Data Mining Project Report

Hao Liang, Yucheng Feng, Licheng Du, Hanwei Hu

Abstract

Credit risk is a major concern for lending institutions, as it can significantly affect their financial stability. Accurately predicting the creditworthiness of loan applicants is critical for making informed lending decisions and minimizing the risk of default. While many use classification models to predict whether a customer will default or not, the models usually suffer from multiple problems, which will result in the potential loss of customers (Brown & Mues, 2011). Thus, we came up with a completely different approach to improve bank revenue by using clustering to segment customers and calculate their within-cluster default rate, which will avoid some key issues with the classification model approach. Given those default rates as risks, bank managers and analysts can generate different loaning products for each cluster to make sure a product can be profitable enough. With the goal clearly stated, we aim to build a clustering model to create customer segmentation using three different machine learning algorithms: K-means, hierarchical clustering (agglomerative), and K-prototypes.

To achieve our goal, we will first perform exploratory data analysis (EDA) to gain insights into the dataset and identify any patterns or correlations. We will then preprocess the data, including handling missing values, encoding categorical variables, and scaling numerical features. We will then apply three different classification algorithms - K-means, agglomerative clustering, and K-prototypes - to our data. We will fine-tune the number of clusters generated by each model using model-specific methods (e.g., WCSS, silhouette score) and then compare their performance using silhouette score. Ultimately, we will select the best model and analysis on the clusters it generated.

Paper Review And Modeling Approaches

Data Source

The data for this project is the Baseline for Credit Risk dataset obtained from Kaggle. The dataset used in this project contains information about various loan applicants, such as their age, income, credit history, and other demographic features. The target variable is a binary classification indicating whether or not the applicant defaulted on their loan. Our objective is to train our models on this dataset and evaluate their performance based on recall.

Modeling Approach

We chose the clustering method instead of classification to counter credit default risk because normally classification suffers from low precision. It also requires a significant amount of data to train. With the low recall issue, the model will lead to an overwhelming number of false positives (non-default individuals being classified as “will default”), which will lead to a significant loss of potential customers (Brown & Mues, 2011) and, thus, revenue. Instead of focusing on the individual likelihood of default, we came up with the idea of measuring the default likelihood of customer segmentation, which is achieved by clustering models. By clustering customers into groups, similar customers will share the same probability of default within the group. Therefore, in the long run, with a sufficient number of customers, given the risk of default (probability of default), banks can easily derive different loaning strategies to ensure, within each cluster, strategies are profitable and able to maximize this profit (the problem of default is transferred into a binomial optimization problem). This is superior to just viewing the whole customer as a group, where we can fine-tune each segment and produce the local maximum profit for each segment which builds up to the global maximum.

This method will start with the use of clustering algorithms, we reviewed several papers on suitable methods for credit data. Gholamian, Jahanpour, and Sadatrasoul (2013) suggested that hierarchical algorithms and K-means both had superior performance with different measures. Based on this information, we picked hierarchical algorithms and K-means for this project. However, since the “general” K-means method cannot account for categorical variables. We also picked the K-prototypes method, a sub-method of the K-means algorithm, in addition to the original method. Therefore, we decided on using K-means, hierarchical (agglomerative), and K-prototypes as our clustering method. To compare the result of the clustering model, we decided to use the silhouette score to determine which result was the best.

After having the best model and its result, we will calculate the proportion of customers in each cluster that defaulted. These proportions will just be our estimated risk for each customer segment that is ready for the bank to use.

Data Analysis and Model Development

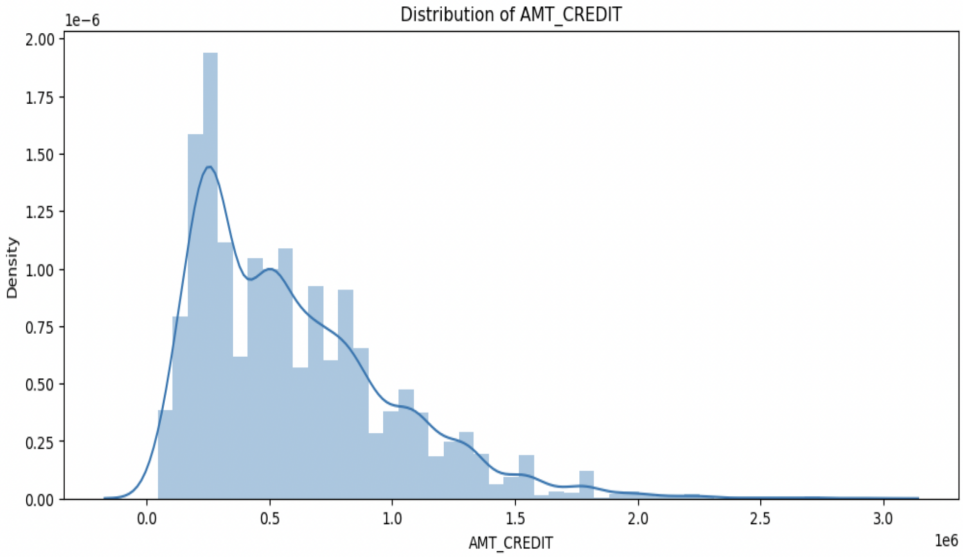
EDA And Feature Engineering

Drop Unnecessary Columns

Upon previewing the dataset, we discovered that some columns had numerous missing values while others were incomprehensible based on their descriptions, and we decided to drop some columns. For instance, columns related to loaner apartments, such as "BASEMENTAREA_AVG," "ELEVATOR_AVG," and "LIVINGAREA_AVG," all had identical descriptions, making it challenging to differentiate between them. Therefore, we retained only one column, "APARTMENTS_AVG," to serve as an overall apartment rating. Additionally, we dropped other columns like "FLAG_DOCUMENT_x" as they merely indicated whether a client provided a particular document or not, with no information about the document's content. Concerning clients' phone numbers, we opted to retain only whether they had their mobile numbers on file, even though multiple columns were available.

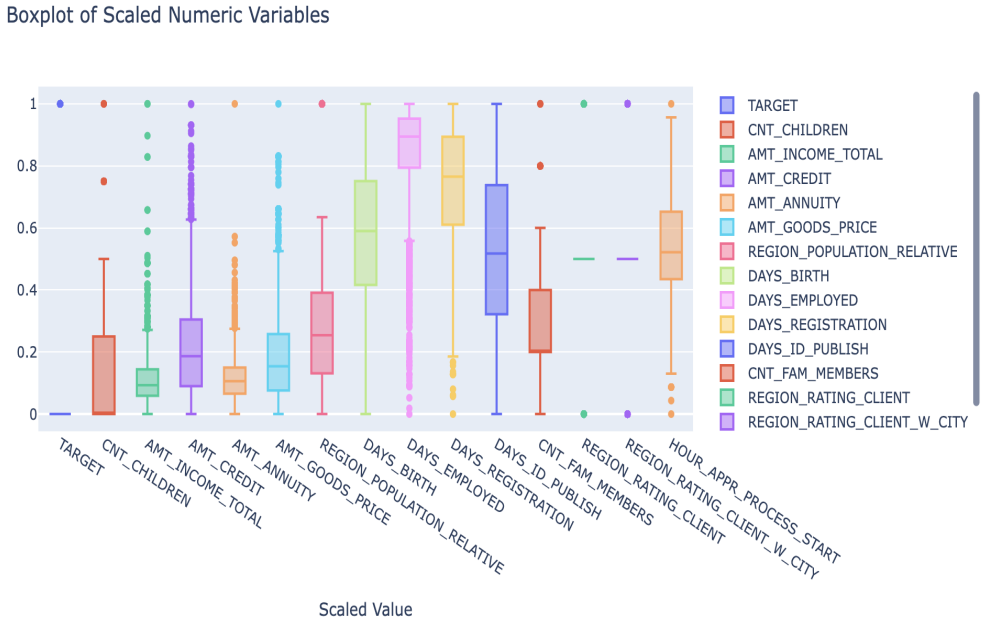
To perform additional data transformations, we divided the data frame into categorical and numerical columns.

