Assignment 02

Clustering Algorithm on Given Credit Card Dataset

Contents

1. Introduction 3

2. Data preprocessing 3

Part A: K-means Clustering 4

K-means Clustering 4

3. Algorithm Working 4

3.1 Results 5

Part B: Hierarchical Clustering 7

Hierarchical Clustering 7

4. Algorithm Working 8

4.1. Results 9

Part C: Density-based Clustering 10

Density-based Clustering 10

5. Algorithm Working 10

5.1 Results 11

6. Comparison of Clustering Algorithms 12

7. Conclusion 13

References 14

1. Introduction

In today's data-driven world, understanding customer behavior is crucial for businesses to tailor their marketing strategies effectively. Clustering analysis serves as a powerful tool in uncovering patterns within large datasets, allowing businesses to segment their customer base and personalize their marketing approaches. Clustering techniques, such as K-means, hierarchical clustering, and density-based clustering, enable businesses to group customers with similar traits or behaviors together. Which is helpful in;

- Targeted Marketing Campaigns
- Customer Segmentation
- Product Recommendations
- Retention Strategies

2. Data preprocessing

We began by loading a dataset (CC_GENERAL.csv) using the Pandas library, which contained financial information about credit card holders. Initially, we inspected the dataset to understand its structure and identified missing values, revealing several columns with null values. To address this, we utilized the SimpleImputer from scikit-learn, replacing missing values with the mean of each respective column. Next, categorical variables in the dataset were encoded into numerical form using LabelEncoder, facilitating compatibility with machine learning algorithms. Following this, the data was standardized using StandardScaler to ensure that all features were on the same scale, minimizing the impact of differing measurement units on clustering algorithms.

1. **Importing Libraries and Data:**
    - **Methodology:**
        - Imported necessary libraries: pandas, numpy, sklearn, matplotlib, and plotly.
        - Loaded the dataset CC GENERAL.csv using pandas.
    - **Results**
        - Successfully loaded Dataset into a DataFrame credit_cards.
2. **Data Preprocessing:**
    - **Methodology:**
        - Checked for any null values in the dataset.

- Dropped rows with missing values using dropna().

- **Results:**
  - No missing values were found.
  - Clean and complete Dataset obtained.

3. **Feature Selection and Scaling:**
   - **Methodology:**
     - Checked the distribution of classes in the column to understand the class imbalance.
     - Selected relevant features for clustering: BALANCE, PURCHASES, and CREDIT_LIMIT
     - Scaled these features using MinMaxScaler to bring them to a common range [0,1].
   - **Results:**
     - Normalized features, ensuring they are on a similar scale.

Part A: K-means Clustering

K-means Clustering

K-means clustering is a popular unsupervised machine learning algorithm used for partitioning a dataset into K distinct, non-overlapping clusters. It aims to group data points based on their similarity, where each cluster is represented by its centroid (the mean of all data points in the cluster). The algorithm iteratively assigns each data point to the nearest centroid and updates the centroids until convergence, minimizing the sum of squared distances between data points and their respective centroids. K-means is efficient and effective for clustering tasks but requires prior knowledge of the number of clusters K and may converge to local optima depending on the initial placement of centroids.

3. Algorithm Working

In this section, K-means clustering algorithm is used for identifying the appropriate number of clusters, along with the corresponding Sum of Squared Errors (SSE):

1. **K-means Clustering:**

- Applied K-means clustering with a predefined number of clusters (5) for initial analysis.
- Assigned cluster labels to the original dataset under the column CREDIT_CARD_SEGMENTS.

2. **Elbow Method to Determine Optimal K:**

- Defined the elbow_method function to calculate SSE for a range of cluster numbers (1 to 10).
- Iterated over each value of K, performed K-means clustering, and stored the SSE (inertia).
- Plotted the SSE values against the number of clusters to identify the "elbow point" where the SSE reduction diminishes significantly.

3. **Applying the Optimal Number of Clusters:**

- From the elbow plot, determined that 4 clusters were optimal (as there was no significant SSE reduction past 4 clusters).
- Applied K-means clustering with 4 clusters.
- Mapped cluster labels to user-friendly names for easy interpretation.

