Chapter 6: Clustering DMP Cindy

[**Data Acquisition**](#_hche4mprku3i) **1**

[**Data Preprocessing**](#_xloqkijfuzm5) **2**

[**Code Development**](#_yye27b320e4r) **2**

[K-means](#_x3npjszifne8) 2

[**Package Used**](#_4m5wlzkcs229) **6**

[Elbow Method and K-means](#_ml3ujokoh9b6) 6

[PCA and K-means](#_5ll0754i1s1t) 7

[Hierarchical Clustering](#_ujajk7gomb96) 8

[DBSCAN](#_miwby5mmzyk6) 9

[**Theory Exercise**](#_7gjiznv99km8) **11**

[**Self-assessment**](#_w8jqfvdkga37) **11**

[**Reference**](#_8pttcs8x0u7c) **12**

# 

# Data Acquisition

The dataset I chose for this portfolio is related to **customer credit cards**. A *Customer Credit Card Information* Dataset can be used for Identifying Loyal Customers, Customer Segmentation, Targeted Marketing, and other such use cases in the Marketing Industry. It is important for businesses to make smart marketing strategies for targeting potential customers.

Our credit card dataset contains 7 features with 660 observations.

| Features | Description |
| --- | --- |
| Sl\_No | Serial number (Unique to each customer) |
| Customer Key | Customer ID (Unique to each customer) |
| AvgCreditLimit | Average credit limit |
| TotalCreditCards | Number of total credit cards |
| Totalvisitsbank | Number of total visits |
| Totalvisitsonline | Number of total online visits |
| Totalcallsmade | Number of total calls |

The shape of data: 660 rows and 7 columns

No missing values

Dataset [download link](https://www.kaggle.com/aryashah2k/credit-card-customer-data) from Kaggle

A few tasks that can be performed using this dataset is as follows:

* Perform Data-Cleaning, Preprocessing, Visualization on the Dataset.
* Implement K-Means Clustering models.
* Use the Elbow method to pick an appropriate value of k
* Use hierarchical clustering to visualize
* Use DBSCAN
* Create RFM (Recency, Frequency, Monetary) Matrix to identify Loyal Customers.

# Data Preprocessing

Since *SI\_NO* and *Customer Key* are unique to each customer, I decided to drop these two columns because they are not useful for clustering.

df=df.drop(columns = ['Sl\_No', 'Customer Key'])

Now, the credit card dataset contains 5 features with 660 observations.

# Code Development

[Link to Colab](https://colab.research.google.com/drive/1n-6oOTOKd-IGMpixv2KZjFlvliX4BSY5)

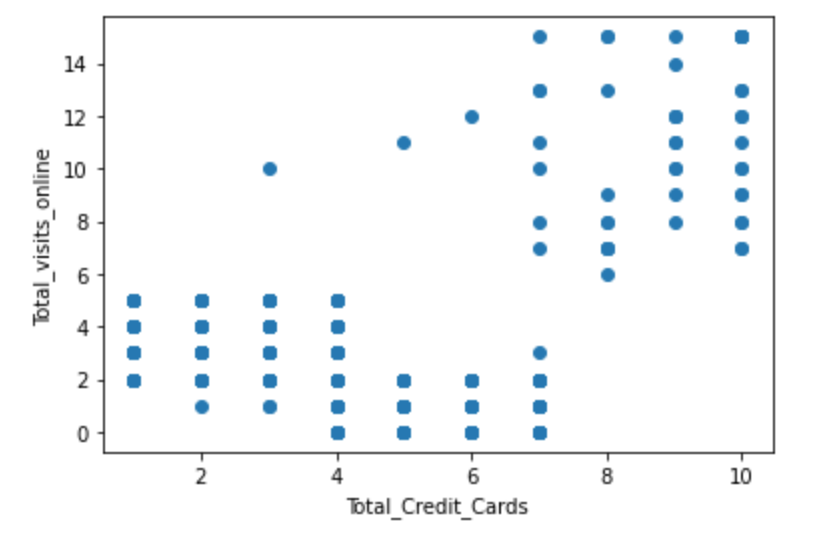
## K-means

I implement the K-means algorithm from scratch. I separated the algorithm into several substeps and **make each of them a function**.