1. Transformation
    a. Numerical Variables:
        i. **Via visualization**, most numerical variables have right-skewed distribution, which is reasonable in terms of Credit Data. Most people have a small amount of credits, income, while a few with very high credits and Income, and so on. To solve this problem, we consider trying log transformation, BoxCox transformation to change distribution.

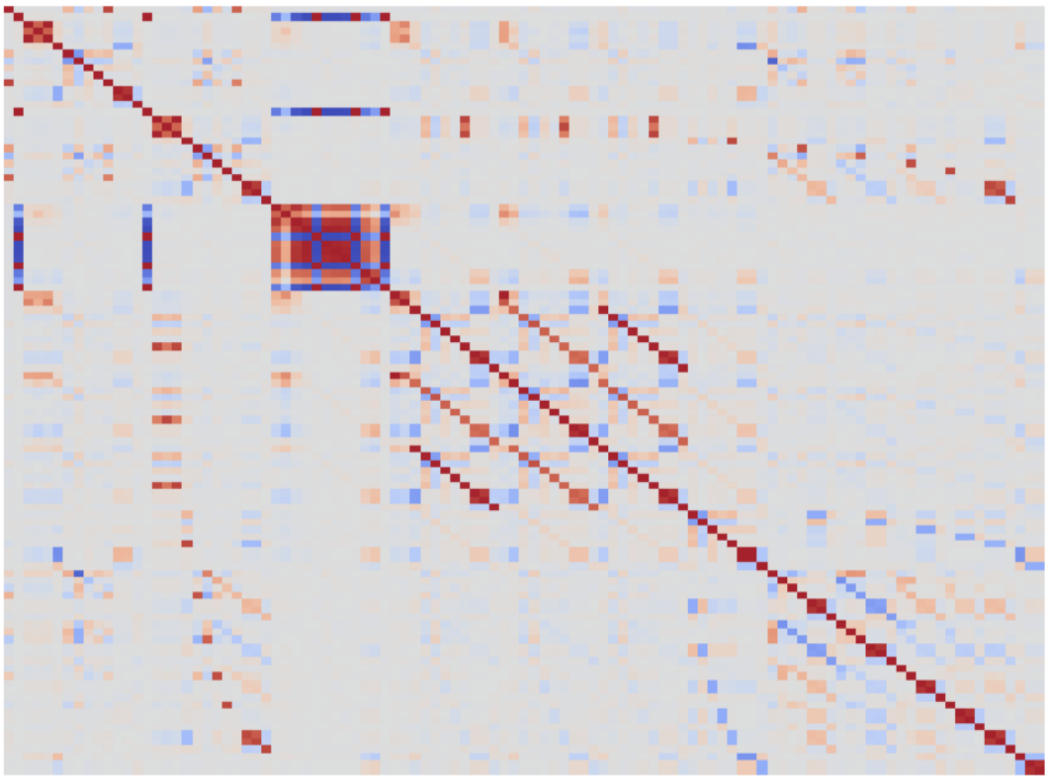


        ii. **Min_max Transform all numerical variables.** This transform will keep the original distribution of all variables. With normalized numerical values, it is beneficial for us to interpret different features

together at the same scale and it helps improve the model's efficiency and performance.



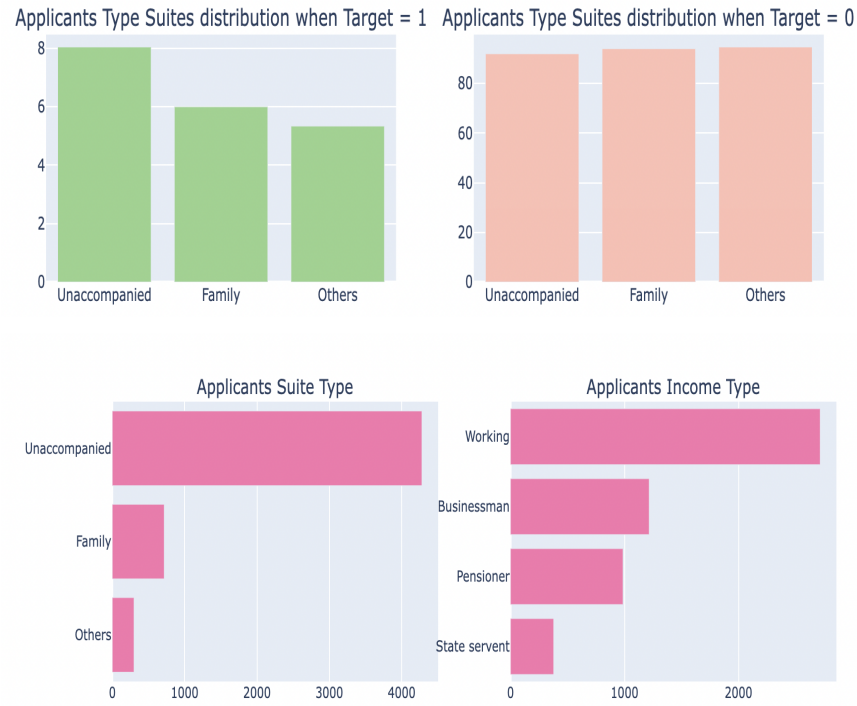
iii. **Feature Crossing:** We applied Polynomial Feature from Sklearn to our numerical features and created interaction features. We believe that Feature crossing can be beneficial for clustering because it allows the algorithm to consider interactions between features, which can capture more complex relationships in the data. By creating new features that represent the interaction between two or more original features, we can better capture the underlying structure of the data.



The correlation plot shows that some features have interaction as the red dots and blue dots indicate (gray means 0).

b. Categorical:

i. Some variables have **high cardinality**, with some categories only having a few data points. Under this circumstance, we combine some rare categories into one category, or a big category according to the meanings. For example, `NAME_TYPE_Suites` have some types like Children, Group people A, Group people B, etc., with few observations so we combine them as `Others`(Hashing)



ii. We explored how each categorical variable changed for two Target groups. It turns out that, When Target = 0, the number of different types is similar, but among people with Target =1, the number of different types varies.

iii. The final step is to create Dummy variables for these variables so that the data can be passed into clustering models, which only take numeric values.

2. **Missing values:** After we dropped columns with all null values, we found the left columns with few NA, so we just dropped all of these NA.

3. **Feature Selection:**

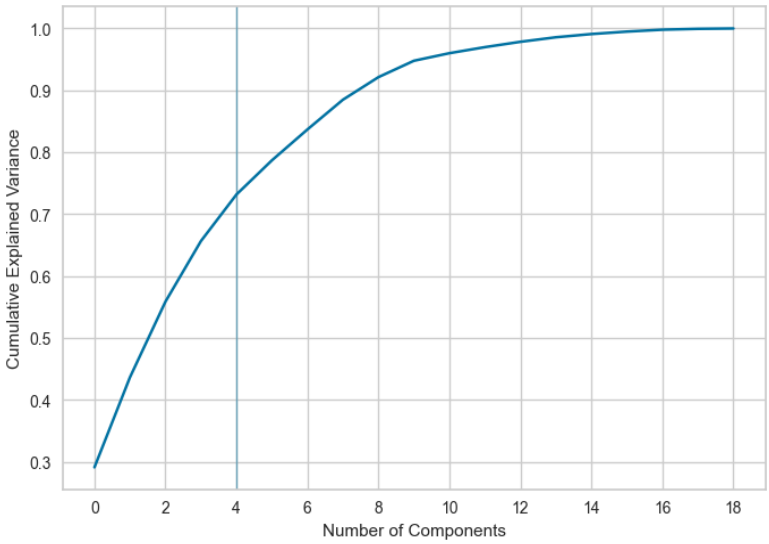
1. As we have talked about above, some features are irrelevant to our clustering goal, so we just dropped those meaningless features such as FLAG_DOCUMENT_x
2. After analyzing the features which have integer types of values, some of them are actually categorical. After careful discussion, we decided to drop them as they are not too relevant to our problems.

