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

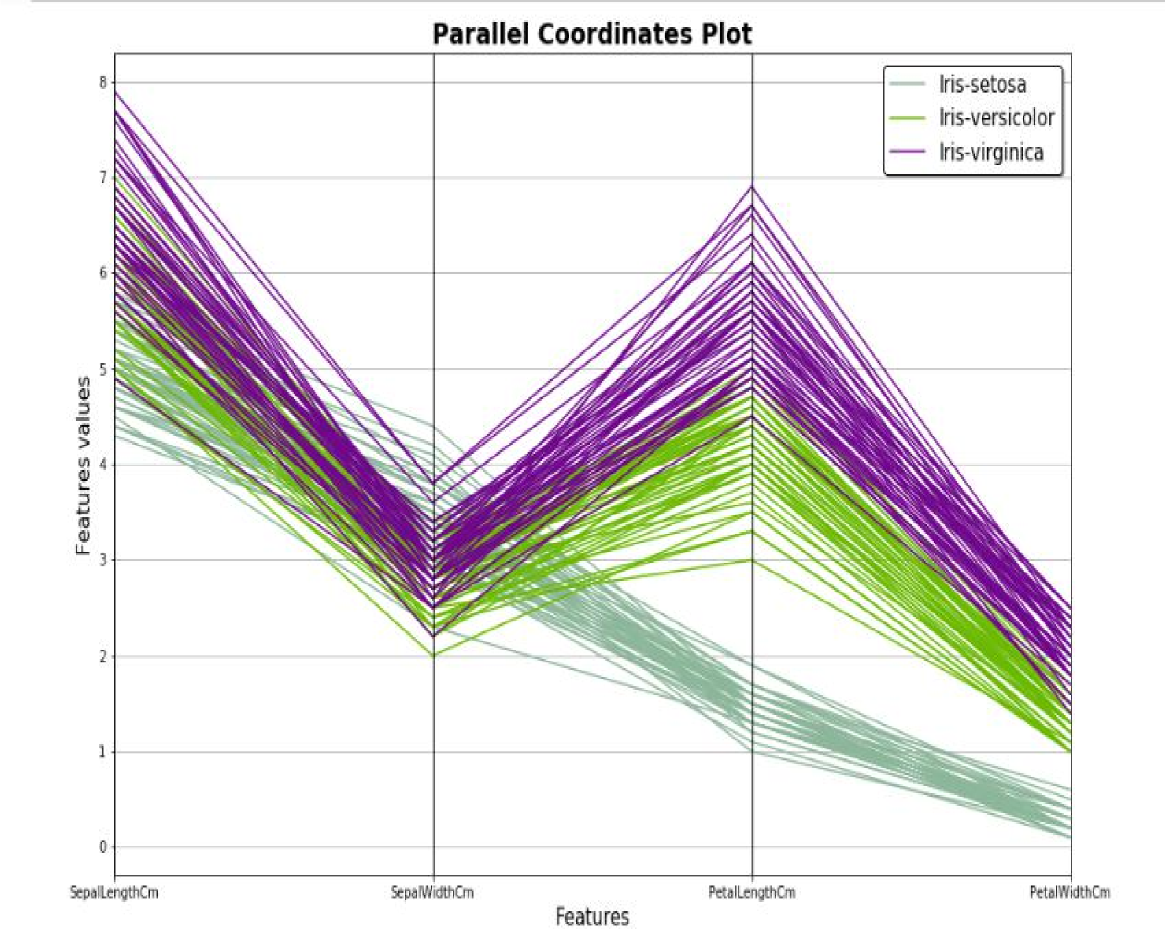
dataset = pd.read_csv('iris.csv')

feature_columns = ['SepalLengthCm', 'SepalWidthCm', 'PetalLengthCm','PetalWidthCm']

