OPTIMIZATION PROJECT REPORT

Name: Manjunatha B K (BT23RESCH12001)

Python library **"SciPy"** is used to solve the given Batch reactor problem. The system consists of three chemical species namely, A, B and C and the rate constants are given by k1 and k2. The Differential equations were provided to explain the kinetics of the batch reactor.

The objective of this project was to **find T(t), such that concentration of B is maximum at tf = 1 when, reaction is isothermal.**

Algorithm:

The objective was achieved by employing **ODEs and an optimizer**. The algorithm starts by defining the ODE system based on the given chemical reactions and temperature-dependent rate constants. Then it sets up an objective function to maximize the concentration of B at the end of the batch reaction.

Define the ODE System:

The reaction system is described by a set of three first-order ordinary differential equations (ODEs) representing the changes in concentrations of species A, B, and C over time.

The rate equations are based on the given reaction scheme $A \rightarrow B \rightarrow C$ and the temperature-dependent rate constants k1 and k2.

Objective Function:

The objective function is designed to be maximized. It takes a temperature (T) as input, integrates the ODE system using the **solve_ivp** function from **scipy.integrate**, and **returns the negative concentration of B at the final time (tf = 1).**

The negative value is used because the **minimize_scalar** function in **scipy.optimize minimizes the objective function**, so maximizing the negative concentration achieves the goal of maximizing the concentration of B.

Initial guess:

The optimization requires an initial guess. However, in this case, the minimize_scalar function does not explicitly require an initial guess. It uses a bound optimization approach, where the search is limited to the specified temperature range. Here the range was from 298K to 398K. So, **the initial guess would be 348K** that is the middle of the 298K and 398K.

Optimizer:

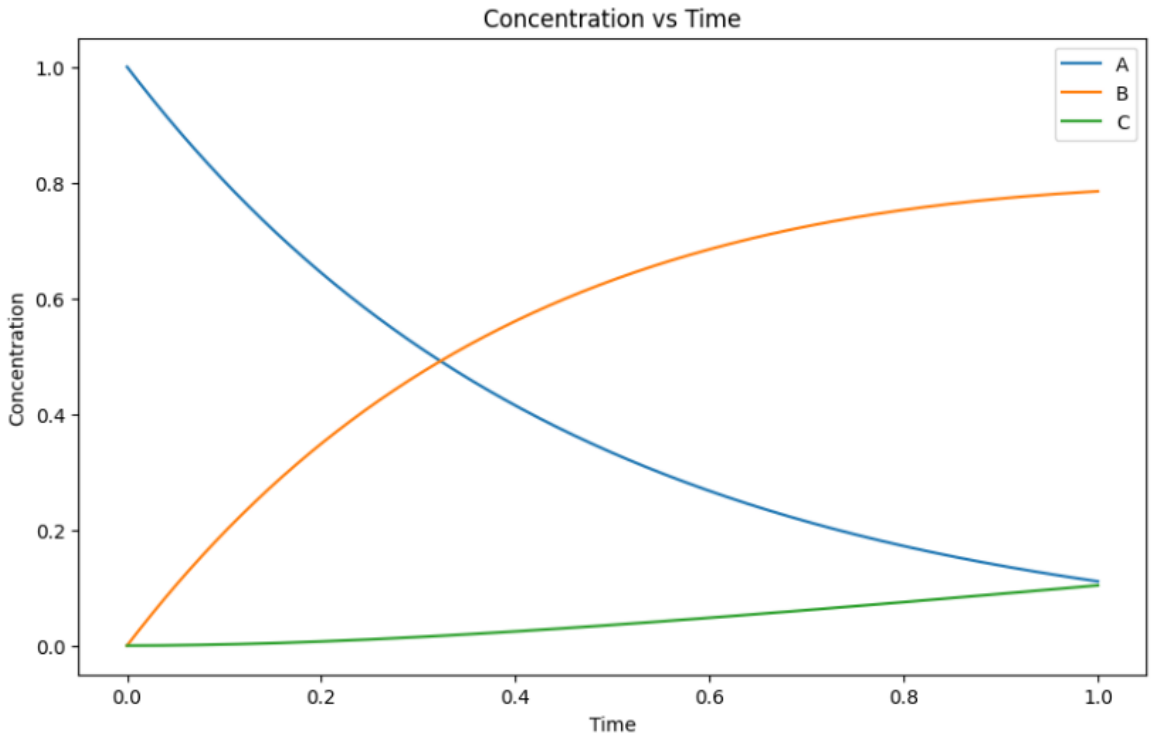
The minimize_scalar function from the scipy.optimize module is used for scalar optimization. It searches for the minimum (or maximum, in this case) of a scalar function of one variable within a specified range.

