Assignment 1- Dhanush Myneni
Advanced Machine Learning

The first assignment of Advanced Machine Learning Course, 64061 focused on carrying out a basic machine learning project to get acquainted with the concepts of machine learning and revisit the python language concepts.

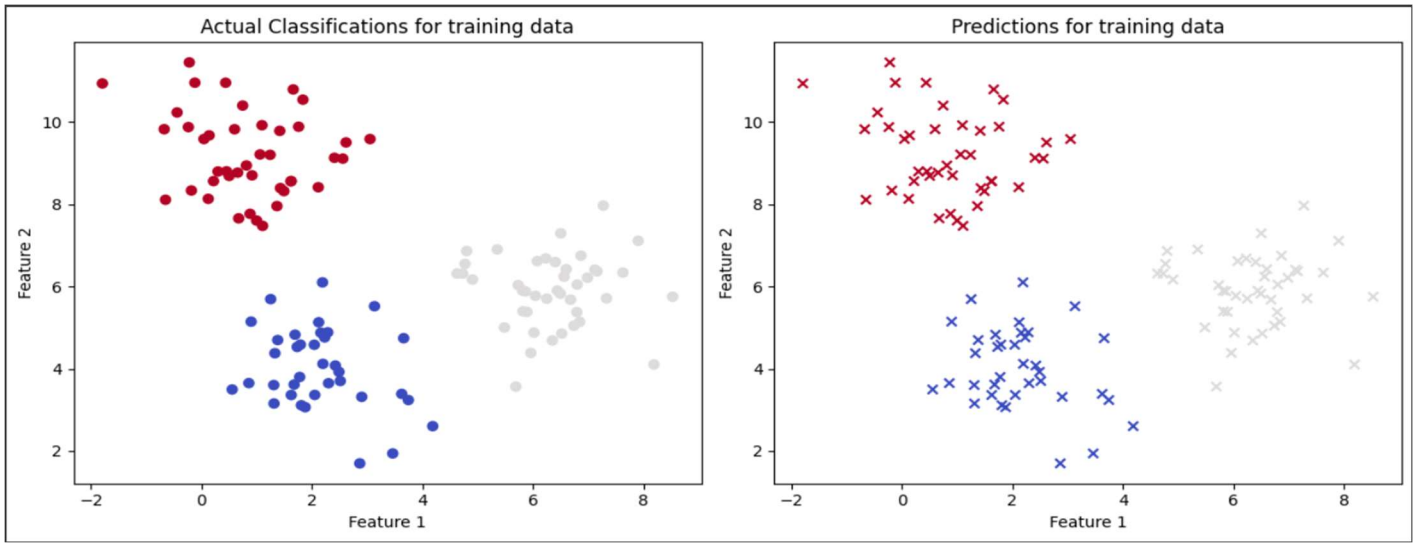
Dataset Description: The fictitious dataset had been created using the make_blobs function and the no. of samples in the dataset were 150 which was divided into three different clusters. We have imported all the necessary libraries to carry out the project including libraries such as NumPy, matplotlib and a few other libraries from the sklearn package.

Train-test Split: The dataset was divided in the ratio of 80-20. This implies that 80% of the data was taken as a training set while the remainder of 20% was considered as the test set. To talk in number of samples, 120 samples of the 150 generated were considered as training data while the remaining 30 are considered as testing data.

Machine learning model used: The model used for our study is the KNN classifier. KNN stands for K nearest neighbor and as the name implies, the data points are predicted based on the closest neighbor to K in the feature space. In the KNN model, we could do hyperparameter tuning. However, in our study we have continued with the default settings of 5, which means that the K implies the no. of nearest neighbors the classifier considers while making a prediction. The model was then trained on 120 samples.

Results Accuracy: The KNN model was then fit on the training as well as the test data. On the training data, the model achieved an accuracy score of 100% indicating that it was correctly able to classify all the sample points accurately. Later, it was fitted on the testing data and the model was again successful in classifying all the data points accurately. The output labels or the predicted labels were aligning completely with the actual labels.

Visualizations: The actual class for training data and the predictions for training data have been plotted to visualize the model's performance. To visualize, matplotlib library of python has been used. The first graph visualizes the true class labels or the actual labels. The classes were color coded as per its label. The second plot visualized the predictions made and since the model achieved 100% accuracy, the predictions align with the actuals.



Appendix

```
from sklearn.datasets import make_blobs #importing the necessary library
import matplotlib.pyplot as plt      #importing matplotlib to perform visualizations
import numpy as np         #importing numpy for data manipulation
from sklearn.metrics import accuracy_score


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)


from sklearn.model_selection import train_test_split  #importing the training, testing split to divide the data.


training_data, testing_data, training_labels, testing_labels = train_test_split(
    data, labels, train_size=0.8, test_size=0.2, random_state=12
) #split the data into the ratio of 80% training data and 20% testing data


# Create and fit a nearest-neighbor classifier
from sklearn.neighbors import KNeighborsClassifier
# classifier "out of the box", no parameters
knn = KNeighborsClassifier()
knn.fit(training_data, training_labels)


# output accuracy score


print("Predictions from the classifier:")
predicted_data = knn.predict(training_data)
print(predicted_data)
print("Target values:")
print(training_labels)
```

```
print(accuracy_score(predicted_data, training_labels))

# plot your different results
plt.figure(figsize=(12, 5))

# Plot the actual classifications for training data
plt.subplot(1, 2, 1)
plt.scatter(training_data[:, 0], training_data[:, 1], c=training_labels, cmap='coolwarm', marker='o')
plt.title('Actual Classifications for training data')
plt.xlabel('Feature 1')
plt.ylabel('Feature 2')

# Plot the predicted classifications for training data
plt.subplot(1, 2, 2)
plt.scatter(training_data[:, 0], training_data[:, 1], c=predicted_data, cmap='coolwarm', marker='x')
plt.title('Predictions for training data')
plt.xlabel('Feature 1')
plt.ylabel('Feature 2')

plt.tight_layout()
plt.show()
```