3_DecisionTree

August 27, 2024

[44]:
```
import pandas as pd
import matplotlib.pyplot as plt
from sklearn import tree
from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report
from sklearn.metrics import accuracy_score
from sklearn.model_selection import train_test_split

# Creating a small but useful decision tree that predicts whether a website is␣
 ↪legitimate or a phishing site

df = pd.read_csv("phishing.csv", sep=";")
df.describe()
```

[44]:

	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        11055.000000
mean      -0.483673      0.721574                0.344007            0.719584
std        0.875289      0.692369                0.569944            0.694437
min       -1.000000     -1.000000               -1.000000           -1.000000
25%       -1.000000      1.000000                0.000000            1.000000
50%       -1.000000      1.000000                0.000000            1.000000
75%        1.000000      1.000000                1.000000            1.000000
max        1.000000      1.000000                1.000000            1.000000

             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]
```

[56]:
```
# Split the data into features and target variable
x = df.drop('Result', axis=1)
y = df['Result']

# Split the data into training and testing sets
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2,␣
 ↪random_state=42)
```

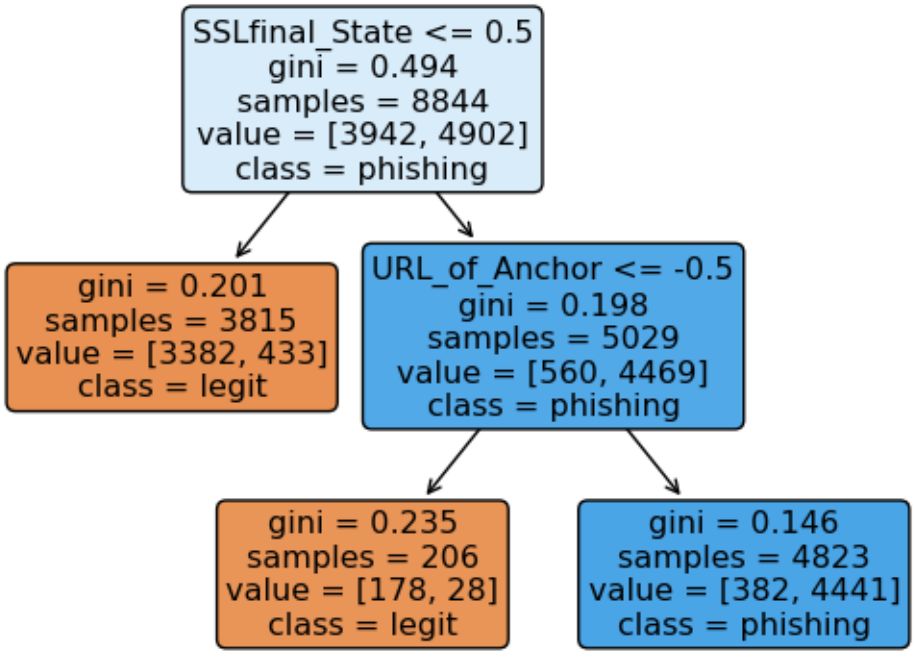
```
# Create and fit the decision tree classifier
# Make the tree small for easier understanding using current parameter values
clf = tree.DecisionTreeClassifier(max_depth=2, min_samples_split=5000)
clf.fit(x_train, y_train)
```

[56]:
```
DecisionTreeClassifier(max_depth=2, min_samples_split=5000)
```

[54]:
```
# Convert the feature names to a list of strings
feature_names = df.columns[:30].tolist()

# Create a plot of the decision tree
tree.plot_tree(clf,
               feature_names=feature_names,  # List of feature names
               class_names=['legit', 'phishing'],  # Class names for the labels
               filled=True,  # Color the nodes according to the class label
               rounded=True)  # Round the corners of the nodes

# Show the plot
plt.show()
```



[50]:
```
# Predict using the test set
Y_pred = clf.predict(x_test)
```

```python
# Create an output confusion matrix
cm = confusion_matrix(y_test, Y_pred)
print("Confusion matrix:\n", cm)

# Calculate and print accuracy
accuracy = accuracy_score(y_test, Y_pred)
print("Accuracy calculated from the test set = %.3f" % (accuracy))

# Print classification report
print(classification_report(y_test, Y_pred, target_names=['no', 'yes']))
```

```
Confusion matrix:
 [[ 865   91]
 [ 102 1153]]
Accuracy calculated from the test set = 0.913
              precision    recall  f1-score   support

          no       0.89      0.90      0.90       956
         yes       0.93      0.92      0.92      1255

    accuracy                           0.91      2211
   macro avg       0.91      0.91      0.91      2211
weighted avg       0.91      0.91      0.91      2211
```

1 Written Instructions for Reading the Plot

1. **SSL Status**:
   - If SSL is okay, the website is probably legitimate.
   - If SSL is not okay, proceed to step 2.
2. **URL Anchors**:
   - If the URL anchors match the anchors of the website, it is most likely a legitimate website.
   - If the URL anchors do not match, the website is probably used for phishing.