INFORMATION SECURITY ANALYSIS AND AUDIT

CSE3501

NAME: KSHITIJ DHYANI

REGISTRATION NUMBER: 18BIT0131

FACULTY: Prof. SUMAIYA THASEEN

LAB: L31+L32

SLOT: G1

TOPIC: INTRUSION DETECTION USING MACHINE LEARNING

GITHUB REPOSITORY

https://github.com/wimpywarlord/darknet2020ML

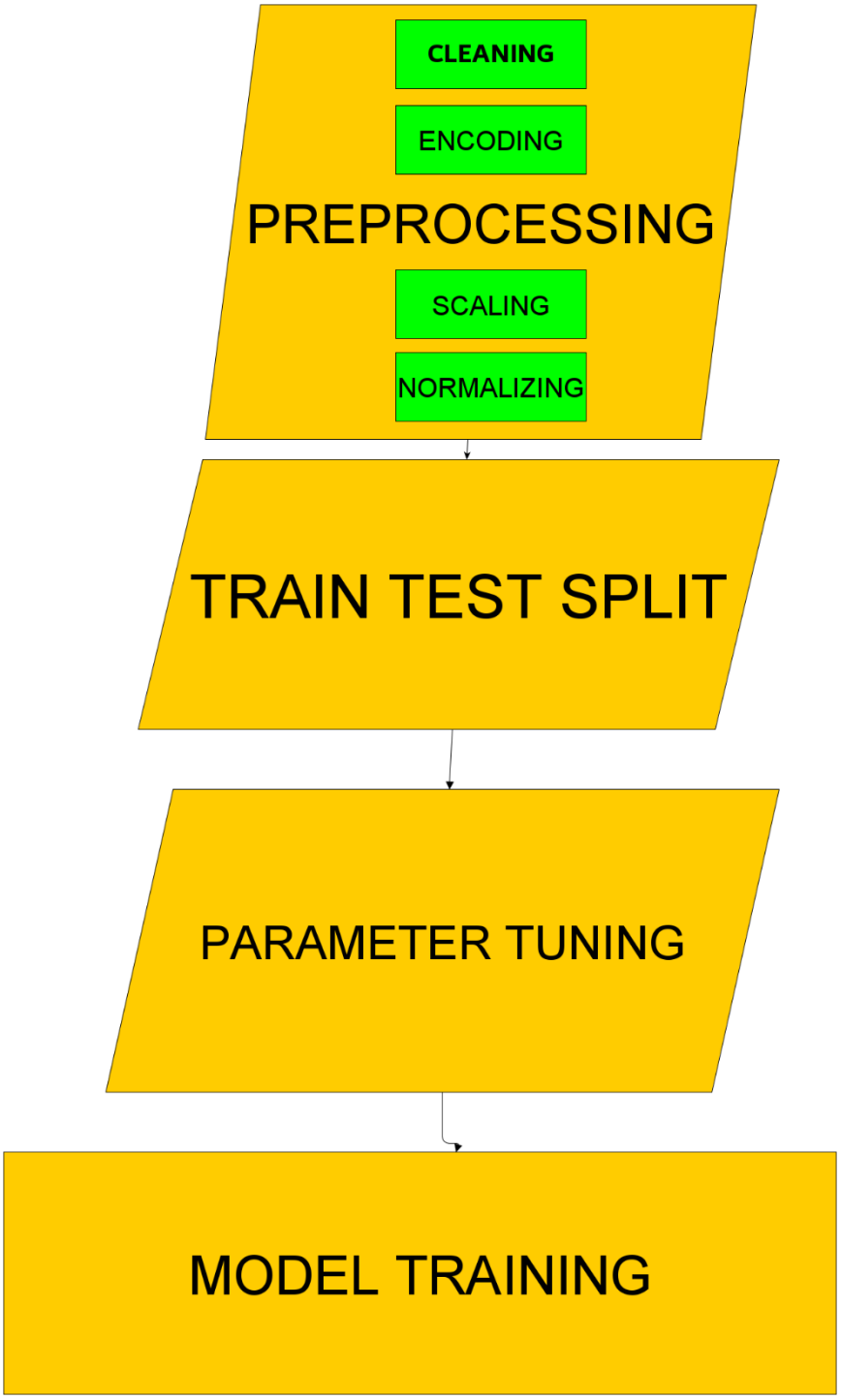
THE ABOVE LINK HAS ALL THE CODE AND ALSO THE DATASET.

