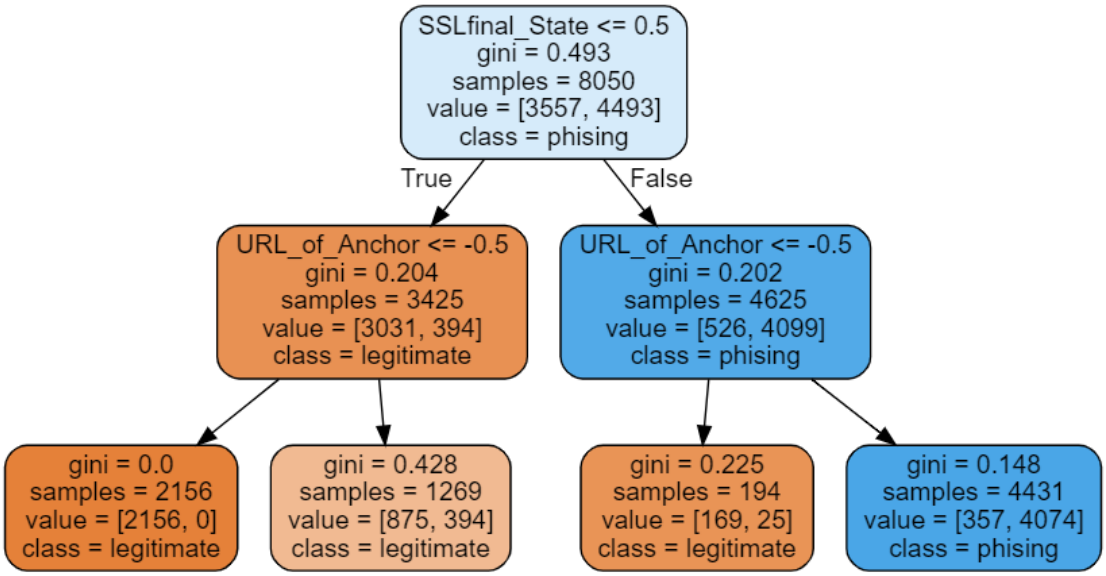
t[74]:



1.

2

- Check if use https and Issuer Is Trusted &and Age of Certificate≥ 1 Years or Using https and Issuer Is Not Trusted
    - If **YES**, check if % of URL Of Anchor <31%.
        - If **YES** then the website is LEGITIMATE
        - If **NO** then it also should be LEGITIMATE, but with lower confidence
    - If **NO**, check if % of URL Of Anchor <31%.
        - If **YES** then the website is LEGITIMATE
        - If **NO**, then website is PHISING

3.

The accuracy is 91%

4.

```
from sklearn import datasets, model_selection, svm, metrics
import numpy as np
import pandas as pd
import threading
from sklearn.model_selection import train_test_split
from sklearn import tree
import matplotlib.pyplot as plt
from sklearn import preprocessing as pp


##Prepare data##

filename=r'phishing.csv'
data_train=pd.read_csv(filename,index_col=None,na_values='?',sep = ';')

data_train=data_train.dropna()
colnames = data_train.columns.get_values()

print("\nDESCRIBE DATA:\n",data_train.describe())
data_train.shape
```

DESCRIBE DATA:

	having_IP_Address	URL_Length	Shortining_Service	having_At_Symbol
count	11055.000000	11055.000000	11055.000000	11055.000000
mean	0.313795	-0.633198	0.738761	0.700588
std	0.949534	0.766095	0.673998	0.713598
min	-1.000000	-1.000000	-1.000000	-1.000000
25%	-1.000000	-1.000000	1.000000	1.000000
50%	1.000000	-1.000000	1.000000	1.000000
75%	1.000000	-1.000000	1.000000	1.000000
max	1.000000	1.000000	1.000000	1.000000

	double_slash_redirecting	Prefix_Suffix	having_Sub_Domain
count	11055.000000	11055.000000	11055.000000
mean	0.741474	-0.734962	0.063953
std	0.671011	0.678139	0.817518
min	-1.000000	-1.000000	-1.000000
25%	1.000000	-1.000000	-1.000000
50%	1.000000	-1.000000	0.000000
75%	1.000000	-1.000000	1.000000
max	1.000000	1.000000	1.000000

	SSLfinal_State	Domain_registeration_length	Favicon	...
count	11055.000000	11055.000000	11055.000000	...
mean	0.250927	-0.336771	0.628584	...
std	0.911892	0.941629	0.777777	...
min	-1.000000	-1.000000	-1.000000	...
25%	-1.000000	-1.000000	1.000000	...
50%	1.000000	-1.000000	1.000000	...
75%	1.000000	1.000000	1.000000	...
max	1.000000	1.000000	1.000000	...

	popUpWindow	Iframe	age_of_domain	DNSRecord	web_traffic
count	11055.000000	11055.000000	11055.000000	11055.000000	11055.000000
mean	0.613388	0.816915	0.061239	0.377114	0.287291
std	0.789818	0.576784	0.998168	0.926209	0.827733
min	-1.000000	-1.000000	-1.000000	-1.000000	-1.000000
25%	1.000000	1.000000	-1.000000	-1.000000	0.000000
50%	1.000000	1.000000	1.000000	1.000000	1.000000
75%	1.000000	1.000000	1.000000	1.000000	1.000000
max	1.000000	1.000000	1.000000	1.000000	1.000000

