NETWORK INTRUSION DETECTION USING

MACHINE LEARNING ALGORITHMS

A PROJECT REPORT

for

INFORMATION SECURITY ANALYSIS AND AUDIT

(CSE3501)

in

B. Tech (Information Technology)

By

KSHITIJ DHYANI

(18BIT0131)

JAHNAVI MISHRA

(18BIT0243)

Fall semester, 2020

Under the Guidance of

Prof. SUMAIYA THASEEN I

Associate Professor Grade 1, SITE

VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

School of Information Technology and Engineering

YOUTUBE VIDEO

https://www.youtube.com/watch?v=aDmXbgoX_K4

GITHUB

https://github.com/wimpywarlord/darknet2020ML

Performance Analysis:

- **Prediction of Label 1 :**

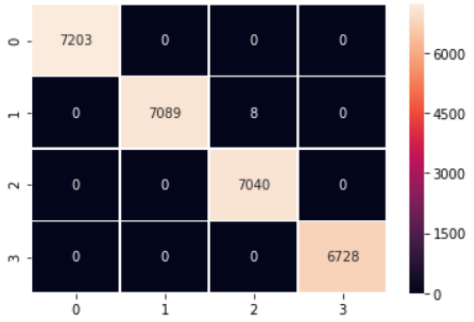
- **After Hyperparameter Tuning:**

Evaluation Metric	Gradient Boosting Classifier	AdaBoost Classifier
ACCURACY	**1.00**	**0.944**
PRECISION	**1.00**	**0.999**
RECALL	**1.00**	**0.969**
F-SCORE	**1.00**	**0.984**
SUPPORT	**Class 0: 7203** **Class 1: 7097** **Class 2: 7040** **Class 3: 6728**	**Class 0: 7116** **Class 1: 7210** **Class 2: 6834** **Class 3: 6908**

- **Confusion Matrix**

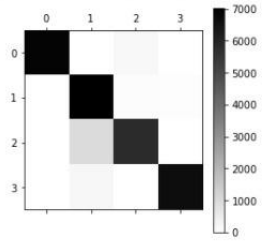
GBM:

```
In [132]: from sklearn.metrics import confusion_matrix
          #Get the confusion matrix
          cf_matrix1 = confusion_matrix(y_test,pred)
          print(cf_matrix1)

          [[7203    0    0    0]
           [   0 7089    8    0]
           [   0    0 7040    0]
           [   0    0    0 6728]]

In [133]: import seaborn as sns
          sns.heatmap(cf_matrix1, annot=True, fmt="d", linewidths=.5)

Out[133]: <matplotlib.axes._subplots.AxesSubplot at 0x214b245cb70>
```



For AdaBoost:

```
In [142]: df_con

Out[142]: array([[6898,   22,  196,    0],
                 [   2, 7032,   94,   82],
                 [   0,  999, 5825,   10],
                 [   0,  242,    1, 6665]], dtype=int64)

In [143]: import matplotlib.pyplot as plt
          def plot_confusion_matrix(df_con, title='Confusion matrix', cmap=plt.cm.gray_r):
              plt.matshow(df_con, cmap=cmap) # imshow
              #plt.title(title)
              plt.colorbar()


          plot_confusion_matrix(df_con)
```



- **Results and Conclusion:**

As seen above, GBM gives better results for prediction of Label 1 in Darknet 2020 as compared to AdaBoost.

➢ **Binary Classification on Label 1:**

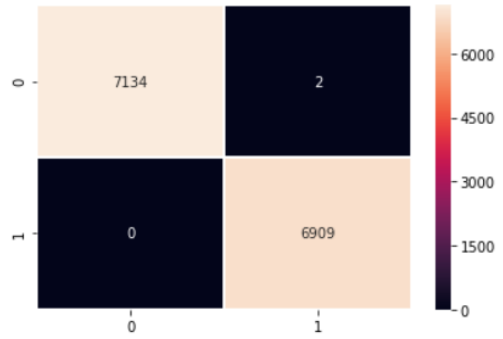
- **After Hyper Parameter Tuning:**

Evaluation Metric	Gradient Boosting Classifier	AdaBoost Classifier
ACCURACY	**1.00**	**1.00**
PRECISION	**1.00**	**1.00**
RECALL	**1.00**	**1.00**
F-SCORE	**1.00**	**1.00**
SUPPORT	**Class 0: 7133** **Class 3: 6898**	**Class 0: 7094** **Class 3: 6935**
AUC_ROC SCORE	**1.00**	**1.00**

- **Confusion Matrix**

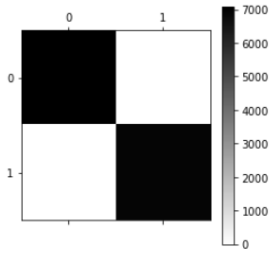
For GBM:

```
In [70]: from sklearn.metrics import confusion_matrix
         #Get the confusion matrix
         cf_matrix1 = confusion_matrix(y_test,pred1)
         print(cf_matrix1)

         [[7134    2]
          [   0 6909]]

In [71]: import seaborn as sns
         sns.heatmap(cf_matrix1, annot=True, fmt="d", linewidths=.5)

Out[71]: <matplotlib.axes._subplots.AxesSubplot at 0x1656da62a58>
```



For AdaBoost:

```
In [75]: from sklearn.metrics import confusion_matrix
         df_con = confusion_matrix(y_test, y_pred)

In [76]: df_con

Out[76]: array([[7094,    0],
                [   0, 6935]], dtype=int64)

In [77]: import matplotlib.pyplot as plt
         def plot_confusion_matrix(df_con, title='Confusion matrix', cmap=plt.cm.gray_r):
             plt.matshow(df_con, cmap=cmap) # imshow
             #plt.title(title)
             plt.colorbar()


         plot_confusion_matrix(df_con)
```

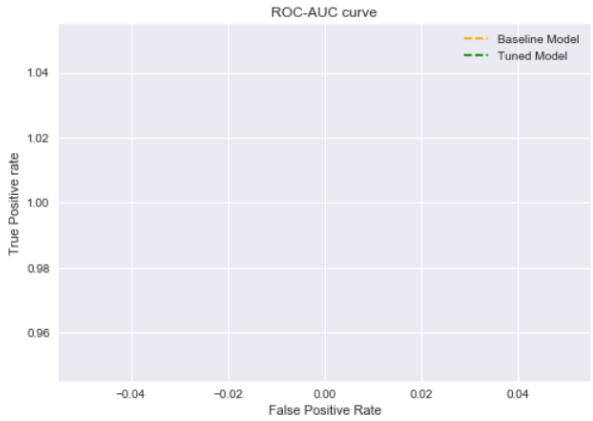


- **AUC_ROC Curve:**

For GBM:

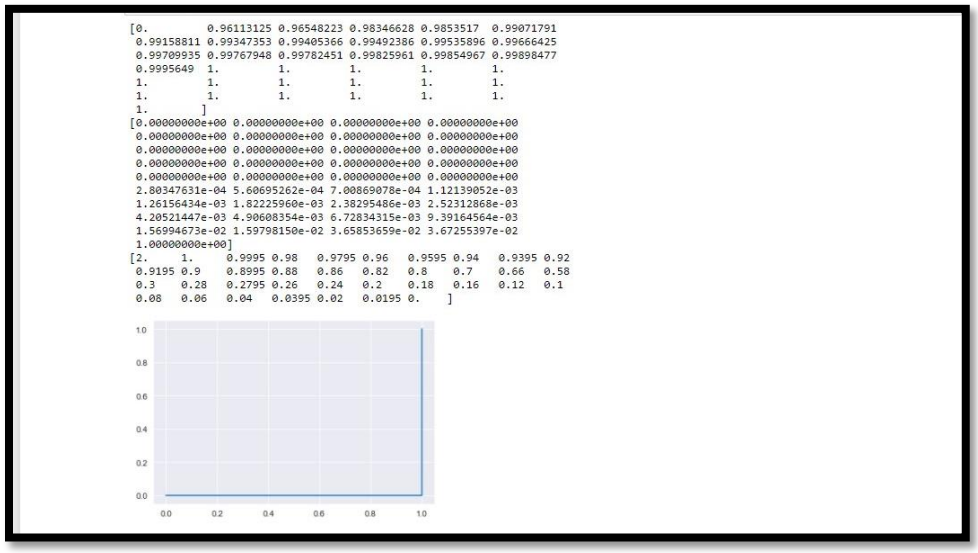
```
Baseline Model: ROC AUC=1.000
Tuned Model: ROC AUC=1.000
```

```
C:\Users\JAHNAVI MISHRA\Anaconda3\lib\site-packages\sklearn\metrics\_ranking.py:813: UndefinedMetricWarning: No positive samples in y_true, true positive value should be meaningless
  UndefinedMetricWarning)
```



The graph is blank because auc score for both the models is 1. This means that the area under the roc_curve is 1 and hence whole graph is covered under that area. So the graph is blank

For AdaBoost:



```
[0.         0.96113125 0.96548223 0.98346628 0.9853517  0.99071791
 0.99158811 0.99347353 0.99405366 0.99492386 0.99535896 0.99666425
 0.99709935 0.99767948 0.99782451 0.99825961 0.99854967 0.99898477
 0.9995649  1.         1.         1.         1.         1.
 1.         1.         1.         1.         1.         1.
 1.         1.         1.         1.         1.         1.
 1.        ]
[0.00000000e+00 0.00000000e+00 0.00000000e+00 0.00000000e+00
 0.00000000e+00 0.00000000e+00 0.00000000e+00 0.00000000e+00
 0.00000000e+00 0.00000000e+00 0.00000000e+00 0.00000000e+00
 0.00000000e+00 0.00000000e+00 0.00000000e+00 0.00000000e+00
 0.00000000e+00 0.00000000e+00 0.00000000e+00 0.00000000e+00
 0.00000000e+00 0.00000000e+00 0.00000000e+00 0.00000000e+00
 2.80347631e-04 5.60695262e-04 7.00869078e-04 1.12139052e-03
 1.26156434e-03 1.82225960e-03 2.38295486e-03 2.52312868e-03
 4.20521447e-03 4.90608354e-03 6.72834315e-03 9.39164564e-03
 1.56994673e-02 1.59798150e-02 3.65853659e-02 3.67255397e-02
 1.00000000e+00]
[2.     1.     0.9995 0.98   0.9795 0.96   0.9595 0.94   0.9395 0.92
 0.9195 0.9    0.8995 0.88   0.86   0.82   0.8    0.7    0.66   0.58
 0.3    0.28   0.2795 0.26   0.24   0.2    0.18   0.16   0.12   0.1
 0.08   0.06   0.04   0.0395 0.02   0.0195 0.    ]
```


The graph is blank because auc score for both the models is 1.

This means that the area under the roc_curve is 1 and hence whole graph is covered under that area.

So the graph is blank.

- **Results and Conclusion:**

As seen above, both GBM and AdaBoost give similar results for binary classification of Label 1 in Darknet 2020 and are equally and highly efficient.

However GBM Still has a little bit of an upper edge.

➢ Prediction of Label 2:

The accuracy obtained by GBM before tuning was around 0.65-0.75 but by AdaBoost it came out to be around 0.18-0.21 only.

After tuning, the best accuracy GBM obtained was 0.87.

So GBM is better for this prediction and AdaBoost is not at all fit for this prediction. Results depend upon the target label and dataset as well. So same models may work differently for different labels of a same dataset.

GBM:

Accuracy of the GBM on test set after tuning : 0.866

	precision	recall	f1-score	support
0.0	0.90	0.96	0.93	14665
1.0	0.89	0.86	0.87	14541
2.0	0.90	0.94	0.92	14571
3.0	0.83	0.66	0.74	14609
4.0	0.73	0.75	0.74	14480
5.0	0.88	0.82	0.85	14632
6.0	0.92	0.90	0.91	14507
7.0	0.98	0.97	0.98	14583
8.0	0.75	0.88	0.81	14453
9.0	0.79	0.80	0.80	14490
10.0	0.95	0.98	0.96	14585
accuracy			0.87	160116
macro avg	0.87	0.87	0.86	160116
weighted avg	0.87	0.87	0.86	160116

AdaBoost :

Kshitij Tired using his algorithm of adaboost and achieved a accuracy of just 2

In [50]:
```
# Create adaboost classifer object
abc = AdaBoostClassifier(n_estimators=50,
                         learning_rate=1)
# Train Adaboost Classifer
model = abc.fit(X_train, y_train)

#Predict the response for test dataset
y_pred = model.predict(X_test)
```

```
C:\Users\kshitij\anaconda3\lib\site-packages\sklearn\utils\validation.py:72: DataConversionWarning: A
column-vector y was passed when a 1d array was expected. Please change the shape of y to (n_samples,
), for example using ravel().
  return f(**kwargs)
```

In [51]:
```
# Model Accuracy, how often is the classifier correct?
print("Accuracy:",metrics.accuracy_score(y_test, y_pred))
```

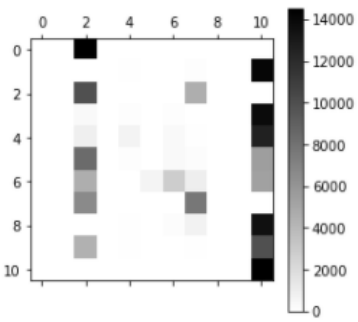
```
Accuracy: 0.2235316895250943
```


```
Accuracy: 0.2235316895250943
```


In [56]:
```
from sklearn.metrics import precision_recall_fscore_support as score


precision, recall, fscore, support = score(y_test, y_pred)

print('precision: {}'.format(precision))
print('recall: {}'.format(recall))
print('fscore: {}'.format(fscore))
print('support: {}'.format(support))
```

```
precision: [0.         0.11864407 0.19974005 0.         0.5538099  0.00400534
 0.72945033 0.52994543 0.         0.         0.16135527]
recall: [0.00000000e+00 4.85268631e-04 6.82868471e-01 0.00000000e+00
 4.81886535e-02 2.04123290e-04 1.97906690e-01 5.37684269e-01
 0.00000000e+00 0.00000000e+00 1.00000000e+00]
fscore: [0.00000000e+00 9.66583817e-04 3.09075157e-01 0.00000000e+00
 8.86625165e-02 3.88450084e-04 3.11343091e-01 5.33786801e-01
 0.00000000e+00 0.00000000e+00 2.77874095e-01]
support: [14533 14425 14628 14558 14630 14697 14618 14449 14538 14615 14425]
```