4. **Visualization:**

- Used Plotly to create a 3D scatter plot to visualize the clusters.
- Plotted BALANCE, PURCHASES, and CREDIT_LIMIT for each cluster.
- Updated the plot layout for better readability and presentation.

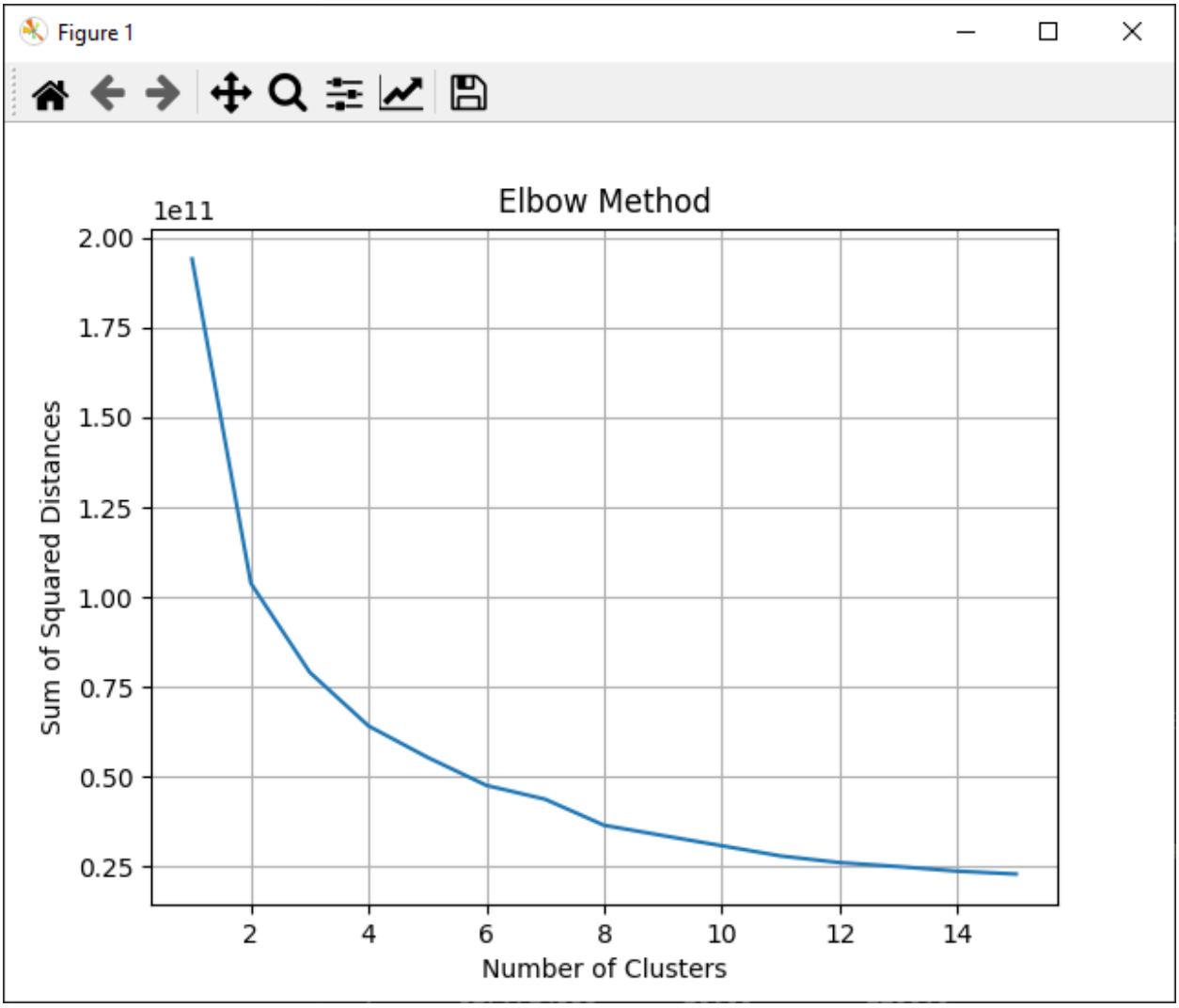
3.1 Results

```
The number of rows in the DataFrame is:  8950
The number of columns in the DataFrame is:  18
  CUST_ID      BALANCE  ...  PRC_FULL_PAYMENT  TENURE
0  C10001    40.900749  ...          0.000000      12
1  C10002  3202.467416  ...          0.222222      12
2  C10003  2495.148862  ...          0.000000      12
3  C10004  1666.670542  ...          0.000000      12
4  C10005   817.714335  ...          0.000000      12
```

```
[5 rows x 18 columns]
      BALANCE  PURCHASES  CREDIT_LIMIT
0       40.900749      95.40        1000.0
1     3202.467416       0.00        7000.0
2     2495.148862     773.17        7500.0
4      817.714335      16.00        1200.0
5     1809.828751    1333.28        1800.0
...           ...        ...           ...
8943     5.871712      20.90         500.0
8945    28.493517     291.12        1000.0
8947    23.398673     144.40        1000.0
8948    13.457564       0.00         500.0
8949   372.708075    1093.25        1200.0

[8636 rows x 3 columns]
0     Cluster 1
1     Cluster 4
2     Cluster 4
4     Cluster 1
5     Cluster 1
6     Cluster 3
7     Cluster 1
8     Cluster 4
9     Cluster 4
10    Cluster 1
Name: CREDIT_CARD_SEGMENTS, dtype: object
```

Elbow Method used to determine the optimal number of clusters (k) in K-means clustering. The goal is to identify the point where adding more clusters does not significantly improve the model, indicated by a diminishing return in the reduction of the within-cluster sum of squares (WCSS).



Part B: Hierarchical Clustering

Hierarchical Clustering

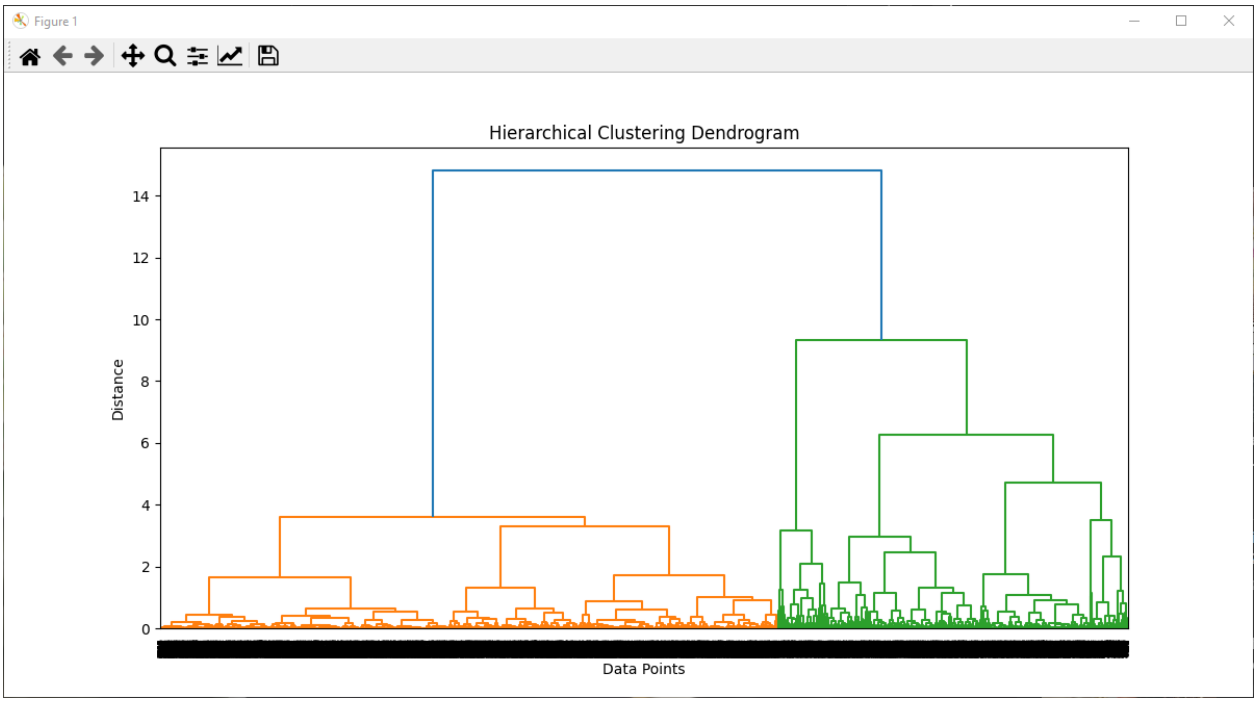
Hierarchical clustering is a popular unsupervised learning algorithm used in data mining and machine learning for grouping similar objects into clusters. It builds a hierarchy of clusters either by merging smaller clusters into larger ones (agglomerative) or by splitting larger clusters into