	Page_Rank	Google_Index	Links_pointing_to_page	Statistical_report

```
count  11055.000000  11055.000000            11055.000000        11055.0000
00
mean      -0.483673      0.721574                0.344007            0.7195
84
std        0.875289      0.692369                0.569944            0.6944
37
min       -1.000000     -1.000000               -1.000000           -1.0000
00
25%       -1.000000      1.000000                0.000000            1.0000
00
50%       -1.000000      1.000000                0.000000            1.0000
00
75%        1.000000      1.000000                1.000000            1.0000
00
max        1.000000      1.000000                1.000000            1.0000
00

              Result
count  11055.000000
mean       0.113885
std        0.993539
min       -1.000000
25%       -1.000000
50%        1.000000
75%        1.000000
max        1.000000

[8 rows x 31 columns]
(11055, 31)
```

```
from sklearn.tree import DecisionTreeClassifier
###MAKE DECISION TREE###

#data_train
X_all = data_train.drop(['Result'], axis=1)
y_all = data_train['Result']

test_size=2050
train_size=8050

X_train, X_test, y_train, y_test = model_selection.train_test_split(X_a
ll, y_all, test_size=test_size, train_size=train_size)
```

Notes

On default settings on tree makes bit unbalanced classification. fine tuned min_samples_split value to 1500. min_samples_leaf value tuning didn't really do nothing special between range 2-100 but if it is increased to 200 balance of the classification gets worse.

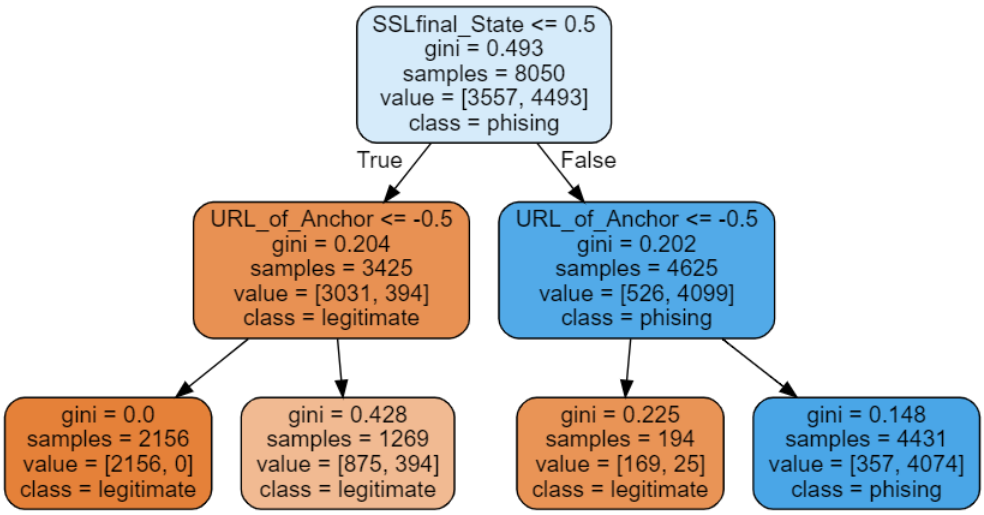
```
from sklearn.metrics import confusion_matrix
from sklearn.metrics import precision_score,confusion_matrix,f1_score,c
lassification_report

clf = DecisionTreeClassifier(criterion="gini",
                             min_samples_split=1500,
                             min_samples_leaf=50,
                             splitter='best',
                             max_depth=2)
clf.fit(X_train, y_train)
```

```
pre = clf.predict(X_test)

accuracy_score = metrics.accuracy_score(y_test, pre)

print("accuracy:", accuracy_score)
cf_matrix = confusion_matrix(y_test, pre)
print()

print(pd.crosstab(y_test, pre, rownames=['True'], colnames=['Predicted'
], margins=True),"\n")

print(classification_report(y_test,pre))
```

```
accuracy: 0.9102439024390244

Predicted   -1     1   All
True
-1         809    85   894
1           99  1057  1156
All        908  1142  2050

              precision    recall  f1-score   support

          -1       0.89      0.90      0.90       894
           1       0.93      0.91      0.92      1156

    accuracy                           0.91      2050
   macro avg       0.91      0.91      0.91      2050
weighted avg       0.91      0.91      0.91      2050
```

```
from sklearn.tree import plot_tree
import graphviz
plt.figure()
plot_tree(clf, filled=True,class_names = ['legitimate','phising'])
plt.show()

from sklearn.tree import export_graphviz
# Export as dot file
export_graphviz(clf, out_file='tree.dot',
                feature_names = colnames[:30],
                class_names = ['legitimate','phising'],
                rounded = True, proportion = False,
                precision = 2, filled = True)
```

t[74]:




```
##Edited plot
import pydotplus
import pydot
import graphviz

#Without this graphviz exetables not found
import os
os.environ['PATH'] = os.environ['PATH']+';'+os.environ['CONDA_PREFIX']+
r"\Library\bin\graphviz"

tree_data = export_graphviz(clf, feature_names=colnames[:30], out_file=
None,class_names = ['legitimate','phising'], filled=True, rounded=True)
tree_graph = pydotplus.graph_from_dot_data(tree_data)

tree_graph.set_size('"10,10!"')
tree_graph.write_png('img_treeDot.png')
#tree_graph.write_pdf("pdf_treeDot.pdf")

graphviz_graph = graphviz.Source(tree_graph.to_string())
graphviz_graph
```

```
<graphviz.files.Source at 0x1ff5f82c808>
```

Plot is on the top of the document