Assignment: Decision Tree

Bhavya Gandhi

A012

70362019019

The programming assignment for the Decision Tree model is

1. Using the Scikit-Learn Library train the Decision Tree Classifier to the attached Phishing vs Benign URL data set using all the features at once. (Dataset is originally from here: https://www.unb.ca/cic/datasets/url-2016.html)

Answer:

In [16]:
```
import pandas as pd
import numpy as np
import seaborn as sns
from matplotlib import pyplot as plt
from matplotlib import figure
from sklearn.preprocessing import LabelEncoder
from sklearn import tree
from sklearn.model_selection import train_test_split

data= pd.read_csv("phishing.csv")
data.head()
data['URL_Type_obf_Type'].unique()
```

Out[16]:
```
array(['benign', 'phishing'], dtype=object)
```

In [20]:
```
le_URL_Type_obf_Type=LabelEncoder()
data['URL_Type_obf_Type_n']=le_URL_Type_obf_Type.fit_transform(data['URL_Type_obf_Type'])
data
```

Out[20]:

	Querylength	domain_token_count	path_token_count	avgdomaintokenlen	longdomaintokenlen	avgpathtokenlen	tld	charcompvowels	charcompace	ldl_
0	0	2	12	5.500000	8	4.083334	2	15	7	
1	0	3	12	5.000000	10	3.583333	3	12	8	
2	2	2	11	4.000000	5	4.750000	2	16	11	
3	0	2	7	4.500000	7	5.714286	2	15	10	
4	19	2	10	6.000000	9	2.250000	2	9	5	
...	...	...	...	...	...	...	...	...	...	
15362	0	2	3	8.000000	13	3.333333	2	3	2	
15363	0	3	0	9.000000	16	NaN	3	0	0	
15364	0	3	2	6.666666	10	3.000000	3	3	2	
15365	0	2	3	8.000000	13	3.333333	2	4	2	

In [21]:
```
X=data.drop(['URL_Type_obf_Type'], axis='columns')
X
```

Out[21]:

	Querylength	domain_token_count	path_token_count	avgdomaintokenlen	longdomaintokenlen	avgpathtokenlen	tld	charcompvowels	charcompace	ldl_
0	0	2	12	5.500000	8	4.083334	2	15	7	
1	0	3	12	5.000000	10	3.583333	3	12	8	
2	2	2	11	4.000000	5	4.750000	2	16	11	
3	0	2	7	4.500000	7	5.714286	2	15	10	
4	19	2	10	6.000000	9	2.250000	2	9	5	
...	...	...	...	...	...	...	...	...	...	
15362	0	2	3	8.000000	13	3.333333	2	3	2	
15363	0	3	0	9.000000	16	NaN	3	0	0	
15364	0	3	2	6.666666	10	3.000000	3	3	2	
15365	0	2	3	8.000000	13	3.333333	2	4	2	
15366	0	2	3	9.000000	15	3.000000	2	2	1	

15367 rows × 80 columns

In [23]:
```
target=X['URL_Type_obf_Type_n']
Y=X.drop('URL_Type_obf_Type_n',axis='columns')
Y
```

Out[23]:

	Querylength	domain_token_count	path_token_count	avgdomaintokenlen	longdomaintokenlen	avgpathtokenlen	tld	charcompvowels	charcompace	ldl_
0	0	2	12	5.500000	8	4.083334	2	15	7	
1	0	3	12	5.000000	10	3.583333	3	12	8	
2	2	2	11	4.000000	5	4.750000	2	16	11	

In [23]:
```
target=X['URL_Type_obf_Type_n']
Y=X.drop('URL_Type_obf_Type_n',axis='columns')
Y
```

Out[23]:

	Querylength	domain_token_count	path_token_count	avgdomaintokenlen	longdomaintokenlen	avgpathtokenlen	tld	charcompvowels	charcompace	ldl_
0	0	2	12	5.500000	8	4.083334	2	15	7	
1	0	3	12	5.000000	10	3.583333	3	12	8	
2	2	2	11	4.000000	5	4.750000	2	16	11	
3	0	2	7	4.500000	7	5.714286	2	15	10	
4	19	2	10	6.000000	9	2.250000	2	9	5	
...	...	...	...	...	...	...	...	...	...	
15362	0	2	3	8.000000	13	3.333333	2	3	2	
15363	0	3	0	9.000000	16	NaN	3	0	0	
15364	0	3	2	6.666666	10	3.000000	3	3	2	
15365	0	2	3	8.000000	13	3.333333	2	4	2	
15366	0	2	3	9.000000	15	3.000000	2	2	1	

15367 rows × 79 columns

In [24]:
```
Y.replace([np.inf, - np.inf], np.nan, inplace = True)
Y.fillna(Y.mean(),inplace=True)
Y.isnull().sum()
```

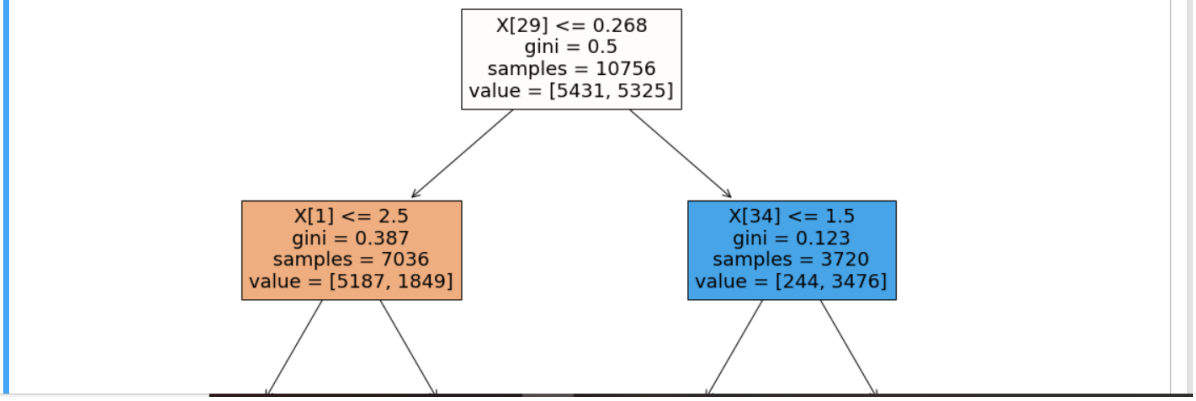
Out[24]:
```
Querylength              0
domain_token_count       0
path_token_count         0
avgdomaintokenlen        0
```

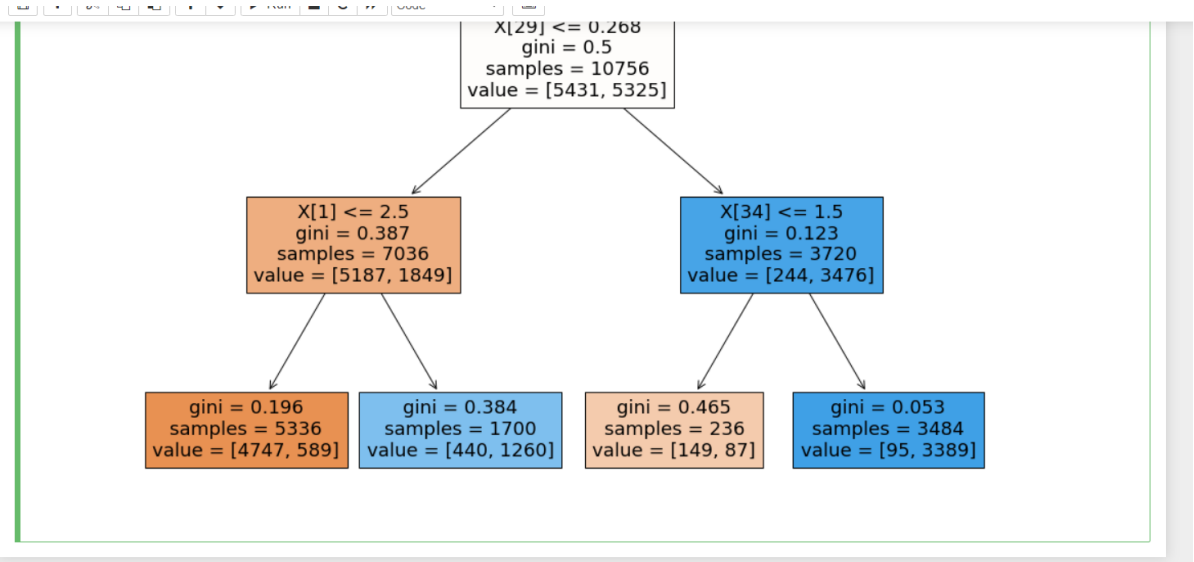
2. Run the model for trees of dept 1,2,3,4,5 and 6 for the Gini Impurity and Entropy measures for each tree dept. Compare the results of these 12 cases and discuss your results.

```
In [77]: X_train, X_test, y_train, y_test = train_test_split(Y, target, test_size=0.2, random_state=10)
         for i in range(1,7):
             clf = DecisionTreeClassifier(criterion="entropy", max_depth=(i))
             clf = clf.fit(X_train,y_train)
             y_pred = clf.predict(X_test)
             print("Accuracy",i,":",metrics.accuracy_score(y_test, y_pred))

         Accuracy 1 : 0.8018867924528302
         Accuracy 2 : 0.8988288874430709
         Accuracy 3 : 0.9007807417046194
         Accuracy 4 : 0.9372153545868576
         Accuracy 5 : 0.9486011711125569
         Accuracy 6 : 0.9544567338972023

In [76]: X_train, X_test, y_train, y_test = train_test_split(Y, target, test_size=0.2, random_state=10)
         for i in range(1,7):
             clf = DecisionTreeClassifier(criterion="gini", max_depth=(i))
             clf = clf.fit(X_train,y_train)
             y_pred = clf.predict(X_test)
             print("Accuracy",i,":",metrics.accuracy_score(y_test, y_pred))

         Accuracy 1 : 0.8148991541964866
         Accuracy 2 : 0.9079375406636304
         Accuracy 3 : 0.9362394274560832
         Accuracy 4 : 0.9525048796356539
         Accuracy 5 : 0.9554326610279765
         Accuracy 6 : 0.9583604424202993
```

3. Take the best performing tree of dept 2 from above. Visualize the tree and discuss your observations. (For visualizing decision trees see:

```
In [75]: clf = DecisionTreeClassifier(criterion="gini", max_depth=2)
         clf = clf.fit(X_train,y_train)
         plt.figure(figsize=(15,10))
         tree.plot_tree(clf,filled=True)

Out[75]: [Text(418.5, 453.0, 'X[29] <= 0.268\ngini = 0.5\nsamples = 10756\nvalue = [5431, 5325]'),
          Text(209.25, 271.8, 'X[1] <= 2.5\ngini = 0.387\nsamples = 7036\nvalue = [5187, 1849]'),
          Text(104.625, 90.59999999999997, 'gini = 0.196\nsamples = 5336\nvalue = [4747, 589]'),
          Text(313.875, 90.59999999999997, 'gini = 0.384\nsamples = 1700\nvalue = [440, 1260]'),
          Text(627.75, 271.8, 'X[34] <= 1.5\ngini = 0.123\nsamples = 3720\nvalue = [244, 3476]'),
          Text(523.125, 90.59999999999997, 'gini = 0.465\nsamples = 236\nvalue = [149, 87]'),
          Text(732.375, 90.59999999999997, 'gini = 0.053\nsamples = 3484\nvalue = [95, 3389]')]
```





Conclusion: We can observe that for a certain depth value the accuracy keeps on increasing but after reaching to maximum point, the accuracy starts to decrease even with increasing depth.