Network Intrusion Detection Using Data Analytics (Traditional ML and Spark ML Approach)

***Abstract*—** This paper focuses on the analysis of the network intrusion dataset using data analytics with the aim to develop the most effective way of network intrusion detection. There are several datasets available on the internet, but according to my research, one of the best datasets is available on the website of Canadian Institute for Cybersecurity (CIC) which had the improvised and better version of KDD'99 dataset. The objective is to analysis the CIC dataset and develop machine learning algorithm using Naïve Bayes for improving the accuracy of the network intrusion detection. On the other hand, this paper will also take the advantage of using Spark together with Naïve Bayes algorithm. The results based on Naïve Bayes alone are compared with the combined approach of using Spark with ML models like Logistic Regression, Decision Tree, Random Forest and Naïve Bayes Multinomial.

Keywords—Network Intrusion, Machine Learning (ML), KDD'99 Dataset, Spark, Naïve Bayes.

I. Introduction

In the recent time with the advancement of internet services and communication technologies, network intrusion has become one of the serious issues that the world is facing. Although there are many tools that have been employed like firewall, antivirus, and network intrusion detection systems, it is essential to develop more innovative solution to protect systems from different type of intrusions such as unauthorized access, Distributed Denial of Service (DDoS) attacks, Malwares etc.

The first ever idea of network intrusion detection system was proposed by Jim Anderson in the year 1980 (J.P. Anderson, 1980). Since then, many advancements have been made in this area. With the new security threats and network vulnerability, researchers are trying to find a way for an effective solution because the existing systems have been inefficient in detecting new types of attacks and reducing the false alarm rates (Hoque et al., 2012).

To overcome these shortcomings, various machine learning (ML) and deep learning (DL) methods were proposed (Prasad & Rohokale, 1970). These methods come under the big umbrella of artificial intelligence which aims at learning from the useful information that can be extracted from the network traffic data.

The machine learning methods are mainly dependent on the useful information of the network traffic data. There were many ML algorithms that were proposed by various researchers for intrusion detection system such as Random Forest, K-Nearest Neighbors (KNN) algorithm, Bayesian probability and so on. Researchers also tried integrating the Spark with ML methods for better performance. Therefore, this paper focuses on providing the performance comparison between the methods of Spark with ML algorithms and the traditional approach of applying ML algorithm alone.

As not much of work is done detecting intrusion using Naïve Bayes, this paper tends to apply the algorithm and see how accurate it is and then compare it with the results of algorithm application in Spark. The aim would be to compare how accurate the algorithms work in both the environment and the performance it shows in terms of processing time. The paper also provides the background study and provides scope for the future work. The main focus of the work is to provide researchers more information about the algorithm and the accuracy with their processing time

II. Literature review

As a context has been taken as a reference from the authors (Cemerlic, Yang, 2008) proposed network intrusion detection using Bayesian Network. The experimental setup in this model were done in two phases on the DARPA Dataset. The first phase includes testing and training of Bayesian Model on the DARPA Dataset and the other phase includes capturing of real-world data and testing the model. Predictions were made and error rate were quite low in DARPA Dataset but quite high in the real-world situation.

Authors of paper (Altwaijry & Algarny, 1970) proposed the detection method using Bayesian Probability. The model was implemented using only 10% of the dataset and the attacks were classified into four categories – DOS (Denial of Service Attacks), Probe (sometimes known as Probing), U2R (User to Root Attacks) and R2L (Remote to Local Attacks). The analysis was done with different subsets of data too for giving better insight of the dataset. Finally, it shows some experimental results demonstrating the improvement of Detection Rate for Remote to user (R2L) attacks.

(Mehdi, 2007) proposes Bayesian classification procedure that was linked to unsupervised learning algorithm. The evaluation of the model was done using the parameter of minimum entropy. This model was so effective that it could also be used in real-world situation as claimed by the authors. The dataset used for this paper was created artificially and testing and training was done based on the generated dataset.

Kato and Klyuev in the year 2017 (Kato & Klyuev, 1970) proposed the detection method using Apache Hadoop and

Spark. The system given in this was implemented using Hive SQL and unsupervised learning algorithms. In this GMM Model (i.e., Gaussian Mixture Model), K-Means algorithm and OCSV Model (i.e., One Support Vector Model) were used. This paper shows that Hadoop and Spark can handle large dataset efficiently and accurately. The accuracy of the model was 86.2% and duration for processing was around 6 to 8 minutes.

As a context has been taken as a reference from the authors (Zhang, 2018) has done detection using Random Forest Method on Spark platform. The system was done using three frameworks – NetFlow, pre-processing of data and classification-based detection. This system used showed more effectiveness as it was done on Spark. This system performs well even in high speed and is suitable for real time detection.

