1. Introduction

A network intrusion is any unauthorized activity on a computer network. Software to detect network intrusions aim at protecting a computer network from unauthorized users, including perhaps insiders. With the enormous growth of computer networks usage and the huge increase in the number of applications running on top of it, network security is becoming increasingly more important. Almost all the computer systems suffer from security vulnerabilities which are both technically difficult and economically costly to be solved by the manufacturers. Therefore, the role of Intrusion Detection Systems (IDSs), as special-purpose devices to detect anomalies and attacks in the network, is becoming more important.

The research in the intrusion detection field has been mostly focused on anomaly-based and misuse-based detection techniques for a long time. While misuse-based detection is generally favoured in commercial products due to its predictability and high accuracy, in academic research anomaly detection is typically conceived as a more powerful method due to its theoretical potential for addressing novel attacks.

In this project, we will build a network intrusion detector, a predictive model capable of distinguishing between ‘’bad’’ connections, called as intrusions or attacks, and ‘’good’’ or normal connections and also which algorithm predicts more accurately that the attack has happened. Here we are using a benchmark dataset KDDCup 99 Dataset which includes a wide variety of intrusions simulated in a military network environment. The data used to build the Intrusion detector was prepared and managed by MIT Lincoln Labs.

1.1 Problem Statement

Intrusion detection System using Naïve Bayes Classifier and KNN Algorithms on KDDCup99 Data Set

1.2 Objectives

The objective was to survey and evaluate research in intrusion detection.The cyber attacks are usually aimed at accessing, changing or destroying sensitive information.So analysing which algorithm will help better to find that attack has happened, and save information from getting misused.

1.3 Scope

Our Project will be capable of detecting intrusion in network using benchmark dataset

2. Literature Survey

2.1 Existing System

Panda, Mrutyunjaya & Patra, Manas. (2007). Network intrusion detection using naive bayes. 7.

i. This paper gives a comparative study of several anomaly detection schemes for identifying novel network intrusion detections.
ii. Presented experimental results on KDDCup'99 data set. Experimental results have demonstrated that our naïve bayes classifier model is much more efficient in the detection of network intrusions, compared to the neural network based classification techniques.

Denial of Service Intrusion Detection System (IDS) Based on Naïve Bayes Classifier using NSL KDD and KDD Cup 99 Datasets

i. This paper will introduce Naïve Bayes (NB) Classifier supported by discrete the continuous feature and feature selection methods to classify network events as an attack (DoS, Probe, R2L and U2R) or normal.

ii. The performance of the proposed system was evaluated by using KDD 99 CUP and NSL KDD Datasets.

iii. And proposal improves the performance of NIDS in term of accuracy and detecting DOS attack, where it detected 94%, 97% and 98% of DoS attacks for three experimental test datasets in KDD Cup 99 dataset when used twelve features selected by gain ratio

Evaluation of Different Data Mining Algorithms with KDD CUP 99 Data Set

i. In this paper, a comprehensive set of 20 algorithms will be evaluated on the KDD dataset being tried to detect attacks on the four attack categories: Probe, DoS, U2R, R2L.

ii. This survey will be the measure of researchers to depend on to compare their results they get from the use of KDD 99 with Data Mining algorithms with the best results of the survey and thus the comparison easier and faster

iii. The decision tree to get a better intrusion detection rates up higher than the 96% level and low false alerts from the rest of classifier data mining algorithms.

2.2 Proposed System

The proposed framework for the network intrusion detection of attacks that uses two machine learning algorithms, they are Naïve Bayes and K-Nearest Neighbor algorithms to detect the attacks. The framework comprises the following tasks: data pre-processing, feature reduction, machine learning model, and performance evaluation.

Data preprocessing is done using One-hot encoding which converts the categorical features into binary features. The KDDCup 99 dataset consists of 41 features of which 10 features have been extracted using PCA(Principal Component Analysis). On these reduced features we train and test the dataset and get the accuracy and compare these accuracies to find out which algorithm gives more accurate prediction of attack.

