PART 1

In [64]:
```
import numpy as np
import pandas as pd
from sklearn.ensemble import AdaBoostClassifier
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score
from sklearn import tree
import matplotlib.pyplot as plt
```

In [65]:
```
df = pd.read_csv('DataSetForPhishingVSBenignUrl.csv')
```

In [66]:
```
df
```

Out[66]:

	Querylength	domain_token_count	path_token_count	avgdomaintokenlen	longdomaintokenlen
0	0	4	5	5.500000	14
1	0	4	5	5.500000	14
2	0	4	5	5.500000	14
3	0	4	12	5.500000	14
4	0	4	6	5.500000	14
...	...	...	...	...	...
36702	29	4	14	5.750000	12
36703	0	4	13	3.750000	8
36704	58	3	27	6.666666	16
36705	35	3	13	4.333334	9
36706	40	3	25	6.666666	16

36707 rows × 80 columns

In [67]:
```
#Formatting Data Frame to contain only Phishing and Benign
df = df.loc[(df['URL_Type_obf_Type'] == 'phishing') | (df['URL_Type_obf_Typ
df
```

Out[67]:

	Querylength	domain_token_count	path_token_count	avgdomaintokenlen	longdomaintokenlen
7930	0	2	12	5.500000	8
7931	0	3	12	5.000000	10
7932	2	2	11	4.000000	5
7933	0	2	7	4.500000	7
7934	19	2	10	6.000000	9
...	...	...	...	...	...
30004	0	2	3	8.000000	13
30005	0	3	0	9.000000	16
30006	0	3	2	6.666666	10
30007	0	2	3	8.000000	13
30008	0	2	3	9.000000	15

15367 rows × 80 columns

In [68]:
```
df = df.dropna()
```

In [69]:
```
df
```

Out[69]:

	Querylength	domain_token_count	path_token_count	avgdomaintokenlen	longdomaintokenlen
7930	0	2	12	5.500000	8
7931	0	3	12	5.000000	10
7934	19	2	10	6.000000	9
7935	0	2	10	5.500000	9
7938	0	2	9	2.500000	3
...	...	...	...	...	...
29978	0	2	5	5.500000	8
29986	0	2	5	3.500000	5
29992	0	2	4	7.000000	12
29996	0	3	5	4.666666	10
29999	0	2	4	8.000000	13

6723 rows × 80 columns

In [70]:
```
X= df.iloc[0:,0:79]
```

In [71]:
```
X
```

Out[71]:

	Querylength	domain_token_count	path_token_count	avgdomaintokenlen	longdomaintokenlen
7930	0	2	12	5.500000	8
7931	0	3	12	5.000000	10
7934	19	2	10	6.000000	9
7935	0	2	10	5.500000	9
7938	0	2	9	2.500000	3
...	...	...	...	...	...
29978	0	2	5	5.500000	8
29986	0	2	5	3.500000	5
29992	0	2	4	7.000000	12
29996	0	3	5	4.666666	10
29999	0	2	4	8.000000	13

In [72]:
```
y = df.iloc[0:, 79]
```

In [73]:
```
y
```

Out[73]:
```
7930       benign
7931       benign
7934       benign
7935       benign
7938       benign
           ...
29978    phishing
29986    phishing
29992    phishing
29996    phishing
29999    phishing
Name: URL_Type_obf_Type, Length: 6723, dtype: object
```

In [74]:
```
X_train, X_test, y_train, y_test = train_test_split( X, y, test_size = 0.3,
```

In [75]:
```
X_train.shape
```

Out[75]:
```
(4706, 79)
```

In [76]:
```
y_train.shape
```

Out[76]:
```
(4706,)
```

In [77]:
```
# Decision Tree classifier with criteria - Gini Index
tree_gini = DecisionTreeClassifier(criterion = "gini", random_state = 50)
tree_gini.fit(X_train, y_train)
```

Out[77]:
```
DecisionTreeClassifier(random_state=50)
```

In [52]:
```
tree_gini.score(X_train,y_train)
```

Out[52]:
```
1.0
```

In [78]:
```
tree_gini.score(X_test,y_test)
```

Out[78]:
```
0.9514129895884977
```

In [79]:
```
# Create a decision tree object
tree_gini = DecisionTreeClassifier()
# Fit on training data
tree_gini.fit(X_train, y_train)
```

Out[79]:
```
DecisionTreeClassifier()
```

In [80]:
```
#Create adaboost classifer object
abc = AdaBoostClassifier(base_estimator=tree_gini)

# Train Adaboost Classifer
model = abc.fit(X_train, y_train)

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

```
/Users/soniarahman/opt/anaconda3/lib/python3.8/site-packages/sklearn/bas
e.py:441: UserWarning: X does not have valid feature names, but AdaBoostC
lassifier was fitted with feature names
  warnings.warn(
```

In [81]:
```
print("Accuracy:",accuracy_score(y_test, y_pred))
```

```
Accuracy: 0.9489340604858701
```

In [82]:
```
from sklearn.model_selection import GridSearchCV
```

In [83]:
```
tree_param = {'criterion':['gini','entropy'],
              'max_depth':[1,3,6,9,12,15,18]
             }
grid = GridSearchCV(DecisionTreeClassifier(), tree_param, cv=5)
grid.fit(X_train, y_train)
```

Out[83]:
```
GridSearchCV(cv=5, estimator=DecisionTreeClassifier(),
             param_grid={'criterion': ['gini', 'entropy'],
                         'max_depth': [1, 3, 6, 9, 12, 15, 18]})
```

In [84]:
```
print(grid.best_score_)
```

```
0.9617491443127539
```

In [85]:
```
print(grid.best_params_)
```

```
{'criterion': 'gini', 'max_depth': 15}
```

In [86]:
```
print(grid.best_estimator_)
```

```
DecisionTreeClassifier(max_depth=15)
```

In [88]:
```
tree = grid.best_estimator_
tree.fit(X_train, y_train)
# Create adaboost classifer object
abc = AdaBoostClassifier(n_estimators=100,
learning_rate=1,
random_state=8,
base_estimator=tree)
# Train Adaboost Classifer
model = abc.fit(X_train, y_train)
# Predict the response for test dataset
y_pred = model.predict(X_test)
print("Accuracy:",accuracy_score(y_test, y_pred))
```

```
Accuracy: 0.9781854238968766

/Users/soniarahman/opt/anaconda3/lib/python3.8/site-packages/sklearn/bas
e.py:441: UserWarning: X does not have valid feature names, but AdaBoostC
lassifier was fitted with feature names
  warnings.warn(
```

#

Using adaboost, the decision tree obtained an accuracy score of 0.9781854238968766 for this set, which is much higher than the previous week's score of 0.969261279127417. We get a higher score when we use adaboost. In both assignments, the gini index is greater than the entropy. The adaboost classifier and the set parameters resulted in 15 as the max depth with the best parameter and estimator.