The optimization is performed over a temperature range from 298 K to 398 K, as specified by the bounds parameter.

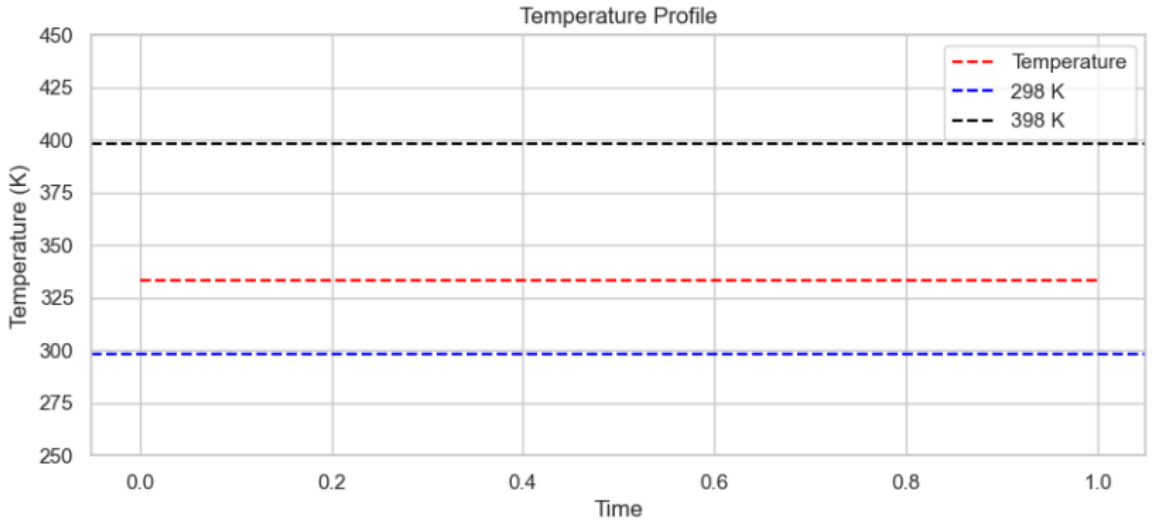
Results:

The final temperature at which concentration of B would be maximum is 333.0791 K.

The following plots are plotted to interpret the results.



Here we can see concentration of B is maximum at the time point tf =1



Also, the temperature profile was showed at which the concentration of B would be maximum. The temperature obtained was within the specified bounds.

