A PROJECT REPORT ON

INTRUSION DETECTION SYSTEM USING MACHINE LEARNING

Mini project submitted in partial fulfilment of the requirements for the award of the degree of

BACHELOR OF TECHNOLOGY

IN

INFORMATION TECHNOLOGY

(2019-2023)

BY

Thunuguntla Srichakranath	**19241A1255**
Parakala Venkata Anirudh	**19241A1238**
Podduturi Hruthvik Reddy	**19241A1240**
Shaik Sohel	**19241A1251**

Under the Esteemed guidance of

Dr. N. Rajasekhar

Professor, Dept of IT

DEPARTMENT OF INFORMATION TECHNOLOGY

GOKARAJU RANGARAJU INSTITUTE OF ENGINEERING AND TECHNOLOGY

(AUTONOMOUS)

HYDERABAD

CERTIFICATE

This is to certify that it is a bonafide record of Mini Project work entitled "**Intrusion Detection System Using Machine Learning**" done by **T. Srichakranath (19241A1255), P. Venkata Anirudh (19241A1238), P. Hruthvik Reddy(19241A1240), Shaik Sohel (19241A1251)**, students of **B.Tech(IT)** in the department of Information Technology, Gokaraju Rangaraju Institute of Engineering and Technology during the period 2019-2023 in the partial fulfilment of the requirements for the award of degree of **BACHELOR OF TECHNOLOGY IN INFORMATION TECHNOLOGY** from GRIET, Hyderabad.

Dr. N. Rajasekhar
Professor, IT Department
Internal Project Guide

Dr. N. V. Ganapati Raju
Professor
Head of IT Department

Project external

ACKNOWLEDGEMENT

We take immense pleasure in expressing gratitude to our internal guide **Dr. N. Rajasekhar, Professor,** Information Technology, GRIET. We express our sincere thanks for his encouragement, suggestions and support, which provided the impetus and paved the way for the successful completion of the project work.

We wish to express our gratitude to **Dr. N. V. Ganapati Raju,** Head of IT Department and our project co-ordinators **Mrs. K. Archana,** Assistant Professor**, Mrs. A. Srilakshmi,** Assistant Professor and **Dr. K. Prasanna Lakshmi,** Professor for their constant support during the project.

We express our sincere thanks to **Dr. Jandhyala N Murthy,** Director, GRIET, and **Dr. J. Praveen,** Principal, GRIET, for providing us the conductive environment for carrying through our academic schedules and project with ease.

Email: tsrichakranath@gmail.com
Contact No: 9160763234
Address: Nizampet, Hyderabad

Email: anirudhparakala@gmail.com
Contact No: 7993082271
Address: KPHB, Hyderabad

Email: hruthvikreddy@gmail.com
Contact No: 9100069060
Address: Mathrusri Nagar, Hyderabad

Email: shaiksohel5588@gmail.com
Contact No: 8919219508
Address: Nizamabad

DECLARATION

This is to certify that the project entitled **"Intrusion Detection System Using Machine Learning"** is a bonafide work done by us in the partial fulfilment of the requirements for the award of the degree **BACHELOR OF TECHNOLOGY IN INFORMATION TECHNOLOGY** from Gokaraju Rangaraju Institute of Engineering and Technology, Hyderabad.

We also declare that this project is a result of our own effort and has not been copied or imitated from any source. Citations from any websites, books and paper publications are mentioned in Bibliography.

This work was not submitted earlier at any other University or Institute for the award of any degree.

Thunuguntla Srichakranath	**19241A1255**
Parakala Venkata Anirudh	**19241A1238**
Podduturi Hruthvik Reddy	**19241A1240**
Shaik Sohel	**19241A1251**

TABLE OF CONTENTS

ABSTRACT

More personal information and important data are in the computers now more than ever. So, it is important that there is a strong security to keep data from being stolen and misused. An **Intrusion detection system (IDS)** is used to detect if there are any intrusions that happen in a system and notify the administrator if there are any. In IDS reports malicious activities or policy violations to the admin through a system called security information and event management system. Multiple support systems are used to generate the log information and an SIEM combines this information and then is used to raise alarms. These alarms can sometimes become false alarms. Support all major communication formats plus internet & StarLink Wireless Radios, universal primary/backup.

Network Intrusion detection systems are fixed at strategic locations in the network which have high possibilities of detecting an attack. Once an attack is identified the admin of the system is notified by raising an alert. An analysis is done on the packet information by checking their signatures with the known attacks in the library. The primary task is to create an application which can effectively detect intrusions and differentiate actual intrusions from false alarms, notify the admin.

1. INTRODUCTION

1.1 Introduction to Project

Cyber security also called as information security refers to safe guarding systems linked to the internet from the cyber threats. Most of the data in today's world is digitalized and are stored in digital devices connected to the internet. Most of the data that will be under constant threat will be financial information and passwords. So, one must be with inherent security plan to protect the confidentiality of information. IDS is a solution which analyses the packet information of the incoming traffic and notifies if it is a valid packet or not. An alarm kind of signal is raised on receiving any malicious packet. Generally, these are off two types: Signature based and Anomaly based. The first kind of system is only limited to a few types of intrusions whereas the second type detects anomalous behavior depending on deviations from normal behavior.

An IDS is helpful in analyzing the total attacks and their categories. Using the information and the statistics given by IDS organizations can change their security implementations. There is a problem with IDSes. False negatives, which means a threat is mistaken as a legitimate traffic and is freely flows through the network and the system. Due to this problem no one will be aware of the any intrusion that has happened which can sometimes cause serious threats to the organization like information theft, data tampering, ransomware attacks etc. It is good to have an IDS to generate false positive alarms which means mistaking a legitimate traffic to be malicious and keep the IT people alert rather than having an IDS to be generating more false negatives.

A few ML models have been used in the project and the model acquiring the highest accuracy is used for prediction. Two separate datasets were used in this project. One for training and testing the model. The other dataset is used for prediction. This project made use of several models like Naive Bayes Algorithm, Decision Tree Classifier, logistic regression, Linear regression, Support vector machine (SVC) for the supervised learning. A signature-based intrusion detection system using decision tree model is used

to classify the incoming packet information into 10 different categories. Our signature-based IDS has a detection rate higher than 89% and developed a system to recognize 9 different types of cyberattacks and malware injections.

.

1.2 Existing Systems

The current existing signature-based intrusion detection systems are being loaded with some fixed number of attack types and are used to detect those. Developing such a system requires a lot of coding effort. And adding any type of new attack category becomes a very hectic task. Since a network packet contains a lot of information, many mathematical calculations have to be done in order to process the code for a new attack category. In the current world the replications of new viruses, worms with very little configuration changes are becoming common in the cyber world. So, it is so hard to deliberate calculations for many intrusions.

1.3 Proposed system

We developed a signature-based intrusion detection with a very much reduced coding effort. Introducing a new anomaly or attack type into the system is very easy. Just training the system with the new attack category one time is enough. If any intrusions of that type happen the system detects it a good accuracy of around 90%. The system can perfectly determine which packet is normal. So whenever there happens any intrusion, the system may not detect the type of intrusion perfectly but it can perfectly differentiate the packet as an anomalous packet and raises an alert. True positivity rate of predicting a normal packet is 100%.

2. REQUIREMENT ENGINEERING

2.1 Recommended Hardware Requirements

- Processor –Intel Octa Core- i7- CPU@2.80Hz (64-bit OS).
- Memory – 8GB RAM

2.2 Software Requirements

- Windows 10
- Anaconda navigator
- Python
- Spyder
- Jupyter notebook

3. LITERATURE SURVEY

Constant developments are going on in the field of IDS.

[1] This paper gave us an insight on how the ids system is developed differently depending on the type of system and network, for example on mobile networks The ids system is based on the data gathered on web browsing usage and SMS. This paper also gave us an understanding of how the different machine learning models perform vastly different based on the given network systems.

[2] This paper gave us a clear understanding of the working of an IDS, what classifies an IDS and how is IDS different from other network security systems. We learned the different types of ids and their features and flaws. This paper also provided knowledge on how neural network perform on models trying to build an ids

[3] This paper gave us an insight on how Big Data affects the complexity accuracy and speed of the classification model using machine learning, As Big data consists of large amounts of data coming in at very rapid speed the classification becomes complex. one of proposed method in the paper to overcome this problem is to use SVM model.

4. TECHNOLOGY

4.1 Introduction to Python

Python is an easy language to learn. Released in the early 90's by Guido van Rossum, reduced the number of lines to code using indentation and the readability has become easy. Numerous modules and libraries have been included in python to enable easy programming to all industrial scales. An improvement in managing memory reduced a lot of complexity.

Python can be incorporated with most of the currently available technologies and languages like HTML with python, IOT using python etc. Each of which use one or more modules and libraries for implementation. Being very flexible to use many new technologies are being developed using python.

