simplilearn

Assignment: Phishing Detector with LR

The comments/sections provided are your cues to perform the assignment. You don't need to limit yourself to the number of rows/cells provided. You can add additional rows in each section to add more lines of code.

If at any point in time you need help on solving this assignment, view our demo video to understand the different steps of the code.

Happy coding!

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.
- **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 }

Initiation

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

In [1]:
```python
#import packages
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
```

In [2]:
```python
#load Dataset and
phishingData = pd.read_csv('phishing.txt')
phishingData.head()
```

Out[2]:

	-1	1	1.1	1.2	-1.1	-1.2	-1.3	-1.4	-1.5	1.3	...	1.9	1.10	-1.11	-1.12	-1.13	-1.14	1.11	1.12	-1.15
0	1	1	1	1	1	-1	0	1	-1	1	...	1	1	-1	-1	0	-1	1	1	1
1	1	0	1	1	1	-1	-1	-1	-1	1	...	1	1	1	-1	1	-1	1	0	-1
2	1	0	1	1	1	-1	-1	-1	1	1	...	1	1	-1	-1	1	-1	1	-1	1
3	1	0	-1	1	1	-1	1	1	-1	1	...	-1	1	-1	-1	0	-1	1	1	1
4	-1	0	-1	1	-1	-1	1	1	-1	1	...	1	1	1	1	1	-1	1	-1	-1

5 rows × 31 columns

In [3]:
```python
phishingData.describe()
```

Out[3]:

	-1	1	1.1	1.2	-1.1	-1.2	-1.3
count	11054.000000	11054.000000	11054.000000	11054.000000	11054.000000	11054.000000	11054.000000
mean	0.313914	-0.633345	0.738737	0.700561	0.741632	-0.734938	0.064049
std	0.949495	0.765973	0.674024	0.713625	0.670837	0.678165	0.817492
min	-1.000000	-1.000000	-1.000000	-1.000000	-1.000000	-1.000000	-1.000000
25%	-1.000000	-1.000000	1.000000	1.000000	1.000000	-1.000000	-1.000000
50%	1.000000	-1.000000	1.000000	1.000000	1.000000	-1.000000	0.000000
75%	1.000000	-1.000000	1.000000	1.000000	1.000000	-1.000000	1.000000
max	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000

8 rows × 31 columns

In [4]:
```python
phishingData.shape
```

Out[4]: (11054, 31)

Exercise 1

Use 70% of data as training data and the remaining 30% as test data.

In [5]:
```python
#classify features and label sets
X = phishingData.iloc[:,:-1].values
y = phishingData.iloc[:,30].values
```

In [6]:
```python
#split features and label into training ang testing data
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=0.3,random_state=4)
```

In [7]:
```python
#perform feature scaling
from sklearn.preprocessing import StandardScaler
scalar = StandardScaler()
X_train = scalar.fit_transform (X_train)
X_test = scalar.fit_transform (X_test)
```

Build a phishing website classifier using Logistic Regression with “C” parameter = 100.

In [8]:
```python
#Logistic Regression Classifier
from sklearn.linear_model import LogisticRegression
LRclassifier = LogisticRegression(C=100,random_state=0)
LRclassifier.fit(X_train,y_train)
```

Out[8]: LogisticRegression(C=100, random_state=0)

In [9]:
```python
LRpredict = LRclassifier.predict(X_test)
LRpredict
```

Out[9]: array([-1, -1, 1, ..., 1, 1, -1], dtype=int64)

In [10]:
```python
#LRC training score
LRclassifier.score(X_train,y_train)
```

Out[10]: 0.9298177588212485

In [11]:
```python
#LRC test score
LRclassifier.score(X_test,y_test)
```

Out[11]: 0.9267410310521556

Print count of misclassified samples in the test data prediction as well as the accuracy score of the model

In [12]:
```python
#confusion matrix for printing count of misclassified samples in the test data predic
from sklearn.metrics import confusion_matrix
confusionMatrix = confusion_matrix(y_test,LRpredict)
confusionMatrix
```