Authors of paper (Altwaijry, 2016) also proposed Bayesian Probability. But the main purpose of this paper was to improve the accuracy of detection of R2L (i.e., Remote to Local Attack) which accuracy of other attacks did not show improvement. In this paper, various algorithms were compared for the detection method and high accuracy of 85.35% for detecting R2L attack was obtained. The future work of this paper says various Bayesian filters could be used in parallel to improve accuracy.

(Al-Sharo et al., 1970) focuses on the main challenges related to machine learning technologies and the difficulties related to Big Data like Hadoop, Hive and Cloud. The challenge especially in the field of Health and Medical Sector was explained in detail. The challenges related to visualization of big data was also brought into light in this paper.

After reading all these papers we can see most of them used Bayesian Probability, Random Forest and KNN. Most of them tried to improve detection rate by detecting either a specific attack or on a single type of attack. Therefore, I decided to work on all types of attacks. The papers written on Detection using Spark ML or Hadoop were either just theoretical or they did not have any comparison with the traditional approach which led me to do a comparative analysis in this paper. Although all the models applied had quite good detection rate but very few achieved above 90% accuracy which we are going to try to achieve.

III. METHODOLOGY

A. Data Description

The source of the dataset for the project is: https://www.unb.ca/cic/datasets/nsl.html. It is a website of Canadian Institute for cybersecurity. The dataset provides an improvised version of KDD'99 dataset mentioned in paper (Tavallaee et al., 1970) and the dataset has better statistical observations for the models to apply.

The data for analysis is in .txt format that has to be converted into CSV format using pandas library. This is because the CSV format is much easier to perform analysis of the data and perform operations on it. We will split 80% of records in training and rest in testing data. The main aim is to identify attack types and the network protocols from the dataset to check which categorical value is best to predict an attack and which is the best to make predictions. We will use this dataset for both traditional as well as Spark ML approach. Some of the main features of the dataset are explained in Table 1.

Column Name	Purpose
Duration	Number of seconds of a connection.
Protocol	Type of protocol for the instance, e.g., TCP, UDP etc.
Flag	Gives the status of the connection.
Src_bytes	Number of bytes from source to destination
Dst_bytes	Number of bytes from destination to source
Service	Type of network service on the destination for instance, ttp or telnet
Attack	Tells whether detection is normal or any type of real attack

Table 1: Some Main Features in the dataset

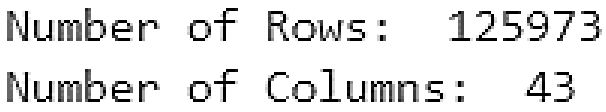


Figure 1: Count of Dataset

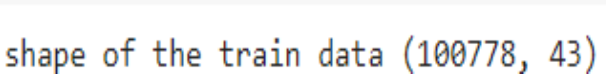


Figure 2: Count of Training Data after Split

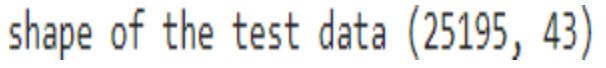


Figure 3: Count of Testing Data after Split

B. Pre-requisites and Environment Setup

- Google Colab used as it is the easiest to setup and it has the capacity to handle large dataset with low processing time.
- Importing necessary python libraries for performing traditional ML tasks like:
  - [1] numpy which is used to work operations on arrays,
  - [2] pandas which is important for data manipulation,
  - [3] sklearn also known as Scikit-learn which has tools to perform machine learning

operations like classification, regression etc.,

[4] matplot library for data visualization

[5] seaborn for drawing informative statistical graphs

[6] plotly.express for producing informative figures

- Create environment for Spark Hadoop and import pyspark libraries for performing Spark with ML algorithms.
- Import dataset from drive or local system. In our case we upload it from google drive for which have to mount our drive into google colab first.

C. Design

The data analysis of network intrusion data for our project is being carried out in following steps:

- Firstly, the data set is loaded.
- Data processing is performed which includes data cleaning such as removing NaN values, exploratory analysis to get insights of the categorical values, then perform scaling numerical attributes and then encoding categorical attributes.
- Then we perform Decision Tree Algorithm for predicting attacks using categorical values.
- Then we apply Naïve Bayes to find out AUC, f1 and recall score.
- After applying Naïve Bayes using traditional approach, we return to Spark ML and apply Naive Bayes Multinomial here with several algorithms such as Logistic Regression, Decision Tree and Random Forest and calculate the processing time and compare the scores with Traditional Approach.

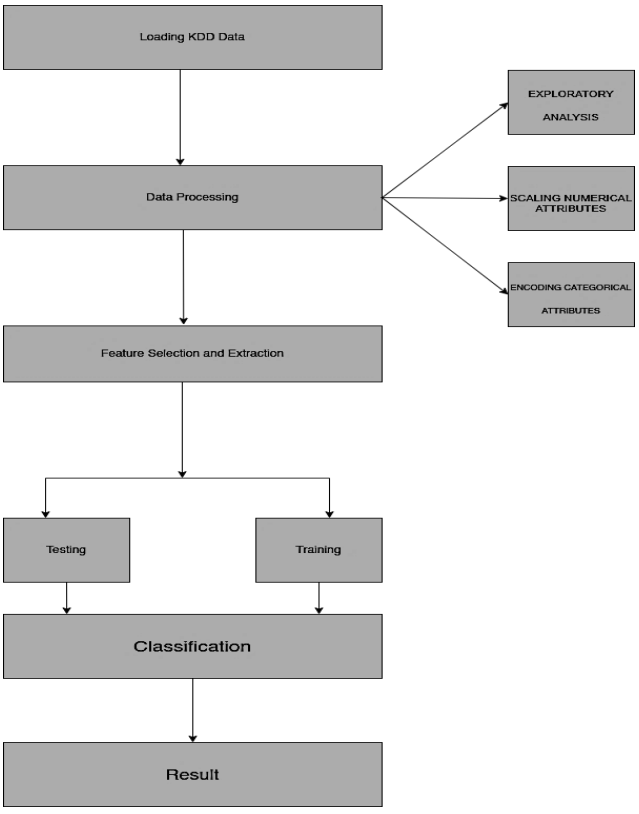


Figure 4: Flowchart for the implementation of Data Analysis

D. Analysis of Dataset Using for Traditional Approach of ML

To perform exploratory analysis of training and testing data we will be using matplot library and plotly.express for the representing the data in the form of graphs.

Representing Categorical Values

Firstly, we will assign class labels to all the attacks as 1 and to normal (not attacks) as 0 then represent it in the form of graphs. From fig. 5 and fig. 6 below we can observe that class 0 has more datapoints as compared to class 1 in both training as well as testing data.

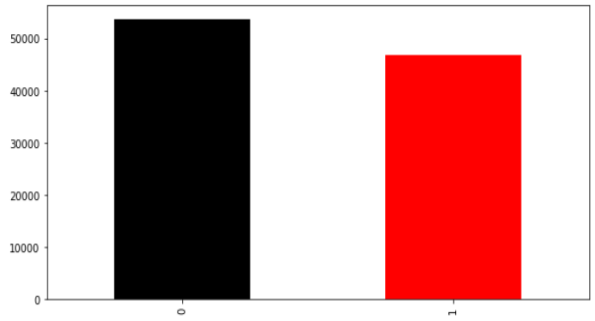


Figure 5: Class Distribution of Attacks in Training Data

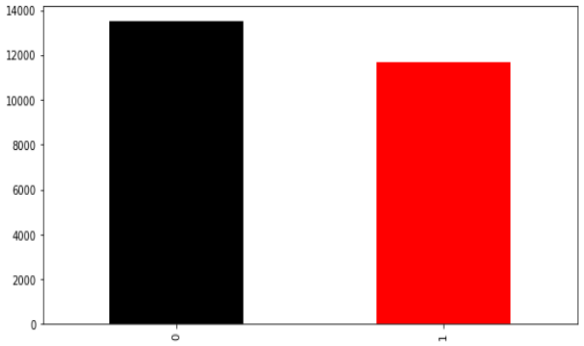


Figure 6: Class Distribution of Attacks in Testing Data

Data cleaning operations is applied to check whether the data has duplicate values or any null values. Then we will apply models to check which categorical value is best for predicting attacks. But before that just for visualization purpose we want to see which protocol type has most attacks in training data. The graph in fig. 7 shows the distribution of that.

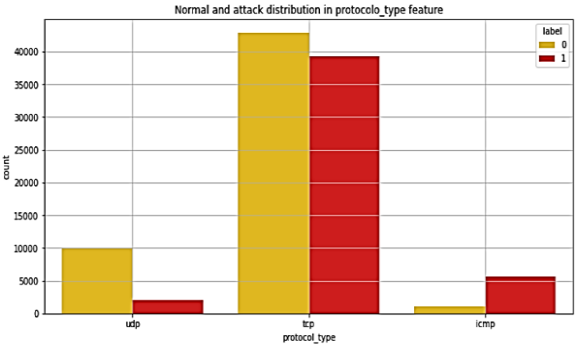


Figure 7: Protocol type distribution with respect to attacks label

For this visualization we know that tcp is the most used protocol type. The icmp type has more attacks than normal class label. Whereas udp has least percent of attacks compared to others.

Then for predicting how accurate the categorical values are in predicting the attack, we implement decision tree classifier.

