Phishing Detector with LR

DESCRIPTION

Background of Problem Statement :

You are expected to write the code for a binary classification model (phishing website or not) using Python Scikit-Learn that trains on the data and calculates the accuracy score on the test data. You have to use one or more of the classification algorithms to train a model on the phishing website dataset.

Problem Objective :

The dataset is a text file which provides the following resources that can be used as inputs for model building :

1. A collection of website URLs for 11000+ websites. Each sample has 30 website parameters and a class label identifying it as a phishing website or not (1 or -1).
2. The code template containing these code blocks:

- Import modules (Part 1)
- Load data function + input/output field descriptions

The dataset also serves as an input for project scoping and tries to specify the functional and non-functional requirements for it.

Domain: Cyber Security and Web Mining

Questions to be answered with analysis :

1. Write the code for a binary classification model (phishing website or not) using Python Scikit-Learn that trains on the data and calculates the accuracy score on the test data.
2. Use one or more of the classification algorithms to train a model on the phishing website dataset.

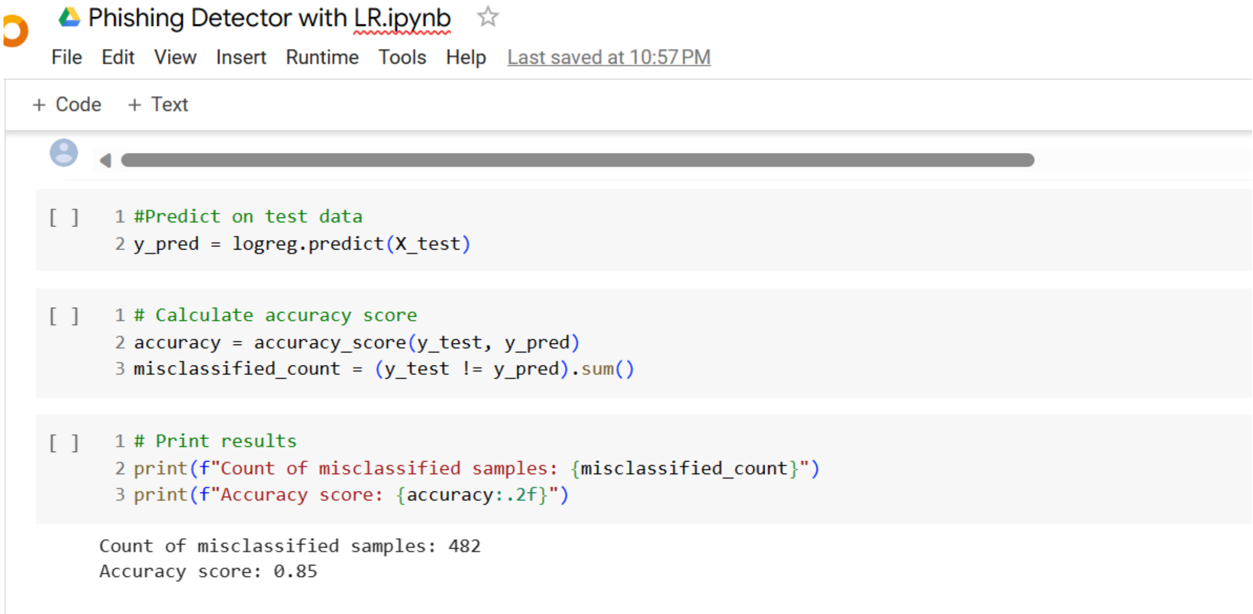
Analysis Tasks to be performed:

- **Initiation :**

1. Begin by creating a new ipynb file and load the dataset in it.

- **Exercise 1** :

1. Build a phishing website classifier using Logistic Regression with “C” parameter = 100.
2. Use 70% of data as training data and the remaining 30% as test data.
   [Hint: Use Scikit-Learn library LogisticRegression]
   [Hint: Refer to the logistic regression tutorial taught earlier in the course]
3. Print count of misclassified samples in the test data prediction as well as the accuracy score of the model.



```python
#Predict on test data
y_pred = logreg.predict(X_test)
```

```python
# Calculate accuracy score
accuracy = accuracy_score(y_test, y_pred)
misclassified_count = (y_test != y_pred).sum()
```

```python
# Print results
print(f"Count of misclassified samples: {misclassified_count}")
print(f"Accuracy score: {accuracy:.2f}")
```

```
Count of misclassified samples: 482
Accuracy score: 0.85
```

- **Exercise 2 :**

1. Train with only two input parameters - parameter Prefix_Suffix and 13 URL_of_Anchor.
2. Check accuracy using the test data and compare the accuracy with the previous value.
3. Plot the test samples along with the decision boundary when trained with index 5 and index 13 parameters.

Hint :

- The dataset is a “.txt” file with no headers and has only the column values.

- The actual column-wise header is described above and, if needed, you can add the header manually.
- The header list is as follows :

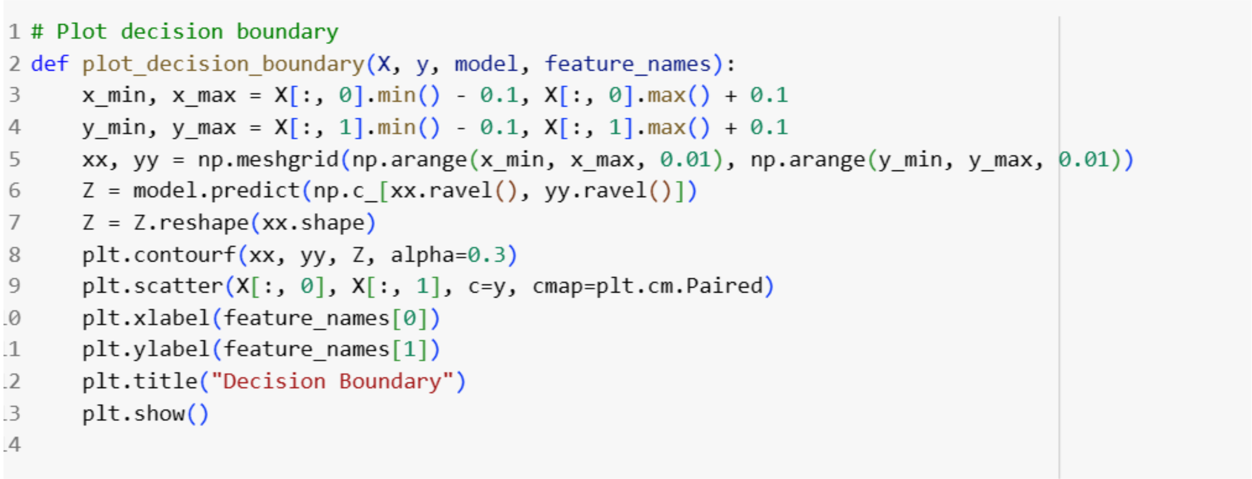
```
[ 'UsingIP', 'LongURL', 'ShortURL', 'Symbol@', 'Redirecting//',
        'PrefixSuffix-', 'SubDomains', 'HTTPS', 'DomainRegLen', 'Favicon',
        'NonStdPort', 'HTTPSDomainURL', 'RequestURL', 'AnchorURL',
        'LinksInScriptTags', 'ServerFormHandler', 'InfoEmail', 'AbnormalURL',
        'WebsiteForwarding', 'StatusBarCust', 'DisableRightClick',
        'UsingPopupWindow', 'IframeRedirection', 'AgeofDomain',
        'DNSRecording', 'WebsiteTraffic', 'PageRank', 'GoogleIndex',
        'LinksPointingToPage', 'StatsReport', 'class' ]
```

Dataset Description :

Field	Description
UsingIP	(categorical - signed numeric) : { -1,1 }
LongURL	(categorical - signed numeric) : { 1,0,-1 }
ShortURL	(categorical - signed numeric) : { 1,-1 }
Symbol@	(categorical - signed numeric) : { 1,-1 }
Redirecting//	(categorical - signed numeric) : { -1,1 }
PrefixSuffix-	(categorical - signed numeric) : { -1,1 }
SubDomains	(categorical - signed numeric) : { -1,0,1 }
HTTPS	(categorical - signed numeric) : { -1,1,0 }
DomainRegLen	(categorical - signed numeric) : { -1,1 }
Favicon	(categorical - signed numeric) : { 1,-1 }
NonStdPort	(categorical - signed numeric) : { 1,-1 }
HTTPSDomainURL	(categorical - signed numeric) : { -1,1 }
RequestURL	(categorical - signed numeric) : { 1,-1 }
AnchorURL	(categorical - signed numeric) : { -1,0,1 }
LinksInScriptTags	(categorical - signed numeric) : { 1,-1,0 }
ServerFormHandler	(categorical - signed numeric) : { -1,1,0 }
InfoEmail	(categorical - signed numeric) : { -1,1 }
AbnormalURL	(categorical - signed numeric) : { -1,1 }
WebsiteForwarding	(categorical - signed numeric) : { 0,1 }
StatusBarCust	(categorical - signed numeric) : { 1,-1 }
DisableRightClick	(categorical - signed numeric) : { 1,-1 }
UsingPopupWindow	(categorical - signed numeric) : { 1,-1 }
IframeRedirection	(categorical - signed numeric) : { 1,-1 }
AgeOfDomain	(categorical - signed numeric) : { -1,1 }
DNSRecording	(categorical - signed numeric) : { -1,1 }
WebsiteTraffic	(categorical - signed numeric) : { -1,0,1 }
PageRank	(categorical - signed numeric) : { -1,1 }
GoogleIndex	(categorical - signed numeric) : { 1,-1 }