4.2 YOU CAN USE PYTHON FOR PRETTY MUCH ANYTHING

One significant advantage of learning Python is that it's a general-purpose language that can be applied in a large variety of projects. Below are just some of the most common fields where Python has found its use:

- Data science
- Web development
- Computer graphics
- Basic game development
- Mapping and geography (GIS software)
- Scientific and mathematical computing

4.3 PYTHON IS WIDELY USED IN DATA SCIENCE

Python makes it easy to do tasks related to ML, NLP and many complex data manipulations etc. These advantages of python over other languages are making it a great tool to do complicated and scientific calculations.

Some of the python libraries are mentioned below:

4.3.1 PANDAS

Pandas is a free library available in python. This is one of the libraries that generally comes with the download of python. This is mainly used for data manipulation and is mainly used with another library called NumPy. This library is generally imported with an alias name as “pd”.

4.3.2 NUMPY

NumPy is abbreviated as Numerical Python is an open-source python library used for processing multi-dimensional data. The general alias name for NumPy is “np”. This is a foundational python package that adds support for huge, higher dimensional arrays and matrices, with an added massive library of high-level mathematical operations.

4.3.3 MATPLOTLIB

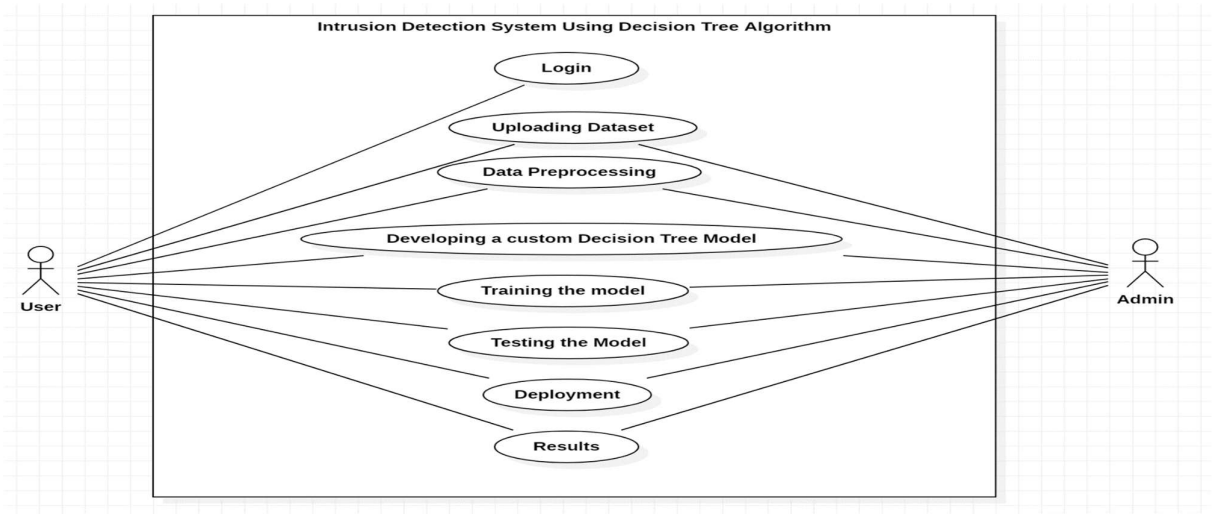
Matplotlib enables easy data visualization. One of the main advantages of using matplotlib is that it comes with inbuilt functions that help in visualizing and analysing the data in a much more convenient manner. Plotting graphs, maps, implementing pie charts can be easily done by Matplotlib

4.3.4 SCIKIT-LEARN

Scikit learn is one of the key libraries available in Python that is mainly used in the development of ML models. It provides a suite of efficient tools for machine learning and statistical modelling within a python interface. Scikit learn module provides many functions that can be used for data pre-processing, splitting the datasets, data decomposition, for viewing the confusion matrices, model accuracies, classification reports etc.

5. DESIGN REQUIREMENT ENGINEERING

5.1 Use Case Diagram



Actors: user, admin

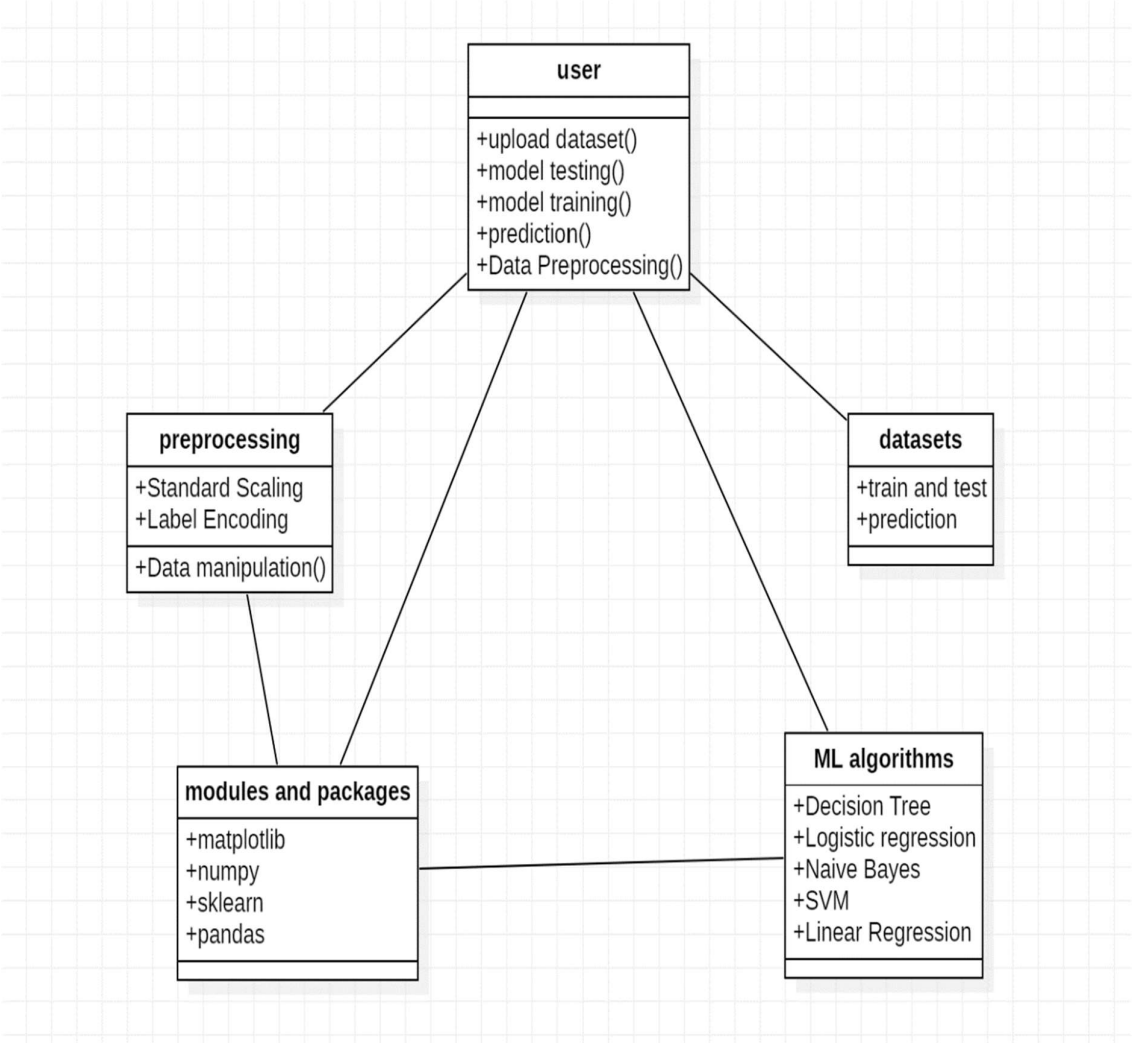
Use cases:

1) Login
2) Uploading Dataset
3) Data -preprocessing
4) Developing a custom Decision Tree Model
5) Training the Model
6) Testing the Model
7) Deployment
8) Results

Relationships:

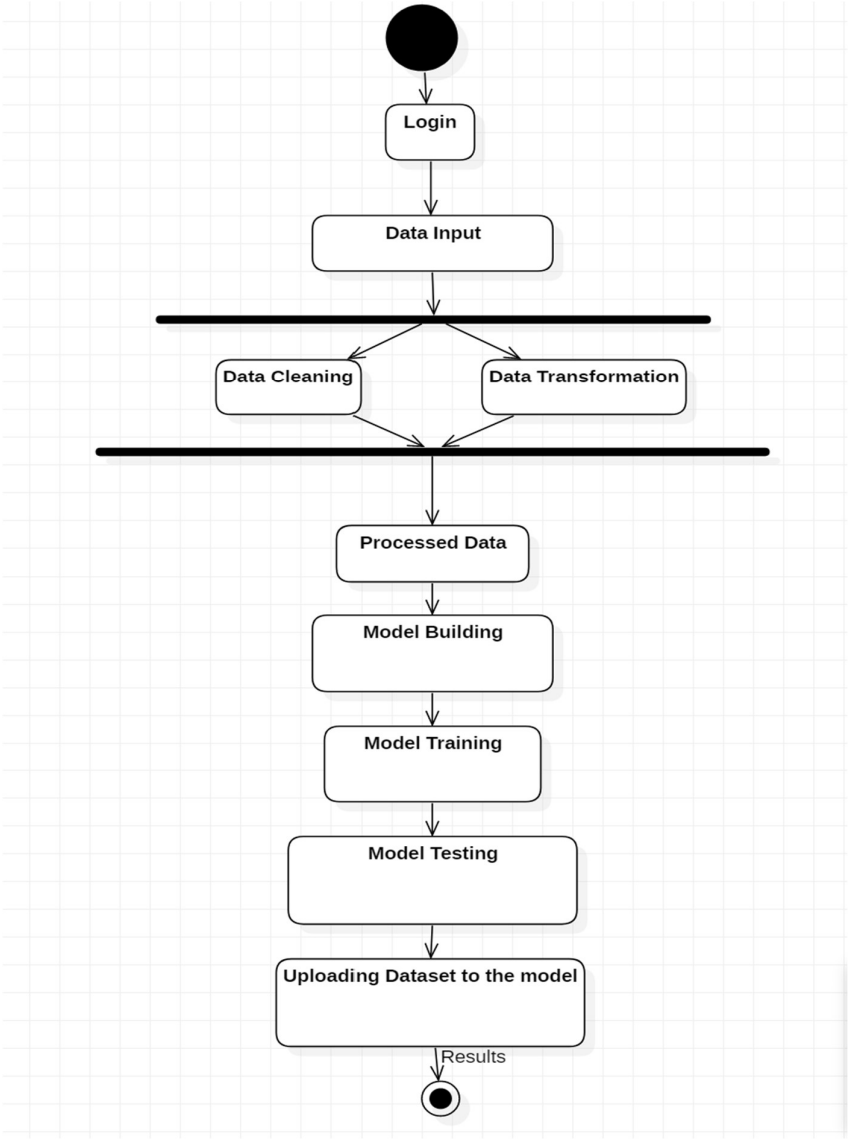
1) The first use case i.e., login is connected only to the user.
2) The rest of all the use cases are connected to both the actors who are user and admin

5.2 Class Diagram



Classes	Attributes	Operations
User		Upload dataset, model training, model testing,prediction, data preprocessing
Preprocessing	Standard Scaling,Label Encoding	Data manipulation
modules and packages	matplotlib,numpy,sklearn,pandas	
datasets	train and test,prediction	
ML algorithms	Decision Tree,Logistic Regression,Naïve Bayes, SVM, Linear Regression	

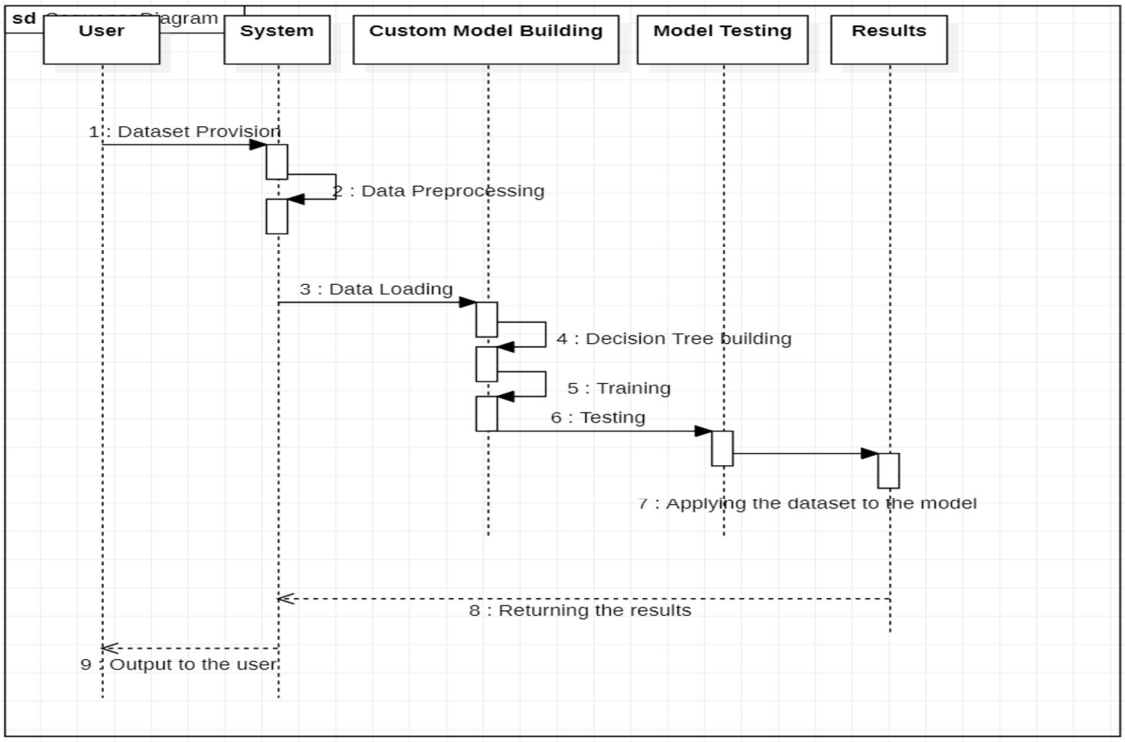
5.3 Activity Diagram



Scenario:

The user first logs into the system. Then data input is given and then the preprocessing techniques takes place that are Data cleaning and data transformation. The processed data is applied to the created models and then trained. The trained models then undergo testing procedure. Finally, the model with highest test accuracy is used for making the prediction.

5.4 Sequence Diagram



Scenario:

1)Initially the user gives the system the dataset.

2)The system then processes the data to required formats and then is loaded for building custom ML models

3) Once the models are built those models are trained, tested and then their accuracies are checked to get the highest efficiency

4) Once the checking is done a new dataset is loaded for prediction.

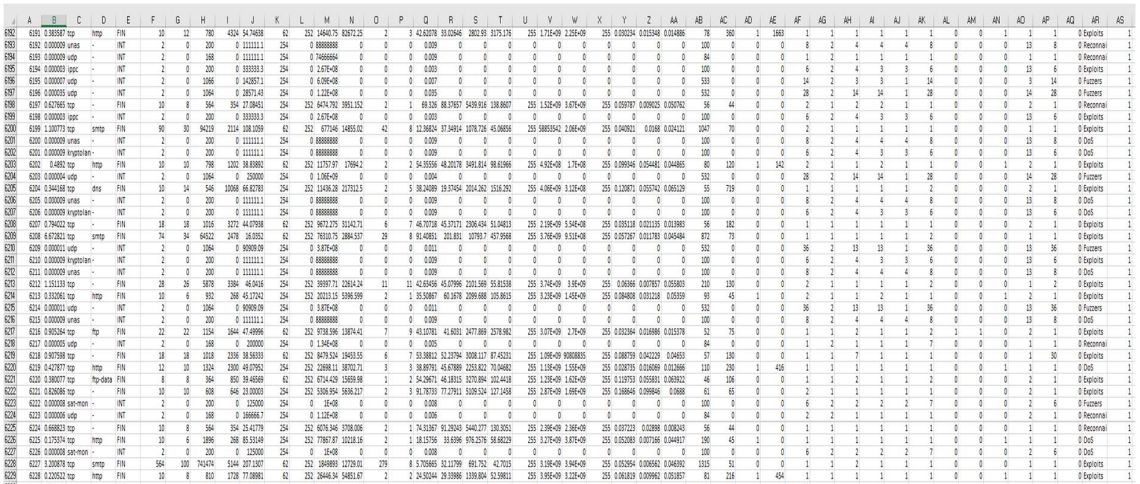
5)The predicted results are then returned to the user.

6.Implementation

6.1 Datasets

The datasets used in the project are UNSW-NB 15 for all training, testing and prediction. The UNSW-NB 15 dataset has packet information related to 9 different types of attacks.

Around 83000 records are divided for training and testing purpose, 173000 records are used for making the prediction. The dataset has a total of 45 attributes including the attack category.



