Course: Minors in Data Science
Subject: Machine Learning

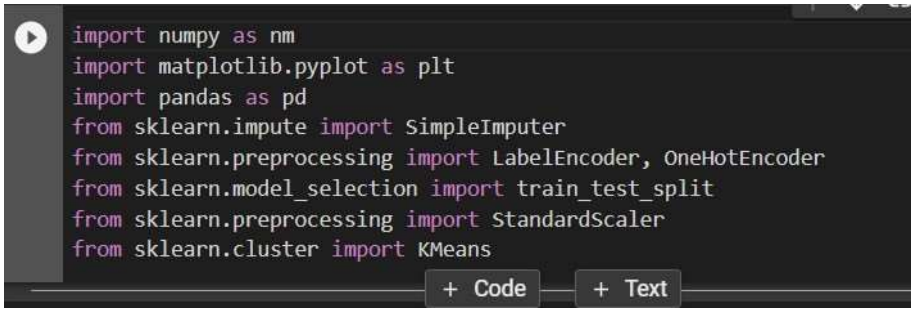
Experiment no: 09

Name: Ayush Jain SAP ID: 60004200132

K-Means

Aim:	Explore K means clustering with variations on different datasets.
Tool used:	Google Colab
Theory:	• **K-Means Clustering Algorithm:** 1. The K-Means clustering algorithm computes centroids and repeats until the optimal centroid is found. It is presumptively known how many clusters there are. It is also known as the flat clustering algorithm. The number of clusters found from data by the method is denoted by the letter 'K' in K-means. 2. In this method, data points are assigned to clusters in such a way that the sum of the squared distances between the data points and the centroid as small as possible. It is essential to note that reduced diversity within clusters lead to more identical data points within the same cluster. Step 1: First, we need to provide the number of clusters, K, that need to be generated by this algorithm Step 2: Next, choose K data points at random and assign each t a cluster. Briefly, categorize the data based on the number of data points. Step 3: The cluster centroids will now be computed. Step 4: Iterate the steps below until we find the ideal centroids, which is assigning of data points to clusters that do not vary. 4.1. The sum of squared distance between data points and centroids would be calculated first. 4.2. At this point, we need to allocate each data point to the cluster that is closest to the others (centroid) 4.3. Finally, compute the centroids for the clusters by averaging all of the clusters' data points • **When using K-Means, we must keep following points in mind:** 1. It is suggested to normalize the data while dealing with clustering algorithms such as K-Means since such algorithms employ distance-based measurement to identify the similarity between data points. 2. Because of the iterative nature it may become stuck in a local optimum and fail to converge to the global optimum. As a result, it is advised to employ distinct centroids' initialisation.

Course: Minors in Data Science
Subject: Machine Learning

Code with Output:	

```
import numpy as nm
import matplotlib.pyplot as plt
import pandas as pd
from sklearn.impute import SimpleImputer
from sklearn.preprocessing import LabelEncoder, OneHotEncoder
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.cluster import KMeans
```

Course: Minors in Data Science
Subject: Machine Learning

```
url = "http://s3.amazonaws.com/assets.datacamp.com/course/Kaggle/train.csv"
df = pd.read_csv(url)
x = df.iloc[:,1:-1].values
y = df.iloc[:,-1].values

label_encoder_x = LabelEncoder()
x[:, 0] = label_encoder_x.fit_transform(x[:, 0])
onehot_encoder = OneHotEncoder()
x = onehot_encoder.fit_transform(x).toarray()
labelencoder_y = LabelEncoder()
y = labelencoder_y.fit_transform(y)

imputer = SimpleImputer(missing_values = nm.nan, strategy = 'mean')
imputerimputer = imputer.fit(x[:, 1:3])
x[:, 1:3] = imputer.transform(x[:, 1:3])

st_x = StandardScaler()
x = st_x.fit_transform(x)

lowest_sse = None
final_centroids = None
final_labels = None
num_iterations = None

for i in range(10):
    kmeans = KMeans(n_clusters=3, init='random', n_init=10, max_iter=300, random_state=i)
    kmeans.fit(x)

    if lowest_sse is None or kmeans.inertia_ < lowest_sse:
        lowest_sse = kmeans.inertia_
        final_centroids = kmeans.cluster_centers_
        final_labels = kmeans.labels_
        num_iterations = kmeans.n_iter_

print("Lowest SSE:", lowest_sse)
print("Final Centroids:\n", final_centroids)
print("Number of Iterations to Converge:", num_iterations)
print("Predicted Labels for First 10 Points:", final_labels[:10])
```

```
Lowest SSE: 1842947.7290527415
Final Centroids:
 [[-1.2669898   1.2669898   1.76776695 ... -0.06715343 -0.03352008
  -1.835115  ]
 [ 0.54406953 -0.54406953 -0.17514946 ... -0.06715343  0.02896191
   0.23621686]
 [-0.62659673  0.62659673  0.19509954 ...  0.07842736 -0.03352008
  -0.26579424]]
Number of Iterations to Converge: 4
Predicted Labels for First 10 Points: [1 2 2 2 1 1 1 2 2 2]
```

Course: Minors in Data Science
Subject: Machine Learning

```
url = "http://s3.amazonaws.com/assets.datacamp.com/course/Kaggle/test.csv"
df = pd.read_csv(url)
x = df.iloc[:,1:-1].values
y = df.iloc[:,-1].values

label_encoder_x = LabelEncoder()
x[:, 0] = label_encoder_x.fit_transform(x[:, 0])
onehot_encoder = OneHotEncoder()
x = onehot_encoder.fit_transform(x).toarray()
labelencoder_y = LabelEncoder()
y = labelencoder_y.fit_transform(y)

imputer = SimpleImputer(missing_values = nm.nan, strategy = 'mean')
imputerimputer = imputer.fit(x[:, 1:3])
x[:, 1:3] = imputer.transform(x[:, 1:3])

st_x = StandardScaler()
x = st_x.fit_transform(x)

lowest_sse = None
final_centroids = None
final_labels = None
num_iterations = None

for i in range(10):
    kmeans = KMeans(n_clusters=3, init='random', n_init=10, max_iter=300, random_state=i)
    kmeans.fit(x)

    if lowest_sse is None or kmeans.inertia_ < lowest_sse:
        lowest_sse = kmeans.inertia_
        final_centroids = kmeans.cluster_centers_
        final_labels = kmeans.labels_
        num_iterations = kmeans.n_iter_

print("Lowest SSE:", lowest_sse)
print("Final Centroids:\n", final_centroids)
print("Number of Iterations to Converge:", num_iterations)
print("Predicted Labels for First 10 Points:", final_labels[:10])
```

```
Lowest SSE: 467313.2000176081
Final Centroids:
 [[ 1.70485837 -0.5349335  -1.04403065 ... -0.06933752 -0.04897021
  -1.89562828]
 [-0.09554098  0.03451184  0.05473293 ... -0.06933752 -0.04897021
   0.09937763]
 [ 0.15701353 -0.05725308 -0.08950284 ...  0.12260344  0.08658971
  -0.16250875]]
Number of Iterations to Converge: 2
Predicted Labels for First 10 Points: [1 2 1 1 2 1 2 1 2 1]
```

Conclusion: With the help of this experiment, we studied and implemented K-Means Clustering Algorithm which is used for Unsupervised Learning. We explored it on different datasets.