NETWORK INTRUSION DETECTION SYSTEM BASED WEB APPLICATION

A PROJECT REPORT

Submitted by

DHRUV VAIDH,19BCE0817

GUNIK LUTHRA,19BCE2285

YASHVARDHAN SINGH BHADAURIA,19BCE2129

Course Code: CSE3502
Course Title:Information Security Management

Under the guidance of
Dr. Ramani S
Associate Professor
SCOPE, VIT, Vellore.

SCHOOL OF COMPUTER SCIENCE AND ENGINEERING

May -2021

INDEX

Abstract:

Intrusion detection is one of the key interests in network administration and security. There is a need to protect the networks from known vulnerabilities and at the same time take steps to identify new and unknown but potential device abuses by creating more robust and effective systems for intrusion detection.

An intrusion detection system (IDS) audits the traffic flowing in the network for suspicious activity and signals when any malicious activity is discovered. For most Intrusion Detection Systems discovering anomalies and generating reports for the intrusions discovered are the elementary responsibilities. In some cases, if a malicious activity or intermittent traffic is detected in the network, intrusion detection systems are programmed to take actions like deterring traffic sent from incredulous IP addresses. An IDS is a software that is developed to detect intrusions within the network. It is employed to hunt and pinpoint the intruders within the network.

In this project, we have built a Flask Web Application that takes in some inputs through a form displayed on the Front-End and passes to our ML Model which predicts what type of intrusion-attack class it is. If it is an intrusion, we have 4 Attack Classes as output, namely DOS, PROBE, R2L and U2R. If it isn't an intrusion, the model would display 'NORMAL' as output. Coming to the ML part, we have built a lot of Machine Learning Classifiers. They help in increasing the accuracy at which intrusions are detected. Before building the classifier we will be required to select the most optimal features using Feature selection. We will be employing the concept of Select Best-K Technique. Because of this the attacks are detected more efficiently even in highly congested networks. It also leads to a lesser number of false positives and therefore a lower rate of false alarms. The proposed network intrusion detection system can classify packets in real-time based on the packets collected from the network flow.

Keywords: Network Intrusion Detection System, malicious activity, DOS, PROBE, R2L, U2R

1. Introduction:

1.1 Theoretical Background:

Intrusion detection is one of the key interests in network administration and security. There is a need to protect the networks from known vulnerabilities and at the same time take steps to identify new and unknown but potential device abuses by creating more robust and effective systems for intrusion detection. An intrusion detection system (IDS) audits the traffic flowing in the network for suspicious activity and signals when any malicious activity is discovered. For most Intrusion Detection Systems discovering anomalies and generating reports for the intrusions discovered are the elementary responsibilities. In some cases, if a malicious activity or intermittent traffic is detected in the network, intrusion detection systems are programmed to take actions like deterring traffic sent from incredulous IP addresses. An IDS is a software that is developed to detect intrusions within the network. It is employed to hunt and pinpoint the intruders within the network.

NETWORK-BASED IDS: The data fed into this type of IDS is obtained from the flow of packets in the network having data.

MISUSE DETECTION: This type of IDS has the functionality to trace previously recognized patterns of pernicious activity going on in the network and spot intrusions.

The IDS we will be developing is a network-based IDS that is programmed to detect any misuse of the network resources (misuse detection) i.e. it detects malicious packets flowing in a network.

1.2 Motivation:

Lately, intrusion detection has become a major concern in the field of network security and administration. Considering intrusion as a security threat, a network needs a system which protects it from known vulnerabilities and at the same time takes measures to detect unknown ones for efficient functioning of the network. The detection system must be accurate up to some extent in detecting attacks with a possible minimum number of false positives.

1.3. Aim of the proposed work:

Lately, intrusion detection has become a major concern in the field of network security and administration. Considering intrusion as a security threat, a network needs a system which protects it from known vulnerabilities and at the same time take measures to detect unknown ones for efficient functioning of the network. The detection system must be accurate up to some extent in detecting attacks with a possible minimum number of false positives.

1.4. Objectives of the proposed work:

Intrusion detection is one of the key interests in network administration and security. There is a need to protect the networks from known vulnerabilities and at the same time take steps to identify new and unknown but potential device abuses by creating more robust and effective systems for intrusion detection. An intrusion detection system (IDS) audits the traffic flowing in the network for suspicious activity and signals when any malicious activity is discovered. An IDS is a software that is developed to detect intrusions within the network. It is employed to hunt and pinpoint the intruders within the network. NETWORK-BASED IDS: The data fed into this type of IDS is obtained from the flow of packets in the network having data. MISUSE DETECTION: This type of IDS has the functionality to trace previously recognized patterns of pernicious activity going on in the network and spot intrusions.

2. Literature Survey:

a. INTRUSION DETECTION USING SOFT COMPUTING TECHNIQUES

METHODOLOGY

Soft Computing can be characterized as the capacity of a program or potentially a framework to learn and improve their performance on a specific task or group of tasks over a time.

EXPLANATION

Soft computing includes Machine learning-based techniques, which are broadly classified as Bayesian approaches, support vector machines, neural networks, fuzzy logic, and genetic algorithms.

ADVANTAGES

The advantage of the approach of using mutual information and resulting linear rules seems very effective because of the reduced complexity and higher detection rate.

DISADVANTAGES

The approach has the disadvantage that it considers only the discrete features and not the continuous ones.

b. INTRUSION DETECTION SYSTEMS

ABSTRACT

An intrusion detection system (IDS) are devices or software that are used to monitor networks for any unkind activities that bridge the normal functionality of systems hence causing some policy violation. This paper reviews some of the intrusion detection systems and software that highlight their main classifications and their performance evaluations and measure.They are:Signature-Based Detection,Anomaly-Based Detection,Network Intrusion Detection System.

NETWORK INTRUSION DETECTION SYSTEMS

The NIDS can capture and analyze data to detect known attacks by comparing patterns or signatures of the database or detection of illegal activities by scanning traffic for anomalous activity. NIDS is also referred to as “packet-sniffers,” because it captures the packets passing through the communication mediums.

SIGNATURE BASED DETECTION

Signature-based detection system (also called misuse based), this type of detection is very effective against known attacks, and it depends on the receiving of regular updates of patterns. But signature-based detection does not work well when the user uses advanced technologies like NOP generators, payload encoders, and encrypted data channels.

ANOMALY BASED DETECTION

An anomaly-based intrusion detection system is an intrusion detection system for detecting both network and computer intrusions and misuse by monitoring system activity and classifying it as either normal or anomalous. The classification is based on heuristics , rather than patterns or signatures, and attempts to detect any misuse that falls out of normal system operation.

c. A HYBRID INTRUSION DETECTION SYSTEM DESIGN FOR COMPUTER NETWORK SECURITY

METHODOLOGY

IDSs collect network traffic information from some point on the network or computer system and then use this information to secure the network. Intrusion detection systems can be misuse-detection or anomaly detection based.

EXPLANATION

The hybrid IDS is obtained by combining packet header anomaly detection (PHAD) and network traffic anomaly detection (NETAD) which are anomaly-based IDSs with the misuse-based IDS Snort which is an open-source project.

ADVANTAGES

Misuse detectors are very efficient in detecting attacks without signaling false alarms (FA). They can quickly detect specially designed intrusion tools and techniques.They can provide systems administrators an easy to use tool to monitor their systems even if they are not security experts.

DISADVANTAGES

Misuse detectors can only detect attacks known beforehand.

d. AN EFFICIENT NETWORK INTRUSION DETECTION METHOD BASED ON INFORMATION THEORY AND GENETIC ALGORITHM

METHODOLOGY

This approach uses information theory to filter the traffic data and thus reduce the complexity.We choose the genetic algorithm to classify the Traffic due to its simplicity.

EXPLANATION

We use information theory to identify the most relevant features to be used in our online analysis of the traffic data. We then apply data mining programs and genetic algorithms to compute linear classifiers that accurately capture the difference between intrusions and normal activities. Finally, we use the linear classifier with the trained thresholds and coefficients to detect a wide range of network attacks.

ADVANTAGES

The linear structure of the rule makes the intrusion detection system efficient in real time processing of the traffic data. The evaluation of our approach showed that the hybrid method of using discrete and continuous features can - achieve a betler detection rate of network attacks.

DISADVANTAGE

The linear structure of the rule makes the intrusion detection system efficient in real time processing of the traffic data. The evaluation of our approach showed that the hybrid method of using discrete and continuous features can - achieve a betler detection rate of network attacks.

e. ANOMALY DETECTION SCHEMES IN NETWORK INTRUSION

METHODOLOGY

Anomaly detection is a key element of intrusion detection in which perturbations of normal behavior suggest the presence of intentionally or unintentionally induced attacks, faults, defects, etc.

EXPLANATION

These approaches attempt to build some kind of a model over the normal data and then check to see how well new data fits into that model.

ADVANTAGES

Experimental results indicate that the most successful anomaly detection techniques were able to achieve the detection rate of 74% for attacks involving multiple connections and detection rate of 56% for more complex single connection attacks, while keeping the false alarm
rate at 2%.

DISADVANTAGES

Data generated from network traffic monitoring tends to have very high volume, dimensionality and heterogeneity, making the performance of serial data mining algorithms unacceptable for on-line analysis.

f. HYBRID APPROACHES FOR MODELLING INTRUSION DETECTION SYSTEM

METHODOLOGY

Presenting two hybrid approaches for modeling IDS. Decision trees (DT) and support vector machines (SVM) are combined as a hierarchical hybrid intelligent system model (DT–SVM) and an ensemble approach combining the base classifiers.

EXPLANATION

The hybrid intrusion detection model combines the individual base classifiers and other hybrid machine learning paradigms to maximize detection accuracy and minimize computational complexity.

ADVANTAGES

The ensemble approach gave 100% accuracy for Probe class, and this suggests that if proper base classifiers are chosen 100% accuracy might be possible for other classes too.

DISADVANTAGES

In a hierarchical hybrid intelligent system , the overall functioning of the system depends on the correct functionality of all the layers.

g. UTILISING NEURAL NETWORKS FOR EFFECTIVE INTRUSION DETECTION

METHODOLOGY

A proactive and dynamic model, based on neural networks
that could be utilized to minimise and control intrusion to an organisation's computer system. The model will be based on the assumption that each user is unique and leaves a unique footprint on a computer system when using it.

EXPLANATION

The approach consists of two steps. During the first step a reference behaviour pattern is built up for each user on the system and during the second step the current behaviour of each user is compared against the reference pattern.

ADVANTAGES

Neural networks have the potential to overcome one of the shortcomings of current anomaly intrusion detection systems; that is, complex and high computational overheads and incorrect training processes.

DISADVANTAGES

Neural networks usually require much more data than traditional machine learning algorithms, as in at least thousands if not millions of labeled samples. This isn't an easy problem to deal with and many machine learning problems can be solved well with less data if you use other algorithms.

h. MODEL GENERATION FOR INTRUSION DETECTION SYSTEM USING GENETIC ALGORITHM

METHODOLOGY

Novel approach to detect malicious intrusions by using a complex artificial intelligence method known as a genetic algorithm applied to an Intrusion Detection System.

EXPLANATION

A genetic algorithm is a method of artificial intelligence problem-solving based on the theory of Darwinian evolution applied to mathematical models. The genetic algorithm designed for this experiment promoted a high detection rate of malicious behavior and a low false positive rate of normal behavior classified as
malicious.

ADVANTAGES

The genetic algorithm is successfully able to generate an accurate empirical behavioral model from training data and then able to successfully apply this empirical knowledge to data never seen before.

DISADVANTAGES

A genetic algorithm is quite straightforward in general, but it could be complex in most cases.

i. A LIGHTWEIGHT INTRUSION DETECTION FRAMEWORK FOR WIRELESS SENSOR NETWORKS

METHODOLOGY

A simple, lightweight detection framework for the prevention and detection of common routing attacks in WSNs.

EXPLANATION

A lightweight intrusion detection framework integrated for clustered sensor networks and algorithms to minimize the triggered intrusion modules in clustered WSNs by using an over-hearing mechanism to reduce the sending alert packets.

ADVANTAGES

It is effective, even when the density of the network is high and there is a high probability of collisions in WSNs.
The detection modules involve less energy consumption than techniques proposed in previous works, using an over-hearing mechanism to reduce the transmission of alert packets.

DISADVANTAGES

Further research needs to be performed with detailed simulation of different attack scenarios.

3. Overview of the Proposed System:

3.1 Introduction and Related Concepts

Intrusion is considered to compromise the integrity, confidentiality, or availability of valuable assets on the computer systems. An intrusion detection system (IDS) audits the traffic flowing in the network for suspicious activity. It then alerts the system administrator when any malicious activity is discovered in the network. The primary functions of intrusion detection systems are discovering anomalies and producing detailed reports for the intrusions discovered. In some cases, if malicious activity or intermittent traffic is detected in the network, intrusion detection systems are programmed to take actions like deterring traffic sent from incredulous IP addresses. An IDS is a programmable software that is developed to detect intrusions within the network. It is employed to hunt and pinpoint the intruders causing chaos within the network. The main principles of IDS are integrity, availability, confidentiality and accountability. An IDS is built using both software and hardware. It can detect highly dangerous intrusions within the network. The main purpose of IDS is to detect unauthorized packets and malicious communications that happen in computer systems and networks. The most vital ingredient for the success of intrusion detection systems is feature selection. We can classify the Intrusion Prevention Systems into 4 kinds:

a. **Network Based Intrusion Prevention System** –

It detects and identifies any suspicious or unwanted traffic flowing within the network by evaluating the activity of various protocols for the network.

b. **Wireless Intrusion Prevention System** –

It detects and identifies any suspicious or unwanted traffic flowing within the network by evaluating the activity of the wireless protocols within the network.

c. **Network Behavior Analysis** –

It evaluates the entire network to spot for vulnerable threats. For example, Remote Code Executions, Denial of Service.

d. **Host Based Intrusion Prevention System –**

In this type of system a software is installed on the host to monitor for any pernicious activity. This is done by evaluating the host system.

Intrusions are considered as a sequence of steps taken to compromise the integrity, confidentiality, or availability of valuable assets on the computer systems. Intruders gain unauthorized access to the resources available on the system. They use all kinds of techniques to gain access to confidential information and manipulate the data available on the system. This can sometimes damage the system and render it worthless. An IDS can be considered to be a blend of software and hardware units that can be used to identify and pinpoint unauthorized experiments to gain access to the network. All the network related activities can be audited by an IDS which in turn can be used to suspect the traces of invasions within the network. The end goal of an IDS is to trigger alerts when a suspicious activity has occurred by notifying the System Administrator. Intrusion detection techniques can be classified into 2 types:

A. **Anomaly Detection**: In this kind of detection the system alerts malicious tasks by identifying deviations i.e. how differently are the network activities occurring as compared to regular patterns.