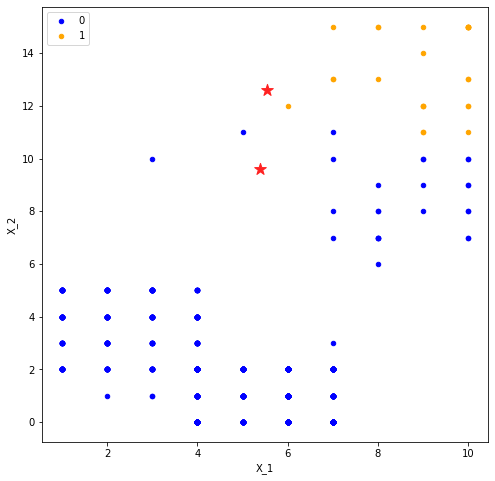
* Initial centroid assignment
  + I do this randomly with pick a float number between the max and the min of that attribute.
* Assign data points to the nearest cluster
  + I used Euclidean Distance as the measurement of distance and compare the distance between each data point and each cluster. I then assign the cluster with the shortest distance.
* Update centroid
  + recompute the new centroids by taking the average of a group of data points
* Stopping criteria
  + I calculated the change in SSE after each iteration and if the change in SSE is smaller than a threshold, I will stop the K-means clustering.

Orignal distribution:

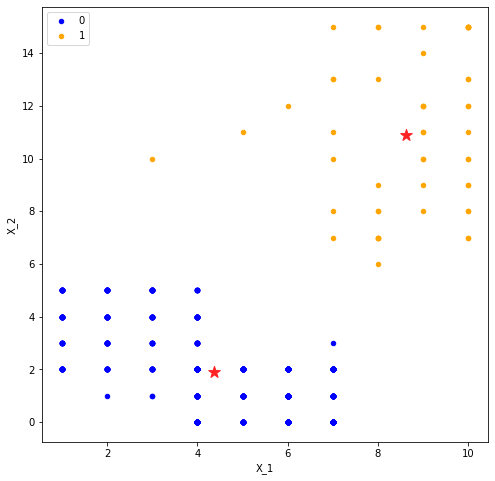
From the scatterplot, we can infer that there might be two groups for clustering. One is the customers with a large number of total credit cards and a large number of online visits. Let’s see if our K-means algorithm can successfully identify the potential clusters.



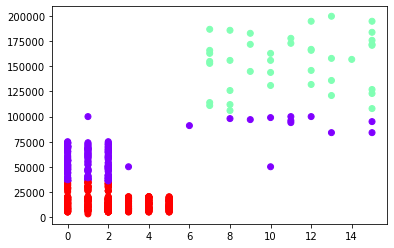
Initial random centroids:



After five iterations: The centroids move a lot and the two groups are clear!



I also tried clustering with K=3, but I still believed K=2 is more appropriate according to the plot.



# Package Used

[Link to Colab](https://colab.research.google.com/drive/1XFjv667yCZud7TRrkux6gfk-T3ldoabW)

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

from sklearn.cluster import KMeans

from sklearn.preprocessing import StandardScaler

from sklearn.decomposition import PCA

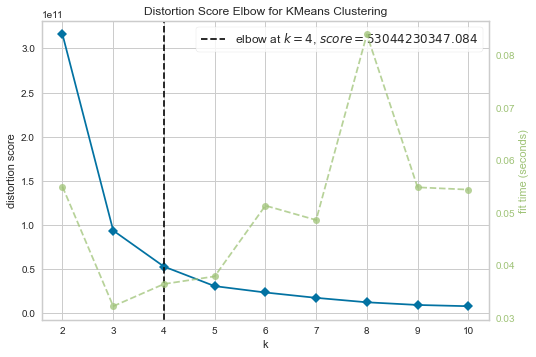
from yellowbrick.cluster import KElbowVisualizer

from sklearn.cluster import AgglomerativeClustering

from sklearn.cluster import DBSCAN

## Elbow Method and K-means

First, we need to decide which number of clusters is the most appropriate choice. I used the Elbow method to help me do the decision.



According to the Elbow method, I believe **K=4** is the most appropriate number of clusters.

I fit the package K-means model but since the dataset's dimension is still high, it is not good for our visualization.

Having a 2d or 3d dimensional dataset is easy to visualize the clustering result.

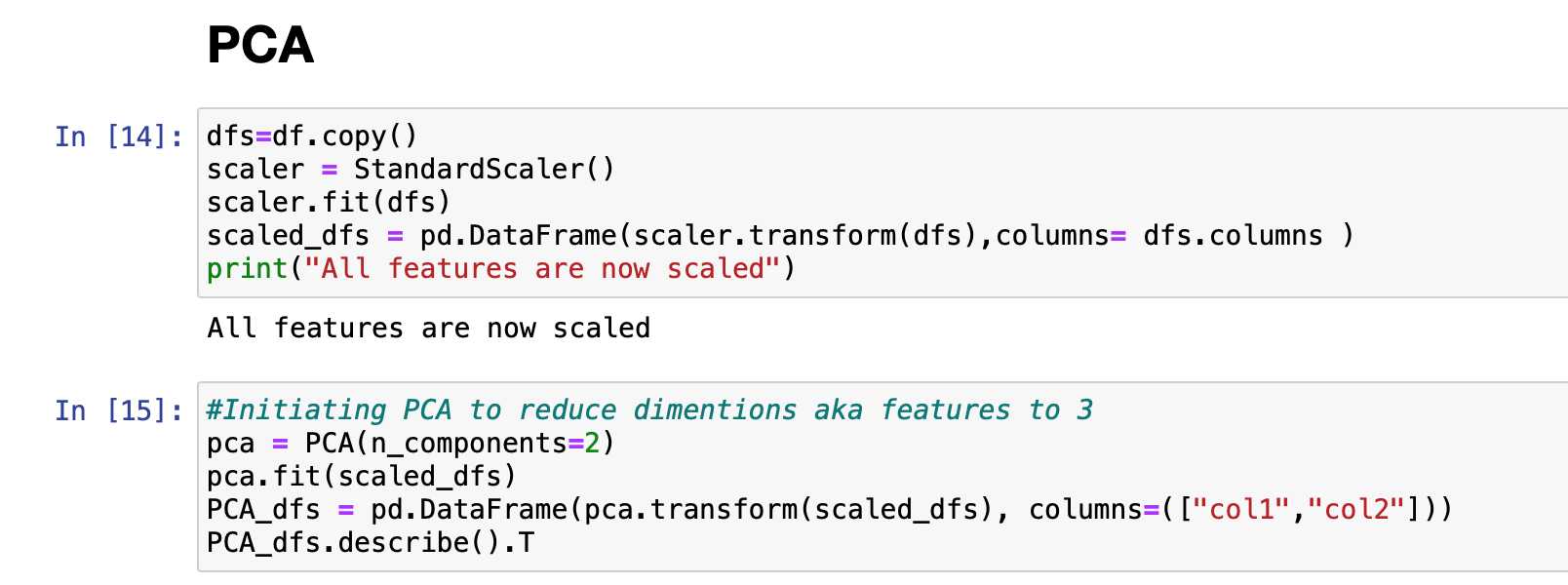
## PCA and K-means

Therefore, our next step is to condense our dataset to lower dimensions and one possible way to do that is through PCA. (I learned this from STA362: Multivariate Statistics class)

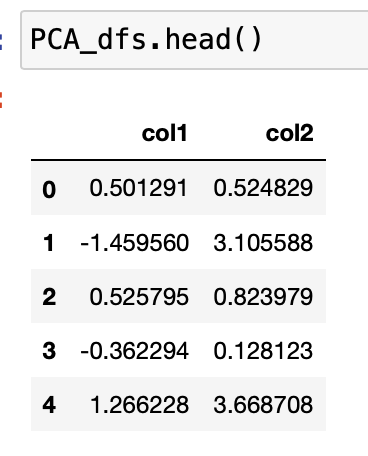
​Dimensionality reduction is the transformation of data from a high-dimensional space into a low-dimensional space so that the low-dimensional representation retains some meaningful properties of the original data, ideally close to its intrinsic dimension.

Principal component analysis (PCA) is the process of computing the principal components and using them to perform a change of basis on the data, sometimes using only the first few principal components and ignoring the rest.

I was trying to convert our dataset to only 2 attributes. And before applying PCA, **the data needed to be scaled.**

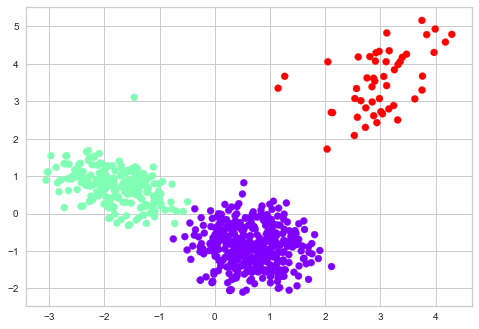


This is what the data frame looks like after scaling and the PCA technique. We only retained information by using two columns.



