REAL TIME NETWORK ANOMALY DETECTION USING BIG DATA

Umangkumar Patel
Information and Communication technology
School of Engineering and Applied Science,
Ahmedabad University
Ahmedabad, India
umangkumar.p.btechi16@ahduni.edu.in

Ayush Panara
Information and Communication technology
School of Engineering and Applied Science,
Ahmedabad University
Ahmedabad, India
ayush.p.btechi16@ahduni.edu.in

Yash Mehta
Information and Communication technology
School of Engineering and Applied Science,
Ahmedabad University
Ahmedabad, India
yash.m.btechi16@ahduni.edu.in

***Abstract*—Nowadays, Cyber-attacks have become very systematic and complex. The high computation volume and consistent changes in network traffic have made it harder to break down information and recognize anomalous practices inside. Thus, large data techniques have turned out to be essential to identify attacks and block them. Use of large data analytics in security creates the ability to gather massive amounts of digital information to analyze and draw insights that can make it possible to predict and stop cyber-attacks. Traditional frameworks are inefficient because they mostly rely on signature based structure they can't distinguish latest attack vectors and it is hard to carry out operations and examination of tremendous measure of security information at the same time**

Keywords—Cyber-attack, Big data, Random Forest, Artificial Neural Network

I. INTRODUCTION

Nowadays, Cyber-attacks have become very systematic and complex. An average website with 30K+ daily users generates around 10, 00,000 events per day and the devices in the average enterprise's internal network generates 10,000 events per day. The high computation volume and consistent changes in network traffic have made it harder to break down information and recognize the anomalous practices happing inside. Thus, large data techniques have turned out to be essential to identify attacks and block them. Use of large data analytics in security creates the ability to gather massive amounts of digital information to analyse and draw insights that can make it possible to predict and stop cyber-attacks. Traditional frameworks are inefficient because they mostly rely on the signature-based structure they can't distinguish the latest attack vectors and it is hard to carry out operations and examination of tremendous measure of security alerts at the same time.

And to add to this problem that we are currently facing through, comes the internet if things, where according to a recent survey, a person will be having approximately 7 sensors attack to its body in near future which will be monitoring his actions and performing relative operations based on the data received. But this increment in the connectivity of number of devices, will further increase the problem rate to exponentially high because at that point of time, when the world would be moving towards 5G even a common man's data will be at stack apart from the private and government firms and all this will result into an upsurge in the number of events generated per day. And thus security backed and reinforced by bid data will ensure that the security threats are remediated immediately.

The goal of this paper is analyse the classification techniques which have previously proved useful in data mining and classification problems and further compare and map out the best performing data classification technique on regard to this problem. There are many algorithms which are currently available, out of which we have focused on the techniques like Gaussian Naïve Bayes, Artificial Neural Networks(ANNs), Decision Trees and Finally Random Forest Classification. After which through a series of analysis we keep eliminating the east performing techniques and take the rest if the techniques for the next level of test. At last we compare each of the remaining techniques and proposes a techniques which can useful in Network Intrusion Detection Systems (NIDSs) and particularly in Anomaly Detection Systems (ADSs) reason being they perform key role in identifying weather the network in being intruded or not.

The Dataset which we taken into account here is UNSW-NB 15 dataset that created through IXIA PerfectStorm tool in the Cyber Range Lab of the Australian Centre for Cyber Security (ACCS) and the reason behind selecting the above dataset was its focus on modern attacks fashions and new patterns of normal traffic. The dataset has in total 49 Different attributes which comprise of flow based between hosts and the network packets inspection to discriminate between the observations, either normal or abnormal. Also, the set focuses on nine types of attacks, namely, Fuzzers, Analysis, Backdoors, DoS, Exploits, Generic, Reconnaissance, Shellcode and Worms. Tcpdump was used in collecting raw data required for making this set which was around 100GB after which, the data was filtered and scaled down to 49 different attributes. Thus, when the initial data was plotted on a scatter plot, this was the output received.

The section 2 talks about the method and our approach towards solving the problem, Section 3 talks about the experimental results that we have obtained when a certain method is applied. Finally Section 4 talks about the future scope concludes the paper.

