

Experiment-2.1

Student Name: Ashish Kumar

Branch: ME CSE AIML

Semester: 02

Subject Name: Machine Learning Lab

UID: 23MAI10008

Section/Group: 23MAI-1

Date of Performance: 21/02/2024

Subject Code: 23CSH-651

Aim of the Experiment :

Implementing K-Means Algorithm using Python.

Theory :

K-Means Clustering is an Unsupervised Machine Learning algorithm, which groups the unlabeled dataset into different clusters. The goal of clustering is to divide the population or set of data points into a number of groups so that the data points within each group are more comparable to one another and different from the data points within the other groups.

K-Means Clustering works well with numerical data, where the concept of distance between data points is meaningful. It's commonly applied to continuous variables. It does not predict labels for new data; it assigns them to existing clusters based on similarity. It is a centroid based algorithm, where each cluster is associated with a centroid.

Here K defines the number of pre-defined clusters that need to be created in the process, as if $K=2$, there will be two clusters, and for $K=3$, there will be three clusters, and so on.

Steps for K-Means Clustering Algorithm:

- 1) Select the number K to decide the number of clusters.
- 2) Select k random points from the data as centroids.
- 3) Calculate the distance of all observations to each of the k centroids.
- 4) Assign the observations to the closest centroid.
- 5) Find the new location of the centroid by taking the mean of all the observations in each cluster.
- 6) Repeat Steps 3-5 until the centroids do not change position..

Example:

Customer segmentation in marketing: K-means groups customers based on purchasing behavior, allowing businesses to tailor marketing strategies for different segments.

Code for Experiment :

```
# Importing libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

# Load the dataset
data = pd.read_csv('clustering.csv')
data.head()

# Read the features
X = data[["LoanAmount", "ApplicantIncome"]]

# Visualise data points
plt.scatter(X["ApplicantIncome"], X["LoanAmount"], c='black')
plt.xlabel('Annual Income')
plt.ylabel('Loan Amount (In Thousands)')
plt.title('Original Dataset')
plt.show()

# Step1 - Choose the Number of Clusters (K)
K=3

# Step2 - Select Random Centroid for each Cluster
Centroids = (X.sample(n=K))

# Plot the Centroids
plt.scatter(X["ApplicantIncome"], X["LoanAmount"], c='black')
```

```
plt.scatter(Centroids["ApplicantIncome"],Centroids["LoanAmount"],c='red')  
plt.xlabel('Annual Income')  
plt.ylabel('Loan Amount (In Thousands)')  
plt.title('Dataset with Initial Centroids')  
plt.show()
```

```
print('Initial Centroids: \n',Centroids)  
# Step3 - Assign all the points to the closest cluster centroid  
# Step4 - Recompute centroids of newly formed clusters  
# Step5 - Repeat step 3 and 4
```

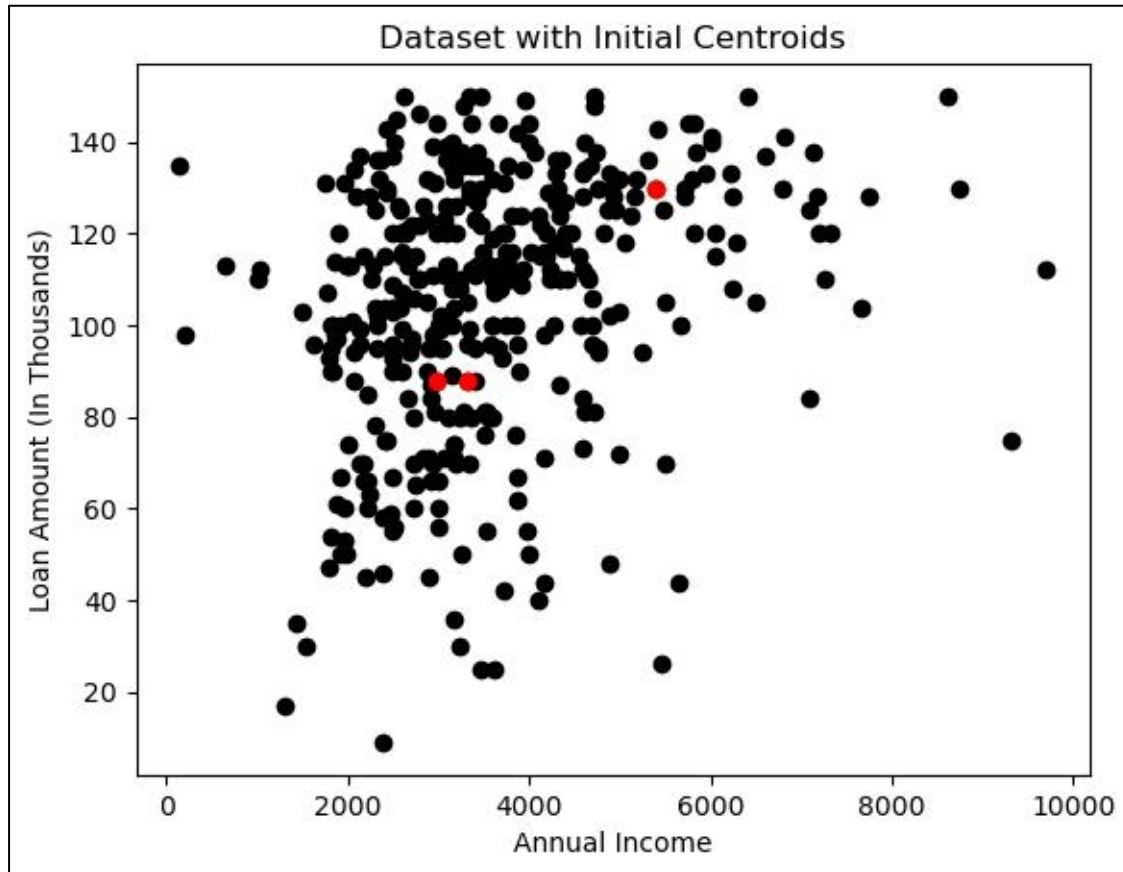
```
diff = 1  
j=0  
p=1  
print("\nDifference between Previous and Current Centroids:")  
while(diff != 0):  
    # XD will contain distance of each point from each cluster  
    XD=X.copy()  
    i=1  
    for index1,row_c in Centroids.iterrows():  
        ED=[]  
        for index2,row_d in X.iterrows():  
            # Calculate distance of each point to centroid  
            d1=(row_c["ApplicantIncome"]-row_d["ApplicantIncome"])**2  
            d2=(row_c["LoanAmount"]-row_d["LoanAmount"])**2  
            d=np.sqrt(d1+d2)  
            ED.append(d)  
        XD[i]=ED  
        i=i+1  
  
    # Store the Cluster number for each point  
    C=[]
```

```
for index,row in XD.iterrows():
    min_dist=row[1]
    pos=1
    for i in range(K):
        if row[i+1] < min_dist:
            min_dist = row[i+1]
            pos=i+1
    C.append(pos)

XD["Cluster"]=C
Centroids_new = XD.groupby(["Cluster"]).mean()[["LoanAmount","ApplicantIncome"]]
if j == 0:
    diff=1
    j=j+1
else:
    diff = (Centroids_new['LoanAmount'] - Centroids['LoanAmount']).sum() +
            (Centroids_new['ApplicantIncome'] - Centroids['ApplicantIncome']).sum()
    print("Round",p," : ",diff.sum())
    p=p+1
Centroids = XD.groupby(["Cluster"]).mean()[["LoanAmount","ApplicantIncome"]]

# Plot Clusters with differnt colors
color=['blue','green','cyan']
for k in range(K):
    data=XD[XD["Cluster"]==k+1]
    plt.scatter(data["ApplicantIncome"],data["LoanAmount"],c=color[k])
plt.scatter(Centroids["ApplicantIncome"],Centroids["LoanAmount"],c='red')
plt.xlabel('Income')
plt.ylabel('Loan Amount (In Thousands)')
plt.title('Different Clusters with Centroids')
plt.show()
```

Result/Output :

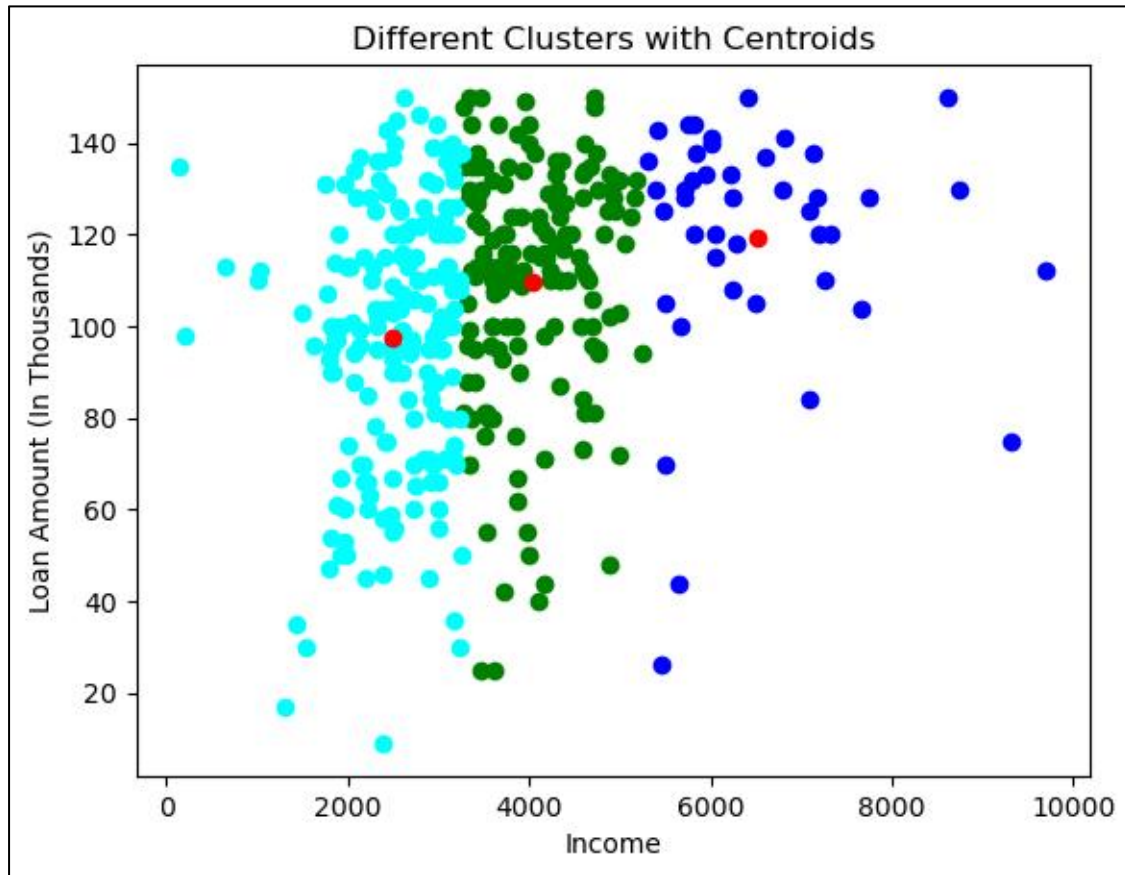


```
jupyter ML_Experiment5_ASHISH_23MAI10008 Last Checkpoint: 5 minutes ago (unsaved changes)
```

```
File Edit View Insert Cell Kernel Widgets Help Trusted
```

```
Initial Centroids:
      LoanAmount ApplicantIncome
238      130.0           5391
87       88.0           3316
375      88.0           2987

Difference between Previous and Current Centroids:
Round 1 : 245.2644471616966
Round 2 : 296.6167306177822
Round 3 : 234.6779564386627
Round 4 : 205.13084046778445
Round 5 : 96.60716651619987
Round 6 : 80.42253001270922
Round 7 : 99.50821192263949
Round 8 : 27.38200063816943
Round 9 : 18.274686272279013
Round 10 : 9.21023994083339
Round 11 : 18.345487493007468
Round 12 : 46.27013250786139
Round 13 : 0.0
```



Learning outcomes (What I have learnt):

1. I learnt about various python libraries like pandas, matplotlib and numpy.
2. I learnt about the concept of K-Means Clustering Algorithm.
3. I learnt about how to select the value of K to determine clusters.
4. I learnt about how to assign data points to different clusters.
5. I learnt about how to calculate the Euclidean Distance.