K clustering project Report

K means clustering is a part of unsupervised learning in AI is used to group similar data points in a process called clustering.

K- means clustering is a popular unsupervised machine learning algorithm used for partitioning a dataset into a predefined number of clusters.

Objective of k means Clustering

The main objective of k-means clustering is to partition your data into a specific number (k) of groups, where data points within each group are similar and dissimilar to points in other groups. It achieves this by minimizing the distance between data points and their assigned cluster's center, called the centroid.

Implementing K-Means Clustering in Python From Scratch

Time to fire up our Jupyter notebooks (IDE you use) and get our hands dirty in Python.

First, import all the required libraries:

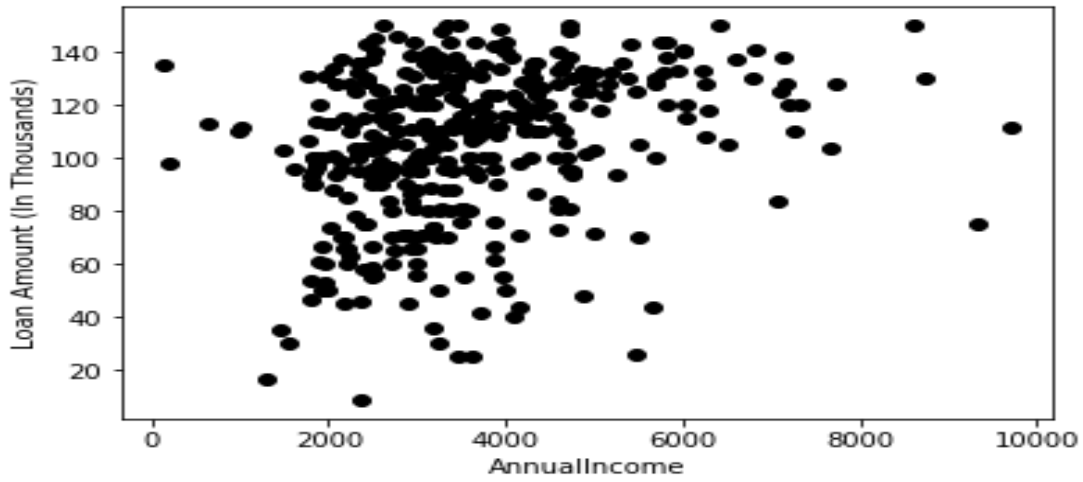
```
1.import pandas as pd
2.import numpy as np
3.import random as rd
4.import matplotlib.pyplot as plt
```

Now, we will read the CSV file and look at the first five rows of the data:

```
1.data = pd.read_csv('clustering.csv')
2.data.head()
```

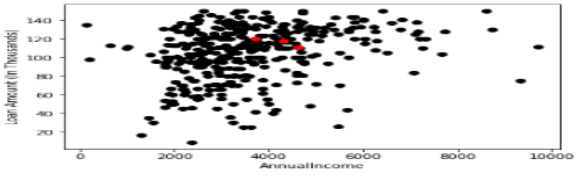
Loan_ID	Gender	Married	Dependents	Education	Self_Employed	Applicant Income	Coapplicant Income	Loan Amount	Loan_Amount_TERM
LP001003	Male	Yes	1	Graduate	No	4583	1508.0	128.0	360.0
LP001005	Male	Yes	0	Graduate	Yes	3000	0.0	66.0	360.0
LP001006	Male	Yes	0	Not Graduate	No	2583	2358.0	120.0	360.0
LP001008	Male	No	0	Graduate	No	6000	0.0	141.0	360.0
LP001013	Male	Yes	0	Not Graduate	No	2333	1516.0	95.0	360.0

We will be taking only two variables from the data – “LoanAmount” and “ApplicantIncome.” This will make it easy to visualize the steps as well. Let’s pick these two variables and visualize the data points:



Steps 1 and 2 of K-Means were about choosing the number of clusters (k) and selecting random centroids for each cluster. We will pick 3 clusters and then select random observations from the data as the centroids:

```
1.  1.# Step 1 and 2 - Choose the number of clusters (k) and 2.select random centroid for
    each cluster
2.  3.#number of clusters
3.  4.K=3
4.  5.# Select random observation as centroids
5.  Centroids = (X.sample(n=K))
6.  6.plt.scatter(X["ApplicantIncome"],X["LoanAmount"],c='black')
7.  plt.scatter(Centroids["ApplicantIncome"],Centroids["LoanAmount"],c='red')
8.  plt.xlabel('AnnualIncome')
9.  plt.ylabel('Loan Amount (In Thousands)')
10. plt.show()
```



Here, the red dots represent the 3 centroids for each cluster. Note that we have chosen these points randomly, and hence every time you run this code, you might get different centroids.Next, we will define some conditions to implement the K-Means Clustering algorithm. Let’s first look at the code:

```python
# Step 3 - Assign all the points to the closest cluster centroid

# Step 4 - Recompute centroids of newly formed clusters

Step 5 - Repeat step 3 and 4

diff = 1

j=0

while(diff!=0):

XD=X

i=1

for index1,row_c in Centroids.iterrows():

ED=[]

for index2,row_d in XD.iterrows():

d1=(row_c["ApplicantIncome"]-row_d["ApplicantIncome"])**2

d2=(row_c["LoanAmount"]-row_d["LoanAmount"])**2

d=np.sqrt(d1+d2)

ED.append(d)

X[i]=ED

i=i+1

C=[]

for index,row in X.iterrows():

min_dist=row[1]

pos=1

for i in range(K):

if row[i+1] < min_dist:

pos=i+1

C.append(pos)

Centroids_new = X.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

print(diff.sum())

Centroids = X.groupby(["Cluster"]).mean()[["LoanAmount","ApplicantIncome"]
```

```
0.0
-6.23770917712692
-6.16844100601777
-6.160752402517077
-18.485593876594225
-55.93913897407591
-47.009664062845815
74.07828057315557
338.33088353063154
```