Project - Phishing Detector using LR

Project - Phishing Detector using LR

Description :

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: a. Import modules (Part 1) b. 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.

Background of the 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. Domain : Cyber Security and Web Mining Dataset Description :

Data Dictionary – Variable and 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 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']

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.

Project Guidelines :

1. Initiation :

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

2. Exercise 1 :

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

● 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]

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

3. Exercise 2 :

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

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

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

importing pandas and numpy as pd and np

In [1]:
```
import numpy as np
import pandas as pd
```

Import a "phishing.txt" file and Creating a dataframe for analysis

In [2]:
```
data=pd.read_csv("phishing.txt",names=[ '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' ])
```

In [3]:
```
data.head()
```

Out[3]:

	UsingIP	LongURL	ShortURL	Symbol@	Redirecting//	PrefixSuffix-	SubDomains	HTTPS	DomainRegLen	Favicon	...	UsingPopupWindow	Ifran
0	-1	1	1	1	-1	-1	-1	-1	-1	1	...	1	
1	1	1	1	1	1	-1	0	1	-1	1	...	1	
2	1	0	1	1	1	-1	-1	-1	-1	1	...	1	
3	1	0	1	1	1	-1	-1	-1	1	1	...	1	
4	1	0	-1	1	1	-1	1	1	-1	1	...	-1	

5 rows × 31 columns

In [4]:
```
data['class'].value_counts()
```

Out[4]:
```
 1    6157
-1    4898
Name: class, dtype: int64
```

All columns are numeric and does not having any null value

11055 rows × 31 columns

In [5]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 11055 entries, 0 to 11054
Data columns (total 31 columns):
UsingIP                11055 non-null int64
LongURL                11055 non-null int64
ShortURL               11055 non-null int64
Symbol@                11055 non-null int64
Redirecting//          11055 non-null int64
PrefixSuffix-          11055 non-null int64
SubDomains             11055 non-null int64
HTTPS                  11055 non-null int64
DomainRegLen           11055 non-null int64
Favicon                11055 non-null int64
NonStdPort             11055 non-null int64
HTTPSDomainURL         11055 non-null int64
RequestURL             11055 non-null int64
AnchorURL              11055 non-null int64
LinksInScriptTags      11055 non-null int64
ServerFormHandler      11055 non-null int64
InfoEmail              11055 non-null int64
AbnormalURL            11055 non-null int64
WebsiteForwarding      11055 non-null int64
StatusBarCust          11055 non-null int64
DisableRightClick      11055 non-null int64
UsingPopupWindow       11055 non-null int64
IframeRedirection      11055 non-null int64
AgeofDomain            11055 non-null int64
DNSRecording           11055 non-null int64
WebsiteTraffic         11055 non-null int64
PageRank               11055 non-null int64
GoogleIndex            11055 non-null int64
LinksPointingToPage    11055 non-null int64
StatsReport            11055 non-null int64
class                  11055 non-null int64
dtypes: int64(31)
memory usage: 2.6 MB
```

Exercise 1

- Build a phishing website classifier using Logistic Regression with “C” parameter = 100.
- Use 70% of data as training data and the remaining 30% as test data.
- Print count of misclassified samples in the test data prediction as well as the accuracy score of the model.

In [6]:
```
features = data.iloc[:,:-1].values
label = data.iloc[:,-1].values
```

In [7]:
```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test = train_test_split(features,
                                                 label,
                                                 test_size = 0.3,
                                                 random_state =4)
```

In [8]:
```
# c = Inverse regularization parameter
#  smaller values specify stronger regularization
#  Higher values specify less regularization
from sklearn.linear_model import LogisticRegression
logistic_model= LogisticRegression(C = 100)
logistic_model.fit(x_train,y_train)
print("training score = ",logistic_model.score(x_train,y_train))
print("testing score = ",logistic_model.score(x_test,y_test))
```

```
C:\Users\nilesh\Anaconda3\lib\site-packages\sklearn\linear_model\logistic.py:432: FutureWarning: Default solver will be changed to 'lbfgs' in 0.22. Specify a solver to silence this warning.
  FutureWarning)

training score =  0.9281468079607134
testing score =  0.9285498944829665
```

In [9]:
```
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test , logistic_model.predict(x_test))
cm
```

Out[9]:
```
array([[1304,  125],
       [ 112, 1776]], dtype=int64)
```

In [10]:
```
# cm.diagonal().sum()/cm.sum()
from sklearn.metrics import accuracy_score
print("accuracy = ",accuracy_score(y_test , logistic_model.predict(x_test)))
```