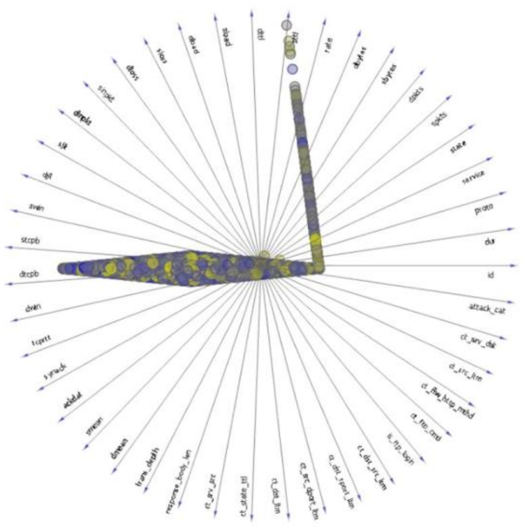


Fig. 1. Classification of attack category of raw data

II. Methodology

In order to perform the above mentioned details, the first thing that we planned to do was the statistical analysis of the observations and the attributes are explained and then pre-process the data based on filtering. The aim behind doing this process was to make sure that we remove empty entries, invalid (NaN) data, etc. Also, it was pre-processing which was able to give us some insights on the dataset on which the work as we were able get information like: types of attacks available in the dataset, number of attack and normal data instances in the dataset, unique values for each row, unique protocols in the dataset and finally information ordinal features like Source Time to live (sttl).

The process which followed afterwards was focused on deriving some preliminary information from the dataset like: Fraction of connections that are attacks, correlation for features in the full dataset and performed operation like Column-Specific Transformations Experimentation which was performed to ensure that the last two fields namely attack category and attack label are removed from the dataset. The second which we focused on after filtering and getting some primary information dataset was to select the classification that we will be working on in this project. The experiment was started from every basic technique of classification technique namely Gaussian naive Bayes Theorem. The two main reasons behind starting from Bayes theorem was firstly, it has low computer efficiency and secondly it is able to perform well in text processing which highly correlates with the current dataset for the reason being the probability of occurrence of any word given the class label, is independent of the probability of occurrence of any other word, given that label. But, the ROC gave an accuracy just around 86% which is quite low and thus we shifted towards a more shift technique namely Artificial Neural Networks. The reason behind selecting this technique was the reason that neural networks are trained on Back propagation algorithm which fairly work with most of th classification problems, further they have an added advantage of different combinations of functions. Thus, as a back propagation neural network (BPNN) can be highly successful component for a dataset classification with a suitable combination of training, learning and transfer functions, it was used for the current problem's classification with the combination of ReLu activation function for the convolutional layers, tanh for the deep layers and at last SoftMax function. But here the average precision accuracy achieved was just around 78% and similar was achieved when tested combination like ReLu-ReLu-Softmax, tanh-ReLu-Softmax, etc. Thus, we went towards number focused classification technique named Keras Classification, and thus gave us some promising results which fell at around 94% of accuracy.

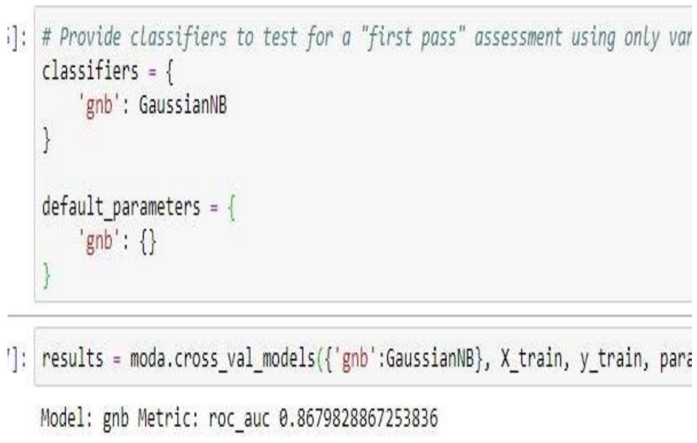
We, than focused on Decision trees reason being here, we do not have to implicitly perform any feature extraction for the dataset, also it has immunity towards the non-linear relationships of the features and probability of such relations being present in his dataset are quit high. But, only things worth worrying for while implementing this classification technique is over fitting, reason being decision trees follow pruning which is greedy in nature and thus tries to gain a locally optimal solution rather than looking for a broader scope and similar kind of thing occurred in current dataset because when tested on trained dataset it gave accuracy of 99% where in checked on the remaining section of dataset, accuracy fell to just 94%.

Finally we focused on random Forest Technique, reason being it one of the supervised learning algorithm that produces great result without hyper-parameter modification, in most of the cases. Moreover it is an ensemble of decision trees in which it builds multiple decision trees and then merges them together to get a more accurate result. And similar to what we had thought, the technique as able to give us an accuracy of 99% when tested on the training section and 95% when testes on the remaining section of the dataset which was highest of all the classifiers worked till now and thus further experimentation were performed using this technique. As, our main goal was to obtained an optimal trade-off between computational usage and accuracy, feature extraction was performed and then same classification was trained using 5,10,21 and compared with the accuracy achieved when all attributes were used. Whose further details are discussion experimental results section.

III. Experimental Results

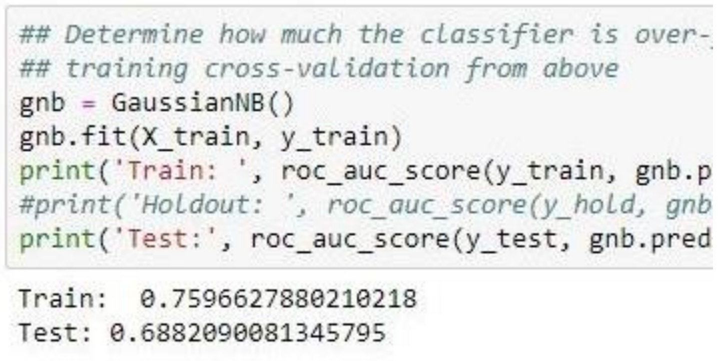
The dataset on which the experimentation was performed contained 175341 instances in which the target was column label which gave a binary output (i.e.: 1 for the attack, 0 for no attack). Starting with Gaussian Naive Bayes, it can be seen that there were many places it lacked behind because as seen in Figure 2, the accuracy was better but it decreased to as short as 68% when tested against in the remaining dataset.

Similar can been seen in our Figure 4 in which ANN with combination of ReLu-tanh-Softmax gave an accuracy of approximately 78%.



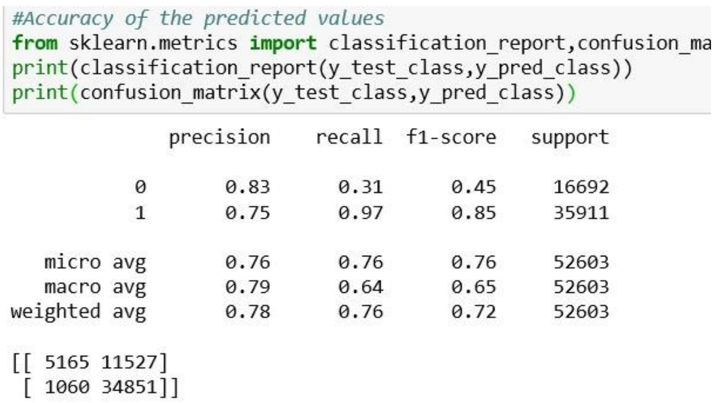
```
]: # Provide classifiers to test for a "first pass" assessment using only var
   classifiers = {
       'gnb': GaussianNB
   }

   default_parameters = {
       'gnb': {}
   }

']: results = moda.cross_val_models({'gnb':GaussianNB}, X_train, y_train, para

   Model: gnb Metric: roc_auc 0.8679828867253836
```

Fig. 2. roc_auc of Gaussian Naïve Bayes



```
## Determine how much the classifier is over-
## training cross-validation from above
gnb = GaussianNB()
gnb.fit(X_train, y_train)
print('Train: ', roc_auc_score(y_train, gnb.p
#print('Holdout: ', roc_auc_score(y_hold, gnb
print('Test:', roc_auc_score(y_test, gnb.pred

Train:  0.7596627880210218
Test: 0.6882090081345795
```

