```
In [72]: #Confusion matrix for dt
         predictions = dt_bin.predict(X_test)
         cm = confusion_matrix(y_test, predictions, labels=dt_bin.classes_)
         disp = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=dt_bin.classes_)
         disp.plot()
         plt.show()
```

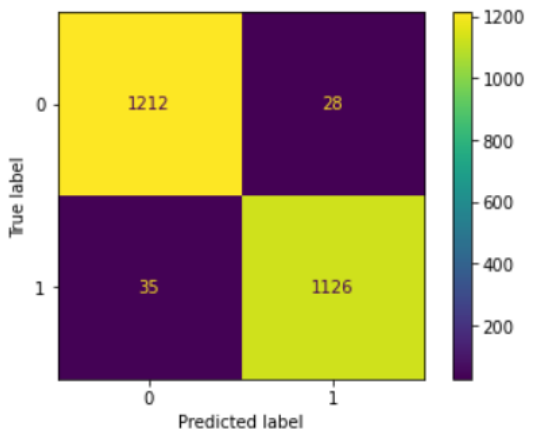
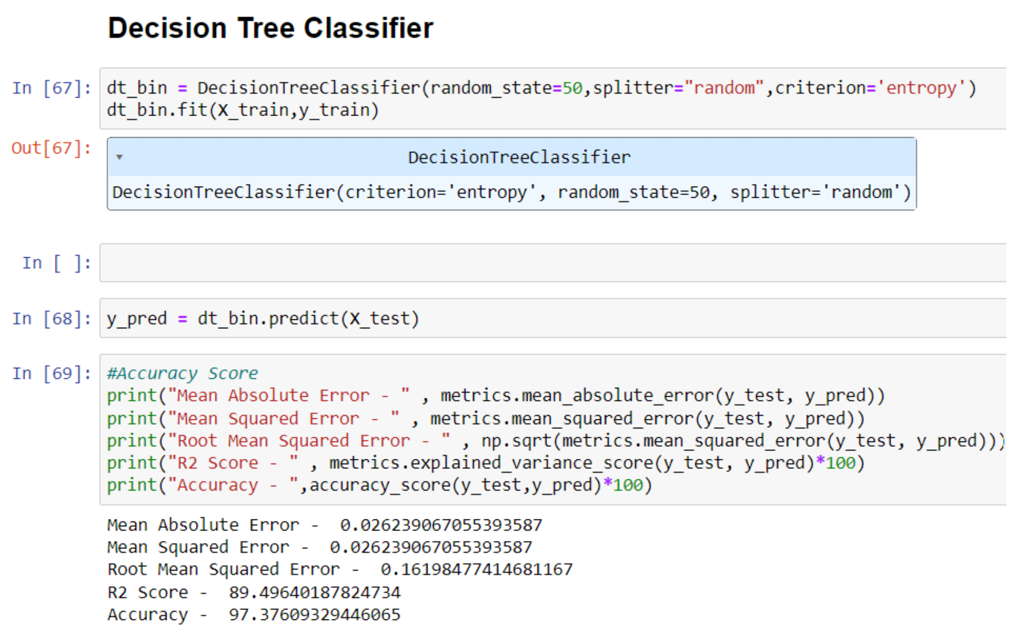


Figure 28: Confusion matrix of Decision Tree classifier



Decision Tree Classifier

```
In [67]: dt_bin = DecisionTreeClassifier(random_state=50,splitter="random",criterion='entropy')
         dt_bin.fit(X_train,y_train)

Out[67]:                          DecisionTreeClassifier
         DecisionTreeClassifier(criterion='entropy', random_state=50, splitter='random')

In [ ]:

In [68]: y_pred = dt_bin.predict(X_test)

In [69]: #Accuracy Score
         print("Mean Absolute Error - " , metrics.mean_absolute_error(y_test, y_pred))
         print("Mean Squared Error - " , metrics.mean_squared_error(y_test, y_pred))
         print("Root Mean Squared Error - " , np.sqrt(metrics.mean_squared_error(y_test, y_pred)))
         print("R2 Score - " , metrics.explained_variance_score(y_test, y_pred)*100)
         print("Accuracy - ",accuracy_score(y_test,y_pred)*100)

         Mean Absolute Error -  0.026239067055393587
         Mean Squared Error -  0.026239067055393587
         Root Mean Squared Error -  0.16198477414681167
         R2 Score -  89.49640187824734
         Accuracy -  97.37609329446065
```

Figure 29: Accuracy of Decision Tree Classifier

After comparing the accuracy of all the three classifier algorithms, namely-

(1) Support Vector Machine

(2) Random Forest

(3) Decision Tree

we observe that the highest accuracy is shown by the SVM classifier which proves it to be the best performing classifier among all three.

CHAPTER 8: FUTURE SCOPE

IDS by ML is one of the most developing new technology under Machine Learning concept. Infact everyday in every technical industry some new pattern of malware we can found easily, for that we need to improve our developing system, make the future of the technology more better.

so here we have listed the following future ideas and scope of ids:

1. Algorithms can be improved by using more complex semantics of security ontology.

2. S-LAID solution can be tested in different application domains of sensor network.

3. Location verification protocol can be extended.

4. Anomaly detection technique can be extended for improvement.

5. Election procedure can be implemented; IDS scalability and definition of detection policy need to be determined, more specifically

6. Feature selection in anomaly detection can be done by data mining; Rule based approach can be extended to provide anomaly detection model with better performance and flexibility.

7. Probable direction of intrusion detection and security would be merging traditional IDS with prevention mechanisms to not provide an in-depth analysis of what went wrong, but instead protect your systems by preventing the attack from occurring in the first place and then providing a detailed analysis of what was prevented from happening. A considerable faction of the security community adheres to this way of thinking.

CHAPTER 9: CONCLUSION

With this we have come to an end of our final year project. We have faced many difficulties while making this project. We like to thank our Mentor Prof Sanjukta Bhattacharya who guided us till the very end of this project. We also like to express our gratitude to out Head of Department, Prof Shiladitya Munshi who motivated us to move forward when the going got tough. We would also like to thank our wonderful IT faculty members who helped us in time of difficulties.

In this project we have done training and testing of three different classifiers, and the same dataset was used in all the classifiers. Their accuracy, confusion matrix and graphical plot was generated, and after observing the results of all the three classifiers, we came to the conclusion that the Support Vector Machine Classifier gave the best result in our scenario.

Finally we would like to express our gratitude to our friends and the almighty who helped us in situations where we were stuck and could have not moved forward without the much needed help provided by them. We hope that our project would do good to the society and help it to be a better place, free from intrusions.

CHAPTER 10: BIBLIOGRAPHY

10.1 CONFERENCE PAPERS

- Title- “UNSW-NB15 Dataset Feature Selection and Network Intrusion Detection using DeepLearning “
  Author- V. Kanimozhi, Prem Jacob

- Title – “Unsw-Nb15 Dataset and Machine Learning Based Intrusion Detection Systems”

  Author- Avinash R. Sonule, Mukesh Kalla, Amit Jain, D. S. Chouhan

- Title- “Classification Techniques for Intrusion Detection – An Overview
  Author- P.Amudha , S.Karthik, S.Sivakumari

10.2 Websites

- https://www.threatstack.com/
- https://techvidvan.com/tutorials/
- www.tutorialspoint.com
- https://www.xoriant.com/
- https://monkeylearn.com/
- https://www.javatpoint.com/
- https://towardsdatascience.com/
- https://en.wikipedia.org/wiki/Intrusion_detection_system