Out[12]:
```
array([[1355,  155],
       [  88, 1719]], dtype=int64)
```

In [13]:
```python
# classify as features(Prefix_Suffix and URL_of_Anchor) and label with index 5
X = phishingData.iloc[0:5,[6,14]].values
y = phishingData.iloc[0:5,30].values
```

In [14]:
```python
#split features and label into training ang testing data
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=0.3,random_state=4)
```

In [15]:
```python
#perform feature scaling
from sklearn.preprocessing import StandardScaler
scalar = StandardScaler()
X_train = scalar.fit_transform (X_train)
X_test = scalar.fit_transform (X_test)
```

In [16]:
```python
#Logistic Regression Classifier
from sklearn.linear_model import LogisticRegression
LRclassifier1 = LogisticRegression(C=100,random_state=0)
LRclassifier1.fit(X_train,y_train)
```

Out[16]: LogisticRegression(C=100, random_state=0)

In [17]:
```python
LRpredict1 = LRclassifier1.predict(X_test)
LRpredict1
```

Out[17]: array([-1, 1], dtype=int64)

In [18]:
```python
#LRC training score
LRclassifier1.score(X_train,y_train)
```

Out[18]: 1.0

In [19]:
```python
#LRC test score
LRclassifier1.score(X_test,y_test)
```

Out[19]: 1.0

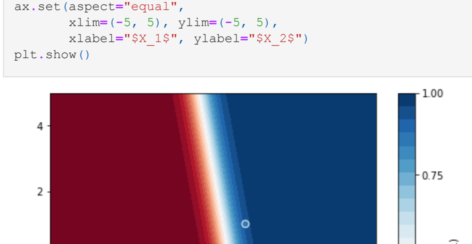
In [20]:
```python
#confusion matrix for printing count of misclassified samples in the test data predic
from sklearn.metrics import confusion_matrix
LRconfusionMatrix1 = confusion_matrix(y_test,LRpredict1)
LRconfusionMatrix1
```

Out[20]:
```
array([[1, 0],
       [0, 1]], dtype=int64)
```

In [21]:
```python
#visualize the Test set
xx, yy = np.mgrid[-5:5:.01, -5:5:.01]
grid = np.c_[xx.ravel(), yy.ravel()]
probs = LRclassifier1.predict_proba(grid)[:, 1].reshape(xx.shape)
print(probs)
```

```
[[1.69212931e-11 1.70416335e-11 1.71628297e-11 ... 1.98095309e-08
  1.99504118e-08 2.00922946e-08]
 [1.75868947e-11 1.77119688e-11 1.78379323e-11 ... 2.05887418e-08
  2.07351643e-08 2.08826281e-08]
 [1.82786780e-11 1.84086718e-11 1.85395901e-11 ... 2.13986032e-08
  2.15507852e-08 2.17040495e-08]
 ...
 [9.99998835e-01 9.99998844e-01 9.99998852e-01 ... 9.99999999e-01
  9.99999999e-01 9.99999999e-01]
 [9.99998879e-01 9.99998887e-01 9.99998895e-01 ... 9.99999999e-01
  9.99999999e-01 9.99999999e-01]
 [9.99998922e-01 9.99998929e-01 9.99998937e-01 ... 9.99999999e-01
  9.99999999e-01 9.99999999e-01]]
```

In [22]:
```python
f, ax = plt.subplots(figsize=(8, 6))
contour = ax.contourf(xx, yy, probs, 25, cmap="RdBu",
                      vmin=0, vmax=1)
ax_c = f.colorbar(contour)
ax_c.set_label("$P(y = 1)$")
ax_c.set_ticks([0, .25, .5, .75, 1])
ax.scatter(X_test[:, 0], X_test[:, 1],c = (y_test == 1 ), s=50,
           cmap="RdBu", vmin=-.2, vmax=1.2,
           edgecolor="white", linewidth=1)
ax.set(aspect="equal",
       xlim=(-5, 5), ylim=(-5, 5),
       xlabel="$X_1$", ylabel="$X_2$")
plt.show()
```



Exercise 2

Train with only two input parameters - parameter Prefix_Suffix and 13 URL_of_Anchor

In [23]:
```python
# classify as features(Prefix_Suffix and URL_of_Anchor) and label with index 13
X = phishingData.iloc[0:13,[6,14]].values
y = phishingData.iloc[0:13,30].values
```

In [24]:
```python
#split features and label into training ang testing data
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=0.3,random_state=4)
```

In [25]:
```python
#perform feature scaling
from sklearn.preprocessing import StandardScaler
scalar = StandardScaler()
X_train = scalar.fit_transform (X_train)
X_test = scalar.transform (X_test)
```

In [26]:
```python
#Logistic Regression Classifier
from sklearn.linear_model import LogisticRegression
LRclassifier11 = LogisticRegression(C=100,random_state=0)
LRclassifier11.fit(X_train,y_train)
```

Out[26]: LogisticRegression(C=100, random_state=0)

In [27]:
```python
LRpredict11 = LRclassifier11.predict(X_test)
LRpredict11
```

Out[27]: array([1, 1, -1, -1], dtype=int64)

Check accuracy using the test data and compare the accuracy with the previous value

In [28]:
```python
#LRC training score
LRclassifier11.score(X_train,y_train)
```

Out[28]: 0.8888888888888888

In [29]:
```python
#LRC test score
LRclassifier11.score(X_test,y_test)
```

Out[29]: 1.0

Training with only two input parameters has test score of '1', which is higher than the previous score of '0.92'.

In [30]:
```python
#confusion matrix for printing count of misclassified samples in the test data predic
from sklearn.metrics import confusion_matrix
LRconfusionMatrix11 = confusion_matrix(y_test,LRpredict11)
```

In [31]:
```python
#visualize the Test set
xx, yy = np.mgrid[-5:5:.01, -5:5:.01]
grid = np.c_[xx.ravel(), yy.ravel()]
probs = LRclassifier11.predict_proba(grid)[:, 1].reshape(xx.shape)
print(probs)
```

```
[[4.35180374e-07 4.39590919e-07 4.44046165e-07 ... 1.00132212e-02
  1.01136787e-02 1.02151336e-02]
 [4.41905577e-07 4.46384282e-07 4.50908379e-07 ... 1.01663907e-02
  1.02683689e-02 1.03713594e-02]
 [4.48734711e-07 4.53282629e-07 4.57876640e-07 ... 1.03218788e-02
  1.04254003e-02 1.05299489e-02]
 ...
 [6.55234823e-01 6.57509234e-01 6.59776432e-01 ... 9.99977362e-01
  9.99977589e-01 9.99977814e-01]
 [6.58690882e-01 6.60954291e-01 6.63210365e-01 ... 9.99977706e-01
  9.99977930e-01 9.99978151e-01]
 [6.62130161e-01 6.64382383e-01 6.66627151e-01 ... 9.99978046e-01
  9.99978266e-01 9.99978484e-01]]
```

Plot the test samples along with the decision boundary when trained with index 5 and index 13 parameters.

In [32]:
```python
f, ax = plt.subplots(figsize=(8, 6))
contour = ax.contourf(xx, yy, probs, 25, cmap="RdBu",
                      vmin=0, vmax=1)
ax_c = f.colorbar(contour)
ax_c.set_label("$P(y = 1)$")
ax_c.set_ticks([0, .25, .5, .75, 1])
ax.scatter(X_test[:, 0], X_test[:, 1],c = (y_test == 1 ), s=50,
           cmap="RdBu", vmin=-.2, vmax=1.2,
           edgecolor="white", linewidth=1)
ax.set(aspect="equal",
       xlim=(-5, 5), ylim=(-5, 5),
       xlabel="$X_1$", ylabel="$X_2$")
plt.show()
```