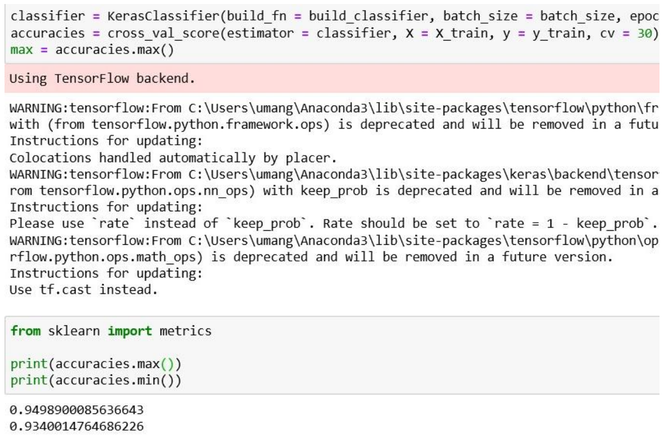
Fig. 3. Accuracy of Gaussian naïve bayes



```
#Accuracy of the predicted values
from sklearn.metrics import classification_report,confusion_ma
print(classification_report(y_test_class,y_pred_class))
print(confusion_matrix(y_test_class,y_pred_class))

              precision    recall  f1-score   support

           0       0.83      0.31      0.45     16692
           1       0.75      0.97      0.85     35911

   micro avg       0.76      0.76      0.76     52603
   macro avg       0.79      0.64      0.65     52603
weighted avg       0.78      0.76      0.72     52603

[[ 5165 11527]
 [ 1060 34851]]
```

Fig. 4. Confusion Matrix and Report of ANN

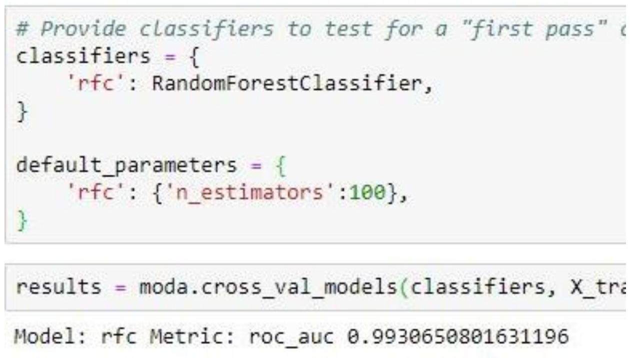
Similarly for the Keras Classifier of ANN, we were able to get the accuracy of 94% which can be seen in Figure 5.



```
classifier = KerasClassifier(build_fn = build_classifier, batch_size = batch_size, epoc
accuracies = cross_val_score(estimator = classifier, X = X_train, y = y_train, cv = 30)
max = accuracies.max()

Using TensorFlow backend.

WARNING:tensorflow:From C:\Users\umang\Anaconda3\lib\site-packages\tensorflow\python\fr
with (from tensorflow.python.framework.ops) is deprecated and will be removed in a futu
Instructions for updating:
Colocations handled automatically by placer.
WARNING:tensorflow:From C:\Users\umang\Anaconda3\lib\site-packages\keras\backend\tensor
rom tensorflow.python.ops.nn_ops) with keep_prob is deprecated and will be removed in a
Instructions for updating:
Please use `rate` instead of `keep_prob`. Rate should be set to `rate = 1 - keep_prob`.
WARNING:tensorflow:From C:\Users\umang\Anaconda3\lib\site-packages\tensorflow\python\op
rflow.python.ops.math_ops) is deprecated and will be removed in a future version.
Instructions for updating:
Use tf.cast instead.

from sklearn import metrics

print(accuracies.max())
print(accuracies.min())

0.9498900085636643
0.9340014764686226
```

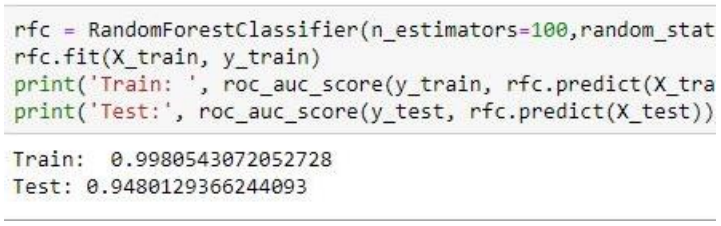
Fig. 5. Artificial Neural Network using KerasClassifier Accuracy

Finally, the Roc along with accuracy was obtained in Random Forest Classification which can be seen in Figure 6 and 7.



```
# Provide classifiers to test for a "first pass" c
classifiers = {
    'rfc': RandomForestClassifier,
}

default_parameters = {
    'rfc': {'n_estimators':100},
}

results = moda.cross_val_models(classifiers, X_tra

Model: rfc Metric: roc_auc 0.9930650801631196
```

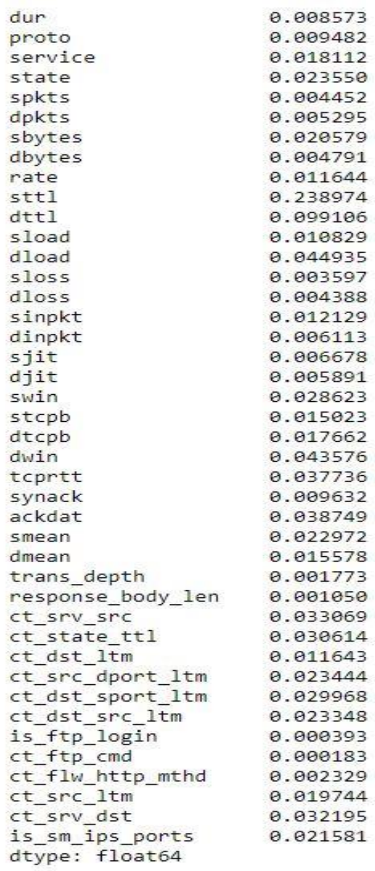
Fig. 6. Cross Value Score of Random Forest Classifier



```
rfc = RandomForestClassifier(n_estimators=100,random_stat
rfc.fit(X_train, y_train)
print('Train: ', roc_auc_score(y_train, rfc.predict(X_tra
print('Test:', roc_auc_score(y_test, rfc.predict(X_test))

Train:  0.9980543072052728
Test: 0.9480129366244093
```

Fig. 7. Accuracy(roc_auc score) of Random Forest Classsifier

Figure 8 shows you the Rank list which was performed on all Variables to describe their importance and Figure 9 gives you the visualization of the same.



```
dur                        0.008573
proto                      0.009482
service                    0.018112
state                      0.023550
spkts                      0.004452
dpkts                      0.005295
sbytes                     0.020579
dbytes                     0.004791
rate                       0.011644
sttl                       0.238974
dttl                       0.099106
sload                      0.010829
dload                      0.044935
sloss                      0.003597
dloss                      0.004388
sinpkt                     0.012129
dinpkt                     0.006113
sjit                       0.006678
djit                       0.005891
swin                       0.028623
stcpb                      0.015023
dtcpb                      0.017662
dwin                       0.043576
tcprtt                     0.037736
synack                     0.009632
ackdat                     0.038749
smean                      0.022972
dmean                      0.015578
trans_depth                0.001773
response_body_len          0.001050
ct_srv_src                 0.033069
ct_state_ttl               0.030614
ct_dst_ltm                 0.011643
ct_src_dport_ltm           0.023444
ct_dst_sport_ltm           0.029968
ct_dst_src_ltm             0.023348
is_ftp_login               0.000393
ct_ftp_cmd                 0.000183
ct_flw_http_mthd           0.002329
ct_src_ltm                 0.019744
ct_srv_dst                 0.032195
is_sm_ips_ports            0.021581
dtype: float64
```

Fig. 8. Feature Importance Score

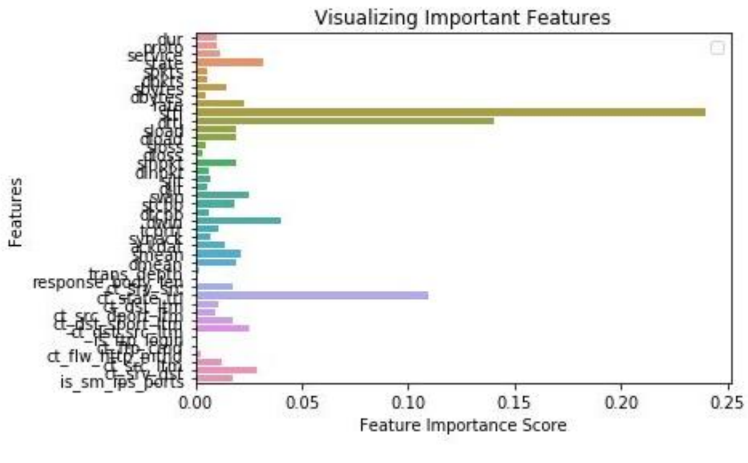
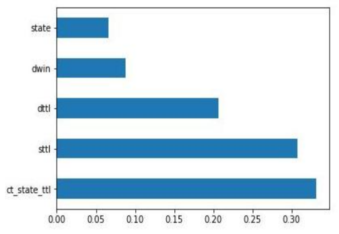
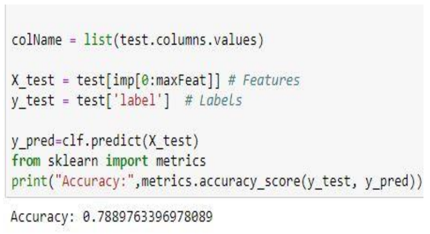


