Credit Card Customer Segmentation

BITS Pilani Hyderabad Campus

CS F415 Data Mining Project

Tarimala Vignesh Reddy
f20210234@hyderabad.bits-pilani.ac.in

Maheshwar Bora
f20211505@hyderabad.bits-pilani.ac.in

Aayush Kumar Singh
f20210430@hyderabad.bits-pilani.ac.in

***Abstract*—In the competitive landscape of financial services understanding customer behavior through their transactional data is essential for FinTech companies and banks to make personalized marketing strategies and enhancing user engagement. This paper aims to explore techniques to segment customers into clusters depending on their credit card purchase details and identify the spending patterns of customers and how they correlate with their balance, cash paid, instalments taken over a period of time and other factors.**

By understanding the distinct needs and preferences of different customer segments, credit card issuers can create targeted marketing campaigns. For example, they can design specific promotions or offers that are more likely to resonate with a particular group, leading to increased engagement and customer satisfaction.

The paper aims to implement clustering algorithms like Partitioning Algorithms, Hierarchial Algorithms, Density-based Algorithms and using LLM-based embeddings to cluster. The results indicated that the KMeans model with Polynomial Kernel significantly outperformed other methods while, achieving the highest Silhouette Score. Three distinct clusters emerged from the data, correlating strongly with low, middle, and high-income groups. Each group displayed unique characteristics in terms of spending frequency, average transaction value, and types of purchases.

These findings allow credit card companies to tailor specific offers and promotions to each income segment. For instance, high-income customers, who frequently engage in luxury purchases, could be targeted with premium offers, while middle-income segments might respond better to cashback on essential goods. This targeted approach not only optimizes marketing expenditures but also enhances customer satisfaction and loyalty. By leveraging the insights gained from advanced clustering techniques, credit card issuers can refine their customer engagement strategies and ensure a more personalized banking experience, thus fostering a deeper connection with their clientele

Keywords: Clustering, Customer Segmentation, Large Language Models

I. Introduction

This project aims to segment customers into clusters depending on their credit card purchase details and identify the spending patterns of customers and how they correlate with their balance, cash paid, and instalments taken over a period of time and other factors. Through this clustering approach, we wish to identify groups of customers based on spending patterns using different clustering algorithms and also compare the performance of different algorithms on the given dataset.

The problem is intriguing as it can help accelerate customer analysis in the financial sector. Segmenting customers into different groups based on spending can help financial institutions like banks create marketing strategies aimed at a particular customer group. It can increase the effectiveness of marketing campaigns. Banks can also assess the likelihood of a customer defaulting based on recent credit spending patterns and take proactive measures. Clustering helps to understand the customers and dynamics of the market.

There are several factors which make the problem hard.

- **High Dimensionality of data**: Credit card data is multidimensional making it difficult to find meaningful clusters as the standard distance metrics like Euclidean and Manhattan distance fail in higher dimensions. Naive approaches like the KMeans algorithm usually fail due to the high dimensionality of the data.
- **Variablility and Heterogenity**: Customers spending patterns are not always consistent. This sometimes makes it difficult to form meaningful clusters.
- **Noise and Outliers**: Credit card data contains noise, errors and outlier data which are not representative of typical customer behaviour. Therefore it is essential to handle outliers at the preprocessing phase to make meaningful clusters.
- **Interpretability**: The clustering problem in general is ill-defined because there is no clear metric to judge the interpretability of the clusters formed. Based on the clusters formed we have to group customers into several categories like low income, middle income etc but such a boundary is not very easy to define.

Previous research papers focus on the traditional algorithms for clustering like KMeans, DBScan etc but we intend to use these algorithms with slight modification and also use Large Language Model embeddings of the dataset to perform clustering and compare them with other traditional methods.

The primary components of our approach to clustering credit card customers involved a multi-step process beginning with data collection and data preprocessing. The process we followed was to apply multiple clustering algorithms to the dataset and compare them using different metrics.

II. Related Work

This section reviews some research papers which have worked on the same problem of clustering customers based on their credit scores. We will try to analyze techniques used in various papers.

I. Karo et al. [1] used the K-Means algorithm to cluster customers. They used the Euclidean distance metric to measure the proximity of the clusters. They computed the the Within cluster sum of squares for all the clusters for different number of clusters.They followed a similar approach to preprocessing by handling missing values and specifying a bound for outlier data. The only shortcoming of the approach followed by this paper is they did not use other kernels in place of Euclidean Distance. But the conclusions finally drawn were similar that the clusters formed were not coherent as the Silhouette Scores obtained were never close to 1 in their research as well.

In the paper by S. Tripathi et al. [2] the authors used several conventional clustering algorithms and performed a comparative analysis between the algorithms. They applied K-Means and Hierarchial(Agglomerative and Divisive) Clustering on the customer relationship management dataset. They used a wide variety of indices like the Silhouette Index, Davies Bouldin Index and WCSS to determine the performance of the algorithms. They concluded that K-Means gave better performance on a large sample of data while Hierarchial Clustering performed better on a smaller sample of data. The only gaps in the research were not to test other variants of K-Means using different kernels and also not testing the more popular conventional clustering algorithms which perform well on dense datasets like DBSCAN.

A. Pethukova et al. [3] explored the use of Text-Embeddings in cluster analysis. The paper investigates the use of textual embeddings including Transformer embeddings and Large Language Model Embeddings and their performance in clustering. They performed experiments using LLAMA and Falcon model embeddings. They concluded that LLM embeddings are not immensely better than conventional algorithms and require careful preprocessing and formatting of the plain-text before generating the text embeddings.

