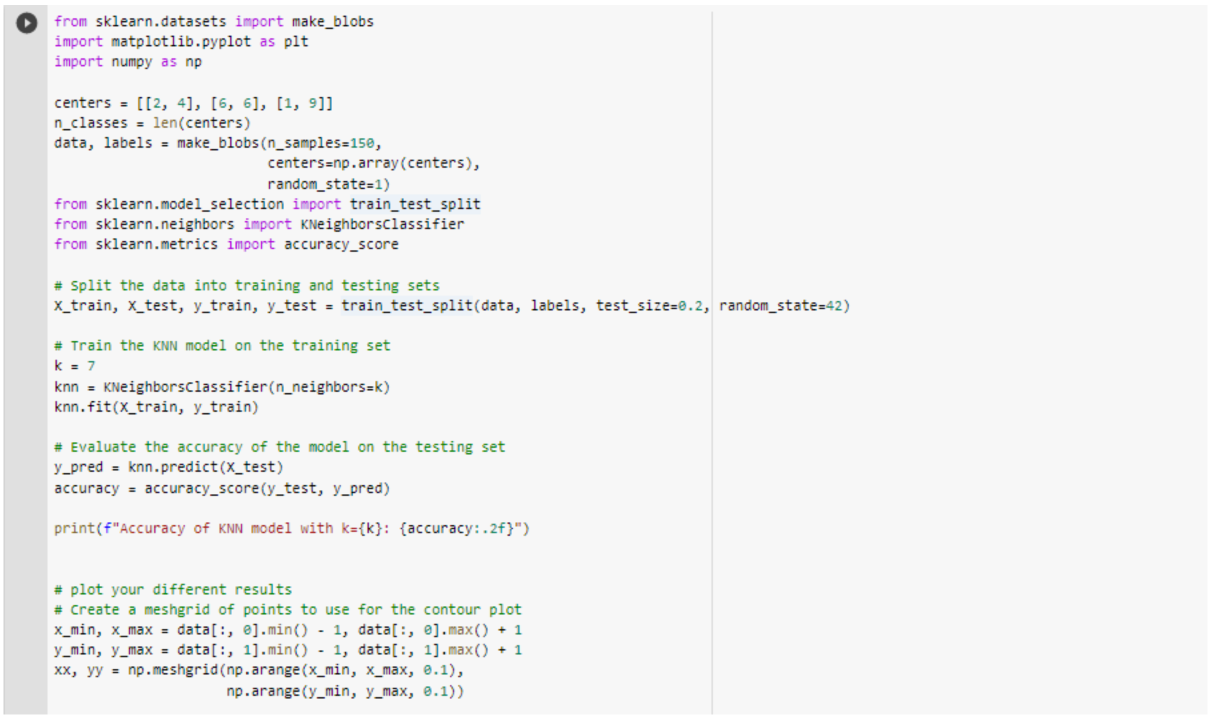
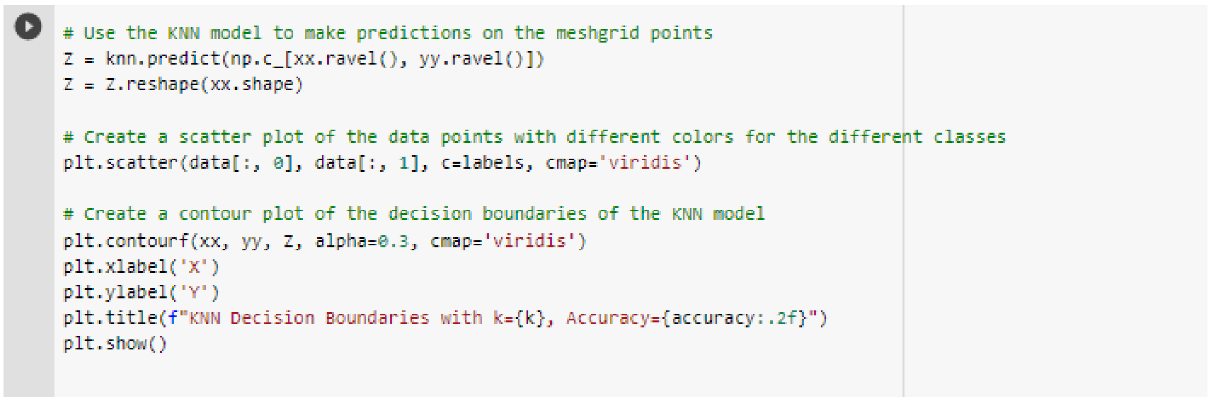
# SUMMARY FOR ASSIGNMENT 1

