Assignment 1

Student's name: Thuy Thien Tram Nguyen

StudentID: 811293917

PART 1: Code interpretation:

Step 1: Import modules to sklearn laboratory:

- At first, we imported the dataset name "make_blobs" to generate an artificial dataset with multiple classes (clusters), accuracy_score to evaluate the dataset outcome, then we imported the Training dataset to sklearn dataset, sklearn.metrics and sklearn.moedl_selection respectively. Moreover, we use matplotlib.pyplot to create visualizations of the data and "numpy" is a numerical computing library that helps with operations on arrays.

```
# Step 1: Import modules for this project
from sklearn.datasets import make_blobs
from sklearn.metrics import accuracy_score
from sklearn.model_selection import train_test_split
import matplotlib.pyplot as plt
import numpy as np
```

Picture 1

Step 2: Create a new stimulated classification dataset:

- In this step, the variable "center" defines the locations of the centers of the clusters in 2D space. For instance, there are 3 clusters located at [2, 4], [6, 6], and [1, 9]. Then we create the labels, each label corresponds to one of the three clusters (classes) and 150 data points are randomly generated around these centers.

```
# Step 2: Create a new simulated classification dataset
# Adjusting informative and redundant features to be consistent with n_features
centers = [[2, 4], [6, 6], [1, 9]]
n_classes = len(centers)
data, labels = make_blobs(n_samples=150, centers=np.array(centers), random_state=1)
```

Picture 2

Step 3: Split the dataset into 80% training and 20% of testing

```
# Step 3: Training and testing split
train_data, test_data, train_labels, test_labels = train_test_split(data, labels,
                                                                    train_size=0.8,
                                                                    test_size=0.2,
                                                                    random_state=12)
```

Picture 3

Step 4: Create an Instance of KNN Classification:

- After importing "KneighborsClassifier" to sklearn. Neighbor, and with knn = KNeighborsClassifier(), no parameter is needed because it will default value as 5, which means the classifier will consider the 5 nearest neighbors to determine the class of a new data point.

```
# Step 4: Create and fit a nearest-neighbor classifier
from sklearn.neighbors import KNeighborsClassifier
knn = KNeighborsClassifier()
knn.fit(train_data, train_labels)
```

Picture 4

Step 5: Make predictions on training data and report accuracy

Accuracy_score() helps compare the predicted labels (train_data_predicted) with the true labels (train_labels) and calculates the accuracy. The Training accuracy = 1.0 indicates the model has captured the patterns in the training set very well.

```
# Step 5: Make predictions on training data and report accuracy

train_data_predicted = knn.predict(train_data)
print("Predictions from the classifier on training data:")
print(train_data_predicted)
print("Target values (Training):")
print(train_labels)
train_accuracy = accuracy_score(train_data_predicted, train_labels)
print(f"Training accuracy: {train_accuracy}")
```

Picture 5

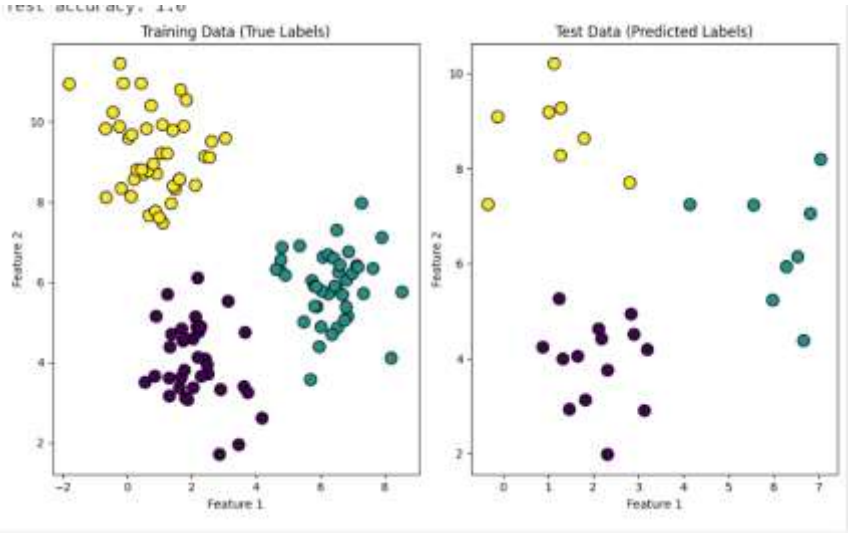
Step 6: Make predictions on test data and report accuracy

Now we start predicting for the 20% of test set . Test accuracy = 1, means it can predict new, unseen data points with perfect accuracy. This reason for this well-down test accuracy because our sample size is only 150 variables and the center location of each clusters are located far away. That is why when we plot the result, we can literally see our 3 clusters.

```
# Step 6: Make predictions on test data and report accuracy
test_data_predicted = knn.predict(test_data)
print("\nPredictions from the classifier on test data:")
print(test_data_predicted)
print("Target values (Test):")
print(test_labels)
test_accuracy = accuracy_score(test_data_predicted, test_labels)
print(f"Test accuracy: {test_accuracy}")
```

Picture 6

Step 7: **Plot the results**



Look at the plot, it can be shown that the Right plot has classified the test data, and give us the insight that values are classified very well comparing with those classes in the training set.