**IT 542: Pattern Recognition and Machine Learning**

**Assignment 3**

ID:201811024

Name- Rajat Kumar

1. **Check suitability of naïve-bayes classifier from Assignment-2 on IRIS data from UCI Machine Learning Repository. Consider 40 samples from each class as training data, use remaining 10 from each class as testing data. Perform 3-fold cross validation.**

**Code:**

from sklearn import datasets

from sklearn.model\_selection import KFold

iris = datasets.load\_iris()

from sklearn.naive\_bayes import GaussianNB

gnb = GaussianNB()

y\_pred = gnb.fit(iris.data, iris.target).predict(iris.data)

print("Number of mislabeled points out of a total %d points : %d" %(iris.data.shape[0],(iris.target != y\_pred).sum()))

Output:

Number of mislabeled points out of a total 150 points : 6

# 3 fold validation

from sklearn.model\_selection import KFold # import KFold

X =iris.data # create an array

Y = iris.target # Create another array

kf = KFold(n\_splits=3,shuffle=True) # Define the split - into 2 folds

kf.get\_n\_splits(X) # returns the number of splitting iterations in the cross-validator

print(kf)

for train\_index, test\_index in kf.split(X):

X\_train=X[train\_index]

X\_test=X[test\_index]

Y\_train=Y[train\_index]

Y\_test=Y[test\_index]

y\_pred = gnb.fit(X\_train,Y\_train).predict(X\_test)

print("Number of mislabeled points out of a total %d points : %d" %(X\_test.shape[0],(Y\_test != y\_pred).sum()))

num=(Y\_test == y\_pred).sum()

denom=X\_test.shape[0]

print('Accuracy:',(num/denom)\*100, '%')

**OUTPUT:**

Number of mislabeled points out of a total 50 points : 1

Accuracy: 98.0 %

Number of mislabeled points out of a total 50 points : 1

Accuracy: 98.0 %

Number of mislabeled points out of a total 50 points : 4

Accuracy: 92.0 %

**(2) Implement k-NN classifier and use it for IRIS data with k = 1, 3, 5 and 11. Perform 3-fold validation.**

import numpy as np

import pandas as pd

# Importing the dataset

dataset = pd.read\_csv('iris.csv')

feature\_columns = ['SepalLengthCm', 'SepalWidthCm', 'PetalLengthCm','PetalWidthCm']

X = dataset[feature\_columns].values

y = dataset['Species'].values

from sklearn.preprocessing import LabelEncoder

le = LabelEncoder()

y = le.fit\_transform(y)

from sklearn.cross\_validation import train\_test\_split

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size = 0.2, random\_state = 0)

import matplotlib.pyplot as plt

import seaborn as sns

%matplotlib inline

from pandas.plotting import parallel\_coordinates

plt.figure(figsize=(15,10))

parallel\_coordinates(dataset.drop("Id", axis=1), "Species")

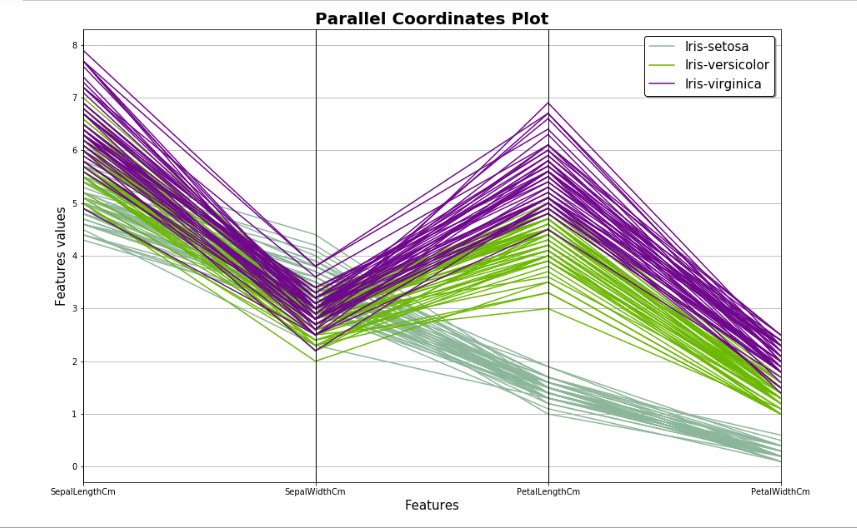
plt.title('Parallel Coordinates Plot', fontsize=20, fontweight='bold')

plt.xlabel('Features', fontsize=15)

plt.ylabel('Features values', fontsize=15)

plt.legend(loc=1, prop={'size': 15}, frameon=True,shadow=True, facecolor="white", edgecolor="black")

plt.show()



