S.CHANDU AP22122040001

```
import math
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn import datasets


from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
from sklearn.metrics import confusion_matrix
from collections import Counter
from sklearn.preprocessing import Normalizer
from sklearn.metrics import accuracy_score

iris = datasets.load_iris() # loading iris data
iris_df = pd.DataFrame(data= np.c_[iris['data'], iris['target']], columns= iris['feature_r
iris_df.head()
```

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	target
0	5.1	3.5	1.4	0.2	0.0
1	4.9	3.0	1.4	0.2	0.0
2	4.7	3.2	1.3	0.2	0.0
3	4.6	3.1	1.5	0.2	0.0
4	5.0	3.6	1.4	0.2	0.0

Double-click (or enter) to edit

```
x= iris_df.iloc[:, :-1]
y= iris_df.iloc[:, -1]

x.tail()
```

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)
145	6.7	3.0	5.2	2.3
146	6.3	2.5	5.0	1.9
147	6.5	3.0	5.2	2.0
148	6.2	3.4	5.4	2.3
149	5.9	3.0	5.1	1.8

```
y.head()
```

```
0    0.0
1    0.0
2    0.0
3    0.0
4    0.0
Name: target, dtype: float64
```

```
y.tail()
```

```
145    2.0
146    2.0
147    2.0
148    2.0
149    2.0
Name: target, dtype: float64
```

```
x_train, x_test, y_train, y_test= train_test_split(x, y,shuffle= True, test_size= 0.3, rar
x_train= np.asarray(x_train)
y_train= np.asarray(y_train)
x_test= np.asarray(x_test)
y_test= np.asarray(y_test)
print(f'training set size: {x_train.shape[0]} samples \ntest set size: {x_test.shape[0]} s
```

```
training set size: 105 samples
test set size: 45 samples
```

```
# Normalizing data
scaler= Normalizer().fit(x_train) # the scaler is fitted to the training set
normalized_x_train= scaler.transform(x_train) # the scaler is applied to the training set
normalized_x_test= scaler.transform(x_test) # the scaler is applied to the test set
```

```
# calculating euclidean distance
def euc_Dist(x_train, x_test_data):

  dist= []
  for row in range(len(x_train)):
    curr_train= x_train[row]
    curr_dist= 0
    for col in range(len(curr_train)):
      curr_dist += (curr_train[col] - x_test_data[col]) **2
    curr_dist= np.sqrt(curr_dist)
    dist.append(curr_dist)
  distances= pd.DataFrame(data=dist,columns=['dist'])
  return distances
```

```
# finding nearest neighbours
def nearest_neighbors(dist_data, K):
```

```
  neighbour_df= dist_data.sort_values(by=['dist'], axis=0)
  neighbour_df= neighbour_df[:K]
  return neighbour_df

# votes function to calculate majority voting
def votes(neighbour_df, y_train):
  count= Counter(y_train[neighbour_df.index])
  y_pred= count.most_common()[0][0] # Majority Voting
  return y_pred


# KNN for detecting K nearest neighbours for the test sample
def KNN(x_train, y_train, x_test, K):
  y_predicted=[]
  for x_test_point in x_test:
      distance_point = euc_Dist(x_train, x_test_point)
      df_nearest_point= nearest_neighbors(distance_point, K)
      y_pred_point = votes(df_nearest_point, y_train)
      y_predicted.append(y_pred_point)
  return y_predicted


K=int(input("Enter k value: "))
y_pred_final= KNN(x_train, y_train, x_test, K)
print(y_pred_final)
```

```
Enter k value: 4
[1.0, 0.0, 2.0, 1.0, 1.0, 0.0, 1.0, 2.0, 1.0, 1.0, 2.0, 0.0, 0.0, 0.0, 0.0, 1.0, 2.0
```

```
print(len(y_pred_final))
```

```
45
```

```
accuracy_final = accuracy_score(y_test, y_pred_final)
print("The test accuracy of K-NN is: ", accuracy_final)
```

```
The test accuracy of K-NN is:  1.0
```

```
# confusion matrix
cm = confusion_matrix(y_test, y_pred_final,normalize='true')
print("Confusion matrix:\n",cm)
```

```
Confusion matrix:
 [[1.         0.         0.        ]
 [0.         0.92307692 0.07692308]
 [0.         0.15384615 0.84615385]]
```

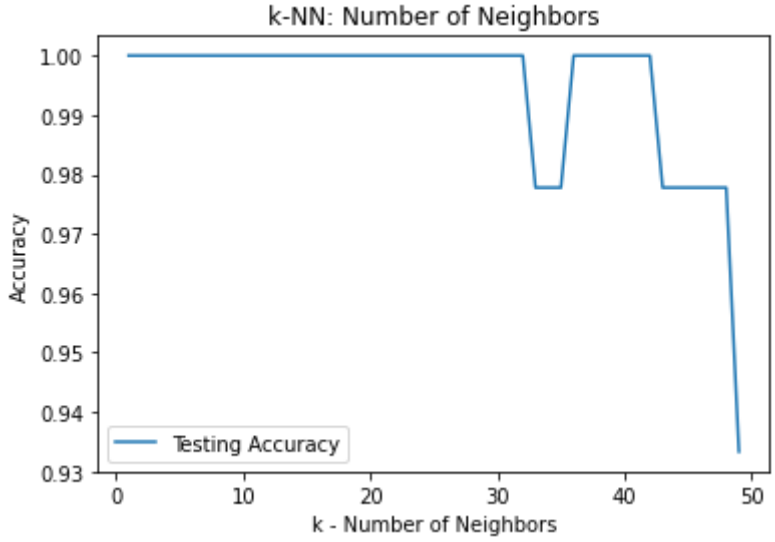
```
# Accuracies for varying k values
accuracies = []
iter = 50
```

```
for i in range(1,iter):
  K=i
  y_pred_final= KNN(x_train, y_train, x_test, K)
  accuracy_final = accuracy_score(y_test, y_pred_final)
  accuracies.append(accuracy_final)

print("The test accuracy of K-NN is: ", accuracies)
print("Maximum accuracy obtained: ", max(accuracies))
print("Minimum accuracy obtained: ", min(accuracies))
```

```
The test accuracy of K-NN is:  [1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.
Maximum accuracy obtained:  1.0
Minimum accuracy obtained:  0.9333333333333333
```

```
no_neighbors = np.arange(1, iter)
for i, k in enumerate(no_neighbors):
  knn = KNN(x_train, y_train, x_test, k)
plt.title('k-NN: Number of Neighbors')
plt.plot(no_neighbors, accuracies, label = 'Testing Accuracy')
plt.legend()
plt.xlabel('k - Number of Neighbors')
plt.ylabel('Accuracy')
plt.show()
```



Colab paid products - Cancel contracts here

✓ 3s completed at 2:07 PM