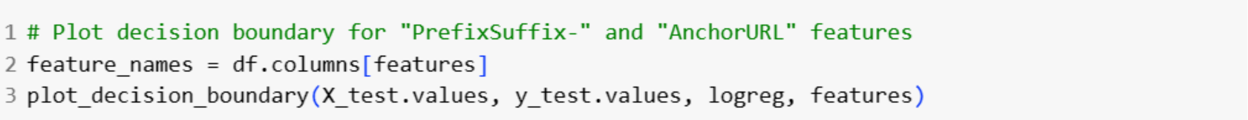
LinksPointingToPage	(categorical - signed numeric) : { 1,0,-1 }
StatsReport	(categorical - signed numeric) : { -1,1 }
Class	(categorical - signed numeric) : { -1,1 }

Dataset Size : 11055 rows x 31 columns

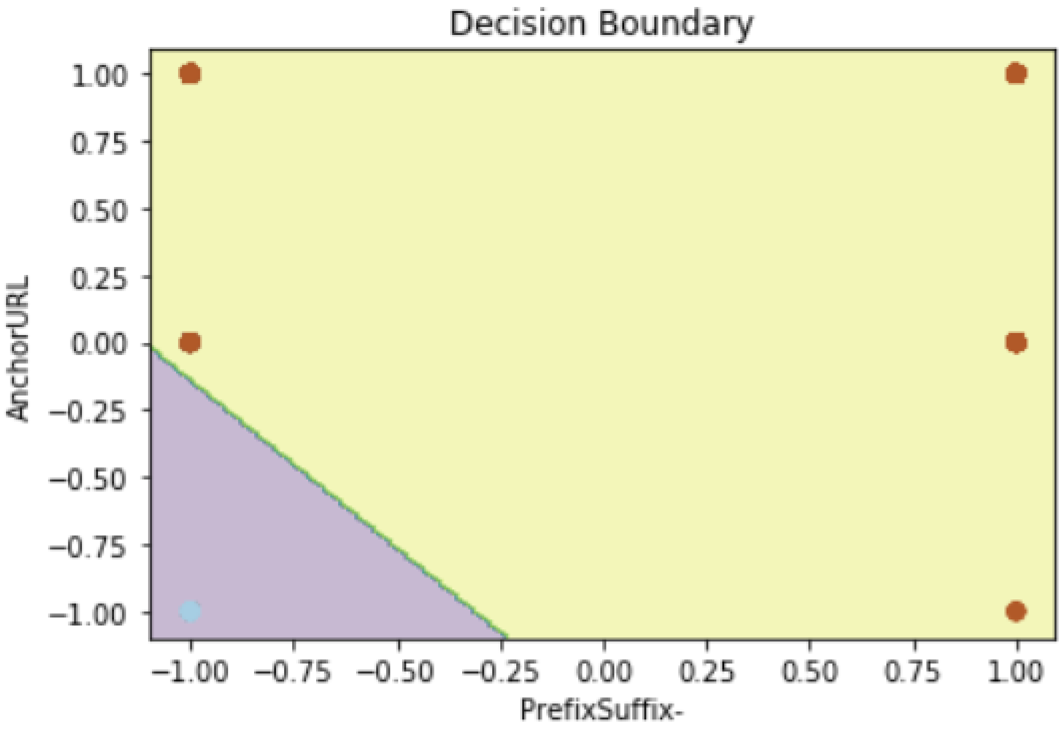


```python
# Plot decision boundary
def plot_decision_boundary(X, y, model, feature_names):
    x_min, x_max = X[:, 0].min() - 0.1, X[:, 0].max() + 0.1
    y_min, y_max = X[:, 1].min() - 0.1, X[:, 1].max() + 0.1
    xx, yy = np.meshgrid(np.arange(x_min, x_max, 0.01), np.arange(y_min, y_max, 0.01))
    Z = model.predict(np.c_[xx.ravel(), yy.ravel()])
    Z = Z.reshape(xx.shape)
    plt.contourf(xx, yy, Z, alpha=0.3)
    plt.scatter(X[:, 0], X[:, 1], c=y, cmap=plt.cm.Paired)
    plt.xlabel(feature_names[0])
    plt.ylabel(feature_names[1])
    plt.title("Decision Boundary")
    plt.show()

```



```python
# Plot decision boundary for "PrefixSuffix-" and "AnchorURL" features
feature_names = df.columns[features]
plot_decision_boundary(X_test.values, y_test.values, logreg, features)
```


Decision Boundary
AnchorURL
PrefixSuffix-