



# VIT<sup>®</sup>

**Vellore Institute of Technology**  
(Deemed to be University under section 3 of UGC Act, 1956)

## INFORMATION SECURITY ANALYSIS AND AUDIT

CSE3052 – F2 SLOT

TEAM MEMBER	REGISTRATION NUMBER
KUNAL KWATRA	19BCE1784
<u>TOPIC</u> DETECTION OF DDOS ATTACK USING SUPERVISED MACHINE LEARNING	

# INDEX

S. No.	TOPIC	PAGE
1.	Introduction	1
	Scope	2
	Novelty	3-4
	Related Work	5-6
	Dataset	6-7
	Expected Result	7
2.	Flow of Work	7
	Architecture	8-9
3.	Algorithms Used	9
	Implementation	10-37
4.	Comparison of algorithms	38
	Results	38
5.	Inference	38
	References	39

# INTRODUCTION

DDoS(Distributed Denial Of Service) is an attempt to stop the working of a network or service or server by overwhelming the targeted network with the flood of internet traffic.

DDoS assaults accomplish viability by using numerous compromised PC frameworks as wellsprings of assault traffic. Taken advantage of machines can incorporate PCs and other arranged assets like IoT gadgets.

These attacks are done using the network of bots (networked devices infected with malware) called the botnet. Once the attacker gets the access to the botnet, he can flood the network of target using the IP address of the systems in botnet, resulting in interruption in the working of the victim system/network/server.

And since each bot has a legitimate IP address, it becomes difficult to differentiate between an attack and a normal access.

This project aims at recognition of the ddos attack in the network flow by analysing various attributes of the incoming network. For the classification of the network various classification algorithms are being used and their results are analysed on the basis of the difference in their accuracy.

The raw dataset used in the dataset consists of 41 attributes, out of which 3 has categorical data and others have floating data. The dataset is obtained from Kaggle and is tempted or modified.

For the model training all the attributed are taken into account, and due to dataset imbalance, it is passed over the smote technique to balance the dataset.

The processed dataset is used to train and evaluate different algorithms, namely Naïve Bayes Classifier, Decision Tree Algorithm, XGBoost algorithm, Random Forest Algorithm and a self-trained Hybrid Algorithm.

The classification result of the model is presented as binary, i.e., 0 and 1.

# PROJECT SCOPE STATEMENT

- **PROJECT OBJECTIVE**

To develop a system to detect DDOS attack using Supervised Machine Learning with a good accuracy.

- **DELIVERABLES**

1. Different ML models trained using different algorithms to detect DDOS attack
2. To provide a comparison between the algorithms using their accuracy score.

- **MILESTONES**

Review1: Declaration of aim and achievable target of project.

Review2: Implementation of 4 algorithms without pre-processing.

Review3: Implementation of all algorithms along with pre-processing of dataset and comparison of algorithms.

- **TECHNICAL REQUIREMENTS**

1. Interactive python IDE for implementation.
2. A normal PC to run the models.
3. Basic ML libraries pre-installed in system.

- **Limitations**

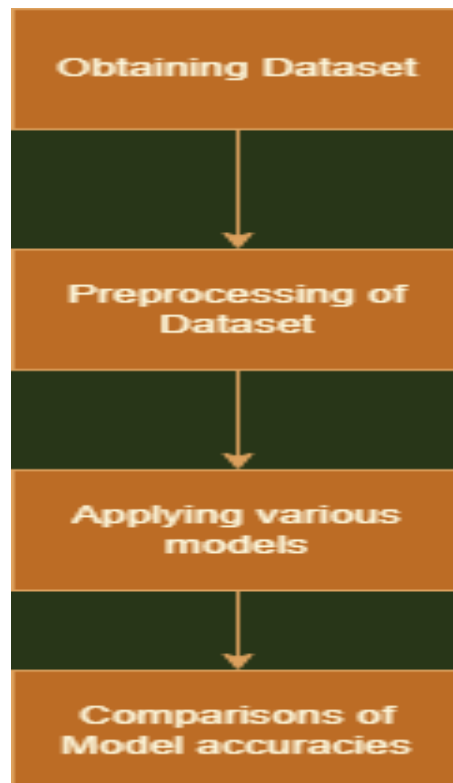
1. Training of models can be limited due to less capability of PC.
2. Slow execution of model training due to heavy operations.

# NOVELTY

The presented work is all done by me and haven't been copied from anywhere.

Implementation of all the already available algorithms is done after pre-processing of the dataset.

The methodology followed for implementation of every algorithm is:



The pre-processing performed includes :

- Removing 6 attributes
- One hot Encoding for categorical attributes
- Normalization of values
- Conversion of class labels to binary
- SMOTE To handle imbalanced Data

After pre-processing, 4 already available algorithms (namely Naïve Bayes Classifier, Decision Tree Algorithm, XGBoost algorithm, Random Forest

Algorithm) along with a self-trained Hybrid algorithm are applied on the Dataset.

Finally, the results of different algorithms are compared using their accuracy attained after evaluation.

## Related Work

In [1], The authors proposed a methodology for the improvement of a generalized ML-based model for the detection of DDoS attacks. After exploring a range of attributes of the dataset chosen for this study, they suggested an integrated feature selection (IFS) technique which consists of three ranges and integration of two unique methods, that is, filter and embedded techniques to choose features that especially make contributions to the detection of a range of kinds of DDoS attacks. They used a light gradient boosting machine (LGBM) algorithm for training the model for the classification of benign and malicious flows.

In [2], Saikat Das, Deepak Venugopal and Sajjan Shiva suggested ensemble unsupervised ML method to implement an intrusion detection system which has the good accuracy to discover DDoS attacks. The aim of this study is to expand the DDoS attack detection accuracy whilst reducing the false positive rate. The NSL-KDD dataset and twelve characteristic features from current research are used for experimentation to evaluate our ensemble results with those of our individual and different present models.

Authors in [3], addresses the prediction of application-layer DDoS attacks in real-time with exceptional machine mastering models. They utilized the two machine learning strategies Random Forest (RF) and Multi-Layer Perceptron (MLP) thru the Scikit ML library and big data framework for the detection of Distributed Denial of Service (DDoS) attacks. In addition to the detection of DDoS attacks, They optimized the overall performance of the model by way of minimizing the prediction time as in contrast with different available processes using the big data framework. They accomplished a mean accuracy of 99.5% of the models both with and without big data approaches.

In [4], Authors recommend DDoSNet, an intrusion detection system towards DDoS attacks in SDN environments. Their technique is based totally on Deep

Learning (DL) technique, combining the Recurrent Neural Network (RNN) with autoencoder. They considered the model using dataset CICDDoS2019, which includes a complete range of DDoS attacks and addresses the gaps of the present contemporary datasets. They obtained a considerable enhancement in attack detection, as in contrast to different benchmarking methods. Hence, the model offers notable confidence in securing these networks.