10 Extracted Features are

	Feature Name	Description
1	Label	Anomaly or normal behaviour
2	Duration	Duration of the active connection.
3	Protocol_type	Connection protocol (e.g. tcp, udp).
4	Service	Destination service (e.g. telnet, ftp)
5	Flag	Status flag of the connection
6	src bytes	Bytes sent from source to destination
7	dst_bytes	Bytes sent from destination to source
8	Land	One if the connection is from/to the same host/port; Otherwise 0
9	wrong_fragment	No. of wrong fragments
10	Urgent	No. of urgent packets

3. Requirement Specification

3.1 Functional Requirements

Introduction- Intrusion Detection Systems (IDS) become necessary to protect data from intruders and reduce the damage of the information system and networks especially in cloud environment which is next generation Internet based computing system that supplies customizable services to the end user to work or access to the various cloud applications.

Inputs- In our project we are using Bench Mark Data Set KDD Cup 99 data set as input

Processing- One-Hot-Encoding is used to transform all categorical features into binary features. The One-Hot-encoding takes a matrix of integers, denoting the values on by categorical features. The output will be a sparse matrix where each column corresponds to one possible value of one feature. Therefore the features first need to be transformed with Label Encoder, to transform every category to a number. In the second step, we will be applying various machine and deep learning algorithms. These algorithms analyze the large datasets and mechanism which show the intrusion in the given KDD Cup dataset with different accuracies.

Sl.No	Algorithm Name	Accuracy
1.	Naïve Bayes Algorithm	99.72 %
2.	K Nearest Neighbor Algorithm	99.93 %

3.2 Non Functional Requirements

Type	Description
Performance	1.The system should be able to classify anomalies and normal packets with the accuracy of more than 95%. 2.The pre-processing time of the intrusion detection system should be within seconds.
Usability	1.The system should be available all the time.