We then fit the K-means with K=3 and here is our visualization result. I’m very satisfied with the clustering because all the three groups look clear and separated. The green group and purple groups contain more points than the red group because they have different densities. We can explore more about this in our DBSCAN section since DBSCAN deals with different density groups.

K-means with PCA did a good job in data visualization.



## Hierarchical Clustering

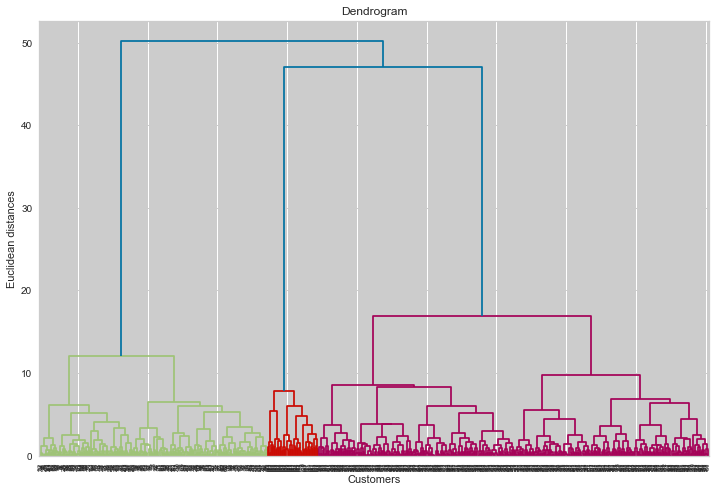
There are mainly two types of hierarchical clustering:

* Agglomerative hierarchical clustering
* Divisive Hierarchical clustering

I used Agglomerative hierarchical clustering here by merging the closest pair of clusters and repeating this step until only a single cluster is left.

Here is a dendrogram of this Agglomerative hierarchical clustering process and dendrogram is a tree-like diagram that records the sequences of merges or splits.

Now, we can set a threshold distance and draw a horizontal line (*Generally, we try to set the threshold in such a way that it cuts the tallest vertical line*). The number of clusters will be the number of vertical lines which are being intersected by the line drawn using the threshold. It is clear that using 3 groups is appropriate.