B. **Misuse Detection**: In this kind of system intrusions are detected on the basis of already known patterns i.e. previously occurred malicious activity. This method can be used to identify and pinpoint known attack patterns more accurately. Misuse Detection has a disadvantage over Anomaly Detection i.e. only those kinds of intrusions can be identified which can correlate to already known patterns of attack. An ideal IDS will monitor all the happenings within the network and then decide whether those happenings are malicious or normal. The decision is based on system availability, confidentiality and integrity of the information resources. The various aspects to be considered while building and IDS are the following: data pre-processing, collection of data, detecting intrusions and curating reports with appropriate responses..
An Intrusion Detection System works in the following manner: Collecting Data, Selecting Features, Analyzing the Data, and the Actions to be Performed.

a. **Collecting Data**: We need to gather reports on the traffic flowing in the network like hosts alive, protocols used and the various forms of traffic flowing.

b. **Selecting Features**: After collecting a huge amount of data, the next step is to pick all those required features which we want to work upon.

c. **Analyzing the Data**: In this step the data about the features which are selected data is evaluated to help us determine if the data is unnatural or not.

Actions to be Performed: When a malicious attack has taken place the system administrator is alarmed or notified by the IDS. The details about the type of attack are also provided by the IDS. The IDS closes the unnecessary network ports and processes to further mitigate the attacks from happening.

3.2 Framework, Architecture or Module for the Proposed System (with explanation)

Dataset:

An accurate intrusion detection system requires a good set of labeled examples to sharply detect intrusions in the real world. In order to achieve higher accuracies while modeling an intrusion detection system, a dataset that represents the real world more accurately can only be used. The NSL-KDD data set is a polished version of its forerunner KDD 99 data set.

NSL-KDD data set is employed to check the validity and performance of the different classification algorithms in classifying the attacks in the traffic patterns of the network.

Description:

It has all the basic records of the complete KDD data set required for various models. There are a bunch of files to download on the internet for ML practitioners and researchers.

Each record has a total of 43 attributes for identifying the various features of the network packets and a label is named to every packet. This label is used to classify that record either as an attack category or as none/normal. The fundamental points of the features are their name, their description, and sample values. The following table contains information about all the 43

attributes present within the NSL-KDD data set. The 42nd attribute unfolds data about labels of each record which represents the packets in the network and that they're categorized as 1 ordinary class and 4 attack-type classes. The 4 attack classes used to classify the records in the dataset are DoS, Probe, R2L, and U2R. The mentioned table is as follows:

Attribute No.	Attribute Name	Description	Sample Data
1	Duration	Time duration of the connection	0
2	Protocol_type	Protocol utilized for the Connection	Tcp
3	Service	Destination network service used	ftp_data
4	Flag	Status of the connection – Normal or Error	SF
5	Src_bytes	Number of information bytes transferred from sorce to destination in single connection	491

The advantages in adopting the NSL-KDD dataset are

1. There won't be any similar records that have a stronger rate of decline in the test sample.
2. As compared to the KDD-99, the NSL-KDD shows a lesser number of data points. Therefore, using them for training machine learning models is less costly algorithmically.
3. The number of records chosen for each tough group of rates is inversely proportional to the percentage of records in the original KDD dataset.

NSL-KDD train and test data distribution

The 4 attack categories available within the NSL-KDD data set are:

1. **DOS**: This kind of attack leads to draining of the victims' resources and making it incapable in responding to legitimate requests. This is one of the 4 attack categories.

   Example: syn flooding. The suitable features from the dataset for this attack class are: "serror_rate" and "flag_SF".

2. **U2R** (unauthorized access to local root privileges): In this kind of attack, an attacker tries to obtain root/administrator privileges by taking advantage of some vulnerability within the victim's system. The attacker usually uses a traditional account to login into a victim's system. The suitable attributes from the dataset for this attack class are: **"root_shell", "service_http", and "dst_host_same_src_port_rate"**.

3. **Probe**: This kind of attack involves obtaining sensitive information present in the victim's computer/device. The suitable attributes for this attack class are: **"Protocol_type_icmp" and "dst_host_same_src_port_rate"**.

4. **R2L** (Remote to Local Attack): This kind of attack involves unapproved access of the victim's device by gaining root access where he/she can view the data within that device with root privileges and all this is done from a far off(remote) machine by the attacker. E.g. password bruteforce attack. The suitable features from the dataset for this attack class are: **"dst_bytes", "dst_host_srv_diff_host_rate", and "dst_host_same_src_port_rate"**

PROPOSED IDS METHODOLOGY FOR THE ML MODEL:

The primary goal is to design a plan for detecting intrusions within the system with the least possible number of features within the dataset. Based on the data from previous papers published, we can tell that only a subdivision of features in the dataset are derivative to the Intrusion Detection System. We have to cut back the dimensionality of the dataset to build an improved classifier in a justifiable amount of time. The approach we are going to use has a total of 4 stages: In the first stage, we pick out the significant features for every class using feature selection. In the next we combine the various features, so that the final cluster of features are optimal and relevant for each attack class. The third stage is for building a classifier. Here, the optimal features found in the previous stage are sent as input into the classifier. In the last stage, we test the model by employing a test dataset.

The proposed approach consists of 4 fundamental stages:

1. Feature selection
2. Combining the optimal features
3. Building a classifier
4. Evaluation

FEATURE SELECTION:

We already know that network intrusion systems require bulk amounts of raw data to be dealt with. The most crucial building block of developing such a sophisticated system is Feature Selection. One thing which has a major effect on the dimensionality of the dataset is the complexity of the model. It results in obtaining high computational time and costs and also low classification accuracies. A significant number of techniques and approaches are used to get rid of repetitive and insignificant features.

These methods are chosen in such a way that we select only those features which are optimal and can be used to enhance the performance of the model.

We will be using two methodologies for feature reduction in this project.They are:

1) Analysis of our Data Audit Report and then dropping columns on the basis of the following parameters:-
    a) **Based on low variance** (near zero variance)
    b) **High missings** (>25% missings)
    c) **High correlations** between two numerical variables

2) Incorporating the Select K-Best technique. The classes in the **sklearn.feature_selection** module have been used for feature selection/dimensionality reduction on sample sets, either to improve estimators' accuracy scores or to boost their performance on very high-dimensional datasets.This technique has certain advantages as well. They are:

    a) **Reduces Overfitting**: Less redundant data means less possibility of making decisions based on redundant data/noise.
    b) **Improves Accuracy**: Less misleading data means modeling accuracy improves.
    c) **Reduces Training Time**: Less data means that algorithms train faster.

COMBINING OPTIMAL FEATURES:

In the last stage of feature selection, we combine the list of features generated for each attack into a single list. For some of the attack classes the highest ranks i.e. the top 4 features chosen for classification. But for some types of attack classes we can only take 1 feature since that particular feature is at the top of the rank table and the remaining features are at the very bottom of the table. So, the final set of combined optimal features can be used to entirely distinguish the attack types.