In [5], authors proposed a set of new entropy-based features that assist to become aware of DDoS attacks precisely and brought a novel multi classifier system based totally on the proposed set of multiple entropy-based aspects and machine learning classifiers to extend the generality and accuracy of detecting low-intensity and high-intensity DDoS attacks. Experiment results confirmed that approach accomplished greater precision and greater recall values in contrast to countless latest approaches.

## **DATASET**

The raw dataset taken from Kaggle was the DDOS dataset-raw.

Initially it had 41 attributes and a set of class labels.

A variety of pre-processing techniques are applied on the data set including:

- Removing 6 attributes

Attributes with maximum number values as 0s are removed.

- One hot Encoding for categorical attributes

- Normalization of values

To convert floating values to range between 0 & 1.

- Conversion of class labels to binary

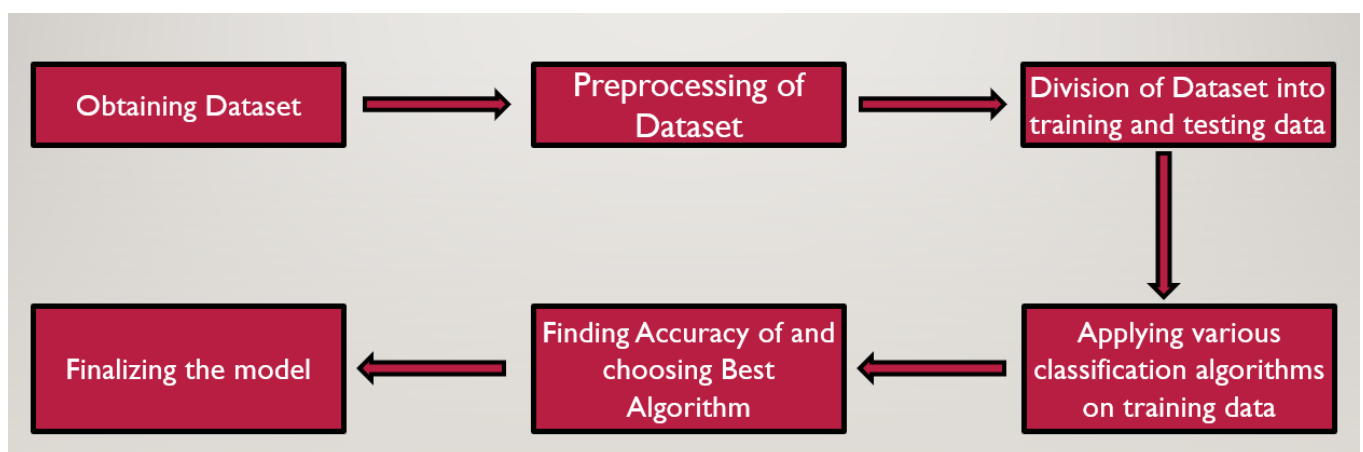
- SMOTE To handle imbalanced Data

After pre-processing, the dataset has 116 features and binary class labels.

## EXPECTED RESULTS

Expected results for the project are to successfully apply all the algorithm and find the one with the best results to detect the DDOS attacks.

## FLOW OF WORK



After importing the dataset and performing all the pre-processing steps.

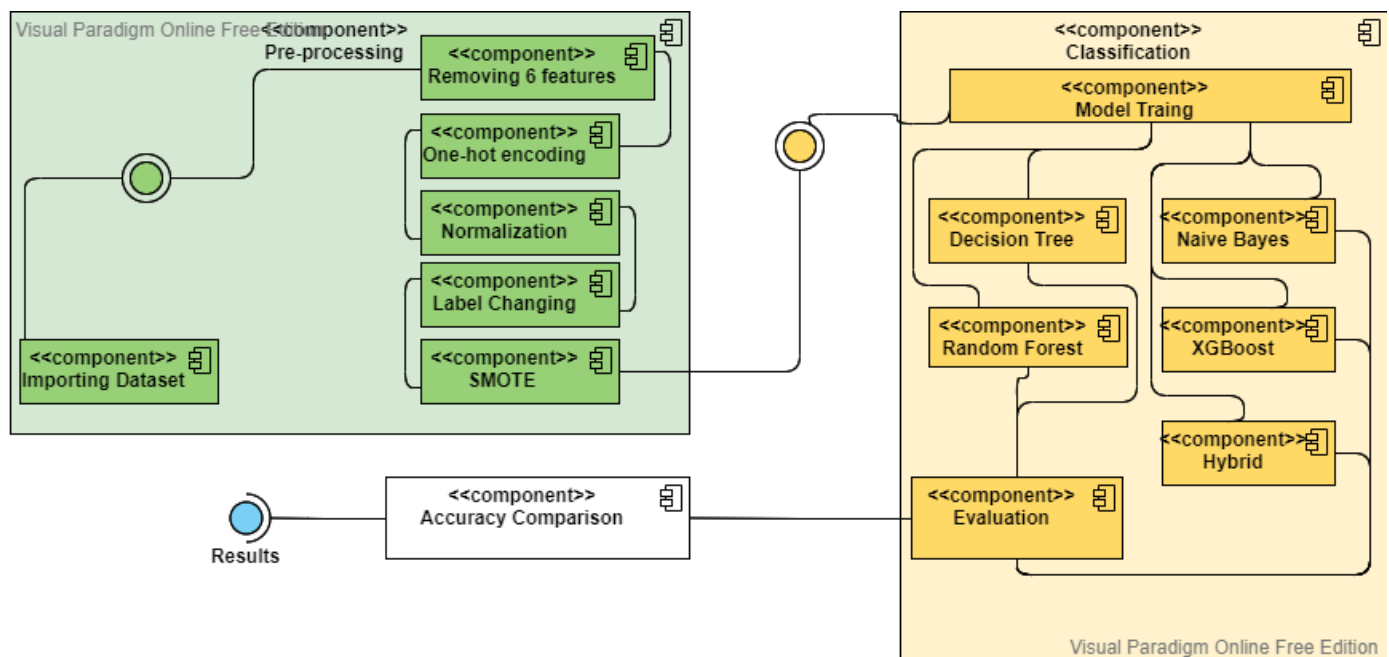
Dataset is divided into training and testing data. Training data is used to fit over the model to make it learn and testing data is used to evaluate the trained model.

Further, various classification algorithms (Decision Tree, Naïve Bayes, Random Forest, XGBoost and self-implemented hybrid algorithm) are used for classification.

And finally, the results of all the models are compared using their accuracy score. The one with the best accuracy score is chosen.



# COMPONENT DIAGRAM



In the pre-processing:

- I removed 6 features with maximum 0s as its values, changing number of attributes from 41 to 35.
- Implemented one-hot encode to change categorical data into binary changing number of features from 35 to 116.
- Normalized the data to standard form using min-max scalar technique.
- Changed the labels from DDOS\_attack\_type/Normal to 0/1 to fit the data into classification model.
- Implemented smote to handle the imbalanced data[Data having a significantly large number of one type of class label out numbering the others.]