Fig. 9. Feature Importance Score Barplot

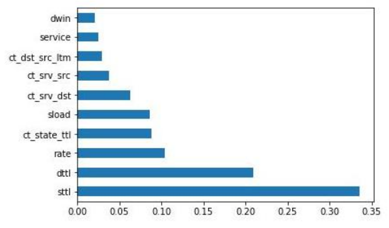
The same technique was than performed on top 5, 10 and 21 variables whose results can be seen in Figure 10, Figure 11 and Figure 12 respectively.

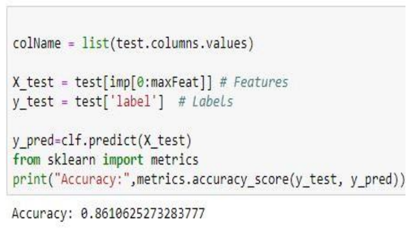




```
colName = list(test.columns.values)

X_test = test[imp[0:maxFeat]] # Features
y_test = test['label']  # Labels

y_pred=clf.predict(X_test)
from sklearn import metrics
print("Accuracy:",metrics.accuracy_score(y_test, y_pred))

Accuracy: 0.7889763396978089
```

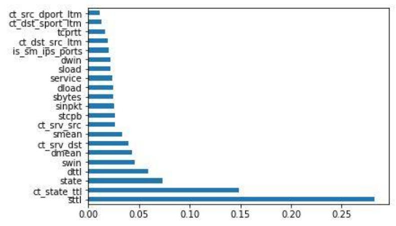
Fig. 10. Most 5 importance feature as training set and accuracy

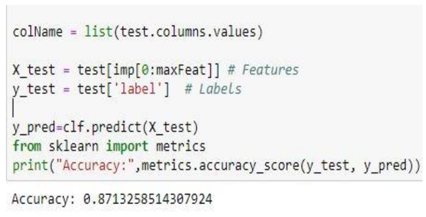




```
colName = list(test.columns.values)

X_test = test[imp[0:maxFeat]] # Features
y_test = test['label']  # Labels

y_pred=clf.predict(X_test)
from sklearn import metrics
print("Accuracy:",metrics.accuracy_score(y_test, y_pred))

Accuracy: 0.8610625273283777
```

Fig. 11. Most 10 importance feature as training set and accuracy





```
colName = list(test.columns.values)

X_test = test[imp[0:maxFeat]] # Features
y_test = test['label']  # Labels

y_pred=clf.predict(X_test)
from sklearn import metrics
print("Accuracy:",metrics.accuracy_score(y_test, y_pred))

Accuracy: 0.8713258514307924
```

Fig. 12. Most 21 importance features as training and accuracy

Finally Figure 13 shows you the change in the accuracy of RFC with unit addition of variable every time.

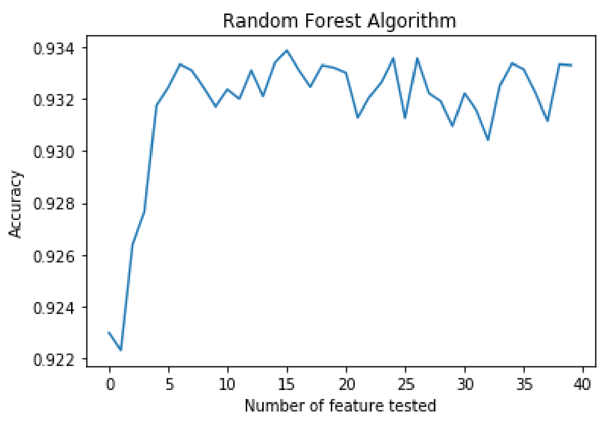


Fig. 13. Random Forest Algorithm

Thus, at the end Figure 14 and Figure 15 shows you the difference between classification before and after respectively.

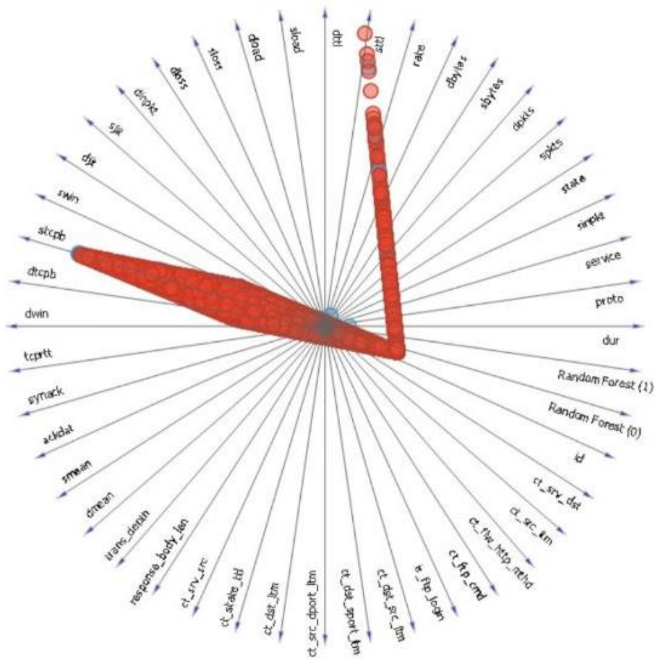


Fig. 14. Initial plot of the cluster data

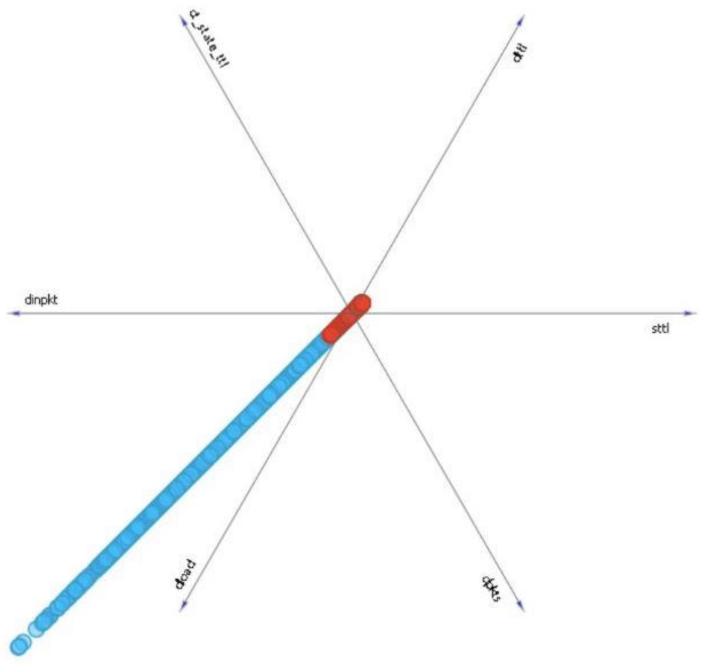


Fig. 15. Final plot after classification of data

IV. Conclusion

It can be clearly seen from the experimental results that all the techniques have been applied and dealt with. It can also be clearly seen that the Random Forest Classifier outperformers all the other techniques in terms of accuracy. Further, the technique has been applied to a group photo and there too a reasonable amount of accuracy has been obtained. Thus, it can be said that Random Forest Classifier (RFC) can provide a fair amount of information regarding the possible occurrence of an attack and thus can be used in places of interest to detect and stop any kind of emergency before it occurs. As for our future, we have currently detected the labels on whether there is an attack or not and thus further plan to implant the algorithm to further classify which kind of attack it is.

References:

[1] Amor, Nahla Ben, et al. “Naive Bayes vs Decision Trees in Intrusion Detection Systems.” *Proceedings of the 2004 ACM Symposium on Applied Computing - SAC 04*, 2004, doi:10.1145/967900.967989.

[2] Baras, John S., and Maben Rabi. “Intrusion Detection with Support Vector Machines and Generative Models.” *Lecture Notes in Computer Science Information Security*, 2002, pp. 32–47., doi:10.1007/3-540-45811-5_3.

[3] Baras, John S., and Maben Rabi. “Intrusion Detection with Support Vector Machines and Generative Models.” *Lecture Notes in Computer Science Information Security*, 2002, pp. 32–47., doi:10.1007/3-540-45811-5_3.

[4] Divekar, Abhishek, et al. “Benchmarking Datasets for Anomaly-Based Network Intrusion Detection: KDD CUP 99 Alternatives.” *2018 IEEE 3rd International Conference on Computing, Communication and Security (ICCCS)*, 2018, doi:10.1109/cccs.2018.8586840.

[5] Marchal, Samuel, et al. “A Big Data Architecture for Large Scale Security Monitoring.” *2014 IEEE International Congress on Big Data*, 2014, doi:10.1109/bigdata.congress.2014.18.

[6] Moustafa, Nour, and Jill Slay. “UNSW-NB15: a Comprehensive Data Set for Network Intrusion Detection Systems (UNSW-NB15 Network Data Set).” *2015 Military Communications and Information Systems Conference (MilCIS)*, 2015, doi:10.1109/milcis.2015.7348942.

[7] Moustafa, Nour, and Jill Slay. “The Evaluation of Network Anomaly Detection Systems: Statistical Analysis of the UNSW-NB15 Data Set and the Comparison with the KDD99 Data Set.” *Information Security Journal: A Global Perspective*, vol. 25, no. 1-3, 2016, pp. 18–31., doi:10.1080/19393555.2015.1125974.

[8] Patcha, Animesh, and Jung-Min Park. "An Overview of Anomaly Detection Techniques: Existing Solutions and Latest Technological Trends." *Computer Networks*, vol. 51, no. 12, 2007, pp. 3448–3470., doi:10.1016/j.comnet.2007.02.001.

[9] Sisodia, Mahendra Singh, and Virendra Raghuwanshi. "Anomaly Base Network Intrusion Detection by Using Random Decision Tree and Random Projection: A Fast Network Intrusion Detection Technique." *Network Protocols and Algorithms*, vol. 3, no. 4, 2011, doi:10.5296/npa.v3i4.1342.

[10] Sung, A.h., and S. Mukkamala. "Identifying Important Features for Intrusion Detection Using Support Vector Machines and Neural Networks." *2003 Symposium on Applications and the Internet, 2003. Proceedings.*, doi:10.1109/saint.2003.1183050.

Appendix A:

Library Code:(inbuilt Function)

(1) accuracy_score Function
from sklearn.metrics import accuracy_score

(2) For Bar Plot of most important feature in dataset
import seaborn as sns(barplot)

(3) Dataset to be splitted in train, test
from sklearn.model_selection import train_test_split

(4) from keras.layers import Dense
Creating Artificial Neural Network

Existing Code:(External Code)

(1) This is used for converting your data into numeric from categorical type as a part of data pre-processing.
```
from sklearn.preprocessing import LabelEncoder
encoder =  LabelEncoder()
y1 = encoder.fit_transform(y)
```

(2) Detailed ananlysis of accuracy score and confusion matrix
```
from sklearn.metrics import classification_report
print(classification_report(y_test_class,y_pred_class)
```

(3) Cross Value Score, ROC_AUC
```
from sklearn.model_selection import cross_val_score
```

(4) KerasClassifier for ANN
```
def build_classifier():
    classifier = Sequential()
    classifier.add(Dense(47,
kernel_initializer="truncated_normal", activation = 'elu', input_shape = (X.shape[1],)))
    classifier.add(Dropout(0.3))
    classifier.add(Dense(4,
kernel_initializer="truncated_normal", activation = 'elu', )) #outputlayer
    classifier.compile(optimizer = 'adam', loss = 'binary_crossentropy', metrics = ["binary_accuracy"])
    return classifier
classifier = KerasClassifier(build_fn = build_classifier, batch_size = 100, epochs = 200, verbose=0)
accuracies = cross_val_score(estimator = classifier, X = X_train, y = y_train,cv=30)
```

Contribution:

Literature Survey	Ayush, Yash
Data Preprocessing	Umang
ANN	Ayush, Yash
Gaussian Naïve Bayes	Yash, Umang
Decision Tree	Umang, Ayush
Random Forest	Umang, Ayush, Yash