DETECTION OF CYBER ATTACK IN NETWORK USING MACHINE LEARNING TECHNIQUES

A project report submitted in partial fulfilment of the requirements for the award of degree of

BACHELOR OF TECHNOLOGY

IN

COMPUTER SCIENCE AND ENGINEERING

Submitted by
M YOGESH – 121810316054

Under the esteemed guidance of

NEELIMA SANTOSHI K

Assistant Professor

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

GITAM

(Deemed to be University)

VISAKHAPATNAM

MARCH – 2022

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

GITAM INISTITUTE OF TECHNOLOGY

GITAM

(Deemed to be University)

CERTIFICATE

This is to certify that the project report entitled "**DETECTION OF CYBER ATTACK IN NETWORK USING MACHINE LEARNING TECHNIQUES**" is a bonafide record of work carried out by **M. YOGESH (121810316054), M. CHANDRA SAGAR (121810316028)** students submitted in partial fulfilment of requirement for the award of degree of Bachelors of Technology in Computer Science and Engineering.

Project Guide

Neelima

Neelima Santoshi K

Assistant Professor

Department of Computer Science & Engineering
GITAM Institute of Technology
Gandhi Institute of Technology and Management (GITAM)
(Deemed to be University)
Visakhapatnam-530 045

R·Sireesha

Head of the Department

Dr. R.Sireesha

Professor

Head of the Department
Department of Computer Science & Engineering
GITAM Institute of Technology
Gandhi Institute of Technology and Management (GITAM)
(Deemed to be University)
Visakhapatnam-530 045

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

GITAM INISTITUTE OF TECHNOLOGY

GITAM

(Deemed to be University)

DECLARATION

We, hereby declare that the project report entitled "**DETECTION OF CYBER ATTACK IN NETWORK USING MACHINE LEARNING TECHNIQUES**" is an original work done in the Department of Computer Science and Engineering, GITAM Institute of Technology, GITAM (Deemed to be University) submitted in partial fulfilment of the requirements for the award of the degree of B.Tech. in Computer Science and Engineering. The work has not been submitted to any other college or University for the award of any degree or diploma.

Date: 10-04-2022

Registration No(s)	**Name(s)**	**Signature(s)**
121810316054	**M.YOGESH**	**Yogesh**

ACKNOWLEDGEMENT

We would like to thank our project guide **Mrs. Neelima Santoshi K**, Assistant Professor, Department of CSE for his stimulating guidance and profuse assistance. We shall always cherish our association with him for his guidance, encouragement and valuable suggestions throughout the progress of this work. We consider it a great privilege to work under his guidance and constant support.

We also express our thanks to the project reviewers **Dr. Srinivasa Rao Bendi**, Assistant Professor, and **Srinivas Y**, Professor, Department of CSE, GITAM (Deemed to be University) for their valuable suggestions and guidance for doing our project.

We consider it is a privilege to express our deepest gratitude to Prof. R. SIREESHA, Head of the Department, Computer Science Engineering for his valuable suggestions and constant motivation that greatly helped us to successfully complete this project. Our sincere thanks to Dr. C. Dharma Raj, Principal, GITAM Institute of Technology, GITAM (Deemed to be University) for inspiring us to learn new technologies and tools.

Finally, we deem it a great pleasure to thank one and all that helped us directly and indirectly throughout this project.

M YOGESH **(Regno: 121810316054)**

TABLE OF CONTENTS

CHAPTER NO. **TITLE** **PAGE NUMBER**

LIST OF FIGURES:

LIST OF TABLES:

ABSTRACT

Contrasted with the past, upgrades in PC and correspondence improvements have given extensive and propelled changes. The use of latest improvements gives exceptional advantages to people, organizations, and governments, be that as it is, messes a few up against them. For instance, the safety of significant data, security of positioned away statistics stages, accessibility of statistics and so forth. Contingent upon those problems, virtual worry primarily based totally oppression is one of the maximum significant problems on this day and age. Digital worry, which made a first-rate deal of problems people and establishments, has arrived at a stage that would undermine open and state safety with the aid of using extraordinary gatherings, for example, criminal association, proficient humans and virtual activists. Along those lines, Intrusion Detection Systems (IDS) has been created to preserve a strategic distance from virtual assaults. Right now, mastering the bolster support vector machine (SVM) calculations had been applied to understand port sweep endeavours depending on the new CICIDS2017 dataset with 97.80%, 69.79% precision rates had been achieved individually. Rather than SVM we are able to introduce a few different algorithms like the random forest, CNN, ANN in which those algorithms can accumulate accuracies.

1. INTRODUCTION

Nowadays machine learning is growing rapidly make people dependent on machine learning techniques and classifiers than ever before. And same time the number of security intrusions has growing rapidly. Therefore, the security is important. This says that the security and reliability of devices, as well as effective protection against various networks attacks that create vulnerabilities in installed security system. the intrusion detection system is considered one of the machine learning tools to monitor suspicious activities. in the modern world everyone is using their internet through smartphones and laptops so that the internet facility should be 24 ×7 without interruption. Before finding malicious attacks, one should know about the basic nature of such attacks.

The utilization of new developments give mind blowing benefits to individuals associations and legislatures in any case mess some facing them for example the assurance of critical information security of taken care of data stages availability of data, etc dependent upon these issues computerized dread based abuse is one of the main issues nowadays advanced dread which made a lot of issues individuals and foundations has shown up at a level that could sabotage open and public safety by various social events, for instance, criminal affiliation capable individuals and computerized activists thusly interruption discovery frameworks ids has been made to avoid computerized attacks.

ı.1 STRUCTURE OF PROJECT (SYSTEM ANALYSIS)

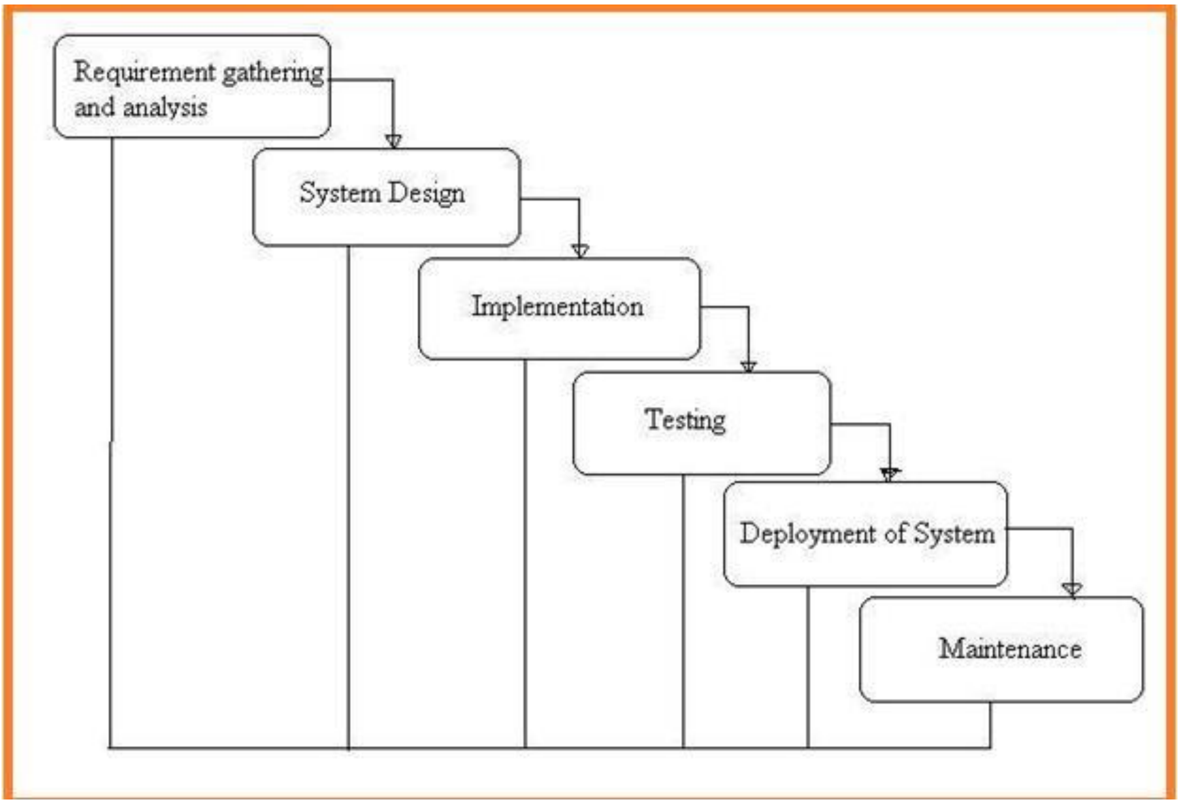


Fig.1.1.1 structure analysis of project

2. LITERATURE REVIEW

R. Christopher, "Port scanning techniques and the defense against them," SANS Institute, 2001.

Port Scanning is one of the most popular techniques attackers use to discover services that they can exploit to break into systems. All systems that are connected to a LAN or the Internet via a modem run services that listen to well-known and not so well-known ports. By port scanning, the attacker can find the following information about the targeted systems: what services are running, what users own those services, whether anonymous logins are supported, and whether certain network services require authentication. Port scanning is accomplished by sending a message to each port, one at a time. The kind of response received indicates whether the port is used and can be probed for further weaknesses.

S. Staniford, J. A. Hoagland, and J. M. McAlerney, "Practical automated detection of stealthy portscans," Journal of Computer Security, vol. 10, no. 1-2, pp. 105–136, 2002.

Port scanning is a common activity of considerable importance. It is often used by computer attackers to characterize hosts or networks which they are considering hostile activity against. Thus, it is useful for system administrators and other network defenders to detect port scans as possible preliminaries to a more serious attack. It is also widely used by network defenders to understand and find vulnerabilities in their own networks. Thus, it is of considerable interest to attackers to determine whether or not the defenders of a network are port scanning it regularly.

M. C. Raja and M. M. A. Rabbani, "Combined analysis of support vector machine and principal component analysis for ids," in IEEE International Conference on Communication and Electronics Systems, 2016, pp. 1–5.

Compared to the past security of networked systems has become a critical universal issue that influences individuals, enterprises and governments. The rate of attacks against networked systems has increased melodramatically, and the strategies used by the attackers are continuing to evolve. For example, the privacy of important information, security of stored data platforms, availability of knowledge etc. Depending on these problems, cyber terrorism is one of the most important issues in today's world. Cyber terror, which caused a lot of problems to individuals and institutions, has reached a level that could threaten public and country security by various groups such as criminal organizations, professional persons and cyber activists. Intrusion detection is one of the solutions against these attacks.

S. Aljawarneh, M. Aldwairi, and M. B. Yassein, "Anomaly-based intrusion detection system through feature selection analysis and building hybrid efficient model," Journal of Computational Science, vol. 25, pp. 152–160, 2018.

n network security, intrusion detection plays an important role. Feature subsets obtained by different feature selection methods will lead to different accuracy of intrusion detection. Using individual feature selection method can be unstable in different intrusion detection scenarios. In this paper, the idea of ensemble is applied to feature selection to adjust feature subsets.

I. Sharafaldin, A. H. Lashkari, and A. A. Ghorbani, "Toward generating a new intrusion detection dataset and intrusion traffic characterization." in ICISSP, 2018, pp. 108–116.

With exponential growth in the size of computer networks and developed applications, the significant increasing of the potential damage that can be caused by launching attacks is becoming obvious. Meanwhile, Intrusion Detection Systems and Intrusion Prevention Systems are one of the most important defense tools against the sophisticated and ever-growing network attacks. Due to the lack of adequate dataset, anomaly-based approaches in intrusion detection systems are suffering from accurate deployment, analysis and evaluation. Based on our study over eleven available datasets since 1998, many such datasets are out of date and unreliable to use. Some of these datasets suffer from lack of traffic diversity and volumes, some of them do not cover the variety of attacks, while others anonymized packet information and payload which cannot reflect the current trends, or they lack feature set and metadata. However, it does not represent new network protocols since nearly 70% of today's network traffics are HTTPS and there are no HTTPS traces in this dataset. Moreover, the distribution of the simulated attacks is not based on real world statistics (Ali Shiravi and Ghorbani, 2012). (University of New South Wales 2013): This dataset includes normal training and validating data and 10 attacks per vector (Creech and Hu, 2013). It contains FTP and SSH password brute force, Java based Meterpreter, Add new Superuser, Linux Meterpreter payload and C100 Web Shel attacks.

3. PROBLEM ANALYSIS

3.1 Existing Approach

Blameless Bayes and Principal Component Analysis (PCA) were been used with the KDD99 dataset by Almansob and Lomte [9]. Similarly, PCA, SVM, and KDD99 were used Chithik and Rabbani for IDS [10]. In Aljawarneh et al's. Paper, their assessment and examinations were conveyed reliant on the NSL-KDD dataset for their IDS model [11] Composing inspects show that KDD99 dataset is continually used for IDS [6]– [10]. There are 41 highlights in KDD99 and it was created in 1999. Consequently, KDD99 is old and doesn't give any data about cutting edge new assault types, example, multi day misuses and so forth. In this manner we utilized a cutting-edge and new CICIDS2017 dataset [12] in our investigation.

3.2 Drawbacks

1) It has Strict Regulations
2) Difficult to perform with for non-technical users
3) Restrictive for resources
4) It Constantly needs Patching
5) Constantly being attacked by the malicious users

3.3 Proposed System

Main steps of the algorithm:

1) firstly, Normalization of every dataset

2) later, it Convert that dataset into testing and training.

3) develop a attack models with the help of using RF, decision tree and SVM algorithms.

4) at last, Evaluate every model’s performance

3.4 Advantages

- It Protects from malicious attacks on your web pages.
- Removing and guaranteeing malicious elements in a pre-existing network.
- It Prevents users from unauthorized user access to the networks.
- Deny's programs from specific resources that could be infected.
- By this it can be Secure confidential information

4. OVERVIEW OF TECHNOLOGIES

4.1 Software requirements

The utilitarian necessities or the general portrayal records incorporate the item point of perspective and features, OS and working environment, gaphic requirements, plan requirements and client documentation. The appointment of necessities and execution imperatives gives the overall outline of the task concerning what the areas of strength and shortfall are and how to handle them Software and Hardware Requirement.

- Python idel 3.7 version (or)
- Anaconda 3.7 (or)
- Jupiter (or)
- Google colab

4.2 Hardware requirements

Minimum HW are dependent on the specific software that developed by a given Enthought Python / Canopy / Virtual studio Code user. Applications that need to collect large arrays/objects in memory will needs more RAM, whereas applications that have to perform numerous calculations or tasks more quickly will require a faster processor.

Operating system	**: windows, linux**
Processor	**: minimum intel i3**
Ram	**: minimum 4 gb**
Hard disk	**: minimum 250gb**

4.3 About Dataset

Dengue data:

The data set is collected from UCI Machine Learning Repository. It has TWO datasets those are dengue features and dengue labels.

The data is collected 1869 records for dengue features and 1456 records are collected for dengue labels.

City, year, weekofyear, ndvi_ne, ndvi_nw, ndvi_se, ndvi_sw, precipitation_amt_mm,reanalysis_air_temp_k,reanalysis_dew_point_temp_k,reanalysis_precip_amt_kg_per_m2,reanalysis_relative_humidity_percent, station_avg_temp_c

Sj, 2008, 18, 29-4-2008, -0.0189, -0.0189, o. 1027286, 0.0912, 78.6, 298.4928571, 298.55, 294.5271429, 25.37, 78.78142857, 26.52857143

Above all bold names are the dataset column names and below all values are the dataset values of dengue features dataset.

City, year, weekofyear, total cases

Sj, 1990, 18, 4

Sj, 1990,19,5

Above all bold names are the dataset column names and below all values are the dataset values of dengue labels dataset.

4.4 Algorithms

- **Random forest**
- **Support vector machine**
- **Dession tree**

5. METHODOLOGY

5.1 Logistic Regression

This algorithm gives perceivability into discrete arrangements of classes and uses the sigmoid capacity to recover the stamping worth of at least 2 classes. There are various sorts of this algorithm, like,

- Binary
- Multi
- Ordinal.

Binary Logistic Regression (BLR) is utilized in this paper. Sigmoid Function is utilized in this algorithm and this guides a worth to another esteem and these qualities scale from 0 to 1. The sigmoid capacity is given by:

$$S(z) = \frac{1}{1+e^{-z}}$$

Fig.5.1.1 Sigmoid Function

Here S(z)is the yield somewhere in the range of 0 and 1, z is the function’s input. What's more, e is the regular log's base. An edge esteem called the choice bound is chosen to map the likelihood score which the order work gets back to a discrete class.

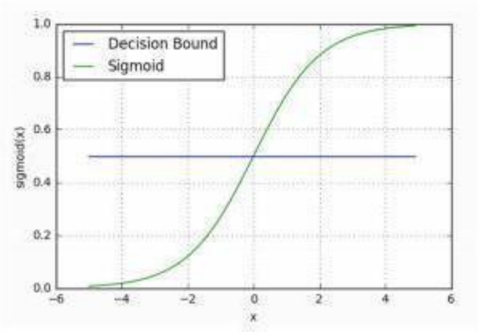


Fig.5.1.2 Logistic Regression

From these sigmoid capacities and choice limits, we can process the forecast result of the characterization by the Logistic Regression model. A resource separation utilizes the sigmoid capacity to change over the outcome into a number of chances; the point is to diminish work expenses to accomplish better openings. Cost work is determined as displayed in

$$\text{Cost}\left(h_\theta(x),\, y\right) = \begin{cases} \log\left(h_\theta(x)\right), & y = 1, \\ -\log\left(1 - h_\theta(x)\right), & y = 0. \end{cases}$$

Fig.5.1.3 Cost work

This calculation was carried out by bringing in the library Logistic Regression from Scikit learn in the way: from sklearn.linear_model import Logistic Relapse. The classifier was then fit on the preparation elements and marks. The work predict_probability was utilized to assess the likelihood. The capacity anticipate was utilized to make the genuine expectations for class names.

5.2 System Design

UML Diagrams

The System Design Document portrays the framework prerequisites, working environment, system and subsystem archt., records and information base plan, input designs, yield formats, human-machine interface, handling logic, and outer connection interfaces.

5.2.1 Use Case Diagram

A use case diag. in the Unified Modeling Language is a sort of behavioral chart characterized by and made from a Use-case analysis. Its motivation is to introduce a graphical outline of the usefulness given by a framework regarding entertainers, their objectives (addressed as use cases), and any conditions between those utilization cases. The primary motivation behind a use case chart is to show what framework capacities are performed for which actor. Jobs of the actors in the framework can be portrayed.

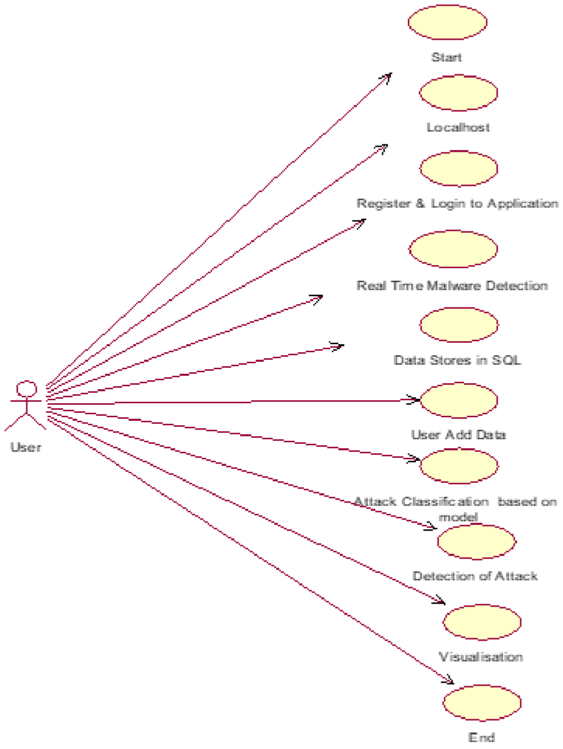

Start
Localhost
Register & Login to Application
Real Time Malware Detection
Data Stores in SQL
User
User Add Data
Attack Classification based on model
Detection of Attack
Visualisation
End

Fig.5.2.1.1: Use Case Diagram

5.2.2 Class Diagrams

In programming, a class diag. in the Unified Modelling Language (UML) is a kind of static design outline that depicts the construction of a framework by showing the framework's classes, their properties, tasks (or techniques), and the connections among the classes. It makes sense of which class contains data.

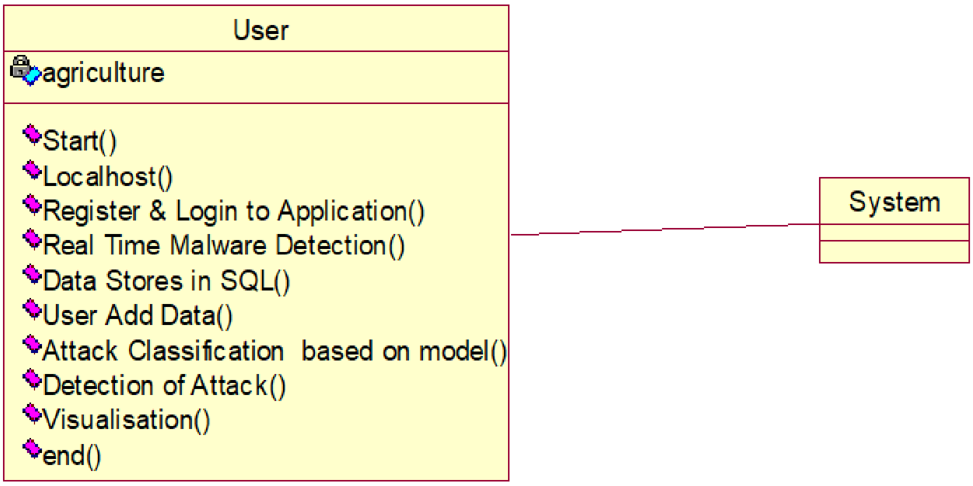


Fig.5.2.2.1: Class Diagram

6.2.3 Sequence Diagram

A sequence diagram in Unified Modelling Language (UML) is a sort of collaboration chart that shows how cycles work with each other and in what request. It is a build of a Message Sequence Chart. sequence diagrams are now and then called event diagrams, Event scenarios, and timing diagm.