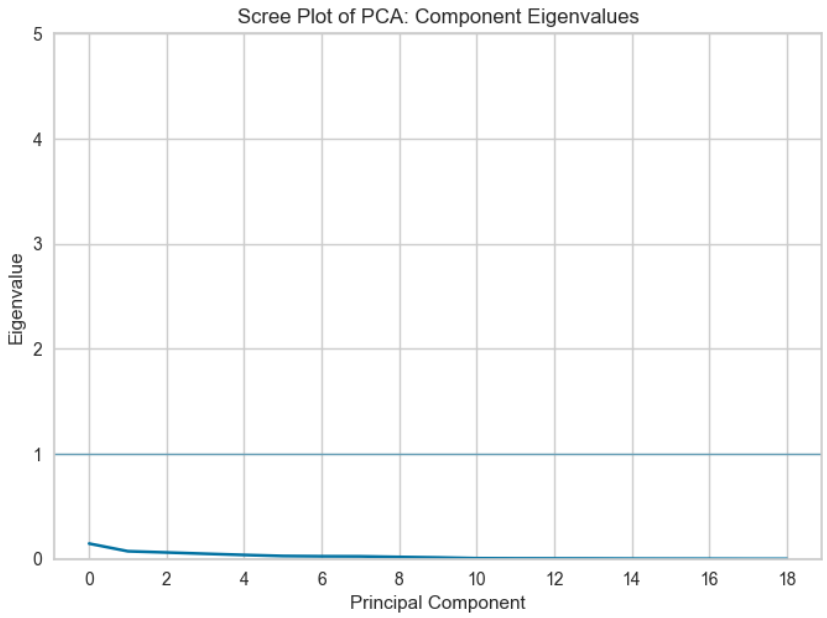
4. **Anomaly detection:**
    1. We found that there are too many points that have `days of employed` = 365243, so we assume it may be some mistake data, so we just dropped them from the data.
    2. We applied a tree-based model called Isolation Tree to detect those outliers within the data since it is a robust algorithm that can handle different types of data, including both numerical and categorical variables, and it is not affected by the presence of irrelevant or redundant features. In the end, half a hundred rows are dropped, which reduces the number of outliers a bit.

Model Development

In this model development section, we mainly applied three different cluster methods, K-means, K-prototypes, and Hierarchical clusterings. Before fitting the models, we pre-processed the data, such as removing outliers and normalizing the data (min-max scaling).

Before directly applying any cluster methods directly, we first applied the Principal Component Analysis (PCA) to reduce the dimensionality of our data, since there are now 19 numeric features in our data, which would be too computationally expensive when we want to fit and compare different cluster algorithms and can help to simplify our analysis and improve the performance of our models. Using the PCA package in Scikit-learn, we first plot the cumulative Explained Variance to select the best number of PCA components we want to apply. We chose 4 components since they can explain about 73% of the variance in our numerical features' data. In addition, the scree plot confirmed 4 components are an optimal choice for the numeric data.