smaller ones (divisive). methodology and approach used in the provided hierarchical clustering code:

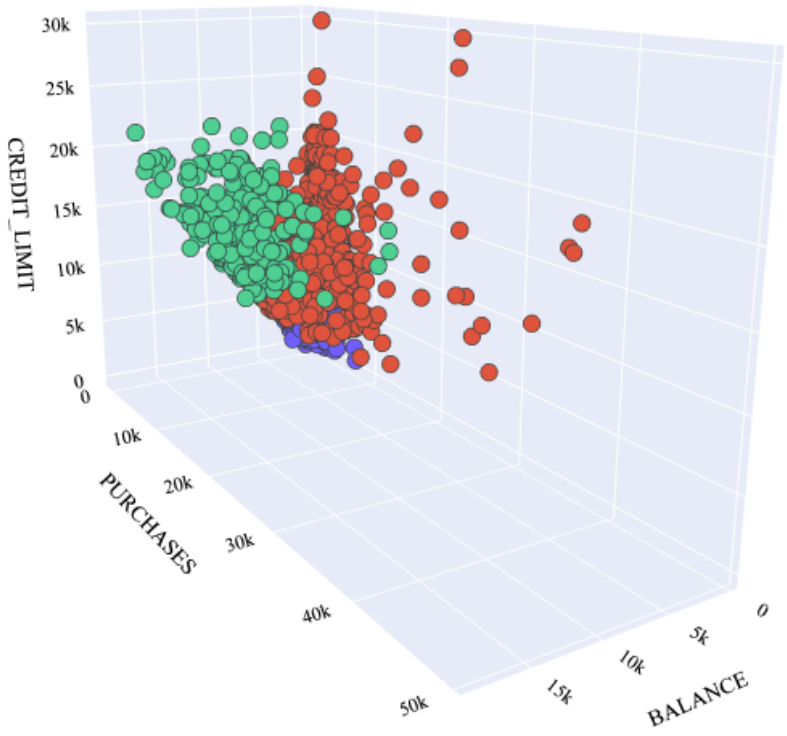
4. Algorithm Working

- **Hierarchical Clustering:**
  - **Linkage:**
    - Hierarchical clustering is performed using the linkage function from scipy.cluster.hierarchy.
  - **Methodology and Dendrogram:**
    - ward is chosen because it minimizes the variance of clusters being merged at each step.
    - Effective for minimizing the variance within each cluster.
    - Metric: euclidean distance is used to measure the distance between clusters.
    - A dendrogram is plotted to visualize the clustering process. It shows how clusters are progressively merged and helps determine the optimal number of clusters based on the height of the dendrogram branches.
- **Determining Number of Clusters:**
  - **Threshold Setting:**
    - A threshold (max_d) is set on the dendrogram height to determine the number of clusters.
    - It is adjusted based on the dendrogram plot to identify distinct clusters.
- **Assigning Clusters:**
  - The fcluster function assigns each data point to a cluster based on the hierarchical clustering results and the specified threshold (max_d). The criterion='distance' parameter ensures that clusters are formed based on the distance threshold.
- **Visualization:**
  - **3D Scatter Plot:**
    - Plotly (plotly.graph_objects) is used to create an interactive 3D scatter plot.
    - Each cluster is represented by a different color and symbol in the plot.
    - The three selected features (BALANCE, PURCHASES, CREDIT_LIMIT) are plotted on the x, y, and z axes respectively, allowing visualization of how data points are distributed across clusters in the 3D space.

4.1. Results



Hierarchical Clustering:



Part C: Density-based Clustering

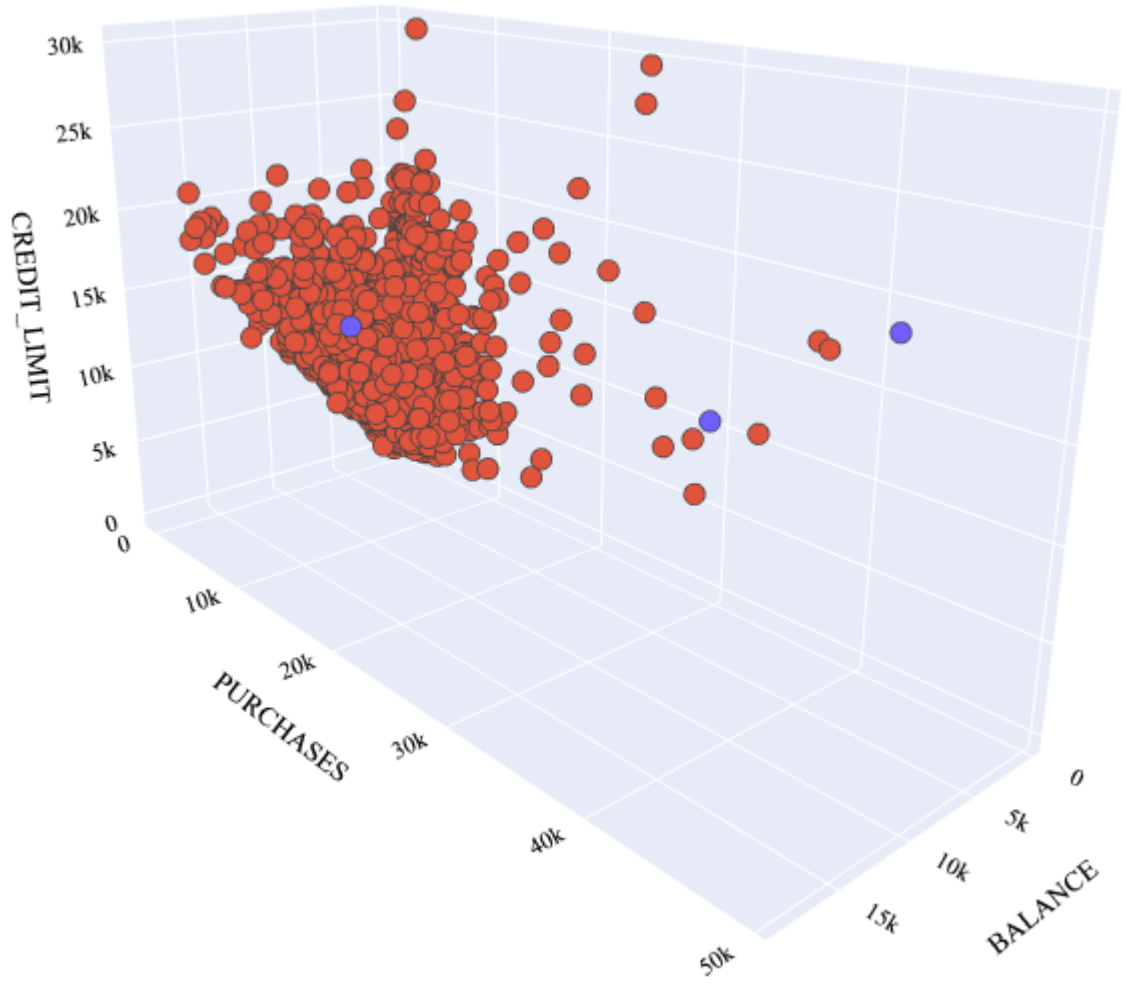
Density-based Clustering

Density-based clustering is a type of clustering algorithm that identifies clusters in a dataset based on the density of data points in the feature space. Unlike centroid-based (e.g., k-means) or hierarchical clustering, density-based methods do not require specifying the number of clusters beforehand and can handle non-linear boundaries and varying cluster shapes effectively. One of the most popular density-based clustering algorithms is DBSCAN (Density-Based Spatial Clustering of Applications with Noise).