4. Design

4.1 Architectural Design

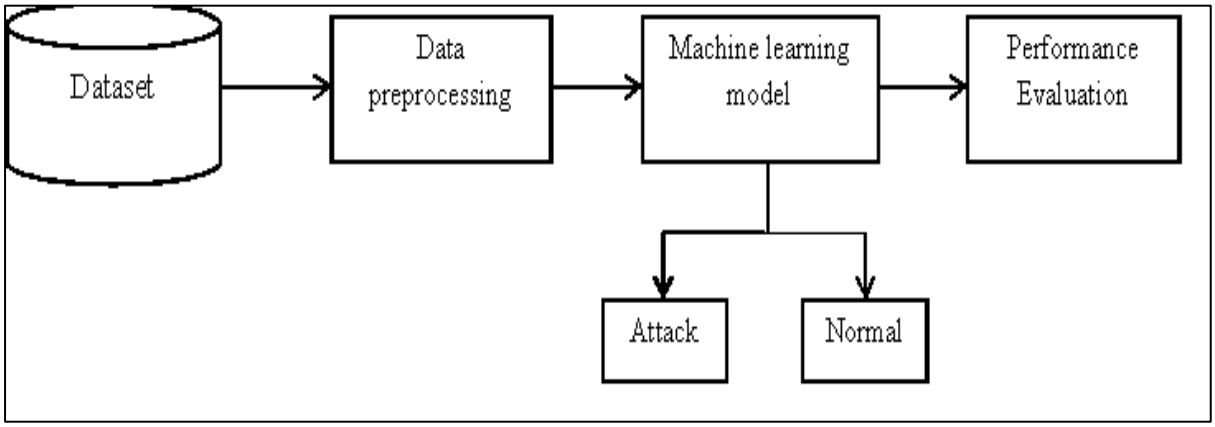

Figure 1 Architectural diagram

4.2 Use Case Diagram

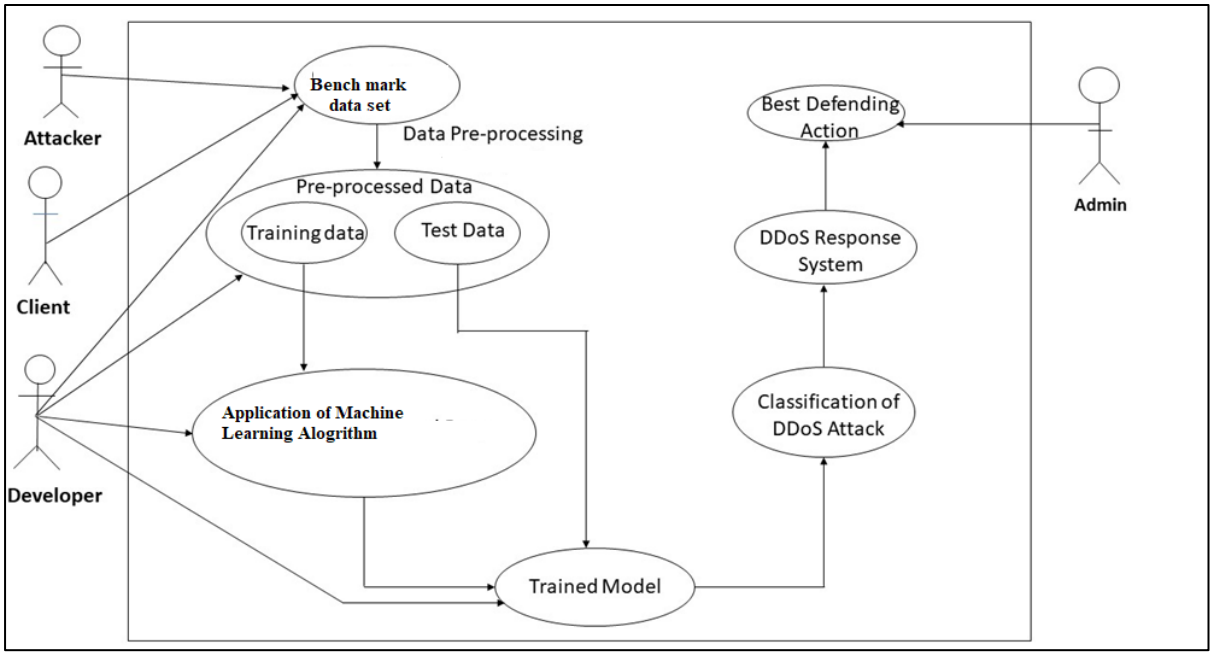

Figure 2 Use Case Diagram

4.3 Data Flow Diagram

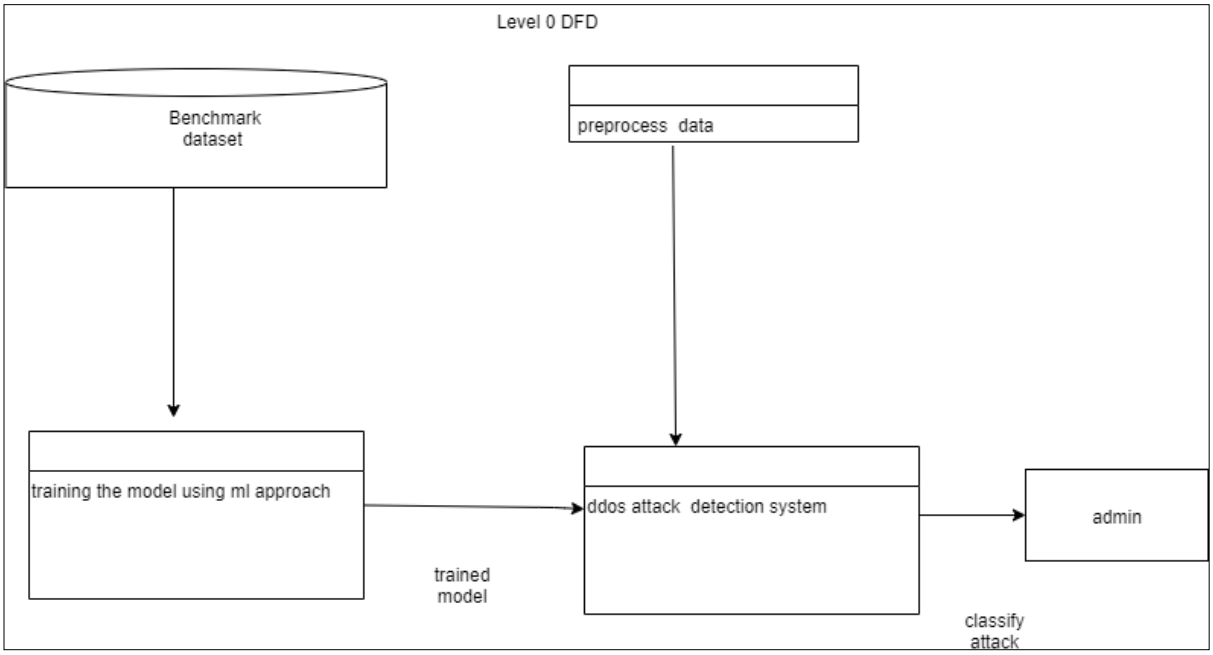

Figure 3.1 Level 0 Data Flow diagram

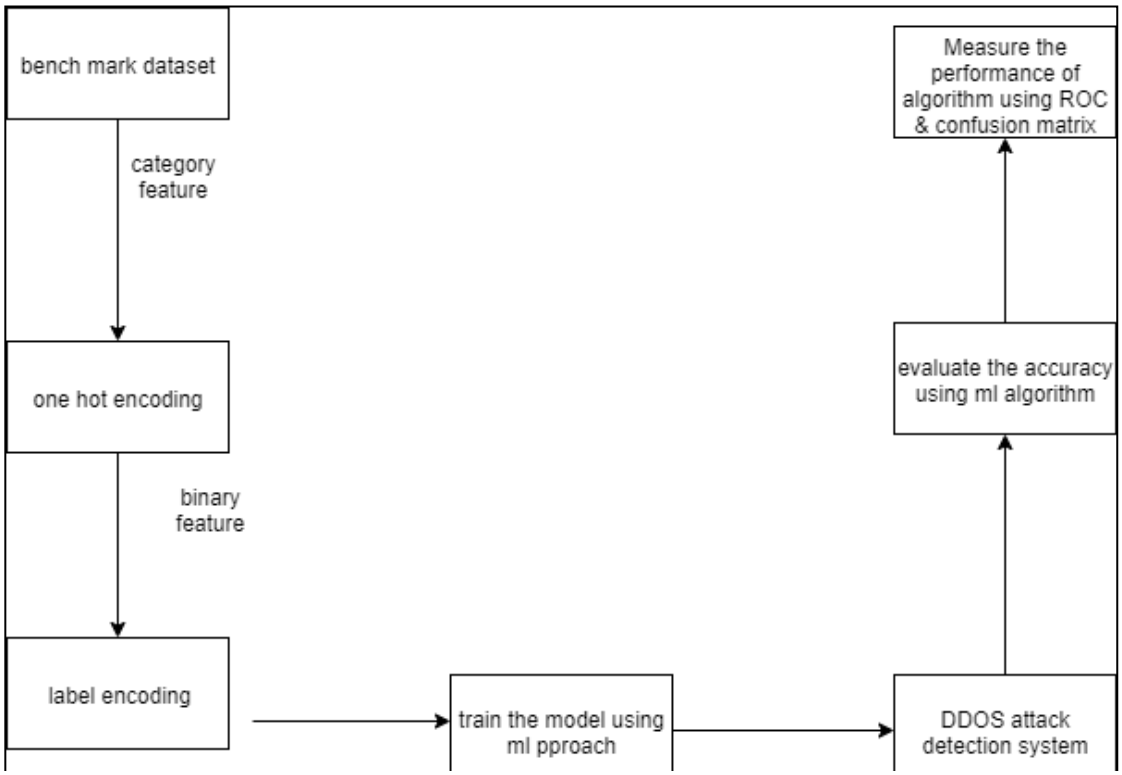

Figure 3.2 Level 1 Data Flow Diagram

5 Implementation

Anaconda

Anaconda is a freemium open source distribution of the Python and R programming languages for large-scale data processing, predictive analytics, and scientific computing, that aims to simplify package management and deployment. Its package management system is conda. Anaconda distribution comes with more than 1,000 data packages as well as the Conda package and virtual environment manager, called Anaconda Navigatorso it eliminates the need to learn to install each library independently.

Jupyter NoteBook

Jupyter Notebook (formerly IPython Notebooks) is a web-based interactive computational environment for creating Jupyter notebooks documents. The "notebook" term can colloquially make reference to many different entities, mainly the Jupyter web application, Jupyter Python web server, or Jupyter document format depending on context.

Python

Python is an interpreted, high-level programming language for general-purpose programming. Created by Guido van Rossum and first released in 1991, Python has a design philosophy that emphasizes code readability, notably using significant whitespace. It provides constructs that enable clear programming on both small and large scales. In July 2018, Van Rossum stepped down as the leader in the language community.

Python features a dynamic type system and automatic memory management. It supports multiple programming paradigms. It includes object oriented, imperative, functional and procedural, and has a large and comprehensive standard library.

Algorithms Used-

Principal Component Analysis (PCA)

Principal component analysis (PCA) is a technique to bring out strong patterns in a dataset by supressing variations. It is used to clean data sets to make it easy to explore and analyse. The algorithm of Principal Component Analysis is based on a few mathematical ideas namely:

- Variance and Convariance
- Eigen Vectors and Eigen values

Formula of Convariance

$$c(x, y) = \sum_{i=1}^{n} \left(\frac{(x-\bar{x})(y-\bar{y})}{n-1} \right)$$

$$\mathrm{C}\begin{bmatrix} cov(x,x) & cov(x,y) \\ cov\,(y,x) & cov(y,y) \end{bmatrix}$$

Formula of Eigen Vectors

$$|c \quad -\lambda \mathrm{I}| = 0$$

Naive Bayes Algorithm

It is a classification technique based on Bayes' theorem with an assumption of independence between predictors. In simple terms, a Naive Bayes classifier assumes that the presence of a particular feature in a class is unrelated to the presence of any other feature. Naive Bayesian model is easy to build and particularly useful for very large data sets. Along with simplicity, Naive Bayes is known to outperform even highly sophisticated classification methods.

The Bayes theorem is used for the calculation of the posterior probability,$p(c|x)$ from $p(c), p(x)$, and $(x|c)$. This classifier assumes the effect of the predictor $value$ (x) on a given and independent of the class (c) and it is independent of the values of another predictor.

$$p(c|x) = \frac{p(x|c)p(c)}{p(x)}$$

$p(c|x)$is the probability of posterior class.
$p(c)$is the probability of the prior class.
$p(x|c)$is the given probability of predictor class.
$p(x|c)$is the probability of the prior predictor.

K-Nearest Neighbor

K-Nearest Neighbors (KNN) is used for classification and regression problems in machine learning.KNN is implemented by using a distance function. If the value of no. of classifiers K=1, then the case will be assigned for the class of its nearest neighbor.

K-Nearest Neighbors (**KNN**) is used for classification and regression problems in machine learning.KNN is implemented by using a distance function. If the value of no. of classifiers K=1, then the case will be assigned for the class of its nearest neighbor. The distance function is given by-

Euclidean $\sqrt{\sum_{i=1}^{k}(x_i - y_i)^2}$ (1)

Manhattan $\sum_{i=1}^{k}|x_i - y_i|$ (2)

Minkowski $\left(\sum_{i=0}^{k}(|x_i - y_i|)^q\right)^{\frac{1}{q}}$ (3)

In the Euclidean equation,

X refers to the distance between a point $x(x1, x2, \ldots., xn)$

Y refers to the distance between a point $y(y1, y2, \ldots., yn)$

In the Manhattan equation,

Distance between two points $(x1, y1)$ and $(x2, y2)$is: $|x_i - y_i|$ $x = (x1, x2, \ldots., xn)$and $y = (y1, y2, \ldots., yn)$

In the Minkowski equation,

The distance of order p (where p is an integer) between two points:

$x = (x1, x2, \ldots., xn)$ and $y = (y1, y2, \ldots., yn)$

Defined as: $\left(\sum_{i=0}^{k}(|x_i - y_i|)^q\right)^{\frac{1}{q}}$

Data Set Contents

	Feature Name	Description
1.	Count	No. of connections to the same host as the current connection in the last two seconds
2.	destination bytes	Bytes sent from destination to source
3.	diff srv rate	percentage of connections to different services
4.	dst host count	count of connections having the same destination hosts
5.	dst host diff srv rate	percentage of different services on the current host
6.	dst host rerror rate	percentage of connections to the current host that has an RST error
7.	dst host same src port rate	percentage of connections to the current host having the same src port
8.	dst host same srv rate	percentage of connections having the same destination host and using the same service
9.	dst host serror rate	percentage of connections to the current host that have an S0 error
10.	dst host srv count	count of connections having the same destination host and using the same service
11.	dst host srv diff host rate	percentage of connections to the same service coming from different hosts
12.	dst host srv rerror rate	percentage of connections to the current host and specified service that have an RST error
13.	dst host srv serror rate	percentage of connections to the current host and specified service that have an S0 error
14.	Duration	Duration of the active connection.
15.	Flag	Status flag of the connection
16	Hot	No. of "hot" indicators
17.	is guest login	One if the login is a "guest." login; Otherwise 0
18.	is host login	One if the login belongs to the "host"; otherwise 0

