##### ABSTRACT

Cyber-crime is proliferating everywhere exploiting every kind of vulnerability to the computing environment. Ethical Hackers pay more attention towards assessing vulnerabilities and recommending mitigation methodologies. The development of effective techniques has been an urgent demand in the field of the cyber security community. Most techniques used in today’s IDS are not able to deal with the dynamic and complex nature of cyber-attacks on computer networks. Machine learning for cyber security has become an issue of great importance recently due to the effectiveness of machine learning in cyber security issues. Machine learning techniques have been applied for major challenges in cyber security issues like intrusion detection, malware classification and detection, spam detection and phishing detection.

Although machine learning cannot automate a complete cyber security system, it helps to identify cyber security threats more efficiently than other software-oriented methodologies, and thus reduces the burden on security analysts. Hence, efficient adaptive methods like various techniques of machine learning can result in higher detection rates, lower false alarm rates and reasonable computation and communication costs. Our main goal is that the task of finding attacks is fundamentally different from these other applications, making it significantly harder for the intrusion detection community to employ machine learning effectively.

Keywords: Cyber-crime, Machine learning, Cyber-security, Intrusion detection system.

|  |  |  |
| --- | --- | --- |
| **FIGURE NO** | **FIGURE NAME** | **PAGE NO** |
| Figure 3.1 | Project Architecture for Detection Of Cyber Attack In Network Using Machine Learning Techniques | 8 |
| Figure 3.2 | Use Case Diagram for Detection Of Cyber Attack In Network Using Machine Learning Techniques | 10 |
| Figure 3.3 | Class Diagram for Detection Of Cyber Attack In Network Using Machine Learning Techniques | 11 |
| Figure 3.4 | Sequence Diagram for Detection Of Cyber Attack In Network Using Machine Learning Techniques | 12 |
| Figure 3.5 | Activity Diagram for Detection Of Cyber Attack In Network Using Machine Learning Techniques | 13 |

|  |  |  |
| --- | --- | --- |
| **SCREENSHOT NO.** | **SCREENSHOT NAME** | **PAGE NO**. |
| Screenshot 5.1 | Input 1 examples for detecting cyber attack | 16 |
| Screenshot 5.2 | Input 1 examples for detecting cyber attack | 16 |
| Screenshot 5.3 | Results of the given examples input 1 for detecting cyber attack | 17 |
| Screenshot 5.4 | Input 2examples for detecting cyber attack | 17 |
| Screenshot 5.5    Screenshot 5.6 | Input 2 examples for detecting cyber attack | 18 |
|  |  |  |
|  |  |  |
|  |  |  |

###### ABSTRACT i

LIST OF FIGURES ii

LIST OF SCREENSHOTS iii

1. [INTRODUCTION 1](#_TOC_250030)

1.1 [PROJECT SCOPE 1](#_TOC_250029)

1.2 [PROJECT PURPOSE 1](#_TOC_250028)

1.3 [PROJECT FEATURES 1](#_TOC_250027)

1. [SYSTEM ANALYSIS 2](#_TOC_250026)

2.1 [PROBLEM DEFINITION 2](#_TOC_250025)

2.2 [EXISTING SYSTEM 3](#_TOC_250024)

2.2.1 LIMITATIONS OF THE EXISTING SYSTEM 4

2.3 [PROPOSED SYSTEM 4](#_TOC_250023)

2.3.1 ADVANTAGES OF PROPOSED SYSTEM 5

2.4 [FEASIBILITY STUDY 5](#_TOC_250022)

2.4.1 [ECONOMIC FEASIBILITY 5](#_TOC_250021)

2.5.2 [TECHNICAL FEASIBILITY 6](#_TOC_250020)

2.4.3 SOCIAL FEASIBILITY 6

2.5 [HARDWARE & SOFTWARE REQUIREMENTS 7](#_TOC_250019)

2.5.1 [HARDWARE REQUIREMENTS 7](#_TOC_250018)

2.5.2 [SOFTWARE REQUIREMENTS 7](#_TOC_250017)

3.[ARCHITECTURE 8](#_TOC_250016)

3.1 [PROJECT ARCHITECTURE 8](#_TOC_250015)

3.2 [DESCRIPTION 9](#_TOC_250014)

3.3 USECASDE DIAGRAM 10

3.4 CLASS DIAGRAM 11

3.5 [SEQUENCE DIAGRAM 12](#_TOC_250011)

* 1. [ACTIVITY DIAGRAM 13](#_TOC_250010)

4.IMPLEMENTATION 14

* 1. SAMPLE CODE 14

5.SCREENSHOTS 29

6.[TESTING](#_TOC_250009) 32

6.1 [INTRODUCTION TO TESTING](#_TOC_250008) 32

6.2 [TYPES OF TESTING](#_TOC_250007) 32

6.2.1 [UNIT TESTING 32](#_TOC_250006)

6.2.2 [INTEGRATION TESTING](#_TOC_250006) 33

6.2.3 [FUNCTIONAL TESTING 3](#_TOC_250005)3

6.3 [TEST CASES 3](#_TOC_250004)4

**7 .CONCLUSION & FUTURE SCOPE** 35

7.1 PROJECT CONCLUSION 35

7.2 [FUTURE SCOPE 3](#_TOC_250002)5

8.REFERENCES 36

8.1 [REFERENCES 3](#_TOC_250001)6

8.2 [GITHUB LINK 3](#_TOC_250000)6

#### INTRODUCTION

##### 1.1 PROJECT SCOPE

This project is titled “Detection of cyber attack in network detection using machine learning”. It actually provides a structured approach for developing a network intrusion detection system using machine learning techniques. Remember to adapt and refine it based on specific requirements, constraints, and available resources. Develop a system that can identify and classify various types of cyber attacks in a network using machine learning algorithms like SVM.

##### 1.2 PROJECT PURPOSE

This project has been developed to detect Cyber Attacks in Network using Machine Learning Techniques. It’s used to develop a robust and effective system for identifying and classifying various types of cyber attacks within a network The primary goals and purposes of this project include**, enhancing network security, early detection of cyber threats, reducing response time, improving accuracy and precision, adaptability to evolving threats, supporting incident response efforts.**

##### 1.3 PROJECT FEATURES

The main features of this project is that the system is capable of identifying cyber attacks in real-time or near-real-time, allowing for swift response and mitigation using machine learning algorithms like SVM-Support Vector Machine and also incorporates a diverse dataset containing labeled instances of both normal and malicious network traffic, ensuring robust model training and using intrusion detection system.

##### SYSTEM ANALYSIS

**SYSTEM ANALYSIS**

System Analysis is the important phase in the system development process. The System is studied to the minute details and analyzed. The system analyst plays an important role of an interrogator and dwells deep into the working of the present system. In analysis, a detailed study of these operations performed by the system and their relationships within and outside the system is done. A key question considered here is, “what must be done to solve the problem?” The system is viewed as a whole and the inputs to the system are identified.

##### 2.1 PROBLEM DEFINITION

In today's rapidly evolving digital landscape, the threat of cyber attacks poses significant risk to the security and integrity of networks. Traditional rule-based intrusion detection systems may not be sufficient to identify complex and novel attack patterns. Therefore, there is a critical need for a proactive and adaptable solution that can accurately detect and classify various types of cyber attacks in real-time.

.

##### 2.2 EXISTING SYSTEM

Within the ever-growing and quickly increasing field of cyber security, it is nearly impossible to quantify or justify the explanations why cyber security has such an outsized impact. Permitting malicious threats to run any place, at any time or in any context is a long way from being acceptable, and may cause forceful injury. It particularly applies to the Byzantine web of consumers and using the net and company information that cyber security groups are finding it hard to shield and contain.

##### 

##### Cyber security may be a necessary thought for people and families alike, also for businesses, governments, and academic establishments that operate inside the compass of the world network or net.

##### With the facility of Machine Learning, we will advance the cyber security landscape. Today’s high-tech infrastructure, that has network and cyber security systems, is gathering tremendous amounts of data and analytics on almost all the key aspects of mission-critical systems. Whereas people still give the key operational oversight and intelligent insights into today’s infrastructure. Most intrusion detection systems are focused on the perimeter attack surface threats, starting with your firewall. That offers protection of your network’s northsouth traffic, but what it doesn’t take into account is the lateral spread (east-west) that many network threats today take advantage of as they infiltrate your organization’s network and remain there unseen. We know this is true because research has shown that only 20% of discovered threats come from northsouth monitoring. When an IDS detects suspicious activity, the violation is typically reported to a security information and event management (SIEM) system where real threats are ultimately determined amid benign traffic abnormalities or other false alarms. However, the longer it takes to distinguish a threat, the more damage can be done. An IDS is immensely helpful for monitoring the network, but their usefulness all depends on what you do with the information that they give you. Because detection tools don’t block or resolve potential issues, they are ineffective at adding a layer of security unless you have the right personnel and policy to administer them and act on any threats. An IDS cannot see into encrypted packets, so intruders can use them to slip into the network. An IDS will not register these intrusions until they are deeper into the network, which leaves your systems vulnerable until the intrusion is discovered. This is a huge concern as encryption is becoming more prevalent to keep our data secure. One significant issue with an IDS is that they regularly alert you to false positives.

###### 

###### 2.2.1 DISADVANTAGES OF EXISTING SYSTEM

Following are the disadvantages of existing system:

* theft of corporate information.
* theft of financial information ( bank details or payment card details)
* theft of money.
* disruption to trading
* Signature-based systems rely on predefined patterns or signatures of known attacks. They are ineffective against new or evolving threats that don't match existing signatures

##### 2.3 PROPOSED SYSTEM

##### Machine Learning algorithms can be used to train and detect if there has been a cyber attack. As soon as the attack is detected, an email notification can be sent to the security engineers or users. Any classification algorithm can be used to categorize if it is a DoS/DDoS attack or not. One example of a classification algorithm is Support Vector Machine (SVM) which is a supervised learning method that analyses data and recognizes patterns. Since we cannot control when, where or how an attack may come our way, and absolute prevention against these cannot be guaranteed yet, our best shot for now is early detection which will help mitigate the risk of irreparable damage such incidents can cause. Organizations can use existing solutions or build their own to detect cyber attacks at a very early stage to minimize the impact. Any system that requires minimal human intervention would be ideal.

###### 

###### 2.3.1 ADVANTAGES OF THE PROPOSED SYSTEM

* Protection from malicious attacks on your network
* Detection and guaranteeing malicious elements within a preexisting network
* Prevents users from unauthorized access to the network
* Securing confidential information
* Deny’s program for certain resources that could be infected

##### 2.4 FEASIBILITY STUDY

The feasibility of the project is analyzed in this phase and business proposal is put forth with a very general plan for the project and some cost estimates. During system analysis the feasibility study of the proposed system is to be carried out. This is to ensure that the proposed system is not a burden to the company. For feasibility analysis, some understanding of the major requirements for the system is essential. Three key considerations involved in the feasibility analysis:

* Economic Feasibility
* Technical Feasibility
* Social Feasibility

###### 2.4.1 ECONOMIC FEASIBILITY

The developing system must be justified by cost and benefit. Criteria to ensure that effort is concentrated on a project, which will give best, return at the earliest. One of the factors, which affect the development of a new system, is the cost it would require.

The following are some of the important financial questions asked during preliminary investigation:

* The costs conduct a full system investigation.
* The cost of the hardware and software.
* The benefits in the form of reduced costs or fewer costly errors.

**2.4.2 TECHNICAL FEASIBILITY**

This study is carried out to check the technical feasibility, that is, the technical requirements of the system. Any system developed must not have a high demand on the available technical resources. This will lead to high demands on the available technical resources. This will lead to high demands being placed on the client. The developed system must have a modest requirement, as only minimal or null changes are required for implementing this system.

###### 2.4.3 SOCIAL FEASIBILITY

The aspect of study is to check the level of acceptance of the system by the user. This includes the process of training the user to use the system efficiently. The user must not feel threatened by the system, instead must accept it as a necessity. The level of acceptance by the users solely depends on the methods that are employed to educate the user about the system and to make him familiar with it. His level of confidence must be raised so that he is also able to make some constructive criticism, which is welcomed, as he is the final user of the system.

##### 2.5 HARDWARE & SOFTWARE REQUIREMENTS

###### 2.5.1 HARDWARE REQUIREMENTS:

Hardware interfaces specify the logical characteristics of each interface between the software product and the hardware components of the system. The following are some hardware requirements.

* Processor: Pentium IV or higher
* RAM: 256 MB
* Space on Hard Disk: minimum 512MB

##### 2.5.2 SOFTWARE REQUIREMENTS:

Software Requirements specifies the logical characteristics of each interface and software components of the system. The following are some software requirements.

* Python
* Mysql
* Django
* Wampserver

##### ARCHITECTURE

##### 3.1 PROJECT ARCHITECTURE

This project architecture shows the procedure followed for classification, starting from input to final prediction.

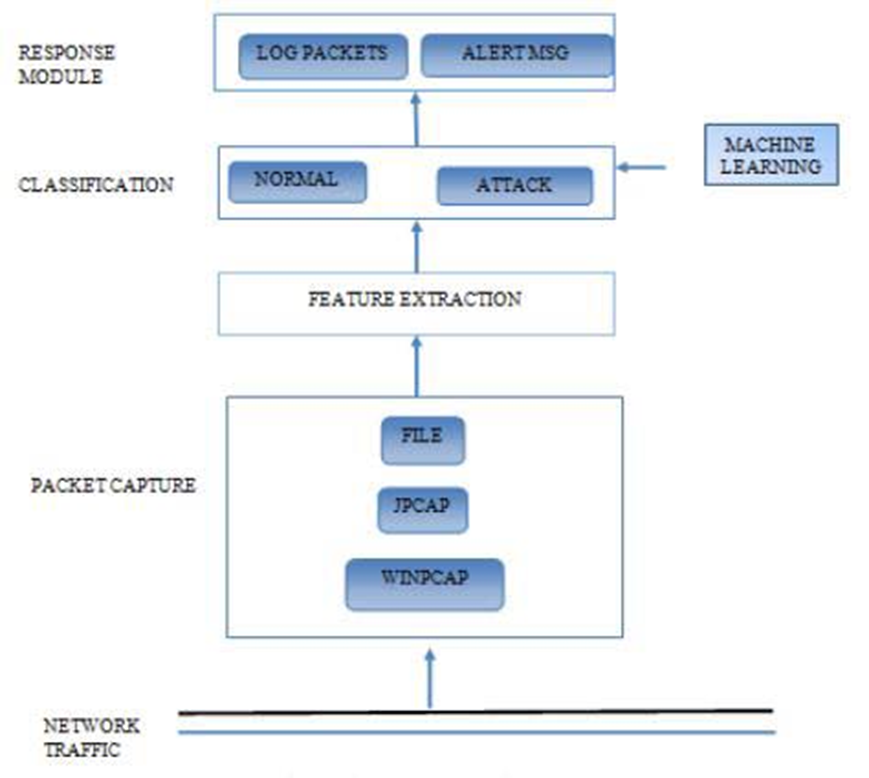


Figure 3.1: Project Architecture of Cyber Attack Detection in Network

###### 3.2 DESCRIPTION

The architecture for theProject encompasses various components that work together to achieve the objective of identifying and classifying cyber attacks. Raw network traffic data is collected from various sources, such as network taps, packet capture devices, or log files from network devices. This data is then cleaned, normalized, and transformed to a format suitable for machine learning algorithms. Preprocessing steps may include packet dissection, protocol parsing, and feature extraction.

Identifying key features such as source/destination IP addresses, ports, protocols, packet sizes, and other network traffic characteristics. Additionally, techniques like dimensionality reduction may be applied to enhance the efficiency of the machine learning models.