```
accuracy =  0.9285498944829665
```

In [11]:
```
cm.sum()-cm.diagonal().sum()
```

Out[11]:
```
237
```

In [12]:
```
misclassified = np.where(y_test != logistic_model.predict(x_test))
misclassified=np.array(misclassified)
print("Misclassified samples = ",misclassified.shape[1])
```

```
Misclassified samples =  237
```

In [13]:
```
misclassified
```

Out[13]:
```
array([[  15,   19,   29,   55,   79,   80,   86,   93,   98,  111,  115,
         155,  158,  161,  165,  191,  195,  203,  213,  228,  237,  246,
         259,  262,  292,  313,  336,  344,  357,  358,  364,  378,  380,
         398,  423,  426,  468,  517,  520,  528,  541,  569,  591,  599,
         616,  628,  646,  660,  664,  671,  675,  693,  736,  742,  766,
         804,  825,  838,  854,  863,  873,  892,  906,  917,  943,  961,
         964,  978,  984,  985, 1005, 1011, 1018, 1027, 1048, 1064, 1080,
        1112, 1125, 1143, 1156, 1164, 1180, 1202, 1203, 1242, 1251, 1254,
        1259, 1265, 1285, 1291, 1296, 1332, 1336, 1344, 1356, 1362, 1375,
        1377, 1379, 1406, 1408, 1458, 1502, 1507, 1516, 1518, 1524, 1547,
        1565, 1579, 1583, 1586, 1588, 1590, 1620, 1631, 1675, 1677, 1696,
        1698, 1703, 1718, 1737, 1747, 1752, 1761, 1791, 1803, 1811, 1823,
        1824, 1834, 1868, 1878, 1879, 1883, 1890, 1908, 1945, 1950, 1980,
        2001, 2002, 2015, 2036, 2037, 2051, 2096, 2127, 2133, 2140, 2173,
        2230, 2233, 2235, 2258, 2300, 2305, 2316, 2322, 2328, 2331, 2367,
        2397, 2417, 2422, 2427, 2429, 2430, 2442, 2455, 2458, 2461, 2462,
        2480, 2484, 2499, 2523, 2555, 2578, 2598, 2603, 2678, 2726, 2728,
        2737, 2741, 2744, 2749, 2766, 2769, 2780, 2785, 2786, 2788, 2790,
        2802, 2814, 2825, 2827, 2834, 2864, 2897, 2916, 2934, 2936, 2945,
        2977, 2990, 3003, 3015, 3016, 3045, 3047, 3073, 3090, 3120, 3149,
        3155, 3174, 3176, 3186, 3188, 3199, 3206, 3227, 3235, 3258, 3266,
        3267, 3287, 3294, 3296, 3299, 3303]], dtype=int64)
```

Exercise 2

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

- Check accuracy using the test data and compare the accuracy with the previous value.
- Plot the test samples along with the decision boundary when trained with index 5 and index 13 parameters.m

In [14]:
```
features1 = data.loc[:,['PrefixSuffix-','AnchorURL']].values
label1 = data.iloc[:,-1].values
```

In [15]:
```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test = train_test_split(features1,
                                                 label1,
                                                 test_size = 0.3,
                                                 random_state =4)
```

In [16]:
```
from sklearn.linear_model import LogisticRegression
logistic_model1 = LogisticRegression(C = 100)
logistic_model1.fit(x_train,y_train)
print("training score = ",logistic_model1.score(x_train,y_train))
print("testing score = ",logistic_model1.score(x_test,y_test))
```

```
training score =  0.8444042388214009
testing score =  0.859511606873681

C:\Users\nilesh\Anaconda3\lib\site-packages\sklearn\linear_model\logistic.py:432: FutureWarning: Default solver will be
changed to 'lbfgs' in 0.22. Specify a solver to silence this warning.
  FutureWarning)
```

In [17]:
```
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test , logistic_model1.predict(x_test))
cm
```

Out[17]:
```
array([[ 966,  463],
       [   3, 1885]], dtype=int64)
```

In [18]:
```
from sklearn.metrics import accuracy_score
print("accuracy = ",accuracy_score(y_test , logistic_model1.predict(x_test)))
```