19.	Land	One if the connection is from/to the same host/port; Otherwise 0
20.	logged in	One if successfully logged in; otherwise 0
21.	num access files	No. of operations on access control files
22.	num compromised	No. of compromised Conditions
23	num failed logins	No. of failed logins
24.	num file creations	No. of file creation Operations
25.	num outbound cmds	No. of outbound commands in an ftp session
26.	num root	No. of "root" accesses
27.	num shells	No. of shell prompts
28.	protocol type	Connection protocol (e.g. tcp, udp).
29.	rerror rate	percentage of connections that have “REJ" Errors
30.	root shell	One if the root shell is obtained; otherwise 0
31.	same srv rate	percentage of connections to the same service
32.	serror rate	percentage of connections that have “SYN" Errors
33.	Service	Destination service (e.g. telnet, ftp)
34.	src bytes	Bytes sent from source to destination
35.	srv count	No. of connections to the same service as the current connection in the last two seconds
36.	srv diff host rate	percentage of connections to different hosts

37.	srv rerror rate	percentage of connections that have "REJ" errors
38.	srv serror rate	percentage of connections that have "SYN" Errors
39.	su attempted	One if "su root" command attempted; otherwise 0
40.	Urgent	No. of urgent packets
41.	Wrong fragment	No. of wrong fragments

