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.

In [131]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')
```

In [ ]:
```
```

In [132]:
```
colnames = [ '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 [133]:
```
data = pd.read_csv('phishing.txt', sep=',', names=colnames, header=None)
```

In [134]:
```
data.head()
```

Out[134]:

	UsingIP	LongURL	ShortURL	Symbol@	Redirecting//	PrefixSuffix-	SubDomains	HTTPS
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
3	1	0	1	1	1	-1	-1	-1
4	1	0	-1	1	1	-1	1	1

5 rows × 31 columns

In [135]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 11055 entries, 0 to 11054
Data columns (total 31 columns):
 #   Column               Non-Null Count  Dtype
---  ------               --------------  -----
 0   UsingIP              11055 non-null  int64
 1   LongURL              11055 non-null  int64
 2   ShortURL             11055 non-null  int64
 3   Symbol@              11055 non-null  int64
 4   Redirecting//        11055 non-null  int64
 5   PrefixSuffix-        11055 non-null  int64
 6   SubDomains           11055 non-null  int64
 7   HTTPS                11055 non-null  int64
 8   DomainRegLen         11055 non-null  int64
 9   Favicon              11055 non-null  int64
 10  NonStdPort           11055 non-null  int64
 11  HTTPSDomainURL       11055 non-null  int64
 12  RequestURL           11055 non-null  int64
 13  AnchorURL            11055 non-null  int64
 14  LinksInScriptTags    11055 non-null  int64
 15  ServerFormHandler    11055 non-null  int64
 16  InfoEmail            11055 non-null  int64
 17  AbnormalURL          11055 non-null  int64
 18  WebsiteForwarding    11055 non-null  int64
 19  StatusBarCust        11055 non-null  int64
 20  DisableRightClick    11055 non-null  int64
 21  UsingPopupWindow     11055 non-null  int64
 22  IframeRedirection    11055 non-null  int64
 23  AgeofDomain          11055 non-null  int64
 24  DNSRecording         11055 non-null  int64
 25  WebsiteTraffic       11055 non-null  int64
 26  PageRank             11055 non-null  int64
 27  GoogleIndex          11055 non-null  int64
 28  LinksPointingToPage  11055 non-null  int64
 29  StatsReport          11055 non-null  int64
 30  class                11055 non-null  int64
dtypes: int64(31)
memory usage: 2.6 MB
```

In [136]:
```
data.isnull().sum()
```

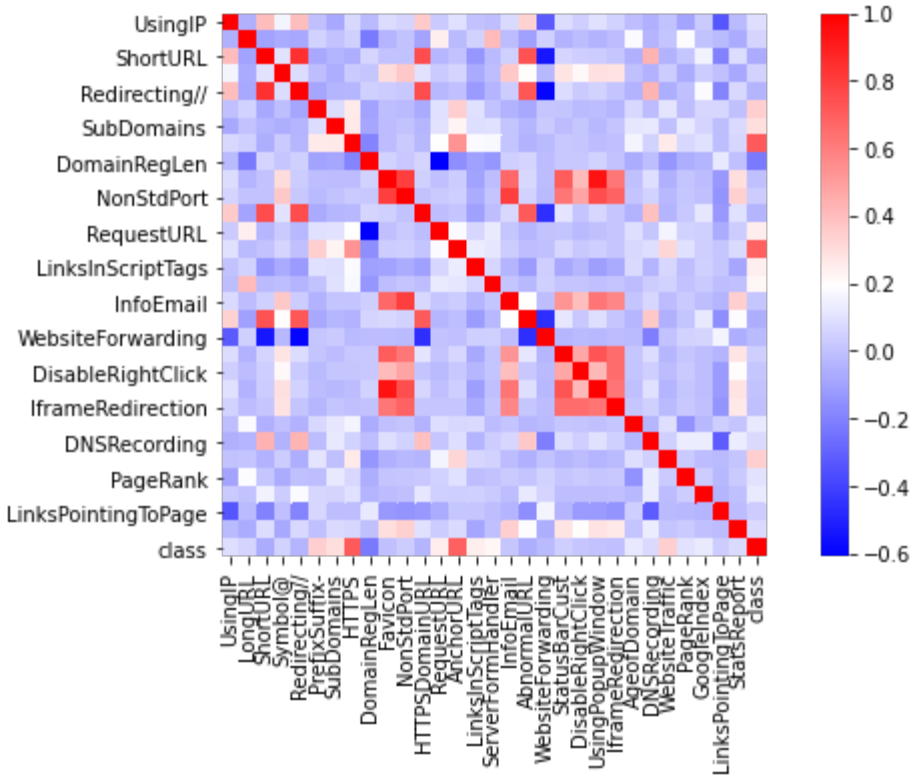
Out[136]:
```
UsingIP                0
LongURL                0
ShortURL               0
Symbol@                0
Redirecting//          0
PrefixSuffix-          0
SubDomains             0
HTTPS                  0
DomainRegLen           0
Favicon                0
NonStdPort             0
HTTPSDomainURL         0
RequestURL             0
AnchorURL              0
LinksInScriptTags      0
ServerFormHandler      0
InfoEmail              0
AbnormalURL            0
WebsiteForwarding      0
StatusBarCust          0
DisableRightClick      0
UsingPopupWindow       0
IframeRedirection      0
AgeofDomain            0
DNSRecording           0
WebsiteTraffic         0
PageRank               0
GoogleIndex            0
LinksPointingToPage    0
StatsReport            0
class                  0
dtype: int64
```

In [ ]:

In [ ]:

In [137]:
```
plt.figure(figsize=(10,5))
sns.heatmap(data= data.corr(), square=True, cmap='bwr')
plt.show
```

Out[137]:
```
<function matplotlib.pyplot.show(close=None, block=None)>
```



In [138]:
```
from sklearn.model_selection import train_test_split
```

In [139]:
```
X = data.drop('class', axis=1)
y = data['class']
```

In [140]:
```
X.shape
```

Out[140]:
```
(11055, 30)
```

In [141]:
```
y.shape
```

Out[141]:
```
(11055,)
```

In [142]:
```
X_train, X_test, y_train, y_test = train_test_split(X,y,train_size=0.
7, random_state=1)
```

In [143]:
```
print(X_train.shape)
print(y_train.shape)
print(X_test.shape)
print(y_test.shape)
```

