VELLORE INSTITUTE OF TECHNOLOGY

CSE4020 Machine Learning

Lab Assessment - 5

17BCE0581

SATYAM SINGH CHAUHAN

KNN Classifier

Importing required Libraries

In [147]:

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
```

Loading Data

In [129]:

```
#Import scikit-learn dataset library
from sklearn import datasets

#Load dataset
wine = datasets.load_wine()
```

Exploring Data

In [130]:

```
# print the names of the features
print(wine.feature_names)
```

```
['alcohol', 'malic_acid', 'ash', 'alcalinity_of_ash', 'magnesium',
'total_phenols', 'flavanoids', 'nonflavanoid_phenols', 'proanthocyan
ins', 'color_intensity', 'hue', 'od280/od315_of_diluted_wines', 'pro
line']
```

In [131]:

```
# print the label species(class_0, class_1, class_2)
print(wine.target_names)
```

```
['class_0' 'class_1' 'class_2']
```

Observing Few Rows of Dataset

In [132]:

```
# print the wine data
print(wine.data[0:5])
```

```
[[1.423e+01 1.710e+00 2.430e+00 1.560e+01 1.270e+02 2.800e+00 3.060e
+00
  2.800e-01 2.290e+00 5.640e+00 1.040e+00 3.920e+00 1.065e+03]
 [1.320e+01 1.780e+00 2.140e+00 1.120e+01 1.000e+02 2.650e+00 2.760e
+00
  2.600e-01 1.280e+00 4.380e+00 1.050e+00 3.400e+00 1.050e+03]
 [1.316e+01 2.360e+00 2.670e+00 1.860e+01 1.010e+02 2.800e+00 3.240e
+00
  3.000e-01 2.810e+00 5.680e+00 1.030e+00 3.170e+00 1.185e+03]
 [1.437e+01 1.950e+00 2.500e+00 1.680e+01 1.130e+02 3.850e+00 3.490e
+00
  2.400e-01 2.180e+00 7.800e+00 8.600e-01 3.450e+00 1.480e+03]
 [1.324e+01 2.590e+00 2.870e+00 2.100e+01 1.180e+02 2.800e+00 2.690e
+00
  3.900e-01 1.820e+00 4.320e+00 1.040e+00 2.930e+00 7.350e+02]]
```

Records of the target set.

In [133]:

```
# print the wine labels (0:Class_0, 1:Class_1, 2:Class_3)
print(wine.target)
```

```
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
2 2 2
 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2]
```

In [134]:

```
# print data(feature)shape
print(wine.data.shape)
```

```
(178, 13)
```

In [135]:

```
# print target(or label)shape
print(wine.target.shape)
```

```
(178,)
```

Splitting Data

In [136]:

```
# Import train_test_split function
from sklearn.model_selection import train_test_split

# Split dataset into training set and test set
# 70% training and 30% test
X_train, X_test, y_train, y_test = train_test_split(wine.data, wine.target, test
_size=0.3)
```

Generating Model for K=5

In [137]:

```
#Import knearest neighbors Classifier model
from sklearn.neighbors import KNeighborsClassifier

#Create KNN Classifier
knn = KNeighborsClassifier(n_neighbors=5)

#Train the model using the training sets
knn.fit(X_train, y_train)

#Predict the response for test dataset
y_pred = knn.predict(X_test)
```

Evaluating Model for K=5

In [138]:

```
#Import scikit-learn metrics module for accuracy calculation
from sklearn import metrics

# Model Accuracy, how often is the classifier correct?
print("Accuracy:",metrics.accuracy_score(y_test, y_pred))
```

```
Accuracy: 0.7222222222222222
```

In [139]:

```
#Import classification_report, confusion_matrix package from scikit-learn metric
s module for Evaluation
from sklearn.metrics import classification_report, confusion_matrix
print(confusion_matrix(y_test, y_pred))
print(classification_report(y_test, y_pred))
```

```
[[18  0  2]
 [ 2 12  6]
 [ 0  5  9]]
              precision    recall  f1-score   support

           0       0.90      0.90      0.90        20
           1       0.71      0.60      0.65        20
           2       0.53      0.64      0.58        14

   micro avg       0.72      0.72      0.72        54
   macro avg       0.71      0.71      0.71        54
weighted avg       0.73      0.72      0.72        54
```

Re-generating Model for K=7

In [140]:

```
#Import knearest neighbors Classifier model
from sklearn.neighbors import KNeighborsClassifier

#Create KNN Classifier
knn = KNeighborsClassifier(n_neighbors=7)