5.1 Result Analysis

Correlation matrix

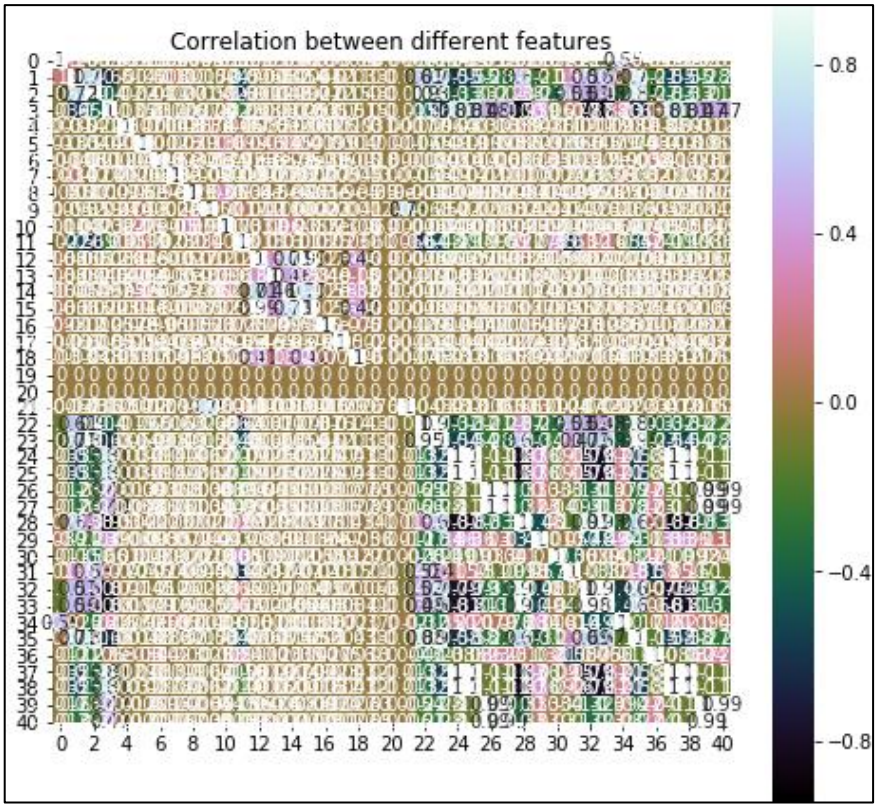

Figure 4 Correlation matrix between different features

Snap Shots of Accuracy of the algorithms

Naïve Bayes Algorithm

```
from sklearn import tree
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import accuracy_score
from sklearn.metrics import classification_report
from sklearn.metrics import mean_squared_error
clf = GaussianNB()
features_train=features_train.astype(int)
labels_train=labels_train.astype(int)
labels_test=labels_test.astype(int)
#training the model using training set
clf.fit(features_train , labels_train)
#predicting the label using test set on trained Model
prediction = clf.predict(features_test)
#calculating accuracy
acc=accuracy_score(prediction, labels_test)
accu.append(acc*100)
print("-----------------------------------------")
print("Accuracy  : ",acc*100," %")
```

```
-----------------------------------------
Accuracy  :  99.72684583670087  %
```

Figure 5 Naive Bayes Algorithm Accuracy

K-Nearest Neighbor Algorithm

```
acc=accuracy_score(y_pred,labels_test)
accu.append(acc*100)
print("Accuracy  : ",acc*100," %")
```

```
Accuracy  :  99.93554789405302  %
```