6.2 Stages in the project

1) Importing modules
2) Uploading datasets
3) Pre-processing the data
4) Splitting for training and testing data
5) Creating Machine Learning models for comparative study
6) Training the models with training data
7) Testing the models
8) Make prediction using a separate dataset

6.2.1 Importing modules

1) Modules that are imported in this project:
    a. NumPy
    b. Pandas
    c. Matplotlib
    d. Scikit-learn
    e. Warnings

```
# Importing Libraries
import matplotlib
import matplotlib.pyplot as plt
import pandas as pd  #Data manipulation and analysis
import numpy as np   #Performs high level manipulation
import sklearn   # provides efficient tools for predictive data analysis

# Preprocessing purpose
from sklearn.preprocessing import StandardScaler
from sklearn.preprocessing import LabelEncoder

# For getting the importances
from sklearn.ensemble import RandomForestClassifier

# Feature Extraction
from sklearn.decomposition import PCA

# Splitting Data
from sklearn.model_selection import train_test_split

# For accuracy,Classification Report, Confusion Matrix
from sklearn import metrics

# For training different ML models
from sklearn import tree
from sklearn.linear_model import LogisticRegression
from sklearn.naive_bayes import BernoulliNB
from sklearn.linear_model import LinearRegression
from sklearn.svm import SVC

# Ignore warnings
import warnings
warnings.filterwarnings('ignore')
```

6.2.2 Uploading datasets

To upload this file first it is downloaded into the system in a .tar format. Then it is unzipped and extracted. The extracted csv files are then uploaded into the environment using the read_csv function in pandas library.

```
# Uploading datasets for training,testing and prediction
train = pd.read_csv('C:/DataSets/partOfTraining/a part of training and testing set/UNSW_NB15_training-set.csv')
pred = pd.read_csv('C:/DataSets/partOfTraining/a part of training and testing set/UNSW_NB15_testing-set.csv')
```

6.2.2.1 Training data description

```
train.describe()
```

	id	dur	spkts	dpkts	sbytes	dbytes	rate	sttl	dttl	sload	...	ct_sr
count	82332.000000	82332.000000	82332.000000	82332.000000	8.233200e+04	8.233200e+04	8.233200e+04	82332.000000	82332.000000	8.233200e+04	...	8
mean	41166.500000	1.006756	18.666472	17.545936	7.993908e+03	1.323379e+04	8.241089e+04	180.967667	95.713003	6.454902e+07	...	
std	23767.345519	4.710444	133.916353	115.574086	1.716423e+05	1.514715e+05	1.486204e+05	101.513358	116.667722	1.798618e+08	...	
min	1.000000	0.000000	1.000000	0.000000	2.400000e+01	0.000000e+00	0.000000e+00	0.000000	0.000000	0.000000e+00	...	
25%	20583.750000	0.000008	2.000000	0.000000	1.140000e+02	0.000000e+00	2.860611e+01	62.000000	0.000000	1.120247e+04	...	
50%	41166.500000	0.014138	6.000000	2.000000	5.340000e+02	1.780000e+02	2.650177e+03	254.000000	29.000000	5.770032e+05	...	
75%	61749.250000	0.719360	12.000000	10.000000	1.280000e+03	9.560000e+02	1.111111e+05	254.000000	252.000000	6.514286e+07	...	
max	82332.000000	59.999989	10646.000000	11018.000000	1.435577e+07	1.465753e+07	1.000000e+06	255.000000	253.000000	5.268000e+09	...	

8 rows × 41 columns

6.2.2.2 Prediction data description

```
pred.describe()
```

	id	dur	spkts	dpkts	sbytes	dbytes	rate	sttl	dttl	sload	...
count	175341.000000	175341.000000	175341.000000	175341.000000	1.753410e+05	1.753410e+05	1.753410e+05	175341.000000	175341.000000	1.753410e+05	...
mean	87671.000000	1.359389	20.298664	18.969591	8.844844e+03	1.492892e+04	9.540619e+04	179.546997	79.609567	7.345403e+07	...
std	50616.731112	6.480249	136.887597	110.258271	1.747656e+05	1.436542e+05	1.654010e+05	102.940011	110.506863	1.883574e+08	...
min	1.000000	0.000000	1.000000	0.000000	2.800000e+01	0.000000e+00	0.000000e+00	0.000000	0.000000	0.000000e+00	...
25%	43836.000000	0.000008	2.000000	0.000000	1.140000e+02	0.000000e+00	3.278614e+01	62.000000	0.000000	1.305334e+04	...
50%	87671.000000	0.001582	2.000000	2.000000	4.300000e+02	1.640000e+02	3.225807e+03	254.000000	29.000000	8.796748e+05	...
75%	131506.000000	0.668069	12.000000	10.000000	1.418000e+03	1.102000e+03	1.250000e+05	254.000000	252.000000	8.888889e+07	...
max	175341.000000	59.999989	9616.000000	10974.000000	1.296523e+07	1.465555e+07	1.000000e+06	255.000000	254.000000	5.988000e+09	...

8 rows × 41 columns

6.2.3 Pre-processing the data

The pre-processing techniques applied on the datasets in this project are:

6.2.3.1 Standard Scaling:

Scikit-learn has a dedicated library for standard scaling operation. The standard scaling technique is applied to all columns in the dataset of type integer or float. A total of 41 columns are involved in standard scaling procedure. The output of the standard scaling procedure is that the standard deviation of the columns involved in the change is converted to 1. All the values in the columns are scaled such that the column has unit standard deviation.

```
SS = StandardScaler()

# extract numerical attributes and scale it to have unit standard deviation
cols = train.select_dtypes(include=['float64','int64']).columns
int_train = SS.fit_transform(train.select_dtypes(include=['float64','int64']))
int_pred = SS.fit_transform(pred.select_dtypes(include=['float64','int64']))

# turn the result back to a dataframe
int_traindf = pd.DataFrame(int_train, columns = cols)
int_preddf = pd.DataFrame(int_pred, columns = cols)
```

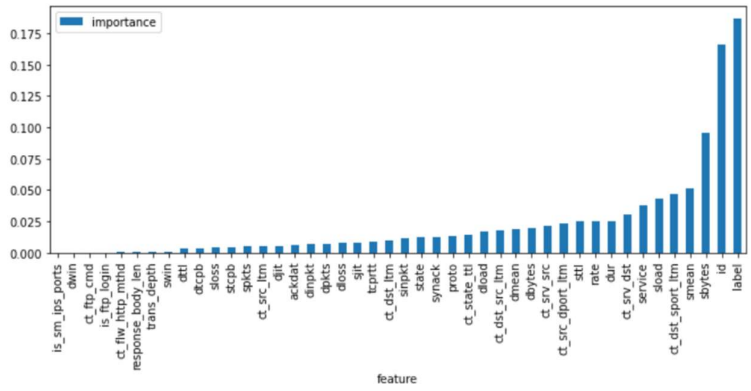
6.2.3.2 Label Encoding:

The label encoding technique is applied to the only 4 columns of the dataset that comes under the nominal category. The 4 columns are service, state, proto, and attack_cat. Each label in these 4 columns is given a number and the numbers are replaced in the columns instead of the labels. For example, the column attack_cat that has the type of attacks has 10 labels in it. So, after label encoding the attack_cat column consists of number from 1 to 10.

```
LE = LabelEncoder()

# extract categorical attributes from both training and test sets
obj_train = train.select_dtypes(include=['object']).copy()
obj_pred = pred.select_dtypes(include=['object']).copy()

# encode the categorical attributes
LE_obj_train = obj_train.apply(LE.fit_transform)
LE_obj_pred = obj_pred.apply(LE.fit_transform)

# separate target column from encoded data
enctrain = LE_obj_train.drop(['attack_cat'], axis=1)
encpred = LE_obj_pred.drop(['attack_cat'],axis=1)
target = pred['attack_cat']
lir_tar_train = LE_obj_train['attack_cat']
```

6.2.4 Plotting importances

```
#Checking which variable is useful in calculating the target variable

rfc = RandomForestClassifier();

# fit random forest classifier on the training set
rfc.fit(train_x, train_y);
# extract important features
score = np.round(rfc.feature_importances_,3)
importances = pd.DataFrame({'feature':train_x.columns,'importance':score})
importances = importances.sort_values('importance',ascending=True).set_index('feature')
# plot importances
plt.rcParams['figure.figsize'] = (11, 4)
importances.plot.bar();
```



6.2.5 Splitting for training and testing

The function used for division is train_test_split. Scikit-learn has dedicated functions for the task to be accomplished. The training dataset is used for splitting the data for training and testing. Only 20% of the data is trained. Off the 5 models used for comparative analysis linear regression models takes a different input for

training. The target variable should be in numeric format for linear regression. The remaining four models takes normal input for training.

```
# Splitting training data for training and testing the ML models
X_train,X_test,Y_train,Y_test = train_test_split(train_x,train_y,test_size=0.80, random_state=2)
mlrx_train,mlrx_test,mlry_train,mlry_test = train_test_split(train_x,lir_tar_train,test_size=0.80, random_state = 2)
```

6.2.6 Creating machine learning models for Comparative study

Five algorithms are used in this project:

1) Decision Tree

2) Logistic Regression

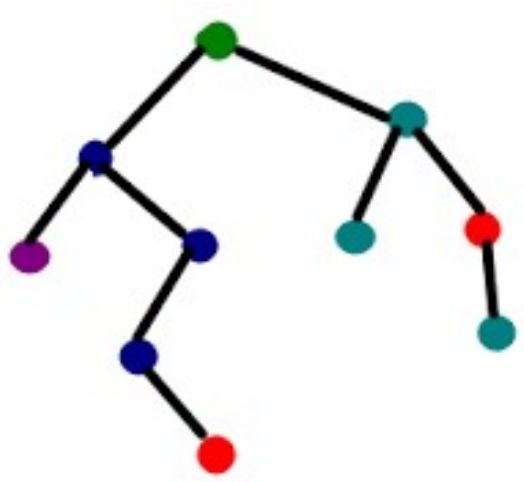
3) Linear Regression

4) Naive Bayes

5) Support Vector Machine

6.2.6.1 Decision tree:

A node in a decision tree is divided into multiple children nodes using various algorithms. Higher the dependency of the node to the target earlier the division takes place. The decision tree checks for all the possible divisions and divides the nodes in such a way that the homogeneity is highest. Attribute selection measures such as entropy, information gain, Gini coefficient, Gini ratio, and chi-square method are used to select the attributes in order to get maximum accuracy.



