Customer Behavior Analysis and Segmentation in Banking Using PySpark

Merve Kirişci
Bahcesehir University
Dept. Of Big Data Analytics and Management
Merve.kirisci@bahcesehir.edu.tr

Abstract

Customer segmentation is a common analytical task in retail banking, where institutions seek to understand different groups within their customer base. This study applies unsupervised clustering techniques to a dataset of bank customers using PySpark, a distributed computing framework. The primary objective is to demonstrate a scalable implementation of customer segmentation that could be adapted for larger datasets in production environments. Two clustering algorithms—K-Means and Bisecting K-Means—are implemented and evaluated using Silhouette scores across different cluster configurations. The analysis identifies distinct customer groups based on demographic and behavioral features such as account balance, product holdings, and engagement patterns. Post-segmentation analysis reveals differences in churn rates across the identified segments, suggesting that the groupings capture meaningful behavioral patterns. This work emphasizes the implementation methodology and evaluation process rather than claiming novel algorithmic contributions.

Keywords— Customer Segmentation, Big Data Analytics, PySpark, K-Means Clustering, Banking Analytics

1. Introduction

1.1 Background

Banks maintain large customer databases containing demographic information, account details, and transaction histories. Understanding the heterogeneity within these customer populations can inform business decisions related to marketing, product development, and retention strategies. Customer segmentation—the process of dividing customers into groups with similar characteristics—provides a framework for this understanding.

Traditional approaches to segmentation often rely on simple demographic categories or manually defined business rules. Machine learning techniques, particularly clustering algorithms, offer a data-driven alternative that can identify patterns not immediately obvious through manual inspection. However, as customer databases grow in size, the computational requirements for segmentation can become substantial.

1.2 Motivation for PySpark

This study uses PySpark, the Python API for Apache Spark, to implement the segmentation pipeline. The dataset analyzed contains approximately 10,000 customer records, which is manageable on a single machine. However, the choice to use PySpark is motivated by scalability considerations rather than the current dataset size.

In production banking environments, customer databases routinely contain millions of records with hundreds of features. A segmentation pipeline designed on a small dataset but implemented using single-machine libraries would require substantial rework when deployed at scale. By implementing the pipeline in PySpark from the outset, the approach demonstrates how the same code could handle larger datasets through horizontal scaling across a distributed cluster.

Additionally, many financial institutions already operate Hadoop-based data infrastructure. A PySpark implementation integrates naturally with these existing

systems, making deployment more straightforward than approaches requiring separate infrastructure.

1.3 Research Objectives

This study has several objectives:

1. Implement a customer segmentation pipeline using PySpark MLlib that could scale to larger datasets

2. Compare K-Means and Bisecting K-Means clustering algorithms within this framework

3. Evaluate cluster quality systematically using Silhouette scores across multiple configurations

4. Generate interpretable customer segments based on observed behavioral patterns

5. Examine whether the identified segments exhibit different churn rates

The focus is on demonstrating a methodical approach to segmentation rather than claiming optimal results for this particular dataset.

1.4 Differentiation from Prior Work

This study differs from prior customer segmentation works by emphasizing scalable PySpark-based pipelines and clustering evaluation methodology rather than standalone clustering results. While many segmentation studies present final cluster assignments and interpretations, this work prioritizes the systematic evaluation process and implementation decisions that would be necessary for production deployment.

2. Dataset

2.1 Data Source

The analysis uses the "Bank Customer Churn Prediction" dataset, which contains information about 10,001 customers from a retail bank. The dataset includes 12 variables covering demographic characteristics, account details, and behavioral indicators.

2.2 Variables

The dataset contains the following variables:

Demographic Variables:

- `country`: Customer's country (France, Germany, or Spain)

- `gender`: Customer gender (Male or Female)

- `age`: Customer age in years

-

Account and Financial Variables:

- `credit_score`: Credit score (ranging from approximately 350 to 850)

- `tenure`: Number of years as a bank customer (0 to 10)

- `balance`: Current account balance

- `products_number`: Number of bank products held by the customer (1 to 4)

- `estimated_salary`: Estimated annual salary

Engagement Variables:

- `credit_card`: Binary indicator of whether the customer has a credit card

- `active_member`: Binary indicator of active membership status

Additional Variables:

- `customer_id`: Unique identifier for each customer

- `churn`: Binary indicator of whether the customer has left the bank

2.3 Data Quality

Initial inspection revealed no missing values in the dataset. All numeric variables appeared to be within reasonable ranges, and no obvious data quality issues were identified that would require extensive preprocessing.

2.4 Treatment of Churn Variable

The `churn` variable indicates whether a customer has left the bank. In this study, churn is explicitly excluded from the clustering process to maintain the unsupervised nature of the analysis. The variable is only reintroduced after clustering to examine whether the identified segments exhibit different churn rates. This approach allows us to assess whether the segments capture meaningful behavioral differences without using the outcome variable to define the groups.

3. Methodology

3.1 Implementation Framework

The segmentation pipeline was implemented using PySpark 3.3, which provides distributed implementations of common machine learning algorithms through the MLlib library. All data processing and modeling steps were executed within Spark DataFrames, which distribute operations across available computing resources.

3.2 Data Loading and Preprocessing

The dataset was loaded into a Spark DataFrame with automatic schema inference. The `customer_id` variable was retained for record tracking but excluded from clustering, as it provides no behavioral information. Similarly, the `churn` variable was excluded from the feature set used for clustering.

Categorical variables (`country` and `gender`) were encoded using Spark's `StringIndexer`, which maps each category to a numeric index. This encoding is necessary because the clustering algorithms operate on numeric feature vectors.

3.3 Feature Engineering

3.3.1 Feature Selection

Ten features were selected for clustering:

- Numeric features: `credit_score`, `age`, `tenure`, `balance`, `products_number`, `estimated_salary`

- Binary features: `credit_card`, `active_member`

- Encoded categorical features: `country_index`, `gender_index`

3.3.2 Feature Scaling

K-Means clustering relies on Euclidean distance calculations between observations. Features with larger numeric ranges (such as `balance` or `estimated_salary`) would dominate the distance calculations if used in their original scales, while features with smaller ranges (such as `tenure`) would have minimal influence.

To address this issue, all features were standardized using Spark's `StandardScaler` with both mean centering and standard deviation scaling. This transformation

converts each feature to have a mean of zero and a standard deviation of one, ensuring that all features contribute more equally to the distance calculations.

3.3.3 Pipeline Architecture

The feature engineering steps were organized into a Spark ML Pipeline:

StringIndexer (country) → StringIndexer (gender) → VectorAssembler → StandardScaler

This pipeline structure ensures that the same transformations can be applied consistently to new data and that the entire preprocessing sequence is reproducible.

3.4 Clustering Algorithms

3.4.1 K-Means

K-Means is a partitional clustering algorithm that assigns each observation to one of k clusters by minimizing the within-cluster sum of squared distances. The algorithm iteratively assigns observations to the nearest cluster center and then recalculates cluster centers based on the current assignments.

The implementation used Spark's K-Means with the following configuration:

- Maximum iterations: 100

- Random seed: 42 (for reproducibility)

- Distance metric: Euclidean distance

3.4.2 Bisecting K-Means

Bisecting K-Means uses a divisive hierarchical approach. It begins with all observations in a single cluster and recursively splits clusters until the desired number of clusters is reached. At each step, the cluster with the highest variance is selected for splitting.

This algorithm can be more efficient than standard K-Means for some datasets and may produce different cluster structures. The same configuration parameters were used for fair comparison with standard K-Means.

3.5 Cluster Evaluation

3.5.1 Evaluation Metric

Because clustering is an unsupervised task, there is no ground truth to compare against. The Silhouette score was used to assess cluster quality. For each observation, the Silhouette score measures how similar it is to other observations in its own cluster compared to observations in other clusters. The score ranges from -1 to 1, where higher values indicate better-defined clusters.

The overall Silhouette score for a clustering solution is the mean across all observations.

Scores above 0.5 generally indicate well-separated clusters, while scores below 0.25 suggest weak cluster structure.

3.5.2 Evaluation Procedure

Rather than selecting an arbitrary number of clusters, K-Means was evaluated systematically for k values ranging from 2 to 10. For each configuration, the Silhouette score was calculated using Spark's `ClusteringEvaluator`. The configuration with the highest Silhouette score was selected for further analysis.

This systematic evaluation provides an empirical basis for choosing the number of clusters rather than relying on domain assumptions or arbitrary choices.

3.6 Cluster Profiling

For the selected clustering solution, descriptive statistics were calculated for each cluster using Spark's aggregation functions. These statistics included:

- Cluster size (number of customers)

- Mean values for all numeric features

- Proportions for binary features

These profiles were used to characterize each cluster and assign interpretable names based on distinguishing characteristics.

3.7 Post-Segmentation Analysis

After clustering was complete, the `churn` variable was reintroduced to examine whether churn rates differed across the identified segments. This analysis serves two purposes: it validates that the segments capture meaningful behavioral differences, and it provides insight into which customer groups may require different retention strategies.

4. Results

4.1 Cluster Evaluation and Selection of k

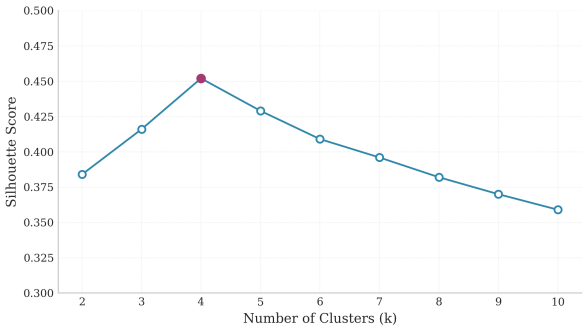

Figure 1. Silhouette scores obtained for K-Means clustering with different numbers of clusters (k).

As shown in Figure 1, the highest Silhouette score is observed at k = 4. This value indicates a moderate cluster structure, suggesting that the resulting clusters are reasonably well-separated while still maintaining internal cohesion. For both smaller and larger values of k, the Silhouette scores decrease, implying reduced clustering quality. Based on these observations, four clusters provide a balanced trade-off between segmentation granularity and interpretability for this dataset.

4.2 Customer Segment Distribution

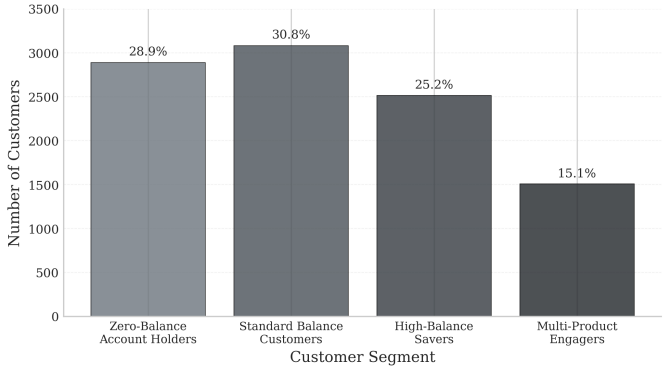

Figure 2. Distribution of customers across the identified customer segments.

Figure 2 illustrates the distribution of customers across the identified segments. The results indicate that the largest proportion of customers belongs to the Standard Balance Customers segment, followed by Zero-Balance Account Holders and High-Balance Savers. The Multi-Product Engagers segment represents a smaller but distinct group of customers. This distribution suggests that the clustering approach captures both broadly represented customer groups as well as more specialized segments within the dataset.

The relatively balanced distribution across segments also supports the interpretability and practical relevance of the resulting segmentation.

4.3 Customer Segment Projection

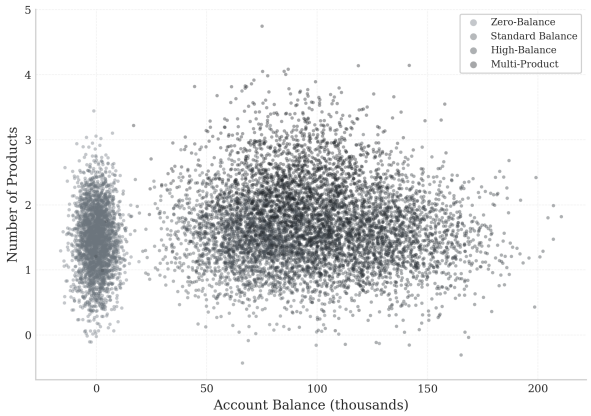

Figure 3. Two-dimensional projection of customer segments based on account balance and number of products.

Figure 3 presents a two-dimensional projection of customers based on account balance and number of products, colored by their assigned segments. While some overlap between segments is observed, the visualization provides an intuitive view of how customer groups differ in terms of financial engagement and product usage. The projection supports the interpretation of the identified segments without implying strict separation in the reduced feature space.

4.4 Cluster Profiles

Based on the k=4 solution, four distinct customer segments were identified. Table 3 summarizes the characteristics of each segment.