```
(7738, 30)
(7738,)
(3317, 30)
(3317,)
```

In [149]:
```
from sklearn.preprocessing import StandardScaler
scalar = StandardScaler()
X_train_scaled = scalar.fit_transform(X_train)
X_test_scaled = scalar.fit_transform(X_test)
```

In [150]:
```
from sklearn.linear_model import LogisticRegression
phishing_model = LogisticRegression(C=100.0)
```

In [151]:
```
phishing_model.fit(X_train_scaled, y_train)
```

Out[151]:
```
LogisticRegression(C=100.0)
```

In [152]:
```
y_pred_proba = phishing_model.predict_proba(X_test)
```

In [153]:
```
y_pred_proba
```

Out[153]:
```
array([[0.99714114, 0.00285886],
       [0.01209425, 0.98790575],
       [0.86652021, 0.13347979],
       ...,
       [0.28716797, 0.71283203],
       [0.96974163, 0.03025837],
       [0.45359719, 0.54640281]])
```

In [154]:
```
y_pred = phishing_model.predict(X_test_scaled)
```

In [155]:
```
y_pred
```

Out[155]:
```
array([-1,  1, -1, ...,  1, -1,  1])
```

In [157]:
```
print(phishing_model.score(X_train_scaled,y_train))
print(phishing_model.score(X_test_scaled,y_test))
```

```
0.9285345050400621
0.9267410310521556
```

In [160]:
```
from sklearn.metrics import confusion_matrix, accuracy_score
print(confusion_matrix(y_pred,y_test))
print(accuracy_score(y_pred, y_test))
```

```
[[1350  103]
 [ 140 1724]]
0.9267410310521556
```

In [179]:
```
# 92% accuracy score
```

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.

In [161]:
```
data.head()
```

Out[161]:

	UsingIP	LongURL	ShortURL	Symbol@	Redirecting//	PrefixSuffix-	SubDomains	HTTPS
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
3	1	0	1	1	1	-1	-1	-1
4	1	0	-1	1	1	-1	1	1

5 rows × 31 columns

In [166]:
```
new_X = data[['PrefixSuffix-','AnchorURL']]
```

In [167]:
```
print(new_X.shape)
new_X.head()
```

```
(11055, 2)
```

Out[167]:

	PrefixSuffix-	AnchorURL
0	-1	-1
1	-1	0
2	-1	0
3	-1	0
4	-1	0

In [168]:
```
y.shape
```

Out[168]: (11055,)

In [169]:
```
X_train, X_test, y_train, y_test = train_test_split(new_X,y,train_size
=0.7, random_state=1)
```

In [170]:
```
from sklearn.preprocessing import StandardScaler
scalar = StandardScaler()
X_train_scaled_new = scalar.fit_transform(X_train)
X_test_scaled_new = scalar.fit_transform(X_test)
```

In [171]:
```
phishing_model_new = LogisticRegression(C=100.0)
```

In [172]:
```
phishing_model_new.fit(X_train_scaled_new,y_train)
```

Out[172]: LogisticRegression(C=100.0)

In [173]:
```
y_pred_new=phishing_model_new.predict(X_test_scaled_new)
y_pred_new
```

Out[173]: array([-1, 1, 1, ..., 1, 1, 1])

In [174]:
```
print(phishing_model_new.score(X_train_scaled_new,y_train))
print(phishing_model_new.score(X_test_scaled_new,y_test))
```

