Experiment No. 8

Title: K-Means Clustering for Classification Prediction

Objectives:

1. Implement K-Means clustering to classify new data points.
2. Understand clustering-based classification techniques.
3. Assign majority class labels to clusters for classification.
4. Predict the class for a new data point based on its cluster membership.
5. Develop a Python-based solution for clustering and classification.

Problem Statement:

Given the following data, which specify classifications for nine combinations of VAR1 and VAR2 predict a classification for a case where VAR1=0.906 and VAR2=0.606, using the result of k-means clustering with 3 means (i.e., 3centroids)

VAR1	VAR2	CLASS
1.713	1.586	0
0.180	1.786	1
0.353	1.240	1
0.940	1.566	0
1.486	0.759	1
1.266	1.106	0
1.540	0.419	1
0.459	1.799	1
0.773	0.186	1

Outcomes:

1. A trained K-Means clustering model that groups the given dataset into three clusters.
2. Majority class labels assigned to each cluster.
3. A classification prediction for the new data point.

Tools Required: 4GB RAM, Anaconda, Notebook

Theory:

K-Means clustering is an unsupervised machine learning algorithm that partitions data into K clusters based on similarity. The algorithm works as follows:

1. Select K initial centroids randomly.
2. Assign each data point to the nearest centroid.
3. Update the centroids by computing the mean of all points in each cluster.
4. Repeat steps 2 and 3 until centroids no longer change significantly.

5. Assign a majority class label to each cluster.
6. Classify a new data point based on the cluster it belongs to.

Algorithm:

Steps:

1. Input: Dataset with (VAR1, VAR2) values and class labels.
2. Apply K-Means clustering to partition data into 3 clusters.
3. Determine the majority class for each cluster.
4. Assign labels to the clusters based on majority class.
5. Predict classification for a new data point by:
    1. Finding its closest cluster centroid.
    2. Assigning the corresponding majority class.
6. Output: Predicted classification.

Source Code:

```
import numpy as np
from sklearn.cluster import KMeans
from scipy.stats import mode
data = np.array([
    [1.713, 1.586],
    [0.180, 1.786],
    [0.353, 1.240],
    [0.940, 1.566],
    [1.486, 0.759],
    [1.266, 1.106],
    [1.540, 0.419],
    [0.459, 1.799],
    [0.773, 0.186]
])
classes = np.array([0, 1, 1, 0, 1, 0, 1, 1, 1])
kmeans = KMeans(n_clusters=3, random_state=42, n_init=10)
clusters = kmeans.fit_predict(data)
cluster_labels = np.zeros(3)
for i in range(3):
    cluster_labels[i] = mode(classes[clusters == i], keepdims=True).mode[0]
new_point = np.array([[0.906, 0.606]])
new_cluster = kmeans.predict(new_point)
predicted_class = cluster_labels[new_cluster[0]]
print(f"Predicted Classification: {int(predicted_class)}")
```

Output:

```
Predicted Classification: 1
```

Conclusion:

_____

_____

_____

_____