MY DATA SET WAS DARKNET 2020.

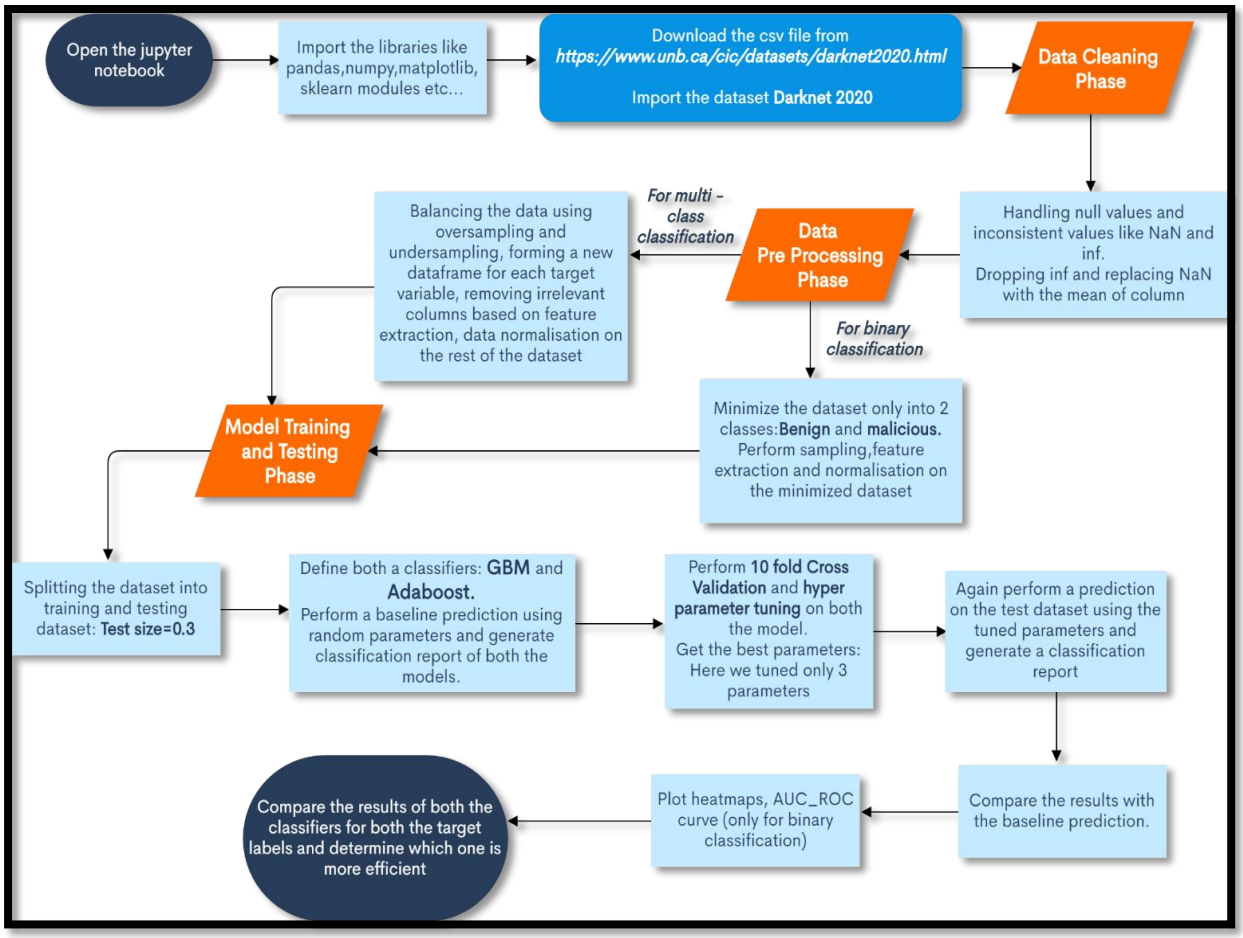
LINK TO THE DATA SET

https://www.unb.ca/cic/datasets/darknet2020.html

THE FLOW OF THE PROJECT



Design:



Description:

➢ About the Dataset:

In **CICDarknet2020** dataset, a two-layered approach is used to generate benign and darknet traffic at the first layer. The darknet traffic constitutes Audio-Stream, Browsing, Chat, Email, P2P, Transfer, Video-Stream and VOIP which is generated at the second layer. To generate the representative dataset, previously generated datasets,
namely, ISCXTor2016 and ISCXVPN2016, have been amalgamated and respective VPN and Tor traffic are combined in corresponding Darknet categories.