6.2.6.2 Logistic regression:

One of the popular algorithms in supervised machine learning is Logistic Regression. It is used to predict categorical variables using particular independent variables predicting the output of the dependent variables. So, the result must be a categorical value or a discrete value. It can be either 0 or 1. Logistic regression gives an "S" -shaped logistic function as output that predicts two maximum values. Logistic regression is an important machine learning algorithm because it provides probabilities and can classify new data using existing datasets. Logistic regression makes it easy to classify observations based on different data types and determine the most effective variables for classification.

6.2.6.3 Linear regression:

It is said to be one of the easiest approaches for classification in Machine Learning. It is used for classifying real-valued or numeric values. In order to produce a result a linear relationship is established between dependent and independent variables. We try to minimise the error between the predicted values and actual values in this mechanism i.e., the best fit will have the least value of error.

6.2.6.4 Naive bayes:

The naive Bayes classifier assumes that there is a feature in the dataset that is independent of all the other variables in the dataset and uses it for prediction. It predicts the place of a particular data in a class of data in the dataset. The class with the highest probability is denoted as the most probable class. This is also known as MAP.

There are three types of naive Bayes algorithms out of which we used Bernoulli naive bayes in the project. Bernoulli Naive Bayes is used for data abiding by a multi-variable Bernoulli distribution in which there can be multiple features, each of which is recognized as a Bernoulli variable. The variables should be evaluated in binary values.

6.2.6.5 SVM:

The end goal of the SVM is to find a hyperplane of data that is fit for the dataset. The dimensions of the output depend on the number of features of the dataset used. It is especially efficient in cases with high number of features and used for classifying complex data which is the case with most real-life data.

```
models = []
models.append(('Decision Tree Classifier', DTC_Classifier))
models.append(('Logistic Regression', LGR_Classifier))
models.append(('Naive Baye Classifier', BNB_Classifier))
models.append(('Support Vector Machine', SVM_Classifier))
```

6.2.7 Training the models

The models are trained with the 20% of the training data supplied. The accuracy, confusion matrix and the classification reports are generated for each of the models and is done with the help of scikit-learn library.

Accuracies for each of the models is as follows:

1) Decision Tree: 100
2) Logistic Regression: 84.67
3) Naive Bayes Classifier: 77.14
4) Support Vector Machine: 85.90s
5) Linear regression: 55.80

```
for i, v in models:
    accuracy = metrics.accuracy_score(Y_train, v.predict(X_train))
    confusion_matrix = metrics.confusion_matrix(Y_train, v.predict(X_train))
    classification = metrics.classification_report(Y_train, v.predict(X_train))
    print()
    print('============================== {} Model Evaluation =============================='.format(i))
    print ("Model Accuracy:" "\n", accuracy)
    print()
    print("Confusion matrix:" "\n", confusion_matrix)
    print()
    print("Classification report:" "\n", classification)
    print()
```

6.2.8 Testing the models

The models trained in the above steps will now undergo testing process. The testing data is now applied to the models and will check the accuracy of the prediction. The model producing the highest accuracy while testing is used for prediction purpose.

```
for i, v in models:
    accuracy = metrics.accuracy_score(Y_test, v.predict(X_test))
    confusion_matrix = metrics.confusion_matrix(Y_test, v.predict(X_test))
    classification = metrics.classification_report(Y_test, v.predict(X_test))
    print()
    print('============================== {} Model Test Results =============================='.format(i))
    print()
    print ("Model Accuracy:" "\n", accuracy)
    print()
    print("Confusion matrix:" "\n", confusion_matrix)
    print()
    print("Classification report:" "\n", classification)
    print()
```

6.2.9 Make prediction using a separate dataset

The prediction result of the model is stored in a variable in an array format. User is prompted to enter some values depending on his choice and the actual values are given out. Then when the second code is run the actual values and the prediction results are printed together.

7. RESULTS

7.1 Training phase – Model Accuracy, Confusion matrix, Classification Report

Decision Tree

Model Accuracy: 100%

Confusion Matrix:

```
============================== Decision Tree Classifier Model Evaluation ==============================
Model Accuracy:
 1.0

Confusion matrix:
 [[ 123    0    0    0    0    0    0    0    0    0]
 [   0  118    0    0    0    0    0    0    0    0]
 [   0    0  792    0    0    0    0    0    0    0]
 [   0    0    0 2235    0    0    0    0    0    0]
 [   0    0    0    0 1211    0    0    0    0    0]
 [   0    0    0    0    0 3767    0    0    0    0]
 [   0    0    0    0    0    0 7431    0    0    0]
 [   0    0    0    0    0    0    0  703    0    0]
 [   0    0    0    0    0    0    0    0   79    0]
 [   0    0    0    0    0    0    0    0    0    7]]
```

Classification report:

```
Classification report:
                precision    recall  f1-score   support

      Analysis       1.00      1.00      1.00       123
      Backdoor       1.00      1.00      1.00       118
           DoS       1.00      1.00      1.00       792
      Exploits       1.00      1.00      1.00      2235
       Fuzzers       1.00      1.00      1.00      1211
       Generic       1.00      1.00      1.00      3767
        Normal       1.00      1.00      1.00      7431
Reconnaissance       1.00      1.00      1.00       703
     Shellcode       1.00      1.00      1.00        79
         Worms       1.00      1.00      1.00         7

      accuracy                           1.00     16466
     macro avg       1.00      1.00      1.00     16466
  weighted avg       1.00      1.00      1.00     16466
```

Logistic Regression

Model Accuracy: 84.67

Confusion Matrix:

```
============================== Logistic Regression Model Evaluation ==============================
Model Accuracy:
 0.8467751730839306

Confusion matrix:
 [[   0    0   18   25   35   43    0    2    0    0]
 [   0    0    2   25   49   37    0    5    0    0]
 [   0    0  311  321   90   49    0   21    0    0]
 [   0    0  293 1492  294   90    0   66    0    0]
 [   0    0   42  172  796  101    6   94    0    0]
 [   0    0    3   69   54 3634    0    7    0    0]
 [   0    0    1    0    0    0 7430    0    0    0]
 [   0    0  122  136  159    6    0  280    0    0]
 [   0    0    3   10   35    0    0   31    0    0]
 [   0    0    0    7    0    0    0    0    0    0]]
```

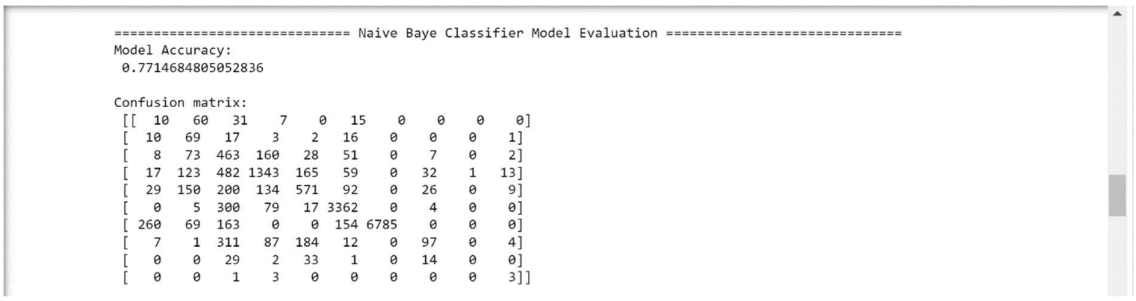
Classification report:

```
Classification report:
                precision    recall  f1-score   support

      Analysis       0.00      0.00      0.00       123
      Backdoor       0.00      0.00      0.00       118
           DoS       0.39      0.39      0.39       792
      Exploits       0.66      0.67      0.66      2235
       Fuzzers       0.53      0.66      0.58      1211
       Generic       0.92      0.96      0.94      3767
        Normal       1.00      1.00      1.00      7431
Reconnaissance       0.55      0.40      0.46       703
     Shellcode       0.00      0.00      0.00        79
         Worms       0.00      0.00      0.00         7

      accuracy                           0.85     16466
     macro avg       0.40      0.41      0.40     16466
  weighted avg       0.83      0.85      0.84     16466
```

Naive Bayes Classifier:

