PROJECT REPORT

Submitted for Data Mining and Business Intelligence (ITA5007)

By

Subrata Sahoo(18MCA1072)

COMPARITIVE STUDY OF CLASSIFICATION ALGORITHMS ON THE NSL-KDD DATASET

Guided by: Prof. Ramesh Ragala

(SCHOOL OF COMPUTING SCIENCE AND ENGINEERING)

April, 2019

ACKNOWLEDGEMENTS

In the completion of this project many people have put in sincere efforts. First of all we would like to express our gratitude to Dr. Ramesh Ragala. for giving us constant guidance thought the project and showing is the way. We are thankful to our university for providing us with the education and recourses needed for the project. Finally, we are extremely thankful to our parents for making us capable of doing this project and providing us with the monetary support required. Gratitude to our fellow mates for helping us and having fruitful discussions regarding the project.

Subrata Sahoo – 18MCA1072

Contents

I. ABSTRACT

As the network-based applications are growing rapidly, the network security mechanisms require more attention to improve speed and precision. The ever-evolving new intrusion types pose a serious threat to network security. Although numerous network security tools have been developed, yet the fast growth of intrusive activities is still a serious issue. Intrusion detection systems (IDSs) are used to detect intrusive activities on the network. Machine learning and classification algorithms help to design "Intrusion Detection Models" which can classify the network traffic into intrusive or normal traffic. In this paper we present the comparative performance of NSL-KDD based data set compatible classification algorithms. These classifiers have been evaluated using 41 attributes. Around 94,000 instances from complete KDD dataset have been included in the training data set and over 48,000 instances have been included in the testing data set. Random Forest, Decision Tree and Naïve Bayes are the three classification algorithms that are used in this comparative study.

Keywords--Machine Learning; Classification Techniques; NSL-KDD Dataset; Data Mining; Random Forest; Decision Tree ; Naïve Bayes.

II. INTRODUCTION

Intrusion detection is a type of security management system for computers and networks. An ID system gathers and analyses information from various areas within a computer or a network to identify possible security breaches, which include both intrusions i.e., the attacks from outside the organization and misuse attacks from within the organization. KDD process is used to denote the process of extracting useful knowledge from large dataset. Data Mining is the process of discovering knowledge from the large amount of dataset. Data Mining is the most vital step in the NSL-KDD process and it applies it to extract patterns from the data. The recent development in data mining has made available a wide variety of algorithms, drawn from the fields of statics, pattern recognition, machine learning and database. We have more chances of data loss; hacking and intrusion have been increased with the growth and popularity of the Internet. When continuously growing, Internet attacks suppose severe challenges to develop a flexible and adaptive security oriented methods. An intrusion can be defined as a series of actions that compromises the integrity, confidentiality or availability of a computer resource. Intrusion Detection System (IDS) is one of the most important components being used to detect the Internet attacks that can be either host based or network based. Intrusion detection is the process of monitoring and analyzing the activities occurring in a computer system or in a network in order to detect signs of security problems. IDS need only to detect threats and as such is placed out-of-band on the network infrastructure, meaning that it is not in the true real-time communication path between the sender and receiver of information. In some cases, the IDS may also respond to anomalous or malicious traffic by taking action such as blocking the user from accessing the network. This project paper is a study on comparing classification algorithms. However, most of these studies have been limited to only a very few classification algorithms. The theme of my thesis is to compare and better understand the prevalent classification algorithms, by evaluating the performance of three different classification algorithms on real network datasets.

III. DATASET

NSL-KDD is a dataset for network-based intrusion detection systems. It contains essential records of the complete KDD99 Cup data set. It is the new version of KDD Cup99 dataset. The testing dataset used for experimental purposes has 22 different attacks out of total 37 present in the dataset. The training dataset used for experimental purposes has 23 different attacks out of total 37 present in the dataset. NSL KDD data set is made up of 41 different attributes and there are five attack classes one of which is normal and other four are different types of attack. The features are given in the diagram below:

Duration	Is_guest login
Protocol_type	Count
Service	Srv_count
Flag	Serror_rate
Src_bytes	Srv_serror_rate
Dst_bytes	Same_srv_rate
Land	Diff_srv_rate
Wrong_fragment	Srv_diff_host_rate
Urgent	Dst_host_count
Hot	Dst_host_srv_count
Num_failed_logins	Dst_ host_Same_srv_rate
Logged_in	Dst_host_diff_srv_rate
Num_compromised	Dst_host_same_src_port_rate
Root_shell	Dst_host_srv_diff_host_rate
Su_attempted	Dst_host_serror_rate
Num_root	Dst_host_srv_serror_rate
Num_file_creations	Dst_host_rerror_rate
Num_shells	Dst_host_srv_rerror_rate
Num_access_files	Class
Num_outbound_cmds	diffic
Is_host_login	

IV. ALGORITHMS USED

A brief overview of three popular algorithms that have been applied to the study of intrusion detection is given below.

1. **Naïve Bayes** :

A Bayesian network is a model that encodes probabilistic relationships among variables of interest. Naive Bayesian classifiers use the Bayes theorem to classify the new instances of data. Each instance is a set of attribute values described by a vector, $X = (x1, x2 \ldots, xn)$.

Considering m classes, the sample X is assigned to the class Ci if and only if $P(X|Ci)P(Ci) > P(X|Cj)P(Cj)$ for all j in (1, m) such that j =6 I.

Naïve Bayesian technique is generally used for intrusion detection in combination with statistical schemes, a procedure that yields several advantages, including the capability of encoding interdependencies between variables and of predicting events, as well as the ability to incorporate both prior knowledge and data. Naïve Bayesian classifiers simplify the computations and exhibit high accuracy and speed when applied to large databases. A disadvantage of using Bayesian networks is that their results are similar to those derived from threshold-based systems, while considerably higher computational effort is required. Lack of available probability data is a significant disadvantage of the naïve Bayesian approach to IDS. Another disadvantage is that in Naïve Bayes approach it is assumed that the data attributes are conditionally independent which is not always so (it should be noted however that despite this, Bayesian classifiers give satisfactory results because focus is on identifying the classes for the instances, not the exact probabilities)

2. **Decision Trees** :

   Decision tree is a predictive modeling technique most often used for classification in data mining. The Classification algorithm is inductively learned to construct a model from the pre-classified data set. Each data item is defined by values of the attributes and classification may be viewed as mapping from a set of attributes to a particular class. Each non-terminal node in the decision tree represents a test or decision on the considered data item. Choice of a certain branch depends upon the outcome of the test. To classify a particular data item, we start at the root node and follow the assertions down until we reach a terminal node (or leaf). A decision is made when a terminal node is approached. An advantage of using decision tree algorithms for IDS is that its construction does not require any domain knowledge. Hence a data mining expert with little knowledge of networking can help build accurate decision tree models. Another significant advantage is that decision trees can handle high dimensional data. This increases the suitability of decision tree algorithms for IDS especially while considering the heterogeneity of network connection data and its ever-increasing size. Decision trees are able to process both numerical and categorical data (this suits the alphanumeric nature of network connection data). Finally, decision tree representations are easy to understand hereby making it easier for the network analyst to identify network trends and deviations from normal traffic. Disadvantages of decision tree algorithms in IDS are that the output attribute must be categorical (normal or anomaly) and limited to one output attribute. Decision tree algorithms are also known to be unstable and trees created from numeric datasets can be complex.

3. **Random Forests**

   A Random forest is a new approach to data exploration, data analysis and predictive modeling. The first algorithm of Random forest was created by Tin Kam Ho by using the random subspace method and the extension of the random forest algorithm was developed by Leo Breiman, who was the father of CART (R). The random forest is a collection of CART – like trees. The Random Forests grows an ensemble of decision tree. The random

forest algorithm uses the bagging technique for bagging technique for building an ensemble of decision trees. Bagging is also known to reduce the variance of algorithm. The ensembles are more effective when the individual models that comprise them are uncorrelated. In traditional bagging with decision trees, the constituent decision trees may end up to be very correlated because the same features will tend to be used repeatedly to split the bootstrap samples. By restricting each split-test to a small, random sample of features, we can decrease the correlation between trees in the ensemble.

V. PERFORMANCE METRICS

The performance metrics used to evaluate classification:

Confusion Matrix:

It contains information about actual and predicted classifications done by a classification system.

		Predicted	
		Normal	Attack
Actual	Normal	TP	FN
	Attack	FP	TN

i. False positive (FP): It defines the number of detected attacks which are actually normal.

ii. False negative (FN): It means wrong prediction i.e. it means it detects instances a normal but in actual they are attacks.

iii. True positive (TP): It means instances that are correctly predicted as normal. iv. True negative (TN): It means instances that are correctly classified or detected as attack.

v. Accuracy: It is the percentage of correct predictions. On the basis of Confusion Matrix, it is calculated by using the formula below:

$$Accuracy = TP+TN/n$$

n is total number of instances.

vi. Mean Absolute Error: It is the mean of overall error made by classification algorithm. Least the error and best will be the classifier.

vii. TPR: True Positive Rate is same as accuracy so we have not considered this metrics. FPR: False Positive Rate is calculated by using the formula:

$$FPR=FPITN+FP$$

viii. Recall: It is the proportion of instances belonging to the positive class that are correctly predicted as positive.

$$Recall=TPITP+FN$$

ix. Precision: It is a measure which estimates the probability that a positive prediction is correct

$$Precision = TPITP+FP$$

x. Training time: It is the time taken by Classifier to build the model on dataset. It is usually measured in seconds.

xi. Kappa: Its value ranges from 0 to 1. 0 means totally disagreement and I means full agreement. It checks the reliability of Classifying algorithm on dataset.

xii. ROC (Receiver Operating Characteristics): It is used to design tdhe curve between TPR and FPR and the area under curve is called as AUC gives the value of ROC. More the area under curve and more will be the value of ROC.

VI. CODE

```
import pandas as pd  from sklearn.metrics import
accuracy_score from sklearn.metrics import
confusion_matrix from sklearn.model_selection
import train_test_split from sklearn.ensemble import
RandomForestClassifier from sklearn.tree import
DecisionTreeClassifier from sklearn.naive_bayes
import GaussianNB  import matplotlib.pyplot as plt;
plt.rcdefaults()
import numpy as np
import matplotlib.pyplot as plt

#importing dataset
#dataset = pd.read_csv('C:\\Users\\Puchu\\.spyder-py3\\kddtrain+_20percent.csv')  file_handler
= open("C:\\Users\\Puchu\\.spyder-py3\\KDDTrain+.csv", "r")

# creating a Pandas DataFrame  # using
read_csv function   # that reads from a csv
file.  dataset = pd.read_csv(file_handler, sep
= ",")

# closing the file handler
file_handler.close()

# creating a dict file
protocol_type = {'tcp': 1,'udp': 2,'icmp':3}
flag = { 'OTH':1,'REJ':2,'RSTO':3,'RSTOS0':4,'RSTR':5,'S0':6,'S1':7,'S2':8,'S3':9,'SF':10,'SH':11}
service =
{'aol':1,'auth':2,'bgp':3,'courier':4,'csnet_ns':5,'ctf':6,'daytime':7,'discard':8,'domain':9,'domain_u':1
0,'echo':11,'eco_i':12,'ecr_i':13,'efs':14,'exec':15,'finger':16,'ftp':17,'ftp_data':18,'gopher':19,'harve
st':20,'hostnames':21,'http':22,'http_2784':23,'http_443':24,'http_8001':25,'imap4':26,'IRC':27,'iso
_tsap':28,'klogin':29,'kshell':30,'ldap':31,'link':32,'login':33,'mtp':34,'name':35,'netbios_dgm':36,'n
etbios_ns':37,'netbios_ssn':38,'netstat':39,'nnsp':40,'nntp':41,'ntp_u':42,'other':43,'pm_dump':44,'p
op_2':45,'pop_3':46,'printer':47,'private':48,'red_i':49,'remote_job':50,'rje':51,'shell':52,'smtp':53,'s
ql_net':54,'ssh':55,'sunrpc':56,'supdup':57,'systat':58,'telnet':59,'tftp_u':60,'tim_i':61,'time':62,'urh_
i':63,'urp_i':64,'uucp':65,'uucp_path':66,'vmnet':67,'whois':68,'X11':69,'Z39_50':70}

# traversing through dataframe
# protocol_type,flag, service column and writing
```

```
# values where key matches
dataset.protocol_type = [protocol_type[item] for item in dataset.protocol_type]
dataset.flag = [flag[item] for item in dataset.flag]  dataset.service =
[service[item] for item in dataset.service]  print(dataset)

#printing the head of the dataset dataset.head()

#splitting dataset into features and class X
= dataset.iloc[:, 0:41].values
y = dataset.iloc[:, 41].values


#splitting dataset into train and test set
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)
#training the algorithm
# Create the RandomForest model
classifier1 = RandomForestClassifier(n_estimators=200, bootstrap = True, max_features = 'sqrt')
#creating the Decision Tree model
classifier2 = DecisionTreeClassifier(criterion = "entropy", max_depth=10)
#creating the Naive Bayes
classifier3 = GaussianNB(priors=None, var_smoothing=1e-09)

# Fit on training data
# training RandomForest
clf1 = classifier1.fit(X_train, y_train)
#training decision tree
clf2 = classifier2.fit(X_train, y_train)
#training Naive Bayes
clf3 = classifier3.fit(X_train, y_train)

#prediction
#for RandomForest pred1 =
clf1.predict(X_test)
#for decision tree
pred2 = clf2.predict(X_test) #for
Naive Bayes
pred3 = clf3.predict(X_test)

#confusion matrix. #for
RandomForest
cm1=confusion_matrix(y_test,pred1)
cm1
#for decision tree
cm2=confusion_matrix(y_test,pred2)
cm2
```

```
#for Naive Bayes
cm3=confusion_matrix(y_test,pred3)
cm3

#accuracy matrix #for
RandomForest
ac1=100*accuracy_score(y_test,pred1) ac1
#for decision tree
ac2=100*accuracy_score(y_test,pred2) ac2
#for Naive Bayes
ac3=100*accuracy_score(y_test,pred3) ac3

#plotting a bar graph
objects = ('NaiveBayes','Decision Tree', 'RandomForest',) y_pos
= np.arange(len(objects))
performance = [ac3,ac2,ac1]

plt.bar(y_pos, performance, align='center', alpha=0.5)
plt.xticks(y_pos, objects) plt.ylabel('Usage')
plt.title('Classifier Accuracy')

plt.show()
```