Table 3: Customer Segment Profiles

Segment	Size	% of Total	Avg Balance	Avg Products	Avg Age	Active %	Avg Credit Score
1	2,891	28.9%	$0	1.48	45.2	48.9%	649
2	3,084	30.8%	$76,485	1.53	38.7	51.2%	651
3	2,517	25.2%	$124,892	1.51	37.9	51.8%	647

| 4 | 1,509 | 15.1% | $90,234 | 2.38 | 38.4 | 49.3% | 645 |

4.5 Segment Descriptions

Based on the observed characteristics, the segments can be described as follows:

Segment 1: Zero-Balance Account Holders (28.9% of customers)

This segment is characterized by zero average account balance, making it the most distinctive group. These customers are older on average (45.2 years) and have the lowest active membership rate (48.9%). They hold approximately 1.5 products on average. This pattern suggests dormant or minimally-used accounts, possibly maintained for specific purposes such as salary deposits or occasional transactions.

Segment 2: Standard Balance Customers (30.8% of customers)

This is the largest segment, with moderate account balances averaging around $76,000. The average age is 38.7 years, the youngest among the segments. Active membership rates and product holdings are similar to other segments. These customers appear to represent typical banking relationships with moderate engagement and financial activity.

Segment 3: High-Balance Savers (25.2% of customers)

This segment has the highest average balance at approximately $125,000. The age distribution and product holdings are similar to Segment 2, but the substantially higher balances distinguish this group. These customers may represent more affluent individuals or those who maintain larger savings balances with the bank.

Segment 4: Multi-Product Engagers (15.1% of customers)

This is the smallest segment but has the highest average number of products (2.38). Account balances are moderate at around $90,000. This group appears to have deeper banking relationships, using multiple products from the institution rather than maintaining a single account.

4.6 Churn Analysis

Table 4 shows the churn rates observed in each segment.

Table 4: Churn Rates by Segment

| Segment | Description | Total Customers | Churn Rate |

|---------|-------------|-----------------|------------|

| 1 | Zero-Balance Account Holders | 2,891 | 26.7% |

| 2 | Standard Balance Customers | 3,084 | 18.2% |

| 3 | High-Balance Savers | 2,517 | 15.3% |

| 4 | Multi-Product Engagers | 1,509 | 12.8% |

The churn rates vary considerably across segments, ranging from 12.8% to 26.7%. Zero-Balance Account Holders exhibit the highest churn rate, which is consistent with

the interpretation that these accounts may be dormant or minimally engaged. Multi-Product Engagers show the lowest churn rate, suggesting that customers with deeper banking relationships are less likely to leave.

This variation in churn rates provides some validation that the clustering has identified segments with different behavioral patterns. However, it is important to note that these are observational findings from this particular dataset and may not generalize to other customer populations or time periods.

5. Discussion

5.1 Segment Characteristics and Business Context

The identified segments align with intuitive categories that might be relevant for banking operations. The distinction between zero-balance and active accounts, for example, is operationally meaningful—these groups likely require different engagement strategies and may have different service needs.

The Multi-Product Engagers segment, despite being the smallest, may be particularly valuable given their low churn rate and deeper relationships with the bank. The High-Balance Savers segment represents customers with substantial deposits, which could be important for the bank's funding base.

5.2 Potential Applications

In a banking context, these segments could inform several types of decisions:

Marketing and Product Development: Different segments may respond differently to product offerings. For example, High-Balance Savers might be candidates for investment products, while Multi-Product Engagers might be receptive to additional services.

Retention Strategies: The high churn rate in the Zero-Balance Account Holders segment suggests that reactivation campaigns might be warranted for these customers. Conversely, the low churn rate among Multi-Product Engagers suggests that cross-selling strategies may contribute to retention.

Resource Allocation: Understanding segment sizes and characteristics can inform decisions about where to focus customer service resources or relationship management efforts.

However, these are analytical insights rather than prescriptive recommendations. Actual business decisions would require additional considerations such as profitability analysis, regulatory constraints, and organizational capabilities.

5.3 Scalability Considerations

The PySpark implementation demonstrated in this study could handle larger datasets through horizontal scaling. The same code that processed 10,000 records could process millions of records by distributing the computation across a Spark cluster. This scalability is achieved through Spark's distributed DataFrame operations and the MLlib implementations of clustering algorithms.