X = dataset[feature_columns].values

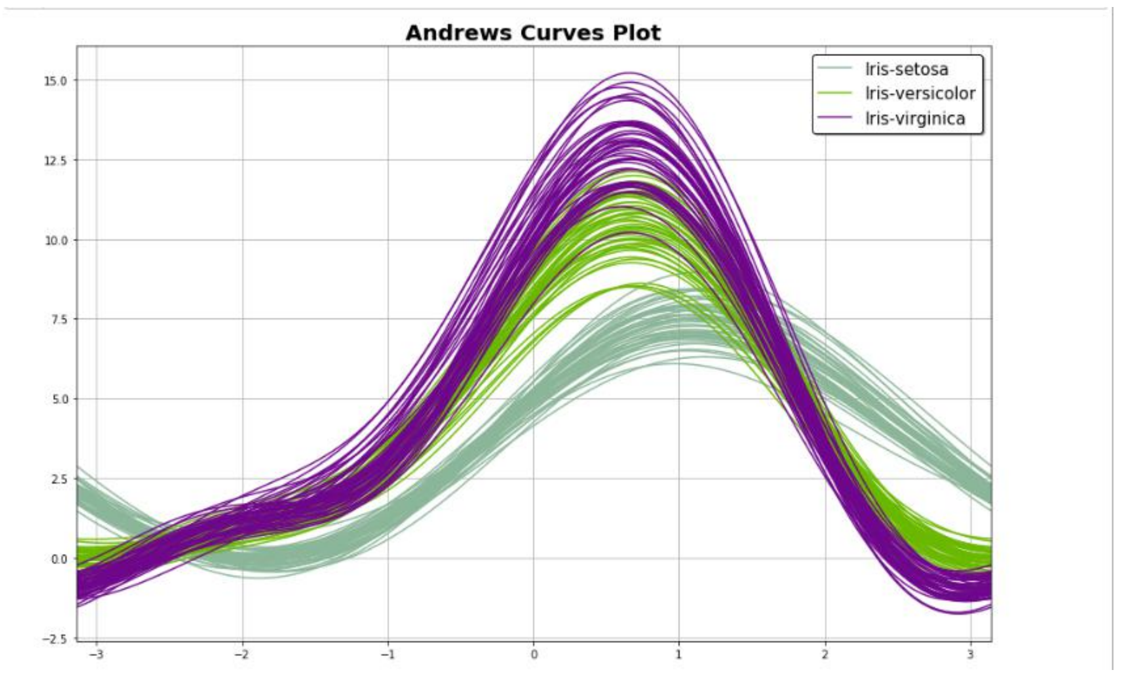
```
y = dataset['Species'].values

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
```

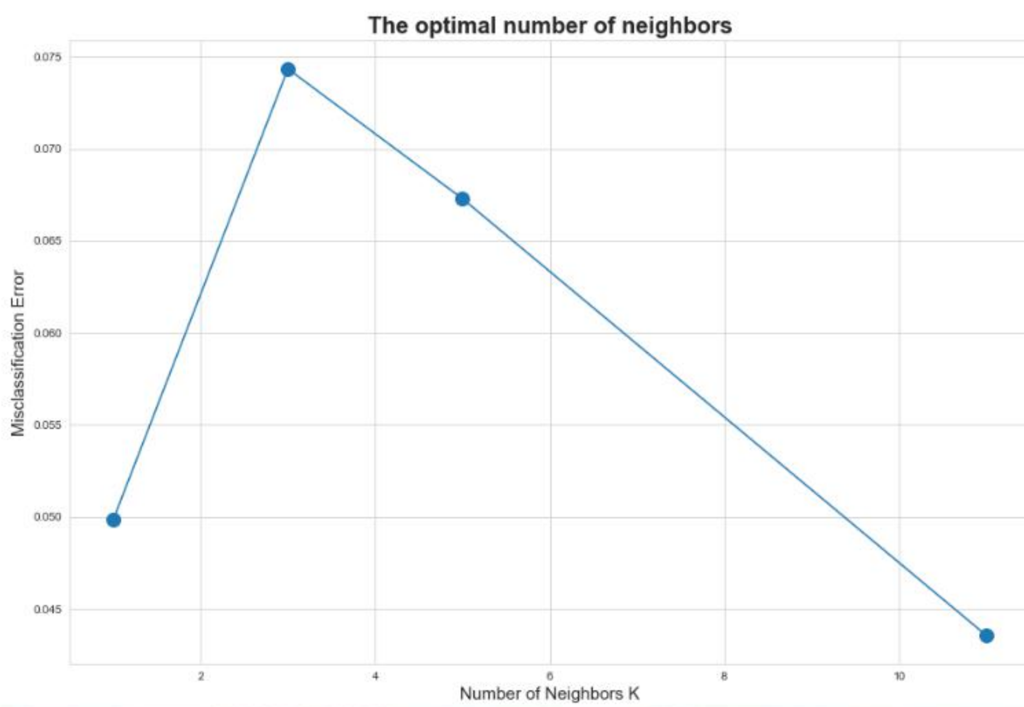
```
from sklearn.model_selection import
cross_val_score


# 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.