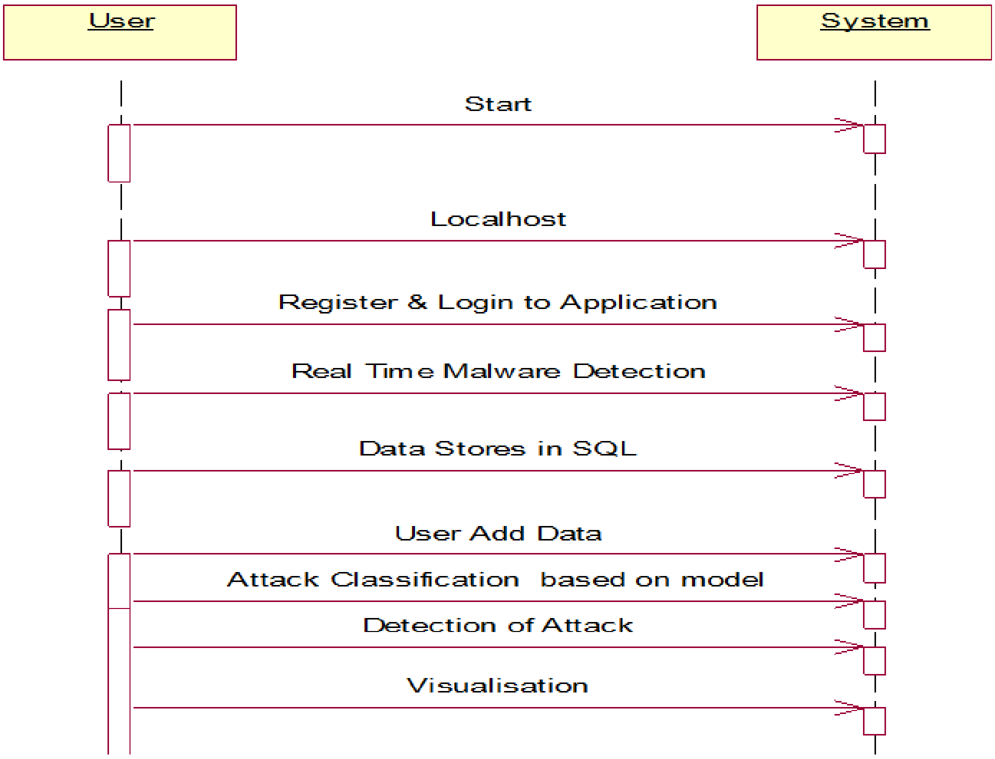

User
System
Start
Localhost
Register & Login to Application
Real Time Malware Detection
Data Stores in SQL
User Add Data
Attack Classification based on model
Detection of Attack
Visualisation

Fig.5.2.3.1: Sequence Diagram

6. IMPLEMENTATION

6.1 Flow Chat

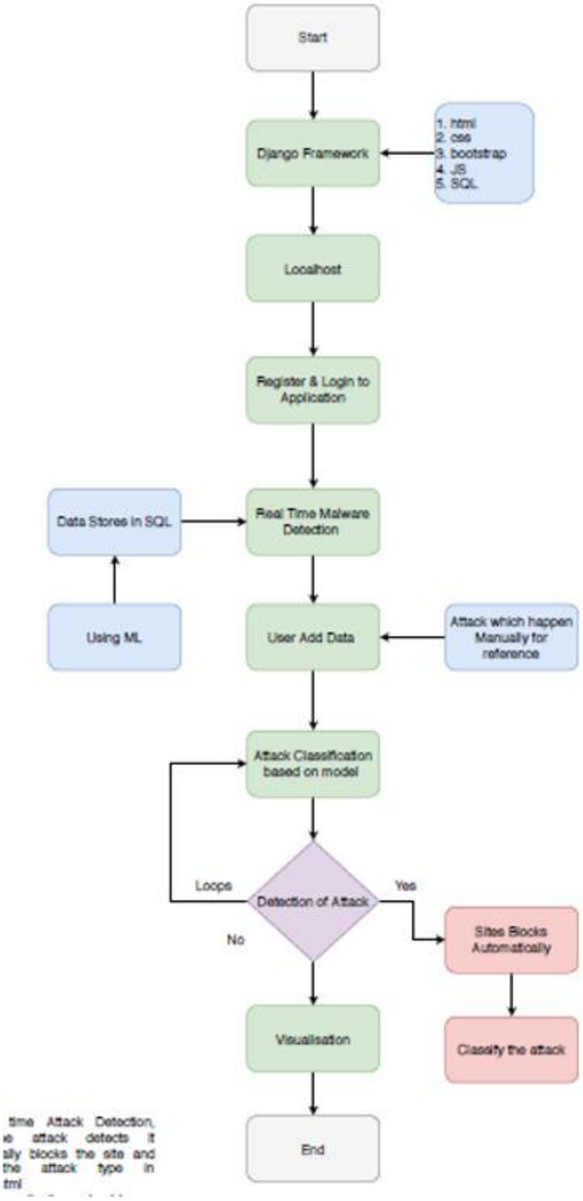


Fig.6.1.1 implementation flow chart

The Implementation is Phase where we attempt to give the useful result of the work done in planning stage and a large portion of Coding in Business rationale lay coms right into it in this stage its primary and critical piece of the venture.

7. CODING & TESTING

7.1 Coding

```
from tkinter import messagebox
from tkinter import *
from tkinter import simpledialog
import tkinter
from tkinter import filedialog
import matplotlib.pyplot as plt
from tkinter.filedialog import askopenfilename
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
import numpy as np
import pandas as pd
from genetic_selection import GeneticSelectionCV
from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix
from sklearn import svm
from keras.models import Sequential
from keras.layers import Dense
import time
main = tkinter.Tk()
main.title("Android Malware Detection")
main.geometry("1300x1200")
global filename
global train
global svm_acc, nn_acc, svmga_acc, annga_acc
global X_train, X_test, y_train, y_test
global svmga_classifier
global nnga_classifier
global svm_time,svmga_time,nn_time,nnga_time
def upload():
    global filename
    filename = filedialog.askopenfilename(initialdir="dataset")
    pathlabel.config(text=filename)
    text.delete('1.0', END)
    text.insert(END,filename+" loaded\n")
def generateModel():
    global X_train, X_test, y_train, y_test
    text.delete('1.0', END)
    train = pd.read_csv(filename)
    rows = train.shape[0]  # gives number of row count
    cols = train.shape[1]  # gives number of col count
    features = cols - 1
    print(features)
    X = train.values[:, 0:features]
    Y = train.values[:, features]
    print(Y)
X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size = 0.2,
random_state = 0)
 text.insert(END,"Dataset Length : "+str(len(X))+"\n");
 text.insert(END,"Splitted Training Length : "+str(len(X_train))+"\n");
  text.insert(END,"Splitted Test Length : "+str(len(X_test))+"\n\n");
def prediction(X_test, cls):  #prediction done here
y_pred = cls.predict(X_test)
```

```
for i in range(len(X_test)):
print("X=%s, Predicted=%s" % (X_test[i], y_pred[i]))
return y_pred

# Function to calculate accuracy
def cal_accuracy(y_test, y_pred, details):
cm = confusion_matrix(y_test, y_pred)
accuracy = accuracy_score(y_test,y_pred)*100
text.insert(END,details+"\n\n")
text.insert(END,"Accuracy : "+str(accuracy)+"\n\n")
text.insert(END,"Report : "+str(classification_report(y_test,
y_pred))+"\n")
text.insert(END,"Confusion Matrix : "+str(cm)+"\n\n\n\n\n")
return accuracy

def runSVM():
   global svm_acc
   global svm_time
   start_time = time.time()
   text.delete('1.0', END)
   cls = svm.SVC(C=2.0,gamma='scale',kernel = 'rbf', random_state = 2)
   cls.fit(X_train, y_train)
   prediction_data = prediction(X_test, cls)
   svm_acc = cal_accuracy(y_test, prediction_data,'SVM Accuracy')
   svm_time = (time.time() - start_time)

def runSVMGenetic():
   text.delete('1.0', END)
```

```
    global svmga_acc
    global svmga_classifier
    global svmga_time
       estimator = svm.SVC(C=2.0,gamma='scale',kernel = 'rbf', random_state = 2)
      svmga_classifier = GeneticSelectionCV(estimator,
cv=5,
verbose=1,
scoring="accuracy",
max_features=5,
n_population=50,
crossover_proba=0.5,
mutation_proba=0.2,
n_generations=40,
crossover_independent_proba=0.5,
mutation_independent_proba=0.05,
tournament_size=3,
n_gen_no_change=10,
caching=True,
n_jobs=-1)
    start_time = time.time()
    svmga_classifier = svmga_classifier.fit(X_train, y_train)
    svmga_time = svm_time/2
    prediction_data = prediction(X_test, svmga_classifier)
    svmga_acc = cal_accuracy(y_test, prediction_data,'SVM with GA Algorithm Accuracy, Classification Report & Confusion Matrix')
def runNN():
    global nn_acc
    global nn_time
    text.delete('1.0', END)
    start_time = time.time()
    model = Sequential()
    model.add(Dense(4, input_dim=215, activation='relu'))
    model.add(Dense(215, activation='relu'))
    model.add(Dense(1, activation='sigmoid'))
    model.compile(loss='binary_crossentropy', optimizer='adam',
metrics=['accuracy'])
    model.fit(X_train, y_train, epochs=50, batch_size=64)
    _, ann_acc = model.evaluate(X_test, y_test)
    nn_acc = ann_acc*100
    text.insert(END,"ANN Accuracy : "+str(nn_acc)+"\n\n")
    nn_time = (time.time() - start_time)
def runNNGenetic():
    global annga_acc
    global nnga_time
    text.delete('1.0', END)
    train = pd.read_csv(filename)
    rows = train.shape[0]  # gives number of row count
    cols = train.shape[1]  # gives number of col count
    features = cols - 1
    print(features)
    X = train.values[:, 0:100]
    Y = train.values[:, features]
    print(Y)
    X_train1, X_test1, y_train1, y_test1 = train_test_split(X, Y, test_size =
0.2, random_state = 0)
```

```
    model = Sequential()
    model.add(Dense(4, input_dim=100, activation='relu'))
    model.add(Dense(100, activation='relu'))
    model.add(Dense(1, activation='sigmoid'))
    model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])
    start_time = time.time()
    model.fit(X_train1, y_train1)
    nnga_time = (time.time() - start_time)
    _, ann_acc = model.evaluate(X_test1, y_test1)
    annga_acc = ann_acc*100
    text.insert(END,"ANN with Genetic Algorithm Accuracy : "+str(annga_acc)+"\n\n")
def graph():
    height = [svm_acc, nn_acc, svmga_acc, annga_acc]
    bars = ('SVM Accuracy','NN Accuracy','SVM Genetic Acc','NN Genetic Acc')
    y_pos = np.arange(len(bars))
    plt.bar(y_pos, height)
    plt.xticks(y_pos, bars)
    plt.show()
def timeGraph():
    height = [svm_time,svmga_time,nn_time,nnga_time]
    bars = ('SVM Time','SVM Genetic Time','NN Time','NN Genetic Time')
    y_pos = np.arange(len(bars))
    plt.bar(y_pos, height)
    plt.xticks(y_pos, bars)
    plt.show()
font = ('times', 16, 'bold')
```

```
title = Label(main, text='Android Malware Detection Using Genetic
Algorithm based Optimized Feature Selection and Machine Learning')

#title.config(bg='brown', fg='white')

title.config(font=font)

title.config(height=3, width=120)

title.place(x=0,y=5)

font1 = ('times', 14, 'bold')

uploadButton = Button(main, text="Upload Android Malware Dataset",
command=upload)

uploadButton.place(x=50,y=100)

uploadButton.config(font=font1)

pathlabel = Label(main)

pathlabel.config(bg='brown', fg='white')

pathlabel.config(font=font1)

pathlabel.place(x=460,y=100)

generateButton = Button(main, text="Generate Train & Test Model",
command=generateModel)

generateButton.place(x=50,y=150)

generateButton.config(font=font1)

svmButton = Button(main, text="Run SVM Algorithm",
command=runSVM)

svmButton.place(x=330,y=150)

svmButton.config(font=font1)

svmgaButton = Button(main, text="Run SVM with Genetic Algorithm",
command=runSVMGenetic)

svmgaButton.place(x=540,y=150)

svmgaButton.config(font=font1)

nnButton = Button(main, text="Run Neural Network Algorithm",
command=runNN)

nnButton.place(x=870,y=150)
nnButton.config(font=font1)

nngaButton = Button(main, text="Run Neural Network with Genetic
Algorithm", command=runNNGenetic)

nngaButton.place(x=50,y=200)

nngaButton.config(font=font1)

graphButton = Button(main, text="Accuracy Graph", command=graph)

graphButton.place(x=460,y=200)

graphButton.config(font=font1)

exitButton = Button(main, text="Execution Time Graph",
command=timeGraph)

exitButton.place(x=650,y=200)

exitButton.config(font=font1)



font1 = ('times', 12, 'bold')

text=Text(main,height=20,width=150)

scroll=Scrollbar(text)

text.configure(yscrollcommand=scroll.set)

text.place(x=10,y=250)

text.config(font=font1)

#main.config()

main.mainloop()
```

7.2 Testing

Software Testing:

Testing is a process of executing a program with the aim of finding error. To make our software perform well it should be error free. If testing is done successfully, it will remove all the errors from the software.

Types of Testing

1. White Box Testing
2. Black Box Testing
3. Unit testing

7.2.1 White Box Testing:

Testing technique based on knowledge of the internal logic of an application's code and includes tests like coverage of code statements, branches, paths, conditions. It is performed by software developers.

7.2.2 Black Box Testing:

A method of software testing that verifies the functionality of an application without having specific knowledge of the application's code/internal structure. Tests are based on requirements and functionality.

7.2.3 Unit Testing:

Software verification and validation method in which a programmer tests if individual units of source code are fit for use. It is usually conducted by the development team.

Blackbox testing is testing the functionality of an application without knowing the details of its implementation including internal program structure, data structures etc. Test cases for black box testing are created based on the requirement specifications. Therefore, it is also called as specification-based testing. Fig.4.1 represents the black box testing:

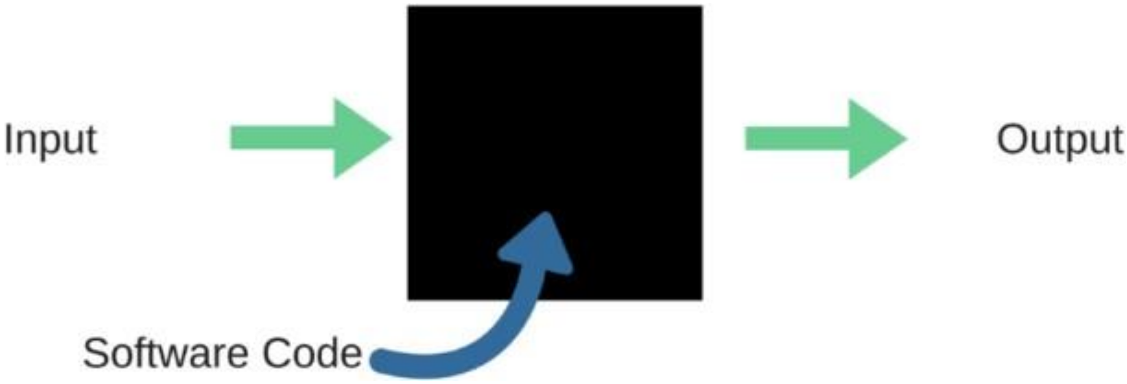


Fig.7.2.2.1: Black Box Testing

When applied to machine learning models, black box testing would mean testing machine learning models without knowing the internal details such as features of the machine learning model, the algorithm used to create the model etc. The challenge, however, is to verify the test outcome against the expected values that are known beforehand.

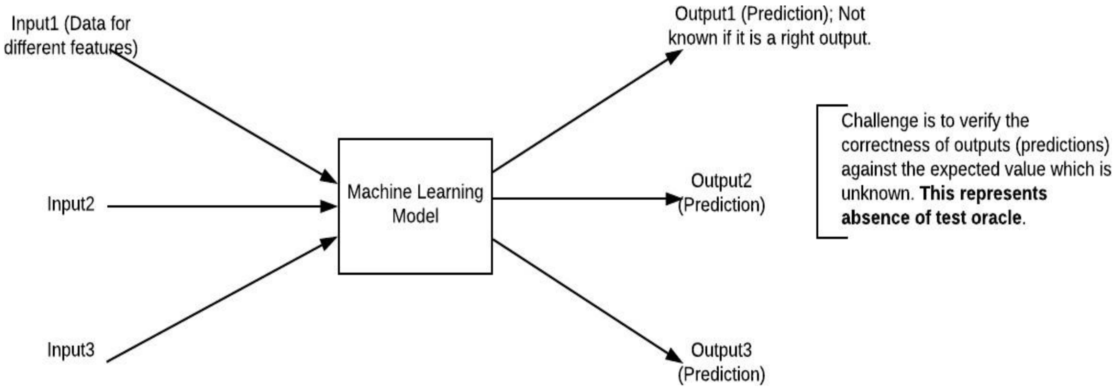

Input1 (Data for different features)
Input2
Input3
Machine Learning Model
Output1 (Prediction); Not known if it is a right output.
Output2 (Prediction)
Output3 (Prediction)
Challenge is to verify the correctness of outputs (predictions) against the expected value which is unknown. **This represents absence of test oracle**.

Fig.7.2.2: Black Box Testing for Machine Learning algorithms

The above Figure represents the black box testing procedure for machine learning algorithms.

Input	**Actual Output**	**Predicted Output**
[16,6,324,0,0,0,22,0,0,0,0,0,0]	0	0
[16,7,263,7,0,2,700,9,10,1153,832,9,2]	1	1

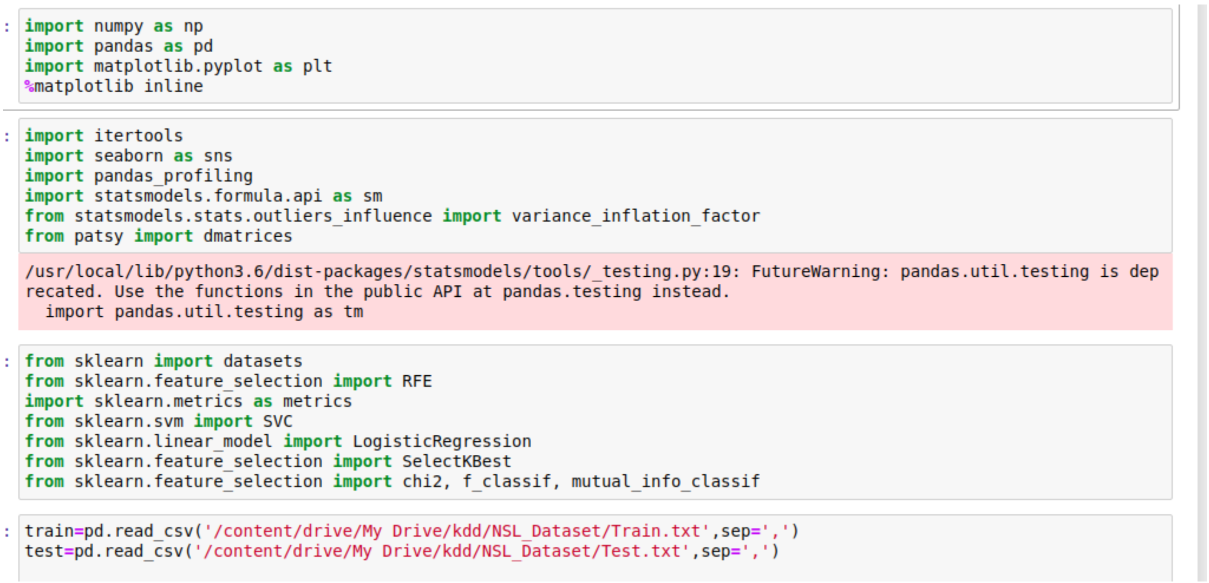
Table.7.2.1: Black box Testing

The model gives out the correct output when different inputs are given which are mentioned in Table Therefore, the program is said to be executed as expected or correct program

Test Case Id	Test Case Name	Test Case Description	Test Steps			Test Case Status	Test Priority
			Step	Expected	Actual		
01	Start the Application	Host the Application and test if it starts making sure the required software is available	If it doesn't start	We cannot Run the application	The Application Hosts success	High	High
02	Home Page	Check the Deployment For properly loading the Environment application	If it doesn't load	We cannot Access the application	The Application is running successfully	High	High
03	User Mode	Verify the working of the application in freestyle mode.	If it doesn't respond	We cannot use the freestyle mode.	The Application displays the Freestyle page	High	High
04	Data Input	Verify if the Application takes input and updates	If it fails to take the input or store in the database	We cannot Proceed further	The Application updates the input to application	High	High

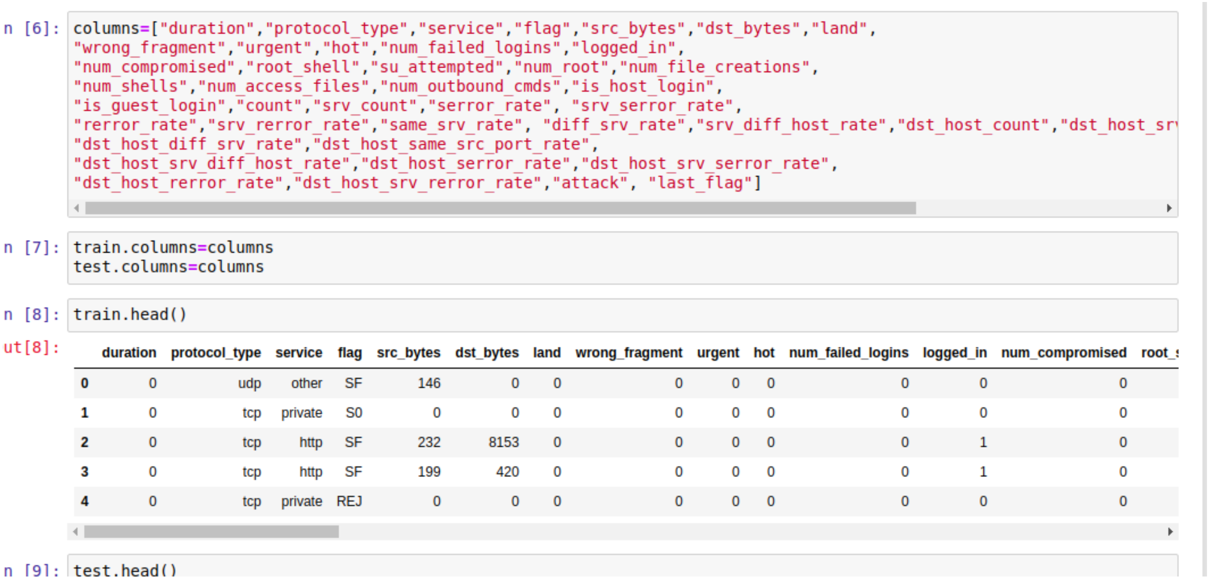
Table.7.2.2 different test cases during execution

8. RESULTS AND DISCUSSIONS



```
: import numpy as np
  import pandas as pd
  import matplotlib.pyplot as plt
  %matplotlib inline

: import itertools
  import seaborn as sns
  import pandas_profiling
  import statsmodels.formula.api as sm
  from statsmodels.stats.outliers_influence import variance_inflation_factor
  from patsy import dmatrices

  /usr/local/lib/python3.6/dist-packages/statsmodels/tools/_testing.py:19: FutureWarning: pandas.util.testing is dep
  recated. Use the functions in the public API at pandas.testing instead.
    import pandas.util.testing as tm

: from sklearn import datasets
  from sklearn.feature_selection import RFE
  import sklearn.metrics as metrics
  from sklearn.svm import SVC
  from sklearn.linear_model import LogisticRegression
  from sklearn.feature_selection import SelectKBest
  from sklearn.feature_selection import chi2, f_classif, mutual_info_classif

: train=pd.read_csv('/content/drive/My Drive/kdd/NSL_Dataset/Train.txt',sep=',')
  test=pd.read_csv('/content/drive/My Drive/kdd/NSL_Dataset/Test.txt',sep=',')
```

8.1 Data pre-processing



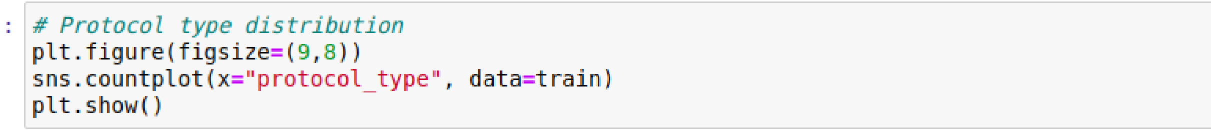
```
n [6]: columns=["duration","protocol_type","service","flag","src_bytes","dst_bytes","land",
       "wrong_fragment","urgent","hot","num_failed_logins","logged_in",
       "num_compromised","root_shell","su_attempted","num_root","num_file_creations",
       "num_shells","num_access_files","num_outbound_cmds","is_host_login",
       "is_guest_login","count","srv_count","serror_rate", "srv_serror_rate",
       "rerror_rate","srv_rerror_rate","same_srv_rate", "diff_srv_rate","srv_diff_host_rate","dst_host_count","dst_host_sr
       "dst_host_diff_srv_rate","dst_host_same_src_port_rate",
       "dst_host_srv_diff_host_rate","dst_host_serror_rate","dst_host_srv_serror_rate",
       "dst_host_rerror_rate","dst_host_srv_rerror_rate","attack", "last_flag"]

n [7]: train.columns=columns
       test.columns=columns

n [8]: train.head()
```

ut[8]:

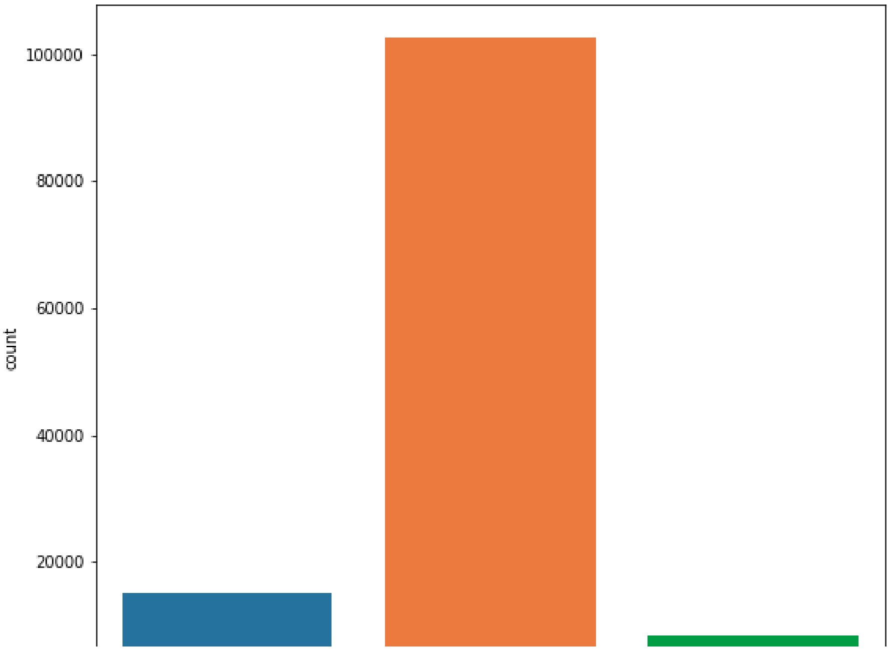
	duration	protocol_type	service	flag	src_bytes	dst_bytes	land	wrong_fragment	urgent	hot	num_failed_logins	logged_in	num_compromised	root_
0	0	udp	other	SF	146	0	0	0	0	0	0	0	0	
1	0	tcp	private	S0	0	0	0	0	0	0	0	0	0	
2	0	tcp	http	SF	232	8153	0	0	0	0	0	1	0	
3	0	tcp	http	SF	199	420	0	0	0	0	0	1	0	
4	0	tcp	private	REJ	0	0	0	0	0	0	0	0	0	

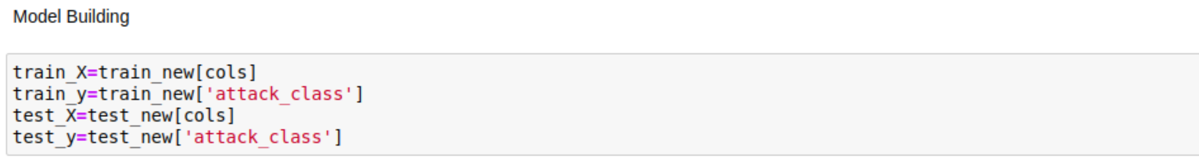
```
n [9]: test.head()
```

8.2 Data EDA:



```
: # Protocol type distribution
plt.figure(figsize=(9,8))
sns.countplot(x="protocol_type", data=train)
plt.show()
```

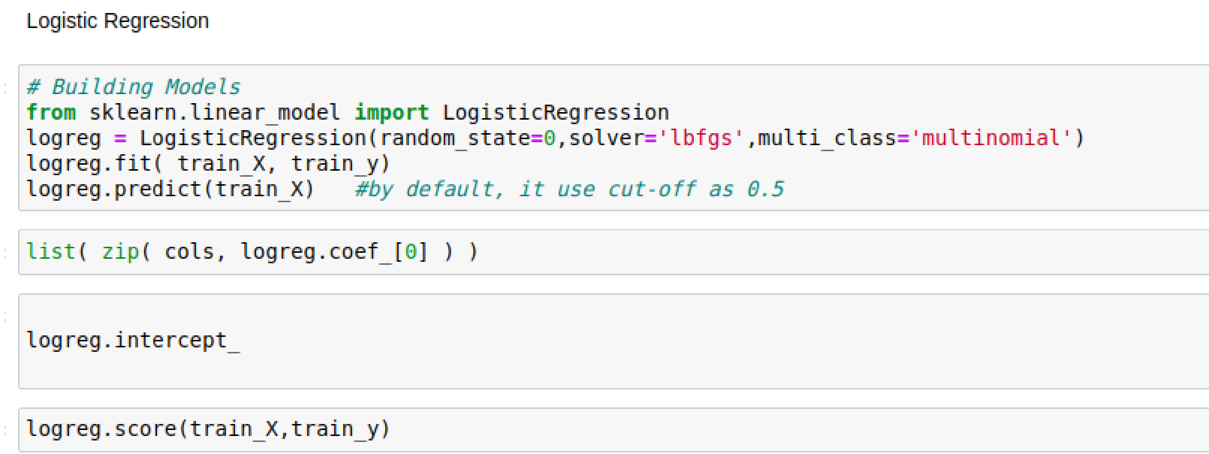




Model Building

```
train_X=train_new[cols]
train_y=train_new['attack_class']
test_X=test_new[cols]
test_y=test_new['attack_class']
```

8.3 ML Deploy



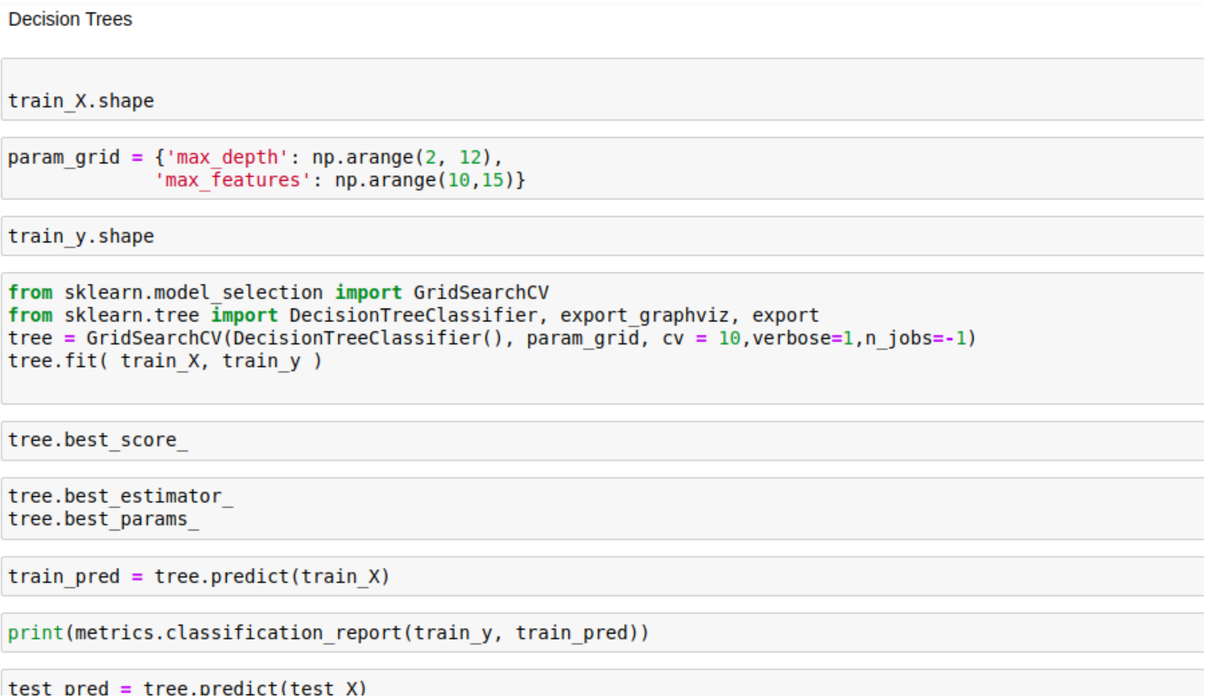
Logistic Regression

```
# Building Models
from sklearn.linear_model import LogisticRegression
logreg = LogisticRegression(random_state=0,solver='lbfgs',multi_class='multinomial')
logreg.fit( train_X, train_y)
logreg.predict(train_X)    #by default, it use cut-off as 0.5
```

```
list( zip( cols, logreg.coef_[0] ) )
```

```
logreg.intercept_
```

```
logreg.score(train_X,train_y)
```



Decision Trees

```
train_X.shape
```

```
param_grid = {'max_depth': np.arange(2, 12),
             'max_features': np.arange(10,15)}
```

```
train_y.shape
```

```
from sklearn.model_selection import GridSearchCV
from sklearn.tree import DecisionTreeClassifier, export_graphviz, export
tree = GridSearchCV(DecisionTreeClassifier(), param_grid, cv = 10,verbose=1,n_jobs=-1)
tree.fit( train_X, train_y )
```

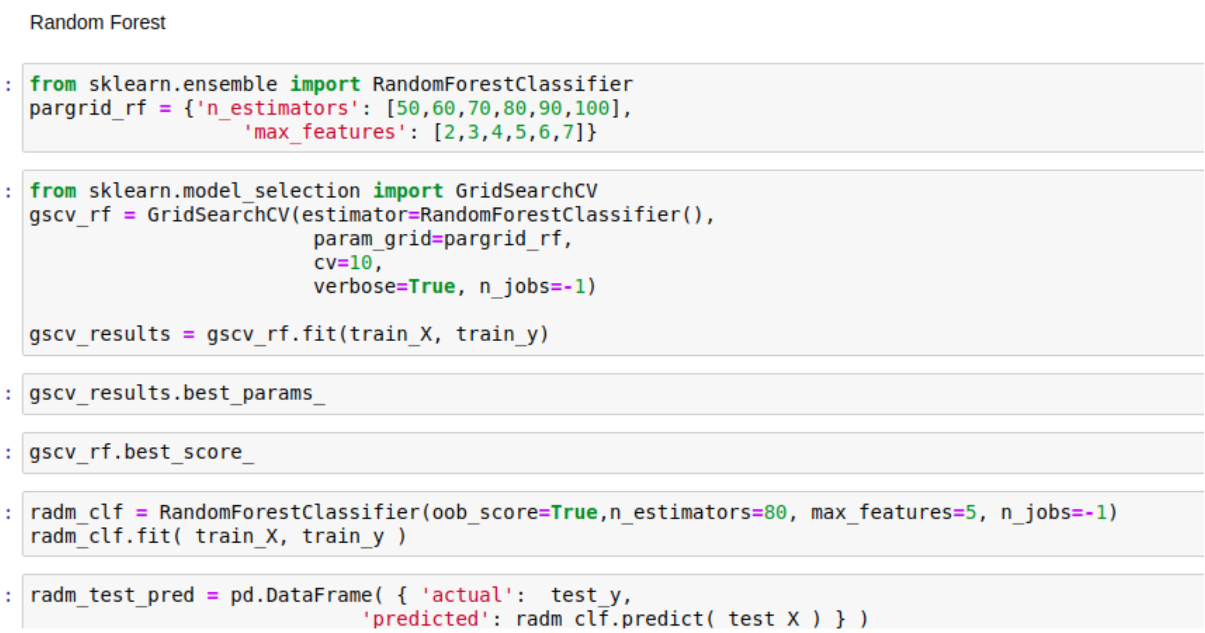
```
tree.best_score_
```

```
tree.best_estimator_
tree.best_params_
```

```
train_pred = tree.predict(train_X)
```

```
print(metrics.classification_report(train_y, train_pred))
```

```
test_pred = tree.predict(test_X)
```



Random Forest

```
: from sklearn.ensemble import RandomForestClassifier
  pargrid_rf = {'n_estimators': [50,60,70,80,90,100],
                  'max_features': [2,3,4,5,6,7]}
```

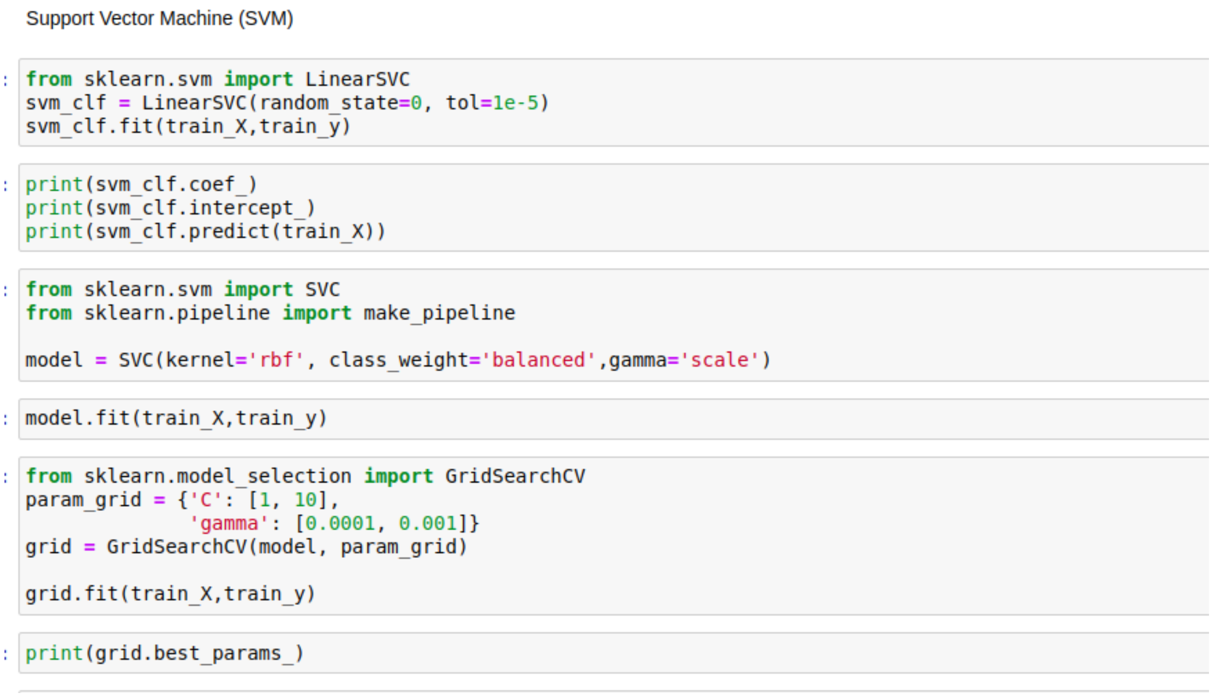
```
: from sklearn.model_selection import GridSearchCV
  gscv_rf = GridSearchCV(estimator=RandomForestClassifier(),
                          param_grid=pargrid_rf,
                          cv=10,
                          verbose=True, n_jobs=-1)

  gscv_results = gscv_rf.fit(train_X, train_y)
```

```
: gscv_results.best_params_
```

```
: gscv_rf.best_score_
```

```
: radm_clf = RandomForestClassifier(oob_score=True,n_estimators=80, max_features=5, n_jobs=-1)
  radm_clf.fit( train_X, train_y )
```

```
: radm_test_pred = pd.DataFrame( { 'actual':  test_y,
                              'predicted': radm_clf.predict( test_X ) } )
```



Support Vector Machine (SVM)

```
from sklearn.svm import LinearSVC
svm_clf = LinearSVC(random_state=0, tol=1e-5)
svm_clf.fit(train_X,train_y)
```

```
print(svm_clf.coef_)
print(svm_clf.intercept_)
print(svm_clf.predict(train_X))
```

```
from sklearn.svm import SVC
from sklearn.pipeline import make_pipeline

model = SVC(kernel='rbf', class_weight='balanced',gamma='scale')
```

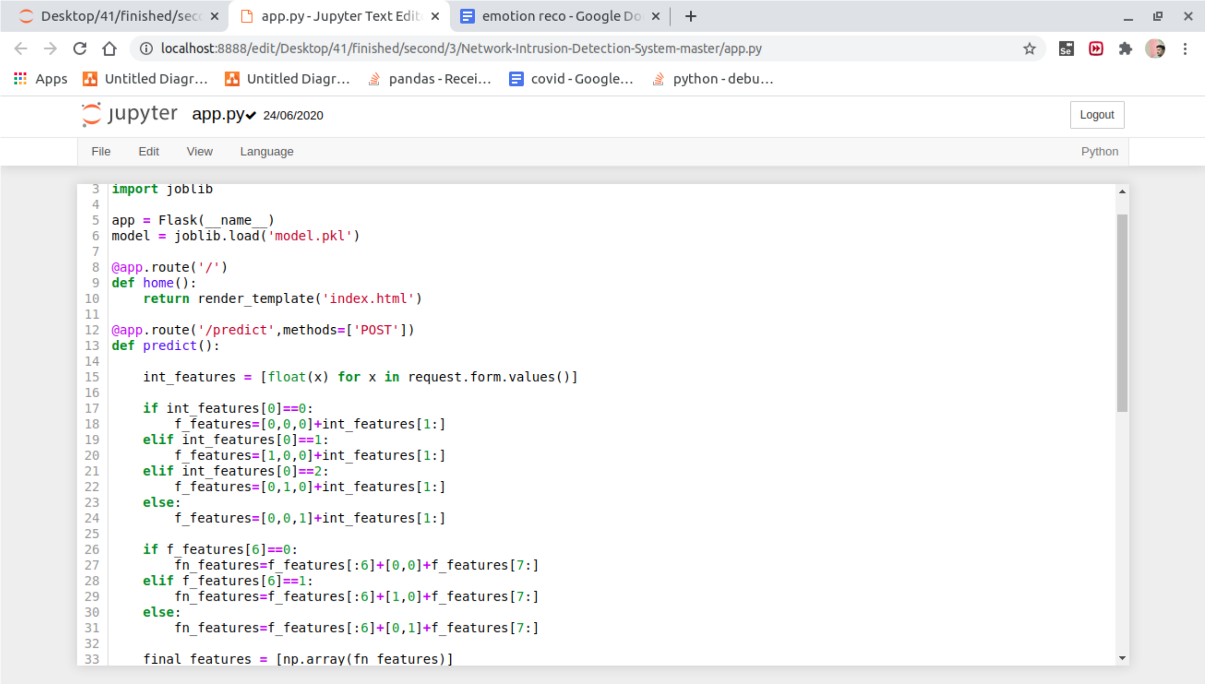
```
model.fit(train_X,train_y)
```

```
from sklearn.model_selection import GridSearchCV
param_grid = {'C': [1, 10],
              'gamma': [0.0001, 0.001]}
grid = GridSearchCV(model, param_grid)

grid.fit(train_X,train_y)
```

```
print(grid.best_params_)
```

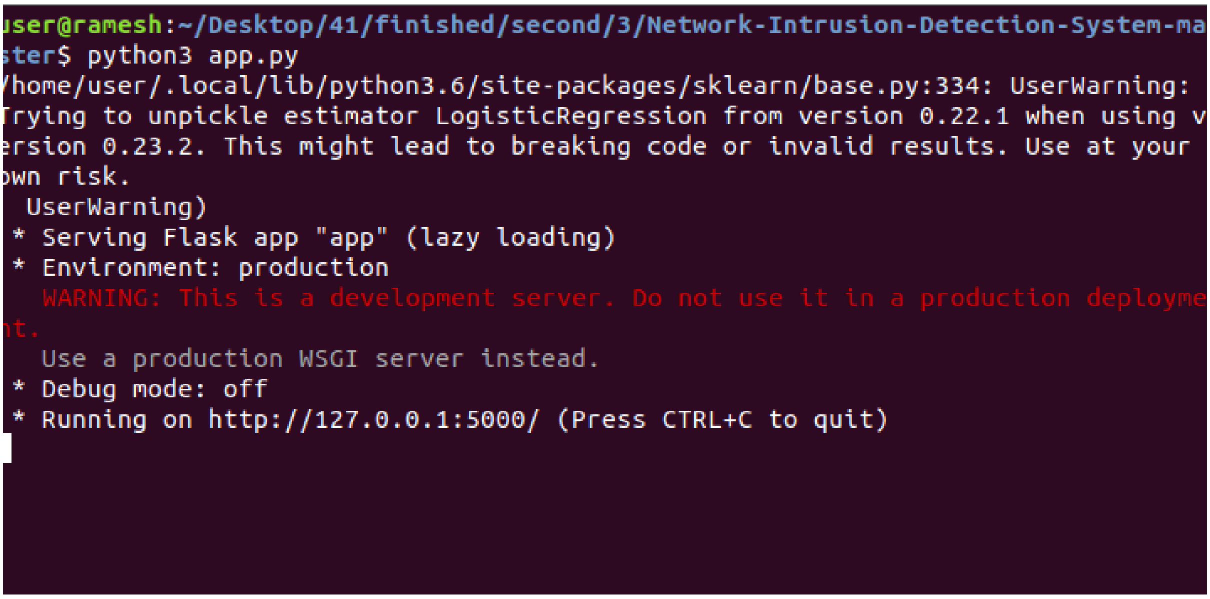
From the score accuracy we concluding the DT & RF give better accuracy and building pickle file for predicting the user input

8.4 Application

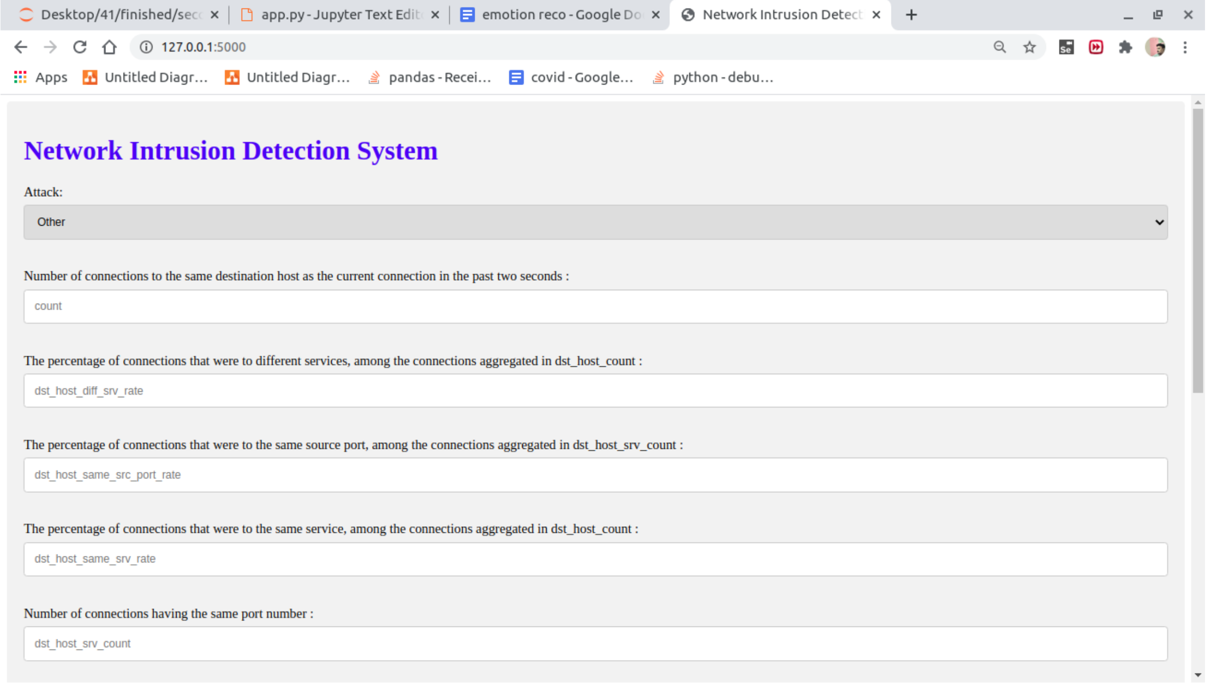


```
import joblib

app = Flask(__name__)
model = joblib.load('model.pkl')

@app.route('/')
def home():
    return render_template('index.html')

@app.route('/predict',methods=['POST'])
def predict():

    int_features = [float(x) for x in request.form.values()]

    if int_features[0]==0:
        f_features=[0,0,0]+int_features[1:]
    elif int_features[0]==1:
        f_features=[1,0,0]+int_features[1:]
    elif int_features[0]==2:
        f_features=[0,1,0]+int_features[1:]
    else:
        f_features=[0,0,1]+int_features[1:]

    if f_features[6]==0:
        fn_features=f_features[:6]+[0,0]+f_features[7:]
    elif f_features[6]==1:
        fn_features=f_features[:6]+[1,0]+f_features[7:]
    else:
        fn_features=f_features[:6]+[0,1]+f_features[7:]

    final_features = [np.array(fn_features)]
```

Localhost - in cmd python app.py



```
user@ramesh:~/Desktop/41/finished/second/3/Network-Intrusion-Detection-System-ma
ster$ python3 app.py
/home/user/.local/lib/python3.6/site-packages/sklearn/base.py:334: UserWarning:
Trying to unpickle estimator LogisticRegression from version 0.22.1 when using v
ersion 0.23.2. This might lead to breaking code or invalid results. Use at your
own risk.
  UserWarning)
 * Serving Flask app "app" (lazy loading)
 * Environment: production
   WARNING: This is a development server. Do not use it in a production deployme
nt.
   Use a production WSGI server instead.
 * Debug mode: off
 * Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
```



Network Intrusion Detection System

Attack:

Other

Number of connections to the same destination host as the current connection in the past two seconds :

count

The percentage of connections that were to different services, among the connections aggregated in dst_host_count :

dst_host_diff_srv_rate

The percentage of connections that were to the same source port, among the connections aggregated in dst_host_srv_count :

dst_host_same_src_port_rate

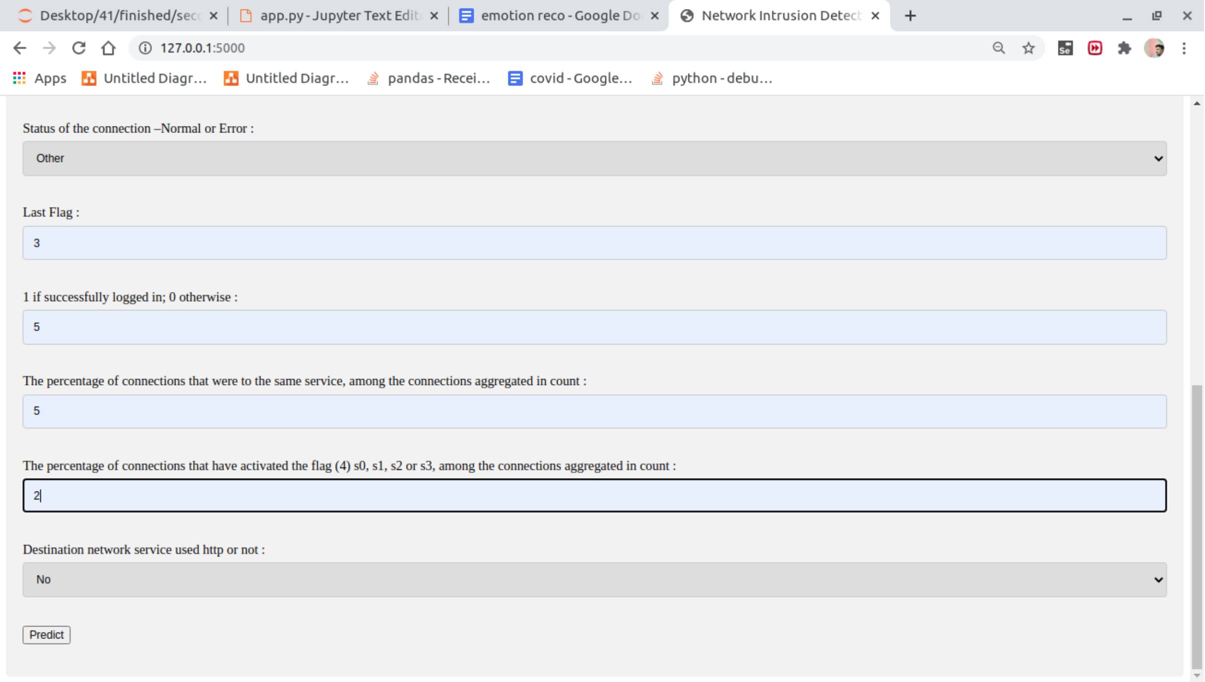
The percentage of connections that were to the same service, among the connections aggregated in dst_host_count :

dst_host_same_srv_rate

Number of connections having the same port number :

dst_host_srv_count

Enter the input



Status of the connection –Normal or Error :

Other

Last Flag :

3

1 if successfully logged in; 0 otherwise :

5

The percentage of connections that were to the same service, among the connections aggregated in count :

5

The percentage of connections that have activated the flag (4) s0, s1, s2 or s3, among the connections aggregated in count :

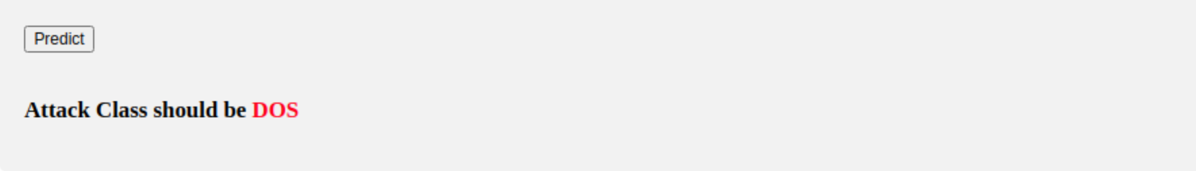
2

Destination network service used http or not :

No

Predict

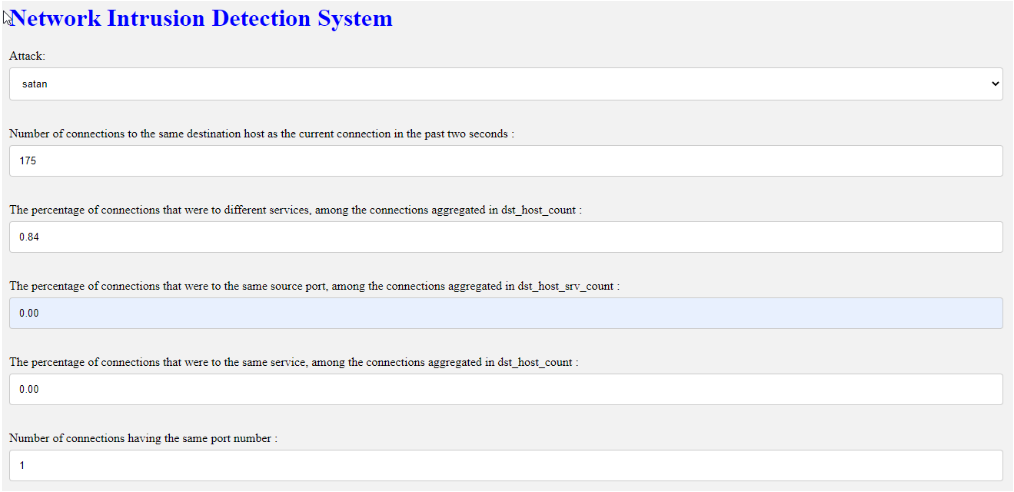
Predict attack -



Predict

Attack Class should be DOS

Case 2:



Network Intrusion Detection System

Attack:

satan

Number of connections to the same destination host as the current connection in the past two seconds :

175

The percentage of connections that were to different services, among the connections aggregated in dst_host_count :

0.84

The percentage of connections that were to the same source port, among the connections aggregated in dst_host_srv_count :