No. of rows: 1.4 Lacs approx..

No. of columns: 85

- First we will be importing all the python libraries and sklearn modules.

➢ Importing the dataset:

The dataset DARKNET 2020 can be downloaded from
https://www.unb.ca/cic/datasets/darknet2020.html

➢ Data Cleaning Phase:

In this phase:

- We will be handling the missing values and replacing them with the mean value of the respective column.
- Also the null and NaN values will also be replaced by the mean of the column.
- The columns also contain infinite values. To deal with such values, first we will find the row index of such values, then convert them into NaN and delete the entire row.

➢ Data Pre-Processing Phase:

For binary classification:

- The dataset is minimized by only including rows that have either benign or malicious classes i.e. Tor and Non-tor classes.
- This dataset is then balanced, that is, the number of rows of each class is made approximately equal by either over sampling or under sampling.
- Target variable is then separated from the dataframe and a new dataframe is created for the target variable. This target variable is then encode using Label Encoder to convert into integer from object.
- The dataset is then normalized using min-max scalar.
- The number of columns is large and we will not be needing all of them for classification. So we will be performing feature extraction using ranking technique to extract only the relevant columns and proceed further with only those columns.
- There are few columns like timestamp and flow-id that can be removed directly. Also columns like Src IP and Dst IP can be modified to be used further instead of deletion.

For multi-class classification:

The complete dataset is used in this case. Rest of the procedure remains same as the binary classification.

- **Model Training and Testing Phase:**

In this phase:

- First the dataset is split into training and testing dataset.
- Then we will be defining two classifiers: **Gradient Boosting Machine** and **AdaBoost**.
- Initially we will be passing random parameters and perform a baseline prediction. Classification Report is generated and accuracy is obtained. Confusion matrix is also obtained.
- Then we will be performing 10-Fold Cross Validation. After this we will be performing Hyperparameter tuning on both the models and tune three parameters each.

The three best parameters will be obtained.

- We will again perform prediction on the test dataset using a new model and passing the three best parameters. Classification report is generated again. Accuracy is obtained.
- The evaluation metrics of this new tuned model is then compared with the baseline model.
- Graphs like roc_auc curves for binary classification and heatmaps are plotted wherever required.
- Also comparison is done between both the classifiers to get the more efficient one.

DISCRIPTION

The project was to perform intrusion detection using machine learning, I was allotted the adaboost decision tree algorithm.

I was given the darknet2020 dataset, which basically a very new dataset. It categorizes the data into TOR, NON-TOR , VPN and NON VPN users. TOR is browser which helps us acess the dark net. Dark Net is a untraceable internet, where all the malicious and illegal activities take place.

I started off by cleaning the data since some of the values were outliers and some were empty or corrupt. This was simply done by removing the nan values and empty rows and columns.

```
In [10]: df.dropna() #dropping null values
```

Out[10]:

	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	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	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	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	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	1.4
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
141525	10.8.8.246-224.0.0.252-55219-5355-17	10.8.8.246	55219	224.0.0.252	5355	17	22/05/2015 01:55:03 PM	411806	2	0	...	0	0	0	0	0.0
141526	10.8.8.246-224.0.0.252-64207-5355-17	10.8.8.246	64207	224.0.0.252	5355	17	22/05/2015 02:09:05 PM	411574	2	0	...	0	0	0	0	0.0
141527	10.8.8.246-224.0.0.252-61115-5355-17	10.8.8.246	61115	224.0.0.252	5355	17	22/05/2015 02:19:31 PM	422299	2	0	...	0	0	0	0	0.0
141528	10.8.8.246-224.0.0.252-64790-5355-17	10.8.8.246	64790	224.0.0.252	5355	17	22/05/2015 02:29:55 PM	411855	2	0	...	0	0	0	0	0.0
	80.239.235.110-						22/05/2015									