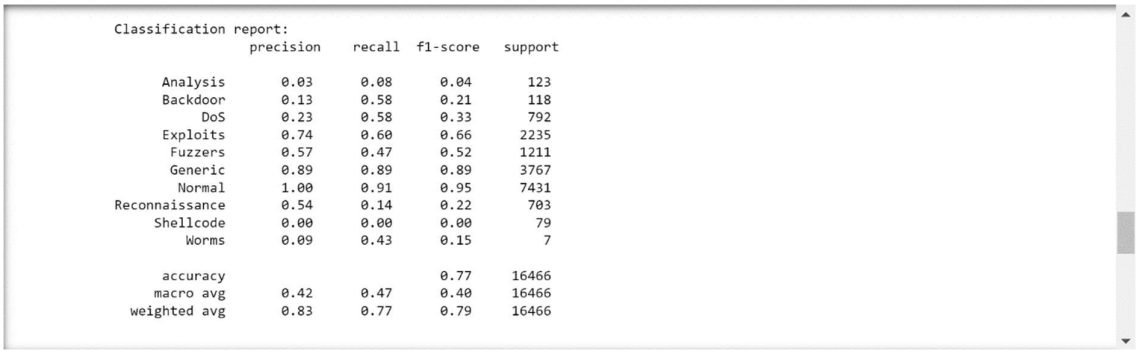
Model Accuracy: 77.14

Confusion matrix:



```
============================== Naive Baye Classifier Model Evaluation ==============================
Model Accuracy:
 0.7714684805052836

Confusion matrix:
 [[  10   60   31    7    0   15    0    0    0    0]
 [  10   69   17    3    2   16    0    0    0    1]
 [   8   73  463  160   28   51    0    7    0    2]
 [  17  123  482 1343  165   59    0   32    1   13]
 [  29  150  200  134  571   92    0   26    0    9]
 [   0    5  300   79   17 3362    0    4    0    0]
 [ 260   69  163    0    0  154 6785    0    0    0]
 [   7    1  311   87  184   12    0   97    0    4]
 [   0    0   29    2   33    1    0   14    0    0]
 [   0    0    1    3    0    0    0    0    0    3]]
```

Classification report:



```
Classification report:
                precision    recall  f1-score   support

      Analysis       0.03      0.08      0.04       123
      Backdoor       0.13      0.58      0.21       118
           DoS       0.23      0.58      0.33       792
      Exploits       0.74      0.60      0.66      2235
       Fuzzers       0.57      0.47      0.52      1211
       Generic       0.89      0.89      0.89      3767
        Normal       1.00      0.91      0.95      7431
Reconnaissance       0.54      0.14      0.22       703
     Shellcode       0.00      0.00      0.00        79
         Worms       0.09      0.43      0.15         7

      accuracy                           0.77     16466
     macro avg       0.42      0.47      0.40     16466
  weighted avg       0.83      0.77      0.79     16466
```

Support Vector Machine:

Model Accuracy: 85.94

Confusion matrix:

```
============================== Support Vector Machine Model Evaluation ==============================
Model Accuracy:
 0.8594072634519616

Confusion matrix:
 [[   0    0    1   36   56    0   28    2    0    0]
 [   0    0    1   24   58    0   30    5    0    0]
 [   0    0   23  514   81    9  113   52    0    0]
 [   0    0   18 1735  258    6  116  102    0    0]
 [   0    0    3  176  899    2   64   67    0    0]
 [   0    0    3   74   48 3632    0   10    0    0]
 [   0    0    0    0    0    0 7431    0    0    0]
 [   0    0    3  146  100    5   18  431    0    0]
 [   0    0    0    9   24    0    0   46    0    0]
 [   0    0    0    7    0    0    0    0    0    0]]
```

Classification Report:

```
Classification report:
                precision    recall  f1-score   support

      Analysis       0.00      0.00      0.00       123
      Backdoor       0.00      0.00      0.00       118
           DoS       0.44      0.03      0.05       792
      Exploits       0.64      0.78      0.70      2235
       Fuzzers       0.59      0.74      0.66      1211
       Generic       0.99      0.96      0.98      3767
        Normal       0.95      1.00      0.98      7431
Reconnaissance       0.60      0.61      0.61       703
     Shellcode       0.00      0.00      0.00        79
         Worms       0.00      0.00      0.00         7

      accuracy                           0.86     16466
     macro avg       0.42      0.41      0.40     16466
  weighted avg       0.83      0.86      0.84     16466
```

7.2 Testing phase – Model Accuracy, Confusion matrix, classification matrix

Decision Tree:

Model Accuracy: 87.55

Confusion Matrix:

```
============================== Decision Tree Classifier Model Test Results ==============================

Model Accuracy:
 0.875565542161358

Confusion matrix:
 [[   48    95   114   175   114     0     0     8     0     0]
 [   73    21    90   154   109     6     0     8     4     0]
 [   90    90  1097  1454   308    55     0   171    32     0]
 [  119   160  1436  6111   517   124     0   334    64    32]
 [   80   109   290   491  3729    46     0    55    51     0]
 [    3     2    81   202    66 14727     0    11    12     0]
 [    0     0     0     0     0     0 29569     0     0     0]
 [    4     5   177   307    49     8     0  2222    20     1]
 [    1     0    34    45    44    10     0    17   145     3]
 [    0     0     0    24     7     2     0     1     2     1]]
```

Classification Report:

```
Classification report:
                precision    recall  f1-score   support

      Analysis       0.11      0.09      0.10       554
      Backdoor       0.04      0.05      0.04       465
           DoS       0.33      0.33      0.33      3297
      Exploits       0.68      0.69      0.68      8897
       Fuzzers       0.75      0.77      0.76      4851
       Generic       0.98      0.98      0.98     15104
        Normal       1.00      1.00      1.00     29569
Reconnaissance       0.79      0.80      0.79      2793
     Shellcode       0.44      0.48      0.46       299
         Worms       0.03      0.03      0.03        37

      accuracy                           0.88     65866
     macro avg       0.52      0.52      0.52     65866
  weighted avg       0.88      0.88      0.88     65866
```

Logistic Regression:

Model Accuracy: 84.57

Confusion matrix:

```
============================== Logistic Regression Model Test Results ==============================

Model Accuracy:
 0.845777791273191

Confusion matrix:
 [[    0     0    65   142   179   154     3    11     0     0]
 [    0     0    17    75   192   163     0    18     0     0]
 [    0     0  1235  1412   326   208     0   116     0     0]
 [    0     0  1117  6098  1012   442     1   227     0     0]
 [    0     0   176   599  3215   439    19   403     0     0]
 [    0     0    12   332   173 14566     0    21     0     0]
 [    0     0     3     2     0     0 29564     0     0     0]
 [    0     0   515   513   706    29     0  1030     0     0]
 [    0     0    21    45   130     0     0   103     0     0]
 [    0     0     1    29     7     0     0     0     0     0]]
```

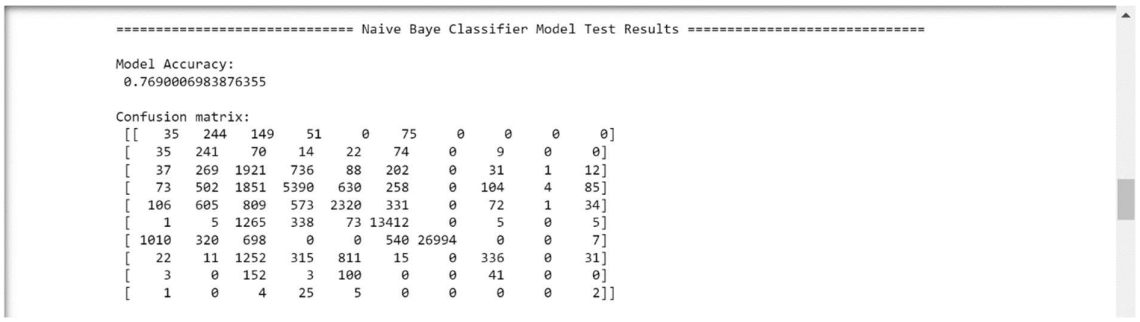
Classification matrix:

```
Classification report:
                precision    recall  f1-score   support

      Analysis       0.00      0.00      0.00       554
      Backdoor       0.00      0.00      0.00       465
           DoS       0.39      0.37      0.38      3297
      Exploits       0.66      0.69      0.67      8897
       Fuzzers       0.54      0.66      0.60      4851
       Generic       0.91      0.96      0.94     15104
        Normal       1.00      1.00      1.00     29569
Reconnaissance       0.53      0.37      0.44      2793
     Shellcode       0.00      0.00      0.00       299
         Worms       0.00      0.00      0.00        37

      accuracy                           0.85     65866
     macro avg       0.40      0.41      0.40     65866
  weighted avg       0.83      0.85      0.84     65866
```

Naive bayes Classifier:

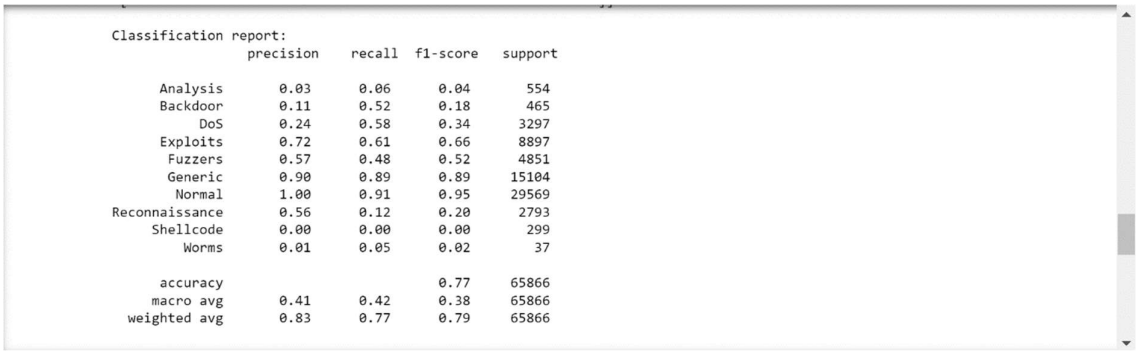
Model Accuracy: 76.90

Confusion Matrix:



```
============================== Naive Baye Classifier Model Test Results ==============================

Model Accuracy:
 0.7690006983876355

Confusion matrix:
 [[   35   244   149    51     0    75     0     0     0     0]
 [   35   241    70    14    22    74     0     9     0     0]
 [   37   269  1921   736    88   202     0    31     1    12]
 [   73   502  1851  5390   630   258     0   104     4    85]
 [  106   605   809   573  2320   331     0    72     1    34]
 [    1     5  1265   338    73 13412     0     5     0     5]
 [ 1010   320   698     0     0   540 26994     0     0     7]
 [   22    11  1252   315   811    15     0   336     0    31]
 [    3     0   152     3   100     0     0    41     0     0]
 [    1     0     4    25     5     0     0     0     0     2]]
```

Classification report:



```
Classification report:
                precision    recall  f1-score   support

      Analysis       0.03      0.06      0.04       554
      Backdoor       0.11      0.52      0.18       465
           DoS       0.24      0.58      0.34      3297
      Exploits       0.72      0.61      0.66      8897
       Fuzzers       0.57      0.48      0.52      4851
       Generic       0.90      0.89      0.89     15104
        Normal       1.00      0.91      0.95     29569
Reconnaissance       0.56      0.12      0.20      2793
     Shellcode       0.00      0.00      0.00       299
         Worms       0.01      0.05      0.02        37

      accuracy                           0.77     65866
     macro avg       0.41      0.42      0.38     65866
  weighted avg       0.83      0.77      0.79     65866
```

Support Vector Machine:

Model Accuracy: 85.80

Confusion matrix:

```
============================== Support Vector Machine Model Test Results ==============================

Model Accuracy:
 0.8580603042540916

Confusion matrix:
 [[    0     0     8   178   222     0   131    15     0     0]
 [    0     0     4   101   238     0    91    31     0     0]
 [    0     0    79  2214   349    23   434   198     0     0]
 [    0     0    57  7011   963    22   485   359     0     0]
 [    0     0    14   714  3656     4   204   259     0     0]
 [    0     0    12   342   148 14556     3    43     0     0]
 [    0     0     0     0     3     0 29566     0     0     0]
 [    0     0     8   628   451    20    37  1649     0     0]
 [    0     0     0    40    74     0     0   185     0     0]
 [    0     0     0    26     8     2     0     1     0     0]]
```

Classification report:

```
Classification report:
                precision    recall  f1-score   support

      Analysis       0.00      0.00      0.00       554
      Backdoor       0.00      0.00      0.00       465
           DoS       0.43      0.02      0.05      3297
      Exploits       0.62      0.79      0.70      8897
       Fuzzers       0.60      0.75      0.67      4851
       Generic       1.00      0.96      0.98     15104
        Normal       0.96      1.00      0.98     29569
Reconnaissance       0.60      0.59      0.60      2793
     Shellcode       0.00      0.00      0.00       299
         Worms       0.00      0.00      0.00        37

      accuracy                           0.86     65866
     macro avg       0.42      0.41      0.40     65866
  weighted avg       0.83      0.86      0.83     65866
```

Linear Regression Training and testing accuracies:

```
1 LIR_Classifier.score(mlrx_train,mlry_train)
```

```
0.5570711022500587
```

```
1 LIR_Classifier.score(mlrx_test,mlry_test)
```

```
0.5579053349947546
```

7.3 Providing User interaction:

```
import colorama
from colorama import Fore

pred_df = pd.concat([int_preddf,encpred],axis=1)

while(True):
    choice = input(Fore.BLUE+'Enter \'single\' for predicting single value\nEnter \'range\' to predict a range of values : '
    if(choice=='range'):
        print()
        start,end = map(int,input(Fore.BLUE+'Enter the range for prediction between [0,175340]: ').split())
        prediction = pred_df[start:end]
        tar = target[start:end]
        break
    elif(choice=='single'):
        predict_column = int(input(Fore.BLUE+'Enter the value between [0,175340]: '))
        prediction = pred_df[predict_column:predict_column+1]
        tar = target[predict_column:predict_column+1]
        break
    else:
        print(Fore.RED+'Enter correct choice')

tar = list(tar)
print(tar)
prediction
```

```
Enter 'single' for predicting single value
Enter 'range' to predict a range of values : range

Enter the range for prediction between [0,175340]: 54500 54530
['Exploits', 'Fuzzers', 'DoS', 'DoS', 'Fuzzers', 'Exploits', 'Exploits', 'Reconnaissance', 'Fuzzers', 'Fuzzers', 'Exploits', 'E
xploits', 'Fuzzers', 'Exploits', 'Reconnaissance', 'Exploits', 'DoS', 'Exploits', 'Exploits', 'Exploits', 'Fuzzers', 'Exploit
s', 'Reconnaissance', 'Reconnaissance', 'Exploits', 'Exploits', 'Exploits', 'Analysis', 'Fuzzers', 'Exploits']
```

	id	dur	spkts	dpkts	sbytes	dbytes	rate	sttl	dttl	sload	...	is_ftp_login	ct_ftp_cmd	ct_flw_http_mthd
54500	-0.655319	-0.209774	-0.133677	-0.172047	-0.049465	-0.103923	0.934663	0.723268	-0.720406	0.671841	...	-0.11859	-0.11859	-0.189768
54501	-0.655299	7.769917	3.592018	-0.172047	0.177902	-0.103923	-0.576759	0.723268	-0.720406	-0.389940	...	-0.11859	-0.11859	-0.189768
54502	-0.655279	-0.209774	-0.133677	-0.172047	-0.049465	-0.103923	0.430836	0.723268	-0.720406	0.317903	...	-0.11859	-0.11859	-0.189768
54503	-0.655260	-0.209774	-0.133677	-0.172047	-0.049465	-0.103923	0.178922	0.723268	-0.720406	0.140934	...	-0.11859	-0.11859	-0.189768
54504	-0.655240	-0.209773	-0.133677	-0.172047	-0.050015	-0.103923	0.027774	0.723268	-0.720406	-0.169115	...	-0.11859	-0.11859	-0.189768
54505	-0.655220	-0.119204	-0.089845	-0.081351	-0.048527	-0.077791	-0.576644	-1.141901	1.560002	-0.389950	...	-0.11859	-0.11859	-0.189768
54506	-0.655200	-0.209774	-0.133677	-0.172047	-0.049465	-0.103923	0.934663	0.723268	-0.720406	0.671841	...	-0.11859	-0.11859	-0.189768
54507	-0.655180	-0.102808	-0.075235	-0.099490	-0.047383	-0.101459	-0.576671	0.723268	1.560002	-0.389941	...	-0.11859	-0.11859	-0.189768
54508	-0.655161	-0.209775	-0.133677	-0.172047	-0.049465	-0.103923	5.469112	0.723268	-0.720406	3.857283	...	-0.11859	-0.11859	-0.189768
54509	-0.655141	-0.209773	-0.133677	-0.172047	-0.050015	-0.103923	0.094951	0.723268	-0.720406	-0.144576	...	-0.11859	-0.11859	-0.189768
54510	-0.655121	-0.209774	-0.133677	-0.172047	-0.049465	-0.103923	0.934663	0.723268	-0.720406	0.671841	...	-0.11859	-0.11859	-0.189768
54511	-0.655101	-0.209775	-0.133677	-0.172047	-0.049465	-0.103923	5.469112	0.723268	-0.720406	3.857283	...	-0.11859	-0.11859	-0.189768
54512	-0.655082	-0.083191	-0.075235	-0.117630	-0.045014	-0.102057	-0.576709	0.723268	1.560002	-0.389927	...	-0.11859	-0.11859	-0.189768
54513	-0.655062	-0.209774	-0.133677	-0.172047	-0.049465	-0.103923	0.934663	0.723268	-0.720406	0.671841	...	-0.11859	-0.11859	-0.189768
54514	-0.655042	-0.209773	-0.133677	-0.172047	-0.049649	-0.103923	0.094951	0.723268	-0.720406	0.006438	...	-0.11859	-0.11859	-0.189768
54515	-0.655022	-0.209775	-0.133677	-0.172047	-0.049465	-0.103923	5.469112	0.723268	-0.720406	3.857283	...	-0.11859	-0.11859	-0.189768
54516	-0.655003	-0.209774	-0.133677	-0.172047	-0.049465	-0.103923	1.438491	0.723268	-0.720406	1.025779	...	-0.11859	-0.11859	-0.189768
54517	-0.654983	0.104134	-0.046014	0.009345	-0.040780	0.029565	-0.576721	-1.141901	1.560002	-0.389939	...	-0.11859	-0.11859	1.236347
54518	-0.654963	-0.209774	-0.133677	-0.172047	-0.049465	-0.103923	0.934663	0.723268	-0.720406	0.671841	...	-0.11859	-0.11859	-0.189768
54519	-0.654943	-0.209775	-0.133677	-0.172047	-0.049465	-0.103923	5.469112	0.723268	-0.720406	3.857283	...	-0.11859	-0.11859	-0.189768
54520	-0.654924	0.896966	0.114703	-0.172047	-0.005086	-0.103923	-0.576790	0.723268	-0.720406	-0.389927	...	-0.11859	-0.11859	-0.189768
54521	-0.654904	-0.209774	-0.133677	-0.172047	-0.049465	-0.103923	1.438491	0.723268	-0.720406	1.025779	...	-0.11859	-0.11859	-0.189768
54522	-0.654884	-0.081975	-0.075235	-0.099490	-0.047383	-0.101459	-0.576695	0.723268	1.560002	-0.389947	...	-0.11859	-0.11859	-0.189768
54523	-0.654864	0.070177	-0.075235	-0.099490	-0.047383	-0.101459	-0.576763	0.723268	1.560002	-0.389961	...	-0.11859	-0.11859	-0.189768
54524	-0.654845	-0.156689	-0.075235	-0.117630	-0.044762	-0.102057	-0.576556	0.723268	1.560002	-0.389859	...	-0.11859	-0.11859	1.236347
54525	-0.654825	-0.209774	-0.133677	-0.172047	-0.049465	-0.103923	1.438491	0.723268	-0.720406	1.025779	...	-0.11859	-0.11859	-0.189768
54526	-0.654805	-0.065410	-0.002182	-0.081351	0.029932	-0.097672	-0.576632	0.723268	1.560002	-0.389365	...	-0.11859	-0.11859	-0.189768
54527	-0.654785	-0.209774	-0.133677	-0.172047	-0.049465	-0.103923	1.438491	0.723268	-0.720406	1.025779	...	-0.11859	-0.11859	-0.189768
54528	-0.654766	-0.209774	-0.133677	-0.172047	-0.049465	-0.103923	1.438491	0.723268	-0.720406	1.025779	...	-0.11859	-0.11859	-0.189768
54529	-0.654746	-0.170947	-0.075235	-0.117630	-0.045941	-0.102057	-0.576459	0.723268	1.560002	-0.389849	...	-0.11859	-0.11859	1.236347