III. Approach and Methodology

A. Dataset Properties

To solve this problem we need a dataset with customers spending patterns over a period of time. The dataset used in this paper contains several properties related to customer credit details and spending patterns. The dataset summarizes the usage behaviours of around 9000 credit card holders over a period of 6 months. It contains 18 behavioural variables including:

1) **Cust ID**: Identification of Credit Card holder (Categorical)
2) **BALANCE**: How frequently the Balance is updated, score between 0 and 1 (1 = frequently updated, 0 = not frequently updated)
3) **PURCHASES**: Amount of purchases made from account
4) **ONEOFF_PURCHASES**: Maximum purchase amount done in one-go
5) **INSTALLMENTS_PURCHASES**: Amount of purchase done in instalments
6) **CASH_ADVANCE**: Cash in advance given by the user
7) **PURCHASES_FREQUENCY**: How frequently the Purchases are being made,score between 0 and 1 (1 = frequently purchased, 0 = not frequently purchased)
8) **ONEOFFPURCHASESFREQUENCY**: How frequently Purchases are happening
9) **PURCHASESINSTALLMENTSFREQUENCY**: How frequently purchases in installments are being done (1 = frequently done, 0 = not frequently done) in one-go (1 = frequently purchased, 0 = not frequently purchased)
10) **CASHADVANCEFREQUENCY**: How frequently the cash in advance being paid
11) **CASHADVANCETRX** : Number of Transactions made with "Cash in Advanced"
12) **PURCHASES_TRX** : Number of purchase transactions made
13) **CREDIT_LIMIT** : Limit of Credit Card for user
14) **PAYMENTS** : Amount of Payment done by user
15) **MINIMUM_PAYMENTS** : Minimum amount of payments made by user
16) **PRCFULLPAYMENT** : Percent of full payment paid by user
17) **TENURE** : Tenure of credit card service for user

B. Algorithms Used

1) K-Means

The K-Means clustering algorithm is a popular unsupervised clustering algorithm. It partitions data points into K clusters in which each observation belongs to a cluster with the nearest mean. The set of n data points is denoted by $\{x_1, x_2, \ldots, x_n\}$ and the set of K clusters is denoted as $\{c_1, c_2, \ldots, c_k\}$ The aim of the K-Means algorithm is to minimise the within-cluster variance which is the Euclidean distance between the data point and the nearest centroid. The objective function of means is given as follows

$$J = \sum_{j=1}^{k} \sum_{x_i \in C_j} \|x_i - \mu_j\|^2 \tag{1}$$

The K-Means algorithm follows the following steps:

1) **Initialisation**: Select K initial centroids from the data points
2) **Assignment**: Each Data point is assigned to the nearest centroid. In traditional K-Means, the distance is calculated using Euclidean Distance.

$$C_j = \{x_i : \|x_i - \mu_j\|^2 \leq \|x_i - \mu_l\|^2, \forall l, l \neq j\} \tag{2}$$

According to Equation 2 each data points distance is calculated with respect to the K clusters before assigning it to the the cluster which is closest to the data point.

3) **Update Centroid**: New centroids of a cluster c_k are calculated as the mean of the data points belonging to the cluster.

$$\mu_j = \frac{1}{|C_j|} \sum_{x_i \in C_j} x_i \tag{3}$$

4) **Iteration**: Keep repeating steps 2 and 3 until the cluster centroids do not change significantly which indicates that the algorithm has converged.

Given the high dimensionality of the dataset used, K-Means using Euclidean distance is not always the best-performing model. So contrary to previous papers we also used different kernels to evaluate the performance of K-Means using other distance metrics.

The kernel function $K(x, y)$ is defined as:

$$K(x, y) = \langle \phi(x), \phi(y) \rangle$$

where ϕ represents a mapping of the input vectors into a higher-dimensional feature space, and $\langle \cdot, \cdot \rangle$ denotes the dot product in that space.

Kernels used on the given dataset include:

- **Gaussian (Radial Basis Function) Kernel:**

$$K(x, y) = \exp\left(-\gamma \|x - y\|^2\right)$$

The Gaussian kernel, characterized by parameter γ, measures the similarity or closeness of two points x and y by their Euclidean distance.

- **Tanh (Hyperbolic Tangent) Kernel:**

$$K(x, y) = \tanh(\alpha x^T y + c)$$

This kernel function, involving parameters α and c, is used in neural networks as an activation function and in SVMs as a similarity measure.

- **Polynomial Kernel:**

$$K(x, y) = (x^T y + c)^d$$

The polynomial kernel computes the similarity as a polynomial function of the dot product of vectors x and y, with degree d and offset c.

2) DBSCAN

DBSCAN which stands for Density-Based Spatial Clustering of Applications with Noise, is a clustering algorithm that is known for handling noise and outliers and also dealing with clusters of multiple shapes and sizes. DBSCAN uses two key parameters

- ϵ: Defines the radius of a neighbourhood around a point.
- MinPts(Minimum Points): The minimum number of points in the ϵ neighbourhood to form a dense region

Points are classified into 3 categories based on the number of points in the ϵ neighbourhood around it:

- **Core Point**: A point which has at least MinPts within its ϵ neighbourhood
- **Border Point**: A point that has fewer than MinPts in its ϵ neighbourhood, but lies in the neighbourhood of a core point.
- **Noise Point**: A point which is neither a Core point nor a Border Point.

A point p is **directly density reachable** from a point q if p is within the ϵ neighbourhood of q and q is a core point. A point p is **density reachable** from a point q if there exists a chain of points $p_1, p_2, \ldots, p_n$ where $p_1 = q$ and $p_n = p$ and each p_{i+1} is directly density reachable from p_i.

The clustering process is as follows

- **Identification of Core Points:** For each point in the dataset, determine if it is a core point by checking if there are at least MinPts within its ϵ-neighbourhood.
- **Formation of Clusters:** Connect each core point to its directly density-reachable points, thus forming a cluster. Extend this connection transitively to include density-reachable points, effectively merging density-connected components into a single cluster.
- **Handling Noise:** Label points that are neither core nor border points as noise.

3) Clustering Using Large Language Models

One of the relatively new methods used was to try and leverage the use of Large Language model embeddings to see if they can capture patterns in data which are ignored by using the dataset as it is. The goal is to compare the performance of the K-Means algorithm on the given dataset and the Large Language Model embeddings of the data to see how they perform relative to each other. *Sentence_transformer* library is used to gain access to the large language models used in this paper. The clustering process is as follows:

- Each feature of the dataset is encoded into a sentence. For example, if the balance of the user is 15000 units the sentence encoded version is *USER_BALANCE=15000*
- The sentence encoding of each feature is concatenated to form the sentence encoding of the data
- the same is repeated for each row in the dataset and is then passed and then used as an input to the sentence_transformer which returns embeddings for the entire data frame.
- Traditional algorithms like K-Means are applied to these LLM embeddings to generate clusters.

4) Agglomerative Hierarchical Clustering

Agglomerative Hierarchical clustering is a bottom-up algorithm used to build clusters in the data. Initially, each data point belongs to its own cluster and as the algorithm proceeds pairs of clusters are merged to become one cluster which contains all the data points. It is beneficial for exploratory data analysis.

$X = \{x_1, x_2, \ldots, x_n\}$ represents the set of n data points. A distance metric $d : X \times X \to \mathbb{R}$ is defined which is used to define the distance between clusters to merge them. A linkage criterion is also defined to mention the distance between clusters as a function of pairwise distances between points in the clusters. Some of the commonly used linkage criteria are

- **Single linkage:** The distance between two clusters is defined as the shortest distance between any two points in the clusters.
- **Complete linkage:** The distance between two clusters is defined as the longest distance between any two points in the clusters.
- **Average linkage:** The distance between two clusters is defined as the average distance between each point in one cluster to every point in the other cluster.

The clustering process proceeds as follows:

1) **Initialization**: n clusters are declared with each point belonging to its own cluster i.e., $C_i = \{x_i\}$ for $i = 1, 2, \ldots, n$. Also, define a linkage criterion which is used to identify the distance between clusters based on the pairwise distances between points of the two clusters.
2) **Find Closest CLusters**: Identify the pair of clusters C_i and C_j that are closest based on the distance metric d, that is, find the pair (C_i, C_j) such that: $d(C_i, C_j) = \min_{k \neq l} d(C_k, C_l)$
3) **Merge Closest Clusters**: the closest clusters are merged to form a single new cluster $C_k = C_i \cup C_j$. The clusters C_i and C_j are now replaced by a single cluster C_k
4) **Update Distance Matrix**: Change the distance matrix to reflect the formation of the new cluster and find its distance to the remaining clusters using the linkage criterion.
5) **Iterate**: Repeat steps from 2-4 until all points belong to a single cluster.

IV. Experiments

A. Dataset Preprocesing

1) Removal of Nominal Data

Nominal Features like CUST_ID were removed from the dataset as they can distort the results of the clustering algorithms due to the fact that they can't be represented on a coordinate-plane and any distance metric used on the nominal attributed are meaningless.

2) Handling Missing Values

The dataset was first checked for missing values. Two Features, namely CREDIT_LIMIT and MINIMUM_PAYMENTS had nan values. The Median filling strategy was used where each 'nan' value was replaced with the median of the remaining values corresponding to the feature. Median strategy was chosen as it is generally more resistant to outliers than using mean or mode.

3) Outlier Detection and Removal

After plotting the Box Plots of the continuous features in the dataset as part of Exploratory Data Analysis, it was observed the data was skewed and some features had a lot of data above the 75% whisker of the plot. Since many conventional clustering techniques don't work well with data having outliers, it was decided to remove the outliers in the data. The data was preventing from exceeding certain upper and lower bounds by specifying them as follows

$$\text{lower\_bound} = Q1 - x \times \text{IQR}$$

$$\text{upper\_bound} = Q3 + x \times \text{IQR}$$

Here $Q1$ is the 1st Quartile, the value below which 25% of the data lies and $Q3$ is the 3rd Quartile, the value above which 75% of the data lie. IQR or Inter Quartile Range represents the distance between $Q1$ and $Q3$. After multiple experiments x was chose as 3 which ensured the most even distribution of the features while classifying the minimum amount of the data as outliers. Any value below $lower\_bound$ was bound to it and above $upper\_bound$ was also bounded by it. So all the features had values between their respective $lower\_bound$ and $upper\_bound$.