Figure 6 K-Nearest Neighbor Algorithm Accuracy

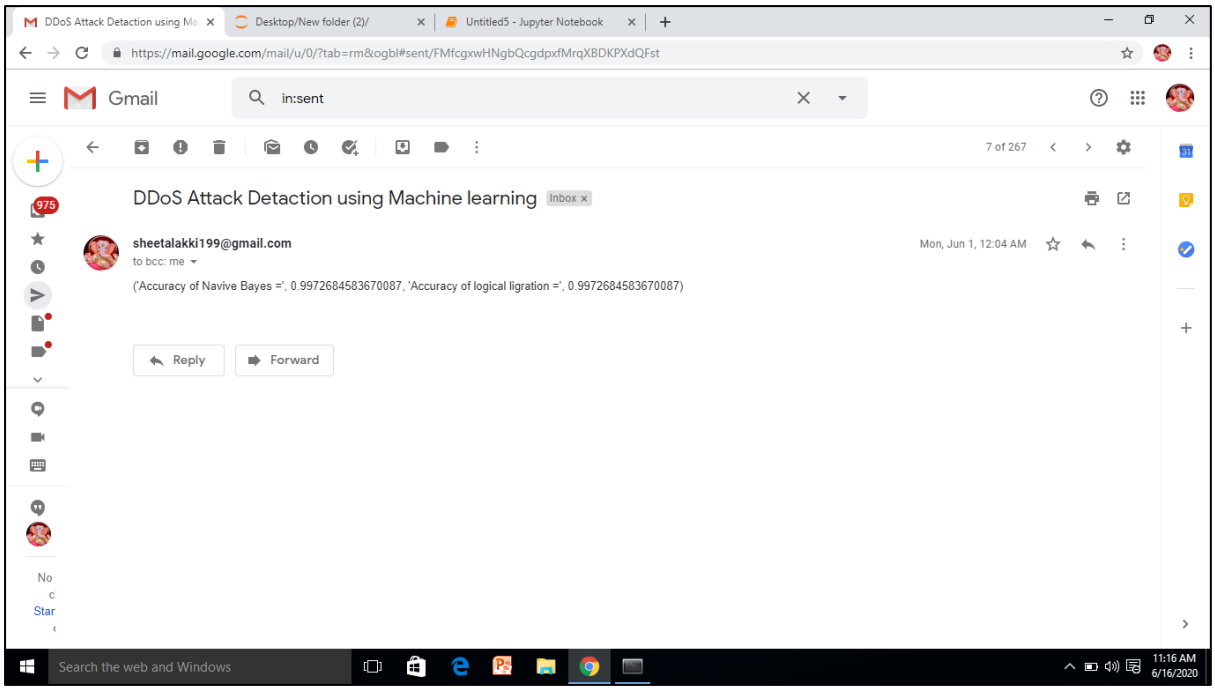

Figure 7 email to admin about accuracies of both algorithms

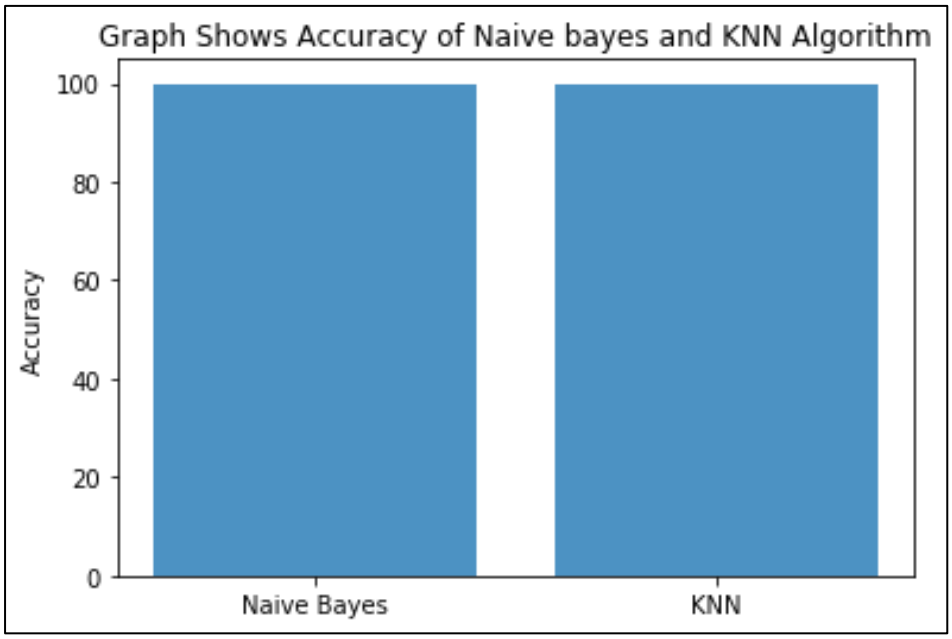

Figure 8 Bar Graph to compare accuracies

6. Conclusion and Future work

In this project, the problem was addressed by Network intrusion Detection for finding the best algorithm that detects the attack more efficiently, using the machine learning model on KDD Cup benchmark dataset. We proposed a Network Intrusion detection system framework comprising three tasks, namely, data pre-processing, machine learning model, and performance evaluation. We used two different architectures to classify the behaviours and found that the KNN algorithm outperforms compared to Naïve Bayes. As future work, the proposed framework can be extended by using variants of deep learning architectures with more robust features to detect intrusion with real time network data.

7. References

1. Panda, Mrutyunjaya & Patra, Manas. (2007). Network intrusion detection using naive bayes. 7.
2. Denial of Service Intrusion Detection System (IDS) Based on Naïve Bayes Classifier using NSL KDD and KDD Cup 99 Datasets, Asst. Prof. Dr. Soukaena H. Hashem, Hafsa Adil (2017)
3. Safaa O. Al-mamory, Firas S. Jassim, Evaluation of Different Data Mining Algorithms with KDD CUP 99 Data Set, Journal of Babylon University/Pure and Applied Sciences/ No.(8)/ Vol.(21): 2013
4. A Detailed Analysis of the KDD CUP 99 Data Set, Mahbod Tavallaee, Ebrahim Bagheri, Wei Lu, and Ali A. Ghorbani, Proceedings of the 2009 IEEE Symposium on Computational Intelligence in Security and Defense Applications (CISDA 2009).
5. Bing Zhang , 1,2 Zhiyang Liu , 1,2 Yanguo Jia , 1,2 Jiadong Ren, and Xiaolin Zhao3, Network Intrusion Detection Method Based on PCA and Bayes Algorithm(2018)
6. L. Akyildiz, W. Su, Y. Sankarasubramanian, E. Cayirci, "An improved network intrusion detection technique based on k-means clustering via Naïve bayes classification ", IEEE Communications Magazine, vol.40, no.8, pp.102-114, August 2002.
7. E. Nikolva and V. Jecheva, "Anomaly based intrusion detection based on KNN algorithm", Journal of Information assurance and security, vol. 2, pp. 184-188, 2007
8. V. Venkatechalam and S. Selvan, "Performance comparison of intrusion detection system classification using various feature reduction techniques", International journal of simulation, vol. 9, no. 1, pp. 30-39, 2008.
9. **.** M. Panda and M.R. Patra, "Network intrusion detection using Naïve Bayes", International journal of computer science and network security, vol. 7, no. 12, pp. 258-263, 2007.