7.4 Prediction Results:

An accuracy of 87.5% is achieved by Decision tree algorithm which records the highest among other algorithms used for comparative study.

```
DTC_prediction_result = DTC_Classifier.predict(prediction)
for i in range(len(DTC_prediction_result)):
    print(tar[i],DTC_prediction_result[i])
```

```
Exploits DoS
Fuzzers Exploits
DoS Fuzzers
DoS DoS
Fuzzers Fuzzers
Exploits Generic
Exploits DoS
Reconnaissance Exploits
Fuzzers Fuzzers
Fuzzers Fuzzers
Exploits DoS
Exploits Fuzzers
Fuzzers Exploits
Exploits DoS
Reconnaissance Fuzzers
Exploits Fuzzers
DoS Fuzzers
Exploits Exploits
Exploits DoS
Exploits Fuzzers
Fuzzers Fuzzers
Exploits Fuzzers
Reconnaissance Exploits
Reconnaissance Backdoor
Exploits Shellcode
Exploits Fuzzers
Exploits Reconnaissance
Analysis Fuzzers
Fuzzers Fuzzers
Exploits Generic
```

8. CONCLUSION

With increasing network traffic in the current scenario, we have developed an IDS in this project by making a comparative study with 5 machine learning algorithms. Out of the five algorithms Decision Tree algorithm produced the highest accuracy of 87.5%. From this we can conclude that given a packet information related to 10 type of attack categories 7 out of 8 predictions are true on an average. Using decision tree, attack types such as DOS, worms, backdoors have very low accuracy score. We can very accurately predict which is a normal packet or a generic packet each of them showing 100% and 98% accuracies.

9. FUTURE ENHANCEMENTS

Having an accuracy less than 90% is sometimes a bit alarming. The accuracy of the IDS can be improved with new techniques and accuracy boosting techniques. Efficient methods can be implemented rather than using traditional approaches for making predictions using ML. New attack categories have to be detected with improved accuracies. The categories like DOS, worms, backdoors accuracies should be boosted. The project can be more improved by finding the most important attributes required for making the prediction. This way it is easy to make prediction basing on few variables rather than having a bunch of irrelevant variables which sometimes might lead to serious and complicated mathematical calculations and unnecessary extra work to be done.

10.BIBLIOGRAPHY

1) Dimitrios Damopoulos, Sofia A. Menesidou, Georgios Kambourakis, Maria Papadaki, Nathan Clarke, Stefanos Gritzalis. Evaluation of anomaly-based IDS for mobile devices using machine learning classifiers DoI: 16 June 2011 link: https://doi.org/10.1002/sec.341

2) Deepika P Vinchurkar, Alpa Reshamwala. A Review of Intrusion Detection System Using Neural Network and Machine Learning Technique DoI: Nov 2012 Link: https://www.researchgate.net/profile/Alpa-Reshamwala/publication/233943805_A_Review_of_Intrusion_Detection_System_Using_Neural_Network_and_Machine_Learning_Technique/links/02bfe50d2f409cdafa000000/A-Review-of-Intrusion-Detection-System-Using-Neural-Network-and-Machine-Learning-Technique.pdf

3) Suad Mohammed Othman, Fadl Mutaher Ba-Alwi, Nabeel T. Alsohybe, Amal Y. Al-Hashida. Intrusion detection model using machine learning algorithm on Big Data environment DoI: 10/11/2018 Link: https://link.springer.com/article/10.1186/s40537-018-0145-4

4) S. Choudhury and A. Bhowal, "Comparative analysis of machine learning algorithms along with classifiers for network intrusion detection," 2015 International Conference on Smart Technologies and Management for Computing, Communication, Controls, Energy and Materials (ICSTM), 2015, pp. 89-95, doi: 10.1109/ICSTM.2015.7225395.

5) Magán-Carrión, Roberto, Daniel Urda, Ignacio Díaz-Cano, and Bernabé Dorronsoro. 2020. "Towards a Reliable Comparison and Evaluation of Network Intrusion Detection Systems Based on Machine Learning Approaches" Applied Sciences 10, no. 5: 1775. https://doi.org/10.3390/app10051775

6) K. A. Taher, B. Mohammed Yasin Jisan and M. M. Rahman, "Network Intrusion Detection using Supervised Machine Learning Technique with Feature Selection," 2019 International Conference on Robotics,Electrical and Signal Processing Techniques (ICREST), 2019, pp. 643-646, doi: 10.1109/ICREST.2019.8644161. Link: https://www.academia.edu/download/65354415/IRJET_V7I1259.pdf

7) https://www.hindawi.com/journals/scn/2019/7130868/
https://towardsdatascience.com/building-an-intrusion-detection-system-using-deep-learning-b9488332b321

8) https://www.codespeedy.com/intrusion-detection-model-using-machine-learning-algorithm-in-python/

9) https://cybersecurity.springeropen.com/articles/10.1186/s42400-019-0038-7#availability-of-data-and-materials

10) https://cybersecurity.springeropen.com/articles/10.1186/s42400-019-0038-7#availability-of-data-and-materials

11) https://www.geeksforgeeks.org/intrusion-detection-system-using-machine-learning-algorithms/

12) https://www.geeksforgeeks.org/best-python-libraries-for-machine-learning/

13) https://anderfernandez.com/en/blog/code-decision-tree-python-from-scratch/
14) https://www.youtube.com/watch?v=qDcl-FRnwSU
15) https://arxiv.org/ftp/arxiv/papers/2101/2101.05067.pdf
16) https://www.geeksforgeeks.org/principal-component-analysis-with-python/

11. PLAGIARISM REPORTS

ORIGINALITY REPORT

19%	14%	5%	16%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	Submitted to Educational Service District 105 Student Paper	6%
2	www.coursehero.com Internet Source	2%
3	tcalive.in Internet Source	1%
4	Submitted to Cornell University Student Paper	1%
5	vtechworks.lib.vt.edu Internet Source	1%
6	scholar.archive.org Internet Source	1%
7	repositorio.uniandes.edu.co Internet Source	1%
8	Submitted to Rowan University Student Paper	<1%
9	www.scribd.com Internet Source	<1%