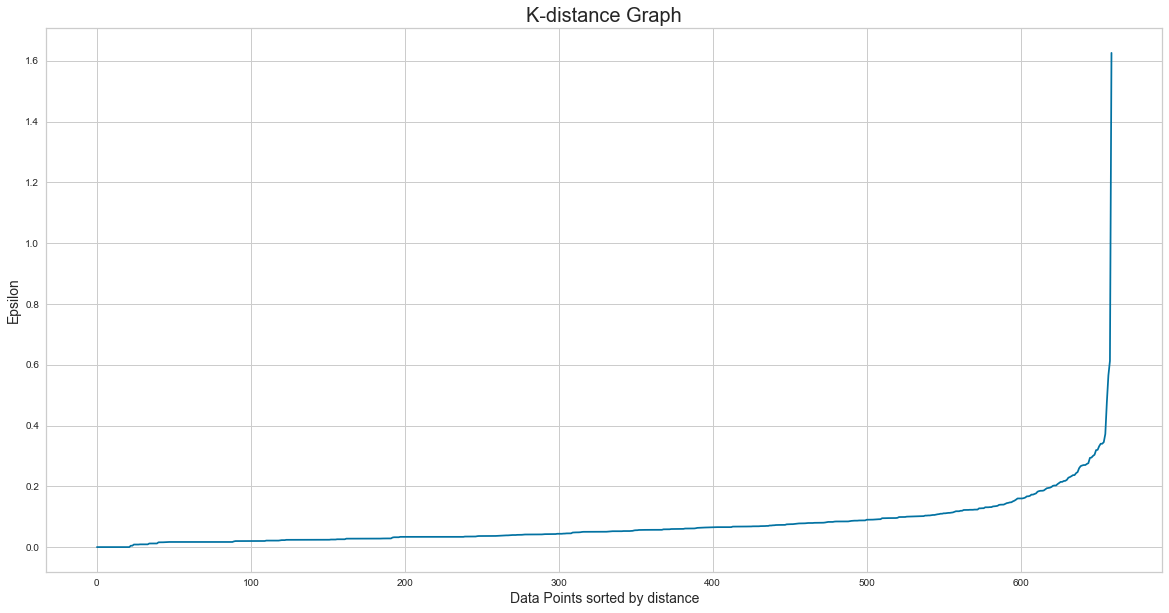


## DBSCAN

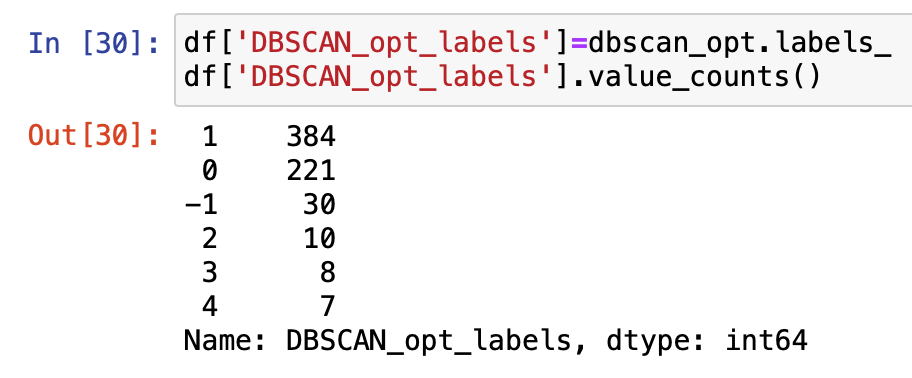
K-Means and Hierarchical Clustering both fail in creating clusters of arbitrary shapes and densities. However, it is common that different groups might have different shapes and densities. That’s why we need DBSCAN clustering. Also, DBSCAN clustering is robust to outliers. It also does not require the number of clusters to be told beforehand, unlike K-Means, where we have to specify the number of centroids.

DBSCAN requires only two parameters: epsilon and minPoints. Epsilon is the radius of the circle to be created around each data point to check the density and minPoints is the minimum number of data points required inside that circle for that data point to be classified as a Core point.

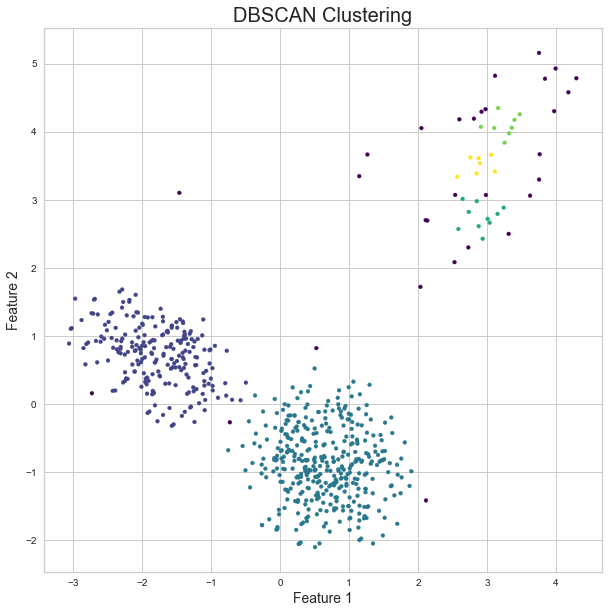
I first plot our K-distance graph and find the value of *epsilon*. It seems that DBSCAN(eps=0.3, min\_samples=6) is a good choice. The optimum value of *epsilon* is at the point of maximum curvature in the K-Distance Graph, which is 0.3 in this case. I choose 6 due to knowledge.



DBSCAN also separates noise from the dataset pretty well. Here, all the positive integers are the different clusters, and -1 is the noise. Let’s plot the results and see what we get.



It is clear that the two groups from the left corner got classified correctly. On the right top side, purple points are considered as noise, and yellow, light green, and deep green are also smaller clusters.



# Theory Exercise

There are two separate PDF files under the same DMP folder and they are my textbook exercises.

# Self-assessment

This is my first time working with an unsupervised learning algorithm and I learned a lot from it. There is no right or wrong answer to the clustering result so it needs more consideration for clustering evaluation. I think visualization is one of the best ways to evaluate clustering and I learned many plotting functions through this DMP exercise. I feel much more confident with handling this kind of data and I hope I can do more self-learn about this topic in the future.

# Reference

<https://www.analyticsvidhya.com/blog/2019/05/beginners-guide-hierarchical-clustering/>

<https://www.analyticsvidhya.com/blog/2020/09/how-dbscan-clustering-works/>

<https://stackoverflow.com/questions/17682216/scatter-plot-and-color-mapping-in-python>