Then we started off by making a correlation matrix and analysing it. This helped us determine which fields were important for the output.

```
dtypes: float64(24), int64(55), object(6)
memory usage: 91.8+ MB
```

```
In [9]: #Correlation matrix to know the dependency of columns on each other
corr = df.corr()
corr.head()
```

Out[9]:

	Src Port	Dst Port	Protocol	Flow Duration	Total Fwd Packet	Total Bwd packets	Total Length of Fwd Packet	Total Length of Bwd Packet	Fwd Packet Length Max	Fwd Packet Length Min	...	Fwd Act Data Pkts	Fwd Seg Size Min	Active Mean	Active Std	A
Src Port	1.000000	-0.246451	-0.097384	0.065328	-0.036259	-0.014248	-0.019712	-0.008271	0.076486	-0.090022	...	-0.028619	0.138354	NaN	NaN	
Dst Port	-0.246451	1.000000	-0.321199	0.039227	0.022094	0.014775	0.004451	0.010865	0.004448	-0.178715	...	0.014722	0.246275	NaN	NaN	
Protocol	-0.097384	-0.321199	1.000000	-0.266954	-0.034735	-0.026164	-0.023039	-0.020874	-0.195123	0.564044	...	-0.023370	-0.872467	NaN	NaN	
Flow Duration	0.065328	0.039227	-0.266954	1.000000	0.142110	0.100288	0.072529	0.057008	0.340744	-0.068930	...	0.145455	0.240411	NaN	NaN	
Total Fwd Packet	-0.036259	0.022094	-0.034735	0.142110	1.000000	0.744834	0.457391	0.635688	0.125575	-0.020982	...	0.698507	0.029652	NaN	NaN	

5 rows × 79 columns

Next we went onto splice the IP address fields, since it was fitting in any of the datatypes.

In [ ]:

In [14]:
```
#splitting the Src IP into octets,getting first two ocets
newIP = []
for value in df['Src IP']:
    IP = value.split(".")
    octet1= IP[0]
    octet2= IP[1]
#     print(octet2)
    newIP.append(float(octet1 + '.' + octet2))
```

In [15]:
```
df1 = pd.DataFrame(newIP)  #a new dataframe with the above obtained series
df1.head()
```

Out[15]:

	0
0	10.152
1	10.152
2	10.152
3	10.152
4	10.152

In [16]:
```
df['Src IP'] = df1  #replacing column Src IP with df1
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	
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.(

Then we encoded the Label field using, ONE HAUT ENCODING. This converted the label of non-tor,tor,vpn,non-vpn to 1s and 0s.

In [20]:
```
# 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 [21]:
```
df.head()
```

Out[21]:

	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	Idl
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

5 rows × 85 columns

Since the dataset had values of infinite scale which is scale of 10^15. We had to divide these columns by 10^15.

In the next 4 steps, i will be performing some operations on the below columns to convert it from exponential values to normal float values.

```
In [23]: df5['Idle Mean']=df5['Idle Mean']/1e15
In [24]: df5['Idle Max']=df5['Idle Max']/1e15
In [25]: df5['Idle Min']=df5['Idle Min']/1e15
In [26]: df5['Idle Std']=df5['Idle Std']/1e7
In [27]: df5.head(5)
```

Out[27]:

	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 Mean	Idle Std	Idle Max	Idle
0	10.152	57158	216.580	443	6	229	1	1	0	0	...	0	0	0	0	0.000000	0.000000	0.000000	0.00
1	10.152	57159	216.580	443	6	407	1	1	0	0	...	0	0	0	0	0.000000	0.000000	0.000000	0.00
2	10.152	57160	216.580	443	6	431	1	1	0	0	...	0	0	0	0	0.000000	0.000000	0.000000	0.00
3	10.152	49134	74.125	443	6	359	1	1	0	0	...	0	0	0	0	0.000000	0.000000	0.000000	0.00
4	10.152	34697	173.194	19305	6	10778451	591	400	64530	6659	...	0	0	0	0	1.437765	0.311772	1.437765	1.43

5 rows × 83 columns

To make the machine learning algorithm not biased, we samples the data using under sampling and oversampling technique's.

```
22919

In [36]: index_non_tor = df5[df5['Label']==0].index
         index_non_vpn = df5[df5['Label']==1].index
         index_tor = df5[df5['Label']==2].index
         index_vpn = df5[df5['Label']==3].index

In [37]: #4. Randomly sample the majority indices with respect to the number of minority classes
         random_indices = np.random.choice(index_non_tor,non_vpn,replace='False')

In [38]: #5. Concat the minority indices with the indices from step 4
         under_sample_indices = np.concatenate([index_non_vpn,random_indices])

In [39]: #Get the balanced dataframe - This is the final undersampled data
         under_sample_df = df5.iloc[under_sample_indices]
         under_sample_df.head()
```

Out[39]:

	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 Mean	Idle Std
93403	131.202	64717.0	131.202	13000.0	6.0	81.0	2.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.000000	0.000000e+00
93404	131.202	42530.0	178.237	443.0	6.0	119829241.0	5.0	3.0	24.0	0.0	...	0.0	0.0	0.0	0.0	1.225994	5.406124e+07
93405	131.202	42534.0	178.237	443.0	6.0	119828205.0	5.0	3.0	24.0	0.0	...	0.0	0.0	0.0	0.0	1.225994	5.406124e+07
93406	131.202	17208.0	77.720	11113.0	17.0	138272.0	2.0	2.0	126.0	85.0	...	0.0	0.0	0.0	0.0	1.430326	9.762882e-03
93407	8.600	0.0	8.000	0.0	0.0	5103.0	2.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.000000	0.000000e+00

5 rows × 83 columns

Then I went on to scale the data using minmax Scaling technique.

```
In [51]: #MinMaxScaling

         import pandas as pd
         from sklearn import preprocessing

         x = balance_df.values #returns a numpy array
         min_max_scaler = preprocessing.MinMaxScaler()
         x_scaled = min_max_scaler.fit_transform(x)
         df_final = pd.DataFrame(x_scaled,columns = balance_df.columns)
         df_final.head()
```

Out[51]:

	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	...	Fwd Seg Size Min	Active Mean	Active Std	Active Max	Active Min
0	0.587586	0.987533	0.512169	0.198367	0.352941	6.666667e-07	0.000004	0.000000	0.000000e+00	0.000000e+00	...	0.454545	0.0	0.0	0.0	0.0
1	0.587586	0.648976	0.697131	0.006760	0.352941	9.985770e-01	0.000017	0.000006	3.119689e-08	0.000000e+00	...	0.454545	0.0	0.0	0.0	0.0
2	0.587586	0.649037	0.697131	0.006760	0.352941	9.985684e-01	0.000017	0.000006	3.119689e-08	0.000000e+00	...	0.454545	0.0	0.0	0.0	0.0
3	0.587586	0.262581	0.301854	0.169574	1.000000	1.152258e-03	0.000004	0.000004	1.637837e-07	1.267845e-07	...	0.181818	0.0	0.0	0.0	0.0
4	0.038515	0.000000	0.027684	0.000000	0.000000	4.251667e-05	0.000004	0.000000	0.000000e+00	0.000000e+00	...	0.000000	0.0	0.0	0.0	0.0

5 rows × 82 columns

Then I did the train test split

In [52]:
```
#DOING THE TRAIN TEST SPLIT
from sklearn.model_selection import train_test_split
# split into train test sets
y=target
X_train, X_test, y_train, y_test = train_test_split(x,y, test_size=0.3)
print(X_train.shape, X_test.shape, y_train.shape, y_test.shape)
```

```
(65492, 82) (28068, 82) (65492, 1) (28068, 1)
```

Then I tuned 3 parameters.

In [72]:
```
#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: 0.940079 using {'learning_rate': 0.1, 'n_estimators': 100}
```

Finally ran the machine learning decision tree adaboost model

In [69]:
```
# 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 [70]:
```
# 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: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)
```

In [71]:
```
# Model Accuracy, how often is the classifier correct?
print("Accuracy:",metrics.accuracy_score(y_test, y_pred))
```

```
Accuracy: 0.9261792788941143
```