Machine learning algorithms, such as Support Vector Machines, or Neural Networks, are evaluated for their performance. Hyperparameters are fine-tuned to optimize the models. The models are trained on the preprocessed data, utilizing both normal and malicious instances.

###### 3.3 USE CASE DIAGRAM

In the use case diagram, we have basically one actor who is the user in the trained model.

A use case diagram is a graphical depiction of a user's possible interactions with a system. A use case diagram shows various use cases and different types of users the system has. The use cases are represented by either circles or ellipses. The actors are often shown as stick figures.

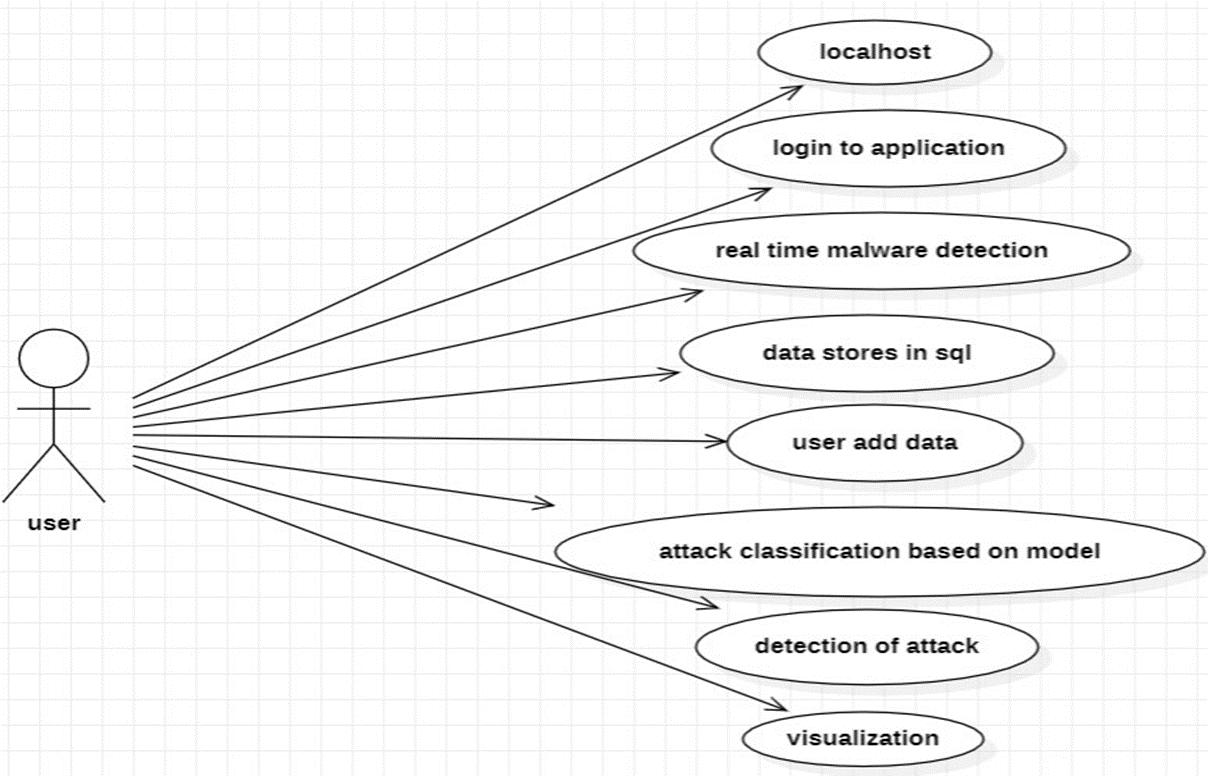


Figure 3.3: Use Case Diagram for Cyber Attack Detection in Network

##### 3.4 CLASS DIAGRAM

Class diagram is a type of static structure diagram that describes the structure of a system by showing the system’s classes, their attributes, operations (or methods), and the relationships among objects.

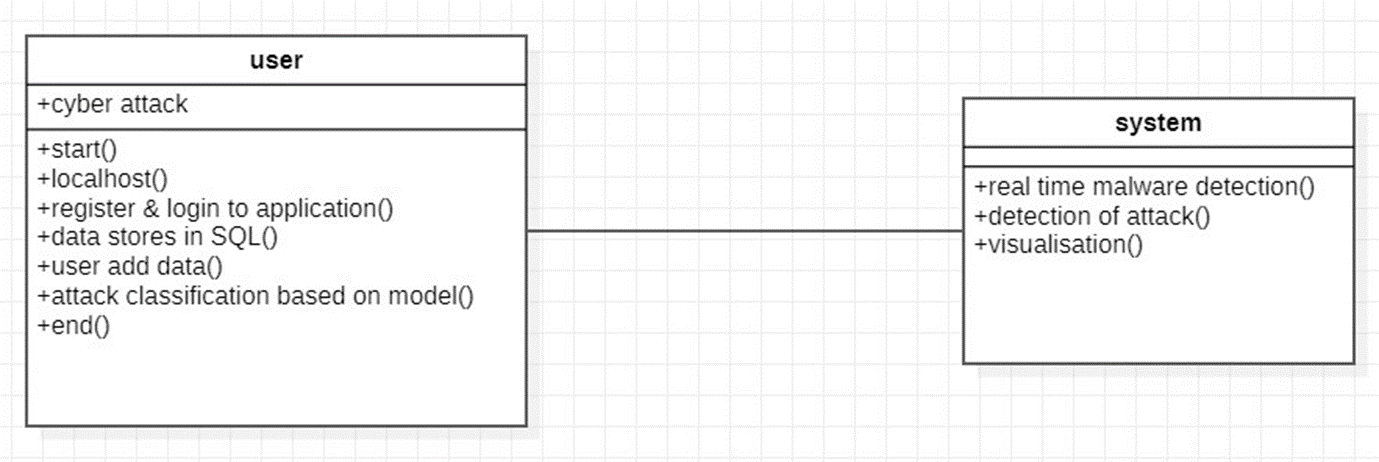
****

Figure 3.4: Class Diagram for Cyber Attack Detection in Network

##### 3.5 SEQUENCE DIAGRAM

A sequence diagram shows object interactions arranged in time sequence. It depicts the objects involved in the scenario and the sequence of messages exchanged between the objects needed to carry out the functionality of the scenario. Sequence diagrams are typically associated with use case realizations in the logical view of the system under development.

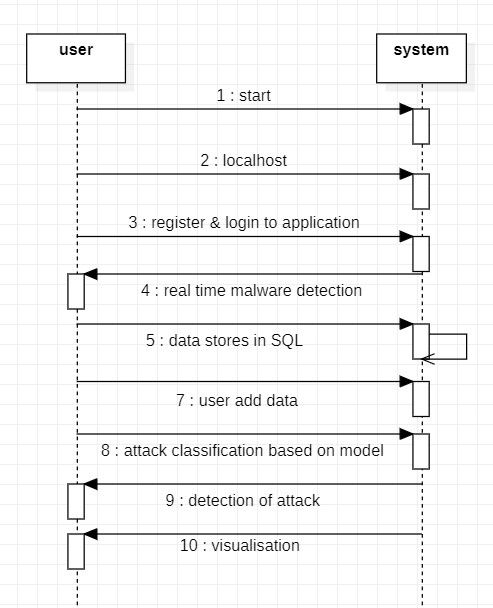


Figure 3.5: Sequence Diagram for Cyber Attack Detection in Network

###### 3.6 ACTIVITY DIAGRAM

Activity diagrams are graphical representations of workflows of stepwise activities and actions with support for choice, iteration and concurrency. They can also include elements showing the flow of data between activities through one or more data stores.

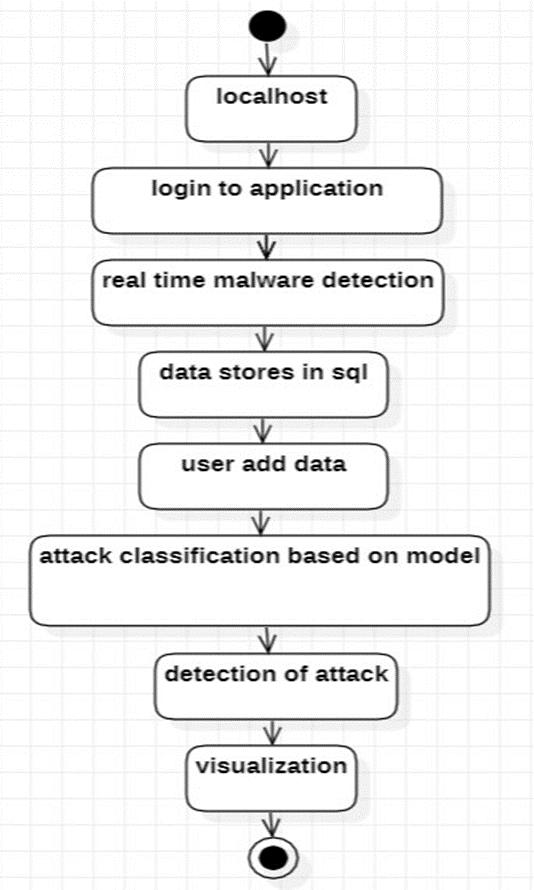


Figure 3.6: Activity Diagram for Cyber Attack Detection in Network

**4. IMPLEMENTATION**

##### 4.1 SAMPLE CODE

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

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('NSL\_Dataset/Train.txt',sep=',')

test=pd.read\_csv('NSL\_Dataset/Test.txt',sep=',')

train.head()

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\_srv\_count","dst\_host\_same\_srv\_rate",

"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"]

train.columns=columns

test.columns=columns

train.head()

test.head()

train.describe().T

test.describe().T

train['attack'].value\_counts()

test['attack'].value\_counts()

Mutinomial Classification

In attack\_class normal means 0, DOS means 1, PROBE means 2, R2L means 3 and U2R means 4.

train['attack\_class']=np.where(train.attack=='normal',0,np.where((train.attack=='back') | (train.attack=='land') | (train.attack=='pod') | (train.attack=='neptune') |

(train.attack=='smurf') | (train.attack=='teardrop') | (train.attack=='apache2') | (train.attack=='udpstorm') |

(train.attack=='processtable') | (train.attack=='worm') | (train.attack=='mailbomb'),1,np.where((train.attack=='satan') | (train.attack=='ipsweep') | (train.attack=='nmap') | (train.attack=='portsweep') |

(train.attack=='mscan') | (train.attack=='saint'),2,np.where((train.attack=='guess\_passwd') | (train.attack=='ftp\_write') | (train.attack=='imap') | (train.attack=='phf') |

(train.attack=='multihop') | (train.attack=='warezmaster') | (train.attack=='warezclient') | (train.attack=='spy') |

(train.attack=='xlock') | (train.attack=='xsnoop') | (train.attack=='snmpguess') | (train.attack=='snmpgetattack') |

(train.attack=='httptunnel') | (train.attack=='sendmail') | (train.attack=='named'),3,4))))

test['attack\_class']=np.where(test.attack=='normal',0,np.where((test.attack=='back') | (test.attack=='land') | (test.attack=='pod') | (test.attack=='neptune') |

(test.attack=='smurf') | (test.attack=='teardrop') | (test.attack=='apache2') | (test.attack=='udpstorm') |

(test.attack=='processtable') | (test.attack=='worm') | (test.attack=='mailbomb'),1,np.where((test.attack=='satan') | (test.attack=='ipsweep') | (test.attack=='nmap') | (test.attack=='portsweep') |

(test.attack=='mscan') | (test.attack=='saint'),2,np.where((test.attack=='guess\_passwd') | (test.attack=='ftp\_write') | (test.attack=='imap') | (test.attack=='phf') |

(test.attack=='multihop') | (test.attack=='warezmaster') | (test.attack=='warezclient') | (test.attack=='spy') |

(test.attack=='xlock') | (test.attack=='xsnoop') | (test.attack=='snmpguess') | (test.attack=='snmpgetattack') |

(test.attack=='httptunnel') | (test.attack=='sendmail') | (test.attack=='named'),3,4))))

Binomial Classification

In attack\_class normal means 0 and attack means 1.

train['attack\_class']=np.where(train.attack=='normal',0,1)

train.attack\_class.value\_counts()

test['attack\_class']=np.where(test.attack=='normal',0,1)

test.attack\_class.value\_counts()

# \*\*Basic Exploratory Analysis\*\*

# Protocol type distribution

plt.figure(figsize=(9,8))

sns.countplot(x="protocol\_type", data=train)

plt.show()

# Protocol type distribution

plt.figure(figsize=(10,15))

sns.countplot(y="service", data=train)

plt.show()

# Protocol type distribution

plt.figure(figsize=(8,8))

sns.countplot(x="flag", data=train)

plt.show()

# Protocol type distribution

plt.figure(figsize=(6,6))

sns.countplot(y="attack", data=train)

plt.show()

# Protocol type distribution

plt.figure(figsize=(6,6))

sns.countplot(x="attack\_class", data=train)

plt.show()

flag\_count = train[['flag', 'attack\_class']].groupby(['flag', 'attack\_class']).size()

flag\_count\_percent = flag\_count.groupby(level=[0]).apply(lambda x: x / x.sum()).reset\_index()

flag\_count\_percent.columns = ['flag', 'attack\_class', 'percent']

sns.factorplot(y="flag",

x = 'percent',

hue="attack\_class",

data = flag\_count\_percent,

size=6,

kind="bar",

palette="muted")

type\_count = train[['protocol\_type', 'attack\_class']].groupby(['protocol\_type', 'attack\_class']).size()

type\_count\_percent = type\_count.groupby(level=[0]).apply(lambda x: x / x.sum()).reset\_index()

type\_count\_percent.columns = ['protocol\_type', 'attack\_class', 'percent']

sns.factorplot(x="protocol\_type",

y = 'percent',

hue="attack\_class",

data = type\_count\_percent,

size=6,

kind="bar",

palette="muted")

identifying relationships (between Y & numerical independent variables by comparing means)

train.groupby('attack\_class').mean().T

# 13. Lets check corrleation between Variables

corrmat = train.corr()

corrmat

Data Audit

tr\_num\_var=train.select\_dtypes(['int64','float64','int32','float32'])

ts\_num\_var=test.select\_dtypes(['int64','float64','int32','float32'])

tr\_cat\_var = train.select\_dtypes('object')

ts\_cat\_var=test.select\_dtypes('object')

# Create Data audit Report for continuous variables

def continuous\_var\_summary(x):

return pd.Series([x.count(), x.isnull().sum(), x.sum(), x.mean(), x.median(),

x.std(), x.var(), x.min(), x.quantile(0.01), x.quantile(0.05),

x.quantile(0.10),x.quantile(0.25),x.quantile(0.50),x.quantile(0.75),

x.quantile(0.90),x.quantile(0.95), x.quantile(0.99),x.max()],

index = ['N', 'NMISS', 'SUM', 'MEAN','MEDIAN', 'STD', 'VAR', 'MIN','P1',

'P5' ,'P10' ,'P25' ,'P50' ,'P75' ,'P90' ,'P95' ,'P99' ,'MAX'])

# Create Data audit Report for categorical variables

def categorical\_var\_summary(x):

Mode = x.value\_counts().sort\_values(ascending = False)[0:1].reset\_index()

return pd.Series([x.count(), x.isnull().sum(), Mode.iloc[0, 0], Mode.iloc[0, 1],

round(Mode.iloc[0, 1] \* 100/x.count(), 2)],

index = ['N', 'NMISS', 'MODE', 'FREQ', 'PERCENT'])

# An utility function to create dummy variable

def create\_dummies(df, colname):

col\_dummies = pd.get\_dummies(df[colname], prefix = colname, drop\_first = True)

df = pd.concat([df, col\_dummies], axis = 1)

df.drop(colname, axis = 1, inplace = True )

return df

tr\_num\_var.apply(continuous\_var\_summary).T.round(2)

ts\_num\_var.apply(continuous\_var\_summary).T.round(2)

# alternate of .describe() for categorical variables

tr\_cat\_var.apply(categorical\_var\_summary).T

ts\_cat\_var.apply(categorical\_var\_summary).T

tr\_num\_var.apply(continuous\_var\_summary).T.round(2)

ts\_num\_var.apply(continuous\_var\_summary).T.round(2)

# get the useful categorical variables

tr\_cat\_var = train[['protocol\_type', 'service','flag','attack']]

# for c\_feature in categorical\_features

for c\_feature in ['protocol\_type', 'service','flag','attack']:

tr\_cat\_var[c\_feature] = tr\_cat\_var[c\_feature].astype('category')

tr\_cat\_var = create\_dummies(tr\_cat\_var, c\_feature)

# get the useful categorical variables

ts\_cat\_var = test[['protocol\_type', 'service','flag','attack']]

# for c\_feature in categorical\_features

for c\_feature in ['protocol\_type', 'service','flag','attack']:

ts\_cat\_var[c\_feature] = ts\_cat\_var[c\_feature].astype('category')

ts\_cat\_var = create\_dummies(ts\_cat\_var, c\_feature)

Final datasets

train\_new= pd.concat([tr\_num\_var, tr\_cat\_var], axis = 1)

test\_new = pd.concat([ts\_num\_var, ts\_cat\_var], axis = 1)

corrm=train\_new.corr()

corrm

plt.figure(figsize = (10, 8))

sns.heatmap(corrm)

train\_new.drop(columns=['land','wrong\_fragment','urgent','num\_failed\_logins',"root\_shell","su\_attempted","num\_root", "num\_file\_creations","num\_shells","num\_access\_files","num\_outbound\_cmds","is\_host\_login","is\_guest\_login", 'dst\_host\_rerror\_rate','dst\_host\_serror\_rate','dst\_host\_srv\_rerror\_rate','dst\_host\_srv\_serror\_rate',

'num\_root','num\_outbound\_cmds','srv\_rerror\_rate','srv\_serror\_rate'],inplace=True)

plt.figure(figsize = (10, 8))

sns.heatmap(train\_new.corr())

Recursive Feature Elimination (RFE)

import warnings

warnings.filterwarnings("ignore")

from sklearn import datasets

from sklearn.feature\_selection import RFE

from sklearn.linear\_model import LogisticRegression

X = train\_new[train\_new.columns.difference(['attack\_class'])]

logreg = LogisticRegression(solver='lbfgs',multi\_class='auto')

rfe = RFE(logreg, 15)

rfe = rfe.fit(X, train\_new['attack\_class'] )

print(rfe.support\_)

print(rfe.ranking\_)

# capturing the important variables

RFE\_features=X.columns[rfe.get\_support()]

RFE\_features

all\_columns = "+".join(train\_new.columns.difference( ['attack\_class'] ))

print(all\_columns)

Variance Inflation Factor assessment

# import the packages for vif

from statsmodels.stats.outliers\_influence import variance\_inflation\_factor

from patsy import dmatrices

# run the dmatrices

a, b = dmatrices(formula\_like='''attack\_class ~ count+diff\_srv\_rate+dst\_bytes+dst\_host\_count+

dst\_host\_diff\_srv\_rate+dst\_host\_same\_src\_port\_rate+

dst\_host\_srv\_diff\_host\_rate+duration+

hot+last\_flag+logged\_in+num\_compromised+

srv\_count+srv\_diff\_host\_rate''', data =train\_new, return\_type = 'dataframe')

# get the VIF

vif = pd.DataFrame()

vif["VIF Factor"] = [variance\_inflation\_factor(b.values, i) for i in range(b.shape[1])]

vif["features"] = b.columns

vif

Final Variables

cols=['count','diff\_srv\_rate','dst\_bytes','dst\_host\_count',

'dst\_host\_diff\_srv\_rate','dst\_host\_same\_src\_port\_rate',

'dst\_host\_srv\_diff\_host\_rate','duration',

'hot','last\_flag','logged\_in','num\_compromised',

'srv\_count','srv\_diff\_host\_rate']

Model Building

train\_X=train\_new[cols]

train\_y=train\_new['attack\_class']

test\_X=test\_new[cols]

test\_y=test\_new['attack\_class']

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)

print(metrics.classification\_report(test\_y, test\_pred))

clf\_tree = DecisionTreeClassifier( max\_depth = 11, max\_features=13)

clf\_tree.fit( train\_X, train\_y )

train\_X.columns

clf\_tree.feature\_importances\_

list(zip(train\_X.columns,clf\_tree.feature\_importances\_ ))

tree\_test\_pred = pd.DataFrame( { 'actual': test\_y,

'predicted': clf\_tree.predict( test\_X ) } )

tree\_test\_pred.sample( n = 10 )

metrics.accuracy\_score( tree\_test\_pred.actual, tree\_test\_pred.predicted )

tree\_cm = metrics.confusion\_matrix( tree\_test\_pred.predicted,

tree\_test\_pred.actual,

[1,0] )

sns.heatmap(tree\_cm, annot=True,

fmt='.2f',

xticklabels = ["Yes", "No"] , yticklabels = ["Yes", "No"] )

plt.ylabel('True label')

plt.xlabel('Predicted label')

metrics.roc\_auc\_score( tree\_test\_pred.actual, tree\_test\_pred.predicted )

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 ) } )

print(metrics.accuracy\_score( radm\_test\_pred.actual, radm\_test\_pred.predicted ))

#print(metrics.roc\_auc\_score( radm\_test\_pred.actual, radm\_test\_pred.predicted ))

tree\_cm = metrics.confusion\_matrix( radm\_test\_pred.predicted,

radm\_test\_pred.actual,

[1,0] )

sns.heatmap(tree\_cm, annot=True,

fmt='.2f',

xticklabels = ["Yes", "No"] , yticklabels = ["Yes", "No"] )

plt.ylabel('True label')

plt.xlabel('Predicted label')

Neural Network Model

from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()

# Fit only to the training data

scaler.fit(train\_X)

# Now apply the transformations to the data:

train\_X = scaler.transform(train\_X)

test\_X = scaler.transform(test\_X)

from sklearn.neural\_network import MLPClassifier

mlp = MLPClassifier(hidden\_layer\_sizes=(30,30,30))

mlp.fit(train\_X, train\_y)

predictions = mlp.predict(test\_X)

from sklearn.metrics import classification\_report,confusion\_matrix

print(confusion\_matrix(test\_y,predictions))

print(classification\_report(test\_y,predictions))

len(mlp.coefs\_)

len(mlp.coefs\_[0])

len(mlp.intercepts\_[0])

mlp.coefs\_

mlp.score(train\_X,train\_y)

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\_)

model = grid.best\_estimator\_

yfit = model.predict(test\_X)

from sklearn.metrics import classification\_report

print(classification\_report(test\_y, yfit))

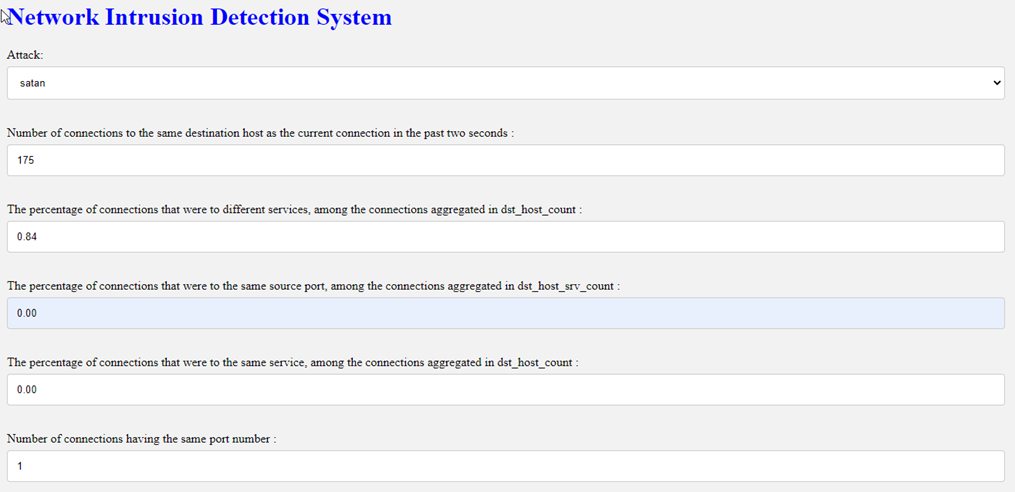
from sklearn.metrics import confusion\_matrix

mat = confusion\_matrix(test\_y, yfit)

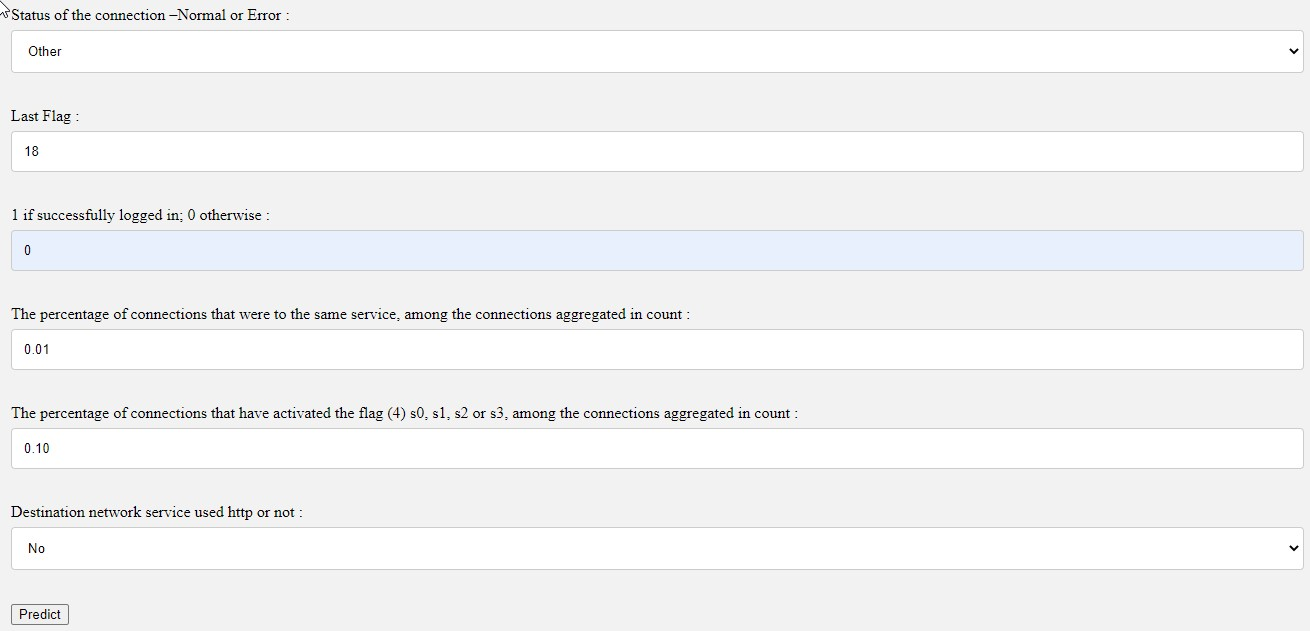
sns.heatmap(mat.T, square=True, annot=True, fmt='d', cbar=False)

plt.xlabel('true label')

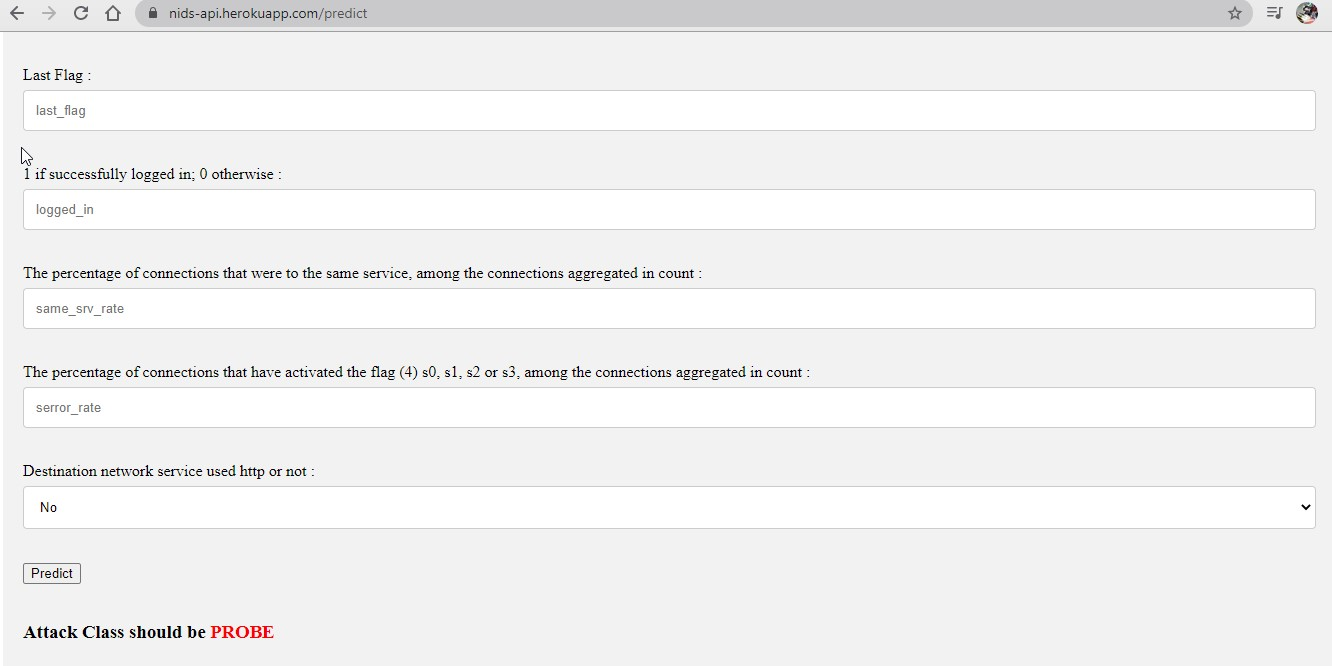
plt.ylabel('predicted label')



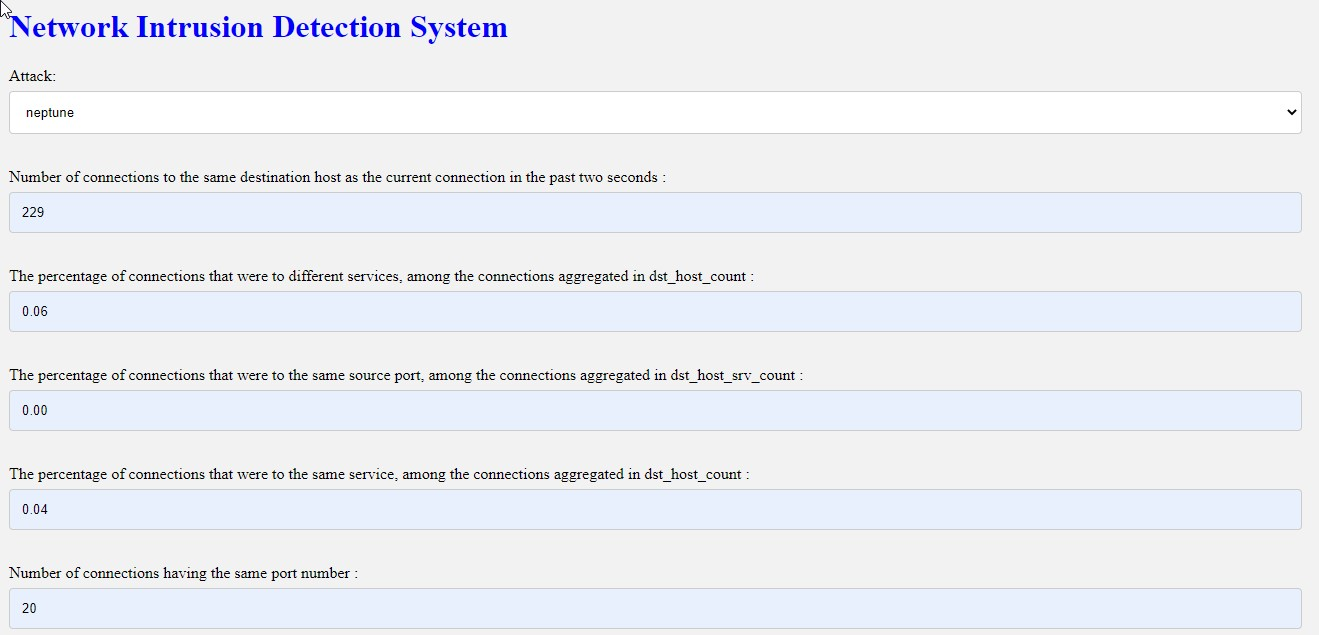
Screenshot 5.1: Input 1 example for detecting cyber attack



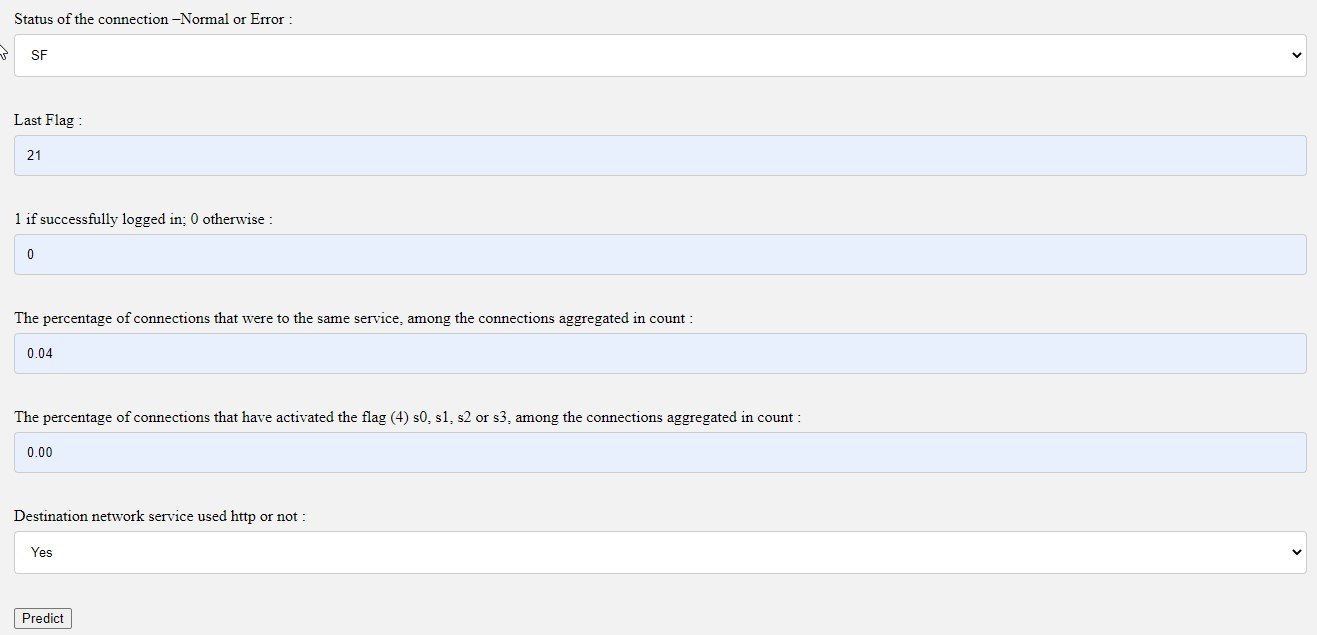
Screenshot 5.2: Input 1 example for detecting cyber attack



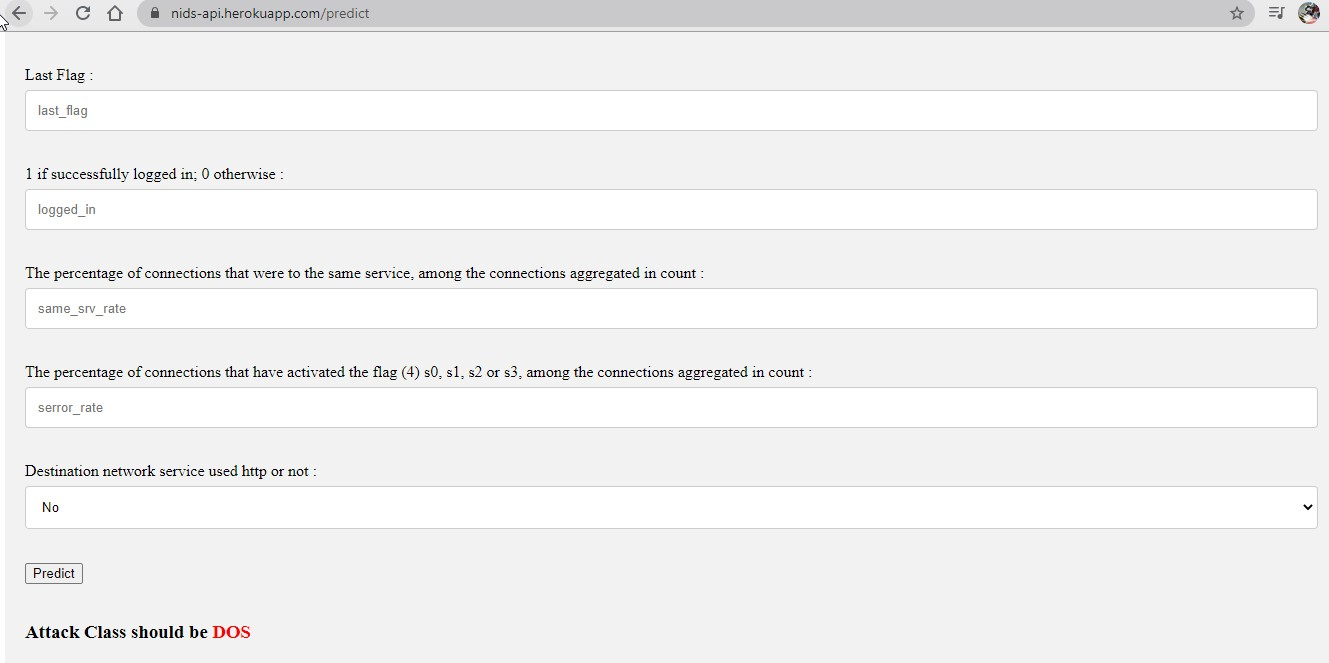
Screenshot 5.3: Result of the above given example input 1 for detecting cyber attack



Screenshot 5.4: Input 2 example for detecting cyber attack



Screenshot 5.5: Input 2 example for detecting cyber attack



Screenshot 5.6: Result of the above given example input 2 for detecting cyber attack

#### 6. TESTING

##### 6.1 INTRODUCTION TO TESTING

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, subassemblies, assemblies and/or a finished product. It is the process of exercising software with the intent of ensuring that the Software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of tests. Each test type addresses a specific testing requirement.