In Classification:

I trained 5 different models using the given dataset, namely – Decision Tree, Naïve Bayes, Random Forest, XGBoost, Self-Implemented Hybrid Algorithm.

In Hybrid, I trained the data using Decision Tree, XGBoost, Naïve Bayes and concatenated the prediction of all the algorithms with the testing

data. Then trained the Random Forest Algorithm using the newly formed dataset.

In Evaluation, testing data was fed to the trained models and the achieved accuracies were compared to find the best classification algorithm.

In Result, the comparison of the trained models is presented according to their calculated accuracies.

## **Algorithms Used**

Following Algorithms are used in the project to classify a network record as Normal or DDOS Attack:

- Naïve Bayes
- Decision Tree
- Random Forest
- Xgboost Algorithm
- Hybrid ALgorithm

## ▼ IMPLEMENTATION OF DECISION TREE

### ▼ Importing Libraries

+ Code

+ Text

Double-click (or enter) to edit

```
import pandas as pd
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import train_test_split
from sklearn import metrics
from sklearn.utils import shuffle
from imblearn.over_sampling import SMOTE
```

### ▼ Importing dataset

```
df = pd.read_csv("/content/drive/MyDrive/ISA_Project/dataset/corrected.csv")
df
```

	0	udp	private	SF	105	146	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
0	0	udp	private	SF	105	146	0	0	0	0	0	0	0	0	0
1	0	udp	private	SF	105	146	0	0	0	0	0	0	0	0	0
2	0	udp	private	SF	105	146	0	0	0	0	0	0	0	0	0
3	0	udp	private	SF	105	146	0	0	0	0	0	0	0	0	0
4	0	udp	private	SF	105	146	0	0	0	0	0	0	0	0	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
311023	0	udp	private	SF	105	147	0	0	0	0	0	0	0	0	0
311024	0	udp	private	SF	105	147	0	0	0	0	0	0	0	0	0
311025	0	udp	private	SF	105	147	0	0	0	0	0	0	0	0	0
311026	0	udp	private	SF	105	147	0	0	0	0	0	0	0	0	0
311027	0	udp	private	SF	105	147	0	0	0	0	0	0	0	0	0

311028 rows × 42 columns

```
df.columns = ['Unnamed: 0', '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', 'same_srv_rate', 'diff_srv_rate',
'srv_diff_host_rate', 'una1', 'una2', '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', 'result']
```

df

	Unnamed: 0	protocol_type	service	flag	src_bytes	dst_bytes	land	wrong_fragment
0	0	udp	private	SF	105	146	0	
1	0	udp	private	SF	105	146	0	
2	0	udp	private	SF	105	146	0	
3	0	udp	private	SF	105	146	0	
4	0	udp	private	SF	105	146	0	
...	...	...	...	...	...	...	...	...
311023	0	udp	private	SF	105	147	0	
311024	0	udp	private	SF	105	147	0	
311025	0	udp	private	SF	105	147	0	
311026	0	udp	private	SF	105	147	0	
311027	0	udp	private	SF	105	147	0	

311028 rows × 42 columns

## Removing 6 unnecessary features having 0 as the maximum count of values

```
for col in df.columns:
    if(df[col].nunique() <5):
        print("\n"+col+" : Unique values : " + str(df[col].nunique()) + " : " + str(df[col].un
```

```
protocol_type : Unique values : 3 : ['udp' 'tcp' 'icmp']
```

```
land : Unique values : 2 : [0 1]
```

```
wrong_fragment : Unique values : 3 : [0 1 3]
```

```
urgent : Unique values : 4 : [0 2 1 3]
```

```

logged_in : Unique values : 2 : [0 1]

root_shell : Unique values : 2 : [0 1]

su_attempted : Unique values : 3 : [0 1 2]

num_shells : Unique values : 4 : [0 2 1 5]

num_outbound_cmds : Unique values : 1 : [0]

is_host_login : Unique values : 2 : [0 1]

is_guest_login : Unique values : 2 : [0 1]

```

```
df.drop(["land", "logged_in", "root_shell", "num_outbound_cmds", "is_host_login", "is_gues
df.head()
```

	Unnamed: 0	protocol_type	service	flag	src_bytes	dst_bytes	wrong_fragment	urg
0	0	udp	private	SF	105	146	0	
1	0	udp	private	SF	105	146	0	
2	0	udp	private	SF	105	146	0	
3	0	udp	private	SF	105	146	0	
4	0	udp	private	SF	105	146	0	

```
print(df.shape)
```

```
(311028, 36)
```

## ▼ Applying Onehot encoding to convert object features into integer

```

temp1 = pd.get_dummies(df.protocol_type, prefix='protocol_type')
temp2 = pd.get_dummies(df.service, prefix='service')
temp3 = pd.get_dummies(df.flag, prefix='flag')
df.drop(['protocol_type', 'flag', 'service'], axis='columns', inplace=True)

```

```

i=1
for col in temp1:
    df.insert(i, col, temp1[col])
    i = i+1
for col in temp2:
    df.insert(i, col, temp2[col])
    i = i+1
for col in temp3:
    df.insert(i, col, temp3[col])

```

```
i = i+1
df.head()
```

	Unnamed: 0	protocol_type_icmp	protocol_type_tcp	protocol_type_udp	service_IRC
0	0	0	0	1	0
1	0	0	0	1	0
2	0	0	0	1	0
3	0	0	0	1	0
4	0	0	0	1	0

5 rows × 112 columns

## ▼ Normalizing the data using min-max scaller method

```
df_nor = df.iloc[:, 0:111]
df_nor
```

	Unnamed: 0	protocol_type_icmp	protocol_type_tcp	protocol_type_udp	service_
0	0	0	0	1	
1	0	0	0	1	
2	0	0	0	1	
3	0	0	0	1	
4	0	0	0	1	
...	...	...	...	...	...
311023	0	0	0	1	
311024	0	0	0	1	
311025	0	0	0	1	
311026	0	0	0	1	
311027	0	0	0	1	

311028 rows × 111 columns

```
#Using Min-Max Feature : series = (series-min(series))/(max(series)-min(series))
for col in df_nor.columns:
    df_nor[col] = (df_nor[col] - df_nor[col].min())/(df_nor[col].max() - df_nor[col].min())
df_nor["result"] = df["result"]
df_nor
```

	Unnamed: 0	protocol_type_icmp	protocol_type_tcp	protocol_type_udp	service_
0	0.0	0.0	0.0	1.0	
1	0.0	0.0	0.0	1.0	
2	0.0	0.0	0.0	1.0	
3	0.0	0.0	0.0	1.0	
4	0.0	0.0	0.0	1.0	
...	...	...	...	...	...
311023	0.0	0.0	0.0	1.0	
311024	0.0	0.0	0.0	1.0	
311025	0.0	0.0	0.0	1.0	
311026	0.0	0.0	0.0	1.0	
311027	0.0	0.0	0.0	1.0	