BUILDING A CLASSIFIER:

The obtained subset of relevant features of each attack type in the above step is now used to build a classifier in this phase.We have predicted the accuracy of a lot of ML classifiers, and finally have chosen the one with the highest accuracy, which in our case came out to be the Logistic Regression model.

MODELS TESTED WITH:

1. **DECISION TREE CLASSIFIER**:

Decision tree is an algorithm which takes decisions at each node of the tree and is widely used for **regression** and **classification**. It is a supervised learning algorithm in Machine learning where which attribute should be at which node is learnt by using a set of labeled examples. The main advantage in using decision trees is that they can be trained very easily and they can even classify nonlinear data. **Overfitting** is one of the main problems of the decision tree as it constructs its nodes by thoroughly going through the entire training set which may result in a low accuracy while using

unknown data like the test set. We got an accuracy of **43%** for this model.

```
Decision Trees

[87]
from sklearn.model_selection import cross_val_score
from sklearn import metrics
import sklearn.tree as dt
from sklearn.tree import DecisionTreeClassifier, DecisionTreeRegressor, export_graphviz, export
from sklearn.model_selection import GridSearchCV

[88]
clf_tree = DecisionTreeClassifier( max_depth = 5)
clf_tree=clf_tree.fit( X_train, y_train )

[89]
y_pred=qda.predict(X_test)
y_pred

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

[90]
from sklearn.metrics import accuracy_score
accuracy_score(y_test, y_pred)

0.4307767377900013
```

2. LOGISTIC REGRESSION

Logistic regression is a statistical model that in its basic form uses a logistic function to model a binary dependent variable, although many more complex extensions exist. In **regression** analysis, logistic regression (or logit regression) is estimating the parameters of a **logistic model** (a form of binary regression).We got an accuracy of **93.2%** for this model.

```
1) LogisticRegression

lr_clf = LogisticRegression(random_state=0, solver='lbfgs',multi_class='multinomial').fit(X_train, y_train)

y_pred=lr_clf.predict(X_test)
y_pred

array([1., 0., 2., ..., 1., 0., 2.])

from sklearn.metrics import accuracy_score
accuracy_score(y_test, y_pred)

0.9324641524256524
```

3. K NEAREST NEIGHBOURS

K nearest neighbors is a simple algorithm that stores all available cases and classifies new cases based on a **similarity measure** (e.g., distance functions). KNN has been used in statistical estimation and pattern recognition already in the beginning of 1970's as a non-parametric technique.We got an accuracy of **75.38%** for this model.

```
1) KNeighborsClassifier

from sklearn.neighbors import KNeighborsClassifier

k_neigh = KNeighborsClassifier(n_neighbors=3)
k_neigh.fit(X_train, y_train)

KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
                     metric_params=None, n_jobs=None, n_neighbors=3, p=2,
                     weights='uniform')

y_pred=k_neigh.predict(X_test)
y_pred

array([1., 0., 2., ..., 1., 0., 0.])

from sklearn.metrics import accuracy_score
accuracy_score(y_test, y_pred)

0.7538482012154549
```

4. LINEAR DISCRIMINANT ANALYSIS

Discriminant analysis is a statistical technique used to classify observations into non-overlapping groups, based on scores on one or more quantitative predictor variables. For example, a doctor could perform a **discriminant analysis** to identify patients at high or low risk for stroke.We got an accuracy of **70.7%** for this model.

1) LinearDiscriminantAnalysis

[79]
```
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
```

[80]
```
lda = LinearDiscriminantAnalysis()
lda.fit(X_train, y_train)
```

```
LinearDiscriminantAnalysis(n_components=None, priors=None, shrinkage=None,
                           solver='svd', store_covariance=False, tol=0.0001)
```

[81]
```
y_pred=lda.predict(X_test)
y_pred
```

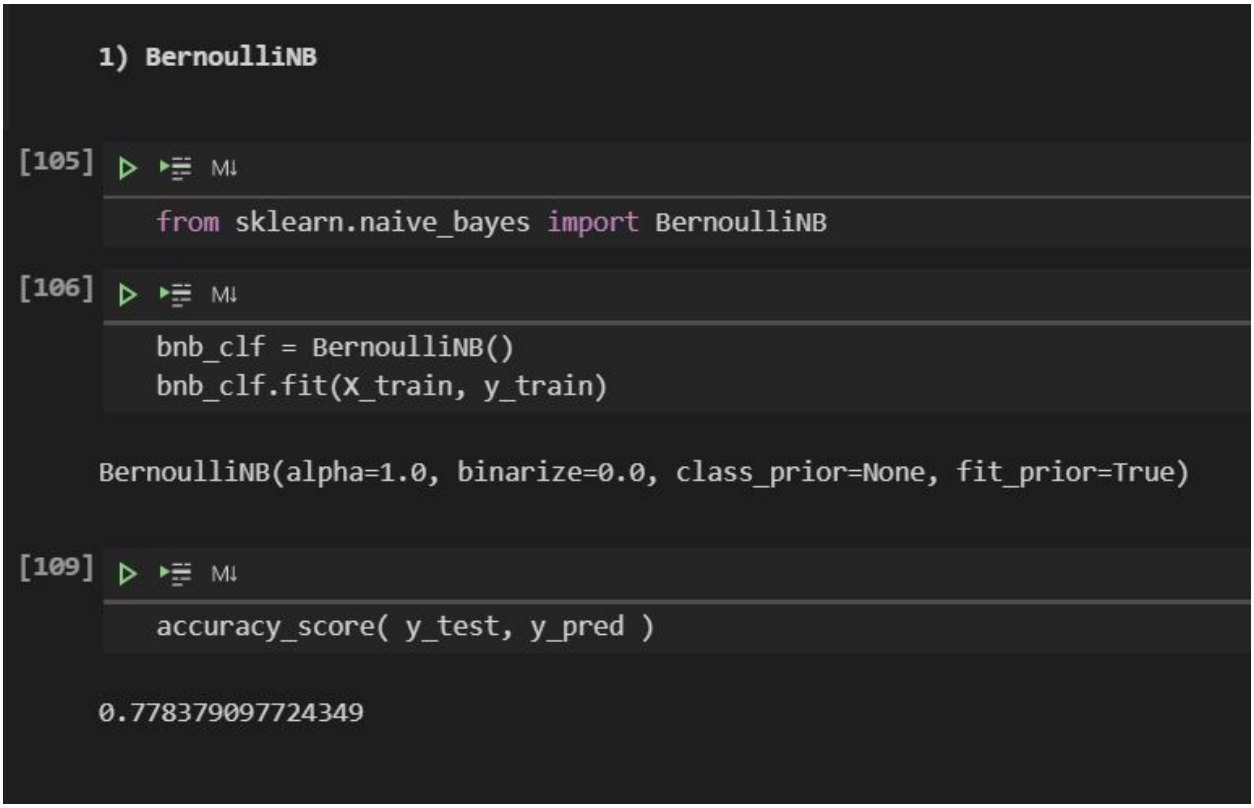
```
array([1., 0., 2., ..., 0., 0., 2.])
```

[82]
```
from sklearn.metrics import accuracy_score
accuracy_score(y_test, y_pred)
```

```
0.7071374706117198
```

5. BERNOULLI NAIVE BAYES

Bernoulli Naive Bayes is a variant of Naive Bayes. The main feature of Bernoulli Naive Bayes is that it accepts features only as binary values like true or false, yes or no, success or failure, 0 or 1 and so on. So when the feature values are binary we know that we have to use Bernoulli Naive Bayes classifier.We got an accuracy of **77.8%** for this model.