In [59]:
```
import matplotlib.pyplot as plt
def plot_confusion_matrix(df_con, title='Confusion matrix', cmap=plt.cm.gray_r):
    plt.matshow(df_con, cmap=cmap) # imshow
    #plt.title(title)
    plt.colorbar()


plot_confusion_matrix(df_con)
```



Code And Output:

➢ Importing Dataset:

```
In [1]: #importing python libraries
        import numpy as np
        import pandas as pd

In [2]: #Reading the dataset: Darknet 2020
        df = pd.read_csv('darknet.csv', error_bad_lines=False)
        df.head(10)

b'Skipping line 328: expected 85 fields, saw 125\n'
C:\Users\JAHNAVI MISHRA\Anaconda3\lib\site-packages\IPython\core\interactiveshell.py:3057: DtypeWarning: Columns (20,21) have mixed types. Specify dtype option on import or set low_memory=False.
  interactivity=interactivity, compiler=compiler, result=result)
```

Out[2]:

	Flow ID	Src IP	Src Port	Dst IP	Dst Port	Protocol	Timestamp	Flow Duration	Total Fwd Packet	Total Bwd packets	...	Active Mean	Active Std	Active Max	Active Min	Idle M
0	10.152.152.11-216.58.220.99-57158-443-6	10.152.152.11	57158	216.58.220.99	443	6	24/07/2015 04:09:48 PM	229	1	1	...	0	0	0	0	0.000000e
1	10.152.152.11-216.58.220.99-57159-443-6	10.152.152.11	57159	216.58.220.99	443	6	24/07/2015 04:09:48 PM	407	1	1	...	0	0	0	0	0.000000e
2	10.152.152.11-216.58.220.99-57160-443-6	10.152.152.11	57160	216.58.220.99	443	6	24/07/2015 04:09:48 PM	431	1	1	...	0	0	0	0	0.000000e
3	10.152.152.11-74.125.136.120-49134-443-6	10.152.152.11	49134	74.125.136.120	443	6	24/07/2015 04:09:48 PM	359	1	1	...	0	0	0	0	0.000000e
4	10.152.152.11-173.194.65.127-34697-19305-6	10.152.152.11	34697	173.194.65.127	19305	6	24/07/2015 04:09:45 PM	10778451	591	400	...	0	0	0	0	1.437765e

➢ Data Cleaning:

Data Cleaning Stage:

In this stage, I will be performing data cleaning like dealing with missing values,inf values,changing the columns like splitting columns if required. So when all the columns that may cause error further due to their datatype or format or problematic values will be rectified and then we can proceed towards data preprocessing stage.

```
In [5]: df.dropna() #dropping null values
```

Out[5]:

	Flow ID	Src IP	Src Port	Dst IP	Dst Port	Protocol	Timestamp	Flow Duration	Total Fwd Packet	Total Bwd packets	...	Active Mean	Active Std	Active Max	Active Min
0	10.152.152.11-216.58.220.99-57158-443-6	10.152.152.11	57158	216.58.220.99	443	6	24/07/2015 04:09:48 PM	229	1	1	...	0	0	0	0
1	10.152.152.11-216.58.220.99-57159-443-6	10.152.152.11	57159	216.58.220.99	443	6	24/07/2015 04:09:48 PM	407	1	1	...	0	0	0	0
2	10.152.152.11-216.58.220.99-57160-443-6	10.152.152.11	57160	216.58.220.99	443	6	24/07/2015 04:09:48 PM	431	1	1	...	0	0	0	0
3	10.152.152.11-74.125.136.120-49134-443-6	10.152.152.11	49134	74.125.136.120	443	6	24/07/2015 04:09:48 PM	359	1	1	...	0	0	0	0
4	10.152.152.11-173.194.65.127-34697-19305-6	10.152.152.11	34697	173.194.65.127	19305	6	24/07/2015 04:09:45 PM	10778451	591	400	...	0	0	0	0

```
In [10]: df1 = pd.DataFrame(newIP)  #a new dataframe with the above obtained series
         df1.head()
```

Out[10]:

	0
0	10.152
1	10.152
2	10.152
3	10.152
4	10.152

```
In [11]: df['Src IP'] = df1  #replacing column Src IP with df1
         df.head()
```

Out[11]:

	Flow ID	Src IP	Src Port	Dst IP	Dst Port	Protocol	Timestamp	Flow Duration	Total Fwd Packet	Total Bwd packets	...	Active Mean	Active Std	Active Max	Active Min	Idle Mean	
0	10.152.152.11-216.58.220.99-57158-443-6	10.152	57158	216.58.220.99	443	6	24/07/2015 04:09:48 PM	229	1	1	...	0	0	0	0	0.000000e+00	0.
1	10.152.152.11-216.58.220.99-57159-443-6	10.152	57159	216.58.220.99	443	6	24/07/2015 04:09:48 PM	407	1	1	...	0	0	0	0	0.000000e+00	0.
2	10.152.152.11-216.58.220.99-57160-443-6	10.152	57160	216.58.220.99	443	6	24/07/2015 04:09:48 PM	431	1	1	...	0	0	0	0	0.000000e+00	0.
3	10.152.152.11-74.125.136.120-49134-443-6	10.152	49134	74.125.136.120	443	6	24/07/2015 04:09:48 PM	359	1	1	...	0	0	0	0	0.000000e+00	0.
4	10.152.152.11-173.194.65.127-34697-19305-6	10.152	34697	173.194.65.127	19305	6	24/07/2015 04:09:45 PM	10778451	591	400	...	0	0	0	0	1.437765e+15	3.

➢ Data Pre-Processing:

Data Preprocessing:Encoding,Sampling,Normalisation

In [15]:
```
# label encoding the data : Label and Label.1
from sklearn.preprocessing import LabelEncoder

Le = LabelEncoder()

df['Label']= Le.fit_transform(df['Label'])
df['Label.1']= Le.fit_transform(df['Label.1'])
```

In [16]:
```
df.head()
```

Out[16]:

	Flow ID	Src IP	Src Port	Dst IP	Dst Port	Protocol	Timestamp	Flow Duration	Total Fwd Packet	Total Bwd packets	...	Active Mean	Active Std	Active Max	Active Min	Idle Mean	Idle
0	10.152.152.11-216.58.220.99-57158-443-6	10.152	57158	216.580	443	6	24/07/2015 04:09:48 PM	229	1	1	...	0	0	0	0	0.000000e+00	0.000000
1	10.152.152.11-216.58.220.99-57159-443-6	10.152	57159	216.580	443	6	24/07/2015 04:09:48 PM	407	1	1	...	0	0	0	0	0.000000e+00	0.000000
2	10.152.152.11-216.58.220.99-57160-443-6	10.152	57160	216.580	443	6	24/07/2015 04:09:48 PM	431	1	1	...	0	0	0	0	0.000000e+00	0.000000
3	10.152.152.11-74.125.136.120-49134-443-6	10.152	49134	74.125	443	6	24/07/2015 04:09:48 PM	359	1	1	...	0	0	0	0	0.000000e+00	0.000000
4	10.152.152.11-173.194.65.127-34697-19305-6	10.152	34697	173.194	19305	6	24/07/2015 04:09:45 PM	10778451	591	400	...	0	0	0	0	1.437765e+15	3.117718

In [26]:
```
df5=df5.astype(float)
```

Below is an attempt to find the index of the maximum value in each column. In this way I will come to know if there is inf in any column and if it is there then what is the row number. These values will then be removed so that there is no error during normalisation.

In [27]:
```
# find the index position of maximun
# values in every column
maxValueIndex = df5.idxmax()

print("Maximum values of columns are at row index position :")
print(maxValueIndex)
```

```
Maximum values of columns are at row index position :
Src IP                         58156
Src Port                      109235
Dst IP                         94335
Dst Port                       43964
Protocol                         263
Flow Duration                 139480
Total Fwd Packet              101860
Total Bwd packets             101860
Total Length of Fwd Packet    101402
Total Length of Bwd Packet    101860
Fwd Packet Length Max         100752
Fwd Packet Length Min          32887
Fwd Packet Length Mean        101402
Fwd Packet Length Std         100791
Bwd Packet Length Max         112979
Bwd Packet Length Min          99924
Bwd Packet Length Mean        100636
Bwd Packet Length Std         100715
Flow Bytes/s                   35853
Flow Packets/s                 32902
Flow IAT Mean                  99045
Flow IAT Std                   93397
```

Sampling :Oversampling and Undersampling

Sampling is used to balance the data i.e to balance the number of each class in the Label. Oversampling: to increase the number of minority classes
Undersampling: to decease the number of majority classes

In [30]:
```
df5['Label'].value_counts()
```

Out[30]:
```
0.0    93309
1.0    23861
3.0    22919
2.0     1392
Name: Label, dtype: int64
```

So the data is highly imbalanced.....

