Assignment 1

Advance machine learning

Step 1: Import Libraries

- For this task, below are the required libraries to accomplish and each library has it own importance.

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.datasets import make_blobs
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score
```

- numpy is an linear algebra library which is useful for handling vectors which we are sending our input as vector format.
- Matplotlib is for visualization, to plot our results.
- sklearn is an machine learning libraries to train our model, also we have some methods use for train_test_split and to compute metrics

Do a 80-20 split of the data

Step 2: Considering Sample data using make_blobs and divided the samples according to the given requirement that is 80 percent train size and 20 per test size.

```
# Generate a random dataset using make_blobs
X, y = make_blobs(n_samples=160, centers=2, random_state=42)

# Split the dataset into training and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, train_size=0.8, test_size=0.2, random_state=42)
```

Perform a KNN analysis of the simulated data

Step3:

Simulated the KNN model using the above dataset requirements.

Initially we need to take k_values ranging from (1,20) to check our results. When we don't give any k range by default 1 is considered.

Later fit the data using the knn.fit which me we are preparing our model for training.

In the last step we use print statement to display our outputs.

```
# Train a KNN model on the training set and test on the test set with different values of K
k_values = np.arange(1, 20)
accuracies = []
for k in k_values:
    knn = KNeighborsClassifier(n_neighbors=k)
    knn.fit(X_train, y_train)
    y_pred = knn.predict(X_test)
    accuracy = accuracy_score(y_test, y_pred)
    accuracies.append(accuracy)
    print("K-Value :", k)
    print("Predictions from the classifier:")
    learn_data_predicted = knn.predict(X_train)
    print(learn_data_predicted)
    print("Target values:")
    print(y_train)
    print(accuracy_score(learn_data_predicted, y_train))
    #Validation
    print("test-KNN")
    score = knn.score(X_test, y_test)
    print(score)
```

With some parameters, we algorithm as auto and, leaf_size=30 and metric= minkowski and weights=uniform and using 5 n_neighbours.

```
k_values = np.arange(1, 20)
accuracies = []
for k in k_values:
    knn = KNeighborsClassifier(algorithm='auto',
                               leaf_size=40,
                               metric='minkowski',
                               p=2,         # p=2 is equivalent to euclidian distance
                               metric_params=None,
                               n_jobs=1,
                               n_neighbors=k,
                               weights='uniform')
    knn.fit(X_train, y_train)
    y_pred = knn.predict(X_test)
    accuracy = accuracy_score(y_test, y_pred)
    accuracies.append(accuracy)
```

```
# Plot the accuracy scores for different values of K
plt.plot(k_values, accuracies)
plt.title('KNN Accuracy for make_blobs Dataset')
plt.xlabel('K')
plt.ylabel('Accuracy')
plt.show()
```

Output accuracy score

Step 4: Results:

Printing and plotting the results for each k-value, we print predictions from the classifier and target values and test_knn.

The test accuracy using the above samples is 1 for all the k-values.

Also plotted the graph for the same.

```
K-Value : 1
Predictions from the classifier:
[1 1 0 0 1 0 0 1 0 0 1 0 1 0 1 1 0 0 0 0 0 0 0 0 1 1 0 1 0 0 0 0 1 1 0 0
 0 1 1 0 0 0 1 1 0 1 1 1 1 0 0 1 0 1 1 0 1 1 1 1 1 0 0 1 1 1 0 1 0 0 0 0 1
 1 0 1 1 1 1 1 0 0 1 0 0 0 1 1 0 1 1 0 1 1 1 1 1 1 1 1 0 1 0 0 1 1 0 1 1 0 1
 1 1 0 0 0 0 1 1 0 0 1 1 1 1 1 1 1 0]
Target values:
[1 1 0 0 1 0 0 1 0 0 1 0 1 0 1 1 0 0 0 0 0 0 0 0 1 1 0 1 0 0 0 0 1 1 0 0
 0 1 1 0 0 0 1 1 0 1 1 1 1 0 0 1 0 1 1 0 1 1 1 1 1 0 0 1 1 1 0 1 0 0 0 0 1
 1 0 1 1 1 1 1 0 0 1 0 0 0 1 1 0 1 1 0 1 1 1 1 1 1 1 1 0 1 0 0 1 1 0 1 1 0 1
 1 1 0 0 0 0 1 1 0 0 1 1 1 1 1 1 1 0]
1.0
test-KNN
1.0
K-Value : 2
Predictions from the classifier:
[1 1 0 0 1 0 0 1 0 0 1 0 1 0 1 1 0 0 0 0 0 0 0 0 1 1 0 1 0 0 0 0 1 1 0 0
 0 1 1 0 0 0 1 1 0 1 1 1 1 0 0 1 0 1 1 0 1 1 1 1 1 0 0 1 1 1 0 1 0 0 0 0 1
 1 0 1 1 1 1 1 0 0 1 0 0 0 1 1 0 1 1 0 1 1 1 1 1 1 1 1 0 1 0 0 1 1 0 1 1 0 1
```

```
K-Value : 11
Predictions from the classifier:
[1 1 0 0 1 0 0 1 0 0 1 0 1 0 1 1 0 0 0 0 0 0 0 0 1 1 0 1 0 0 0 0 1 1 0 0
 0 1 1 0 0 0 1 1 0 1 1 1 1 0 0 1 0 1 1 0 1 1 1 1 1 0 0 1 1 1 0 1 0 0 0 0 1
 1 0 1 1 1 1 1 0 0 1 0 0 0 1 1 0 1 1 0 1 1 1 1 1 1 1 1 0 1 0 0 1 1 0 1 1 0 1
 1 1 0 0 0 0 1 1 0 0 1 1 1 1 1 1 1 0]
Target values:
[1 1 0 0 1 0 0 1 0 0 1 0 1 0 1 1 0 0 0 0 0 0 0 0 1 1 0 1 0 0 0 0 1 1 0 0
 0 1 1 0 0 0 1 1 0 1 1 1 1 0 0 1 0 1 1 0 1 1 1 1 1 0 0 1 1 1 0 1 0 0 0 0 1
 1 0 1 1 1 1 1 0 0 1 0 0 0 1 1 0 1 1 0 1 1 1 1 1 1 1 1 0 1 0 0 1 1 0 1 1 0 1
 1 1 0 0 0 0 1 1 0 0 1 1 1 1 1 1 1 0]
1.0
test-KNN
1.0
K-Value : 12
Predictions from the classifier:
[1 1 0 0 1 0 0 1 0 0 1 0 1 0 1 1 0 0 0 0 0 0 0 0 1 1 0 1 0 0 0 0 1 1 0 0
 0 1 1 0 0 0 1 1 0 1 1 1 1 0 0 1 0 1 1 0 1 1 1 1 1 0 0 1 1 1 0 1 0 0 0 0 1
 1 0 1 1 1 1 1 0 0 1 0 0 0 1 1 0 1 1 0 1 1 1 1 1 1 1 1 0 1 0 0 1 1 0 1 1 0 1
 1 1 0 0 0 0 1 1 0 0 1 1 1 1 1 1 1 0]
```

Plot your different results

```
test-KNN
1.0
```