311028 rows × 112 columns

## ▼ Changing class labels to binary

```
for i in range(len(df_nor["result"])):
    if df_nor['result'][i] == "normal.":
        df_nor['result'][i] = 0
    else:
        df_nor['result'][i] = 1
```

/usr/local/lib/python3.7/dist-packages/ipykernel\_launcher.py:3: SettingWithCopyWarning: A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: [https://pandas.pydata.org/pandas-docs/stable/user\\_guide/10min.html#view](https://pandas.pydata.org/pandas-docs/stable/user_guide/10min.html#view)  
This is separate from the ipykernel package so we can avoid doing imports until  
/usr/local/lib/python3.7/dist-packages/ipykernel\_launcher.py:5: SettingWithCopyWarning: A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: [https://pandas.pydata.org/pandas-docs/stable/user\\_guide/10min.html#view](https://pandas.pydata.org/pandas-docs/stable/user_guide/10min.html#view)  
"""



## ▼ Handling Imbalanced data

```
unique_labels = list(df_nor["result"].unique())
print(unique_labels)
```

[0, 1]

```

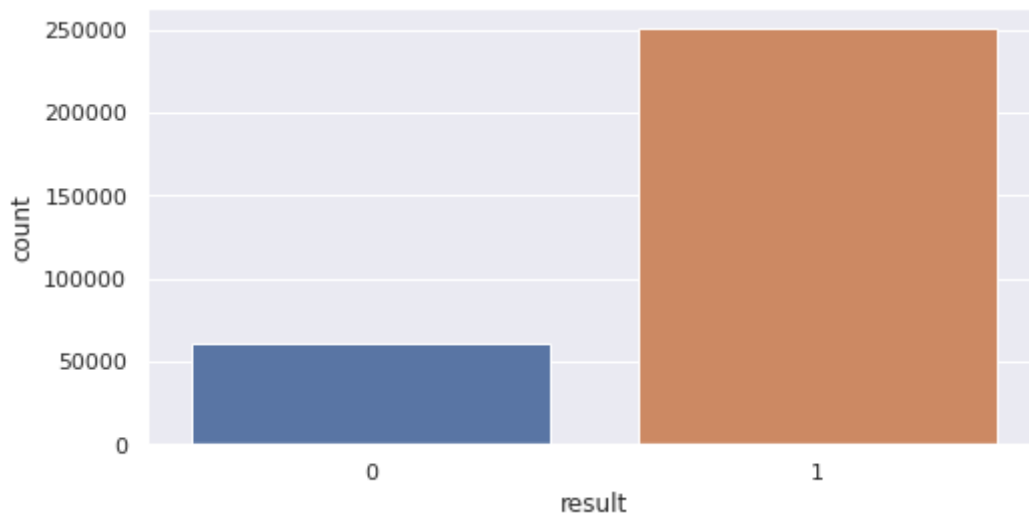
u=dict()
for r in df_nor["result"]:
    if r in u.keys():
        u[r]+=1
    else:
        u[r]=1
print("Count of each class label : \n",u)
import seaborn as sb
sb.set(rc = {'figure.figsize':(8,4)})
sb.countplot(df_nor['result'])

```

```

Count of each class label :
{0: 60592, 1: 250436}
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarning: Pass
FutureWarning
<matplotlib.axes._subplots.AxesSubplot at 0x7f82492df450>

```



```

from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(df_nor.iloc[:, 0:111], df_nor["result"]

```

```

from imblearn.over_sampling import SMOTE
smote = SMOTE()

```

```

X_train_smote, y_train_smote = smote.fit_resample(X_train.astype('float'), y_train.astype(

```

## ▼ Applying Algorithm

```

from sklearn.tree import DecisionTreeClassifier
from sklearn import metrics

```

```

model = DecisionTreeClassifier()
# Train Decision Tree Classifier
model = model.fit(X_train_smote,y_train_smote)

```



```
#Predict the response for test dataset  
y_pred = model.predict(X_test)
```

```
y_pred
```

```
array([1, 0, 1, ..., 1, 1, 1])
```

```
y_test = y_test.to_numpy()  
y_test
```

```
array([1, 0, 1, ..., 1, 1, 1], dtype=object)
```

```
import numpy  
y_test = numpy.array(list(y_test))
```

```
print("Accuracy:",100*metrics.accuracy_score(y_test, y_pred))
```

```
Accuracy: 97.58437021080496
```

## ▼ IMPLEMENTATION OF NAIVE BAYES

### ▼ Importing Libraries

Double-click (or enter) to edit

```
import pandas as pd
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import train_test_split
from sklearn import metrics
from sklearn.utils import shuffle
from imblearn.over_sampling import SMOTE
```

### ▼ Importing dataset

```
df = pd.read_csv("/content/drive/MyDrive/ISA_Project/dataset/corrected.csv")
df
```

	0	udp	private	SF	105	146	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
0	0	udp	private	SF	105	146	0	0	0	0	0	0	0	0	0
1	0	udp	private	SF	105	146	0	0	0	0	0	0	0	0	0
2	0	udp	private	SF	105	146	0	0	0	0	0	0	0	0	0
3	0	udp	private	SF	105	146	0	0	0	0	0	0	0	0	0
4	0	udp	private	SF	105	146	0	0	0	0	0	0	0	0	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
311023	0	udp	private	SF	105	147	0	0	0	0	0	0	0	0	0
311024	0	udp	private	SF	105	147	0	0	0	0	0	0	0	0	0
311025	0	udp	private	SF	105	147	0	0	0	0	0	0	0	0	0
311026	0	udp	private	SF	105	147	0	0	0	0	0	0	0	0	0
311027	0	udp	private	SF	105	147	0	0	0	0	0	0	0	0	0

311028 rows × 42 columns

```
df.columns = ['Unnamed: 0', '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', 'same_srv_rate', 'diff_srv_rate',
'srv_diff_host_rate', 'una1', 'una2', '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', 'result']
```

df

	Unnamed: 0	protocol_type	service	flag	src_bytes	dst_bytes	land	wrong_fragment
0	0	udp	private	SF	105	146	0	
1	0	udp	private	SF	105	146	0	
2	0	udp	private	SF	105	146	0	
3	0	udp	private	SF	105	146	0	
4	0	udp	private	SF	105	146	0	
...	...	...	...	...	...	...	...	...
311023	0	udp	private	SF	105	147	0	
311024	0	udp	private	SF	105	147	0	
311025	0	udp	private	SF	105	147	0	
311026	0	udp	private	SF	105	147	0	
311027	0	udp	private	SF	105	147	0	

311028 rows × 42 columns

## Removing 6 unnecessary features having 0 as the maximum count of values

```
for col in df.columns:
    if(df[col].nunique() <5):
        print("\n"+col+" : Unique values : " + str(df[col].nunique()) + " : " + str(df[col].un

protocol_type : Unique values : 3 : ['udp' 'tcp' 'icmp']

land : Unique values : 2 : [0 1]

wrong_fragment : Unique values : 3 : [0 1 3]

urgent : Unique values : 4 : [0 2 1 3]
```

```

logged_in : Unique values : 2 : [0 1]

root_shell : Unique values : 2 : [0 1]

su_attempted : Unique values : 3 : [0 1 2]

num_shells : Unique values : 4 : [0 2 1 5]

num_outbound_cmds : Unique values : 1 : [0]

is_host_login : Unique values : 2 : [0 1]

is_guest_login : Unique values : 2 : [0 1]

```

```

df.drop(["land", "logged_in", "root_shell", "num_outbound_cmds", "is_host_login", "is_gues
df.head()

```

	Unnamed: 0	protocol_type	service	flag	src_bytes	dst_bytes	wrong_fragment	urg
0	0	udp	private	SF	105	146	0	
1	0	udp	private	SF	105	146	0	
2	0	udp	private	SF	105	146	0	
3	0	udp	private	SF	105	146	0	
4	0	udp	private	SF	105	146	0	

```
print(df.shape)
```

```
(311028, 36)
```

## ▼ Applying Onehot encoding to convert object features into integer

```

temp1 = pd.get_dummies(df.protocol_type, prefix='protocol_type')
temp2 = pd.get_dummies(df.service, prefix='service')
temp3 = pd.get_dummies(df.flag, prefix='flag')
df.drop(['protocol_type', 'flag', 'service'], axis='columns', inplace=True)

```

```

i=1
for col in temp1:
    df.insert(i, col, temp1[col])
    i = i+1
for col in temp2:
    df.insert(i, col, temp2[col])
    i = i+1
for col in temp3:
    df.insert(i, col, temp3[col])

```

```
i = i+1
df.head()
```

	Unnamed: 0	protocol_type_icmp	protocol_type_tcp	protocol_type_udp	service_IRC
0	0	0	0	1	0
1	0	0	0	1	0
2	0	0	0	1	0
3	0	0	0	1	0
4	0	0	0	1	0

5 rows × 112 columns

## ▼ Normalizing the data using min-max scaller method

```
df_nor = df.iloc[:, 0:111]
df_nor
```

	Unnamed: 0	protocol_type_icmp	protocol_type_tcp	protocol_type_udp	service_
0	0	0	0	1	
1	0	0	0	1	
2	0	0	0	1	
3	0	0	0	1	
4	0	0	0	1	
...	...	...	...	...	...
311023	0	0	0	1	
311024	0	0	0	1	
311025	0	0	0	1	
311026	0	0	0	1	
311027	0	0	0	1	

311028 rows × 111 columns

```
#Using Min-Max Feature : series = (series-min(series))/(max(series)-min(series))
for col in df_nor.columns:
    df_nor[col] = (df_nor[col] - df_nor[col].min())/(df_nor[col].max() - df_nor[col].min())
df_nor["result"] = df["result"]
df_nor
```

	Unnamed: 0	protocol_type_icmp	protocol_type_tcp	protocol_type_udp	service_
0	0.0	0.0	0.0	1.0	
1	0.0	0.0	0.0	1.0	
2	0.0	0.0	0.0	1.0	
3	0.0	0.0	0.0	1.0	
4	0.0	0.0	0.0	1.0	
...	...	...	...	...	...
311023	0.0	0.0	0.0	1.0	
311024	0.0	0.0	0.0	1.0	
311025	0.0	0.0	0.0	1.0	
311026	0.0	0.0	0.0	1.0	
311027	0.0	0.0	0.0	1.0	

311028 rows × 112 columns

## ▼ Changing class labels to binary

```
for i in range(len(df_nor["result"])):
    if df_nor['result'][i] == "normal.":
        df_nor['result'][i] = 0
    else:
        df_nor['result'][i] = 1
```

/usr/local/lib/python3.7/dist-packages/ipykernel\_launcher.py:3: SettingWithCopyWarning: A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: [https://pandas.pydata.org/pandas-docs/stable/user\\_guide/10min.html#view](https://pandas.pydata.org/pandas-docs/stable/user_guide/10min.html#view)  
This is separate from the ipykernel package so we can avoid doing imports until  
/usr/local/lib/python3.7/dist-packages/ipykernel\_launcher.py:5: SettingWithCopyWarning: A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: [https://pandas.pydata.org/pandas-docs/stable/user\\_guide/10min.html#view](https://pandas.pydata.org/pandas-docs/stable/user_guide/10min.html#view)  
"""



## ▼ Handling Imbalanced data

```
unique_labels = list(df_nor["result"].unique())
print(unique_labels)
```

[0, 1]

```

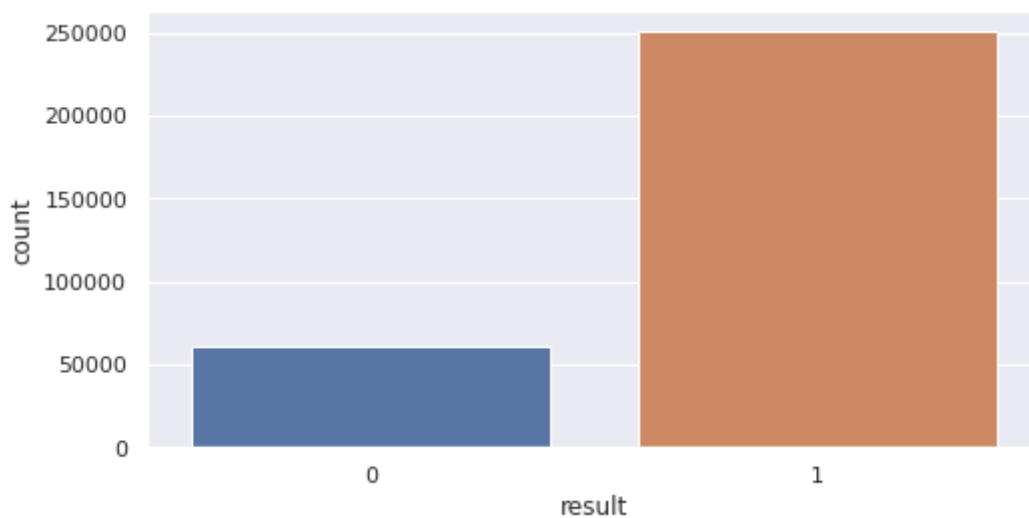
u=dict()
for r in df_nor["result"]:
    if r in u.keys():
        u[r]+=1
    else:
        u[r]=1
print("Count of each class label : \n",u)
import seaborn as sb
sb.set(rc = {'figure.figsize':(8,4)})
sb.countplot(df_nor['result'])

```

```

Count of each class label :
{0: 60592, 1: 250436}
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarning: Pass
FutureWarning
<matplotlib.axes._subplots.AxesSubplot at 0x7fc97306f350>

```



```

from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(df_nor.iloc[:, 0:111], df_nor["result"]

```

```

from imblearn.over_sampling import SMOTE
smote = SMOTE()

```

```

X_train_smote, y_train_smote = smote.fit_resample(X_train.astype('float'), y_train.astype(

```

## ▼ Applying Algorithm

```

from sklearn.naive_bayes import GaussianNB
from sklearn import metrics

```

```

gnb = GaussianNB()
gnb.fit(X_train_smote, y_train_smote)
y_pred = gnb.predict(X_test)

```

```
y_pred
```

```
array([1, 0, 1, ..., 1, 1, 1])
```

```
y_test = y_test.to_numpy()
```

```
y_test
```

```
array([1, 0, 1, ..., 1, 1, 1], dtype=object)
```

```
import numpy
```

```
y_test = numpy.array(list(y_test))
```

```
print("Accuracy:",100*metrics.accuracy_score(y_test, y_pred))
```

```
Accuracy: 83.3649487187731
```





## ▼ IMPLEMENTATION OF XGBOOST

### ▼ Importing Libraries

Double-click (or enter) to edit

```
from google.colab import drive
drive.mount('/content/drive')
```

Mounted at /content/drive

```
import pandas as pd
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import train_test_split
from sklearn import metrics
from sklearn.utils import shuffle
from imblearn.over_sampling import SMOTE
```

### ▶ Importing dataset

[ ] ↳ 2 cells hidden

### ▶ Removing 6 unnecessary features having 0 as the maximum count of values

[ ] ↳ 3 cells hidden

### ▶ Applying Onehot encoding to convert object features into integer

[ ] ↳ 2 cells hidden

### ▶ Normalizing the data using min-max scaller method

[ ] ↳ 2 cells hidden

### ▶ Changing class labels to binary

[ ] ↳ 1 cell hidden

## ▼ Handling Imbalanced data

```
unique_labels = list(df_nor["result"].unique())
print(unique_labels)
```

```
[0, 1]
```

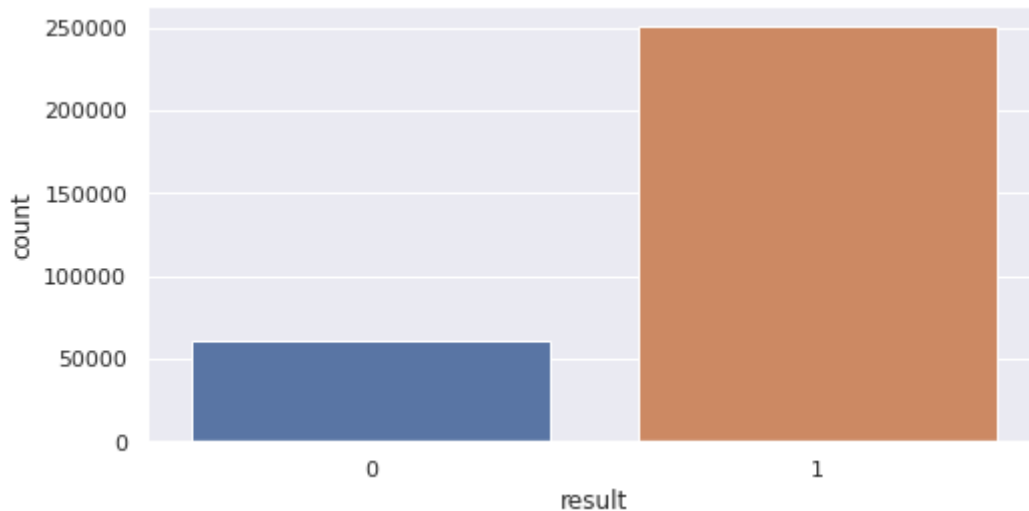
```
u=dict()
for r in df_nor["result"]:
    if r in u.keys():
        u[r]+=1
    else:
        u[r]=1
print("Count of each class label : \n",u)
import seaborn as sb
sb.set(rc = {'figure.figsize':(8,4)})
sb.countplot(df_nor['result'])
```

```
Count of each class label :
```

```
{0: 60592, 1: 250436}
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarning: Pass
FutureWarning
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f535ab0cb50>
```



```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(df_nor.iloc[:, 0:111], df_nor["result"]
```

```
from imblearn.over_sampling import SMOTE
smote = SMOTE()
```

```
X_train_smote, y_train_smote = smote.fit_resample(X_train.astype('float'), y_train.astype('float'))
```

## ▼ Applying Algorithm

```
from xgboost import XGBClassifier
from sklearn import metrics
```

```
model = XGBClassifier()
model.fit(X_train_smote, y_train_smote)
y_pred = model.predict(X_test)
```

```
y_pred
```

```
array([1, 0, 1, ..., 1, 1, 1])
```

```
y_test = y_test.to_numpy()
y_test
```

```
array([1, 0, 1, ..., 1, 1, 1], dtype=object)
```

```
import numpy
y_test = numpy.array(list(y_test))
```

```
print("Accuracy:",100*metrics.accuracy_score(y_test, y_pred))
```

```
Accuracy: 97.3014393038185
```

## ▼ IMPLEMENTATION OF RANDOM FOREST

### ▼ Importing Libraries

Double-click (or enter) to edit

```
from google.colab import drive
drive.mount('/content/drive')
```

Mounted at /content/drive

```
import pandas as pd
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import train_test_split
from sklearn import metrics
from sklearn.utils import shuffle
from imblearn.over_sampling import SMOTE
```

### ▶ Importing dataset

[ ] ↳ 2 cells hidden

### ▶ Removing 6 unnecessary features having 0 as the maximum count of values

[ ] ↳ 3 cells hidden

### ▶ Applying Onehot encoding to convert object features into integer

[ ] ↳ 2 cells hidden

### ▶ Normalizing the data using min-max scaller method

[ ] ↳ 2 cells hidden

### ▶ Changing class labels to binary

[ ] ↳ 1 cell hidden

## ▼ Handling Imbalanced data

```
unique_labels = list(df_nor["result"].unique())
print(unique_labels)
```

```
[0, 1]
```