Now I will undersample class 0.0 wrt class 1.0 and oversample class 3.0 wrt class 2.0

In [31]:
```
#1. Find the number of the minority class
non_tor = len(df5[df5['Label']==0])
non_vpn = len(df5[df5['Label']==1])
vpn = len(df5[df5['Label']==2])
tor = len(df5[df5['Label']==3])

print(non_tor)
print(non_vpn)
print(vpn)
print(tor)
```

```
93309
23861
1392
```

➢ Feature Extraction:

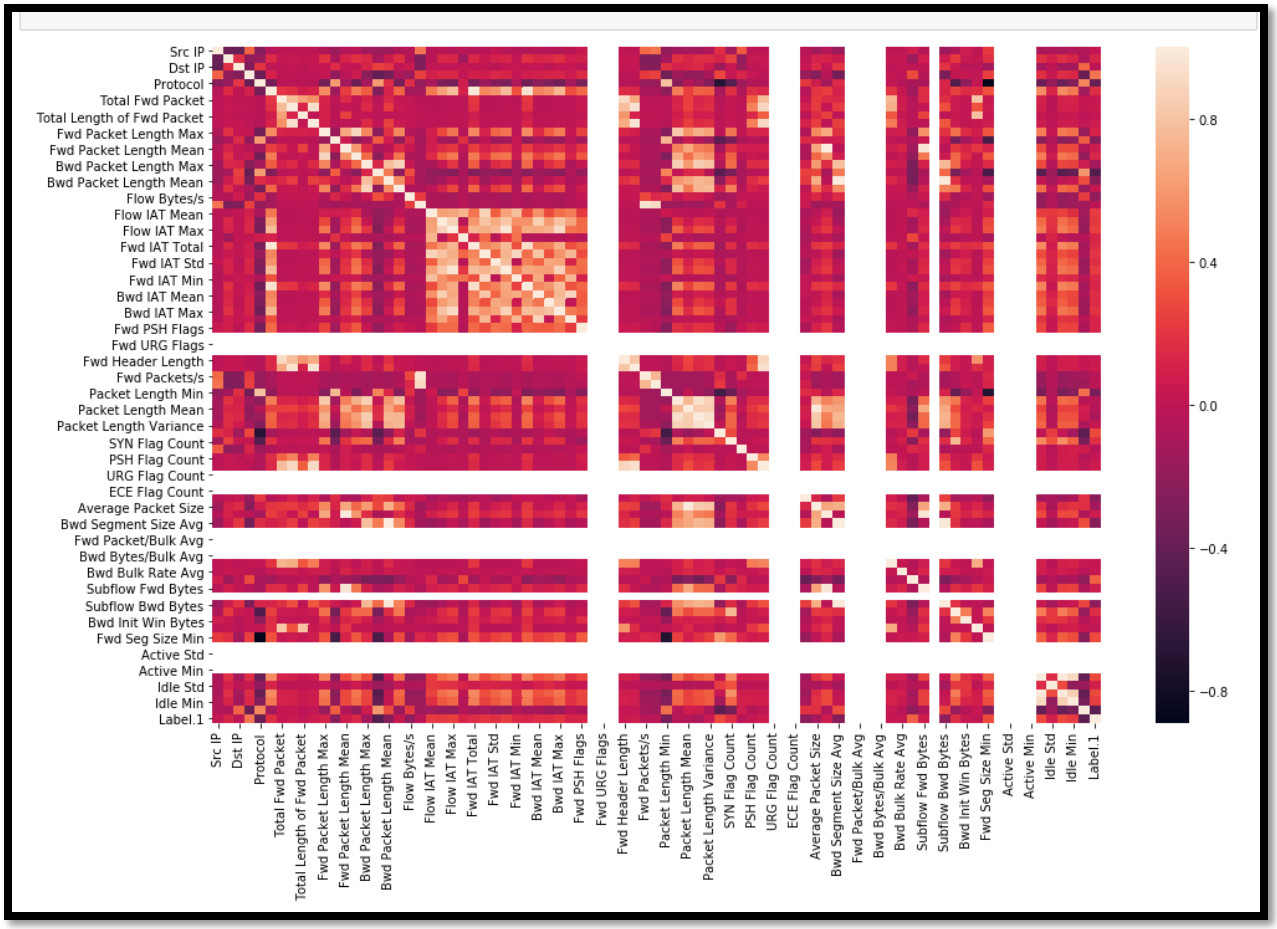
Feature Extraction: Using Correlation(>0.9)

In [50]:
```
#Getting a correlation matrix between the features and target variable
corr1 = balance_df1.corr()
corr1.head()
```

Out[50]:

	Src IP	Src Port	Dst IP	Dst Port	Protocol	Flow Duration	Total Fwd Packet	Total Bwd packets	Total Length of Fwd Packet	Total Length of Bwd Packet	...	Active Mean	Active Std	Active Max	Active Min	Idle M
Src IP	1.000000	-0.367103	-0.397431	0.378296	-0.244134	-0.050456	-0.010951	-0.012396	-0.005863	-0.013162	...	NaN	NaN	NaN	NaN	-0.088
Src Port	-0.367103	1.000000	0.174305	-0.194929	-0.168545	0.150580	0.032549	0.034625	0.014100	0.030962	...	NaN	NaN	NaN	NaN	0.171
Dst IP	-0.397431	0.174305	1.000000	-0.274543	0.197162	-0.000658	-0.004054	0.008211	-0.008902	0.019112	...	NaN	NaN	NaN	NaN	-0.123
Dst Port	0.378296	-0.194929	-0.274543	1.000000	-0.372803	0.103496	-0.003793	-0.002632	-0.011940	-0.011663	...	NaN	NaN	NaN	NaN	0.167
Protocol	-0.244134	-0.168545	0.197162	-0.372803	1.000000	-0.323833	-0.020890	-0.030813	-0.024063	-0.045865	...	NaN	NaN	NaN	NaN	-0.365

5 rows × 83 columns



➢ Model Training and Testing Phase:

```
  return f(**kwargs)
```

In [57]:
```
# Model Accuracy, how often is the classifier correct?
print("Accuracy:",metrics.accuracy_score(y_test, y_pred))
```

```
Accuracy: 1.0
```

In [58]:
```
#NOW I NEED TO DO HYPER PARAMATER TUNING OF ADABOOST
# example of grid searching key hyperparameters for adaboost on a classification dataset
from sklearn.datasets import make_classification
from sklearn.model_selection import RepeatedStratifiedKFold
from sklearn.model_selection import GridSearchCV
from sklearn.ensemble import AdaBoostClassifier
# define dataset
X, y = make_classification(n_samples=1000, n_features=20, n_informative=15, n_redundant=5, random_state=6)
# define the model with default hyperparameters
model = AdaBoostClassifier()
# define the grid of values to search
grid = dict()
grid['n_estimators'] = [10, 50, 100]
grid['learning_rate'] = [ 0.01, 0.1, 1.0]
# define the evaluation procedure
cv = RepeatedStratifiedKFold(n_splits=10, n_repeats=3, random_state=1)
# define the grid search procedure
grid_search = GridSearchCV(estimator=model, param_grid=grid, n_jobs=-1, cv=cv, scoring='accuracy')
# execute the grid search
grid_result = grid_search.fit(X_train, y_train)
# summarize the best score and configuration
print("Best: %f using %s" % (grid_result.best_score_, grid_result.best_params_))
# summarize all scores that were evaluated
means = grid_result.cv_results_['mean_test_score']
stds = grid_result.cv_results_['std_test_score']
params = grid_result.cv_results_['params']
for mean, stdev, param in zip(means, stds, params):
    print("%f (%f) with: %r" % (mean, stdev, param))
```

```
C:\Users\kshitij\anaconda3\lib\site-packages\sklearn\utils\validation.py:73: DataConversionWarning: A column-vector y was passe
d when a 1d array was expected. Please change the shape of y to (n_samples, ), for example using ravel().
  return f(**kwargs)
Best: 0.999990 using {'learning_rate': 1.0, 'n_estimators': 50}
0.935297 (0.003438) with: {'learning_rate': 0.01, 'n_estimators': 10}
0.935297 (0.003438) with: {'learning_rate': 0.01, 'n_estimators': 50}
0.935297 (0.003438) with: {'learning_rate': 0.01, 'n_estimators': 100}
0.935297 (0.003438) with: {'learning_rate': 0.1, 'n_estimators': 10}
0.997393 (0.001209) with: {'learning_rate': 0.1, 'n_estimators': 50}
0.999012 (0.000539) with: {'learning_rate': 0.1, 'n_estimators': 100}
0.996894 (0.001202) with: {'learning_rate': 1.0, 'n_estimators': 10}
0.999990 (0.000055) with: {'learning_rate': 1.0, 'n_estimators': 50}
0.999990 (0.000055) with: {'learning_rate': 1.0, 'n_estimators': 100}
```

```
print('precision: {}'.format(precision))
print('recall: {}'.format(recall))
print('fscore: {}'.format(fscore))
print('support: {}'.format(support))
```

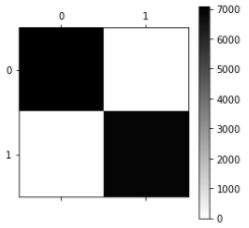
```
precision: [1. 1.]
recall: [1. 1.]
fscore: [1. 1.]
support: [7094 6935]
```

In [75]:
```
from sklearn.metrics import confusion_matrix
df_con = confusion_matrix(y_test, y_pred)
```

In [76]:
```
df_con
```

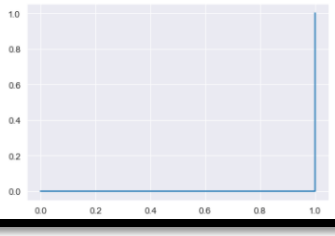
Out[76]:
```
array([[7094,    0],
       [   0, 6935]], dtype=int64)
```