6. GAUSSIAN NAIVE BAYES

A **Gaussian** Naive Bayes algorithm is a special type of **NB** algorithm. It's specifically used when the features have continuous values. It's also assumed that all the features are following a **Gaussian** distribution i.e., normal distribution.We got an accuracy of **79.1%** for this model.

2) GaussianNB

[110]
```
from sklearn.naive_bayes import GaussianNB
```

[111]
```
gnb_clf = GaussianNB()
gnb_clf.fit(X_train, y_train)
```

```
GaussianNB(priors=None, var_smoothing=1e-09)
```

[112]
```
y_pred=gnb_clf.predict(X_test)
y_pred
```

```
array([1., 0., 2., ..., 1., 0., 2.])
```

[114]
```
accuracy_score( y_test, y_pred )
```

```
0.7916869981812537
```

7. SUPPORT VECTOR MACHINE

A **support vector machine** is a **machine** learning model that is able to generalise between two different classes if the set of labelled data is provided in the training set to the algorithm. The main function of the **SVM** is to check for that hyperplane that is able to distinguish between the two classes.We got an accuracy of **71.09%** for this model.

2) SVC

[119]
```
from sklearn.svm import SVC
from sklearn.pipeline import make_pipeline

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

[120]
```
model.fit(X_train, y_train)
```

```
SVC(C=1.0, break_ties=False, cache_size=200, class_weight='balanced', coef0=0.0,
    decision_function_shape='ovr', degree=3, gamma='scale', kernel='rbf',
    max_iter=-1, probability=False, random_state=None, shrinking=True,
    tol=0.001, verbose=False)
```

[121]
```
y_pred=model.predict(X_test)
y_pred
```

```
array([1., 0., 2., ..., 2., 0., 2.])
```

[122]
```
accuracy_score( y_test, y_pred )
```

```
0.7109524020760325
```

8. STOCHASTIC GRADIENT DESCENT

Stochastic gradient descent is an **iterative** method for optimizing an objective function with suitable smoothness properties. It can be regarded as a stochastic approximation of gradient descent optimization, since it replaces the actual gradient by an estimate thereof.We got an accuracy of **77.6%** for this model.

Stochastic Gradient Descent (SGD)

[123]
```
from sklearn.linear_model import LogisticRegression
from sklearn.linear_model import SGDClassifier
from sklearn.model_selection import KFold
from sklearn.metrics import accuracy_score
from sklearn.preprocessing import StandardScaler
```

[124]
```
model = SGDClassifier(loss="hinge", penalty="l2")
model.fit(X_train, y_train)
```

```
SGDClassifier(alpha=0.0001, average=False, class_weight=None,
              early_stopping=False, epsilon=0.1, eta0=0.0, fit_intercept=True,
              l1_ratio=0.15, learning_rate='optimal', loss='hinge',
              max_iter=1000, n_iter_no_change=5, n_jobs=None, penalty='l2',
              power_t=0.5, random_state=None, shuffle=True, tol=0.001,
              validation_fraction=0.1, verbose=0, warm_start=False)
```

[125]
```
y_pred=model.predict(X_test)
y_pred
```

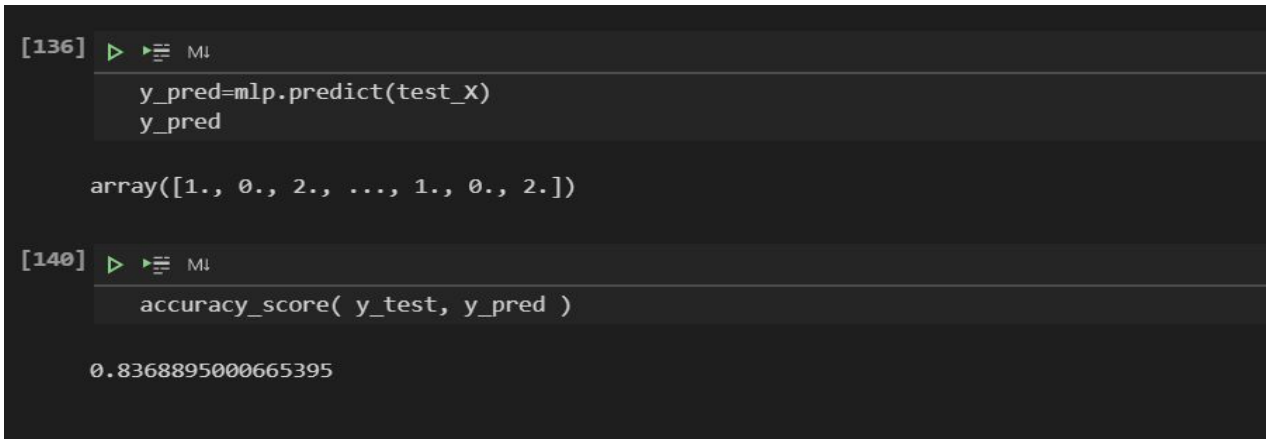
```
array([1., 0., 2., ..., 1., 0., 2.])
```

[126]
```
accuracy_score( y_test, y_pred )
```

```
0.7765159916603824
```

9. NEURAL NETWORK

A neural network is a series of algorithms that endeavors to recognize underlying relationships in a set of data through a process that mimics the way the human brain operates. In this sense, neural networks refer to systems of neurons, either organic or artificial in nature. We implemented a **Multi- Layered Perceptron** (MLP) for our purpose.We got an accuracy of **82%** for this model.



10. RANDOM FOREST

Random forests or random decision forests are an ensemble learning method for classification, regression and other tasks that operate by constructing a multitude of decision trees at training time and outputting the class that is the mode of the classes or mean/average prediction of the individual trees. We got an accuracy of **82.9%** for this model.

2. Random Forest

Random forest is an extension of bagged decision trees.

[146]
```
from sklearn.ensemble import RandomForestClassifier
```

[147]
```
seed = 7
num_trees = 100
max_features = 3
kfold = model_selection.KFold(n_splits=10, random_state=seed)
model = RandomForestClassifier(n_estimators=num_trees, max_features=max_features)
results = model_selection.cross_val_score(model, X_train, y_train, cv=kfold)
print(results.mean())
```

```
0.9995633944095959
```

[148]
```
model.fit(X_train, y_train)
```

```
RandomForestClassifier(bootstrap=True, ccp_alpha=0.0, class_weight=None,
                       criterion='gini', max_depth=None, max_features=3,
                       max_leaf_nodes=None, max_samples=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, n_estimators=100,
                       n_jobs=None, oob_score=False, random_state=None,
                       verbose=0, warm_start=False)
```

[149]
```
y_pred=model.predict(X_test)
y_pred
```

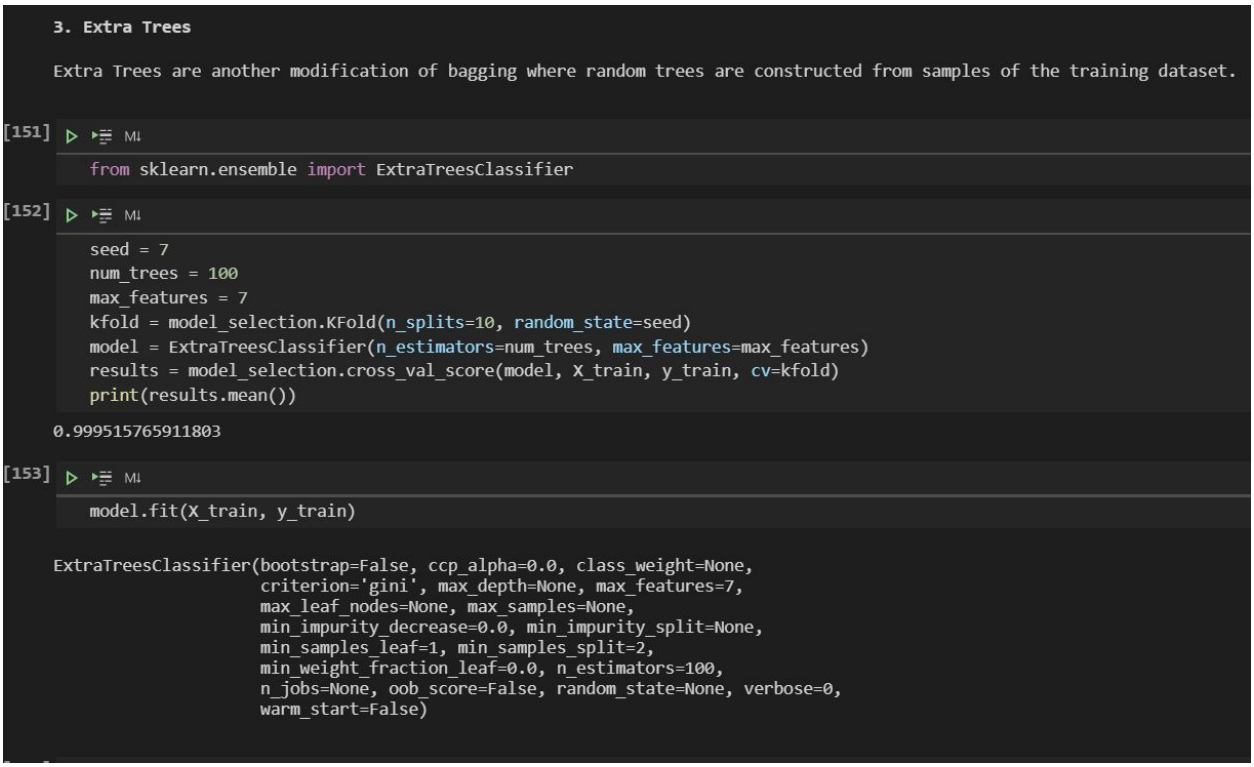
```
array([1., 0., 2., ..., 1., 0., 2.])
```

[150]
```
accuracy_score( y_test, y_pred )
```

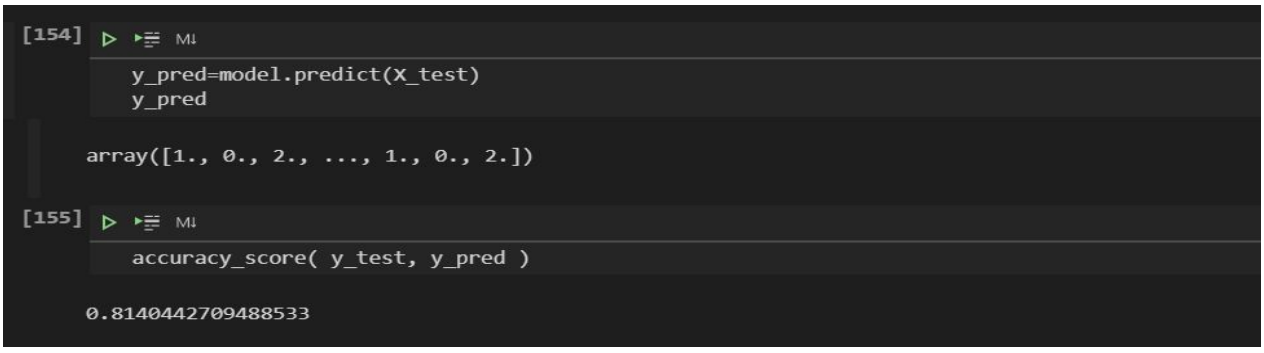
```
0.8297919531561904
```

11. EXTRA TREES

Extremely Randomized Trees Classifier(Extra Trees Classifier) is a type of ensemble learning technique which aggregates the results of multiple de-correlated decision trees collected in a "forest" to output it's classification result. In concept, it is very similar to a Random Forest Classifier and only differs from it in the manner of construction of the decision trees in the forest.We got an accuracy of **81.4%** for this model.



```
3. Extra Trees

Extra Trees are another modification of bagging where random trees are constructed from samples of the training dataset.

[151] ▷ M↓
from sklearn.ensemble import ExtraTreesClassifier

[152] ▷ M↓
seed = 7
num_trees = 100
max_features = 7
kfold = model_selection.KFold(n_splits=10, random_state=seed)
model = ExtraTreesClassifier(n_estimators=num_trees, max_features=max_features)
results = model_selection.cross_val_score(model, X_train, y_train, cv=kfold)
print(results.mean())

0.999515765911803

[153] ▷ M↓
model.fit(X_train, y_train)

ExtraTreesClassifier(bootstrap=False, ccp_alpha=0.0, class_weight=None,
                     criterion='gini', max_depth=None, max_features=7,
                     max_leaf_nodes=None, max_samples=None,
                     min_impurity_decrease=0.0, min_impurity_split=None,
                     min_samples_leaf=1, min_samples_split=2,
                     min_weight_fraction_leaf=0.0, n_estimators=100,
                     n_jobs=None, oob_score=False, random_state=None, verbose=0,
                     warm_start=False)
```



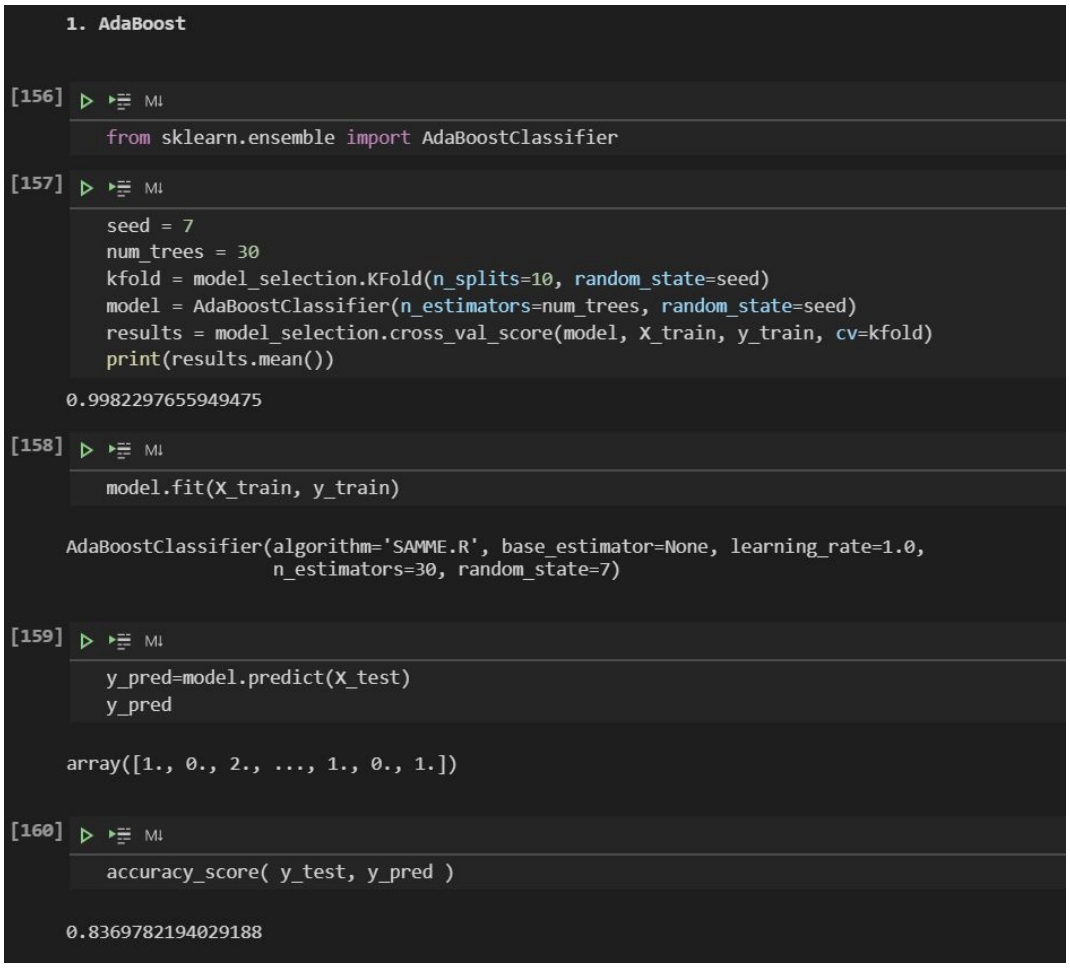
```
[154] ▷ M↓
y_pred=model.predict(X_test)
y_pred

array([1., 0., 2., ..., 1., 0., 2.])

[155] ▷ M↓
accuracy_score( y_test, y_pred )

0.8140442709488533
```

We used some **Boosting algorithms** too for this purpose. Boosting ensemble algorithms creates a sequence of models that attempt to correct the mistakes of

the models before them in the sequence.Once created, the models make predictions which may be weighted by their demonstrated accuracy and the results are combined to create a final output prediction.

1. ADABOOST

AdaBoost was perhaps the first successful boosting ensemble algorithm. It generally works by weighting instances in the dataset by how easy or difficult they are to classify, allowing the algorithm to pay more or less attention to them in the construction of subsequent models. We got an accuracy of **83.69%** for this model.



```
1. AdaBoost

[156] M↓
from sklearn.ensemble import AdaBoostClassifier

[157] M↓
seed = 7
num_trees = 30
kfold = model_selection.KFold(n_splits=10, random_state=seed)
model = AdaBoostClassifier(n_estimators=num_trees, random_state=seed)
results = model_selection.cross_val_score(model, X_train, y_train, cv=kfold)
print(results.mean())

0.9982297655949475

[158] M↓
model.fit(X_train, y_train)

AdaBoostClassifier(algorithm='SAMME.R', base_estimator=None, learning_rate=1.0,
                   n_estimators=30, random_state=7)

[159] M↓
y_pred=model.predict(X_test)
y_pred

array([1., 0., 2., ..., 1., 0., 1.])

[160] M↓
accuracy_score( y_test, y_pred )

0.8369782194029188
```

2. STOCHASTIC GRADIENT BOOSTING

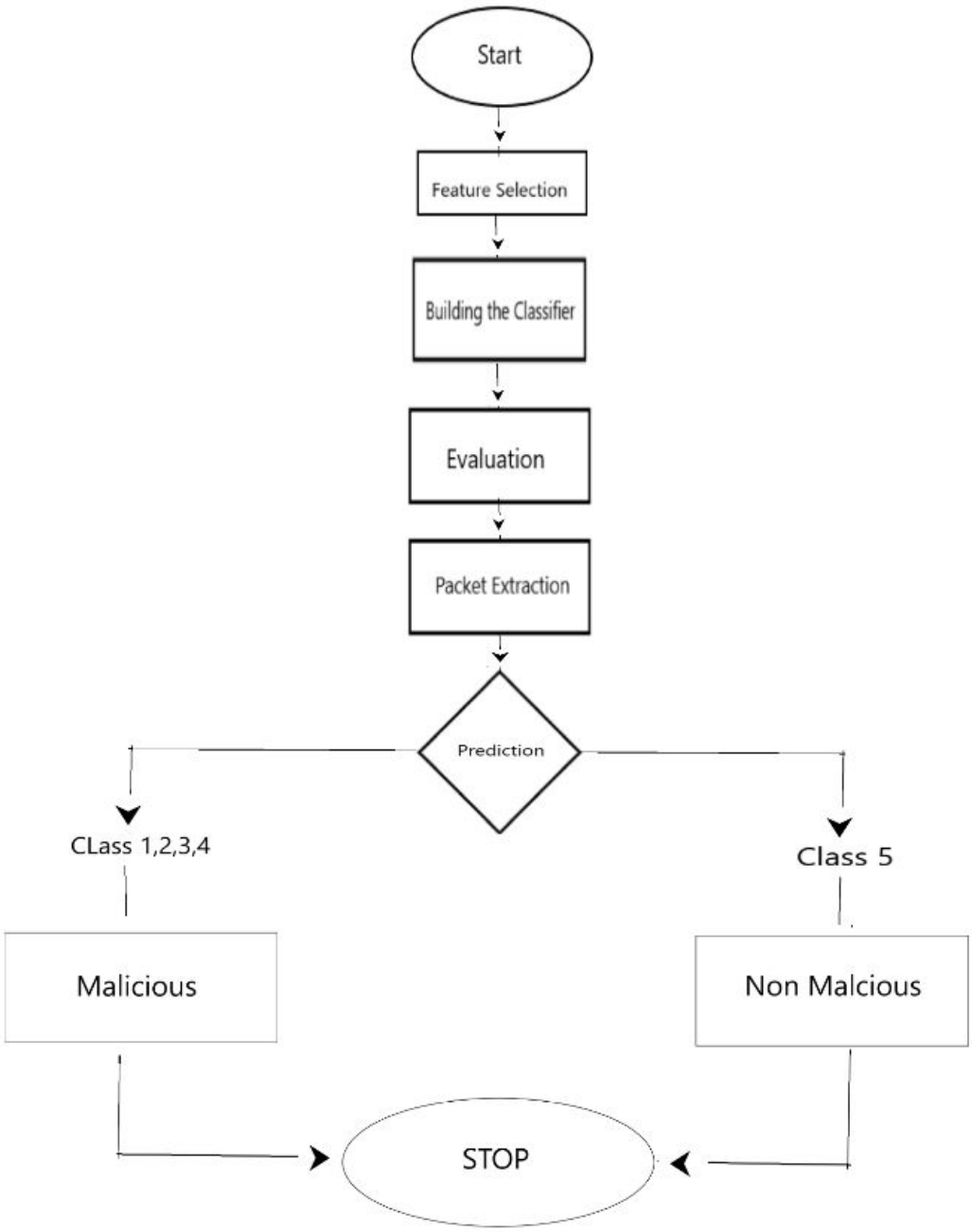
Stochastic Gradient Boosting (also called Gradient Boosting Machines) are one of the most sophisticated ensemble techniques. It is also a technique that is proving to be perhaps one of the best techniques available for improving performance via ensembles.We got an accuracy of **81.25%** for this model.