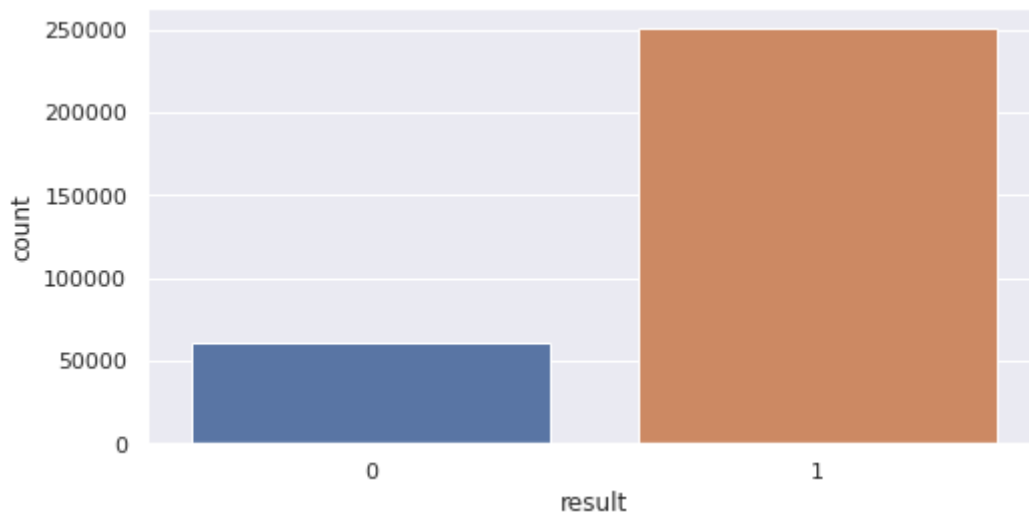
```
u=dict()
for r in df_nor["result"]:
    if r in u.keys():
        u[r]+=1
    else:
        u[r]=1
print("Count of each class label : \n",u)
import seaborn as sb
sb.set(rc = {'figure.figsize':(8,4)})
sb.countplot(df_nor['result'])
```

```
Count of each class label :
```

```
{0: 60592, 1: 250436}
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarning: Pass
FutureWarning
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f729c7a2b10>
```



```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(df_nor.iloc[:, 0:111], df_nor["result"]
```

```
from imblearn.over_sampling import SMOTE
smote = SMOTE()
```

```
X_train_smote, y_train_smote = smote.fit_resample(X_train.astype('float'), y_train.astype('float'))
```

## ▼ Applying Algorithm

```
from sklearn.ensemble import RandomForestClassifier
from sklearn import metrics
```

```
clf = RandomForestClassifier(n_estimators = 100)
clf.fit(X_train_smote, y_train_smote)
y_pred = clf.predict(X_test)
```

y\_pred

```
array([1, 0, 1, ..., 1, 1, 1])
```

```
y_test = y_test.to_numpy()
y_test
```

```
array([1, 0, 1, ..., 1, 1, 1], dtype=object)
```

```
import numpy
y_test = numpy.array(list(y_test))
```

```
print("Accuracy:",100*metrics.accuracy_score(y_test, y_pred))
```

```
Accuracy: 97.62938194600736
```

---

# IMPLEMENTATION OF SELF\_IMPLEMENTED HYBRID ALGORITHM

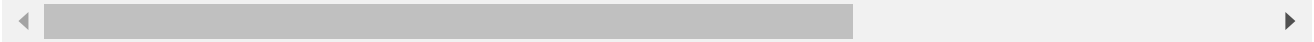
---

## Importing Libraries

Double-click (or enter) to edit

```
from google.colab import drive
drive.mount('/content/drive')
```

Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.n



```
import pandas as pd
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import train_test_split
from sklearn import metrics
from sklearn.utils import shuffle
from imblearn.over_sampling import SMOTE
```

## Importing dataset

```
df = pd.read_csv("/content/drive/MyDrive/ISA_Project/dataset/corrected.csv")
df
```

	0	udp	private	SF	105	146	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
0	0	udp	private	SF	105	146	0	0	0	0	0	0	0	0	0
1	0	udp	private	SF	105	146	0	0	0	0	0	0	0	0	0

```
df.columns = ['Unnamed: 0', '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', 'same_srv_rate', 'diff_srv_rate',
              'srv_diff_host_rate', 'una1', 'una2', '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', 'result']
```

df

	Unnamed: 0	protocol_type	service	flag	src_bytes	dst_bytes	land	wrong_fr
0	0	udp	private	SF	105	146	0	
1	0	udp	private	SF	105	146	0	
2	0	udp	private	SF	105	146	0	
3	0	udp	private	SF	105	146	0	
4	0	udp	private	SF	105	146	0	
...	...	...	...	...	...	...	...	
311023	0	udp	private	SF	105	147	0	
311024	0	udp	private	SF	105	147	0	
311025	0	udp	private	SF	105	147	0	
311026	0	udp	private	SF	105	147	0	
311027	0	udp	private	SF	105	147	0	

311028 rows × 42 columns

## Removing 6 unnecessary features having 0 as the maximum count of values

```
for col in df.columns:
    if(df[col].nunique() <5):
```



```
print("\n"+col+" : Unique values : " + str(df[col].nunique()) + " : " + str(df[col].un
```

```
protocol_type : Unique values : 3 : ['udp' 'tcp' 'icmp']
```

```
land : Unique values : 2 : [0 1]
```

```
wrong_fragment : Unique values : 3 : [0 1 3]
```

```
urgent : Unique values : 4 : [0 2 1 3]
```

```
logged_in : Unique values : 2 : [0 1]
```

```
root_shell : Unique values : 2 : [0 1]
```

```
su_attempted : Unique values : 3 : [0 1 2]
```

```
num_shells : Unique values : 4 : [0 2 1 5]
```

```
num_outbound_cmds : Unique values : 1 : [0]
```

```
is_host_login : Unique values : 2 : [0 1]
```

```
is_guest_login : Unique values : 2 : [0 1]
```

```
df.drop(["land", "logged_in", "root_shell", "num_outbound_cmds", "is_host_login", "is_gues
df.head()
```

	Unnamed: 0	protocol_type	service	flag	src_bytes	dst_bytes	wrong_fragment	urg
0	0	udp	private	SF	105	146	0	
1	0	udp	private	SF	105	146	0	
2	0	udp	private	SF	105	146	0	
3	0	udp	private	SF	105	146	0	
4	0	udp	private	SF	105	146	0	

```
print(df.shape)
```

```
(311028, 36)
```

## ▼ Applying Onehot encoding to convert object features into integer

```
temp1 = pd.get_dummies(df.protocol_type, prefix='protocol_type')
temp2 = pd.get_dummies(df.service, prefix='service')
temp3 = pd.get_dummies(df.flag, prefix='flag')
df.drop(['protocol_type', 'flag', 'service'], axis='columns', inplace=True)
```

```
i=1
```

```

for col in temp1:
    df.insert(i, col, temp1[col])
    i = i+1
for col in temp2:
    df.insert(i, col, temp2[col])
    i = i+1
for col in temp3:
    df.insert(i, col, temp3[col])
    i = i+1
df.head()

```

	Unnamed: 0	protocol_type_icmp	protocol_type_tcp	protocol_type_udp	service_IRC
0	0	0	0	1	0
1	0	0	0	1	0
2	0	0	0	1	0
3	0	0	0	1	0
4	0	0	0	1	0

5 rows × 112 columns

## ▼ Normalizing the data using min-max scaller method

```

df_nor = df.iloc[:, 0:111]
df_nor

```

```

Unnamed: 0  protocol_type_icmp  protocol_type_tcp  protocol_type_udp  service_

```

```

#Using Min-Max Feature : series = (series-min(series))/(max(series)-min(series))
for col in df_nor.columns:
    df_nor[col] = (df_nor[col] - df_nor[col].min())/(df_nor[col].max() - df_nor[col].min())
df_nor["result"] = df["result"]
df_nor