4) Standardization

The features of the dataset are not following the same scale, For example BALANCE has values of the order 10^3 but BALANCE_FREQUENCY has values of the order 10^0. This can cause certain features with large values dominating distance metrics like Euclidean and Manhattan Distance. Thus all the continuous features of the data were standardized to follow a normal distribution with mean 0 and variance 1 using the equation:

$$z = \frac{(x - \mu)}{\sigma} \tag{4}$$

5) Preprocessed Dataset

The Preprocessed dataset has 8950 samples and 17 features(after the removal of CUST_ID) All features are continuous and have a mean 0 and variance 1. The preprocessed dataset has no missing values

B. Evaluation Metrics

1) Silhouette Score

Silhouette Score was used to measure the cohesion of the clusters formed. The silhouette score value for each sample is always between -1 and 1. It is measured as:

$$s = \frac{b - a}{\max(a, b)}$$

a: mean distance between a sample and all other points in the same cluster
b: mean minimum distance between the sample and all points in any other cluster, of which the sample is not a part.

High Silhouette Scores indicate that clusters are tightly formed. Silhouette score were considered to identify the optimal number of clusters by calculating the values for clustering algorithms on different values of K(number of clusters).

2) Within Cluster Sum of Squares(WCSS)

WCSS measures the variance within each cluster. The lower the WCSS the more tight the clusters formed are as the points are close to the centroid. It is calculated as :

$$\text{WCSS} = \sum_{i=1}^{k} \sum_{x \in C_i} \|x - \mu_i\|^2$$

where k denotes the number of clusters, C_i represents the set of all data points in cluster i, x is a data point in cluster C_i and μ_i is the centroid of the data points in cluster C_i.

WCSS value is also used to find the optimal number of clusters using the ELBOW method. WCSS values are plotted on the y-axis against the number of clusters on the x-axis. Normally WCSS decreases as the number of clusters increases. However, the reduction in WCSS becomes marginal at a certain point, forming an "elbow" shape in the graph. This point represents where increasing the number of clusters does not lead to significant improvements in the compactness of the clusters. The elbow point represents the optimal number of clusters.

C. Experimental Setup

1) Data Mining Techniques and Algorithms

Our study employed a variety of clustering techniques t explore data segmentation and pattern discovery from differer perspectives. The clustering algorithms selected were:

- **K-Means CLustering**: Known for its simplicity an efficiency, K-means was used to partition the dataset int k distinct, non-overlapping clusters based on their mea distance from the centroid of the cluster. The number o clusters was determined using the Elbow method, which helped us identify the point where increasing the number of clusters does not provide significant value addition.
- **Hierarchical Clustering**: This method was utilized to build a hierarchy of clusters using a tree-based structure. It helped us understand the data at various levels of granularity. The dendrogram produced by this technique was used to decide the cut-off level for cluster formation.
- **DBSCAN**: DBSCAN was chosen for its ability to find arbitrarily shaped clusters and its robustness to outliers. It requires two parameters eps , the maximum distance between two samples for one to be considered as in the neighborhood of the other and min_samples, the number of samples in a neighborhood for a point to be considered as a core point.
- **LLM Embedding Based Clustering**: Leveraging the advancements in language models, we transformed our data using an LLM to obtain dense embeddings that capture deep semantic similarities. Hierarchical clustering was then applied to these embeddings to group data points with nuanced similarities that traditional methods might overlook.

2) Hyperparameters

- **K-means:** The optimal number of clusters k was determined after multiple iterations using the Elbow method. Additionally, we tested several initialization methods and found that the 'k-means++' initialization provided the most stable centroids.
- **DBSCAN:** The ϵ (eps) and *min_samples* parameters were optimized using reference from previous research papers and brute force techniques to ensure maximum cluster purity without sacrificing the inclusion of core points.
- **LLM-Based Clustering:** The dimensionality of embeddings and the linkage criteria (single, complete, average, ward) in hierarchical clustering were varied to study their impact on the quality of clustering.

V. Results

This section displays the results and the performance metrics of the various algorithms used in the paper and optimal number of cluster calculation

A. K-Means

K-means algorithm was implemented using Euclidean Distance metrics, Polynomial Kernel, Gaussian Kernel and tanh kernel. The graphs below show the silhouette scores and WCSS of the different variants of K-Means implemented and their analysis. As shown in the figure 1 The silhouette score

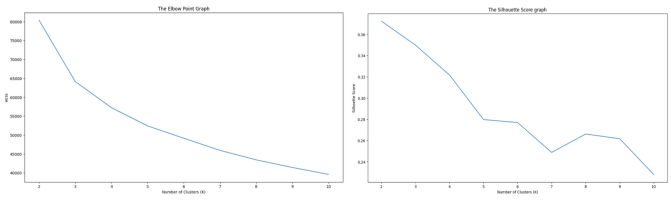

Fig. 1. Kmeans Algorithm using Euclidean Distance

starts high at k=2 and consistently decreases as the number of clusters increases, with minor fluctuations. The decrease is particularly sharp between k=2 and k=3, suggesting that the cohesion of clusters is significantly worse for k=3 than for k=2. The lack of any noticeable peaks beyond k=2 might indicate that the data does not naturally form into more than two high-quality clusters when no kernel transformation is applied.

The WCSS graph shows a smooth and consistent decline as the number of clusters increases. A sharp decrease in the slope of the graph is visible after K=3 as diminishing returns in WCSS are visible beyond that point. So from the above graph we can conclude that the optimal number of clusters for KMeans without any kernels used is 3 and the highest silhouette score for K=3 is 0.35

The three graphs 2, 3 and 4 Show performance of the three different kernels. From 2 we can observe that in 2 the WCSS value does not always decrease which in general indicates that the tanh kernel is not suitable for this dataset mostly owing due to the presence of outliers in the dataset. Therefore we can't conclude the optimal number of clusters

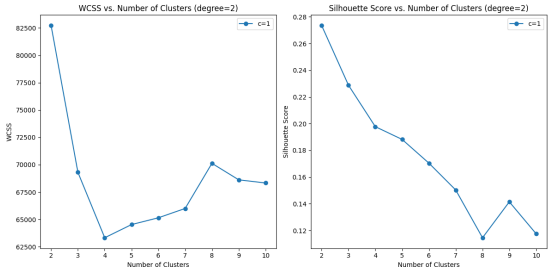

Fig. 2. Kmeans Algorithm using Tanh kernel

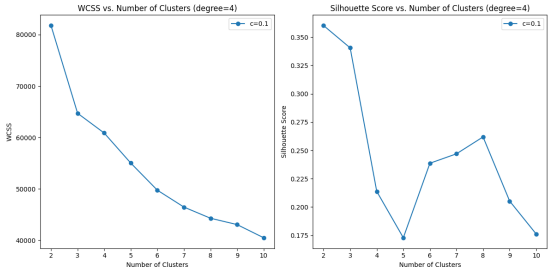

Fig. 3. Kmeans Algorithm using Gaussian kernel

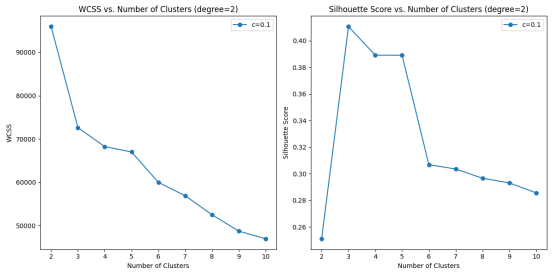

Fig. 4. Kmeans Algorithm using Polynomial kernel

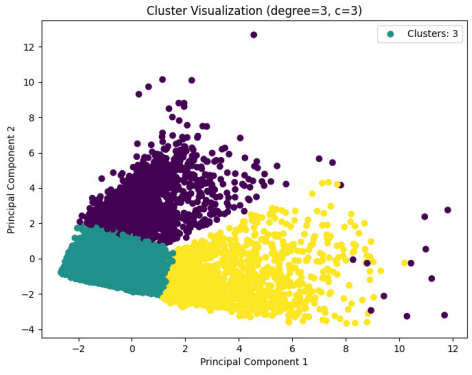

Fig. 5. Kmeans Cluster visualization

from this algorithm. On the other hand the Gaussian kernel displayed in 3 shows a very smooth trend of decreasing WCSS after K=3. It also shows a Silhouette Score of 0.33 for K=3 which is performing poorly compared to using No kernel. The best performing variant of KMeans was using the Polynomial Kernel with power 2. WCSS graph from 4 shows that K=3 is the optimal number of clusters where the clusters are tightly packed together. The silhouette score is also the highest for K=3 and shows a decreasing trend beyond K=3. The highest Silhouette score is 0.43. The trends of the graphs also show that Gaussian and tanh kernel don't result in the formation of meaningful clusters as with increasing number of clusters their performance in clustering decreases. Therefore we can conclude that Only Euclidean distance and Polynomial Kernel are suitable for applying Kmeans on this Dataset.

Figure 5 Shows the clusters formed on a reduced 2-d space on applying Kmeans with Polynomial Kernel using the optimal hyperparameters(degree=3 and c=3). 3 well formed clusters are clearly visible with the purple, green and yellow colours. Given below is the analysis of the customers based on the clusters they belong to. This analysis has been performed by calculating the effect of each feature on the principal components PC1 and PC2. PC1 has strong positive loadings on PURCHASES, PURCHASES_TRX, ONEOFF_PURCHASES, and INSTALLMENTS_PURCHASES while on the other hand PC2 is positively associated with features like CASH_ADVANCE, CASH_ADVANCE_TRX, and BALANCE.

- Green Cluster: Positioned towards the lower-left side on PC1 and PC2, this cluster represents customers with lower spending and cash advance activities. This could be interpreted as customers who are conservative in their credit card usage.
- Yellow Cluster: Extending upwards along PC1, these customers likely have higher spending behaviors compared to the yellow cluster. They might represent a segment that uses their credit cards more actively for purchases.
- Violet Cluster: Given that PC1 relates to spending behaviors, such as total purchases and the number of purchase transactions, a low score on PC1 suggests these customers have lower spending on their credit cards. They make fewer or smaller purchases, indicating they might use their credit cards conservatively for transactions or have a preference for paying with other methods. With PC2 associated with cash advances, balance, and minimum payments, a high score on this component indicates these customers are more likely to use their credit card for cash-related transactions or carry a balance. They might use cash advances more frequently and may only pay the minimum amount due each period, potentially indicating a different kind of financial strain or a preference for liquidity over purchase financing.

B. Hierarchial Clustering

As part of Agglomerative Hierarchial clustering analysis we plotted the WCSS and Silhouette Score values for different

values of k and also tried to identify the clusters formed using a dendogram.

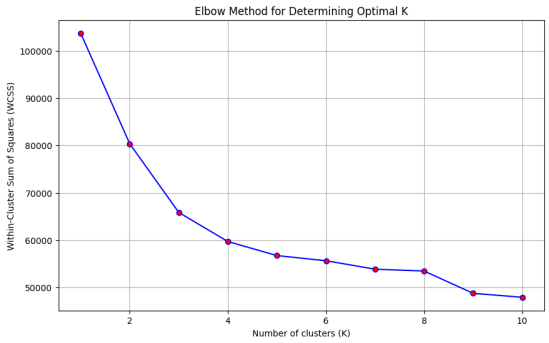

Fig. 6. Hierarchial Clustering WCSS vs K

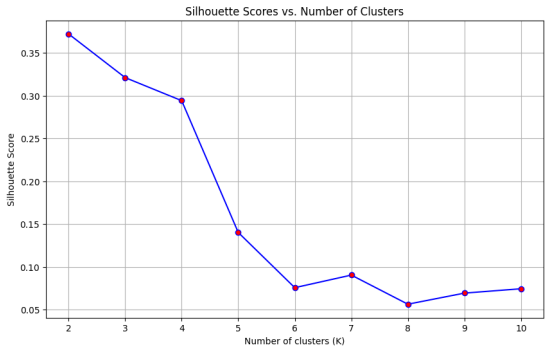

Fig. 7. Hierarchial Clustering Silhouette Score vs K

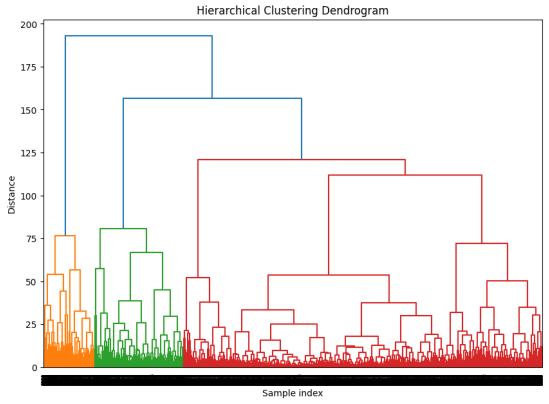

Fig. 8. Hierarchial Clustering Dendogram

The figure 6 shows that the WCSS value for the clustering model decreases rapidly till K=3 and then a sharp kink in the graph is observed which marks the Elbow point. This is reflected in the Silhouette score as we can see a sharp decline in score after 4 clusters. Hierarchial clustering gives the most number of distinct clusters without major drop in silhoutte score despite giving a slightly lower score of 0.37 compared to the best KMeans variant. The model with 3 optimal clusters was used to plot a dendogram to identify the clusters and display them visually.

The dendrogram in figure 8 displays how each cluster is composed by merging smaller clusters together. At the bottom, each individual data point is considered its own cluster, and as you move up, the clusters merge. The height of the merge represents the distance (or dissimilarity) at which clusters are combined.

There are large clusters combined at a high distance, which suggests there is a significant difference between these groups, indicating well-separated clusters at that level. The largest distances at which merges occur are highlighted by the long vertical lines, especially the top blue and red lines. The biggest vertical distance without a horizontal line intersecting it (likely the blue line in this case) suggests the most distinct separation of clusters.

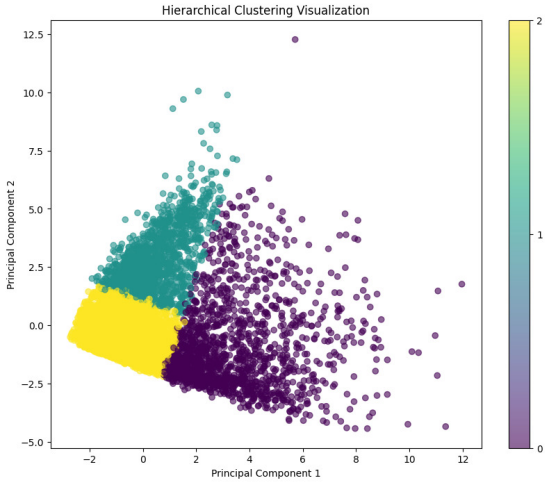

Fig. 9. Hierarchial Clustering cluster plots

From Figure 9 we can observe that there is a significant spread along PC1 and PC2, suggesting that these components capture substantial variance within the data. PC1 seems to have a more extensive range, indicating that it captures a larger portion of the variance.

- Yellow Cluster: Positioned towards the lower-left side on PC1 and PC2, this cluster represents customers with lower spending and cash advance activities. This could be interpreted as customers who are conservative in their credit card usage.
- Blue Cluster: Given that PC1 relates to spending behaviors, such as total purchases and the number of purchase transactions, a low score on PC1 suggests these customers have lower spending on their credit cards. They make fewer or smaller purchases, indicating they might use their credit cards conservatively for transactions or have a preference for paying with other methods. With PC2 associated with cash advances, balance, and minimum payments, a high score on this component indicates these customers are more likely to use their credit card for cash-related transactions or carry a balance. They might

use cash advances more frequently and may only pay the minimum amount due each period, potentially indicating a different kind of financial strain or a preference for liquidity over purchase financing.

- Violet Cluster: Since PC1 has strong positive loadings on PURCHASES, PURCHASES_TRX, ONEOFF_PURCHASES, and INSTALLMENTS_PURCHASES, a high value on PC1 suggests that this cluster represents customers who spend significantly on their credit cards. These could be frequent shoppers, possibly more engaged in high-value transactions or a greater number of transactions. Given that PC2 is positively associated with features like CASH_ADVANCE, CASH_ADVANCE_TRX, and BALANCE, but the purple cluster is low on PC2, it implies these customers are less likely to use cash advances or carry over a balance. They might pay off their balances more consistently, indicating good financial management or a lower risk profile.

Outliers/Isolated Points: There are a few points spread out above the main clusters, especially towards the top of the plot. These may represent customers with unique behaviors that don't group well with the main clusters, possibly those with high balances or unusual spending patterns that warrant individual analysis.

On manual analysis of these clusters the median income of the customers who were clustered in the orange cluster was roughly 1200,3560 in the green part of the dendogram and more than 9000 in the red part of the dendogram. These groups can be corresponded to the Low-income, Middle-income and High Earning income groups respectively based on the disparity in their income levels.

C. Large Language Model Clustering

In this model we tested against different sentence transformer based large Language Model embeddings and multiple prompts to get the best possible clusters. The best performing model was **paraphrase-MiniLM-L6-v2** which maps sentences in ASCII to a 384 dimensional vector. It is well versed in clustering tasks. The underlying architecture used is Siamese BERT and has a total of 22.7 million parameters.

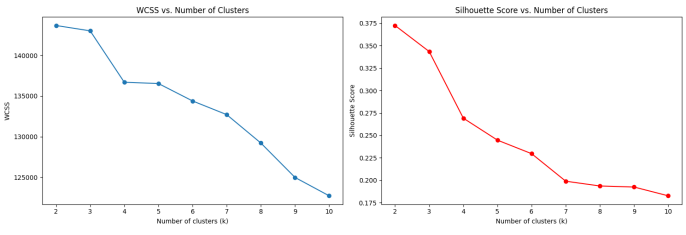

Fig. 10. LLM clustering analysis

According to the graphs in figure 10 WCSS value decreases sharply till K=4 suggesting that the optimal number of clusters according to this algorithm is 4 contrary to what was suggested by KMeans and Hierarchial Clustering. Also the highest Silhouette score was 0.375 which was better than Hierarchial clustering and All except two variants of KMeans. But the rapid development of LLMs with billions of parameters and multiple prompting techniques like Chain of Thought and Zero shot prompting makes it more likely that LLM embeddings will perform better than conventional clustering algorithms in the future.

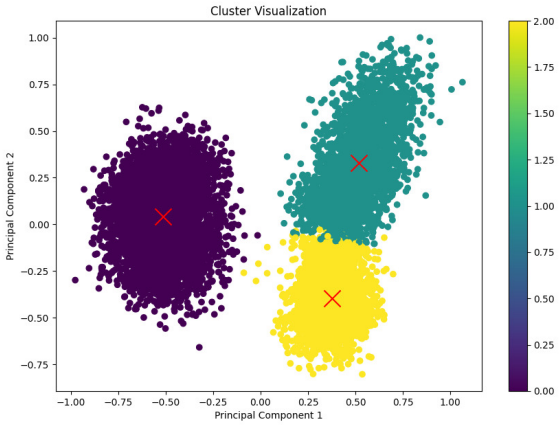

Fig. 11. LLM cluster plot

In the Figure 11, the X marks represent the centroids in each cluster. The purple cluster represents a segment of customers who are conservative in their credit card usage, both in terms of purchases and cash advances. The yellow cluster consists of customers who use their cards modestly for purchases but might lean a bit more on cash advances. The blue cluster is composed of customers with higher purchase volumes, and they might also engage in cash advance activities. This cluster's broader spread suggests diverse spending habits and potentially a mix of credit utilization strategies.

D. DBSCAN

The DBSCAN algorithm was run with different parameter values for epsilon(0.01,0.1,1,10,100) and minPoints (2,3,4,6,8). The best results were obtained for the hyperparameter $epsilon$=1 and minPoints=6.

According to the above figures we can see that WCSS decreases with increasing K. The ELBOW point occurs at K=3. It is also the point with the maximum Silhouette score. However this algorithm gives the worst score of 0.15 compared to all models. This can be attributed to the fact that DBSCAN's performance depends heavily on the density distribution of the data points. If the dataset has varying density regions, DBSCAN might struggle to correctly identify clusters or might merge separate clusters in high-density areas while missing out on more sparse ones.

Additionally, DBSCAN does not perform well with high-dimensional data. In high dimensions, the concept of density becomes less intuitive, and the distance metric (usually Euclidean) used by DBSCAN may not be effective because all

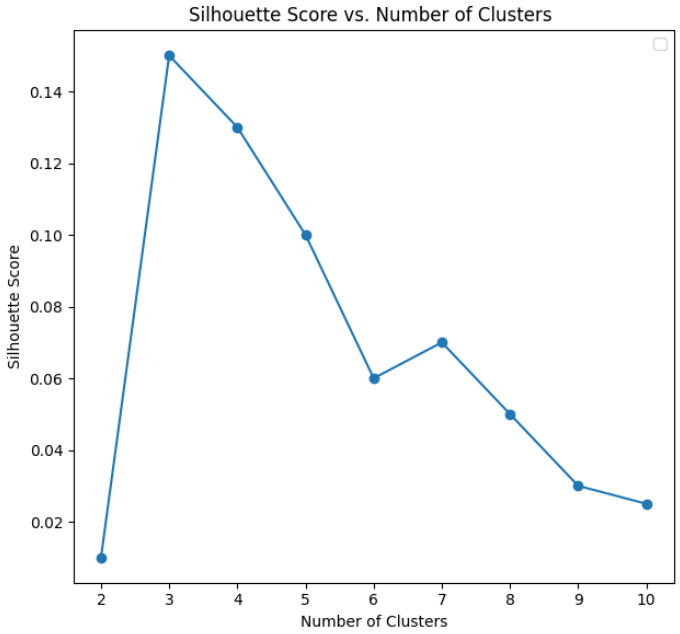

Fig. 12. DBSCAN Silhouette Score

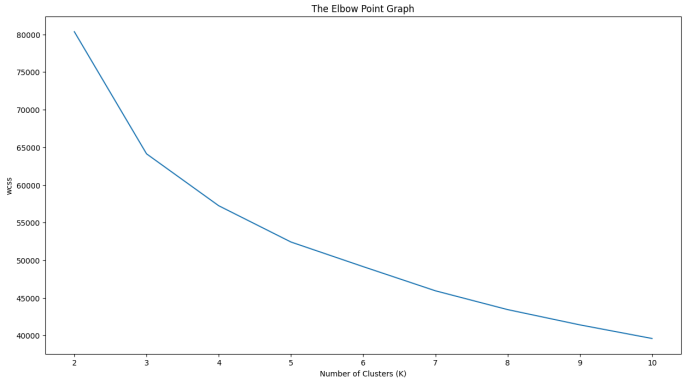

Fig. 13. DBSCAN WCSS

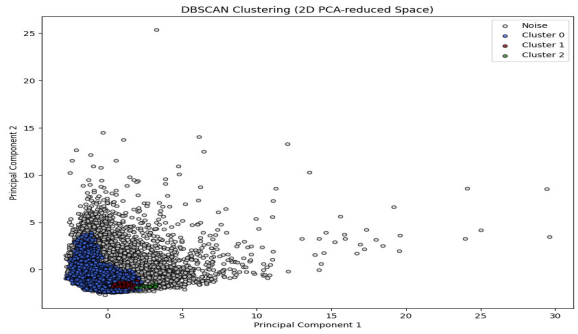

Fig. 14. DBSCAN cluster plots

points tend to be equidistant from each other in very high-dimensional spaces (a phenomenon known as the "curse of dimensionality"). This is the probable reason for the model not working on our dataset properly.

The figure 14 shows the clusters formed by DBSCAN for the optimal hyperparameters $epsilon$=1 and minPoints=6. There's a large primary cluster, a smaller cluster, and several points classified as noise. This indicates a density-based segmentation where most data points are concentrated in a dense region, with some outliers. This method might be helpful to identify core segments and outliers in the credit card dataset. However, the dataset has different densities of the clusters hence DBSCAN fails. Generally, customers with lower expenditure and cash advance activities have higher densities as the number of people falling in these categories is more. Hence, this visually justifies that DBCAN might not be a good algorithm for this task

VI. Conclusion

This research explored various clustering algorithms to analyze a dataset comprising the spending behaviors and credit details of approximately 9000 credit card holders over a six-month period. We focused on comparing the efficacy of traditional algorithms such as K-Means and DBSCAN with more advanced techniques involving kernel functions and embeddings from large language models. Our findings underscored the challenges and limitations of traditional clustering methods like K-Means and DBSCAN when dealing with high-dimensional, non-uniform data. K-Means, while popular for its simplicity and efficiency in handling spherical clusters, faced difficulties with the dataset's complex and multi-dimensional nature. To address this, we incorporated various kernel functions—Gaussian, Tanh, and Polynomial—to enhance the performance of K-Means by mapping data into higher-dimensional feature spaces, thus capturing more complex patterns.

DBSCAN, known for its robustness to noise and ability to discover clusters of arbitrary shapes, was also evaluated. However, its performance was hindered by its sensitivity to the density variation and parameter selection, which proved challenging in the context of our diverse dataset.

To potentially overcome these limitations, we experimented with embeddings generated from large language models, applying traditional clustering algorithms to these embeddings. This approach aimed to leverage the semantic richness of language model embeddings to capture subtle patterns in the data that conventional methods might overlook.

Additionally, we employed Agglomerative Hierarchical Clustering to provide a different perspective on data structure, offering valuable insights into the hierarchical relationships between data points, which can be crucial for certain applications.

Overall, our study highlights the importance of selecting the appropriate clustering technique based on specific data characteristics and the objective of the analysis. It also opens up new pathways for integrating machine learning advancements, such as language model embeddings, into traditional data clustering

practices, paving the way for more nuanced and effective data analysis strategies in financial behavior research.

According to the results and analysis, KMeans with Polynomial cluster is the best performing model for 3 clusters while DBSCAN performed the worst. There is still a lot of room for improvement especially in the models using LLMs as many models using more parameters are continuously being developed so better performance can be achieved. Also new prompting techniques can give better Silhouette scores and more meaningful clusters.

REFERENCES

[1] I. Karo Karo, 'Segmentation of Credit Card Customers Based on Their Credit Card Usage Behavior using The K-Means Algorithm', Journal of Software Engineering, Information and Communication Technology (SEICT), vol. 2, pp. 55–64, 02 2022.

[2] S. Tripathi, A. Bhardwaj, and P. Eswaran, 'Approaches to Clustering in Customer Segmentation', International Journal of Engineering & Technology, vol. 7, p. 802, 07 2018.

[3] A. Petukhova, J. P. Matos-Carvalho, and N. Fachada, 'Text clustering with LLM embeddings', arXiv [cs.CL]. 2024.

[4] D. Deng, 'DBSCAN Clustering Algorithm Based on Density', in 2020 7th International Forum on Electrical Engineering and Automation (IFEEA), 2020, pp. 949–953.

[5] I. Davidson and S. Ravi, 'Agglomerative Hierarchical Clustering with Constraints: Theoretical and Empirical Results', 11 2005, pp. 59–70