```python
# -*- coding: utf-8 -*-
"""
Created on Wed May  6 13:23:47 2020

@author: mrhaboon
"""

import csv
from sklearn import preprocessing
import numpy as np
from sklearn.linear_model import Perceptron
from sklearn.metrics import confusion_matrix
from itertools import combinations
from sklearn.model_selection import cross_val_score
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
from sklearn.decomposition import PCA
from sklearn.preprocessing import PolynomialFeatures
train1=[]
test1=[]

with open('D_Train1.csv') as csv_file:
    reader=csv.reader(csv_file)
    j=0
    for i in reader:
        if j==0:
            j+=1
            continue
        else:
            train1.append([float(x) for x in i])
    train1=np.array(train1)

j=0
with open('D_Test1.csv') as csv_file:
    reader=csv.reader(csv_file)
    for i in reader:
        if j==0:
            j+=1
            continue
        else:
            test1.append([float(x) for x in i])
    test1=np.array(test1)

#%%

train1_data=train1[:,1:]
label_train=train1[:,0]

test1_data=test1[:,1:]
label_test=test1[:,0]

scaler=preprocessing.StandardScaler().fit(train1_data)
std_train=np.concatenate((label_train[:,np.newaxis],scaler.transform(train1_data)),axis=1)
std_test=np.concatenate((label_test[:,np.newaxis],scaler.transform(test1_data)),axis=1)


normer=preprocessing.MinMaxScaler()
norm_train=np.concatenate((label_train[:,np.newaxis],normer.fit_transform(train1_data)),axis=1)
```

```
norm_test=np.concatenate((label_test[:,np.newaxis],normer.fit_transform(test1_data)),axis=1)
```

```
class find_best_perc:

    def __init__(self,train,test,dim=None):
        self.dim=dim
        self.train=train
        self.train_data=self.train[:,1:]
        self.train_labels=self.train[:,0]
        self.test=test
        self.test_data=self.test[:,1:]
        self.test_labels=self.test[:,0]
        if dim==None:
            self.final_model=Perceptron()
            self.final_model.fit(self.train_data,self.train_labels)
            self.performance(self.train_data,self.test_data)
        elif dim=='perm':
            self.gen_perm()
            self.models={}
            best=((0,0),0)
            for i in self.perm:
                tmp_train=self.dim_transform(self.train_data,i)
                self.models[i]=self.gen_model(tmp_train,self.train_labels)
            for i in self.models:
                if self.models[i][0]>best[0][0]:
                    best=(self.models[i],i)
            self.best=best
            self.final_model=Perceptron()
            new_train=self.dim_transform(self.train_data,best[1])
            new_test=self.dim_transform(self.test_data,best[1])
            self.final_model.fit(new_train,self.train_labels)
            self.performance(new_train,new_test)

        elif dim=='PCA':
            best=((0,0),0)
            for i in range(7):
                tmp=PCA(n_components=i+1)
                tmp.fit(self.train_data)
                tmp_train=tmp.transform(self.test_data)
                current_stat=self.gen_model(tmp_train,self.test_labels)
                if current_stat[0]>best[0][0]:
                    best=(current_stat,i+1)
            self.best=best
            self.final_model=Perceptron()
            tran=PCA(n_components=best[1])
            tran.fit(self.train_data)
            new_train=tran.transform(self.train_data)
            new_test=tran.transform(self.test_data)
            self.final_model.fit(new_train,self.train_labels)
            self.performance(new_train,new_test)


        elif dim=='Fisher':
            best=((0,0),0)
            for i in range(3):
```

```
                tran=LinearDiscriminantAnalysis(n_components=i+1)
                tran.fit(self.train_data,self.train_labels)
                new_train=tran.transform(self.train_data)
                current_stat=self.gen_model(new_train,self.train_labels)
                if current_stat[0]>best[0][0]:
                    best=(current_stat,i+1)
            self.best=best
            self.final_model=Perceptron()
            tran=LinearDiscriminantAnalysis(n_components=best[1])
            tran.fit(self.train_data,self.train_labels)
            new_train=tran.transform(self.train_data)
            new_test=tran.transform(self.test_data)
            self.final_model.fit(new_train,self.train_labels)
            self.performance(new_train,new_test)

        elif dim=='poly':
            best=((0,0),0)
            for i in range(8):
                poly=PolynomialFeatures(i+1)
                new_train=poly.fit_transform(self.train_data)
                current_stat=self.gen_model(new_train,self.train_labels)
                if current_stat[0]>best[0][0]:
                    best=(current_stat,i+1)
            self.best=best
            self.final_model=Perceptron()
            poly=PolynomialFeatures(best[1])
            new_train=poly.fit_transform(self.train_data)
            new_test=poly.fit_transform(self.test_data)
            self.final_model.fit(new_train,self.train_labels)
            self.performance(new_train,new_test)

    def performance(self,train,test):
        self.train_conf=confusion_matrix(self.train_labels,self.final_model.predict(train))
        self.test_conf=confusion_matrix(self.test_labels,self.final_model.predict(test))
        if self.dim==None:
            self.cross_acc=self.final_model.score(train,self.train_labels)
        else:
            self.cross_acc=self.best
        self.test_acc=self.final_model.score(test,self.test_labels)
#         self.train_acc=
#         self.test_acc=

    def gen_perm(self):
        self.perm=[]
        tmp=[]
        data_length=len(self.train_data[0])
        for i in range(data_length):
            tmp.append(i+1)
        for i in range(data_length):
            for i in combinations(tmp,i+1):
                self.perm.append(i)

    def dim_transform(self,data,dim):
        result=[]
        for i in data:
            tmp_data=[]
            for j in dim:
```

```
                tmp_data.append(i[j-1])
            result.append(tmp_data)
        return result

    def gen_model(self,data,labels):
        scores=[]
        for i in range(30):
            test_model=Perceptron()
            scores.extend(list(cross_val_score(test_model,data,labels,cv=5)))
        scores=np.array(scores)
        return (scores.mean(),scores.std()**2)

#%%

data_sets=[[train1,test1],[std_train,std_test],[norm_train,norm_test]]

#data_sets=[[train1,test1]]

dim_choice=[None,'Fisher','PCA','poly','perm']
file1=open('Perc_results.txt','w')

i=0
for k in data_sets:
    if i==0:
        file1.write('no std:\n')
    elif i==1:
        file1.write('std:\n')
    else:
         file1.write('norm:\n')
    for j in dim_choice:
        print('we movin')
        tmp=find_best_perc(k[0],k[1],j)
        if j==None:
            j='none'
        file1.write('---'+j+':\n')

        file1.write('Cross Validation Accuracy:'+str(tmp.cross_acc)+'\n')
        file1.write('train confusion matrix:'+str(tmp.train_conf)+'\n')
        file1.write('test set acc: '+str(tmp.test_acc)+'\n')
        file1.write('test confusion'+str(tmp.test_conf)+'\n')

    file1.write('------------------------------------------------------------'+'\n')
    i+=1

file1.close()
```