An Industry-Oriented Mini Project Report

On

“MACHINE LEARNING TECHNIQUES FOR CYBER ATTACKS DETECTION”

Submitted in Partial Fulfillment of the
Academic Requirement for the Award
of Degree of

BACHELOR OF TECHNOLOGY

in

Computer Science and Engineering
(Artificial Intelligence & Machine Learning)

SubmittedBy:

K.ROHITH	(20R01A66E5)
G.HARIKA	(20R01A66D9)
R.SHREYA	(20R01A66G4)
SANA TABASSUM	(20R01A66G7)

Under the esteemed guidance of

Dr.S.Dhanalakshmi

CMR INSTITUTE OF TECHNOLOGY

(UGCAUTONOMOUS)

Approved by AICTE, Permanent Affiliation to JNTUH, Accredited by NBA and NAAC

Kandlakoya(V),MedchalDist-501 401

www.cmrithyderabad.edu.in

2023-24

CMRINSTITUTEOFTECHNOLOGY

(UGCAUTONOMOUS)

Approved by AICTE, Permanent Affiliation to JNTUH, Accredited by NBA and NAAC
Kandlakoya (V),Medchal Dist-501 401
www.cmrithyderabad.edu.in

CERTIFICATE

This is to certify that an Industry oriented Mini Project entitled with "MACHINE LEARNING TECHNIQUES FOR CYBER ATTACK DETECTION" is being submitted by:

K.ROHITH	**(20R01A66E5)**
G.HARIKA	**(20R01A66D9)**
R.SHREYA	**(20R01A66G4)**
SANA TABASSUM	**(20R01A66G7)**

To JNTUH, Hyderabad, in partial fulfillment of the requirement for award of the degree of B.Tech in CSE (AI&ML) and is a record of a bonafide work carried out under our guidance and supervision. The results in this project have been verified and are found to be satisfactory. The results embodied in this work have not been submitted to have any other University for award of any other degree or diploma.

Signature of Guide
Ms.K.Rampriya

Signature of Project Coordinator
Dr.S.Dhanalakshmi

Signature of HOD
Prof.P.Pavan Kumar

EXAMINER

ACKNOWLEDGEMENT

We are extremely grateful to **Dr. M Janga Reddy**, Director, **Dr. B. Satyanarayana**, Principal and **Prof.P.Pavan Kumar** , Head of Department, Dept of Computer Science and Engineering (AI &ML), CMR Institute of Technology for their inspiration and valuable guidance during entire duration.

We are extremely thankful to Dr.S.Dhanalakshmi**,** Mini Project Coordinator and internal guide **Ms. K. Rampriya**, Dept of Computer Science and Engineering (AI & ML), CMR Institute of Technology for their constant guidance, encouragement and moral support throughout the project.

We will be failing in duty if we do not acknowledge with grateful thanks to the authors of the references and other literatures referred in this Project.

We express our thanks to all staff members and friends for all the help and coordination extended in bringing out this Project successfully in time.

Finally, we are very much thankful to our parents and relatives who guided directly or indirectly for every step towards success.

K.ROHITH	**(20R01A66E5)**
G.HARIKA	**(20R01A66D9)**
R.SHREYA	**(20R01A66G4)**
SANA TABASSUM	**(20R01A66G7)**

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. . We may benefit from this union in a number of ways, including by giving ML models better security, enhancing the effectiveness of cyber security techniques, and enabling the efficient detection of zero-day threats with minimal human involvement. We combine cyber security and ML to address two distinct themes in this survey article. By providing ML strategies for cyber security, the purpose of this paper is to give a wide overview of ML methods employed in cyberspace security.

INDEX

LISTOFFIGURES

LISTOFSCREENSHOTS

1. INTRODUCTION

1.1 ABOUTPROJECT

The amount of time spent on the Internet has significantly grown because to advancements in computer system, internet and smart phone. Millions of various networked computers, networks, and related devices make up the global Internet. As a result, online criminals and adversaries now have the Internet as a target. Information confidentiality, availability, and integrity must all be guaranteed via a solid, secure computer system. When an unauthorized individual, software, or unlawful breach accesses a system or network with the aim to cause harm or interfere with regular operations, the computer system's authenticity and privacy are seriously compromised. Cybersecurity refers to a set of safeguarding practices that may be used to secure the digital environment and user activities against unwanted access and assaults.

A cyber-defense system's primary goal is to ensure data security, integrity, and accessibility [2]. Internal loopholes in computer system and network setup and implementation render them at risk of cyber-attacks and threats Weaknesses in the design of information network systems involve improper design, an absence of adequate protocols, and inexperienced or unskilled staff. These shortcomings increase the risk of threats and assaults on your network either from the inside or outside. Quite a few people from various fields are addicted to cyber networks. An agent that alters the behavior and operations of a computer or network negatively and unintentionally using a specific penetration technique is referred to as a threat [3]. Cybersecurity strives to defend against online threats to the confidentiality of data, networks, and algorithms. Between cybercriminals and defense, there has been a competition since the first computer virus emerged in 1970 [4]. It is getting harder and harder to defend against these cybersecurity threats and stay up with their pace. In order to overcome these security difficulties, researchers are currently concentrating on the urgent need to discover new automated security approaches.

One of the most efficient and beneficial methods to achieve this aim is to use independent ML algorithms to identify novel and undiscovered crimes [5]. With the use of machine learning techniques, we can identify spam, detect fraud, malware, phishing, dark or deep web sites, and uncover breaches. Human deficiencies in these particular cybercrime detection approaches can be alleviated by ML techniques. Additionally, detecting and responding to new generation cyberattacks requires a proactive approach.

One of the potential methods to rapidly resist such attacks is machine learning (ML), which has the ability to learn from experience and react to future threats in a timely manner. Today's common cybersecurity technologies include firewalls, antivirus software, intrusion prevention systems (IPS), SEIM solutions, and unified threat management (UTM). Traditional solutions rely on static management of devices in accordance with predetermined network security rules and lack automation (using AI approaches). In terms of performance, error rate, and reaction to cyberattack, AI-based systems perform better than conventional threat detection approaches. They also have lower error rates while detecting and reacting to assaults than conventional systems. ML models play a key role in improving performance and providing robust and intelligent techniques to detect attacks early and mitigate the impact and damage they cause. It combines ML techniques in order to increase precision of accurate and early cyberattack categorization. However, the majority of investigations have used inadequate datasets. Neither of the research emphasized a complete and accurate perspective of cyberattacks and threats against both computer networks and mobile devices.

As a research effort, cybersecurity's history began. In the 1970s, Robert Thomas, a researcher at Cambridge, Massachusetts-based BBN Technologies, invented the first computer "worm." Its name was Creeper. With the phrase "**I'M THE CREEPER: CATCH ME IF YOU CAN.**" The Creeper infected computers by bouncing from system to system. The Reaper was the first antivirus program developed by Ray Tomlinson, who invented email. It was a replicating program that would hunt for and eliminate Creeper. Robert Morris had an idea toward the end of 1988 to gauge the size of the internet. He created software to accomplish this that entered UNIX terminals, traversed networks, and cloned itself. Because the Morris worm was so aggressive, computers were rendered completely inoperable. He was later the first to be found guilty under the Computer Fraud and Abuse Act. Techniques and procedures created to protect electronic data are referred to as cybersecurity. Data is what criminals ultimately desire, after all. Computers, servers, and networks are only tools for accessing data. Effective cybersecurity reduces the possibility of cyberattacks and protects individuals and organizations against unauthorized use of technology and systems.

The potential risks and dangers of all security breaches mentioned are called threats, and attempts to commit breaches are called attacks. There are various ways to describe cybersecurity, including defining it in terms of the most dangerous assaults, such as phishing and malware. Phishing, often referred to as brand cloning, is the practice of gaining access to personally identifiable information in order to manipulate or abuse it by pretending to be an authorized user

1.2 EXISTINGSYSTEM:

- 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. In many cases false positives are more frequent than actual threats. An IDS can be tuned to reduce the number of false positives; however, your engineers will still have to spend time responding to them. If they don’t take care to monitor the false positives, real attacks can slip through or be ignored.

DISADVANTAGES:

- There is a possibility of misjudging.
- Collecting data from the sources might be difficult if the data processing sites are faraway.
- Continuous monitoring of the detecting not be possible.
- The predictions of cyber attacks cannot be more accurate.

1.3 PROPOSEDSYSTEM

- 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.

ADVANTAGES:

- Different architectures are considered for different problems.
- The feature extraction function increases the efficiency.
- The subtle changes can be identified easily and quickly.
- The prediction of cyber attack detection can be given more accurately.

2. REQUIREMENTSSPECIFICATIONS

2.1 REQUIREMENTANALYSIS

2.1.1 HARDWAREREQUIREMENTS

System	Pentium IV 2.4GHz
HardDisk	40 GB
Floppy Drive	1.44Mb
Monitor	14’ Color Monitor
Mouse	Optical Mouse
RAM	512 Mb

2.1.2 SOFTWAREREQUIREMENTS

OperatingSystem	Windows7or above
ProgrammingLanguage	Python3
Front-End	Python
Designing	HTML, CSS, JS
Database	MySQL

2.2 SPECIFICATIONPRINCIPLES

2.2.1 SOFTWAREDESCRIPTION

Python:

Python is one of those rare languages which can claim to be both *simple* and powerful. You will find yourself pleasantly surprised to see how easy it is to concentrate on the solution to the problem rather than the syntax and structure of the language you are programming in.The official introduction to Python is Python is an easy to learn, powerful programming language. It has efficient high-level datastructures and a simple but effective approach to object-oriented programming. Python's elegant syntax and dynamic typing, together with its interpreted nature, make it an ideal language for scripting and rapid application development in many areas on most platforms. I will discuss most of these features in more detail in the next section.

Features of Python

1. **Simple:**

   Python is a simple and user friendly language. Reading a good Python program feels almost like reading English, although very strict English! This pseudo-code nature of Python is one of its greatest strengths. It allows you to concentrate on the solution to the problem rather than the language itself.

2. **Easy to Learn:**

   As you will see, Python is extremely easy to get started with. Python has an extra ordinarily simple syntax, as already mentioned.

3. Free and Open Source:

Python is an example of a *FLOSS* (Free/Libré and Open Source Software). In simple terms, you can freely distribute copies of this software, read its source code, make changes to it, and use pieces of it in new free programs. FLOSS is based on the concept of a community which shares knowledge. This is one of the reasons why Python is so good - it has been created and is constantly improved by a community who just want to see a better Python.

4. High-level Language

When you write programs in Python, you never need to bother about the low-level details such as managing the memory used by your program, etc.

5. Portable

Due to its open-source nature, Python has been ported to (i.e.changed to make it work on) many platforms. All your Python programs can work on any of these platforms without requiring any changes at all if you are careful enough to avoid any system-dependent features. You can use Python on GNU/Linux, Windows, FreeBSD, Macintosh, Solaris, OS/2, Amiga, AROS, AS/400, BeOS, OS/390, z/OS, PalmOS, QNX, VMS, Psion, AcornRISCOS,
VxWorks, PlayStation, Sharp Zaurus, Windows CE and Pocket PC! You can even use a platform like Kivy to create games for your computer *and* for iPhone, iPad, and Android.

6. Interpreted

This requires a bit of explanation.A program written in a compiled language like C or C++ is converted from the source language i.e. C or C++ into a language that is spoken by your computer (binary code i.e. 0sand 1s) using a compiler with various flags and options. When you run the program, the linker / loader software copies the program from harddisk to memory and starts running it.

Python, on the other hand, does not need compilation to binary. You just *run* the program directly from the sourcecode. Internally, Python converts the source code into an intermediate form called bytecodes and then translates this into the native language of your computer and then runs it. All this, actually, makes using Python much easier since you don't have to worry about compiling the program, making sure that the proper libraries are linked and loaded, etc. This also makes your Python programs much more portable, since you can just copy your Python program onto another computer and it just works!

7. ObjectOriented

Python supports procedure-oriented programming as well as object-oriented programming. Inprocedure-oriented languages, the program is built around procedures or functions which are nothing but reusable pieces of programs. In object-oriented languages, the program is built around objects which combined at a and functionality. Python has a very powerful buts implistic way of doing OOP, especially when compared to big languages like C++ or Java.

8. Extensible

If you need a critical piece of code to run very fast or want to have some piece of algorithm not to be open, you can code that part of your program in C or C++ and then use it from your Python program.

9. Embeddable

You can embed Python within your C/C++ programs to give *scripting* capabilities for your program's users.

10.Extensive Libraries

The Python Standard Library is huge indeed. It can help you do various things involving regular expressions, documentation generation, unittesting, threading, databases, webbrowsers, CGI, FTP, email, XML, XML-RPC, HTML, WAVfiles, cryptography, GUI(graphical user interfaces), and other system-dependent stuff. Remember,all this is always available wherever Python is installed. This is called the *Batteries Included* philosophy of Python.

Besides the standard library, there are various other high-quality libraries which you can find at the Python Package Index.

3. SYSTEMDESIGN

3.1 ARCHITECTUREDIAGRAM

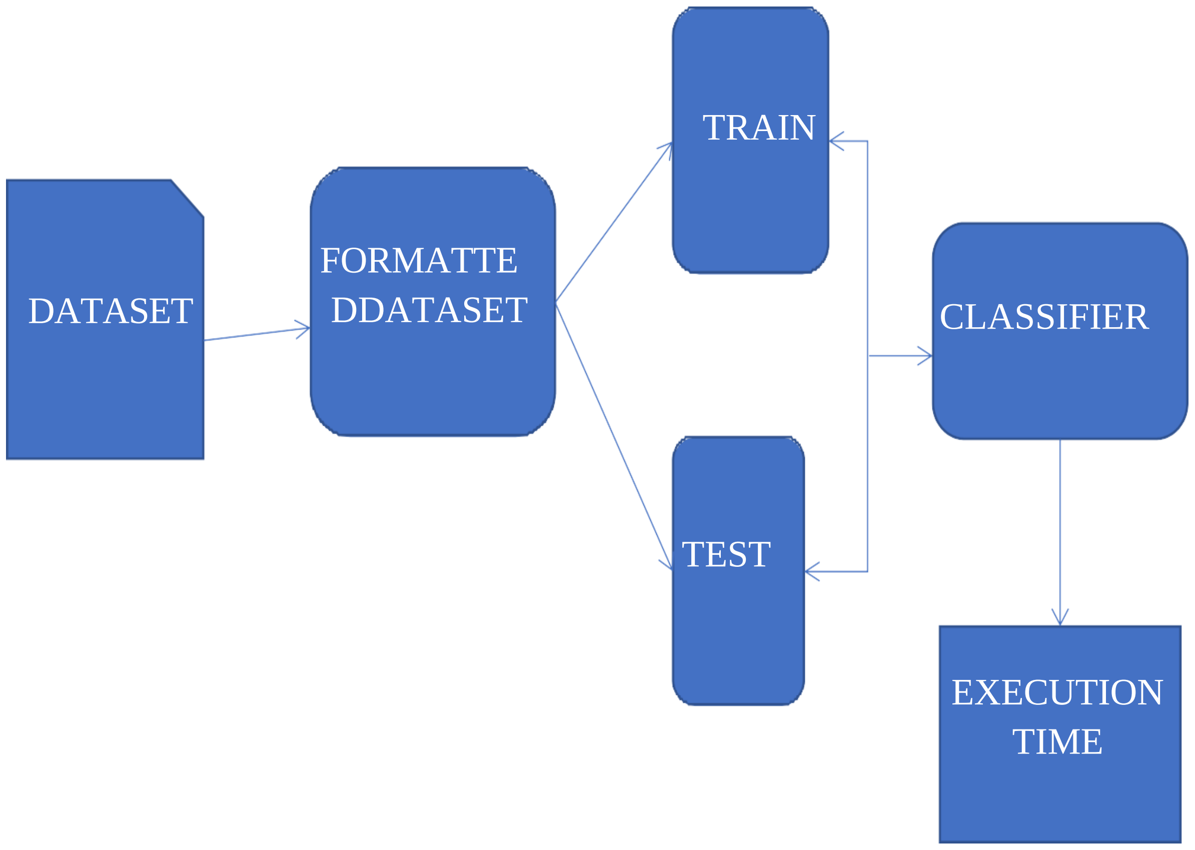
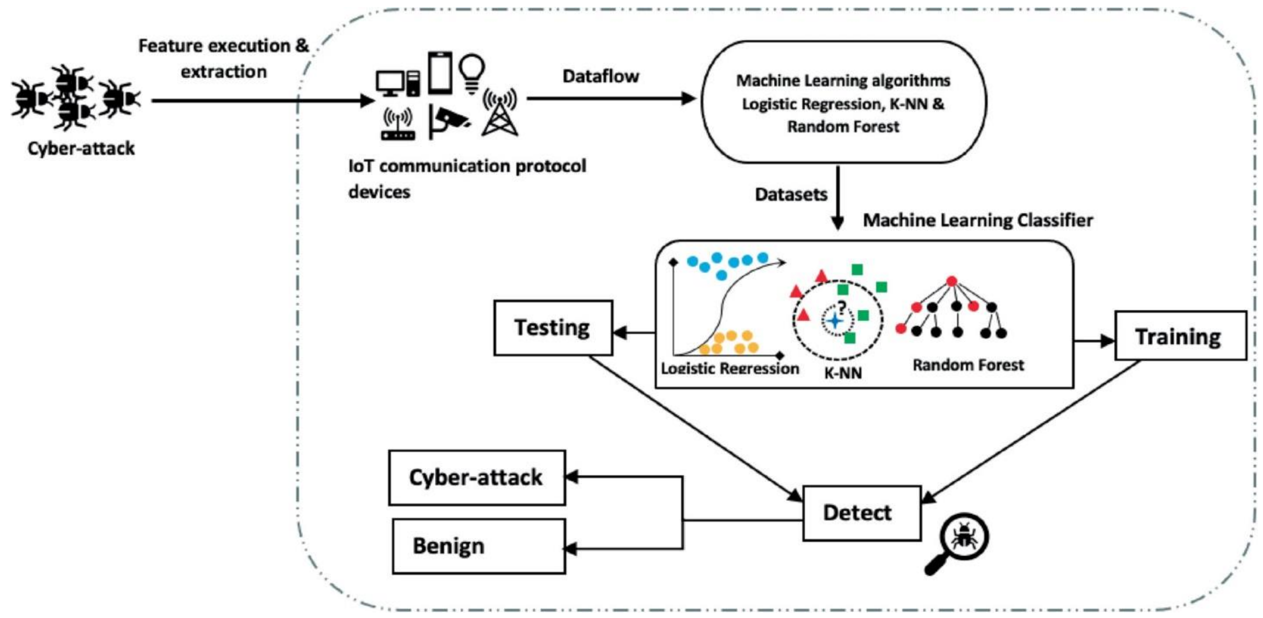


Figure.3.1.ArchitectureDiagram

3.2 UMLDIAGRAM

3.2.1 Dataflow Diagram


Feature execution & extraction
Cyber-attack
IoT communication protocol devices
Dataflow
Machine Learning algorithms Logistic Regression, K-NN & Random Forest
Datasets
Machine Learning Classifier
Testing
Training
Logistic Regression
K-NN
Random Forest
Cyber-attack
Benign
Detect

4 .IMPLEMENTATION

4.1 PROJECTMODULES

In this project there are modules as follows:

- Data Selection and Loading
- Data Preprocessing
- Splitting Data-set into train and test data
- Run SVM classifier
- Run Random Forest classifier
- Run Logistic Regression Classifier
- Run Naïve bayes

4.1.1 Data Selection and Loading

- The term data selection aims at choosing data that should best or edduring data collection or that should be shared/archived after the project is completed.
- In this project, we are using the dataset of previous cyber attacks

4.1.2 Dataset Cleaning

- Using this module we will find out empty values in the dataset and replace with mean or 0 values.

4.1.3 Splitting Data-set into train and test data

- Data splitting is the act of partitioning available data into two portions, usually for cross-validation purposes.
- One Portion of the data is used to develop a predictive model and the other to evaluate the model's performance.
- Separating data into training and testing sets is an important part of evaluating data mining models.
- Typically, when you separate a dataset into a training set and testing set, most of the data is used for training, and a smaller portion of the data is used for testing.

4.1.4 Run SVR Classifier:

Using this module we will train SVM classifier with splitted 80% data and used 20% data to calculate it performance

4.1.5 Run Random Forest Classifier

Using this module we will train Random Forest classifier with splitted 80% data and used 20% data to calculate it performance

4.1.6 Run Logistic Regression Classifier:

Using this module we will train LSTM classifier with splitted 80% data and used 20% data to calculate it performance

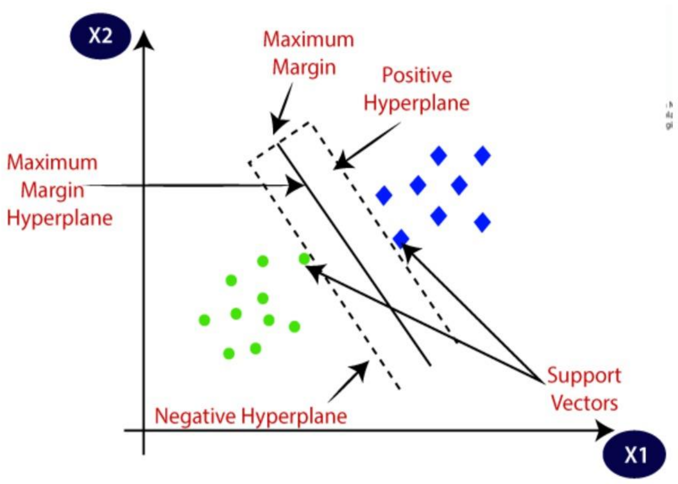
4.1.7 Run Navie bayes:

Using this module we will upload test data and then apply navie bayes classifier to predict the attack

4.2 ALGORITHMS

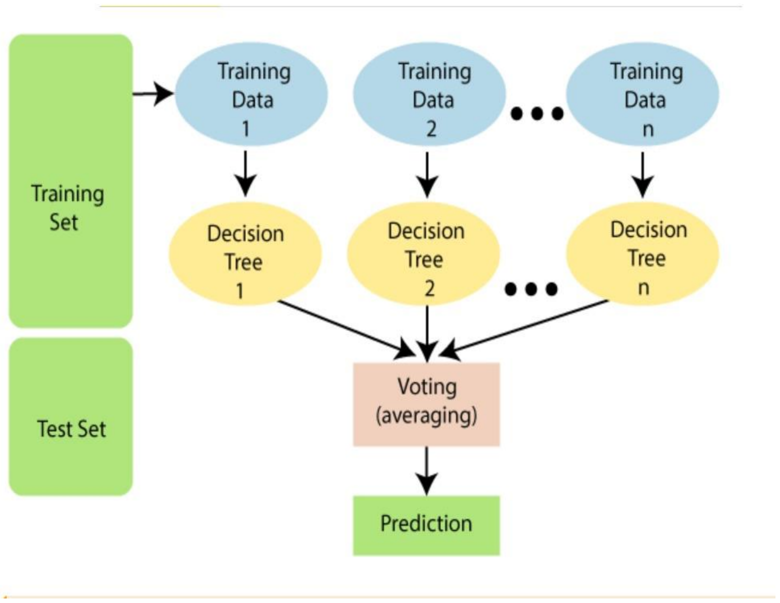
Support vector Machine (SVM)

Support Vector Machine or SVM is one of the most popular Supervised Learning algorithms, which is used for Classification as well as Regression problems. However, primarily, it is used for Classification problems in Machine Learning.The goal of the SVM algorithm is to create the best line or decision boundary that can segregate n-dimensional space into classes so that we can easily put the new data point in the correct category in the future. This best decision boundary is called a hyperplane.



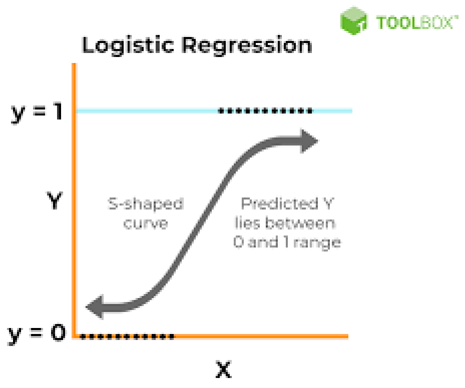
Random forest Regression (RF)

RF belongs to the category of ensemble learning algorithms, having been proposed by Ho in (Ho, 1998). As a base learner of the ensemble, RF uses decision trees. The idea of ensemble learning is that a single predictor is not sufficient for predicting the desired value of test data. The reason being that, based on sample data, a single predictor is not able to distinguish between noise and patterns. RF constructs numerous independent regression trees, a bootstrap sample of the training data is chosen at each regression tree. Therefore, the regression tree continues to grow until it reaches the largest possible size. Whereas, final prediction values a weighted average from predicting all regression trees (Breiman 2001). Random Forest is a popular machine learning algorithm that belongs to the supervised learning technique. It can be used for both Classification and Regression problems in ML. It is based on the concept of **ensemble learning,** which is a process of *combining multiple classifiers to solve a complex problem and to improve the performance of the model.*



Logistic Regression

Logistic regression is one of the most popular Machine Learning algorithms, which comes under the Supervised Learning technique. It is used for predicting the categorical dependent variable using a given set of independent variableLogistic regression predicts the output of a categorical dependent variable. Therefore the outcome must be a categorical or discrete value. It can be either Yes or No, 0 or 1, true or False, etc. but instead of giving the exact value as 0 and 1, **it gives the probabilistic values which lie between 0 and 1**

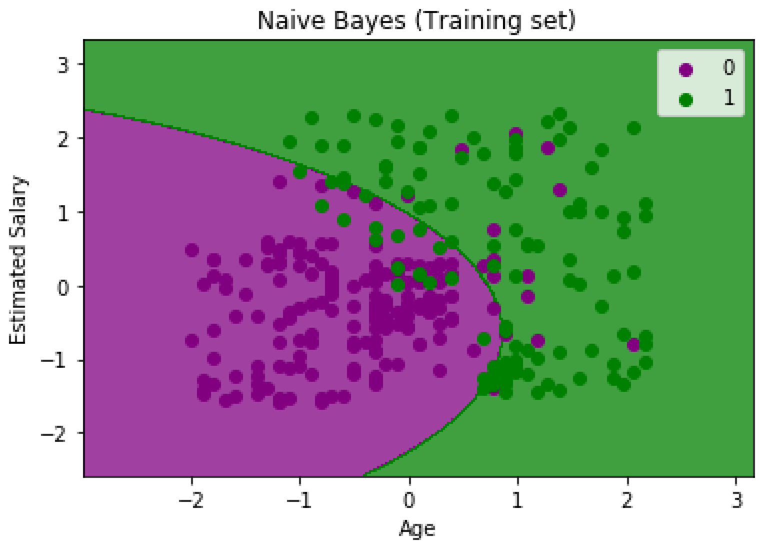


Naïve bayes

Naive Bayes algorithm is a supervised learning algorithm, which is based on **Bayes theorem** and used for solving classification problems.

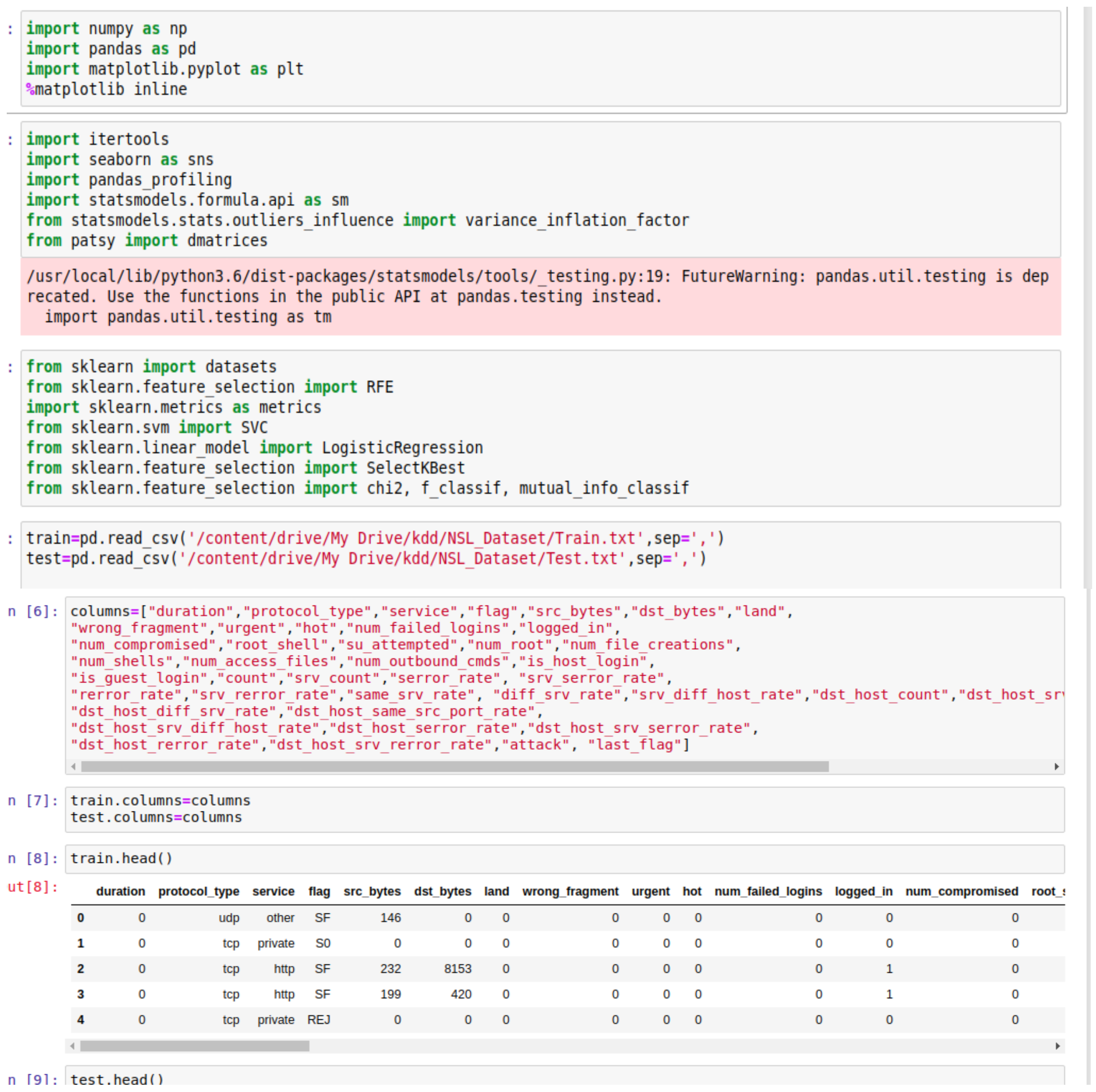
- It is mainly used in *text classification* that includes a high-dimensional training dataset.
- Naïve Bayes Classifier is one of the simple and most effective Classification algorithms which helps in building the fast machine learning models that can make quick predictions.
- **It is a probabilistic classifier, which means it predicts on the basis of the probability of an object**.

Some popular examples of Naïve Bayes Algorithm are **spam filtration, Sentimental analysis, and classifying articles**.



.

4.3 SAMPLECODE



```
: 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=',')
```

```
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
```