#Train the model using the training sets
knn.fit(X_train, y_train)

#Predict the response for test dataset
y_pred = knn.predict(X_test)
```

Evaluating Model for k=7

In [141]:

```
#Import scikit-learn metrics module for accuracy calculation
from sklearn import metrics

# Model Accuracy, how often is the classifier correct?
print("Accuracy:",metrics.accuracy_score(y_test, y_pred))
```

```
Accuracy: 0.7407407407407407
```

In [142]:

```
#Import classification_report, confusion_matrix package from scikit-learn metric
s module for Evaluation
from sklearn.metrics import classification_report, confusion_matrix
print(confusion_matrix(y_test, y_pred))
print(classification_report(y_test, y_pred))
```

```
[[18  0  2]
 [ 2 13  5]
 [ 1  4  9]]
              precision    recall  f1-score   support

           0       0.86      0.90      0.88        20
           1       0.76      0.65      0.70        20
           2       0.56      0.64      0.60        14

   micro avg       0.74      0.74      0.74        54
   macro avg       0.73      0.73      0.73        54
weighted avg       0.75      0.74      0.74        54
```

Comparing Error Rate with the K Value

In [146]:

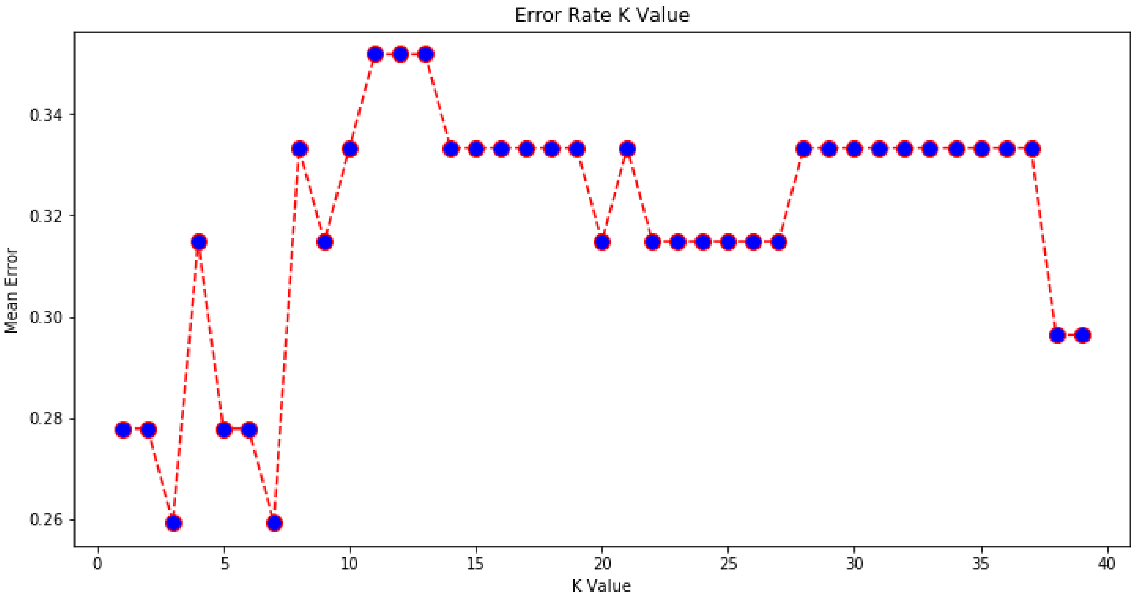
```
error = []
# Calculating error for K values between 1 and 40
for i in range(1, 40):
    #Create KNN Classifier
    knn = KNeighborsClassifier(n_neighbors=i)

    #Train the model using the training sets
    knn.fit(X_train, y_train)

    #Predict the response for test dataset
    pred_i = knn.predict(X_test)

    error.append(np.mean(pred_i != y_test))
```

In [145]:

```
plt.figure(figsize=(12, 6))
plt.plot(range(1, 40), error, color='red', linestyle='dashed', marker='o',
         markerfacecolor='blue', markersize=10)
plt.title('Error Rate K Value')
plt.xlabel('K Value')
plt.ylabel('Mean Error')
```

Out[145]:

```
Text(0, 0.5, 'Mean Error')
```



Conclusion

From the Error Graph we can observe that Error rate is least when we choose k at 3 and 7.

**The results we got we observed that
for k = 5 our model was able to classify 54 records with accuracy of 72%
while when we set the k = 7 our model was able to classify 54 records with accuracy of 74%.**

Results shown by our KNN classifier make us believe that it did a good work with nice Accuracy.

In [ ]: