6) KNN Classifier

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

```
import pandas as pd
import numpy as np

df=pd.read_csv('/content/drive/MyDrive/HCP/Classified Data',index_col=0)
print(df.head())

from sklearn.model_selection import train_test_split

X=df.drop('TARGET CLASS',axis=1)
y=df['TARGET CLASS']

X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.2,random_state=100)
```

```
        WTT       PTI       EQW       SBI       LQE       QWG       FDJ  \
0 0.913917 1.162073 0.567946 0.755464 0.780862 0.352608 0.759697
1  0.635632  1.003722  0.535342  0.825645  0.924109  0.648450  0.675334
2  0.721360  1.201493  0.921990  0.855595  1.526629  0.720781  1.626351
3  1.234204  1.386726  0.653046  0.825624  1.142504  0.875128  1.409708
4  1.279491  0.949750  0.627280  0.668976  1.232537  0.703727  1.115596

        PJF       HQE       NXJ  TARGET CLASS
0  0.643798  0.879422  1.231409             1
1  1.013546  0.621552  1.492702             0
2  1.154483  0.957877  1.285597             0
3  1.380003  1.522692  1.153093             1
4  0.646691  1.463812  1.419167             1
```

```
from sklearn.neighbors import KNeighborsClassifier
knn=KNeighborsClassifier(n_neighbors=7)
knn.fit(X_train,y_train)
pred=knn.predict(X_test)
```

----Arguments----

KNeighborsClassifier(

```
n_neighbors=5,

weights='uniform'( ----'uniform' or 'callable'),

algorithm='auto'({'auto', 'ball_tree', 'kd_tree', 'brute'},Algorithm used to compute the nearest neighbors),

leaf_size=30,

p=2( ----Power parameter. When p = 1, this is
equivalent to using manhattan_distance (l1), and euclidean_distance (l2) for p = 2),

metric='minkowski',

metric_params=None(---- the distance metric to use for the tree),

n_jobs=None,
```

)

```
from sklearn.metrics import classification_report,confusion_matrix
```

```

print(confusion_matrix(y_test,pred))
print(classification_report(y_test,pred))
```

```
[[98 12]
 [ 4 86]]
              precision    recall  f1-score   support
```

```
           0       0.96      0.89      0.92       110
           1       0.88      0.96      0.91        90

    accuracy                           0.92       200
   macro avg       0.92      0.92      0.92       200
weighted avg       0.92      0.92      0.92       200
```

KNN using Standard Scaler

1) Split the Dataset

```
#----Here we are not knowing that what are the features so how to group the data points?
#---If the values of some features are higher than it is required to do the feature scaling otherwise such features
#---it will have much effect on the distance between the features

import pandas as pd
 import numpy as np

df=pd.read_csv('/content/drive/MyDrive/HCP/Classified Data',index_col=0)
print(df.head())

 from sklearn.model_selection import train_test_split

X=df.drop('TARGET CLASS',axis=1)
y=df['TARGET CLASS']

X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.2,random_state=100)
```

```
        WTT       PTI       EQW       SBI       LQE       QWG       FDJ  \
0 0.913917 1.162073 0.567946 0.755464 0.780862 0.352608 0.759697
1  0.635632  1.003722  0.535342  0.825645  0.924109  0.648450  0.675334
2  0.721360  1.201493  0.921990  0.855595  1.526629  0.720781  1.626351
3  1.234204  1.386726  0.653046  0.825624  1.142504  0.875128  1.409708
4  1.279491  0.949750  0.627280  0.668976  1.232537  0.703727  1.115596

        PJF       HQE       NXJ  TARGET CLASS
0  0.643798  0.879422  1.231409             1
1  1.013546  0.621552  1.492702             0
2  1.154483  0.957877  1.285597             0
3  1.380003  1.522692  1.153093             1
4  0.646691  1.463812  1.419167             1
```

2) Scale the Splitted dataset

1st Fit the data

2nd Transform the data

```
 from sklearn.preprocessing import StandardScaler

scaler=StandardScaler()
scaler.fit(X_train) #--- It will drop the target class as we dont want to scale the labels
```

```
StandardScaler()
```

```
scaled_features_X_train=scaler.transform(X_train)
scaled_features_X_test=scaler.transform(X_test)
```

3) Apply KNN Model on the scaled dataset

```
 from sklearn.neighbors import KNeighborsClassifier

knn=KNeighborsClassifier(n_neighbors=1)  #---means k=1
knn.fit(scaled_features_X_train,y_train)
pred_1=knn.predict(scaled_features_X_test)
pred
```

```
array([0, 1, 1, 0, 0, 0, 1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 0,
       1, 0, 1, 0, 1, 0, 1, 1, 0, 1, 1, 0, 1, 1, 1, 0, 0, 1, 1, 0, 1, 1,
```

```
1, 1, 0, 1, 1, 0, 0, 0, 1, 1, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 1, 0,
0, 0, 1, 0, 1, 1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 1, 1, 0, 1, 0, 1, 0,
0, 0, 0, 1, 0, 1, 0, 0, 1, 0, 0, 1, 0, 1, 1, 0, 0, 0, 1, 0, 0, 1,
1, 1, 0, 0, 1, 0, 0, 1, 1, 1, 0, 1, 0, 1, 0, 0, 1, 1, 1, 1, 1, 0,
0, 0, 1, 1, 0, 1, 1, 1, 0, 1, 0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0,
1, 0, 1, 1, 0, 1, 1, 0, 0, 1, 0, 1, 1, 0, 1, 1, 1, 0, 1, 0, 0, 1,
0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 1, 0, 1, 0, 0, 1, 0, 0, 0,
0, 1])
```

4) Find the Classification Report for KNN =1 using scaled Data

```
from sklearn.metrics import classification_report,confusion_matrix

print(confusion_matrix(y_test,pred_1))
print(classification_report(y_test,pred_1))

#---Here you can see that the number of Misclassifications(17) in scaled dataset is more as compared to unscaled dat
```

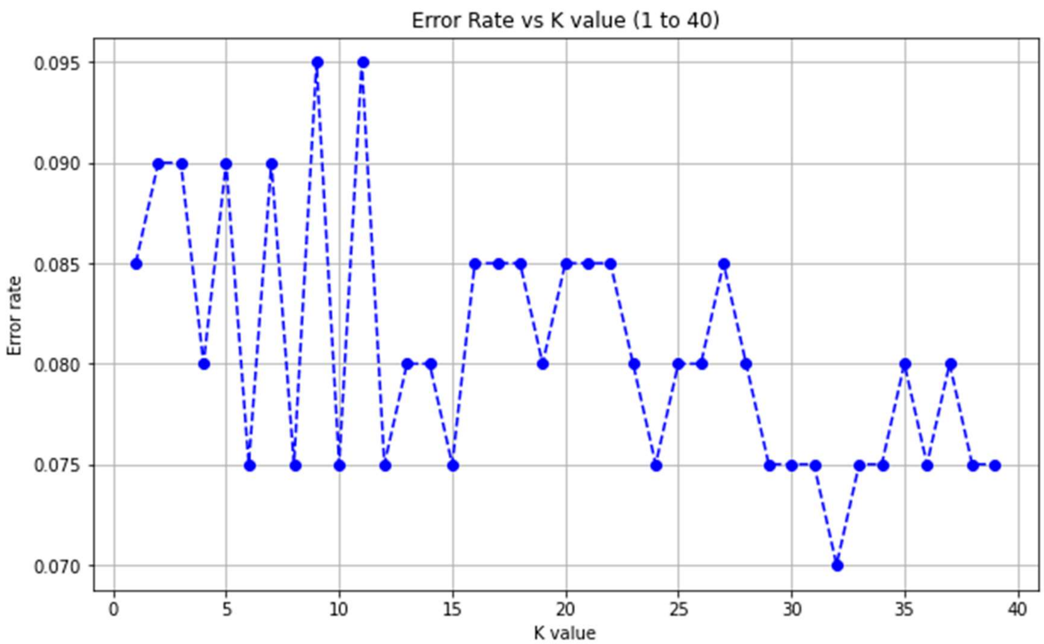
```
[[98 12]
 [ 5 85]]
              precision    recall  f1-score   support

           0       0.95      0.89      0.92       110
           1       0.88      0.94      0.91        90

    accuracy                           0.92       200
   macro avg       0.91      0.92      0.91       200
weighted avg       0.92      0.92      0.92       200
```

'Elbow' method to find correct value of 'k'

```
#----Use elbow method to choose correct value of k
#----Use the model with different values of 'k' and plot the error rate
#---and observe which one has minimum error rate

error_rate=[]  #---empty list

for i in range(1,40):
    knn=KNeighborsClassifier(n_neighbors=i)
    knn.fit(scaled_features_X_train,y_train)
    pred_i=knn.predict(scaled_features_X_test)
    error_rate.append(np.mean(pred_i != y_test))
    #---taking the mean of all prediction and actual labels which are not equal


print(error_rate)
```

```
[0.085, 0.09, 0.09, 0.08, 0.09, 0.075, 0.09, 0.075, 0.095, 0.075, 0.095, 0.075, 0.08, 0.08, 0.075, 0.085, 0.085, 0.085, 0.08, 0.085
```

```
import matplotlib.pyplot as plt

plt.figure(figsize=(10,6))
plt.plot(range(1,40),error_rate,color='blue',linestyle='--',marker='o')
plt.title('Error Rate vs K value (1 to 40)')
plt.xlabel('K value')
plt.ylabel('Error rate')
plt.grid()
```




```
nn=KNeighborsClassifier(n_neighbors=11)
knn.fit(scaled_features_X_train,y_train)
pred_28=knn.predict(scaled_features_X_test)

print(confusion_matrix(y_test,pred_28))
print('\n')
print(classification_report(y_test,pred_28))

#---Compare the confusion matrix for k=1 and for k=28, it has better classsification
#---Misclassifications without Scaled dataset : 20
#---Misclassifications with Scaled dataset :17
```

```
[[95 15]
 [ 4 86]]


              precision    recall  f1-score   support

           0       0.96      0.86      0.91       110
           1       0.85      0.96      0.90        90

    accuracy                           0.91       200
   macro avg       0.91      0.91      0.90       200
 weighted avg       0.91      0.91      0.91
 200
```