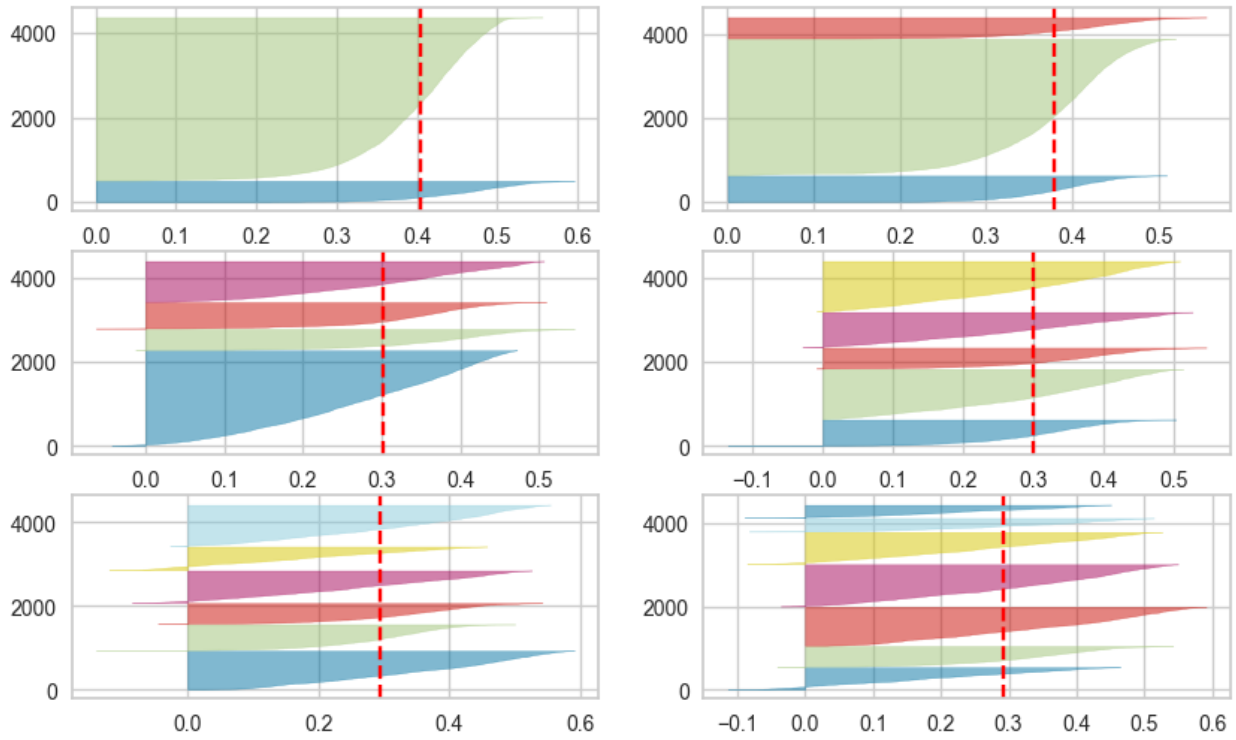


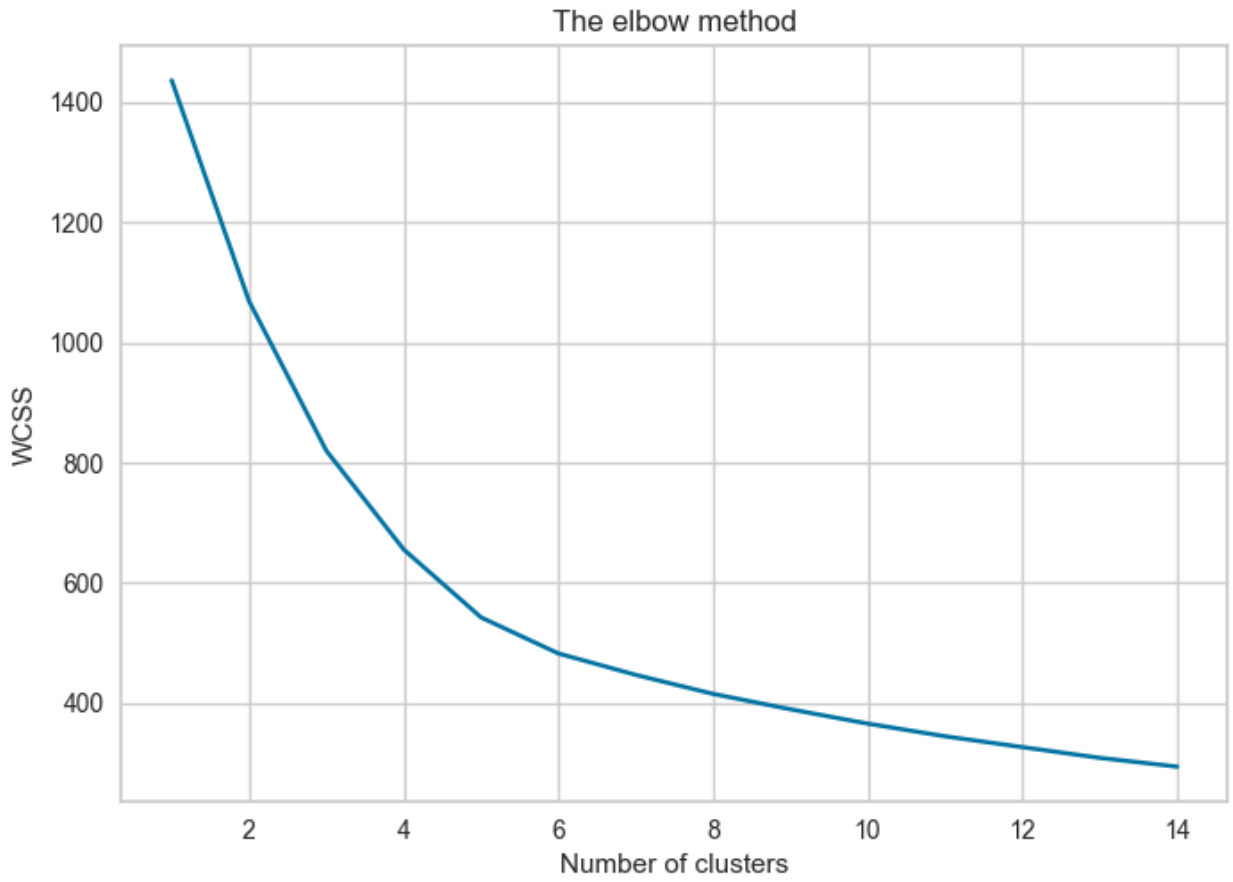
K-means

The first method we tried is K-Mean, one of the most popular unsupervised clustering algorithms. The algorithm works by partitioning the dataset into a fixed number of clusters and assigning each data point to the nearest centroid based on their similarity in features. And we want to use K-means as the benchmark for our clusters.

Following the initial data exploration, we started to implement the K-means algorithm using Scikit-Learn's KMeans class. We used the elbow method to choose the best number of clusters. The plot presented the sum of squared distances between data points and their assigned centroid against the number of clusters. The optimal number

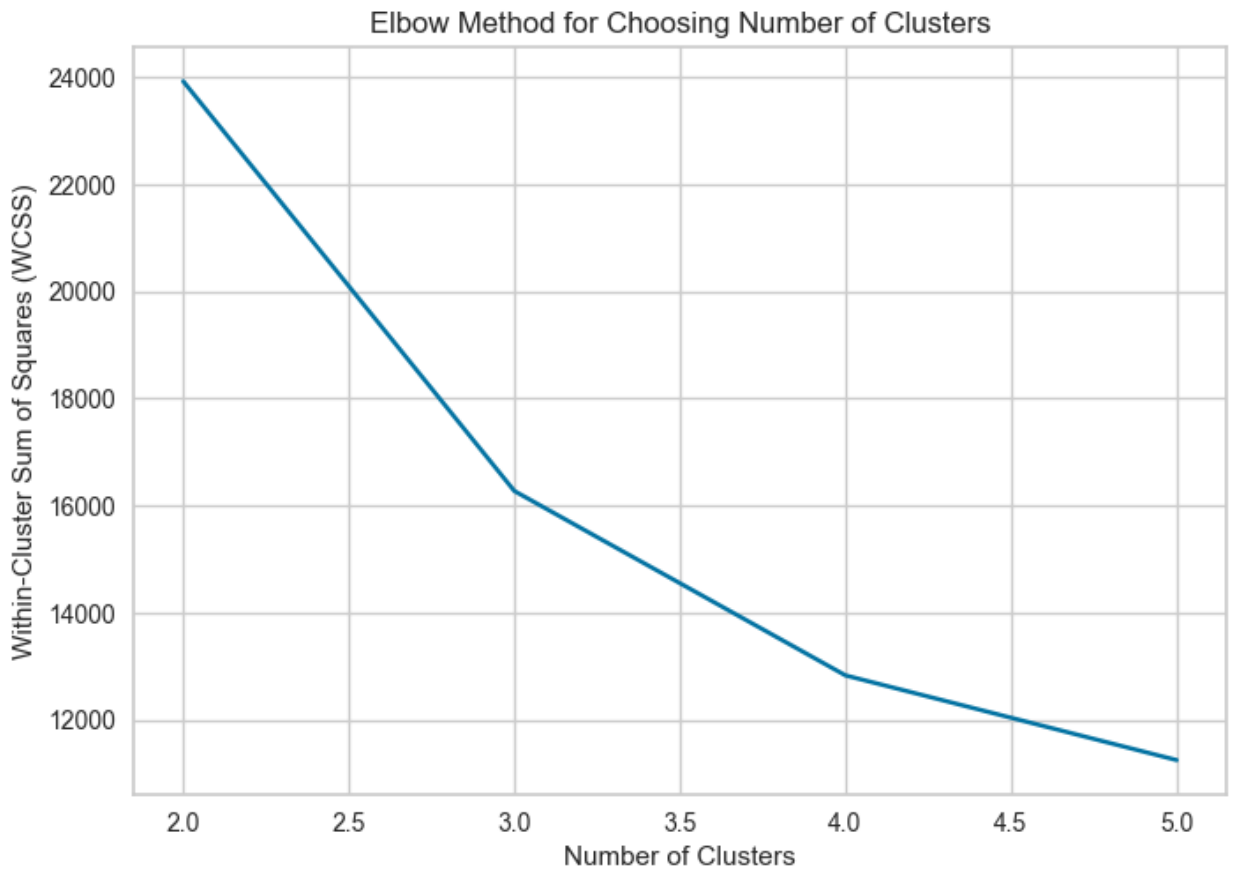
of clusters is 5, where the graph starts to flatten out and the reduction in distortion is minimal. Also, the number of customers in 5 clusters is the most evenly distributed cluster according to the Sihousette plot. Additionally, the k-means++ initialization method was chosen to select centroids that are distant from each other, ensuring a more efficient and accurate clustering process. Once the K-means model was fitted to the dataset, we used the predict() method to assign cluster labels to each customer.





K-Prototypes

Instead of solely focusing on the numeric features, we believed that those categorical are also very important in helping cluster the customers. Thus, we tried the K-prototypes method, as it could handle categorical features as well. Similar to what we did for K-means, based on the elbow method results, we chose the number of clusters as 4. Also, the K-prototypes function can deal with categorical features automatically. There is no need to do any additional data transformation, such as one-hot encoding to those categorical variables. Once the K-prototypes model was fitted to the dataset, we used the predict() method to assign cluster labels to each customer.