```
0.8484104419746704
0.8501658124811576
```

In [175]:
```
from sklearn.metrics import confusion_matrix, accuracy_score
print(confusion_matrix(y_pred_new,y_test))
print(accuracy_score(y_pred_new, y_test))
```

```
[[ 996    3]
 [ 494 1824]]
0.8501658124811576
```

In [180]:
```
# 85% accuracy score
```

In [176]:
```
#Decision boundry

xx,yy = np.mgrid[-5:5:.01, -5:5:.01]
grid = np.c_[xx.ravel(), yy.ravel()]
probs = phishing_model_new.predict_proba(grid)[:,1].reshape(xx.shape)
print(probs)
```

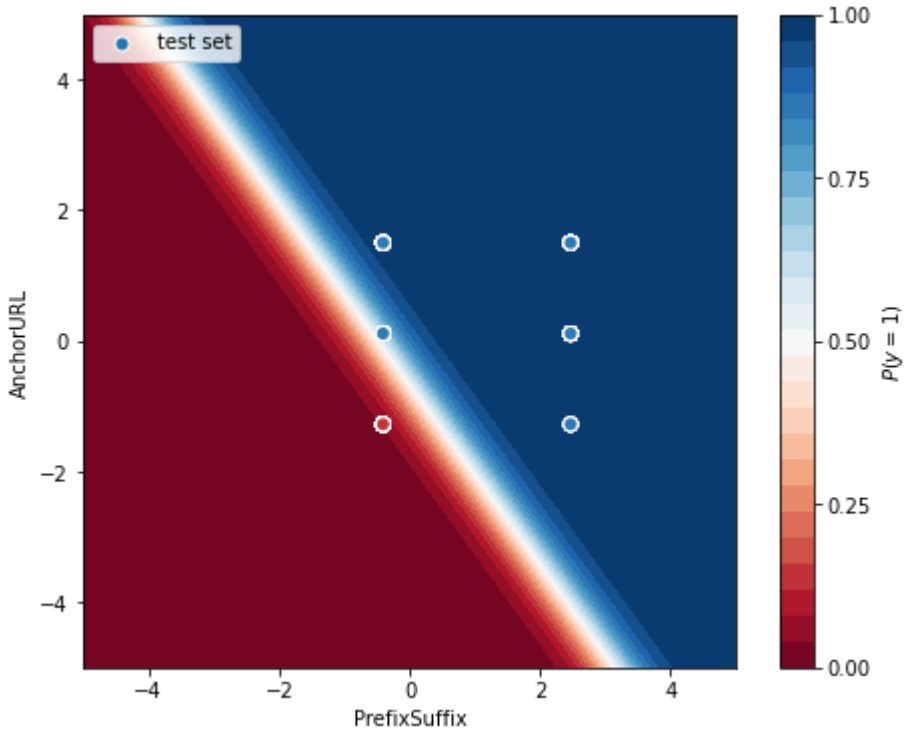
```
[[2.27431060e-13 2.33387481e-13 2.39499901e-13 ... 3.43378239e-02
  3.52054712e-02 3.60942227e-02]
 [2.35691423e-13 2.41864182e-13 2.48198607e-13 ... 3.55406587e-02
  3.64375524e-02 3.73562031e-02]
 [2.44251804e-13 2.50648760e-13 2.57213253e-13 ... 3.67840233e-02
  3.77110672e-02 3.86605371e-02]
 ...
 [9.98433430e-01 9.98473351e-01 9.98512255e-01 ... 1.00000000e+00
  1.00000000e+00 1.00000000e+00]
 [9.98488251e-01 9.98526777e-01 9.98564322e-01 ... 1.00000000e+00
  1.00000000e+00 1.00000000e+00]
 [9.98541157e-01 9.98578336e-01 9.98614569e-01 ... 1.00000000e+00
  1.00000000e+00 1.00000000e+00]]
```

In [188]:
```
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_scaled_new[:,0], X_test_scaled_new[:,1], c=(y_test==
1),s=50,
           cmap = 'RdBu', vmin =-.2, vmax=1.2,
           edgecolor ='white', linewidth=1, label='test set')
ax.set(aspect ='equal',xlim=(-5,5), ylim=(-5,5),
       xlabel='PrefixSuffix', ylabel='AnchorURL')
ax.legend(loc='upper left')

plt.show()
```



In [ ]:

In [ ]: