# Detecting Malicious activities over Network using machine learning

**Your Name here**

**Your Student ID**

**Semester**

Contents

[Detecting Malicious activities over Network using machine learning 1](#_Toc103894356)

[Introduction: 3](#_Toc103894357)

[Objective: 3](#_Toc103894358)

[Methodology: 3](#_Toc103894359)

[Analysis of Results/Findings: 3](#_Toc103894360)

[Libraries: 4](#_Toc103894361)

[Exploratory Data Analysis (EDA): 4](#_Toc103894362)

[Model Building: 6](#_Toc103894363)

[References 8](#_Toc103894364)

# Introduction:

A network is a set of computers and users connected, their connectivity with each other is because they share any kind of resources over the network easily and time efficiently. We are connected to the world via the internet, 5 billion people are using the internet and they are connected. There are many types of networks, LAN, WAN, Military networks, etc. This is not always possible that every user of the network uses the network ethically, many people use the internet for various unethical purposes, like by stealing someone’s identity, their bank details, etc. To secure a network, there are many ways and techniques which are used by network administrators to detect malicious activities over the network. One of the techniques is well known as Intrusion Detection, this is a device or software which is used to detect any abnormal activity over the network. I have a dataset of a military network, using the dataset, I will train different machine learning models which will be able to detect any malicious activity over the networks.

# Objective:

The main objective of this project is to detect abnormal activities over the network, I have a dataset that is collected over a military network. The dataset has many features like protocol type, flag, src bytes, DST bytes, etc. using these features, I will train different machine learning models to detect any abnormal activity over the network.

# Methodology:

The overall methodology of this project is given below:

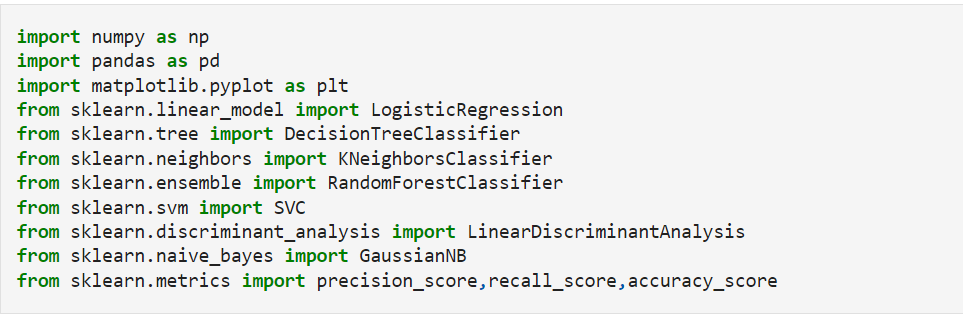
# Analysis of Results/Findings:

In this section, we will look at all the code I have performed and show the results I get. The goal of this project is to detect any abnormal and malicious activity over a network using machine learning. The whole project is divided into the following.

* Libraries
* Exploratory data analysis
* Model Building

## Libraries:

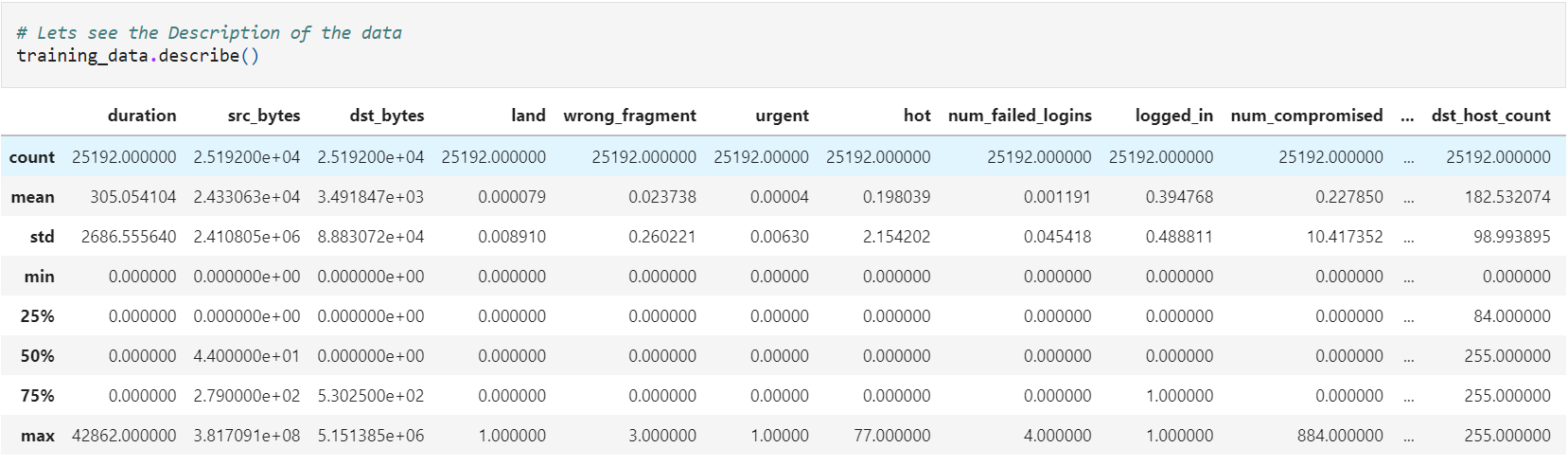
Libraries are the collection of different resources which are used to program something. The first thing in any machine learning or data science project is to import all the required libraries. I am importing the following libraries.



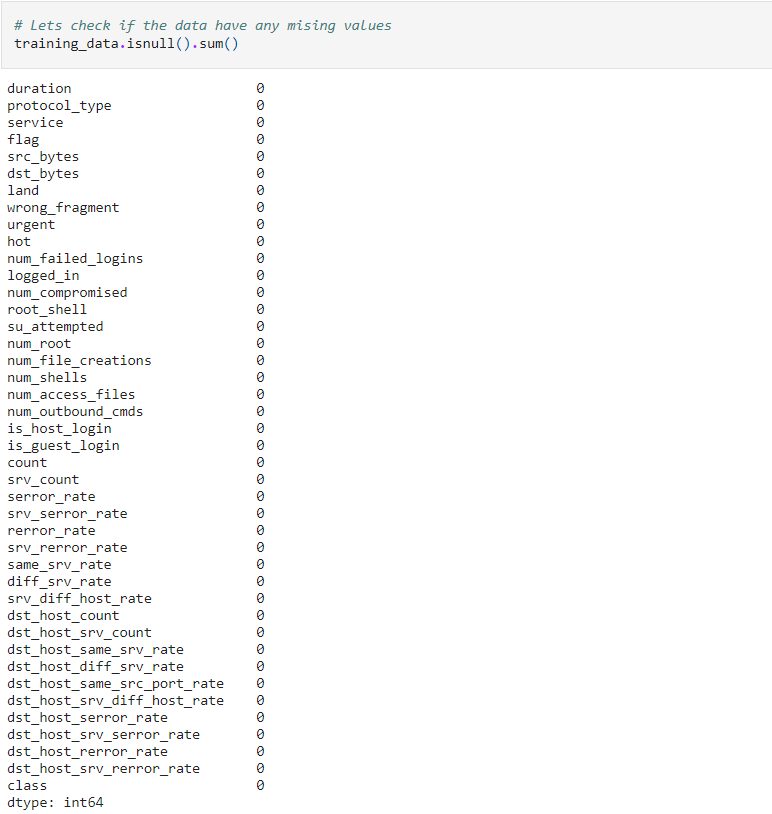
I imported NumPy library which is used to deal with numerical arrays, pandas for data exploration and processing, matplotlib for plotting a different graph of the data, I have imported Logistic regression, decision tree, KNN, Random Forest, SVC, and Linear Discriminate Analysis models from sklearn library, I am also importing the precision score, recall and accuracy score from sklearn library.

## Exploratory Data Analysis (EDA):

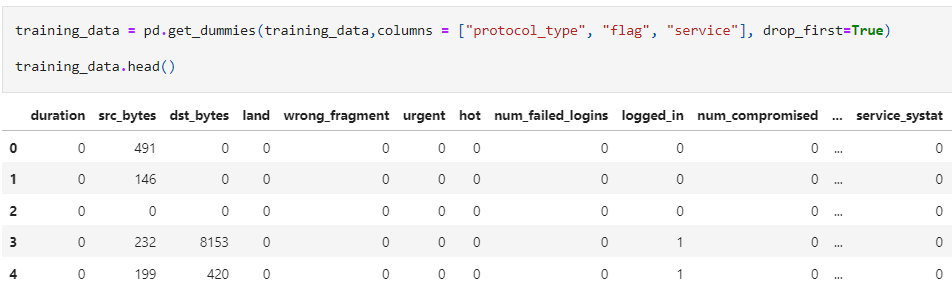
Exploratory data analysis is the process to analyze and explore the data to extract and summarize their main characteristics. This is often called data pre-processing. The real-world data may have huge garbage values and features, to clean the data, I use EDA and prepare the data for final modeling. In EDA, I am performing the following steps.



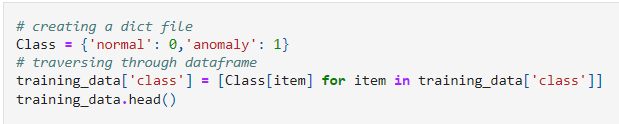
The first thing I did is to check the whole description of the data, the describe function gives us real insights into the data while exploring it. It gives us the mean, std, counts, etc of the data. The description of our data is shown in the above image.



Missing values in a dataset are the main issue we face; these missing values are those values that are not entered in the database while collecting the data or due to any reason. There are many ways to handle these missing values, we use some statistical methods like mean, median, and mode to fill these values and sometimes we remove them. I have checked the missing values in our dataset by using a pandas function isnull(), after running the above script, I got zero missing values in all the features in our dataset. So, our dataset has no missing values.



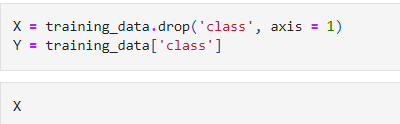
Our dataset has some categorical features, like protocol\_type, flag, and service features have string data. Machine learning models don’t understand categorical features, the solution is to convert the categorical features into numerical features. This is called one-hot-encoding, by applying this technique, all the string data points are converted into numerical values by giving each string data point a specific numerical value. In the script shown above, I am using a pandas get\_dummies() method which will take all the categorical features and convert them into numerical values.

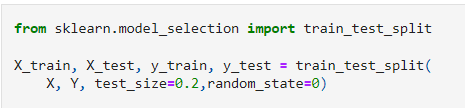


I also have a Class feature in our data that shows that the activity over the network is normal or abnormal, this feature is also categorical, and I will convert it into numerical values. For example, 0 will be assigned to Normal and 1 will be Abnormal.

* 0 for Normal
* 1 for anomaly

The above script will go through the whole feature and convert the categories into numerical values.





In the above image, I am splitting the data into test and train sets, the training set will be used to train the models and the testing data will be used to test our models. I am storing the independent features in the X variable and the dependent variable in Y. then I am splitting the data in the ratio of 80:20, which means 80% data will be used for training purposes and 20% for testing purposes.

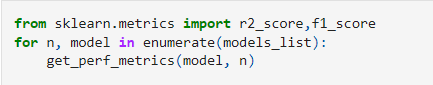
## Model Building:

After performing data processing and splitting the data into test and train sets, the next thing is to build our machine learning models. I have imported all the models from the sklearn library, I am training the following models and then I will give the accuracy comparison.

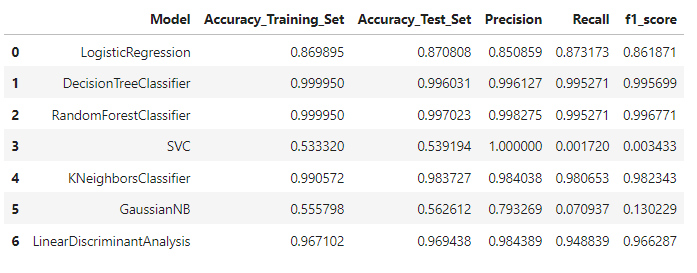
I will train the following models

* Logistic Regression
* Decision Tree Classifiers
* Random Forests
* Support Vector Machines
* K-Nearest Neighbors
* Gaussian Naive Bayes
* LinearDiscriminantAnalysis

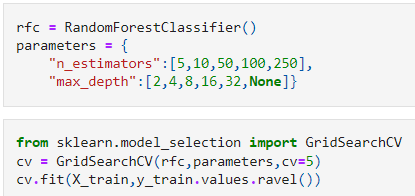




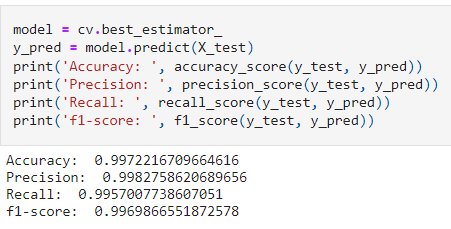
In the image above, the function called get\_perf\_metrics() will take each model from the model list below and the testing and training data as parameters, then it will train the model and store the model accuracy, and recall score and precision in a DataFrame. This way, all the models I have in the list will get trained on the training data and tested on the testing data, and stored the results in the table. After running the above lines of code, I got the following table as a result.



I can see that except for SVC, every model performed very well, the highest accuracies are given by Decision Tree, Random Forest, and KNN which is 99%. Now I will take Random Forest and perform the parameter tunning to see if I can further increase the accuracy?



I have used GridSearchCV to perform parameters tuning of Random Forest, I have a list of different parameters, and passing the list to the GridSearchCV, will give us the best parameters.



After getting the best parameters, I trained the Random Forest classifier again and got an accuracy of 99%.

# References

Dataset: <https://www.kaggle.com/datasets/sampadab17/network-intrusion-detection>

## Appendix:

**import** numpy **as** np

**import** pandas **as** pd

**import** matplotlib.pyplot **as** plt

**from** sklearn.linear\_model **import** LogisticRegression

**from** sklearn.tree **import** DecisionTreeClassifier

**from** sklearn.neighbors **import** KNeighborsClassifier

**from** sklearn.ensemble **import** RandomForestClassifier

**from** sklearn.svm **import** SVC

**from** sklearn.discriminant\_analysis **import** LinearDiscriminantAnalysis

**from** sklearn.naive\_bayes **import** GaussianNB

**from** sklearn.metrics **import** precision\_score,recall\_score,accuracy\_score

*# importing the data into pandas DataFrame*

training\_data **=** pd**.**read\_csv('Train\_Data.csv')

training\_data**.**head() *# Showing the first 5 rows*

*# Lets see the Description of the data*

training\_data**.**describe()

*# Lets check the statistical information of the data*

training\_data**.**info

*# Lets check if the data have any mising values*

training\_data**.**isnull()**.**sum()

training\_data **=** pd**.**get\_dummies(training\_data,columns **=** ["protocol\_type", "flag", "service"], drop\_first**=True**)

training\_data**.**head()

*# Lets check the class feature*

training\_data['class']**.**unique()

*# creating a dict file*

Class **=** {'normal': 0,'anomaly': 1}

*# traversing through dataframe*

training\_data['class'] **=** [Class[item] **for** item **in** training\_data['class']]

training\_data**.**head()

training\_data['class']**.**unique()

*# lets check how many rows are normal and anomalies*

training\_data['class']**.**value\_counts()

X **=** training\_data**.**drop('class', axis **=** 1)

Y **=** training\_data['class']

**from** sklearn.model\_selection **import** train\_test\_split

X\_train, X\_test, y\_train, y\_test **=** train\_test\_split(

X, Y, test\_size**=**0.2,random\_state**=**0)

df\_perf\_metrics **=** pd**.**DataFrame(columns**=**[

'Model', 'Accuracy\_Training\_Set', 'Accuracy\_Test\_Set', 'Precision',

'Recall', 'f1\_score'

])

models\_trained\_list **=** []

**def** get\_perf\_metrics(model, i):

*# model name*

model\_name **=** type(model)**.**\_\_name\_\_

print("Training {} model..."**.**format(model\_name))

*# Fitting of model*

model**.**fit(X\_train, y\_train)

print("Completed {} model training."**.**format(model\_name))

*# Predictions*

y\_pred **=** model**.**predict(X\_test)

*# Add to ith row of dataframe - metrics*

df\_perf\_metrics**.**loc[i] **=** [

model\_name,

model**.**score(X\_train, y\_train),

model**.**score(X\_test, y\_test),

precision\_score(y\_test, y\_pred),

recall\_score(y\_test, y\_pred),

f1\_score(y\_test, y\_pred),

]

print("Completed {} model's performance assessment."**.**format(model\_name))

*# the model list*

models\_list **=** [LogisticRegression(),

DecisionTreeClassifier(),

RandomForestClassifier(),

SVC(),

KNeighborsClassifier(),

GaussianNB(),LinearDiscriminantAnalysis()

]

**from** sklearn.metrics **import** r2\_score,f1\_score

**for** n, model **in** enumerate(models\_list):

get\_perf\_metrics(model, n)

df\_perf\_metrics

rfc **=** RandomForestClassifier()

parameters **=** {

"n\_estimators":[5,10,50,100,250],

"max\_depth":[2,4,8,16,32,**None**]}

**from** sklearn.model\_selection **import** GridSearchCV

cv **=** GridSearchCV(rfc,parameters,cv**=**5)

cv**.**fit(X\_train,y\_train**.**values**.**ravel())

**def** display(results):

print(f'Best parameters are: {results**.**best\_params\_}')

print("\n")

mean\_score **=** results**.**cv\_results\_['mean\_test\_score']

std\_score **=** results**.**cv\_results\_['std\_test\_score']

params **=** results**.**cv\_results\_['params']

**for** mean,std,params **in** zip(mean\_score,std\_score,params):

print(f'{round(mean,3)} + or -{round(std,3)} for the {params}')

display(cv)

model **=** cv**.**best\_estimator\_

y\_pred **=** model**.**predict(X\_test)

print('Accuracy: ', accuracy\_score(y\_test, y\_pred))

print('Precision: ', precision\_score(y\_test, y\_pred))

print('Recall: ', recall\_score(y\_test, y\_pred))

print('f1-score: ', f1\_score(y\_test, y\_pred))

df1\_perf\_metrics **=** pd**.**DataFrame(columns**=**[

'Model', 'Accuracy\_Training\_Set', 'Accuracy\_Test\_Set', 'Precision',

'Recall', 'f1\_score'

])

**def** get\_perf\_metrics\_t(model):

model **=** cv**.**best\_estimator\_

model\_name **=**RandomForestClassifier()

print('Training RandomForestClassifier()')

model**.**fit(X\_train, y\_train)

print('Completed RandomForestClassifier()')

y\_pred **=** model**.**predict(X\_test)

df1\_perf\_metrics**.**loc[0] **=** [

model\_name,

model**.**score(X\_train, y\_train),

model**.**score(X\_test, y\_test),

precision\_score(y\_test, y\_pred),

recall\_score(y\_test, y\_pred),

f1\_score(y\_test, y\_pred),

]

print("Completed RandomForestClassifier() model's performance assessment.")

get\_perf\_metrics\_t(model)

df1\_perf\_metrics

*# lets perform predictions on testing data*

pred **=** model**.**predict(X\_test)

pred