VII. RESULTS

Confusion Matrix:

- **Naïve Bayes**

```
In [33]: cm3=confusion_matrix(y_test,pred3)
    ...: cm3
Out[33]:
array([[  212, 11639],
       [  241, 13103]], dtype=int64)
```

- **Decion Tree**

```
In [32]: cm2=confusion_matrix(y_test,pred2)
    ...: cm2
Out[32]:
array([[11830,    21],
       [   41, 13303]], dtype=int64)
```

- **Random Forest**

```
In [31]: cm1=confusion_matrix(y_test,pred1)
    ...: cm1
Out[31]:
array([[11831,    20],
       [   11, 13333]], dtype=int64)
```

Accuracy:

- **Naïve Bayes**

```
In [30]: ac3=100*accuracy_score(y_test,pred3)
    ...: ac3
Out[30]: 52.84778725937686
```

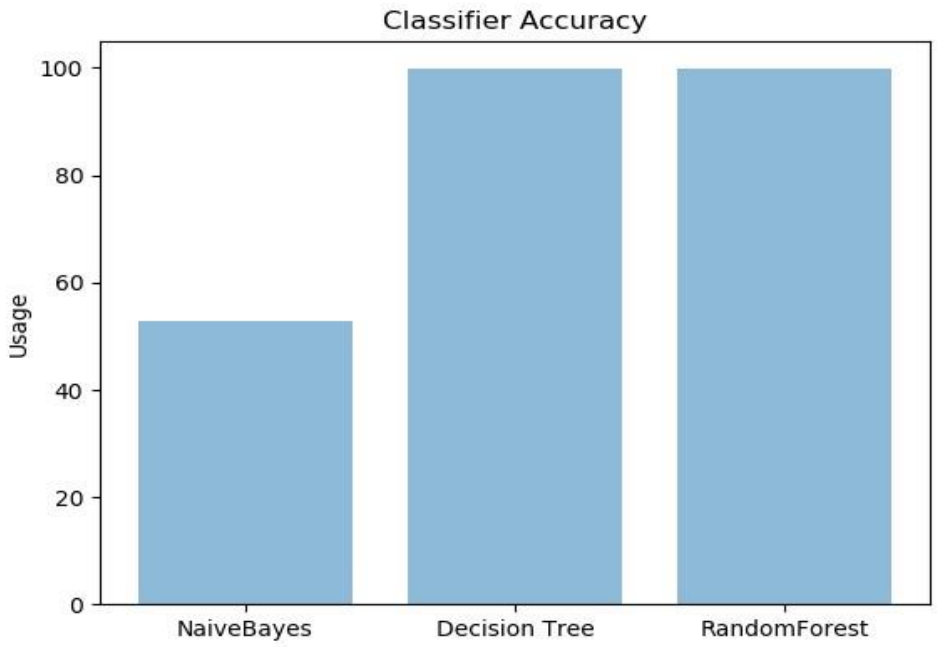
- **Decision Tree**

```
In [27]: ac2=100*accuracy_score(y_test,pred2)
    ...: ac2
Out[27]: 99.75391942845803
```

- **Random Forest**

```
In [26]: ac1=100*accuracy_score(y_test,pred1)
    ...: ac1
Out[26]: 99.87695971422902
```

Bar graph:



VIII. CONCLUSION

The comparative study concludes that, the Random Forest classifier performed the best on NSLKDD dataset, providing an accuracy of 99.87, followed by the Decision Tree classifier which provided an accuracy of 99.75. The accuracy of the Naïve Bayes classifier is 52.84 which is considerably less in comparison to the other two classifiers used.