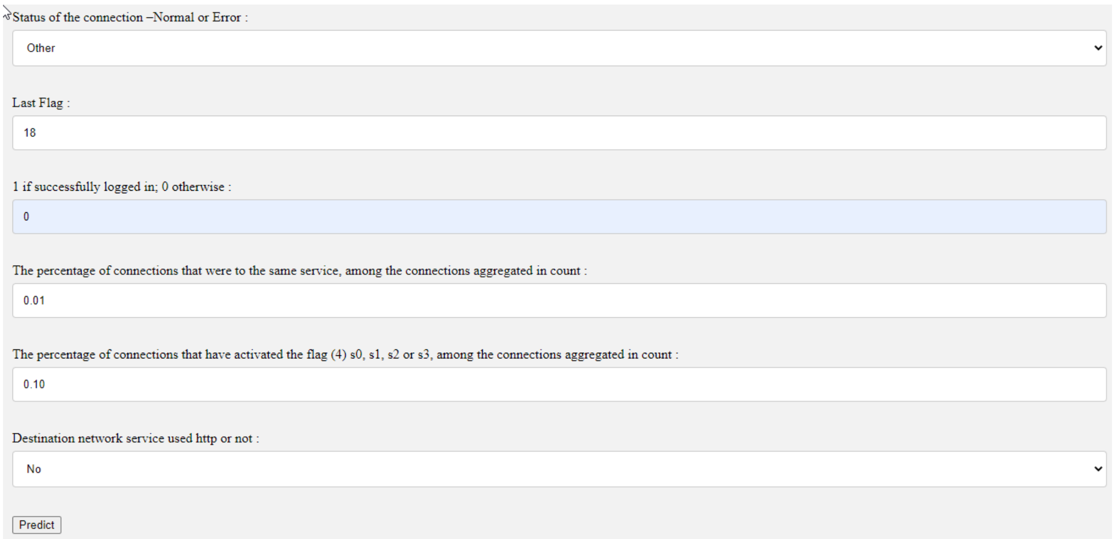
0.00

The percentage of connections that were to the same service, among the connections aggregated in dst_host_count :

0.00

Number of connections having the same port number :

1



Status of the connection –Normal or Error :

Other

Last Flag :

18

1 if successfully logged in; 0 otherwise :

0

The percentage of connections that were to the same service, among the connections aggregated in count :

0.01

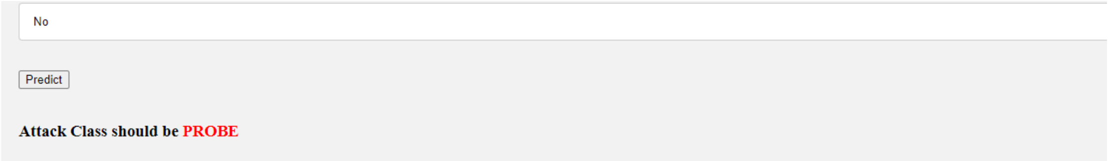
The percentage of connections that have activated the flag (4) s0, s1, s2 or s3, among the connections aggregated in count :

0.10

Destination network service used http or not :

No

Predict attack



No

Attack Class should be PROBE

9. CONCLUSION

Right now, estimations of SVM, Dession Tree, Random Forest and significant learning estimations subject to modern CICIDS2017 dataset were presented generally. Results show that the significant learning estimation performed on a very basic level ideal results over SVM, Dession Tree and RF. We will use port breadth tries as well as other attack types with AI and significant learning computations, Apache Hadoop and sparkle innovations together ward on this dataset later on. Every one of these computation assists us with identifying the digital assault in network. It occurs in the manner that when we think about lengthy back a long time there might be such countless assaults occurred so when these assaults are perceived then the highlights at which esteems these assaults are going on will be put away in some datasets. In this way, by utilizing these datasets we will foresee regardless of whether digital assault is finished. These expectations should be possible by four calculations like SVM, Dession Tree, RF this paper assists with distinguishing which calculation predicts the best precision rates which assists with anticipating best outcomes to recognize the digital assaults occurred or not.

10. REFERENCES

[1] K. Graves, Ceh: Official certified ethical hacker review guide: Exam 312-50. John Wiley & Sons, 2007.

[2] R. Christopher, “Port scanning techniques and the defense against them,” SANS Institute, 2001.

[3] M. Baykara, R. Das¸, and I. Karado ˘gan, “Bilgi g ¨uvenli ˘gi sistemlerinde kullanilan arac¸larin incelenmesi,” in 1st International Symposium on Digital Forensics and Security (ISDFS13), 2013, pp. 231–239.

[4] S. Staniford, J. A. Hoagland, and J. M. McAlerney, “Practical automated detection of stealthy portscans,” Journal of Computer Security, vol. 10, no. 1-2, pp. 105–136, 2002.

[5] S. Robertson, E. V. Siegel, M. Miller, and S. J. Stolfo, “Surveillance detection in high bandwidth environments,” in DARPA Information Survivability Conference and Exposition, 2003. Proceedings, vol. 1. IEEE, 2003, pp. 130–138.

[6] K. Ibrahimi and M. Ouaddane, “Management of intrusion detection systems based-kdd99: Analysis with lda and pca,” in Wireless Networks and Mobile Communications (WINCOM), 2017 International Conference on. IEEE, 2017, pp. 1–6.

[7] N. Moustafa and J. Slay, “The significant features of the unsw-nb15 and the kdd99 data sets for network intrusion detection systems,” in Building Analysis Datasets and Gathering Experience Returns for Security (BADGERS), 2015 4th International Workshop on. IEEE, 2015, pp. 25–31.

[8] L. Sun, T. Anthony, H. Z. Xia, J. Chen, X. Huang, and Y. Zhang, “Detection and classification of malicious patterns in network traffic using benford’s law,” in Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC), 2017. IEEE, 2017, pp. 864–872.

[9] S. M. Almansob and S. S. Lomte, “Addressing challenges for intrusion detection system using naive bayes and pca algorithm,” in Convergence in Technology (I2CT), 2017 2nd International Conference for. IEEE, 2017, pp. 565–568.

[10] M. C. Raja and M. M. A. Rabbani, “Combined analysis of support vector machine and principle component analysis for ids,” in IEEE International Conference on Communication and Electronics Systems, 2016, pp. 1–5.

[11] S. Aljawarneh, M. Aldwairi, and M. B. Yassein, “Anomaly-based intrusion detection system through feature selection analysis and building hybrid efficient model,” Journal of Computational Science, vol. 25, pp. 152–160, 2018.

[12] I. Sharafaldin, A. H. Lashkari, and A. A. Ghorbani, "Toward generating a new intrusion detection dataset and intrusion traffic characterization." in ICISSP, 2018, pp. 108–116.

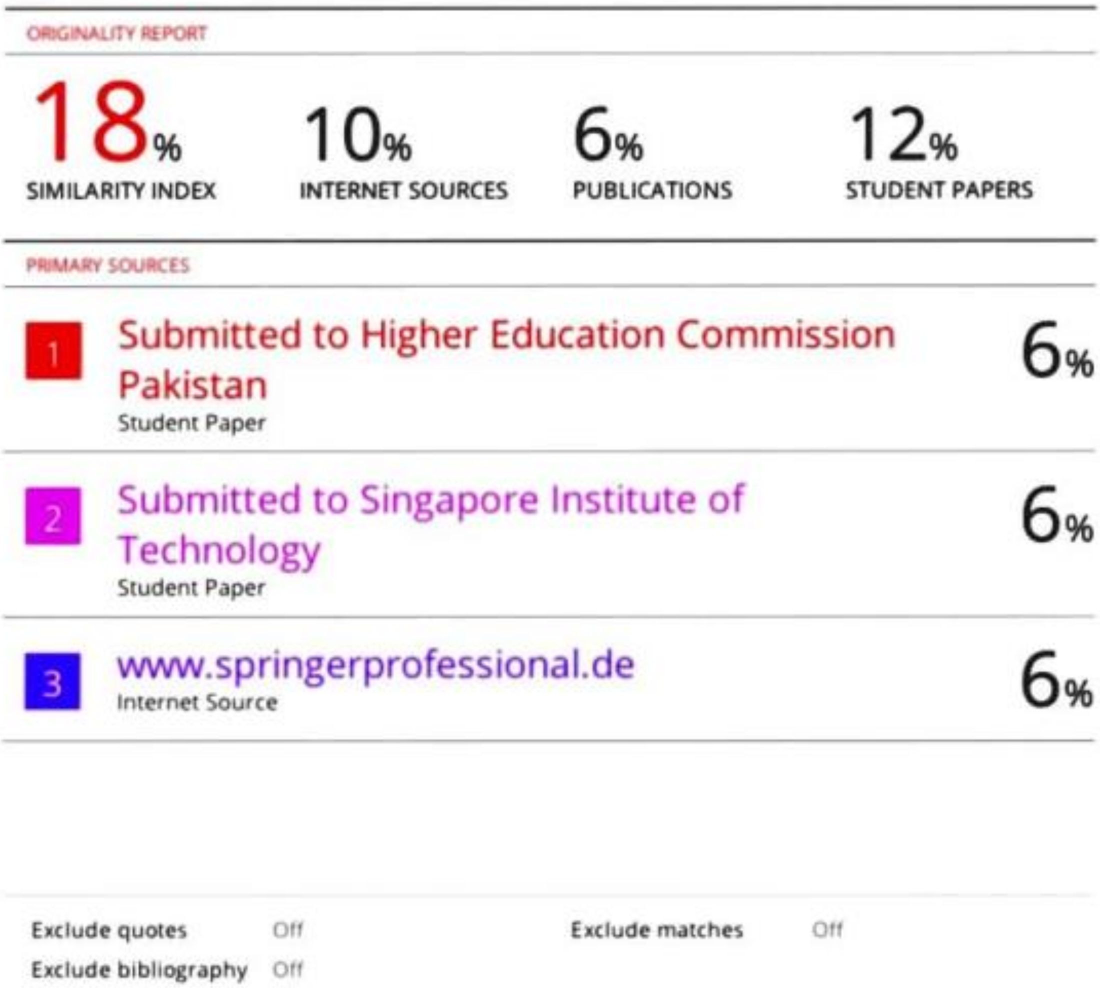
[13] D. Aksu, S. Ustebay, M. A. Aydin, and T. Atmaca, "Intrusion detection with comparative analysis of supervised learning techniques and fisher score feature selection algorithm," in International Symposium on Computer and Information Sciences. Springer, 2018, pp. 141–149.

[14] N. Marir, H. Wang, G. Feng, B. Li, and M. Jia, "Distributed abnormal behavior detection approach based on deep belief network and ensemble svm using spark," IEEE Access, 2018.

[15] P. A. A. Resende and A. C. Drummond, "Adaptive anomaly-based intrusion detection system using genetic algorithm and profiling," Security and Privacy, vol. 1, no. 4, p. e36, 2018.

[16] C. Cortes and V. Vapnik, "Support-vector networks," Machine learning, vol. 20, no. 3, pp. 273–297, 1995.

[17] R. Shouval, O. Bondi, H. Mishan, A. Shimoni, R. Unger, and A. Nagler, "Application of machine learning algorithms for clinical predictive modeling: a data-mining approach in sct," Bone marrow transplantation, vol. 49, no. 3, p. 332, 2014**.**



ORIGINALITY REPORT

18%	10%	6%	12%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	Submitted to Higher Education Commission Pakistan Student Paper	6%
2	Submitted to Singapore Institute of Technology Student Paper	6%
3	www.springerprofessional.de Internet Source	6%

Exclude quotes Off

Exclude bibliography Off

Exclude matches Off