5. Algorithm Working

- **DBSCAN Clustering:**
  - **Algorithm Parameters:**
    - DBSCAN is initialized with parameters eps=0.3 and min_samples=10.
      - eps: The maximum distance between two samples for them to be considered as in the same neighborhood.
      - min_samples: The number of samples (points) in a neighborhood for a point to be considered as a core point.
    - DBSCAN does not require specifying the number of clusters beforehand, as it identifies clusters based on density within the dataset.
- **Cluster Assignment:**
  - fit_predict method of DBSCAN is used to fit the model to the scaled data and predict cluster labels for each data point.
  - Points labeled as -1 are considered as noise points (not assigned to any cluster).
- **Visualization:**
  - **3D Scatter Plot:**
    - Plotly (plotly.graph_objects) is used to create an interactive 3D scatter plot.
    - Each cluster and noise points are plotted with different colors and symbols.
    - The three selected features (BALANCE, PURCHASES, CREDIT_LIMIT) are plotted on the x, y, and z axes respectively, allowing visualization of how data points are distributed across clusters in the 3D space.

5.1 Results


30k
25k
20k
15k
10k
5k
0
CREDIT_LIMIT
0
10k
20k
30k
40k
50k
PURCHASES
0
5k
10k
15k
BALANCE

6. Comparison of Clustering Algorithms

- **Flexibility in Cluster Shape:**
  - **Hierarchical:** Can handle clusters of arbitrary shapes but sensitive to noise and outliers.
  - **K-means:** Assumes clusters are spherical, less effective with non-linearly separable data.
  - **DBSCAN:** Does not assume cluster shape, effective for non-linearly separable data and varying cluster densities.
- **Noise Handling:**
  - **Hierarchical:** Limited capability to handle noise explicitly.
  - **K-means:** No explicit noise handling, all points are assigned to a cluster.
  - **DBSCAN:** Explicitly identifies and labels noise points, useful for outlier detection.
- **Determining Number of Clusters:**
  - **Hierarchical:** Requires interpretation of dendrogram.
  - **K-means:** Elbow method or silhouette score.
  - **DBSCAN:** Automatically determines clusters based on density parameters.
- **Scalability:**
  - **Hierarchical:** Less scalable for large datasets due to computational complexity.
  - **K-means:** More scalable than hierarchical clustering but sensitive to initialization.
  - **DBSCAN:** Scales well for large datasets but sensitivity to parameter selection (eps and min_samples).

6.1 Choice of Algorithm

- **Hierarchical:** Best for understanding hierarchical structure in data when the hierarchy is important.
- **K-means:** Suitable for datasets where clusters are expected to be spherical and of similar size.
- **DBSCAN:** Effective for datasets with complex structures, varying cluster densities, and noise/outlier detection needs.

7. Conclusion

In conclusion, Hierarchical Clustering is valuable for revealing hierarchical relationships but may be computationally expensive and sensitive to noise. K-means Clustering is straightforward to implement and efficient for spherical clusters, yet requires pre-specification of the number of clusters and struggles with non-linear separable data. DBSCAN excels in handling datasets with varying cluster densities and noise, automatically determining the number of clusters based on data density, although it requires careful parameter tuning. Ultimately, selecting the appropriate clustering algorithm hinges on understanding the dataset's structure, the nature of clusters desired, and balancing computational efficiency with algorithmic robustness to noise and outliers.

References

DBSCAN Clustering Algorithm Based on Density. (2020, September 1). IEEE Conference Publication | IEEE Xplore. https://ieeexplore.ieee.org/abstract/document/9356727.

Nandi, A. K., Randhawa, K. K., Chua, H. S., Seera, M., & Lim, C. P. (2022). Credit card fraud detection using a hierarchical behavior-knowledge space model. *PloS One*, *17*(1), e0260579. https://doi.org/10.1371/journal.pone.0260579

Chen, Y., & Zhang, R. (2021). Research on Credit Card Default Prediction Based on k-Means SMOTE and BP Neural Network. *Complexity*, *2021*, 1–13. https://doi.org/10.1155/2021/6618841