Decision Tree Classifier Model

Decision Tree model is a supervised learning method for classification. We are using this model because it is simple to understand and visualize and requires very little data preparation. For performing this we will first perform one hot encoding on all the the categorical values.

```
prototype_vectorizer = CountVectorizer()
train_protocol_type_encoding = prototype_vectorizer.fit_transform(train['protocol_type'])
test_protocol_type_encoding = prototype_vectorizer.transform(test['protocol_type'])
```

```
print("Train_protocol_type_encoding - The shape of gene feature:",train_protocol_type_encoding.shape)
```

```
Train_protocol_type_encoding - The shape of gene feature: (100778, 3)
```

Figure 8: Code and Output of Encoding Protocol type

Then we fit the decision tree algorithm to the training set which is the categorical values and the attack labels. After that we predict the results using auc score. If the auc score is between 0.5-1 then it is considered to be good for prediction. We are going to perform this operation separately for each categorical value in the dataset to predict attacks.

Representing Continuous Variables

We represent continuous features using boxplots of plotly.express library. This is done just for exploratory analysis of data. Fig. 9, 10 and 11 shows how continuous variables are represented. Similarly, all continuous variables are represented like this.

Fig. 9 shows the range of duration of connection in milli seconds. The range of connection with attack has more duration than that of normal.

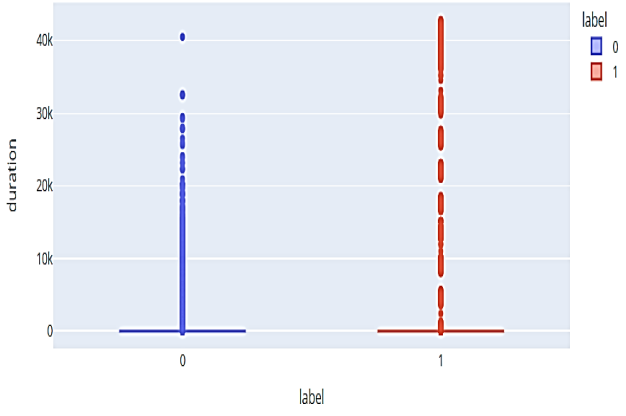


Figure 9: Boxplot of duration variable with respect to attacks label

Similarly, fig. 10 shows the number of bytes sent from source to destination. And as we can see the connection with attacks has greater range than of normal label.

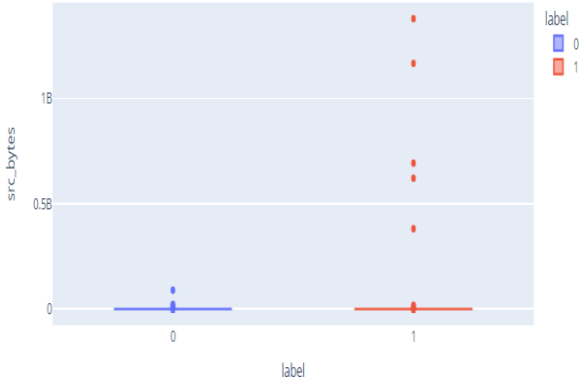


Figure 10: Boxplot of src_bytes variable with respect to attacks label

The plot of number of bytes sent from destination to source in fig. 11 is almost similar to previous one. The range of normal is not much but of attack label is quite high going above 1.2B bytes.

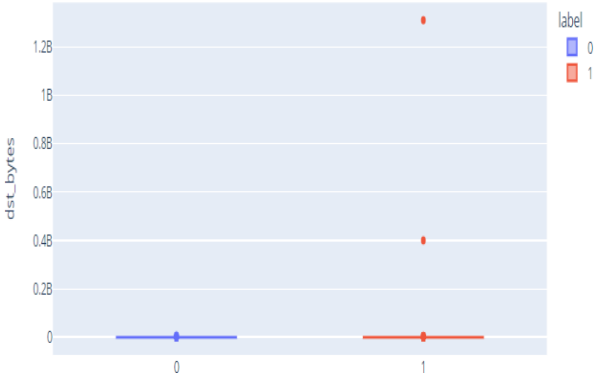


Figure 11: Boxplot of dst_bytes variable with respect to attacks label

Naïve Bayes Model

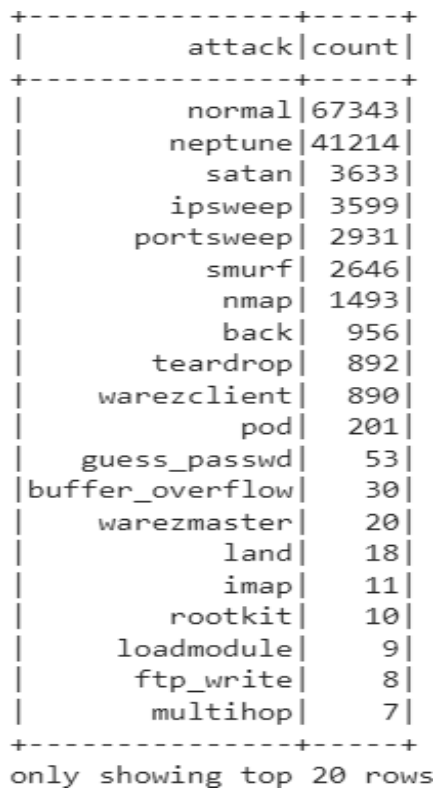
To apply Naïve Bayes Model, we will firstly merge the categorical and numerical features. Then we perform the standardization operation and then we will create a function to plot confusion matrix and calculate f1 score and recall score/detection rate. We will also calculate the processing time for the model using time function. This is to check which approach gives better results and less processing time.

