Day 46

DIY

Q1. Problem Statement: K Means Clustering

Write a Python program that reads the *Credit Card Customer Data.csv* (provided on LMS). The following are the tasks that need to be considered while constructing the solution to Segregate customers based on the data provided with the help of k-means clustering.

1. Load the Given CSV file into a DataFrame
2. Find missing values and drop the unnecessary columns
3. Univariate and bivariate analysis
4. Standardize the whole dataset
5. Find the within-cluster sum of square
6. Find the silhouette score
7. Use a line plot using matplotlib to find scores for different sizes of K and choose the best size for the cluster and build the final model
8. Observe Cluster behavior with different columns.
9. Print Co-ordinates of all centroids and silhouette scores for the final model

Dataset:

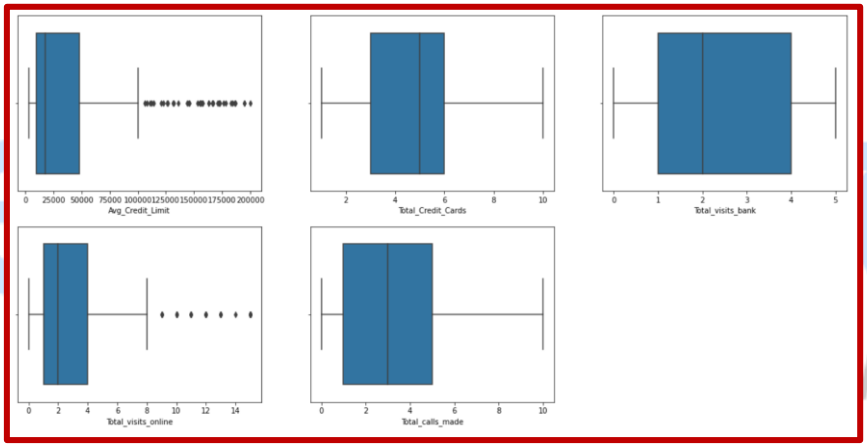
The data set contains information about the customer and their bank data like how many credit cards they have, how many times they used online, how many times they visit the bank etc.

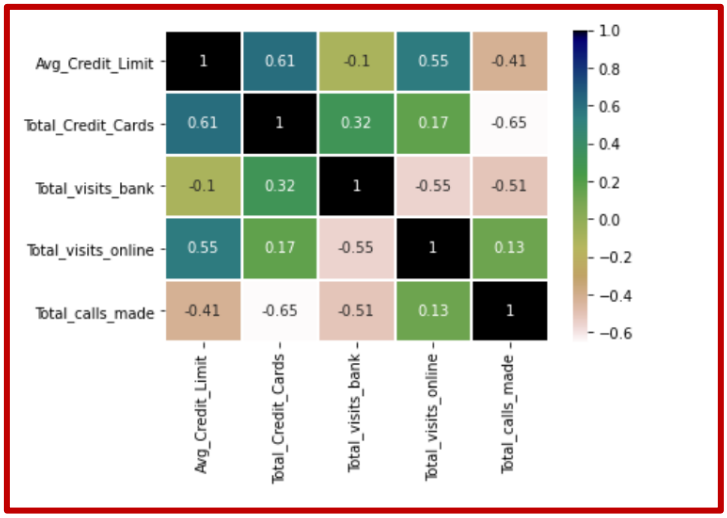
	Sl_No	Customer Key	Avg_Credit_Limit	Total_Credit_Cards	Total_visits_bank	Total_visits_online	Total_calls_made
0	1	87073	100000	2	1	1	0
1	2	38414	50000	3	0	10	9
2	3	17341	50000	7	1	3	4
3	4	40496	30000	5	1	1	4
4	5	47437	100000	6	0	12	3
...	...	...	...	...	...	...	...
655	656	51108	99000	10	1	10	0
656	657	60732	84000	10	1	13	2
657	658	53834	145000	8	1	9	1
658	659	80655	172000	10	1	15	0
659	660	80150	167000	9	0	12	2

660 rows × 7 columns

Sample Output:

3. Univariate and bivariate analysis



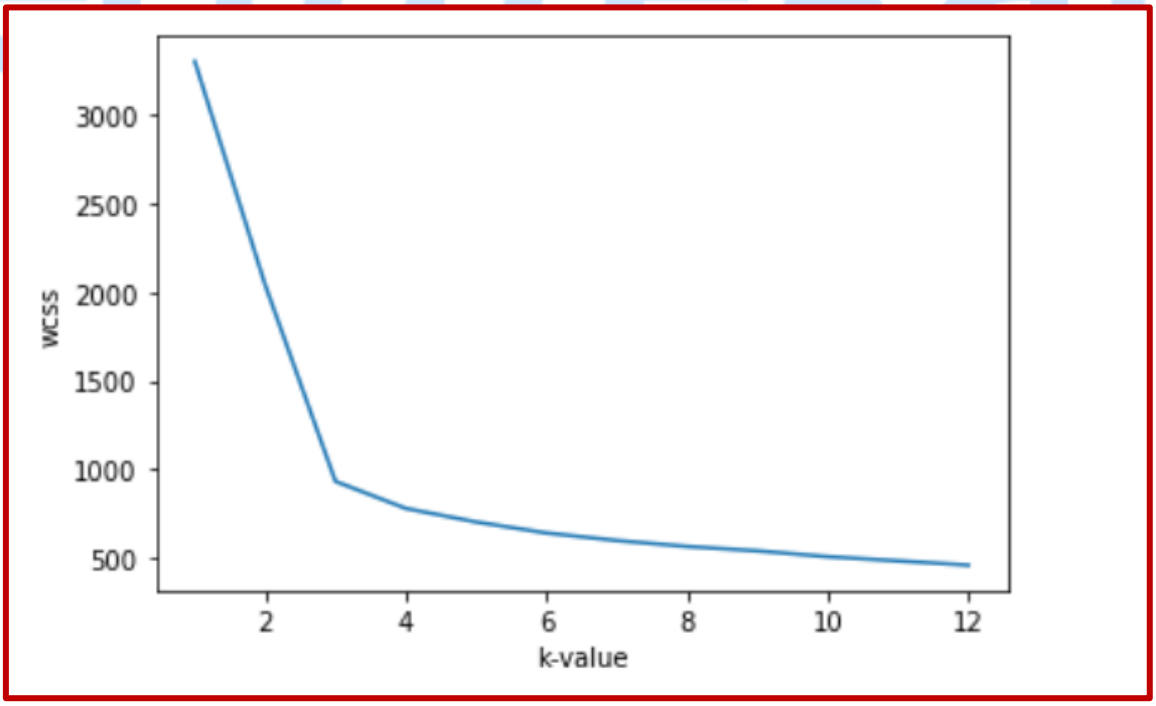


4. Standardize the whole dataset

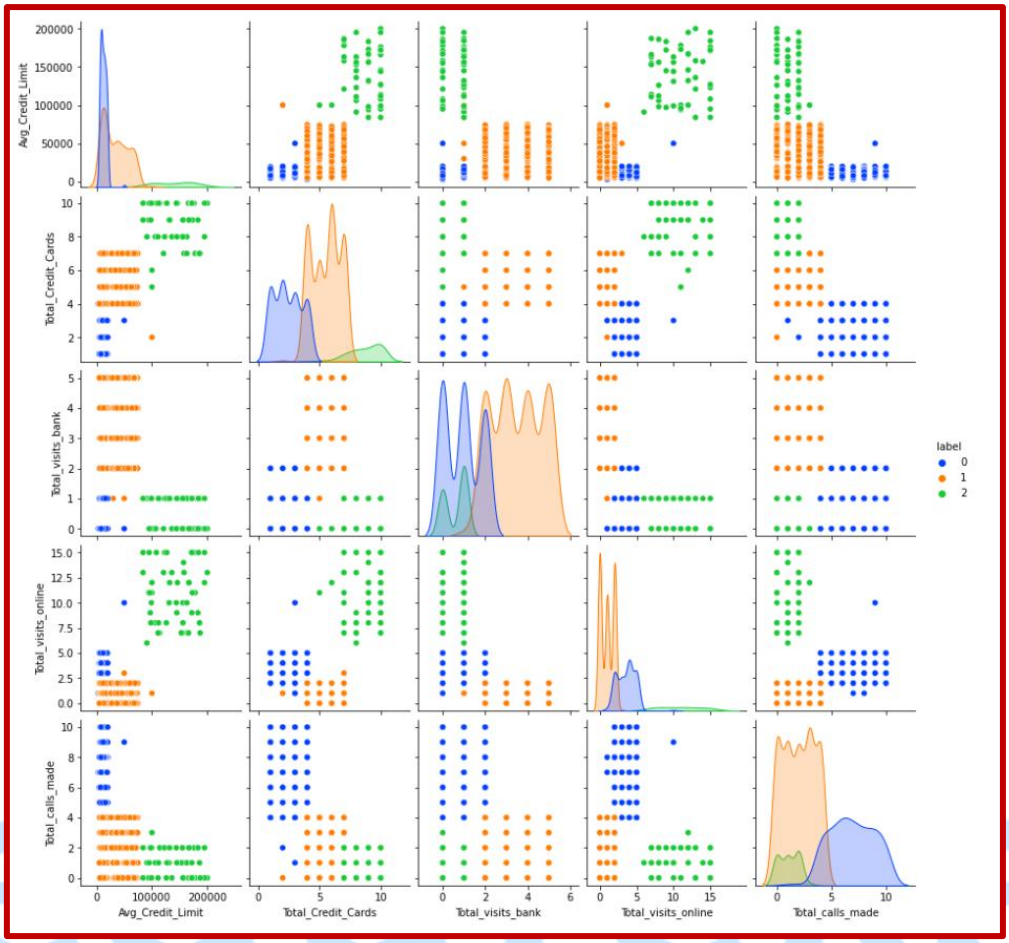
	Avg_Credit_Limit	Total_Credit_Cards	Total_visits_bank	Total_visits_online	Total_calls_made
0	1.740187	-1.249225	-0.860451	-0.547490	-1.251537
1	0.410293	-0.787585	-1.473731	2.520519	1.891859
2	0.410293	1.058973	-0.860451	0.134290	0.145528
3	-0.121665	0.135694	-0.860451	-0.547490	0.145528
4	1.740187	0.597334	-1.473731	3.202298	-0.203739
...	...	...	...	...	...
655	1.713589	2.443892	-0.860451	2.520519	-1.251537
656	1.314621	2.443892	-0.860451	3.543188	-0.553005
657	2.937092	1.520613	-0.860451	2.179629	-0.902271
658	3.655235	2.443892	-0.860451	4.224968	-1.251537
659	3.522245	1.982253	-1.473731	3.202298	-0.553005

660 rows × 5 columns

7. Plot the score for different sizes of K and choose the best size for the cluster and build the final model



8. Observe Cluster behavior with different columns.



9. Print Co-ordinates of all centroids and silhouette scores for the final model

Co ordinates

```
array([[-0.59579625, -1.05962278, -0.9015185 ,  0.32299678,  1.14810882],
       [-0.02106178,  0.37368962,  0.6663945 , -0.55367163, -0.55300488],
       [ 2.83176409,  1.86222621, -1.10576269,  2.82731942, -0.87432983]])
```

final score

```
0.5157182558881063
```