```
accuracy =  0.859511606873681
```

In [19]:
```
cm.sum()-cm.diagonal().sum()
```

Out[19]:
```
466
```

In [20]:
```
misclassified = np.where(y_test != logistic_model1.predict(x_test))
misclassified=np.array(misclassified)
print("Misclassified samples = ",misclassified.shape[1])
```

```
Misclassified samples =  466
```

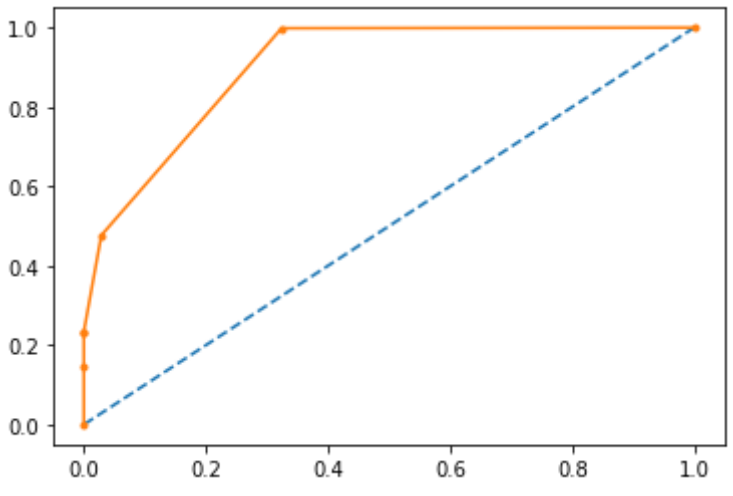
In [21]:
```
misclassified
```

Out[21]:
```
array([[   1,   15,   20,   23,   26,   47,   50,   55,   62,   71,   76,
          79,   82,   86,   97,  104,  123,  125,  127,  128,  132,  137,
         140,  144,  155,  157,  161,  164,  165,  195,  203,  205,  210,
         226,  228,  241,  244,  246,  247,  259,  260,  265,  268,  276,
         281,  285,  292,  294,  295,  313,  336,  338,  344,  347,  358,
         361,  362,  364,  373,  374,  378,  396,  398,  402,  406,  421,
         423,  448,  455,  468,  476,  483,  506,  507,  517,  523,  541,
         552,  560,  569,  574,  581,  588,  599,  600,  607,  612,  616,
         629,  635,  637,  671,  675,  687,  693,  699,  701,  722,  751,
         758,  761,  765,  766,  767,  769,  782,  830,  854,  870,  873,
         879,  931,  935,  948,  955,  956,  961,  975,  978,  980, 1001,
        1005, 1020, 1022, 1028, 1036, 1045, 1046, 1059, 1068, 1080, 1093,
        1099, 1107, 1112, 1116, 1123, 1127, 1129, 1143, 1144, 1158, 1160,
        1180, 1188, 1202, 1203, 1204, 1212, 1221, 1225, 1239, 1243, 1251,
        1252, 1262, 1265, 1270, 1275, 1285, 1302, 1306, 1318, 1324, 1332,
        1336, 1344, 1346, 1348, 1356, 1358, 1362, 1377, 1380, 1381, 1383,
        1392, 1406, 1408, 1412, 1414, 1418, 1420, 1424, 1433, 1442, 1446,
        1448, 1449, 1450, 1451, 1458, 1461, 1464, 1465, 1467, 1469, 1474,
        1495, 1497, 1501, 1516, 1524, 1547, 1559, 1567, 1568, 1574, 1576,
```

In [23]:
```
import matplotlib.pyplot as plt
from sklearn.metrics import roc_auc_score
from sklearn.metrics import roc_curve
proba = logistic_model1.predict_proba(x_test)
proba = proba[:,1]
auc = roc_auc_score(y_test,proba)
print(auc)
fpr,tpr,_ = roc_curve(y_test,proba)
plt.plot([0,1],[0,1],linestyle = "--")
plt.plot(fpr,tpr,marker = ".")
```

```
0.9034649245056992
```

Out[23]:
```
[<matplotlib.lines.Line2D at 0x1f08a061860>]
```



Model 1 Acurracy is better than Model 2

In [24]:
```
pd.DataFrame([[0.928],[0.859]],columns = ["Accuracy"], index=["model 1 " , "model 2"])
```

Out[24]:

	Accuracy
model 1	0.928
model 2	0.859