We will again apply the same model but this time after dropping the label column and correlated features such as ‘srv serror rate’, ‘dst host rerror rate’, ‘dst host same srv rate', 'dst host srv rerror rate', 'is guest login', 'dst host srv serror rate', 'num root', 'dst host serror rate', 'srv rerror rate'. This is to just check how good a model will work without the input of these features.

E. Analysis of Dataset Using Spark ML

After setting up the environment and loading dataset for applying model. We performed some analysis on the whole dataset before splitting.

Fig. 12 shows the count of top 20 attacks. We can observe that normal as most count and Neptune attack has the greatest number of counts among all the attacks.

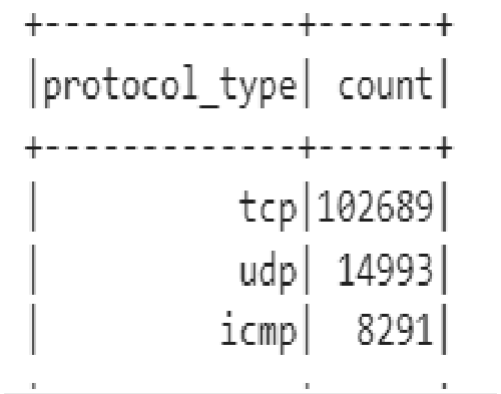


attack	count
normal	67343
neptune	41214
satan	3633
ipsweep	3599
portsweep	2931
smurf	2646
nmap	1493
back	956
teardrop	892
warezclient	890
pod	201
guess_passwd	53
buffer_overflow	30
warezmaster	20
land	18
imap	11
rootkit	10
loadmodule	9
ftp_write	8
multihop	7

only showing top 20 rows

Figure 12: Attacks Count in the Dataset

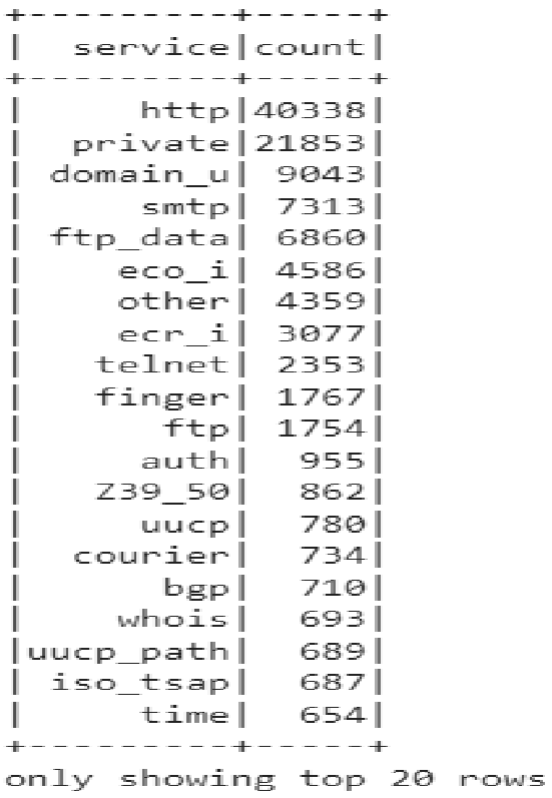
Fig. 13 shows tcp protocol type has the greatest number of count and icmp has the least. This shows tcp protocol is quite in trend in today's date.



protocol_type	count
tcp	102689
udp	14993
icmp	8291

Figure 13: Protocol type Count in the Dataset

Fig. 14 shows http service has the greatest number of count and private the second highest. This implies that http is the most used service in today's time which is evident when we use to browse a website.

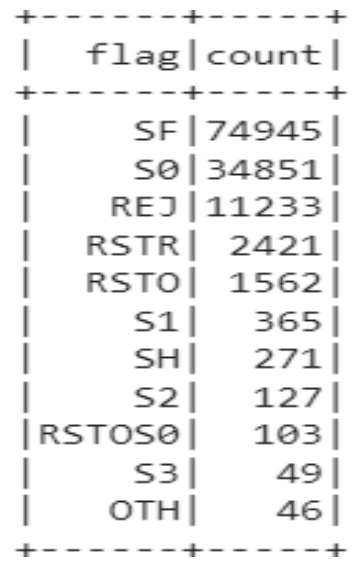


service	count
http	40338
private	21853
domain_u	9043
smtp	7313
ftp_data	6860
eco_i	4586
other	4359
ecr_i	3077
telnet	2353
finger	1767
ftp	1754
auth	955
Z39_50	862
uucp	780
courier	734
bgp	710
whois	693
uucp_path	689
iso_tsap	687
time	654

only showing top 20 rows

Figure 14: Services Count in the Dataset

Fig. 15 shows the flag count in which SF feature has most count and then S0. With the size of dataset from the count itself we can say that only the top 5 flags will be relevant in prediction of attack as the count of others are very less.