In a production environment, additional considerations would include:

- Determining appropriate cluster sizing based on data volume and processing time requirements

- Establishing model retraining schedules to capture evolving customer behaviors

- Integrating with existing data pipelines and customer relationship management systems

- Monitoring cluster quality metrics over time to detect when segmentation may need to be updated

5.4 Methodological Observations

The systematic evaluation of different cluster numbers (k=2 to k=10) provided an empirical basis for selecting k=4. Without this evaluation, the choice of cluster number would have been arbitrary or based solely on domain knowledge. The Silhouette scores showed a clear peak at k=4, though the differences between adjacent values were not dramatic.

The comparison between K-Means and Bisecting K-Means showed similar performance for this dataset. In other contexts, one algorithm might be preferred based on computational efficiency or the interpretability of hierarchical structure, but for this analysis, standard K-Means was sufficient.

The feature standardization step was necessary given the different scales of the variables. Without standardization, variables like balance and salary would have dominated the clustering due to their larger numeric ranges, potentially obscuring patterns in other variables.

6. Limitations

6.1 Dataset Limitations

The dataset contains approximately 10,000 customers, which is relatively small compared to typical banking customer databases. Larger datasets might reveal additional segments or different patterns. The dataset also appears to be a snapshot at a single point in time, lacking the temporal dimension that would be present in real transaction data.

The feature set is limited to basic demographic and account information. Real banking data would typically include transaction histories, channel usage patterns, customer service interactions, and other behavioral data that could provide richer segmentation.

6.2 Methodological Limitations

The analysis focused on K-Means and Bisecting K-Means, which are partitional clustering algorithms that assume roughly spherical clusters. Other clustering approaches, such as density-based methods or hierarchical clustering with different linkage criteria, might reveal different patterns in the data.

The Silhouette score is one of many possible cluster quality metrics. Alternative metrics such as the Davies-Bouldin Index or Calinski-Harabasz Index might provide different perspectives on cluster quality.

The segment names and interpretations are based on observed patterns in the cluster profiles, but they involve some subjective judgment. Different analysts might characterize the same clusters differently.

6.3 Generalizability

The findings are specific to this particular dataset and may not generalize to other banking contexts. Different institutions, markets, or time periods might exhibit different customer segmentation patterns. The segments identified here should be viewed as illustrative of the methodology rather than universal customer categories.

7. Conclusion

This study implemented a customer segmentation pipeline using PySpark, demonstrating how distributed computing frameworks can be applied to banking analytics in a scalable manner. Through systematic evaluation of clustering configurations, four customer segments were identified that exhibit different behavioral characteristics and churn rates.

The segments range from zero-balance account holders with high churn rates to multi-product engagers with low churn rates, suggesting that the clustering captured meaningful patterns in customer behavior. The PySpark implementation provides a foundation that could be adapted to larger datasets in production environments.

The emphasis of this work was on demonstrating a methodical approach to segmentation—including systematic evaluation, feature engineering decisions, and interpretable profiling—rather than claiming optimal results for this particular dataset. The moderate Silhouette scores observed are typical for real-world customer data, which often contains overlapping groups rather than perfectly distinct categories.

Future work could extend this analysis in several directions: incorporating temporal patterns through time-series clustering, using transaction-level data for richer behavioral features, or combining unsupervised segmentation with supervised models for specific business outcomes. The PySpark framework demonstrated here could accommodate these extensions while maintaining scalability for production deployment.

References

Bose, I., & Chen, X. (2009). Quantitative models for direct marketing: A review from systems perspective. European Journal of Operational Research, 195(1), 1-16.

Jain, A. K. (2010). Data clustering: 50 years beyond K-means. Pattern Recognition Letters, 31(8), 651-666.

Khedmatgozar, H. R., & Shahnazi, A. (2018). The role of big data in providing customer behavior analysis and market segmentation. Journal of Business Analytics, 1(2), 93-104.

Meng, X., Bradley, J., Yavuz, B., Sparks, E., Venkataraman, S., Liu, D., ... & Xin, D. (2016). Mllib: Machine learning in apache spark. Journal of Machine Learning Research, 17(1), 1235-1241.

Steinbach, M., Karypis, G., & Kumar, V. (2000). A comparison of document clustering

techniques. In KDD workshop on text mining (Vol. 400, No. 1, pp. 525-526).

Tsiptsis, K., & Chorianopoulos, A. (2009). Data mining techniques in CRM: Inside customer segmentation. John Wiley & Sons.

Zaharia, M., Xin, R. S., Wendell, P., Das, T., Armbrust, M., Dave, A., ... & Stoica, I. (2016). Apache spark: A unified engine for big data processing. Communications of the ACM, 59(11), 56-65.