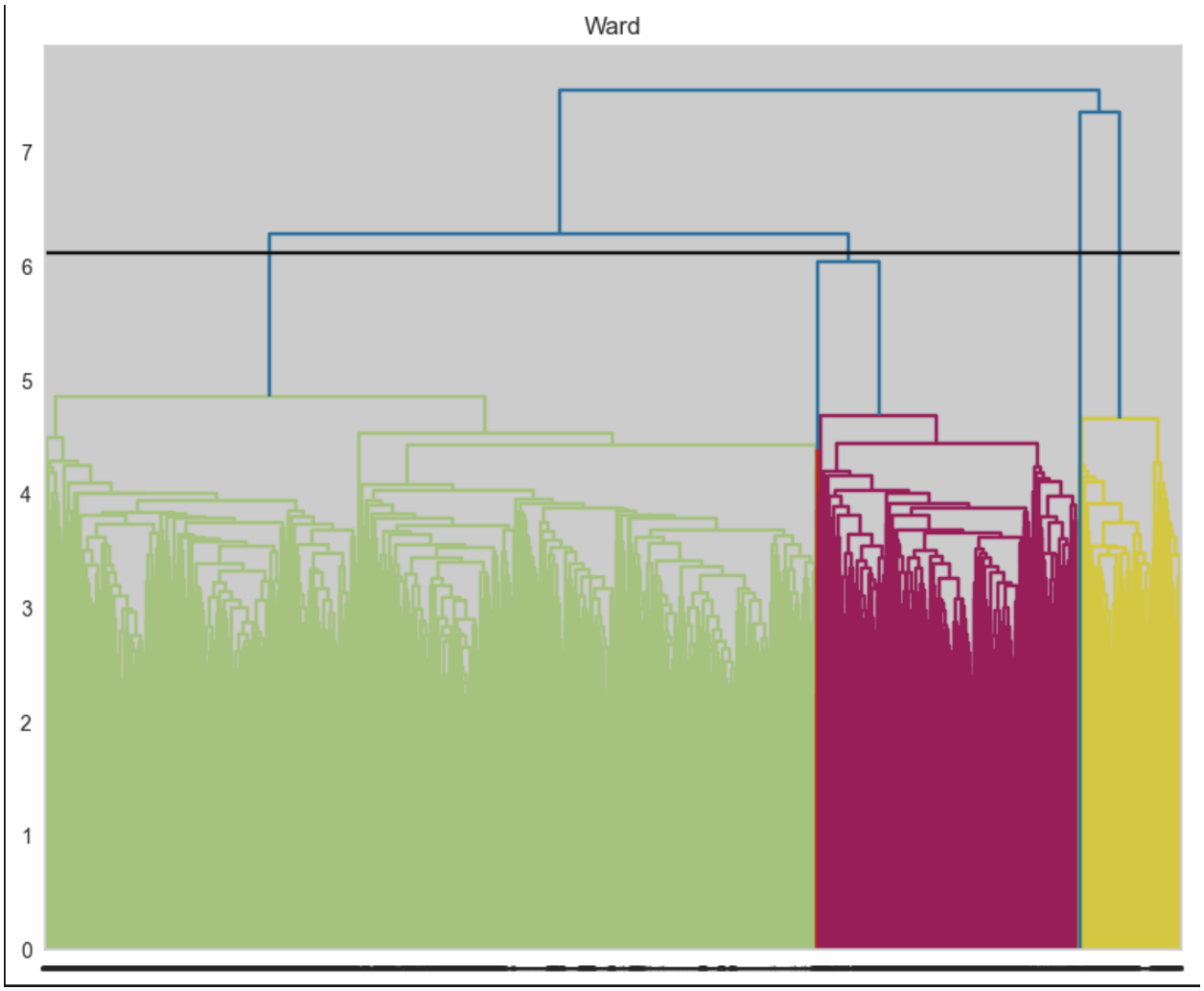
Hierarchical (Agglomerative) Clustering

Hierarchical clustering, which is also referred to as agglomerative clustering, is a technique in machine learning and data science that aims to group similar data points into clusters. The fundamental concept behind hierarchical clustering involves merging pairs of clusters in a step-by-step manner based on their similarity until all data points are part of one single cluster. The algorithm starts with each data point assigned to its own cluster and then gradually merges the two closest clusters until all points are grouped into a single cluster. The similarity between clusters is measured using various distance metrics, including Euclidean distance, Manhattan distance, or cosine distance.

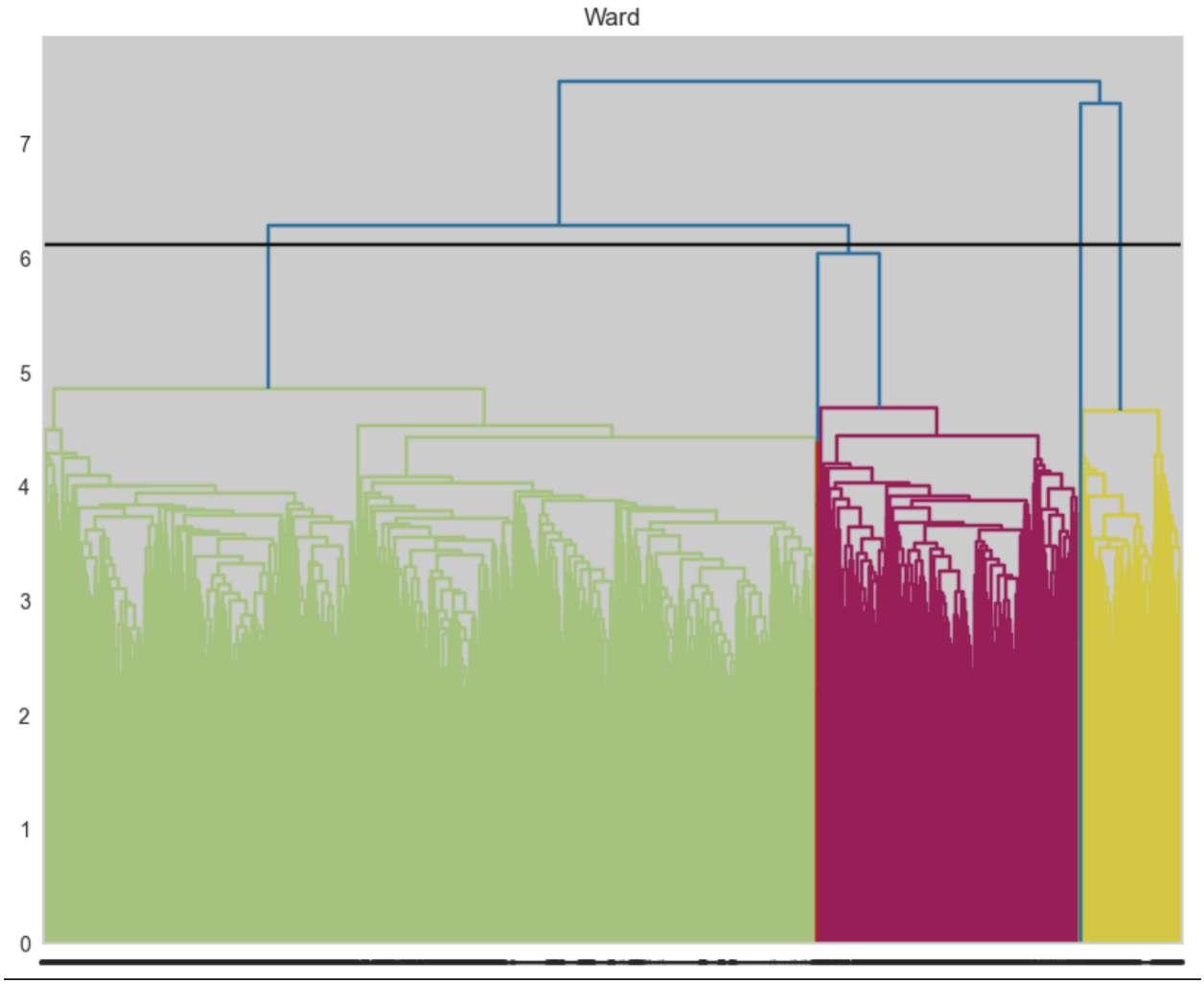
For the Hierarchical Clustering, we fitted two models to our dataset (one with categorical variables, and one without). The process of choosing the best number of clusters is similar to what we did in K-means and K-prototypes. However, since Hierarchical clustering does not have an inertia_ attribute, instead, we used the sum of squared distances for each cluster number. The idea of choosing the number of clusters is still the number when there is no significant improvement in the sum of squared distances. And the best number of clusters without and with categorical variables are 3 and 5. In

addition, in contrast to how K-prototypes deal with categorical variables, Hierarchical Clustering required us to perform the one-hot encoding for those categorical variables. Once the Hierarchical Clustering models were fitted to the dataset, we used the predict() method to assign cluster labels to each customer.

Dendrogram for Numerical Variables Only



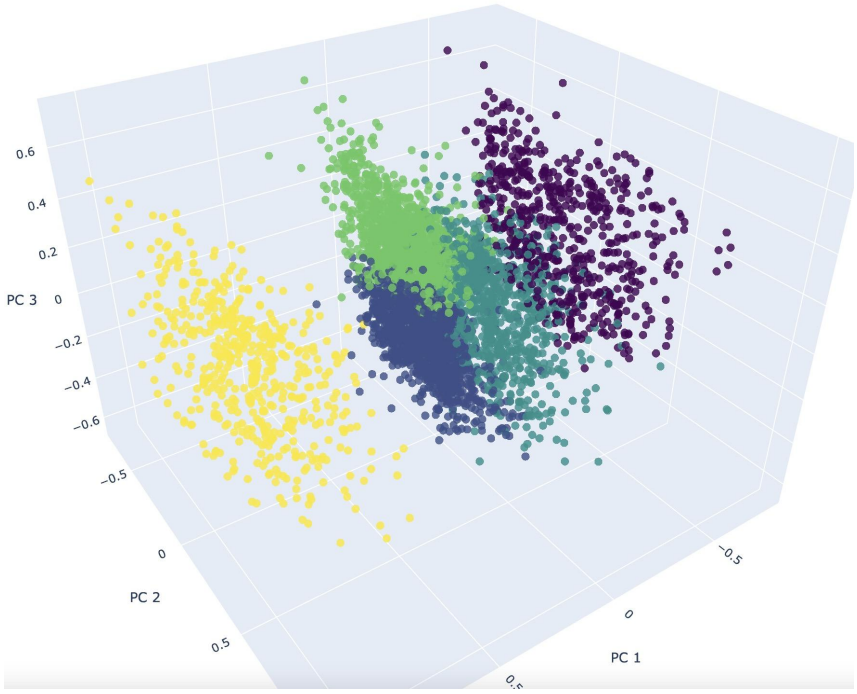
Dendrogram for Numerical Variables and Categorical Variables

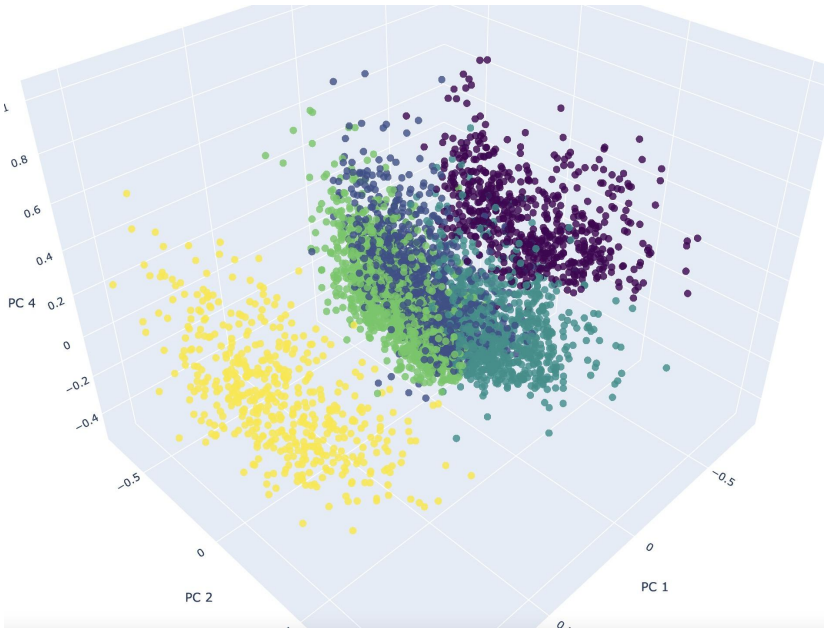


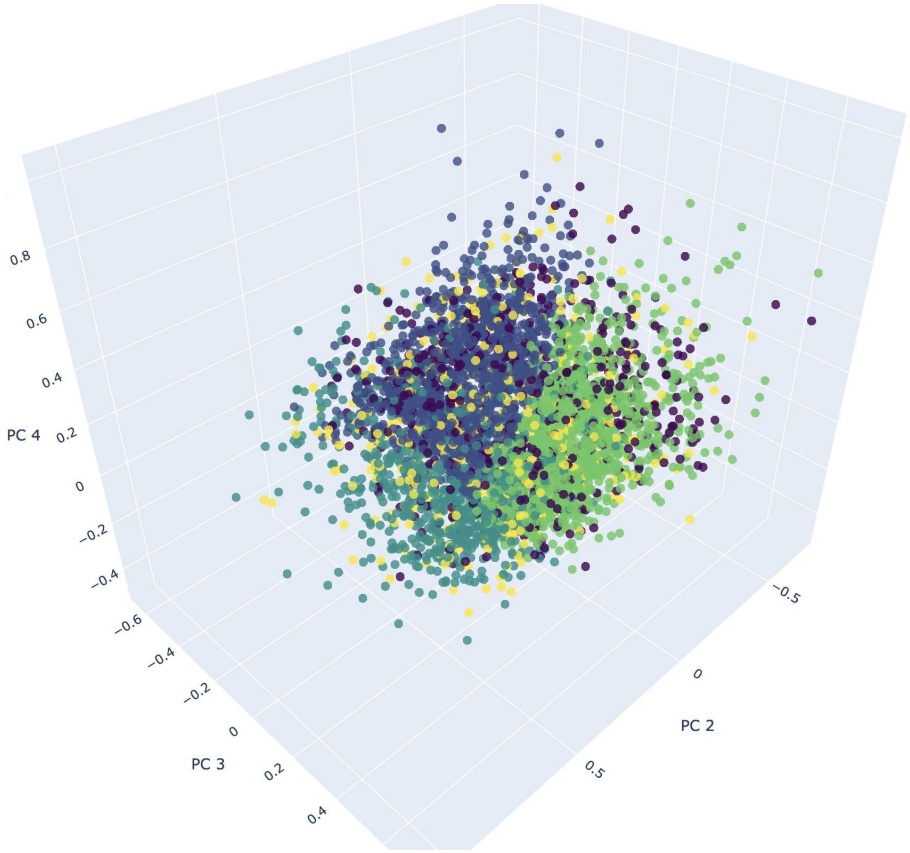
Visualization

In addition to our other features, we provide a 3-dimensional plot using the Plotly package to visually represent the distance between clusters. We have also generated relational graphs for both 5-cluster and 3-cluster PCA. To further illustrate the original variables, we have selected the year employed, credit amount of loans, and income of clients, and their corresponding visualizations are displayed below.

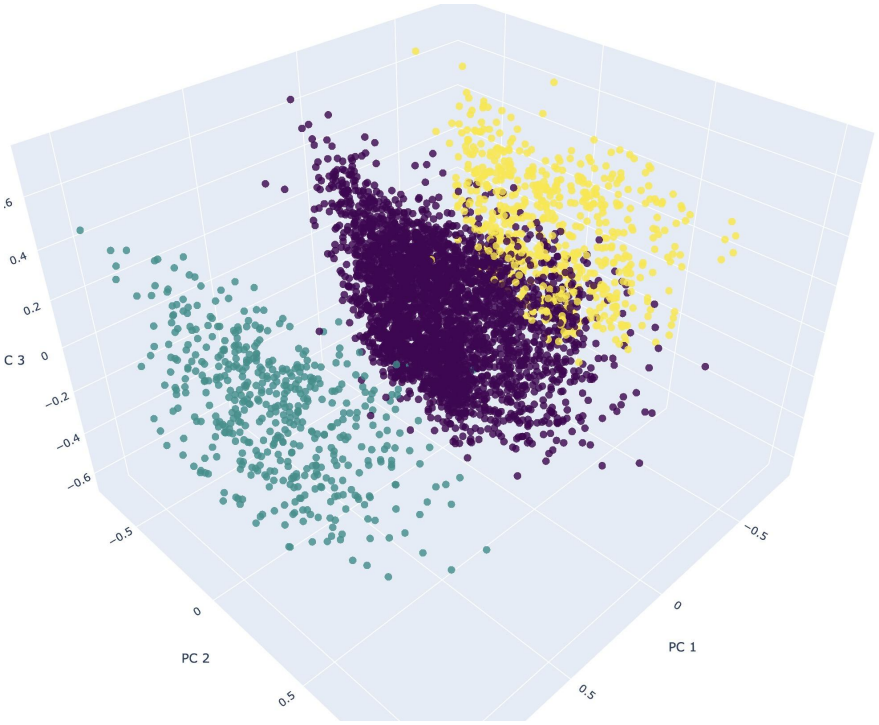
PCA 5 Clusters(K-mean):

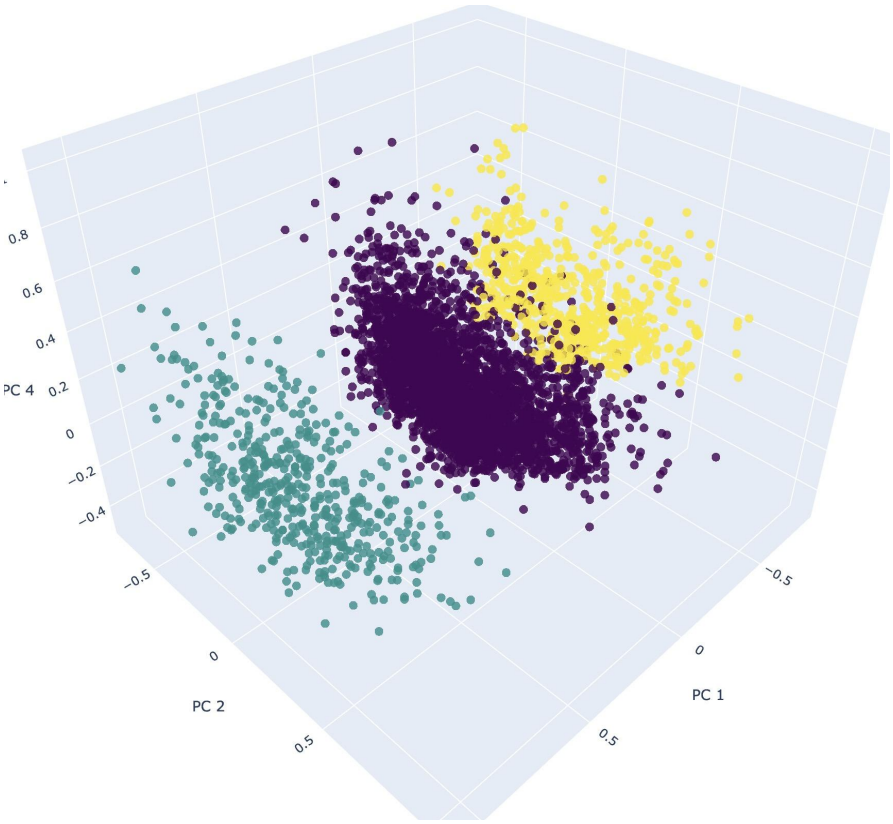


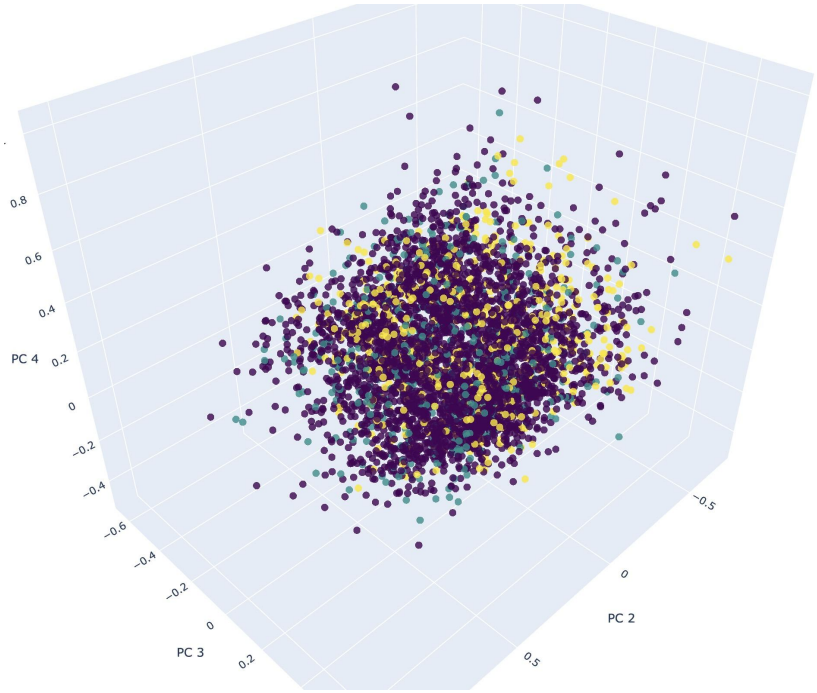



PC 4
0.8
0.6
0.4
0.2
0
−0.2
−0.4
−0.6
PC 3
−0.4
−0.2
0
0.2
0.4
PC 2
−0.5
0
0.5

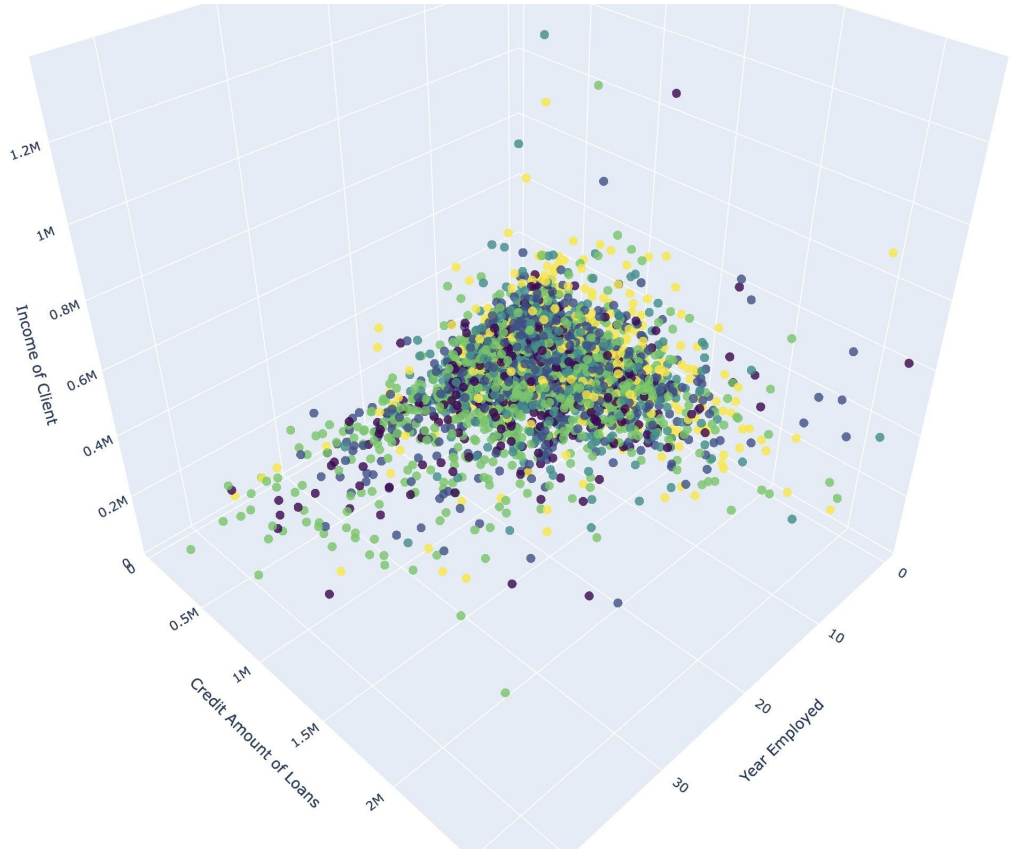
PCA 3 Clusters(Hierarchical):



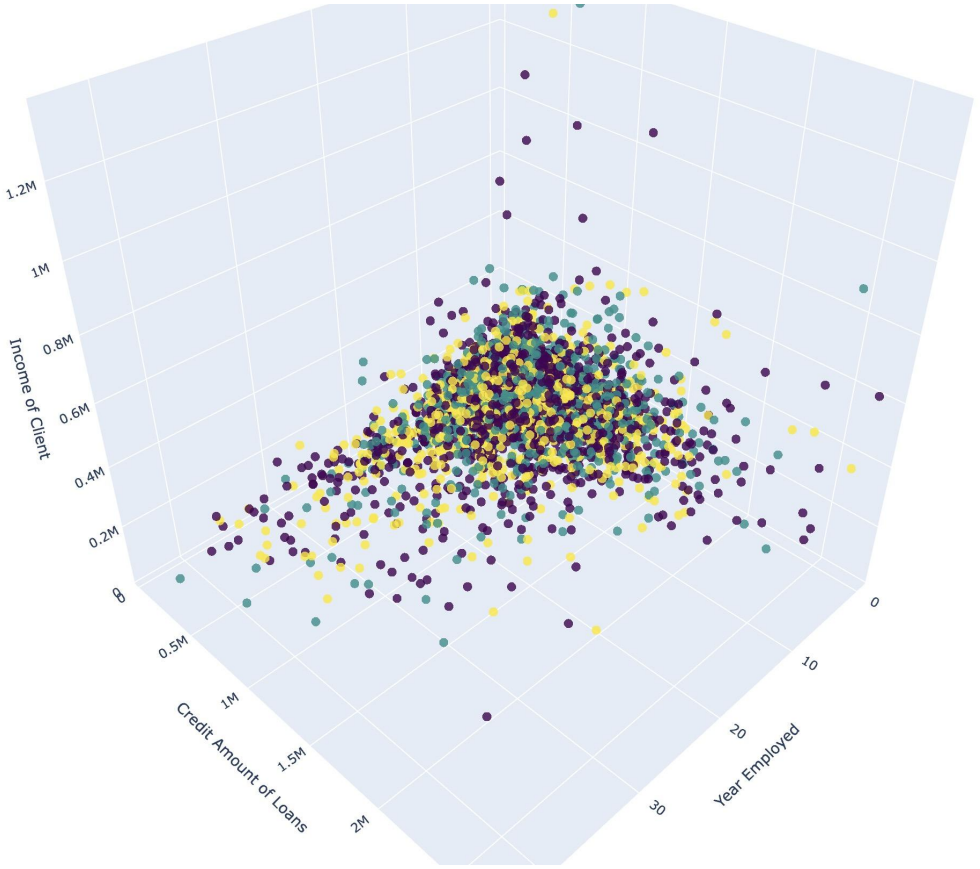



PC 4
0.8
0.6
0.4
0.2
0
−0.2
−0.4
−0.6
PC 3
−0.4
−0.2
0
0.2
PC 2
−0.5
0
0.5

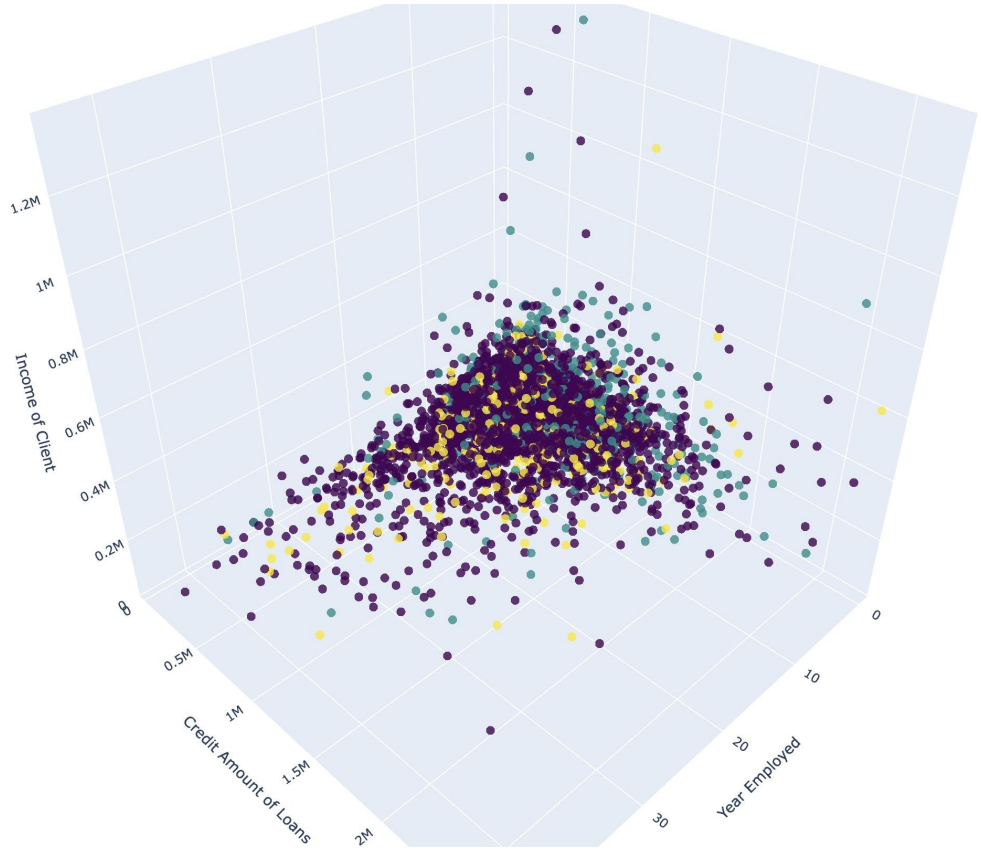
K-mean cluster for original variables:



K-Prototypes cluster for original variables:



Hierarchy cluster for original variables:



Model Results Comparison

The metrics we used to compare the performance of different clustering models are the Calinski-Harabasz index and the Davies-Bouldin index.

The Calinski-Harabasz index, also known as the Variance Ratio Criterion, measures the ratio of the between-cluster variance to the within-cluster variance. A higher Calinski-Harabasz index indicates better clustering performance, as it indicates that the clusters are well-separated and compact.

On the other hand, The Davies-Bouldin index, on the other hand, measures the average similarity between each cluster and its most similar cluster, relative to the average dissimilarity between each cluster and its least similar cluster. A lower Davies-Bouldin index indicates better clustering performance, as it indicates that the clusters are well separated and distinct.

	K-Means	K-Prototypes	Hierarchical (cat. features)	Hierarchical (no cat. features)
Calinski-Harabasz index	1795.113	116.934	1430.120	1638.634
Davies-Bouldin index	1.086	4.700	1.788	1.132

What is worth noticing is that the effect for both numerical and categorical variables is worse than only incorporating numerical variables.

Also, within each clustering method, we want to see how separable each cluster is regarding the percentage of default clients. In the whole dataset, 8.3% of all customers defaulted on their loans. The table below summarizes the percentage of default customers in each cluster of each algorithm (rank for low to high in each cluster):

	Cluster 1	Cluster 2	Cluster 3	Cluster 4	Cluster 5
K-Means	4.0%	6.9%	7.6%	10.3%	11.4%
K-Prototypes	3.8%	8.1%	9.7%	10.7%	
Hierarchical (with cat. features)	5.1%	7.7%	8.3%	8.4%	12.3%
Hierarchical (without cat. features)	4.0%	8.4%	11.4%		

From this table, Hierarchical Cluster (without categorical features) are the most separable methods among all the algorithms.

Lastly, In order to understand the pattern of each cluster grouping the customers, we need to have some interpretability regarding our numeric features. So, it is crucial to

figure out the importance of each feature in the PCA, which can help us understand the characteristics of each cluster. The top five important numeric features are:

- How many family members do the client have
- Number of children the client has
- Credit amount of the loan
- Goods price of the good that the client asked for on the previous application
- Our rating of the region where the client lives

Findings and Conclusion

By comparing the silhouette score of the three models, we found that the K-means clustering is performing better than all other methods considering all cluster quality measurements. Therefore, we chose the result of the K-means (5 clusters) for our customer segmentation. By calculating the proportion of people in each cluster that went to default, we found that those five clusters had distinct rates of default (4.0%, 6.9%, 7.6%, 10.3%, 11.4%), suggesting our algorithm has captured some key factors influencing default risk. Utilization of the risk rates requires business analysts to build loaning strategies for each cluster separately. Depending on different risk rates, different loan limits and interest rates may be generated for each segment.

Reference

Brown, I. & Mues, C. (2011). An experimental comparison of classification algorithms for imbalanced credit scoring data sets.Expert Systems with Applications. 39-3. doi: 10.1016/j.eswa.2011.09.033

Gholamian, M. R., Jahanpour, S., & Sadatrasoul, S. M (2013). A New Method for Clustering in Credit Scoring Problems. Journal of Mathematics and Computer Science. 6-97.