flag	count
SF	74945
S0	34851
REJ	11233
RSTR	2421
RSTO	1562
S1	365
SH	271
S2	127
RSTOS0	103
S3	49
OTH	46

Figure 15: Flag Count in the Dataset

We will apply machine learning algorithms on the split dataset and also calculate the time taken for running spark job. This can be done by importing time library and use time function. We will then note the results for Spark ML and processing time taken and compare it with Traditional approach of ML.

IV. RESULTS

A. *Analysis of Dataset Using for Traditional Approach of ML*

We had applied two models on the dataset. Decision Tree as well as Naïve Bayes. Decision Tree Model was applied on all the categorical values separately to predict the attack whereas Naïve Bayes was applied with all the features together at first then with selected features. The results obtained from them are explained below:

Decision Tree Classifier Model

From protocol type feature I am getting an auc score of 0.61. Therefore, we can say it is helpful in predicting the Attacks.

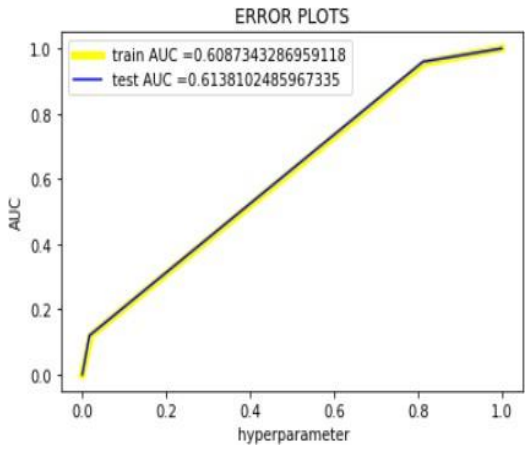


Figure 16: AUC Score for Protocol type in predicting Attack

By using only the service feature the model gives and auc score of 0.93 which is quite interesting. It shows service feature alone is good in predicting attacks.

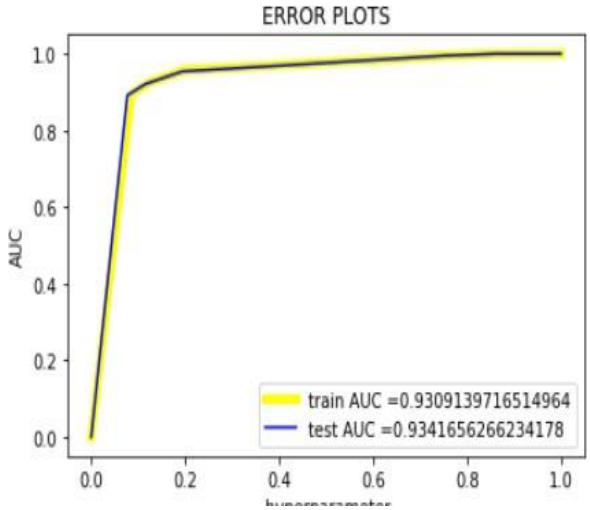


Figure 17: AUC Score for Service in predicting Attack

The flag feature gives auc score of 0.89 which is also quite good but a little less than the previous result we obtained from service.

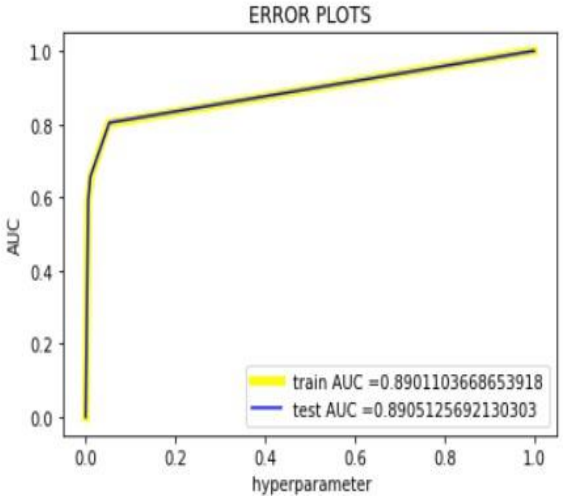


Figure 18: AUC Score for Flag in predicting Attack

Naïve Bayes Model

The test AUC score shown in fig. 19 is 0.982 which means chance of classfying points correctly is 98%. The test AUC and train AUC is almost similar therefore our model is quite balanced.

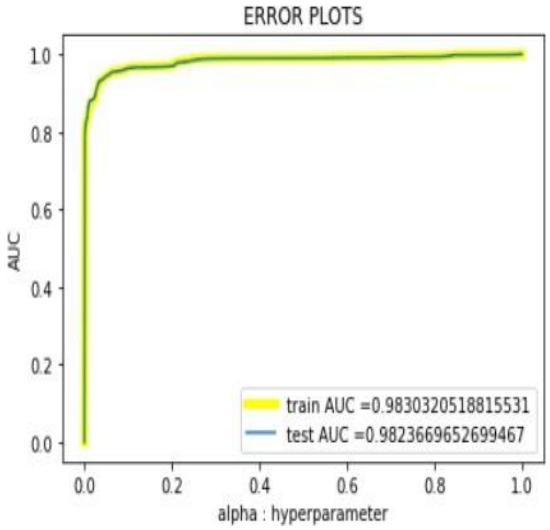


Figure 19: AUC Score when all features included

Our f1 score and detection rate is also quite high as shown in fig.20 which means our model is quite accurate.

```
Train f1 score 0.9250991400436662
Test f1 score 0.9274648517954687
train recall score / detection rate 0.8843360664465978
test recall score / detection rate 0.8871092077087794
```

Figure 20: F1 score and Detection rate Using Naïve Bayes

Although our model showed good results we checked ran the model but this time after removing certain features and the result improved very little by 0.001 which is almost 0 therefore it means it did not have much effect on the model as it was already quite high.

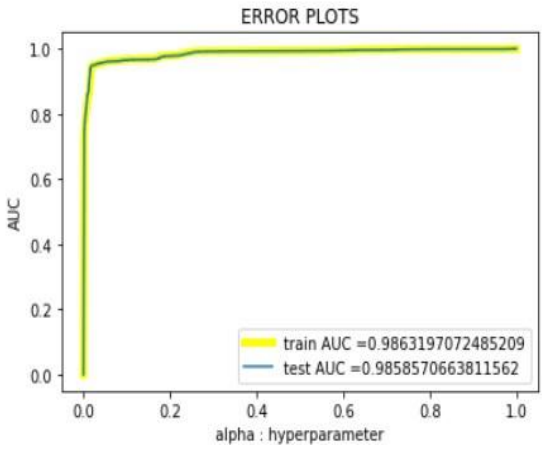


Figure 21: AUC Score with Selected Features

But when we calculated the f1 score and detection rate after removing some features f1 score improved by 0.04 and detection rate by 0.06 which is quite a good result. Therefore, we can say the removing selected features proved to be beneficial in improving our detection rate which was our goal.

```
Train f1 score 0.9615792549622285
Test f1 score 0.9614864570157456
train recall score / detection rate 0.9460973272281972
test recall score / detection rate 0.94406852248394
```

Figure 22: F1 score and Detection rate Using Naïve Bayes for Selected Features

B. Analysis of Dataset Using Spark ML

On applying the Naïve Bayes Model on the same dataset but in Spark environment our accuracy was 41% as shown in fig. 23 and f1 score is also quite low i.e., just 0.43. This proves our model does not work well in the Spark environment. Therefore, we decided to work with different models too in Spark environment.

```
------------------------------
Naive Bayes Multinomial
accuracy = 0.41
weightedPrecision = 0.88
weightedRecall = 0.41
f1 = 0.43
------------------------------
```

Figure 23: Naïve Bayes Model Scores in Spark ML

As we did not know which model works best in the Spark environment, we applied three models to the dataset. They were Logistic Regression, Decision Tree and Random Forest.

As the results in fig. 24, 25, 26 shows all the models we chose are much better than Naïve Bayes. But to pick out the best model Random Forest proved to be the best model giving an accuracy as high as 97% with f1 score of 0.96 and detection rate of 0.97 which is better than Naïve Bayes Model which was done using traditional approach.

```
Logistic Regression
accuracy = 0.80
weightedPrecision = 0.72
weightedRecall = 0.80
f1 = 0.75
------------------------------
```

Figure 24: Logistic Regression Model Scores in Spark ML

```
------------------------------
Decision Tree
accuracy = 0.98
weightedPrecision = 0.98
weightedRecall = 0.98
f1 = 0.98
------------------------------
```

Figure 25: Decision Tree Model Scores in Spark ML

```
Models Evaluation:
------------------------------
Random Forest
accuracy = 0.97
weightedPrecision = 0.96
weightedRecall = 0.97
f1 = 0.96
------------------------------
```

Figure 25: Random Forest Model Scores in Spark ML

C. Comparitive Analysis

Now our aim is to make a comparison of Naïve Bayes model in different environment and compare the processing time. As we can see in Table 2 although our processing time in Spark is quite low but the f1 score and recall score is really poor. Therefore, this model is not suitable on our dataset in Spark Environment

Statistics	Traditional Approach	Spark ML Approach
F1 Score	0.92	0.43
Recall Score	0.88	0.41

Processing Time	10-20 seconds	3-6 seconds

Table 2: Comparative Analysis of Naïve Bayes using Traditional Approach vs Spark

As Naïve Bayes did not prove to be a good fit in Spark ML, we will compare the model that worked best in Spark i.e., Random Forest and compare it with Naïve Bayes in Traditional Approach.