```
2. Stochastic Gradient Boosting

[161]
from sklearn.ensemble import GradientBoostingClassifier

[162]
seed = 7
num_trees = 100
kfold = model_selection.KFold(n_splits=10, random_state=seed)
model = GradientBoostingClassifier(n_estimators=num_trees, random_state=seed)
results = model_selection.cross_val_score(model, X_train, y_train, cv=kfold)
print(results.mean())

0.999023588384522

[163]
model.fit(X_train, y_train)

GradientBoostingClassifier(ccp_alpha=0.0, criterion='friedman_mse', init=None,
                           learning_rate=0.1, loss='deviance', max_depth=3,
                           max_features=None, max_leaf_nodes=None,
                           min_impurity_decrease=0.0, min_impurity_split=None,
                           min_samples_leaf=1, min_samples_split=2,
                           min_weight_fraction_leaf=0.0, n_estimators=100,
                           n_iter_no_change=None, presort='deprecated',
                           random_state=7, subsample=1.0, tol=0.0001,
                           validation_fraction=0.1, verbose=0,
                           warm_start=False)
```

```
[164]
y_pred=model.predict(X_test)
y_pred

array([1., 0., 2., ..., 1., 0., 1.])

[165]
accuracy_score( y_test, y_pred )

0.8125360422304041
```

Since clearly **Logistic Regression** has given us the highest accuracy of about **93.2%** out of all the various classifiers mentioned above, thus we select this model for connecting with our **Web- Application.**

FLOWCHART:



4. Proposed System Analysis and Design

Backend Code

app.py

```python
import numpy as np
from flask import Flask, request, jsonify, render_template
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:]

```

```
        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)]
    predict = model.predict(final_features)

    if predict==0:
        output='Normal'
    elif predict==1:
        output='DOS'
    elif predict==2:
        output='PROBE'
    elif predict==3:
        output='R2L'
    else:
        output='U2R'

    return render_template('prediction.html', output=output)

if __name__ == "__main__":
    app.run()
```

Input Page Screenshot

5. Results and Discussion:

Nowadays, almost every system is prone to attacks. There is a need to increase the security in our daily use systems. This can be achieved by using Intrusion Detection Systems. It helps us in detecting malicious activities efficiently. Another type of Intrusion detection system is used to detect anomalies. But, the current software which detects anomalies detects high number false positives which leads to an increase in the rate of false alarms. Also the results obtained were highly inaccurate and in many cases some types of attacks were not able to be detected. Therefore, we conducted an experiment to assess the accuracies and detection rates of various algorithms using the NSL-KDD dataset. According to our results we were able to conclude that our approach where we calculated accuracy of 10 models and used the best one out of all them, has performed exceptionally well and has shown very low rates of false alarms. We have taken into consideration the factors like Accuracy, Precision, Recall and F-Measure to select our approach.

6. Future Work

In order to get more precise results, for finalising the best classifier, along with their Accuracy, we could calculate other factors like their Precision, Recall and F-Measure too. Once we get all four of these values for each classifier, we'll congregate them all together, to get a normalized score, which will be compared among all the models. The one with the highest score, would be selected.

7. References (API format):

[1] Koushal Kumar, International Journal of Computer Science & Communication Networks,Vol 6(3),153-174, "Intrusion Detection using Soft Computing Techniques",15th July 2020.

[2] Jacob Neyole,Yusuf Muchelule,"A Review of Intrusion Detection Systems",International Journal of Computer Science and Information Technology Research,October 2017 .

[3] M. Ali Aydın, A. Halim Zaim, K. Gökhan Ceylan," A hybrid intrusion detection system design for computer network security.",Elsevier Ltd.,May 2019.

[4] Tao Xia, Guangzhi Qu, Salim Hariri, Internet Technology Laboratory,ECE, University of Arizona,"An Efficient Network Intrusion Detection Method based on Information Theory and Genetic Algorithm.

[5] M. Botha, R. Solms, "Utilizing Neural Networks for Effective Intrusion Detection", ISSA, 2004.

[6] D. Zamboni, "Using Internal Sensors for Computer Intrusion Detection". Center for Education and Research in Information Assurance and Security, Purdue University. August 2001.

[7] W. Lu, I. Traore, "Detecting New Forms of Network Intrusion Using Genetic Programming".Computational Intelligence, vol. 20, pp. 3, Blackwell Publishing, Malden, pp. 475-494, 2004.

[8] Srinivas Mukkamala, Andrew H. Sung, Ajith Abraham, "Intrusion detection using an ensemble of intelligent paradigms", Journal of Network and Computer Applications, Volume 28, Issue 2, April 2005.

[9] S. Kumar, "Classification and Detection of Computer Intrusions", Purdue University, 1995

[10] C. Chang and C. J. Lin, LIBSVM, "A Library for Support Vector Machines", the use of LIBSVM, 2009.

[11] Rung-Ching Chen, Kai-Fan Cheng and Chia-Fen Hsieh, "Using Rough Set and Support Vector Machine for Network Intrusion Detection", International Journal of Network Security & Its Applications (IJNSA), Vol 1, No 1, 2009.

[12] Phurivit Sangkatsanee, Naruemon Wattanapongsakorn and Chalermpol Charnsripinyo, "Real-time Intrusion Detection and Classification", IEEE network, 2009.

[13] Liberios Vokorokos, Alzebeta Kleniova, "Network Security on the Intrusion Detection System Level", IEEE network, 2004.

[14] Thomas Heyman, Bart De Win, Christophe Huygens, and Wouter Joosen, "Improving Intrusion Detection through Alert Verification", IEEE Transaction on Dependable and Secure Computing, 2004.

[15] T. Lin and C.J. Lin, "A study on sigmoid kernels for SVM and the training of non- PSD kernels by SMO-type methods", Technical report, Department of Computer Science, National Taiwan University, 2003.

[16] Lindsay I Smith, "A tutorial on Principal Components Analysis", 2002.

[17] Vapnik, "The Nature of Statistical Learning Theory", Springer-Verlag, New York, 1995.

[18] Rietta FS, Application layer intrusion detection for SQL injection, In Proceedings of the 44th annual Southeast regional conference; pages 531- 536, 2006.

[19] Meisam S.A, Najjar MAA, A Distributed Multi-Approach Intrusion Detection System, In SIN '10 Proceedings of the 3rd international conference on Security of information and networks, pages 238-244, 2010.

[20] Mookhey KK, Burghate N, Detection of SQL Injection and Cross-site Scripting Attack, http://www.symantec.com/connect/articles/detectionsql-injection-and-cross-site-scripting-attacks, 2013-03-08.

[21]M. V. Mahoney, Network traffic anomaly detection based on packet bytes, in: SAC '03: Proceedings of the 2003 ACM symposium on Applied computing, ACM, New York, NY, USA, 2003, pp. 346–350.

[22] K. Wang, S. J. Stolfo, Anomalous payload-based network intrusion detection, in: E. Jonsson, A. Valdes, M. Almgren (Eds.), RAID, Vol. 3224 of Lecture Notes in Computer Science, Springer, 2004, pp. 203–222.