```

	Unnamed: 0	protocol_type_icmp	protocol_type_tcp	protocol_type_udp	service_
0	0.0	0.0	0.0	1.0	
1	0.0	0.0	0.0	1.0	
2	0.0	0.0	0.0	1.0	
3	0.0	0.0	0.0	1.0	
4	0.0	0.0	0.0	1.0	
...	...	...	...	...	
311023	0.0	0.0	0.0	1.0	
311024	0.0	0.0	0.0	1.0	
311025	0.0	0.0	0.0	1.0	
311026	0.0	0.0	0.0	1.0	
311027	0.0	0.0	0.0	1.0	

311028 rows × 112 columns

## ▼ Changing class labels to binary

```

for i in range(len(df_nor["result"])):
    if df_nor['result'][i] == "normal.":
        df_nor['result'][i] = 0
    else:
        df_nor['result'][i] = 1

```

/usr/local/lib/python3.7/dist-packages/ipykernel\_launcher.py:3: SettingWithCopyWarning: A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: <https://pandas.pydata.org/pandas-docs/stable/using.html#viewing-data>

This is separate from the ipykernel package so we can avoid doing imports until /usr/local/lib/python3.7/dist-packages/ipykernel\_launcher.py:5: SettingWithCopyWarning: A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: <https://pandas.pydata.org/pandas-docs/stable/using.html#viewing-data>

"""

## ► Handling Imbalanced data

[ ] ↳ 5 cells hidden

## ▼ Applying Algorithm

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.tree import DecisionTreeClassifier
from xgboost import XGBClassifier
from sklearn import metrics
```

```
train_x, val_x, train_y, val_y = train_test_split(X_train, y_train, stratify=y_train, test_size=0.2)
```

```
gnb = GaussianNB()
gnb.fit(X_train_smote, y_train_smote)
y_val_pred1 = pd.DataFrame(gnb.predict(val_x))
y_test_pred1 = pd.DataFrame(gnb.predict(X_test))
```

```
model = XGBClassifier()
model.fit(X_train_smote, y_train_smote)
y_val_pred2 = pd.DataFrame(model.predict(val_x))
y_test_pred2 = pd.DataFrame(model.predict(X_test))
```

```
model_dec = DecisionTreeClassifier()
model_dec = model_dec.fit(X_train_smote, y_train_smote)
y_val_pred3 = pd.DataFrame(model_dec.predict(val_x))
y_test_pred3 = pd.DataFrame(model_dec.predict(X_test))
```

```
val_x = val_x.reset_index(drop=True)
X_test = X_test.reset_index(drop=True)
```

```
val_input = pd.concat([pd.DataFrame(val_x, columns=X.columns), y_val_pred1, y_val_pred2, y_val_pred3])
test_input = pd.concat([pd.DataFrame(X_test, columns=X.columns), y_test_pred1, y_test_pred2, y_test_pred3])
```

```
print(val_input)
print(test_input)
```

```

      Unnamed: 0  protocol_type_icmp  protocol_type_tcp  ...  0  0  0
0      0.000000          1.0          0.0  ...  1  1  1
1      0.000000          1.0          0.0  ...  1  1  1
2      0.000000          0.0          1.0  ...  0  0  0
3      0.000000          0.0          0.0  ...  0  1  1
4      0.000017          0.0          1.0  ...  0  1  1
...          ...          ...          ...  ...  ...  ...
43539  0.000000          1.0          0.0  ...  1  1  1
```

43540	0.000000	1.0	0.0	...	1	1	1
43541	0.000000	0.0	1.0	...	1	1	1
43542	0.000000	0.0	1.0	...	1	1	1
43543	0.000000	0.0	1.0	...	1	1	1

[43544 rows x 114 columns]

	Unnamed: 0	protocol_type_icmp	protocol_type_tcp	...	0	0	0
0	0.0	1.0	0.0	...	1	1	1
1	0.0	0.0	0.0	...	0	0	0
2	0.0	0.0	1.0	...	1	1	1
3	0.0	1.0	0.0	...	1	1	1
4	0.0	0.0	1.0	...	1	1	1
...	...	...	...	...	...	...	...
93304	0.0	0.0	1.0	...	0	1	1
93305	0.0	1.0	0.0	...	1	1	1
93306	0.0	1.0	0.0	...	1	1	1
93307	0.0	1.0	0.0	...	1	1	1
93308	0.0	0.0	1.0	...	1	1	1

[93309 rows x 114 columns]

```
clf = RandomForestClassifier(n_estimators = 200)
clf.fit(val_input.astype('float'), val_y.astype('int'))
```

```
/usr/local/lib/python3.7/dist-packages/sklearn/utils/validation.py:1679: FutureWarning:
FutureWarning,
RandomForestClassifier(n_estimators=200)
```

```
print(clf.score(test_input.astype('float'), y_test.astype('int'))*100)
```

```
/usr/local/lib/python3.7/dist-packages/sklearn/utils/validation.py:1679: FutureWarning:
FutureWarning,
97.98626070368346
```

## COMPARISON OF ALGORITHMS

ALGORITHM	ACCURACY
Naïve Bayes	(78 – 83 )%
Decision Tree	Approx.(97 %)
Random Forest	Approx.(96.5-97.5)%
XGBoost	Approx.(96-97)%
HYBRID	Approx.(98 % )

## INFERENCE

All the trained algorithms are able to detect the ddos attack with naïve bayes having the lowest accuracy and the self trained hybrid algorithm have the highest accuracy.

# REFERENCES

- [1] Murk Marvi, Asad Arfeen, Riaz Uddin, "A generalized machine learning-based model for the detection of DDoS attacks", 2020 doi: <https://doi.org/10.1002/nem.2152>
- [2] Saikat Das, Deepak Venugopal, Sajjan Shiva, "A Holistic Approach for Detecting DDoS Attacks by Using Ensemble Unsupervised Machine Learning", 2020 Advances in Information and Communication, 2020, Volume 1130, doi: [https://doi.org/10.1007/978-3-030-39442-4\\_53](https://doi.org/10.1007/978-3-030-39442-4_53)
- [3] Awan, M.J.; Farooq, U.; Babar, H.M.A.; Yasin, A.; Nobanee, H.; Hussain, M.; Hakeem, O.; Zain, A.M. Real-Time DDoS Attack Detection System Using Big Data Approach. *Sustainability* 2021, *13*, 10743. <https://doi.org/10.3390/su131910743>
- [4] M. S. Elsayed, N. -A. Le-Khac, S. Dev and A. D. Jurcut, "DDoSNet: A Deep-Learning Model for Detecting Network Attacks," 2020 IEEE 21st International Symposium on "A World of Wireless, Mobile and Multimedia Networks" (WoWMoM), 2020, pp. 391-396, doi: 10.1109/WoWMoM49955.2020.00072.
- [5] A. Koay, A. Chen, I. Welch and W. K. G. Seah, "A new multi classifier system using entropy-based features in DDoS attack detection," 2018 International Conference on Information Networking (ICOIN), 2018, pp. 162-167, doi: 10.1109/ICOIN.2018.8343104.