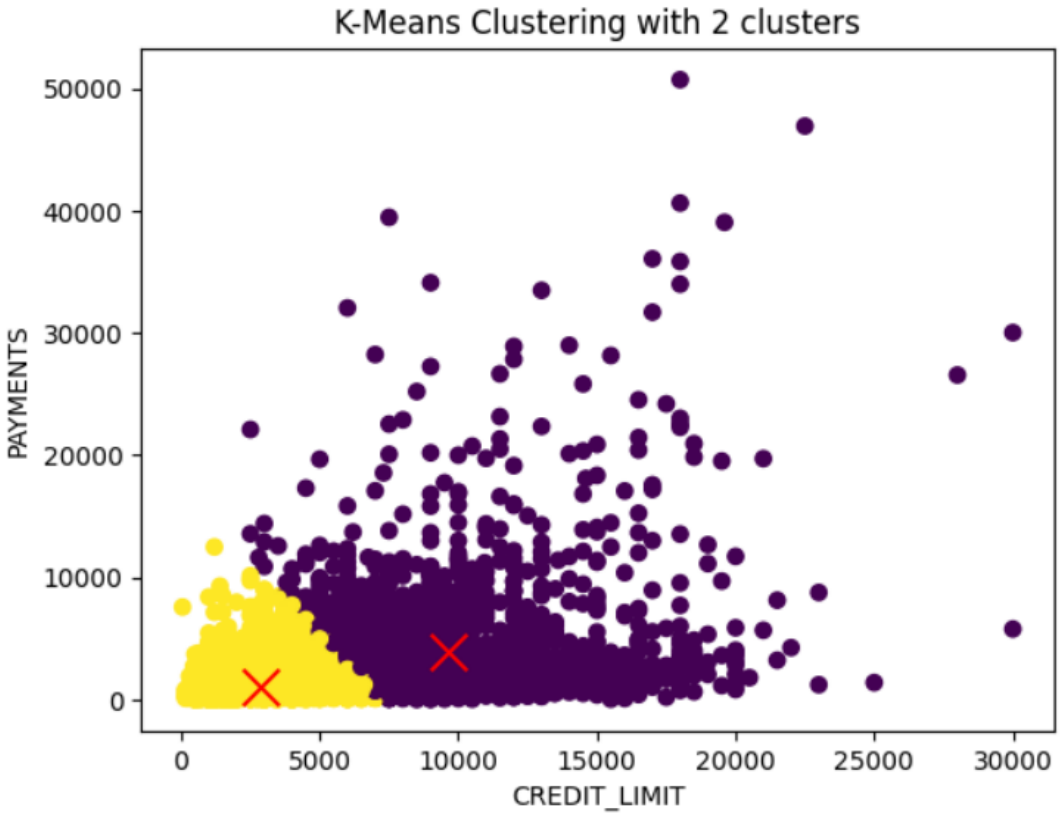
CLUSTERING PROJECT REPORT

Data given: Credit card data.

Algorithm used: K-Means Clustering.

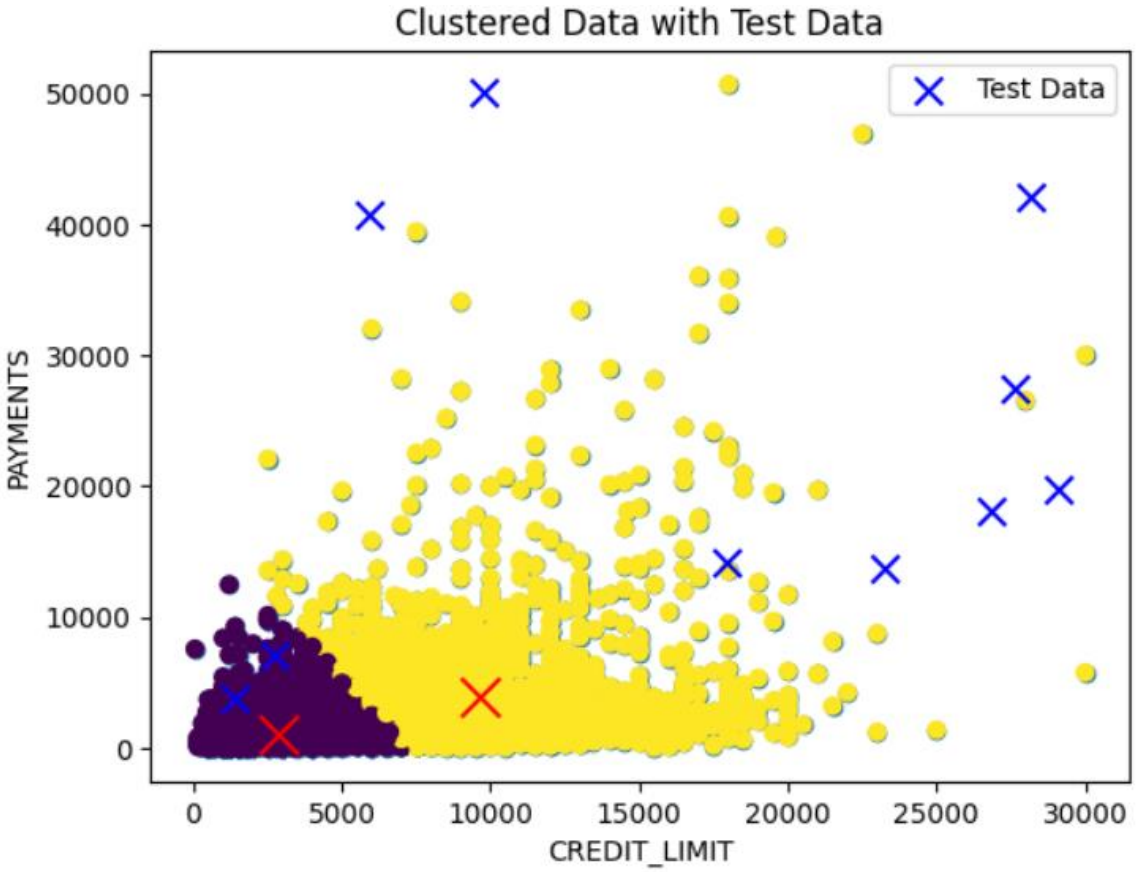
K means clustering was done with 2 clusters. Centroids for the clusters were calculated and the corresponding plot was plotted.

