Phishing_Detection

June 4, 2023

1 Aditi Patra 21BIT0125 Lab5 and DA2

2 Dataset Name: Phishing Detection

3 URL Link: https://www.kaggle.com/datasets/ahmednour/website-phishing-data-set

4 ML Model used: RandomForestClassifier

5 Data Description:

1. SFH (Server Form Handler): Represents the presence of a server form handler. It can take values such as "1" for having a form handler, "0" for not having a form handler, or "-" for an unknown value.
2. popUpWidnow: Indicates the use of pop-up windows. It can take values like "1" for having pop-up windows, "0" for not having pop-up windows, or "-" for an unknown value.
3. SSLfinal_State: Represents the final state of the SSL (Secure Sockets Layer) certificate. It can have values like "1" for a valid certificate, "0" for an invalid certificate, or "-" for an unknown value.
4. Request_URL: Indicates the presence of an abnormal URL. It can take values like "1" for having an abnormal URL, "0" for not having an abnormal URL, or "-" for an unknown value.
5. URL_of_Anchor: Represents the presence of an abnormal URL of an anchor element. It can have values like "1" for having an abnormal URL of an anchor element, "0" for not having an abnormal URL, or "-" for an unknown value.
6. web_traffic: Indicates the level of web traffic. It can take values like "1" for high traffic, "0" for low traffic, or "-" for an unknown value.
7. URL_Length: Represents the length of the URL. It can have values like "1" for a long URL, "0" for a short URL, or "-" for an unknown value.
8. age_of_domain: Indicates the age of the domain. It can take values like "1" for an old domain, "0" for a new domain, or "-" for an unknown value.
9. having_IP_Address: Represents the presence of an IP address in the URL. It can have values like "1" for having an IP address, "0" for not having an IP address, or "-" for an unknown value.

10. Result: The target column indicating whether a website is classified as legitimate (0) or phishing (1).

6 Importing Libraries

[1]:
```
import pandas as pd
from sklearn.model_selection import train_test_split
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import confusion_matrix, classification_report
```

7 Reading the Dataset and giving the appropriate outputs:

[2]:
```
# Read the dataset
phis = pd.read_csv('Website Phishing.csv')
phis.head()
```

[2]:
```
   SFH  popUpWidnow  SSLfinal_State  Request_URL  URL_of_Anchor  web_traffic  \
0    1           -1               1           -1             -1            1
1   -1           -1              -1           -1             -1            0
2    1           -1               0            0             -1            0
3    1            0               1           -1             -1            0
4   -1           -1               1           -1              0            0

   URL_Length  age_of_domain  having_IP_Address  Result
0           1              1                  0       0
1           1              1                  1       1
2          -1              1                  1       0
3           1              1                  0       0
4          -1              1                  1       0
```

[3]:
```
phis.describe()
```

[3]:
```
               SFH  popUpWidnow  SSLfinal_State  Request_URL  URL_of_Anchor  \
count  1353.000000  1353.000000     1353.000000  1353.000000    1353.000000
mean      0.237990    -0.258684        0.327421    -0.223208      -0.025129
std       0.916389     0.679072        0.822193     0.799682       0.936262
min      -1.000000    -1.000000       -1.000000    -1.000000      -1.000000
25%      -1.000000    -1.000000        0.000000    -1.000000      -1.000000
50%       1.000000     0.000000        1.000000     0.000000       0.000000
75%       1.000000     0.000000        1.000000     0.000000       1.000000
max       1.000000     1.000000        1.000000     1.000000       1.000000

       web_traffic   URL_Length  age_of_domain  having_IP_Address       Result
count  1353.000000  1353.000000    1353.000000        1353.000000  1353.000000
```

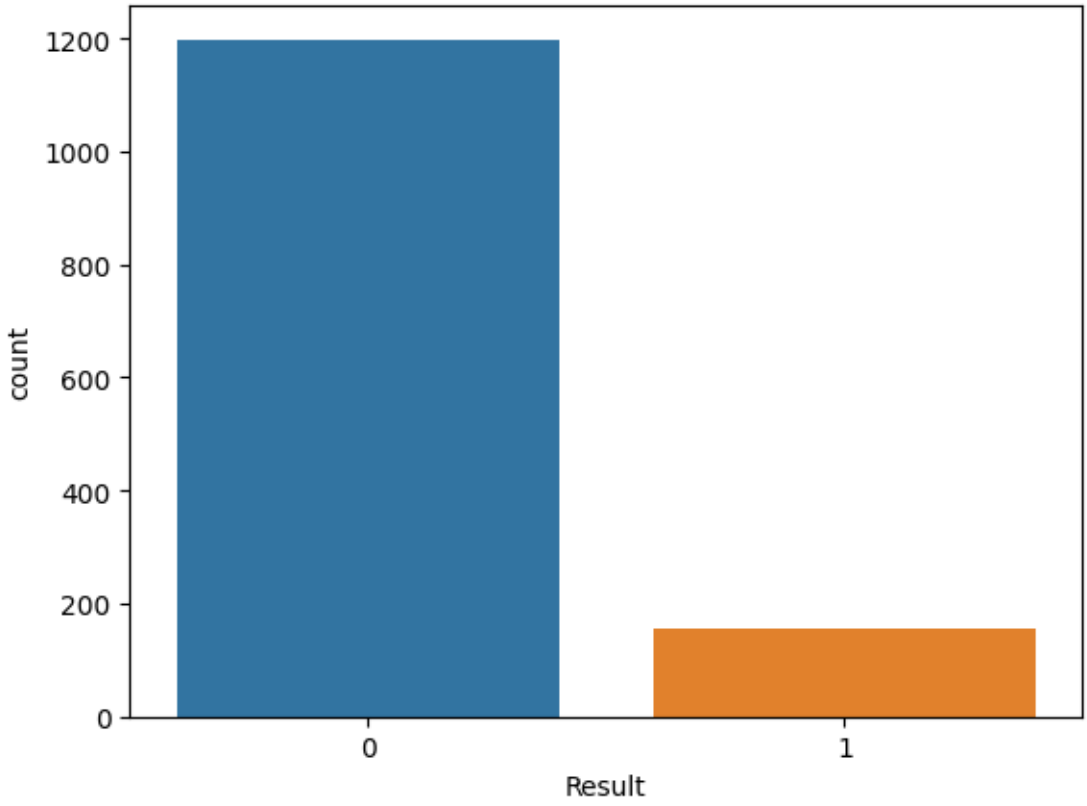
```
mean      0.000000   -0.053215    0.219512   -0.113821    0.114560
std       0.806776    0.762552    0.975970    0.954773    0.318608
min      -1.000000   -1.000000   -1.000000   -1.000000    0.000000
25%      -1.000000   -1.000000   -1.000000   -1.000000    0.000000
50%       0.000000    0.000000    1.000000   -1.000000    0.000000
75%       1.000000    1.000000    1.000000    1.000000    0.000000
max       1.000000    1.000000    1.000000    1.000000    1.000000
```

[4]:
```
phis.shape
```

[4]:
```
(1353, 10)
```

[5]:
```
sns.countplot(x=phis['Result'])
```

[5]:
```
<Axes: xlabel='Result', ylabel='count'>
```



[6]:
```
phis['Result'].value_counts()
```

[6]:
```
Result
0    1198
1     155
```

```
Name: count, dtype: int64
```

[7]:
```
# Define feature columns and target column
feature_cols = ['SFH', 'popUpWidnow', 'SSLfinal_State', 'Request_URL',␣
 ↪'URL_of_Anchor', 'web_traffic', 'URL_Length', 'age_of_domain',␣
 ↪'having_IP_Address']
target_col = ['Result']
```

[8]:
```
# Split the dataset into features (X) and target (y)
X = phis[feature_cols]
y = phis[target_col]
y = y.values.ravel()
```

[9]:
```
# Split dataset into training set and test set
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3)  # 70%␣
 ↪training and 30% test
```

[10]:
```
# Create a RandomForestClassifier
clf = RandomForestClassifier(n_estimators=100)
```

[11]:
```
# Train the model using the training sets
clf.fit(X_train, y_train)
```

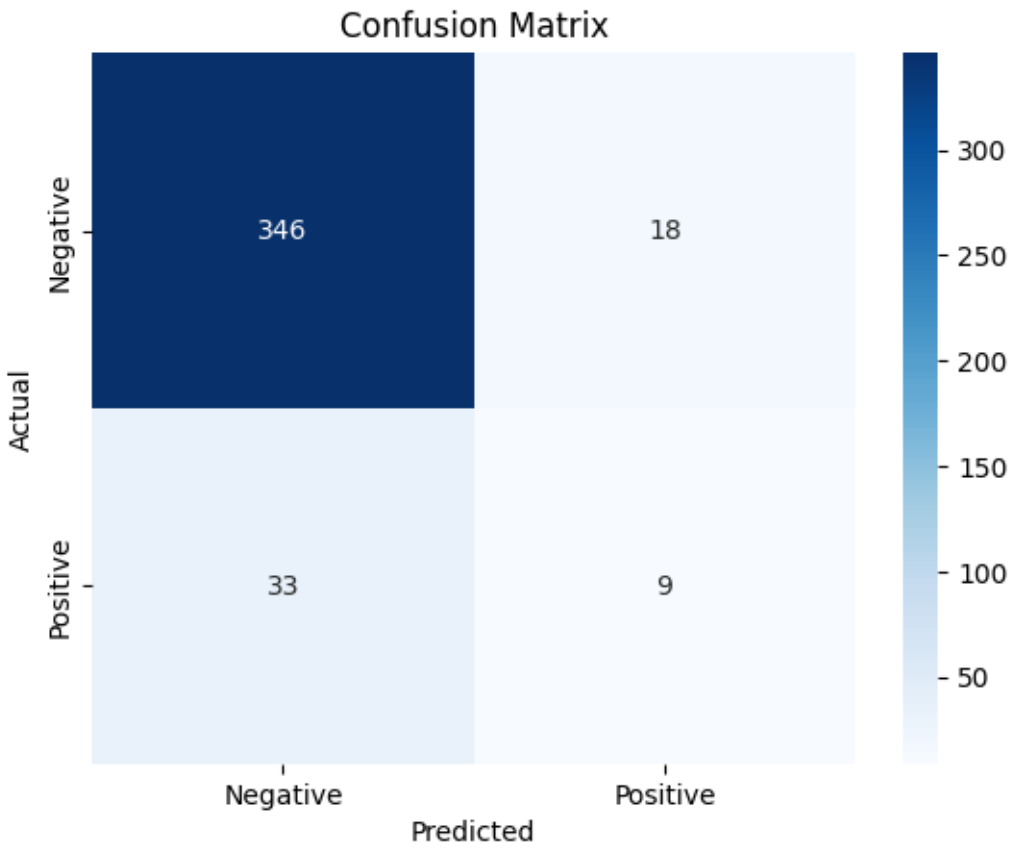
[11]:
```
RandomForestClassifier()
```

[12]:
```
# Predict the labels for the test set
y_pred = clf.predict(X_test)
```

[13]:
```
# Calculate confusion matrix
cm = confusion_matrix(y_test, y_pred)
```

[14]:
```
# Display confusion matrix as heatmap
labels = ['Negative', 'Positive']
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', xticklabels=labels,␣
 ↪yticklabels=labels)
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title('Confusion Matrix')
plt.show()

# Generate classification report
report = classification_report(y_test, y_pred)
print("Classification Report:")
print(report)
```




```
Classification Report:
              precision    recall  f1-score   support

           0       0.91      0.95      0.93       364
           1       0.33      0.21      0.26        42

    accuracy                           0.87       406
   macro avg       0.62      0.58      0.60       406
weighted avg       0.85      0.87      0.86       406
```

[15]:
```python
# Extract confusion matrix elements
tn = cm[0, 0]
fp = cm[0, 1]
fn = cm[1, 0]
tp = cm[1, 1]

# Print confusion matrix
print("Confusion Matrix:")
print(cm)
```

```
# Print confusion matrix elements
print("True Negatives:", tn)
print("False Positives:", fp)
print("False Negatives:", fn)
print("True Positives:", tp)
```

```
Confusion Matrix:
[[346  18]
 [ 33   9]]
True Negatives: 346
False Positives: 18
False Negatives: 33
True Positives: 9
```

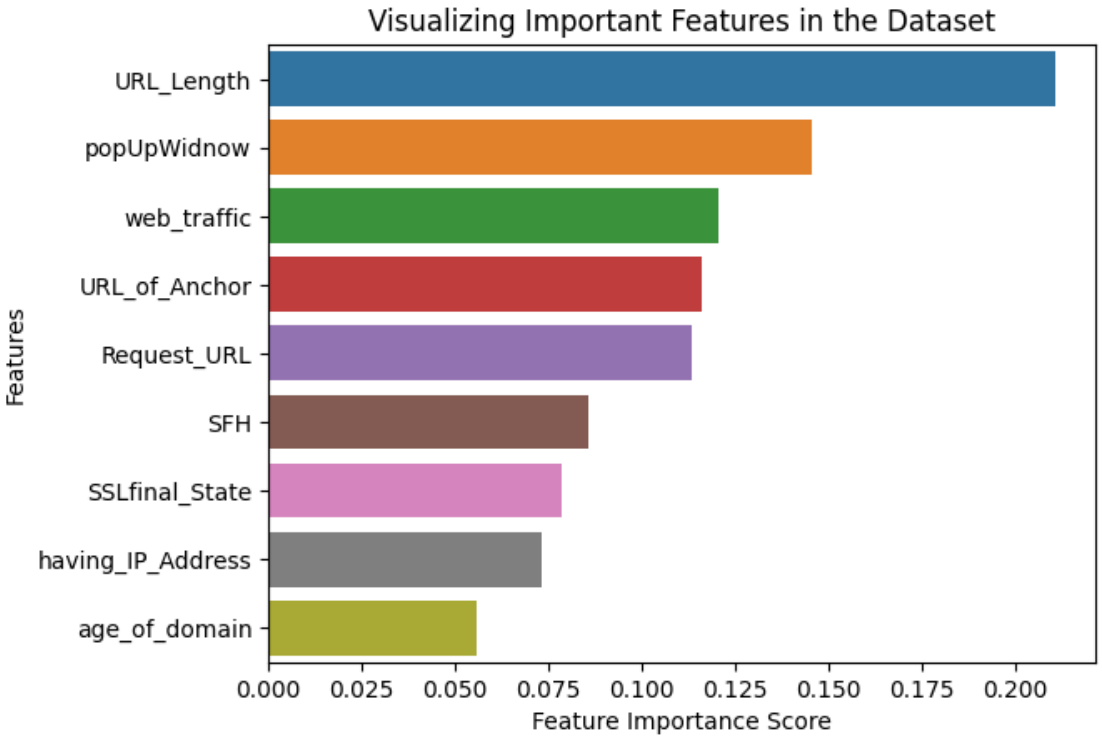
[16]:
```
accuracy = clf.score(X_test, y_test)
error_rate = (fp + fn) / (tn + fp + fn + tp)
sensitivity = tp / (tp + fn)
specificity = tn / (tn + fp)
false_positive_rate = fp / (fp + tn)
```

[17]:
```
# Print evaluation metrics
print("Accuracy:", accuracy)
print("Error Rate:", error_rate)
print("Sensitivity:", sensitivity)
print("Specificity:", specificity)
print("False Positive Rate:", false_positive_rate)
```

```
Accuracy: 0.874384236453202
Error Rate: 0.12561576354679804
Sensitivity: 0.21428571428571427
Specificity: 0.9505494505494505
False Positive Rate: 0.04945054945054945
```

[18]:
```
# Get feature importances
feature_imp = pd.Series(clf.feature_importances_, index=feature_cols).
  ↪sort_values(ascending=False)
```

[19]:
```
# Visualize feature importances
plt.figure()
sns.barplot(x=feature_imp, y=feature_imp.index)
plt.xlabel('Feature Importance Score')
plt.ylabel('Features')
plt.title("Visualizing Important Features in the Dataset")
plt.show()
```


Visualizing Important Features in the Dataset
URL_Length
popUpWidnow
web_traffic
URL_of_Anchor
Request_URL
SFH
SSLfinal_State
having_IP_Address
age_of_domain
Features
0.000
0.025
0.050
0.075
0.100
0.125
0.150
0.175
0.200
Feature Importance Score