##### 6.2 TYPES OF TESTING

###### 6.2.1 UNIT TESTING

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program inputs produce valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application .It is done after the completion of an individual unit before integration. This is a structural testing that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at component level and test a specific business process, application and/or system configuration. Unit tests ensure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expected results.

###### 6.2.2 INTEGRATION TESTING

Integration tests are designed to test integrated software components to determine if they actually run as one program. Integration tests demonstrate that although the components were individually satisfactory, as shown by successfully unit testing, the combination of components is correct and consistent. Integration testing is specifically aimed at exposing the problems that arise from the combination of components.

###### 6.2.3 FUNCTIONAL TESTING

Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals.

Functional testing is centered on the following items:

Valid Input : identified classes of valid input must be accepted.

Invalid Input

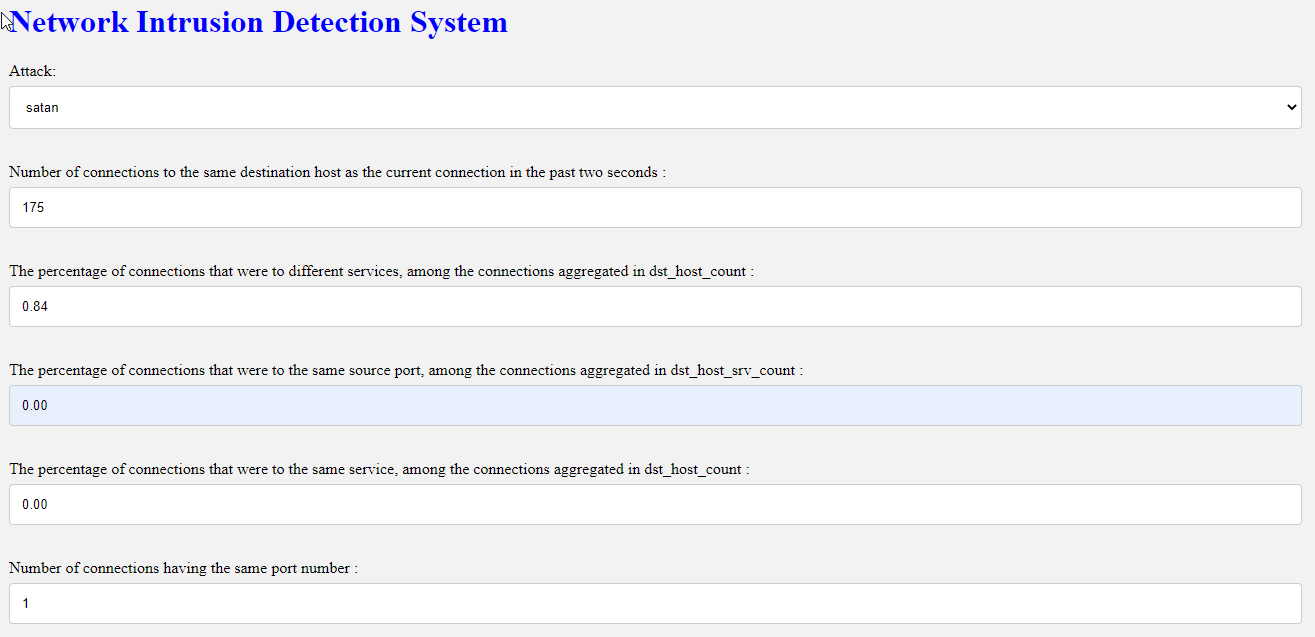
: identified classes of invalid input must be rejected.

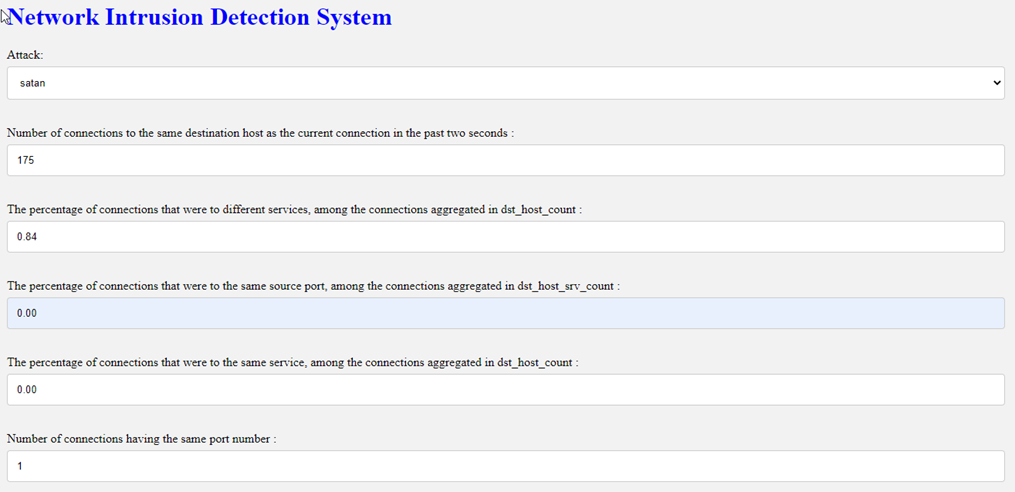
Functions : identified functions must be exercised.

Output : identified classes of application outputs must be exercised.

Systems/Procedures: interfacing systems or procedures must be invoked. Organization and preparation of functional tests is focused on requirements, key functions, or special test cases.

##### 6.3 TEST CASES





These are some of the example testcases

##### CONCLUSION & FUTURE SCOPE

##### PROJECT CONCLUSION

The project, "Detection of Cyber Attacks in Network using Machine Learning Techniques," represents a significant advancement in network security. By leveraging machine learning algorithms (SVM), we have developed a proactive and adaptable system capable of accurately identifying and classifying various types of cyber attacks in real-time. Throughout the project, we addressed the limitations of existing systems, such as their reliance on predefined signatures and rules, as well as their difficulty in adapting to new and evolving threats. Our system demonstrated notable improvements in accuracy, reduced false positives/negatives, and provided a more context-aware approach to network traffic analysis. The project also prioritizes ethical considerations and compliance with data privacy regulations, safeguarding the integrity and confidentiality of sensitive information. Furthermore, detailed documentation and contingency plans have been established to facilitate seamless knowledge transfer and ensure system sustainability.

##### 7.2 FUTURE SCOPE

The project lays the foundation for several potential future enhancements and expansions Continuous research and development in machine learning techniques may lead to more advanced models that further improve detection accuracy and efficiency. Incorporating behavioral analysis and anomaly detection techniques can further enhance the system's ability to identify unusual activities indicative of cyber attacks. Integration with threat intelligence feeds and services can provide the system with real-time updates on emerging threats, allowing for more timely and accurate detection. Integrating with security orchestration, automation, and response (SOAR) platforms can streamline incident response workflows and enable more efficient threat mitigation.

##### 8. BIBLIOGRAPHY

##### REFERENCES

* DipankarDasgupta. Immunity-basedintrusion detection system: A general framework. In Proceedings of the 22nd National Information Systems Security Confer Jonatan Gomez and DipankarDasgupta
* Evolving fuzzy classi\_ers for intrusion detection. In Proceedings of the 2002 IEEE Workshop on Information Assurance,West Point, NY, USA, 2002. ence (NISSC) Arlington, Virginia, USA, 1999.
* Peter Mell Karen Scarfone. Guide to intrusion detection and prevention systems (idps). National Institute of Standards and Technology, NIST SP - 800-94, 2007.
* A. Shabtai, E. Menahem and Y. Elovici. F-Sign: automatic, function-based signature generation for malware, systems, man, and cybernetics, Part C: applications and reviews. Transactions on IEEE, 41, 494–508, 2011.
* M. Sharma and D. Toshniwal. Pre-clustering algorithm for anomaly detection and clustering that uses variable size buckets. RecentAdvances in Information Technology, 515–519, 2012.

##### 8.2 GITHUB LINK

<https://github.com/vamshik-1/cyberbreache-detection-mini-project>