Practical-6

Objective: Design a classifier with k-fold cross validation technique.

Dataset: some toy dataset with x and y coordinates and with classes 0,1.
sample data points are as follows;

	x	y	class
0	0.700335	-0.247068	0.0
1	-3.950019	2.740080	1.0
2	0.150222	-2.157638	1.0
3	-1.672050	-0.941519	1.0
4	2.560483	-1.846577	1.0

Procedure:

1. Load the dataset.
2. Split the dataset into train and test with as 70% ,30%.
3. Now again split the train data set into two;
   one for training(70% of 70)
   Another for cross validation (30% of 70)

Simple cross Validation:

4. For each odd k from 1 to 30 train the k-nearest neighbbour classifier with train dataset and predict on cross validation dataset.
5. We will get the best classifier amoung all those by seeing the metric accuracy.
6. Now we will test the classfier on test data set by selecting the k with got more accuracy.

K Fold cross validation:

7. Create a list of odd numbers from 1 to 50.
8. For each k in the list apply the k nearest neighbour classifier with 10 folds as cross validation.
9. Calculate the mean cross validation score for each k and store it in a list.
10. Obtain the misclassification error from cross validation score.
11. Pick up the best k with least misclassification error.
12. Test the classifier with that best k on test data set.
13. Plot a graph between the k and the and the misclassification error to analyse.

Code:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score
from sklearn.model_selection import cross_val_score
from collections import Counter
from sklearn.metrics import accuracy_score
from sklearn import model_selection



# Part I
# ============================== data preprocessing
==========================================

# define column names
names = ['x', 'y', 'class']

# loading training data
df = pd.read_csv('3.concertriccir2.csv', header=None, names=names)
print(df.head())

# create design matrix X and target vector y
X = np.array(df.iloc[:, 0:2]) # end index is exclusive
y = np.array(df['class']) # showing you two ways of indexing a pandas df




X_1, X_test, y_1, y_test = model_selection.train_test_split(X, y, test_size=0.3, random_state=0)

# split the train data set into cross validation train and cross validation test
X_tr, X_cv, y_tr, y_cv = model_selection.train_test_split(X_1, y_1, test_size=0.3)

for i in range(1,30,2):
    # instantiate learning model (k = 30)
    knn = KNeighborsClassifier(n_neighbors=i)
```

```
    # fitting the model on crossvalidation train
    knn.fit(X_tr, y_tr)

    # predict the response on the crossvalidation train
    pred = knn.predict(X_cv)

    # evaluate CV accuracy
    acc = accuracy_score(y_cv, pred, normalize=True) * float(100)
    print('\nCV accuracy for k = %d is %d%%' % (i, acc))

knn = KNeighborsClassifier(1)
knn.fit(X_tr,y_tr)
pred = knn.predict(X_test)
acc = accuracy_score(y_test, pred, normalize=True) * float(100)
print('\n****Test accuracy for k = 1 is %d%%' % (acc))




# creating odd list of K for KNN
myList = list(range(0,50))
neighbors = list(filter(lambda x: x % 2 != 0, myList))

# empty list that will hold cv scores
cv_scores = []

# perform 10-fold cross validation
for k in neighbors:
    knn = KNeighborsClassifier(n_neighbors=k)
    scores = cross_val_score(knn, X_1, y_1, cv=10, scoring='accuracy')
    cv_scores.append(scores.mean())

# changing to misclassification error
MSE = [1 - x for x in cv_scores]

# determining best k
optimal_k = neighbors[MSE.index(min(MSE))]
print('\nThe optimal number of neighbors is %d.' % optimal_k)

# plot misclassification error vs k
plt.plot(neighbors, MSE)
```

```
for xy in zip(neighbors, np.round(MSE,3)):
    plt.annotate('(%s, %s)' % xy, xy=xy, textcoords='data')

plt.xlabel('Number of Neighbors K')
plt.ylabel('Misclassification Error')
plt.show()

print("the misclassification error for each k value is : ", np.round(MSE,3))
```

Output:

```
CV accuracy for k = 3 is 85%
CV accuracy for k = 5 is 82%
CV accuracy for k = 7 is 80%
CV accuracy for k = 9 is 80%
CV accuracy for k = 11 is 80%
CV accuracy for k = 13 is 79%
CV accuracy for k = 15 is 80%
CV accuracy for k = 17 is 78%
CV accuracy for k = 19 is 79%
CV accuracy for k = 21 is 77%
CV accuracy for k = 23 is 74%
CV accuracy for k = 25 is 74%
CV accuracy for k = 27 is 72%
CV accuracy for k = 29 is 68%
****Test accuracy for k = 1 is 90%
```

```
The optimal number of neighbors is 5.
```