from pandas.plotting import andrews\_curves

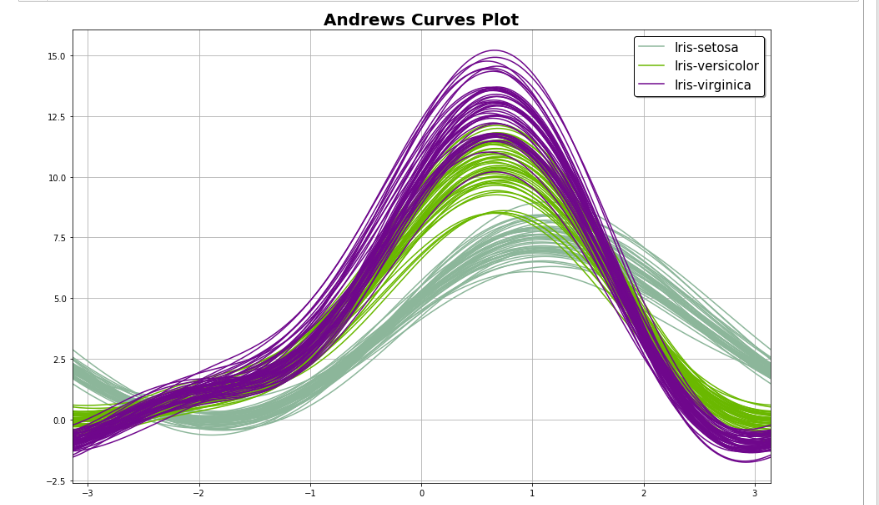
plt.figure(figsize=(15,10))

andrews\_curves(dataset.drop("Id", axis=1), "Species")

plt.title('Andrews Curves Plot', fontsize=20, fontweight='bold')

plt.legend(loc=1, prop={'size': 15}, frameon=True,shadow=True, facecolor="white", edgecolor="black")

plt.show()



# Fitting clasifier to the Training set

# Loading libraries

from sklearn.neighbors import KNeighborsClassifier

from sklearn.metrics import confusion\_matrix, accuracy\_score

from sklearn.model\_selection import cross\_val\_score

# Instantiate learning model (k = 3)

classifier = KNeighborsClassifier(n\_neighbors=3) # 1,3,5,11

# Fitting the model

classifier.fit(X\_train, y\_train)

# Predicting the Test set results

y\_pred = classifier.predict(X\_test)

cm = confusion\_matrix(y\_test, y\_pred)

Cm

accuracy = accuracy\_score(y\_test, y\_pred)\*100

print('Accuracy of our model is equal ' + str(round(accuracy, 2)) + ' %.')

Output:

**Accuracy of our model is equal 96.67 %.**

# creating list of K for KNN

k\_list = list([1,3,5,11])

# creating list of cv scores

cv\_scores = []

# perform 3-fold cross validation

for k in k\_list:

knn = KNeighborsClassifier(n\_neighbors=k)

scores = cross\_val\_score(knn, X\_train, y\_train, cv=3, scoring='accuracy')

cv\_scores.append(scores.mean())

# changing to misclassification error

MSE = [1 - x for x in cv\_scores]

plt.figure()

plt.figure(figsize=(15,10))

plt.title('The optimal number of neighbors', fontsize=20, fontweight='bold')

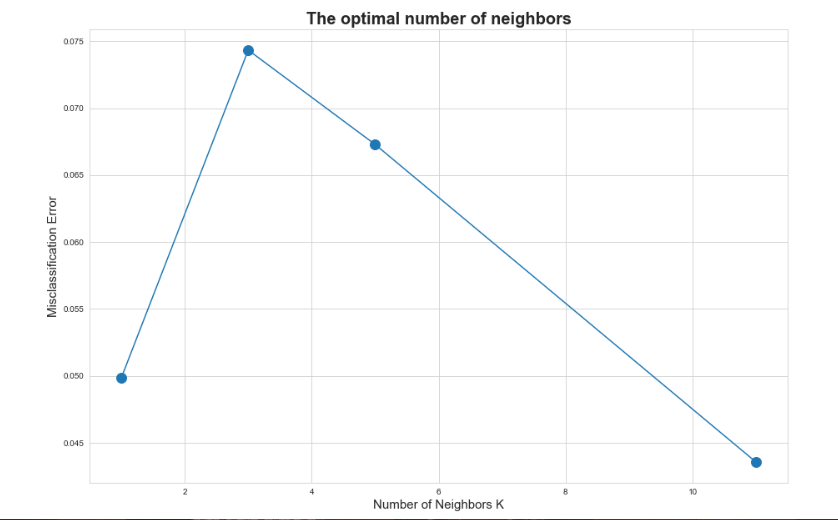
plt.xlabel('Number of Neighbors K', fontsize=15)

plt.ylabel('Misclassification Error', fontsize=15)

sns.set\_style("whitegrid")

plt.plot(k\_list, MSE,marker='o',markersize=12)

plt.show()



# finding best k

best\_k = k\_list[MSE.index(min(MSE))]

print("The optimal number of neighbors is %d." % best\_k)

**OUTPUT:**

The optimal number of neighbors is 11.