```
# -*- coding: utf-8 -*-
"""
Created on Tue Dec 25 18:18:55 2018

@author: SRIKANT
"""
#import packages
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np

#load Dataset and classify as features and label
phishingData = pd.read_csv('phishing.txt')
X = phishingData.iloc[:,:-1].values
y = phishingData.iloc[:,30].values

#split features and label into training ang testing data
from sklearn.cross_validation import train_test_split
X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=0.3,random_state=4)

#perform feature scaling
from sklearn.preprocessing import StandardScaler
scalar = StandardScaler()
X_train = scalar.fit_transform (X_train)
X_test = scalar.fit_transform (X_test)

#Logistic Regression Classifier
from sklearn.linear_model import LogisticRegression
LRclassifier = LogisticRegression(C=100,random_state=0)
LRclassifier.fit(X_train,y_train)

LRpredict = LRclassifier.predict(X_test)

#LRC training score
LRclassifier.score(X_train,y_train)

#LRC test score
LRclassifier.score(X_test,y_test)

#confusion matrix for printing count of misclassified samples in the test data prediction
from sklearn.metrics import confusion_matrix
confusionMatrix = confusion_matrix(y_test,LRpredict)



#=============================================================================================


# 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

#split features and label into training ang testing data
from sklearn.cross_validation import train_test_split
X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=0.3,random_state=4)

#perform feature scaling
from sklearn.preprocessing import StandardScaler
scalar = StandardScaler()
X_train = scalar.fit_transform (X_train)
X_test = scalar.fit_transform (X_test)
```

```
#Logistic Regression Classifier
from sklearn.linear_model import LogisticRegression
LRclassifier1 = LogisticRegression(C=100,random_state=0)
LRclassifier1.fit(X_train,y_train)


LRpredict1 = LRclassifier1.predict(X_test)

#LRC training score
LRclassifier1.score(X_train,y_train)

#LRC test score
LRclassifier1.score(X_test,y_test)

#confusion matrix for printing count of misclassified samples in the test data prediction
from sklearn.metrics import confusion_matrix
LRconfusionMatrix1 = confusion_matrix(y_test,LRpredict1)

#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)

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()

#=============================================================================================


# 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

#split features and label into training ang testing data
from sklearn.cross_validation import train_test_split
X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=0.3,random_state=4)

#perform feature scaling
from sklearn.preprocessing import StandardScaler
scalar = StandardScaler()
X_train = scalar.fit_transform (X_train)
X_test = scalar.transform (X_test)

#Logistic Regression Classifier
```

```
from sklearn.linear_model import LogisticRegression
LRclassifier11 = LogisticRegression(C=100,random_state=0)
LRclassifier11.fit(X_train,y_train)


LRpredict11 = LRclassifier11.predict(X_test)

#LRC training score
LRclassifier11.score(X_train,y_train)

#LRC test score
LRclassifier11.score(X_test,y_test)

#confusion matrix for printing count of misclassified samples in the test data prediction
from sklearn.metrics import confusion_matrix
LRconfusionMatrix11 = confusion_matrix(y_test,LRpredict11)

#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)

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()
```