Teammates:

Sukumar Bodapati – sb5zh@umsystem.edu - 16326105

GitHub link: https://github.com/UMKC-APL-BigDataAnalytics/icp-6-sukumarbodapati

Sri Nikhitha Boddapati – sb4dz@umsystem.edu - 16322565

GitHub link: https://github.com/UMKC-APL-BigDataAnalytics/icp-6-Srinikhitha98

Video link: https://youtu.be/LVdqx7qtrT0

ICP - Big Data App & Analytics -KMean Clustering

Learnings in this ICP:

In this ICP, we have learned about simply and popular unsupervised machine learning algorithm known as K-Means clustering. We collect all the datapoints that have same similarities which is referred as clusters. In this algorithm, we identify the k number of centroids and assign the datapoints to the nearest cluster. The second step is we recompute the centroid until the criteria is met. Plotted elbow curve to find optimal number of clusters. Inertia is a metric that indicates how well a dataset was clustered. It's determined by squaring the distance between each data point and its centroid and summing the squares throughout one cluster.

Objective:

To take different dataset and use the model given in ICP7 to perform the clustering and try different number of clusters.

Challenges:

1. **Clustering Outliners:** When we identify the centroids, Outliers can pull centroids. To overcome this, we must remove or clip the outliners.
2. Defining the cluster size is very big challenge.
3. K-means are not performing well when we have varying size and different density of clusters. We have encounter this we used different datasets.

Tasks:

1.We have imported the libraries required, load the dataframe using pandas and drop the unwanted columns.

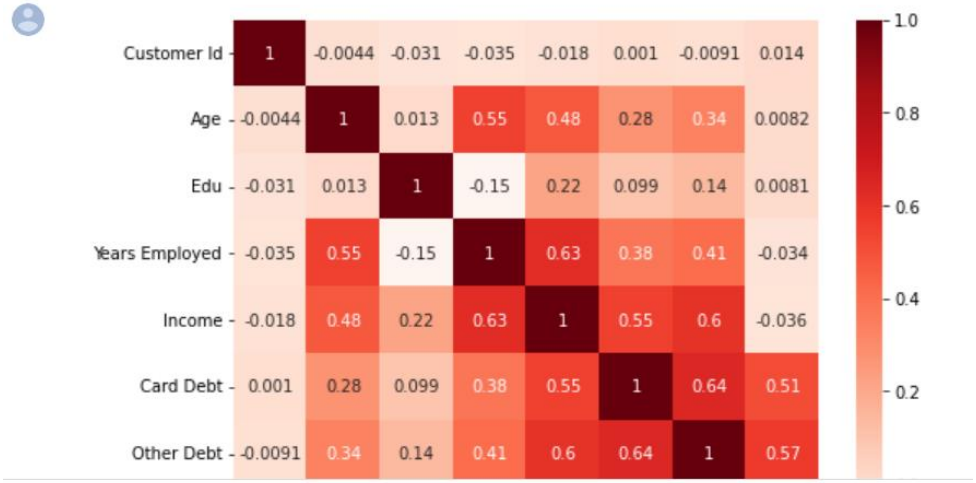
	Customer Id	Age	Edu	Years Employed	Income	Card Debt	Other Debt	DebtIncomeRatio
0	1	41	2	6	19	0.124	1.073	6.3
1	2	47	1	26	100	4.582	8.218	12.8
2	3	33	2	10	57	6.111	5.802	20.9
3	4	29	2	4	19	0.681	0.516	6.3
4	5	47	1	31	253	9.308	8.908	7.2

2.Analyse the statistics of the data.

	Customer Id	Age	Edu	Years Employed	Income	Card Debt	Other Debt	DebtIncomeRatio
count	850.00000	850.000000	850.000000	850.000000	850.000000	850.000000	850.000000	850.000000
mean	425.50000	35.029412	1.710588	8.565882	46.675294	1.576820	3.078773	10.171647
std	245.51816	8.041432	0.927784	6.777884	38.543054	2.125843	3.398799	6.719441
min	1.00000	20.000000	1.000000	0.000000	13.000000	0.012000	0.046000	0.100000
25%	213.25000	29.000000	1.000000	3.000000	24.000000	0.382500	1.045750	5.100000
50%	425.50000	34.000000	1.000000	7.000000	35.000000	0.885000	2.003000	8.700000
75%	637.75000	41.000000	2.000000	13.000000	55.750000	1.898500	3.903250	13.800000
max	850.00000	56.000000	5.000000	33.000000	446.000000	20.561000	35.197000	41.300000

3.We have plotted heatmap for finding the relationship between the datasets.

```
cor=data.corr()
plt.figure(figsize=(8,6))
sns.heatmap(cor,annot=True,cmap=plt.cm.Reds)
plt.show()
```



4.Since we are using k-means algorithm, we need to normalize the data otherwise the difference of magnitude can create a problem.

```
array([[-1.7300143 ,  0.74291541,  0.31212243, ..., -0.68381116,
        -0.59048916, -0.57652509],
       [-1.72593888,  1.48949049, -0.76634938, ...,  1.41447366,
         1.51296181,  0.39138677],
       [-1.72186347, -0.25251804,  0.31212243, ...,  2.13414111,
         0.80170393,  1.59755385],
       ...,
       [ 1.72186347, -1.24795149,  2.46906604, ...,  0.5766659 ,
         0.03863257,  3.45892281],
       [ 1.72593888, -0.37694723, -0.76634938, ..., -0.68757659,
        -0.70147601, -1.08281745],
       [ 1.7300143 ,  2.1116364 , -0.76634938, ...,  0.13611081,
         0.16463355, -0.2340332 ]])
```

5. Setting up K-Means

Now that we have our random data, let's set up our K-Means Clustering.
The KMeans class has many parameters that can be used, but we will be using these three:

- n_clusters: The number of clusters to form as well as the number of centroids to generate.
- n_init: Number of times the k-means algorithm will be run with different centroid seeds. The results will be the best output of n_init consecutive runs in terms of inertia.
    - Value will be: 12
- init: Initialization method of the centroids.
    - Value will be: "k-means++"
    - k-means++: Selects initial cluster centers for k-mean clustering in a smart way to speed up convergence.

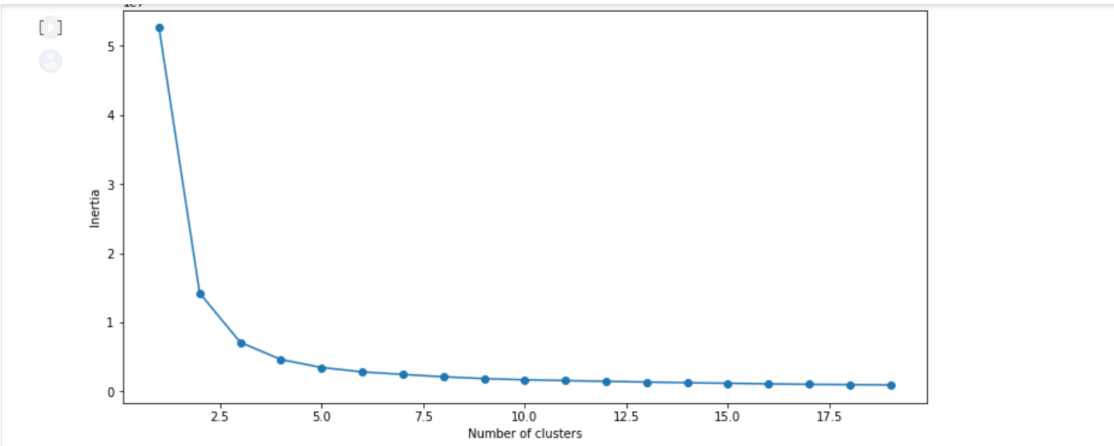
```
#Setting the cluster count
clusterNum = 6

#Created a kmeans object and assigning parameters
k_means = KMeans(init = "k-means++", n_clusters = clusterNum, n_init = 12)
#using fit we are training the model
k_means.fit(X)

#copying the lables into a variable
labels = k_means.labels_
```

6.Calculate the inertia

```
[ ]  # inertia on the fitted data
     k_means.inertia_
```

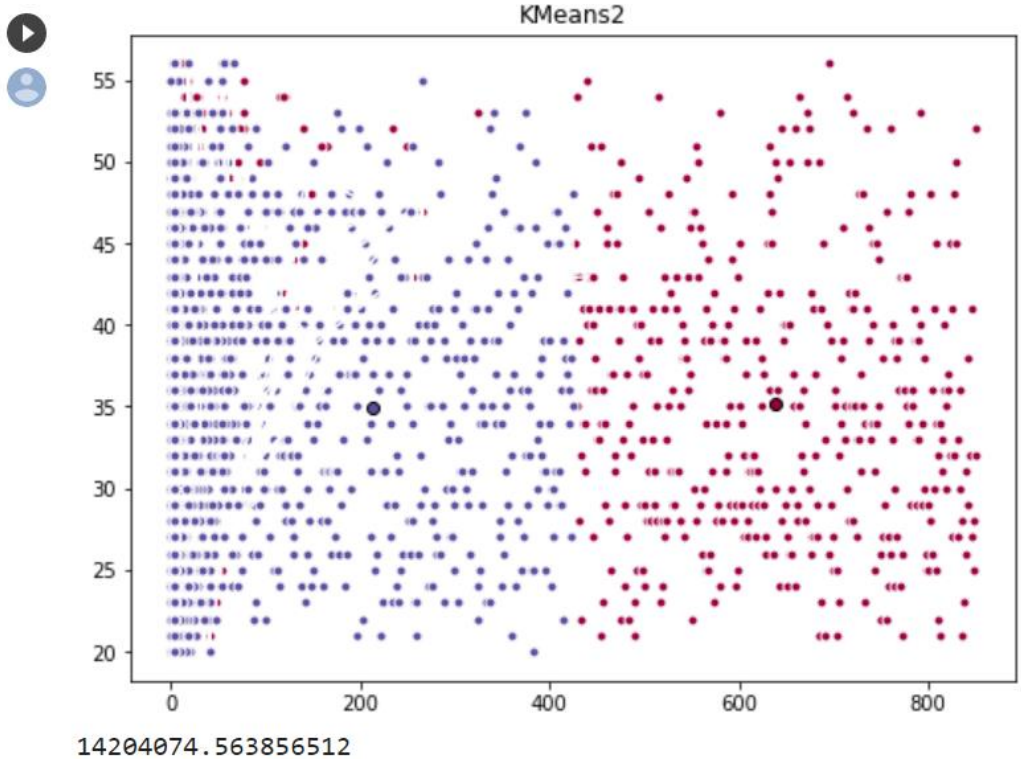
```
2820026.843424972
```

7. Plot the elbow curve:In order to decide optimum clusters we can use we need to plot the elbow curve

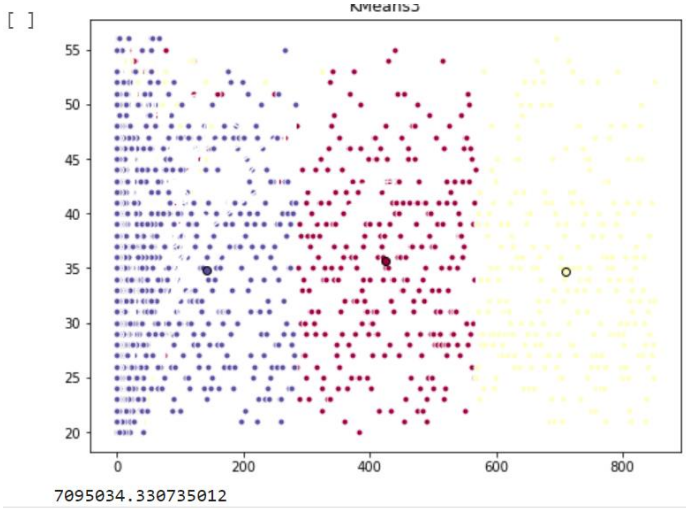


After plotting the graph between the number of clusters and inertia , we can choose the value from 2 to 6.

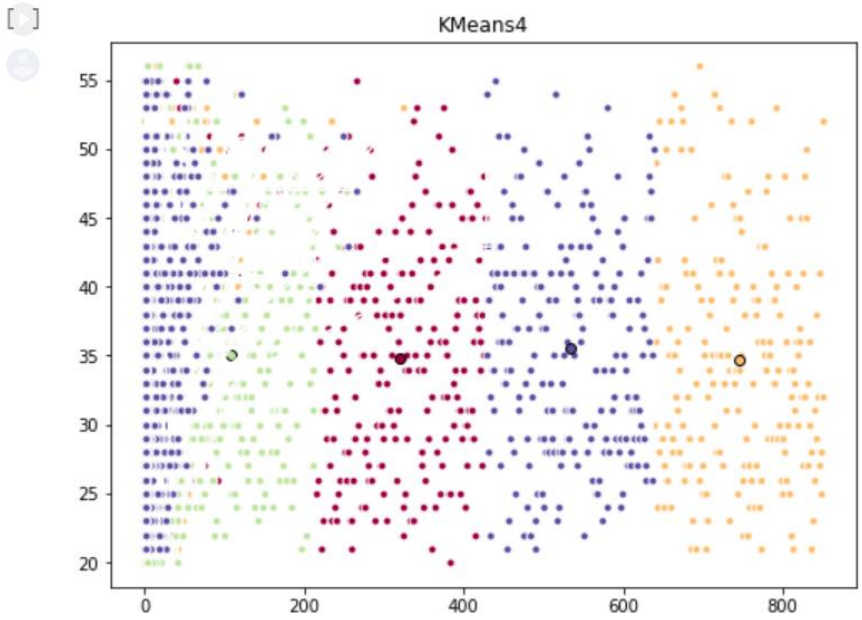
Clusters=2 : When we have given cluster size as 2 , the entire data is divided into two clusters and we calculated respective centroids and inertia value for this.



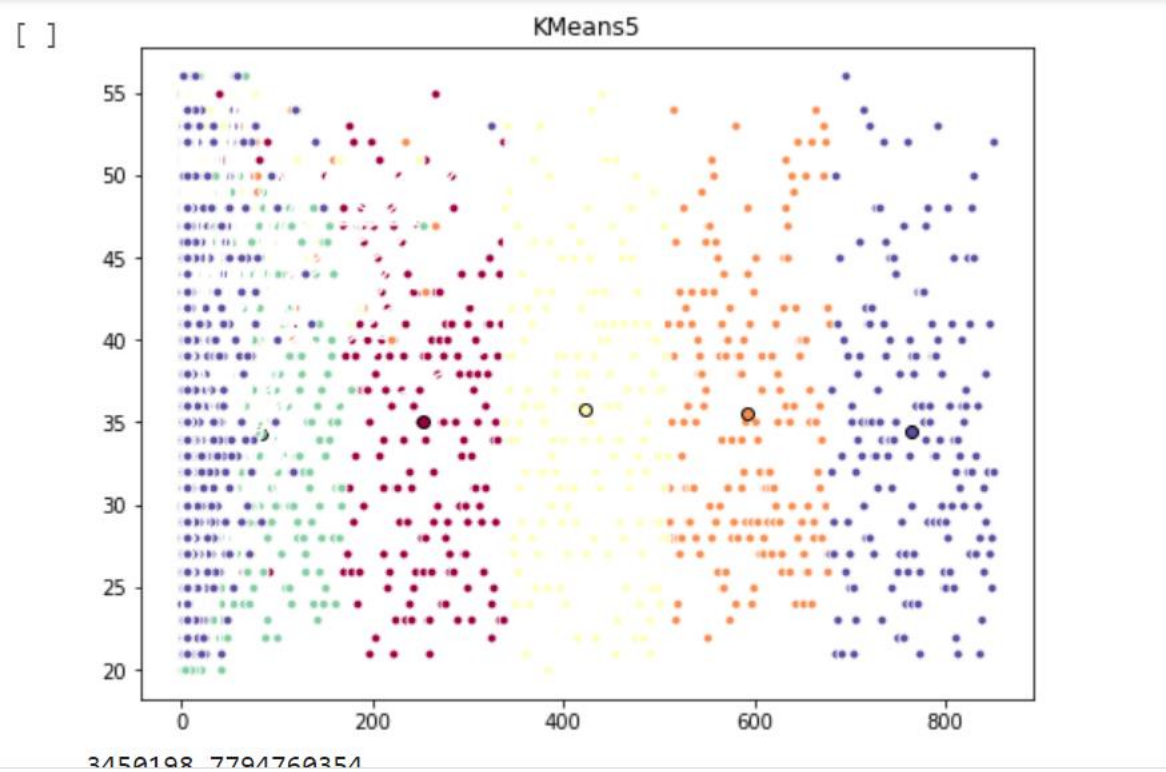
Clusters=3: When we have given cluster size as 3 , the entire data is divided into three clusters and we calculated respective centroids and inertia value for this.



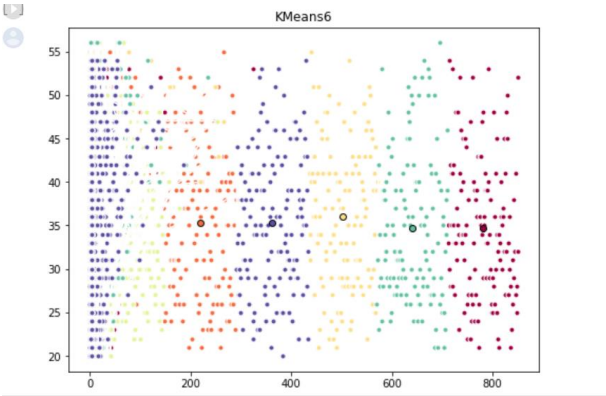
Clusters=4 : When we have given cluster size as 4 , the entire data is divided into four clusters and we calculated respective centroids and inertia value for this.



Clusters =5: When we have given cluster size as 5 , the entire data is divided into five clusters and we calculated respective centroids and inertia value for this.



Clusters=6 : When we have given cluster size as 6 , the entire data is divided into six clusters and we calculated respective centroids and inertia value for this.



However a good model is the one with low inertia and low number of clusters .The below are the inertia values and cluster size:

Cluster	**Inertia value**
2	14204074
3	7095034
4	4604665
5	3450198
6	2819312