In [77]:
```
import matplotlib.pyplot as plt
def plot_confusion_matrix(df_con, title='Confusion matrix', cmap=plt.cm.gray_r):
    plt.matshow(df_con, cmap=cmap) # imshow
    #plt.title(title)
    plt.colorbar()


plot_confusion_matrix(df_con)
```



In [95]:
```
from sklearn import metrics
import numpy as np
import matplotlib.pyplot as plt

fpr, tpr, thresholds = metrics.roc_curve(y_test, y_pred, pos_label=0)
print(fpr)
print(tpr)
print(thresholds)
# Print ROC curve
plt.plot(fpr,tpr)
plt.show()
```

```
[0.         0.96113125 0.96548223 0.98346628 0.9853517  0.99071791
 0.99158811 0.99347353 0.99405366 0.99492386 0.99535896 0.99666425
 0.99709935 0.99767948 0.99782451 0.99825961 0.99854967 0.99898477
 0.9995649  1.         1.         1.         1.         1.
 1.         1.         1.         1.         1.         1.
 1.         1.         1.         1.         1.         1.
 1.        ]
[0.00000000e+00 0.00000000e+00 0.00000000e+00 0.00000000e+00
 0.00000000e+00 0.00000000e+00 0.00000000e+00 0.00000000e+00
 0.00000000e+00 0.00000000e+00 0.00000000e+00 0.00000000e+00
 0.00000000e+00 0.00000000e+00 0.00000000e+00 0.00000000e+00
 0.00000000e+00 0.00000000e+00 0.00000000e+00 0.00000000e+00
 2.80347631e-04 5.60695262e-04 7.00869078e-04 1.12139052e-03
 1.26156434e-03 1.82225960e-03 2.38295486e-03 2.52312868e-03
 4.20521447e-03 4.90608354e-03 6.72834315e-03 9.39164564e-03
 1.56994673e-02 1.59798150e-02 3.65853659e-02 3.67255397e-02
 1.00000000e+00]
[2.     1.     0.9995 0.98   0.9795 0.96   0.9595 0.94   0.9395 0.92
 0.9195 0.9    0.8995 0.88   0.86   0.82   0.8    0.7    0.66   0.58
 0.3    0.28   0.2795 0.26   0.24   0.2    0.18   0.16   0.12   0.1
 0.08   0.06   0.04   0.0395 0.02   0.0195 0.    ]
```




```
328580          0.000013          0.000008          1.193857e-06
448639          0.000005          0.000000          0.000000e+00
```

In [48]:
```
#FIRST I AM RUNNING WITHOUT HYPERPARAMETERE TUNING|
```

In [49]:
```
# Load libraries
from sklearn.ensemble import AdaBoostClassifier
from sklearn import datasets
# Import train_test_split function
from sklearn.model_selection import train_test_split
#Import scikit-learn metrics module for accuracy calculation
from sklearn import metrics
```

In [50]:
```
# Create adaboost classifer object
abc = AdaBoostClassifier(n_estimators=50,
                         learning_rate=1)
# Train Adaboost Classifer
model = abc.fit(X_train, y_train)

#Predict the response for test dataset
y_pred = model.predict(X_test)
```

```
C:\Users\kshitij\anaconda3\lib\site-packages\sklearn\utils\validation.py:72: DataConversionWarning: A column-vector y was passe
d when a 1d array was expected. Please change the shape of y to (n_samples, ), for example using ravel().
  return f(**kwargs)
```

In [51]:
```
# Model Accuracy, how often is the classifier correct?
print("Accuracy:",metrics.accuracy_score(y_test, y_pred))
```

```
Accuracy: 0.2235316895250943
```

In [56]:
```
from sklearn.metrics import precision_recall_fscore_support as score


precision, recall, fscore, support = score(y_test, y_pred)

print('precision: {}'.format(precision))
print('recall: {}'.format(recall))
print('fscore: {}'.format(fscore))
print('support: {}'.format(support))
```

```
precision: [0.         0.11864407 0.19974005 0.         0.5538099  0.00400534
 0.72945033 0.52994543 0.         0.         0.16135527]
recall: [0.00000000e+00 4.85268631e-04 6.82868471e-01 0.00000000e+00
 4.81886535e-02 2.04123290e-04 1.97906690e-01 5.37684269e-01
 0.00000000e+00 0.00000000e+00 1.00000000e+00]
fscore: [0.00000000e+00 9.66583817e-04 3.09075157e-01 0.00000000e+00
 8.86625165e-02 3.88450084e-04 3.11343091e-01 5.33786801e-01
 0.00000000e+00 0.00000000e+00 2.77874095e-01]
support: [14533 14425 14628 14558 14630 14697 14618 14449 14538 14615 14425]
```