Statistics	Naïve Bayes in Traditional Approach	Random Forest in Spark ML
F1 Score	0.92	0.96
Recall Score	0.88	0.97
Processing Time	10-20 seconds	10 -15 seconds

Table 3: Comparative Analysis of Naïve Bayes in Traditional vs Random Forest in Spark ML

Every Model in Spark environment had a different processing time. We ran all the models 3 times to see the average range of processing time for all the 4 models applied. They are mentioned in Table 4.

Model	Average Processing Time
Naïve Bayes	3-6 seconds
Random Forest	10-15 seconds
Decision Tree	8-12 seconds
Logistic Regression	14-20 seconds

Table 4: Comparative Analysis of Processing Time of Different ML Models in Spark

We can observe that Naïve Bayes has the fastest processing time and logistic regression takes the most processing time in Spark.

V. CONCLUSION

Both the models i.e., Decision Tree and Naïve Bayes applied in Traditional Approach of ML were quite balanced and gave good results. But in Spark ML as Naïve Bayes did not give very good result, we had to apply other models to improve the accuracy. But as expected the processing time of Spark ML was much less than Traditional Approach. This proves that Spark is obviously better in handling and processing large amount of data. The other models applied to improve accuracy in Spark were Logistic Regression, Random Forest and Decision Trees. Out of all these models Random Forest had the best detection rate but it's processing time was almost similar to Naïve Bayes in Traditional Approach. But from the results we can also gain that Decision Tree Model in Spark gave equally good accuracy and even had less processing time when compared to Traditional Approach. Therefore, Decision Tree can be an alternative too in Spark Environment.

If we talk about categorical values separately our model showed that service and flag feature are really good to predict an attack. We also observed after doing an exploratory analysis on the data the distribution of unique features in a categorical value is quite imbalanced therefore, we could take only top features from those categorical values and ignore the ones which had quite less count. It wouldn't make much difference to the model. Then if we talked about continuous features the range was always quite high for them for attacks datapoint than normal, the reason being that there is more transfer of data to and fro from source to destination when there is an attack.

Lastly, if we look at the overall results our aim to have a model with good detection rate was achieved and simultaneously running the models in Spark environment was done which had a few hiccups at first but gave the results we desired for with different models.

VI. FUTURE WORK

Although we got some promising results but we believe there is always a chance of improvement. As we saw Naïve Bayes worked well when implemented using Traditional Approach but did not give promising results when implemented in Spark environment. A model should be developed in such a way it gives good results on any type of dataset and in any environment. Therefore, for future work we can work on real time dataset too and see which model works the best and in which environment. Apart from Spark ML, Map-Reduce integrated with Machine Learning can also be used to see how efficient it is on predicting attacks. Also, besides the models mentioned and implemented in our project we can perform implement other machine learning algorithms too such as Dynamic Bayesian, SVM and neural networks can be applied to and see the results.

REFERENCES

Al-Sharo, Y.M. et al. (1970) Figure 5 from classification of Big Data: Machine learning problems and challenges in network intrusion prediction: Semantic scholar, undefined.

Altwaijry, H. (2016) bayesian based Intrusion Detection System, H. Altwaijry and S. Algarny, "bayesian based Intrusion Detection System," Journal of king saud university- computer and information sciences, vol. 24, no. 1, 2012, pp. 1-6. - references - scientific research publishing.

Altwaijry, H. and Algarny, S. (1970) [PDF] Bayesian based Intrusion Detection System: Semantic scholar, undefined.

Anderson, J.P. (1980) Computer Security Threat Monitoring and surveillance, J. P. Anderson, "Computer Security Threat Monitoring and surveillance," James P. Anderson Co., Fort Washington, 1980. - references - scientific research publishing.

Hoque, M.S., Mukit, M.A. and Bikas, M.A.N. (2012) An implementation of intrusion detection system using genetic algorithm, arXiv.org.

Kato, K. and Klyuev, V. (1970) Development of a network intrusion detection system using Apache Hadoop and spark: Semantic scholar, undefined.

Mehdi, M. (2007) A Bayesian networks in intrusion detection systems, www.researchgate.net.

Prasad, R. and Rohokale, V.M. (1970) Artificial Intelligence and machine learning in cyber security: Semantic scholar, undefined.

Tavallaee, M. et al. (1970) [PDF] a detailed analysis of the KDD cup 99 data set: Semantic scholar, undefined.

Yang, C.A. (2008) Network intrusion detection based on Bayesian Networks Alma Cemerlic, www.researchgate.net.

Zhang, H. (2018) Real-time distributed-random-FOREST-based network intrusion detection system using Apache Spark, IEEE Xplore.