Question 1: Pandas library was used for data manipulation and analysis and "**Kmeans**" from sklearn.cluster used for K-Means clustering. Credit card data was read and two columns namely – **CREDIT_LIMIT** and **PAYMENTS** were used for clustering analysis. K means clustering model was initialized with 2 clusters and cluster labels assigned to each data point and centroids for the clusters were retrieved.



The above graphs show the two clusters formed, one in yellow color and one in the purple color. Centroids were marked as X in each cluster.

Question 2 and 3: Randomly 10 data points were generated within the range of clustered data. Labels were assigned to test data using the previously trained K-Means model based on the nearest cluster centers.



The above graph shows the label assigned to data points which were generated randomly.

Question 4: Complete given data was used at the end to train the K-Means model.

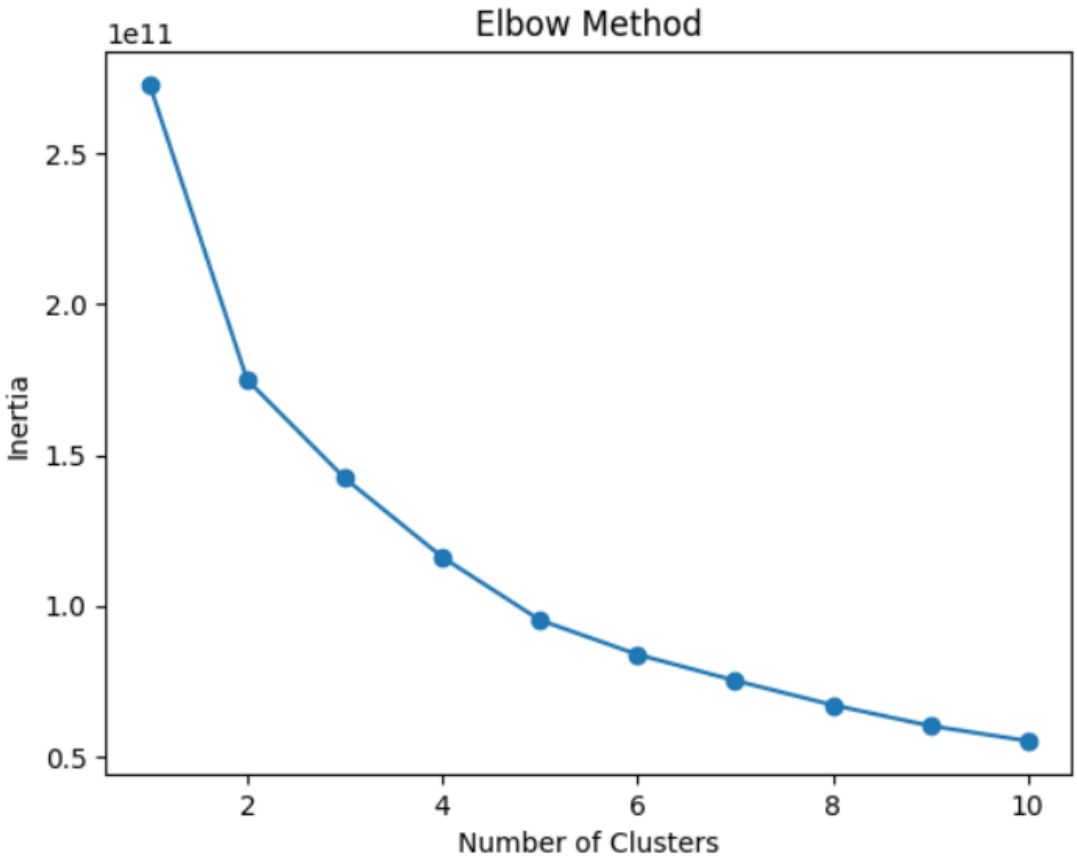
Data used: Creditcard1.
Clusters defined: 3
Max iterations: 300
Tolerance: 1e-5