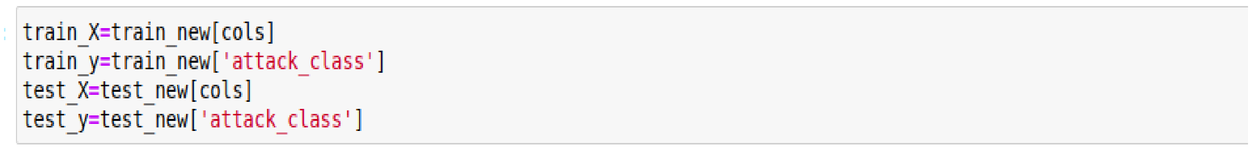
```
n [8]: train.head()
```

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	

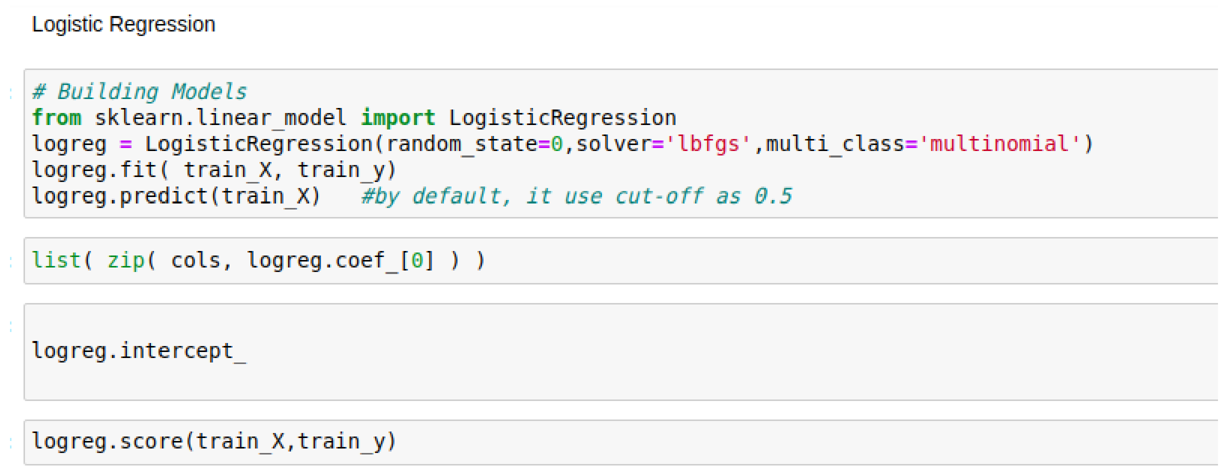
```
n [9]: test.head()
```

Model Building



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

Machine learning Deploy



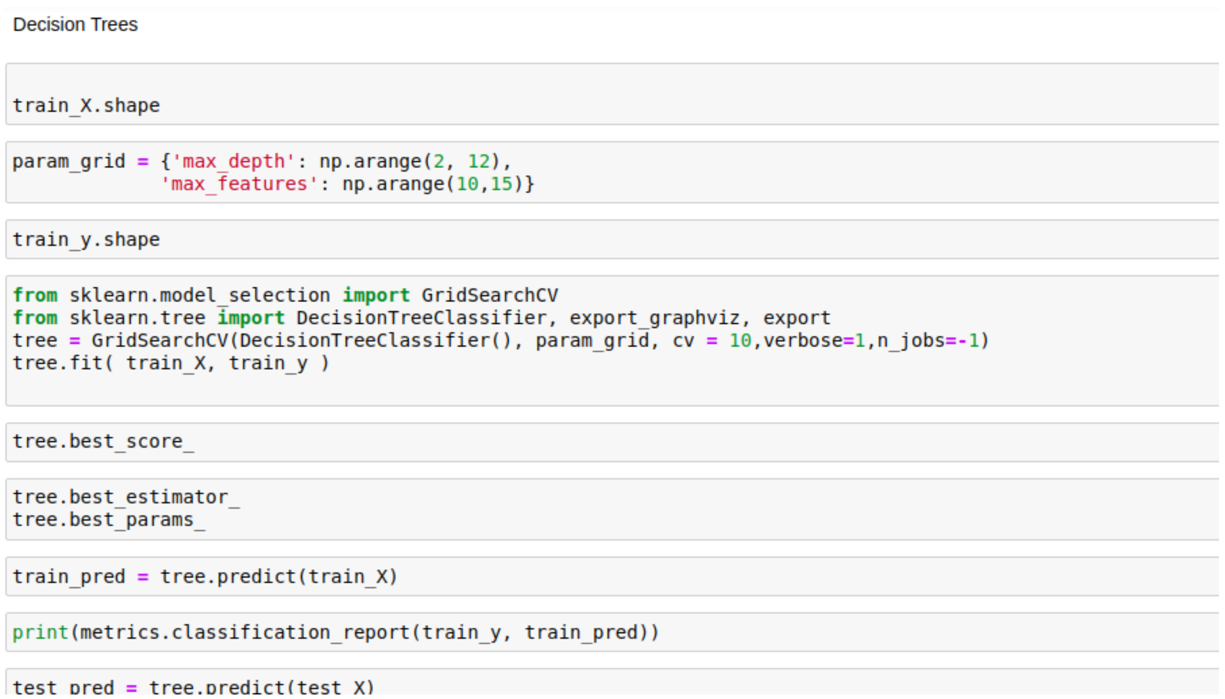
Logistic Regression

```
# 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 )
```

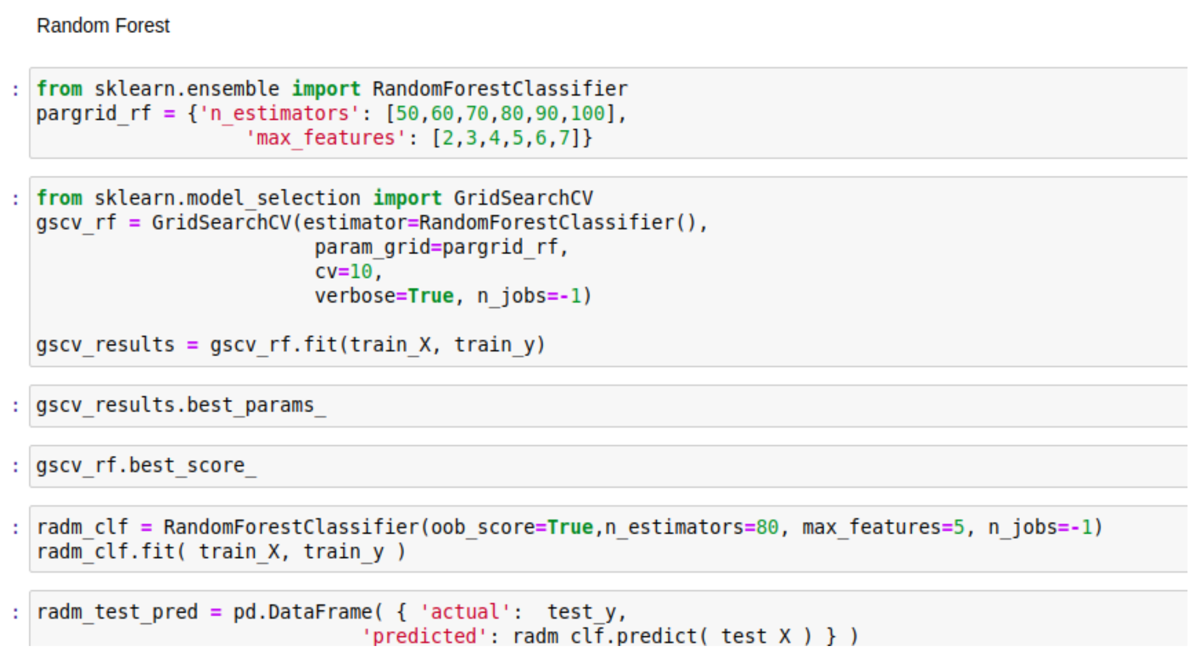
```
tree.best_score_
```

```
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]}
```

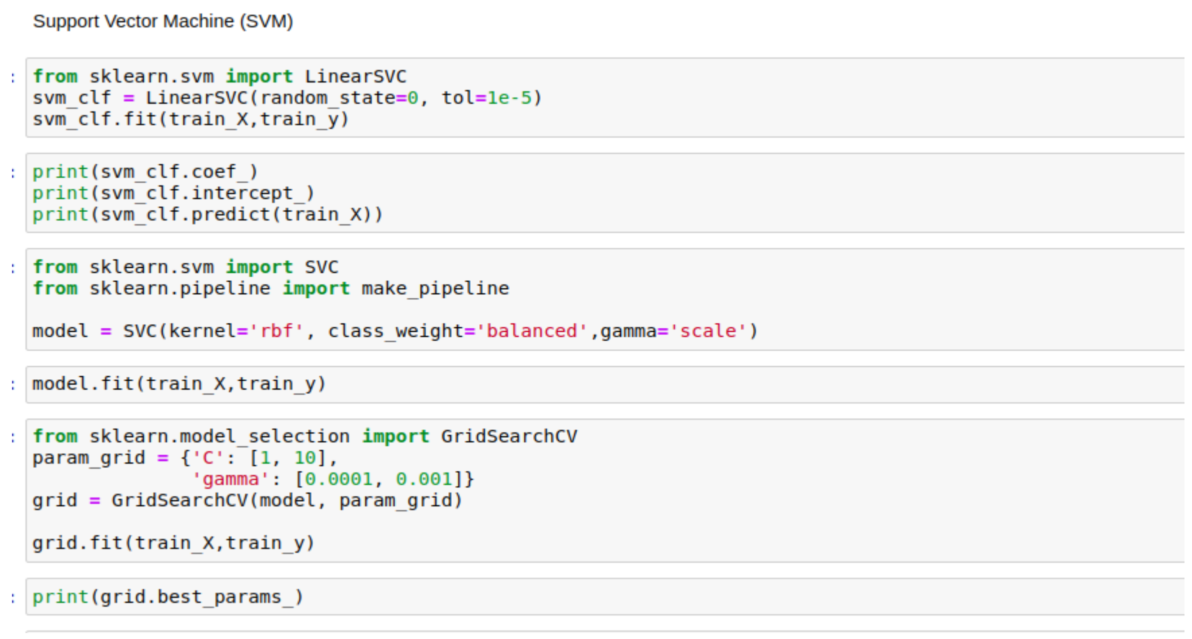
```
from sklearn.model_selection import GridSearchCV
gscv_rf = GridSearchCV(estimator=RandomForestClassifier(),
                        param_grid=pargrid_rf,
                        cv=10,
                        verbose=True, n_jobs=-1)

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_)
```

4. TESTING

4.1Testingmethods

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, sub assemblies, 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 test. Each test type addresses a specific testing requirement.

TYPES OF TESTS

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.

Integration testing

Integration tests are designed to test integrated software components to determine if they actually run as one program. Testing is event driven and is more concerned with the basic outcome of screens or fields. Integration tests demonstrate that although the components were individually satisfaction, 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.

Functional test

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. In addition, systematic coverage pertaining to identify Business process flows; data fields, predefined processes, and successive processes must be considered for testing. Before functional testing is complete, additional tests are identified and the effective value of current tests is determined.

System Test

System testing ensures that the entire integrated software system meets requirements. It tests a configuration to ensure known and predictable results. An example of system testing is the configuration oriented system integration test. System testing is based on process descriptions and flows, emphasizing pre-driven process links and integration points.

White Box Testing

White Box Testing is a testing in which in which the software tester has knowledge of the inner workings, structure and language of the software, or at least its purpose. It is purpose. It is used to test areas that cannot be reached from a black box level.

Black Box Testing

Black Box Testing is testing the software without any knowledge of the inner workings, structure or language of the module being tested. Black box tests, as most other kinds of tests, must be written from a definitive source document, such as specification or requirements document, such as specification or requirements document. It is a testing in which the software under test is treated, as a black box .you cannot "see" into it. The test provides inputs and responds to outputs without considering how the software works.

Unit Testing

Unit testing is usually conducted as part of a combined code and unit test phase of the software lifecycle, although it is not uncommon for coding and unit testing to be conducted as two distinct phases.

Test strategy and approach

Field testing will be performed manually and functional tests will be written in detail.

Test objectives

- All field entries must work properly.
- Pages must be activated from the identified link.
- The entry screen, messages and responses must not be delayed.

Features to be tested

- Verify that the entries are of the correct format
- No duplicate entries should be allowed
- All links should take the user to the correct page.

Integration Testing:

Software integration testing is the incremental integration testing of two or more integrated software components on a single platform to produce failures caused by interface defects.
The task of the integration test is to check that components or software applications, e.g. components in a software system or – one step up – software applications at the company level – interact without error.

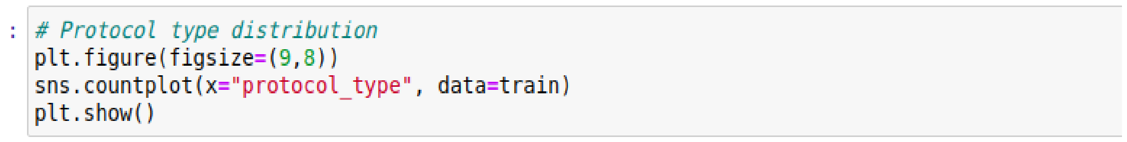
Test Results: All the test cases mentioned above passed successfully. No defects encountered.

Acceptance Testing:

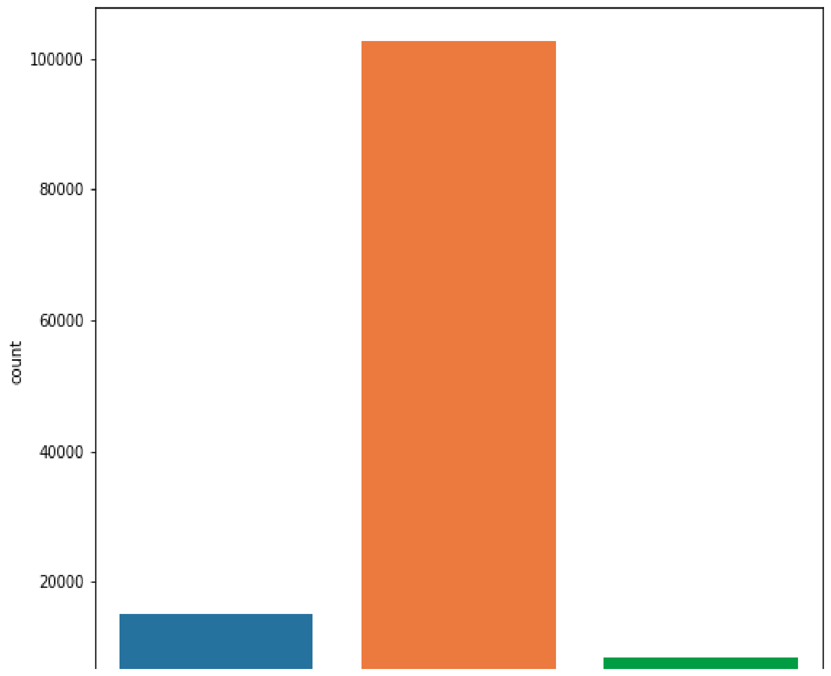
User Acceptance Testing is a critical phase of any project and requires significant participation by the end user. It also ensures that the system meets the functional requirements.

Test Results: All the test cases mentioned above passed successfully. No defects encountered.

5.Results

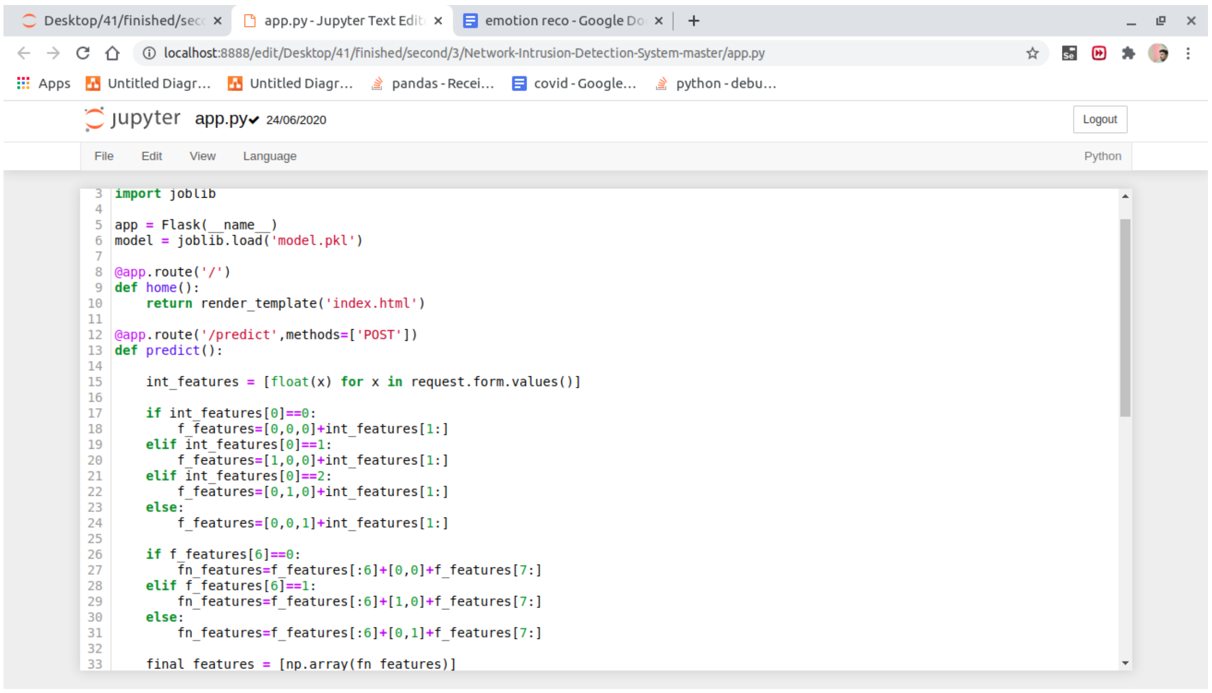


```
# Protocol type distribution
plt.figure(figsize=(9,8))
sns.countplot(x="protocol_type", data=train)
plt.show()
```



Screen shots

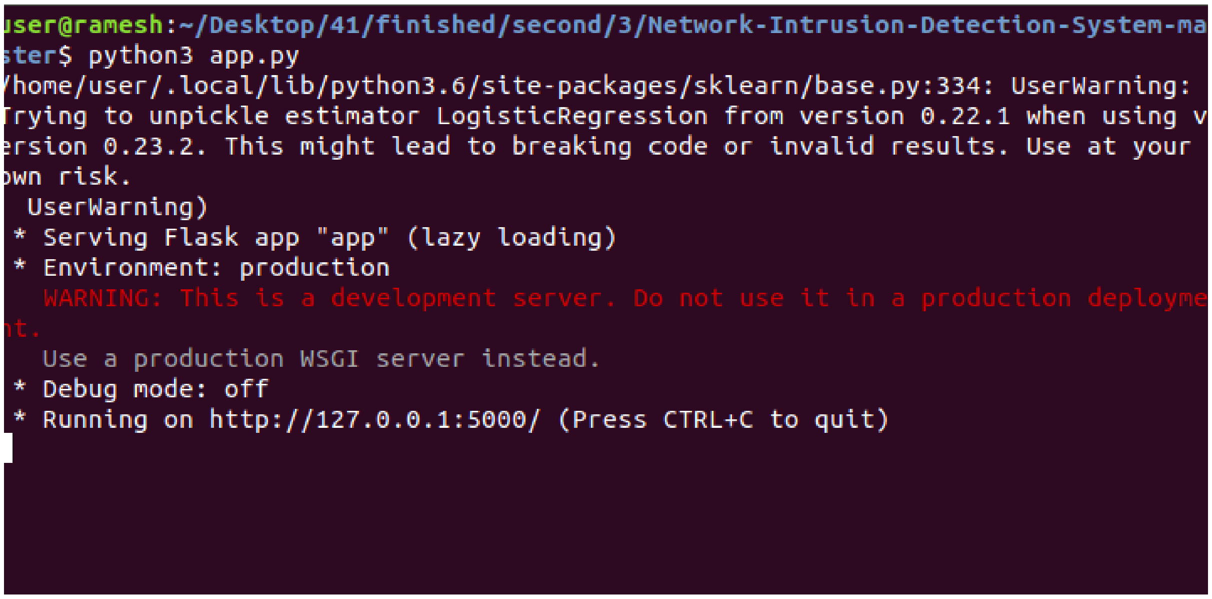
From the score accuracy we concluding the DT & RF give better accuracy and building pickle file for predicting the user input



```
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)]
```

Figure 5.1

Localhost - in cmd python app.py

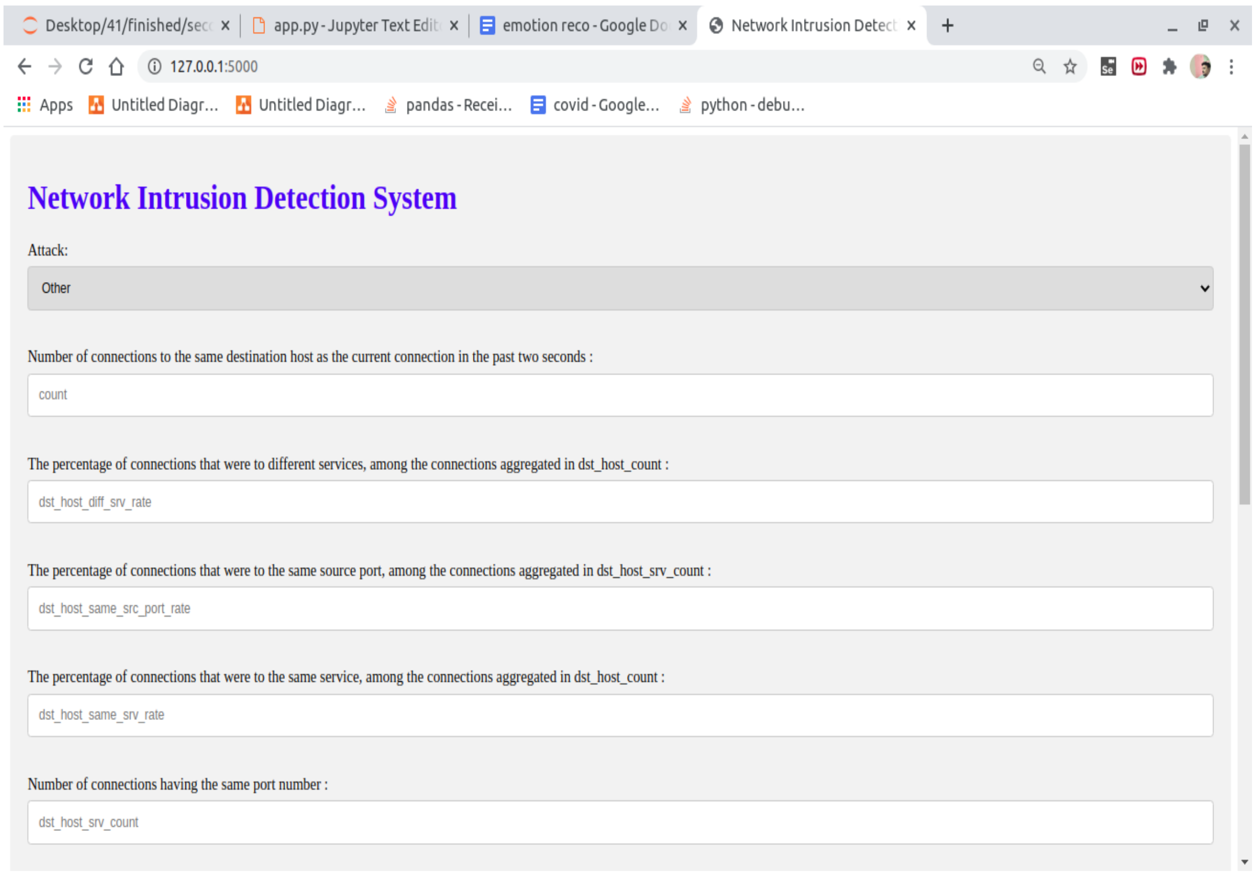


```
user@ramesh:~/Desktop/41/finished/second/3/Network-Intrusion-Detection-System-ma
ster$ python3 app.py
/home/user/.local/lib/python3.6/site-packages/sklearn/base.py:334: UserWarning:
Trying to unpickle estimator LogisticRegression from version 0.22.1 when using v
ersion 0.23.2. This might lead to 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)
```

Figure5.2

In above screen we can see dataset loaded and dataset contains total 4028 records. Now click on 'Dataset Preprocess, Clean & Train Test Split' button to clean dataset and to split dataset into train and test part

Figure6.3



Network Intrusion Detection System

Attack:
Other

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

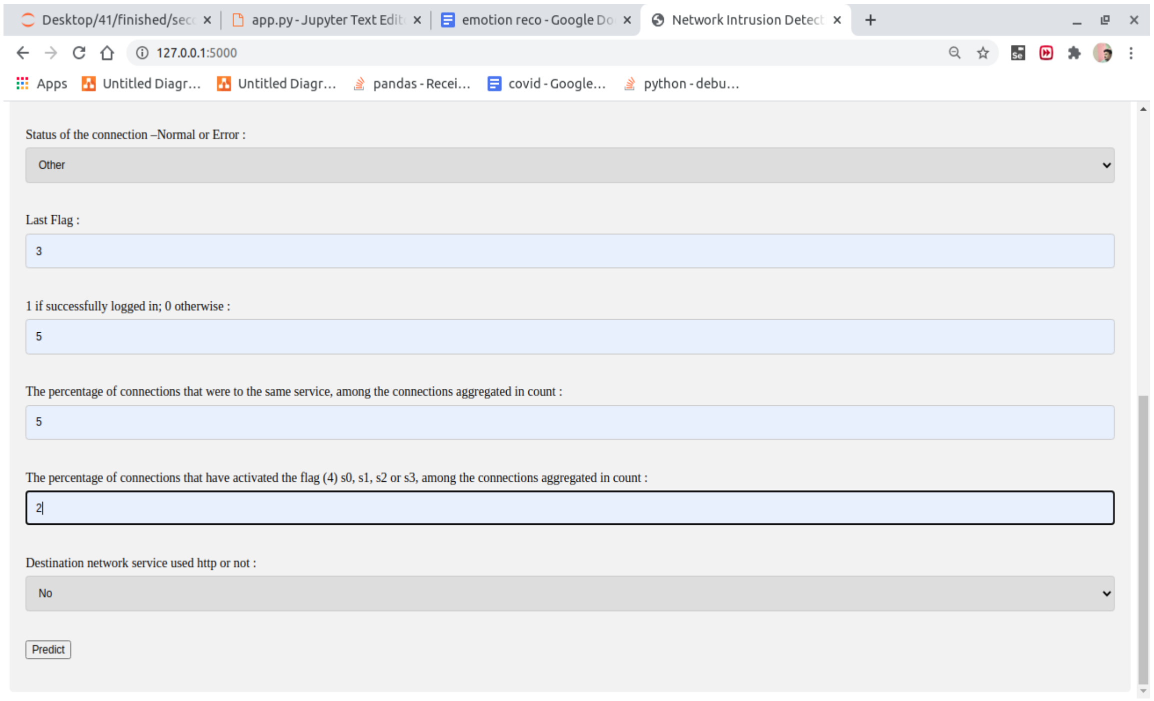
The percentage of connections that were to different services, among the connections aggregated in dst_host_count :
dst_host_diff_srv_rate

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

Figure 5.3



Status of the connection –Normal or Error :
Other

Last Flag :
3

1 if successfully logged in; 0 otherwise :
5

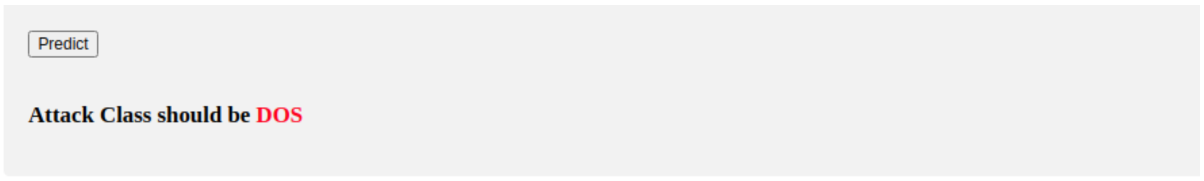
The percentage of connections that were to the same service, among the connections aggregated in count :
5

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

Figure5.4



Attack Class should be DOS

Figure5.5

6.EXPLANATION

This is a Python Flask web application that serves as an interface for a machine learning model. The model is used to classify network traffic into different categories like 'Normal,' 'DOS' (Denial of Service), 'PROBE,' 'R2L' (Unauthorized access from a remote machine), and 'U2R' (Unauthorized access to root).
Let's break down the code step by step:
1. Importing necessary libraries:
   - `numpy` is imported for numerical operations.
   - `Flask` is imported to create a web application.
   - `joblib` is imported to load a pre-trained machine learning model.
   - Other necessary modules and functions are imported for Flask operations.
2. Creating a Flask application:
   - An instance of the Flask class is created and named `app`.
3. Loading a pre-trained machine learning model:
   - The `joblib.load` function is used to load a pre-trained machine learning model from a file named 'model.pkl.'
4. Defining routes:
   - Two routes are defined in the Flask application:
     - `'/'` is associated with the `home` function, which renders an HTML template called 'index.html.'
     - `'/predict'` is associated with the `predict` function, which handles POST requests for making predictions.
     - `'/results'` is associated with the `results` function, which handles POST requests for returning prediction results in JSON format.
5. The `home` function:
   - This function simply renders the 'index.html' template when the root URL `'/'` is accessed.
6. The `predict` function:
   - This function is responsible for processing user input data, making predictions using the loaded machine learning model, and rendering the 'index.html' template with the prediction result.
   - It first extracts input features from the POST request data.
   - It then modifies these features based on conditional statements, which appear to be related to feature engineering.
   - The modified features are used to make predictions using the machine learning model.
   - The predicted class is converted into a human-readable output and passed to the 'index.html'

template.

7. The `results` function:
   - This function is responsible for processing JSON data sent via a POST request, making predictions using the loaded machine learning model, and returning the prediction result in JSON format.
   - It extracts input features from the JSON data, makes predictions, and converts the predicted class into a human-readable output.
   - The output is then returned as a JSON response.
8. Starting the Flask application:
   - The script checks if it is the main module and runs the Flask application using `app.run()` when executed directly.

In summary, this Flask application serves as a user interface for a machine learning model that classifies network traffic. Users can access the web interface, input data, and get predictions for the network traffic category, which is displayed in the 'index.html' template. Additionally, there is an API endpoint (`/results`) for making predictions programmatically and receiving results in JSON format.

7. CONCLUSION

By using this machine learning algorithms the cyber attacks detection has made more easier when compared to the existing systems. The system is implemented by using ANACONDA software , Anaconda is the world's most popular data science platform and the foundation of modern machine learning. We pioneered the use of Python for data science, champion its vibrant community, and continue to steward open-source projects that make tomorrow's innovations possible. Our enterprise-grade solutions enable corporate, research, and academic institutions around the world to harness the power of opensource for competitive advantage, ground breaking research, and a better world.

8.REFERENCES

1. V. Ambalavanan, “Cyber threats detection and mitigation using machine learning,” in Handbook of Research on Machine and Deep Learning Applications for Cyber Security. Hershey, PA, USA: IGI Global, 2020, pp. 132-149.

2. T. Thomas, A. P. Vijayaraghavan, and S. Emmanuel, “Machine learning and cybersecurity,” in Machine Learning Approaches in Cyber Security Analytics. Singapore: Springer, 2020, pp. 37-47. [Online]. Available: https://link.springer.com/chapter/ 10.1007/978-981-15-1706-83.

3. F. Farahmand, S. B. Navathe, P. H. Enslow, and G. P. Sharp, “Managing vulnerabilities of information systems to security incidents,” in Proc. 5th Int. Conf. Electron. Commerce (ICEC), 2003, pp. 348-354.

4. P. Szor, The Art of Computer Virus Research and Defense: ART COMP VIRUS RES DEFENSE p1. London, U.K.: Pearson, 2005. I. Firdausi, C. Lim, A. Erwin, and A. S. Nugroho, “Analysis of machine learning techniques used in behaviorbased malware detection,” in Proc.2nd Int. Conf. Adv.

5. A. L. Buczak and E. Guven, “A survey of data mining and machine learning methods for cyber security intrusion detection,” IEEE Commun. Surveys Tuts. vol. 18, no. 2, pp. 1153-1176, 2nd Quart., 2016.

6. J. M. Torres, C. I. Comesa˜na, and P. J. Garc´ıa-Nieto, “Machine learn- ing techniques applied to cybersecurity,” Int. J. Mach. Learn. Cybern., vol. 10, no. 10, pp. 2823-2836, 2019. Difference Between Threat and Attack. Accessed: Jun. 3, 2020. [Online]. Available: https://www.geeksforgeeks.org/difference-betweenthreat-and-attack/

7. S. Purkait, “Phishing counter measures and their effectiveness-literature review,” Inf. Manage. Comput. Secur., vol. 20, no. 5, pp. 382-420,Nov. 2012.

8. R. M. Mohammad, F. Thabtah, and L. McCluskey, “Tutorial and critical analysis of phishing Websites methods,” Comput. Sci. Rev., vol. 17, pp. 1-24, Aug. 2015.

9. E. H. Spafford, “Computer viruses as artificial life,” Artif. Life, vol. 1, no. 3, pp. 249-265, Apr. 1994. H. Drucker, D. Wu, and V. N. Vapnik, “Support vector machines for

spam categorization," IEEE Trans. Neural Netw., vol. 10, no. 5, pp. 1048-1054, Sep. 1999.

10. N. Jindal and B. Liu, "Review spam detection," in Proc. 16th Int. Conf. World Wide Web, 2007, pp. 1189-1190. S. M. Abdulhamid, M. S. Abd Latiff, H. Chiroma, O. Osho, G. Abdul-Salaam, A. I. Abubakar, and T. Herawan, "A review on mobile SMS spam filtering techniques," IEEE Access, vol. 5, pp. 15650-15666, 2017.

11. D. D. Arifin and M. A. Bijaksana, "Enhancing spam detection on mobile phone short message service (SMS) performance using FP-growth and naive Bayes classifier," in Proc. IEEE Asia Pacific Conf. Wireless Mobile (APWiMob), Sep. 2016, pp. 80-84. Kharon Malware Dataset. Accessed: Aug. 8, 2020. [Online]. Available: http://kharon.gforge.inria.fr/dataset/

12. D. Michie, D. J. Spiegelhalter, and C. Taylor, "Machine learning," Neural Stat. Classification, vol. 13, 1994. S. Dua and X. Du, Data Mining and Machine Learning in Cybersecurity. New York, NY, USA: Auerbach, 2016.

13. S. Angra and S. Ahuja, "Machine learning and its applications: A review," in Proc. Int. Conf. Big Data Anal. Comput. Intell. (ICBDAC), 2017, pp. 57-60.

14. T. M. Alam, K. Shaukat, M. Mushtaq, Y. Ali, M. Khushi, S. Luo, and A. Wahab, "Corporate bankruptcy prediction: An approach towards bet- ter corporate world," Comput. J., pp. 1-16, Jun. 2020.