# ADVANCED MACHINE LEARNING

## CODE AND OUTPUT OF THE ASSIGNMENT

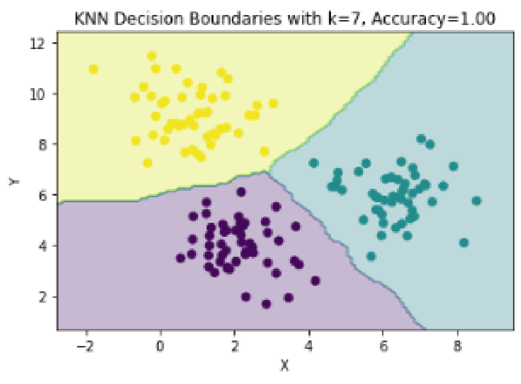


```python
from sklearn.datasets import make_blobs
import matplotlib.pyplot as plt
import numpy as np

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
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(data, labels, test_size=0.2, random_state=42)

# Train the KNN model on the training set
k = 7
knn = KNeighborsClassifier(n_neighbors=k)
knn.fit(X_train, y_train)

# Evaluate the accuracy of the model on the testing set
y_pred = knn.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)

print(f"Accuracy of KNN model with k={k}: {accuracy:.2f}")


# plot your different results
# Create a meshgrid of points to use for the contour plot
x_min, x_max = data[:, 0].min() - 1, data[:, 0].max() + 1
y_min, y_max = data[:, 1].min() - 1, data[:, 1].max() + 1
xx, yy = np.meshgrid(np.arange(x_min, x_max, 0.1),
                     np.arange(y_min, y_max, 0.1))
```



```python
# Use the KNN model to make predictions on the meshgrid points
Z = knn.predict(np.c_[xx.ravel(), yy.ravel()])
Z = Z.reshape(xx.shape)

# Create a scatter plot of the data points with different colors for the different classes
plt.scatter(data[:, 0], data[:, 1], c=labels, cmap='viridis')

# Create a contour plot of the decision boundaries of the KNN model
plt.contourf(xx, yy, Z, alpha=0.3, cmap='viridis')
plt.xlabel('X')
plt.ylabel('Y')
plt.title(f"KNN Decision Boundaries with k={k}, Accuracy={accuracy:.2f}")
plt.show()
```

```
Accuracy of KNN model with k=7: 1.00
```



## Introduction

Using the make blobs method from the sklearn.datasets module, this code creates a fake dataset. There are three classes in the dataset, and each class is represented by a cluster of data points with a unique mean. Based on the obtained data, a KNN classifier is trained using the KNeighborsClassifier model, and its accuracy is assessed using the accuracy score function from the sklearn. metrics module. The algorithm then creates a contour plot showing the KNN model's decision limits and a scatter plot of the data points with different colours for the various classes**.**

Using the centres variable, which contains a list of three coordinate pairs, the algorithm first determines the cluster centres for the data. The length of centres is the value for the n classes variable. Finally, a dataset of 150 samples is created using the make blobs function, with the centres parameter set to np.array (centres) to indicate the centres of the three classes.

## Splitting the data

The required modules for dividing the data into training (80%) and testing sets (20%), building, and testing the KNN model, and determining the model's accuracy are then imported by the code. With the help of the train test split function from the sklearn.model selection module, the data is divided into training and testing sets. The KNeighborsClassifier class from the sklearn.neighbors module is then used to train the KNN model on the training set with k=7 neighbours.

## Accuracy

The accuracy score function from the sklearn.metrics module is then used to assess the model's accuracy on the testing set. And for this particular dataset we got the accuracy 1.00 that is 100% accuracy.

## Plotting the graph

A mesh grid of points is then created by the algorithm to be used for the contour plot of the decision boundaries. Two arrays of coordinates (xx and yy) with a step size of 0.1 are generated using the np. meshgrid function and span the data range. The meshgrid points are then subjected to predictions using the KNN model, and the decision boundaries are contour plotted using the matplotlib . pyplot module's contourf function. The scatter function from the matplotlib.pyplot module is used in the code to build a scatter plot of the data points with distinct colours for the various classes. The KNN model's data points and decision boundaries are plotted as a result, and the model's accuracy is shown in the title of the plot

## Conclusion

The resulting plots give a visual depiction of the KNN model's data points and decision boundaries, which is useful for comprehending how the model generates predictions.