Clusters labels were assigned, and centroids also calculated which were shown as outputs in jupyter notebook.

```
Cluster Labels: [0 1 1 ... 0 0 0]
Centroids: [[8.87422758e+02 2.60969213e+03 1.07941248e+03 6.00213908e+02
  1.15291122e+01]
 [3.11346602e+03 8.76084294e+03 2.46975823e+03 1.27485343e+03
  1.17509356e+01]
 [5.32562451e+03 1.20841139e+04 1.53141917e+04 4.20024151e+03
  1.18484848e+01]]
```

Question 5: Elbow method implementation

The Elbow method is a common method used to find the optimal number of clusters in a K-Means clustering algorithm. For loop iterated over 1 to 10 and model is fitted to the data. The inertia attribute of the K-Means model is then accessed and appended to the Inertia list. The inertia is a measure of how far the points within a cluster are from the center of that cluster. It is calculated as the sum of squared distances between data points and their assigned cluster centroids.



The Elbow Method is used to identify the optimal number of clusters by observing the inertia values across different values of K. As K increases, the inertia tends to decrease because smaller clusters allow the centroids to be closer to individual data points. However, beyond a certain point, the reduction in inertia becomes marginal, forming an "elbow" in the graph.

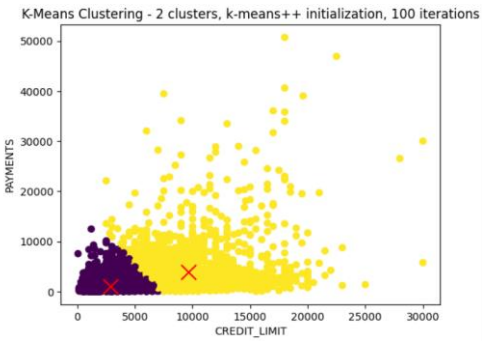
Here in this case the optimal number of clusters is found to be 2.

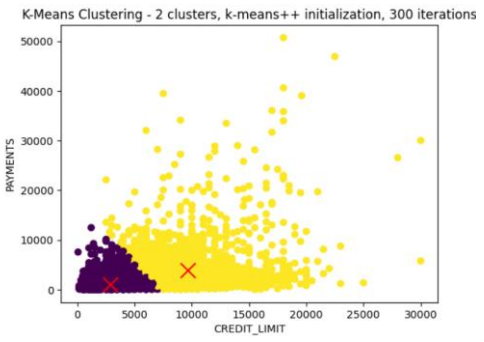
Further analysis:
Analysis was performed by varying the number of clusters, initialization of clusters and Maximum number of iterations.

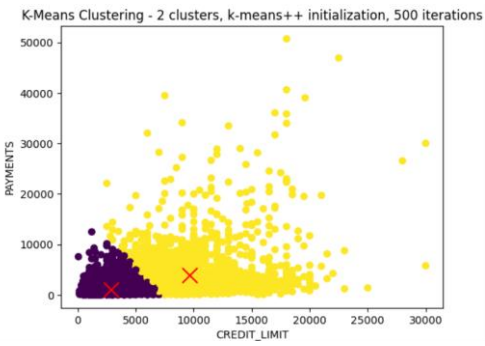
Cluster numbers tried: 2, 3, 4.
Initialization method: K_means++, Random.
Max iteration limits used: 100, 300, 500.

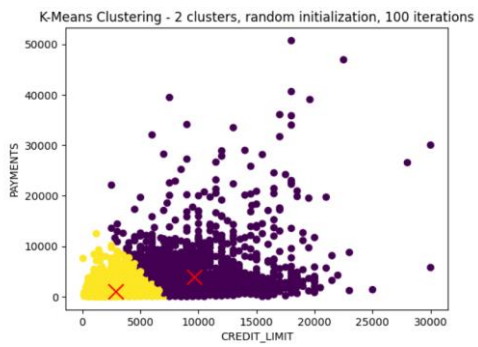
A nested loop was run to visualize the changes by varying the parameters with all possible combinations.

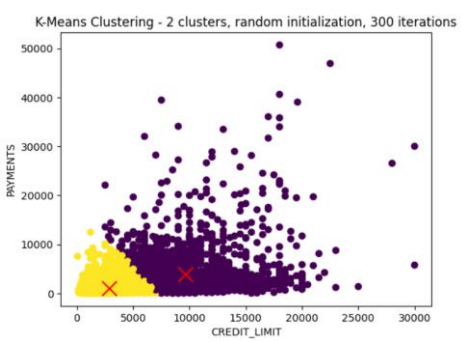
Number of Clusters: 2, Initializations: K-means and Random, Iterations: 100, 300, 500

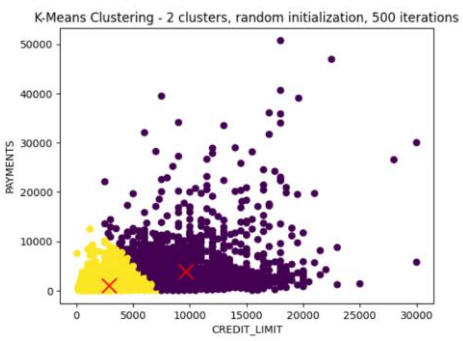




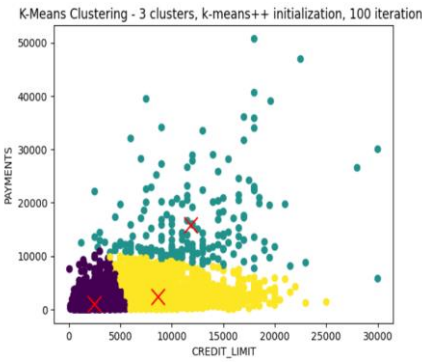


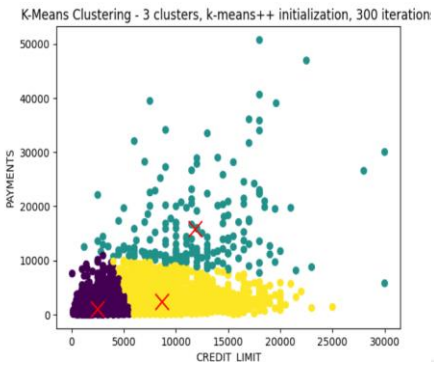


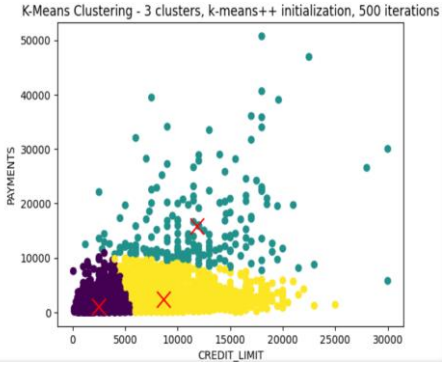


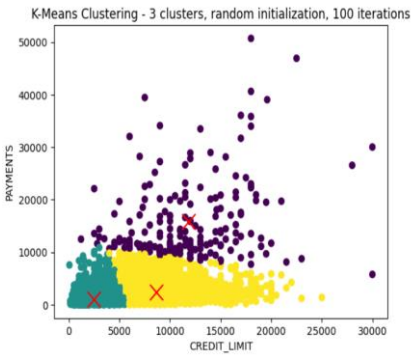


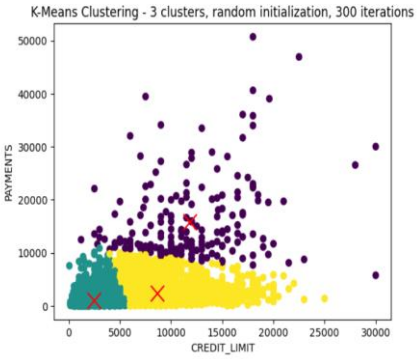
Number of Clusters: 3, Initializations: K-means and Random, Iterations : 100, 300, 500

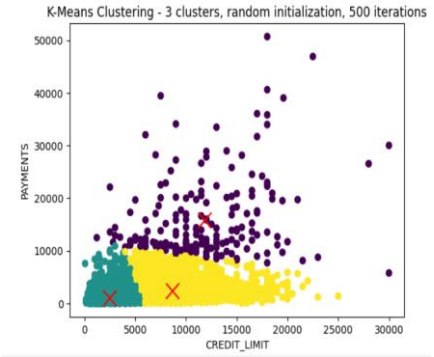




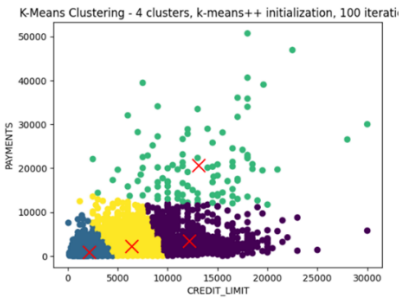


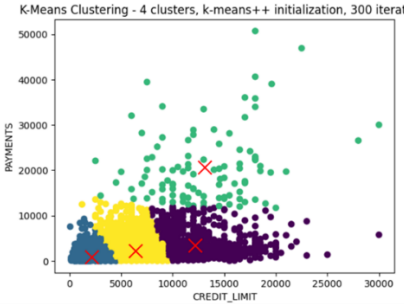


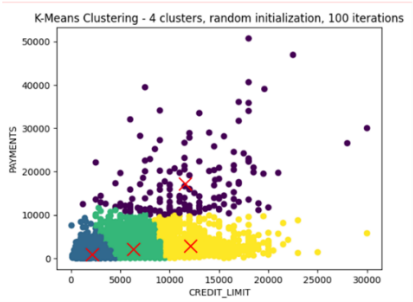


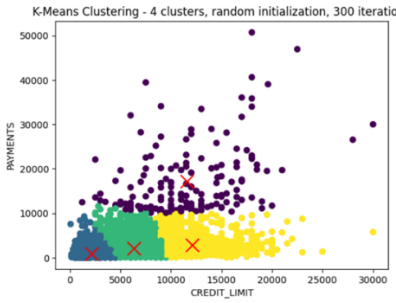


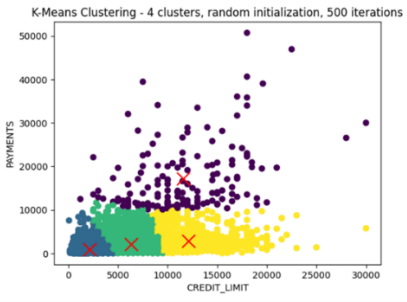
Number of Clusters: 4, Initializations: K-means and Random, Iterations: 100, 300, 500











Observations:

- Changing the initializations parameters such as K_means++ or Random mostly doesn't affect the clustering process and centroid calculation as long as the number of iterations is enough for data training.
- Increasing the number of iterations upon a certain limit, which is more than enough to train the data well, doesn't imply any further changes in the clustering process. However, it depends on the size of the data. The larger the data, the larger the number of iterations required.
- Changing the number of predefined clusters mainly impacts upon the clustering process. Upon changing this parameter, the whole data looks different.

Hence comparatively, Number of clusters plays a major role in clustering process as compared to initialization and Maximum number of iterations allowed.

FUZZY C_MEANS IMPLEMENTATION (PSEUDO-CODE)

Fuzzy C-means (FCM) is a clustering algorithm that assigns degrees of membership to each data point for multiple clusters. Unlike traditional hard clustering algorithms where each data point belongs to a single cluster, FCM allows data points to belong to multiple clusters with varying degrees of membership.

Algorithm:

The algorithm aims to minimize the total intra-cluster variance by iteratively updating cluster centroids and membership degrees based on the fuzziness parameter, which controls the level of cluster overlap.

Pseudo code was written and the corresponding. ipynb file attached along with the report.

Steps employed:

- Libraries which were necessary were imported – NumPy and Matplotlib.
- Initialization of Membership function Matrix: MF matrix represents the degree to which each data point belongs to each cluster.
- Update Centroids Function – centroids are updated based on the current MF matrix. Centroids are calculated using weighted average of data points of a particular cluster which in turn is given by squared membership values.
- MF matrix will be updated based on the current centroids and the fuzziness parameters. Again, distance between data point and centroids are calculated to update MF values in the MF Matrix.
- The main FCM algorithm iteratively updates centroids and the MF matrix as described above until convergence.

Varying parameters:

- Some of the parameters that can be varied are number of clusters, Fuzziness, Maximum iterations and tolerance values.
- For which the same nested loop can be used to see the effect of parameters variation on clustering process.