CUSTOMER CLUSTERING

About Data

Source: ***https://www.kaggle.com/datasets/smaryamazimiasmaroud/credit-cardholders***

Objective:

- ***Cluster customers*** for further marketing campaigns appropriately tailored (initially primary purpose of the dataset),
- ***Predit high-risk customers'*** *based on behaviors* in spending, expenditure, paying back,... (deeper implement analysis)

Supported by:

- ***Perplexity*** for inititative suggestion
- ***Gemini and Copilot*** (inside Visual Code Studio) for code modify and recommendation.

```
Data columns (total 18 columns):
 #   Column                            Non-Null Count  Dtype
---  ------                            --------------  -----
 0   CUST_ID                           8950 non-null   object
 1   BALANCE                           8950 non-null   float64
 2   BALANCE_FREQUENCY                 8950 non-null   float64
 3   PURCHASES                         8950 non-null   float64
 4   ONEOFF_PURCHASES                  8950 non-null   float64
 5   INSTALLMENTS_PURCHASES            8950 non-null   float64
 6   CASH_ADVANCE                      8950 non-null   float64
 7   PURCHASES_FREQUENCY               8950 non-null   float64
 8   ONEOFF_PURCHASES_FREQUENCY        8950 non-null   float64
 9   PURCHASES_INSTALLMENTS_FREQUENCY  8950 non-null   float64
 10  CASH_ADVANCE_FREQUENCY            8950 non-null   float64
 11  CASH_ADVANCE_TRX                  8950 non-null   int64
 12  PURCHASES_TRX                     8950 non-null   int64
 13  CREDIT_LIMIT                      8949 non-null   float64
 14  PAYMENTS                          8950 non-null   float64
 15  MINIMUM_PAYMENTS                  8637 non-null   float64
 16  PRC_FULL_PAYMENT                  8950 non-null   float64
 17  TENURE                            8950 non-null   int64
dtypes: float64(14), int64(3), object(1)
```

feature	description
custid	Unique identification of the credit cardholder (categorical)
balance	Balance amount left in the account for purchases
balancefrequency	Frequency of balance updates (0 = rarely, 1 = frequently)
purchases	Total amount spent on purchases
oneoffpurchases	Maximum single transaction purchase amount
installmentspurchases	Purchases made in installments
cash_advance	Cash advances taken by the user
purchasesfrequency	Frequency of purchases (0 = rarely, 1 = frequently)
oneoffpurchasesfrequency	Frequency of one-time purchases (0 = rarely, 1 = frequently)
purchasesinstallmentsfrequency	Frequency of installment purchases (0 = rarely, 1 = frequently)
cashadvancefrequency	Frequency of cash advances taken
cashadvancetrx	Number of cash advance transactions
purchasestrx	Number of purchase transactions
credit_limit	User's credit card limit
payments	Amount of payments made by the user
minimum_payments	Minimum payment amount made by the user
prcfullpayment	Percentage of the full payment made
tenure	Duration of credit card usage (in months)

	CUST_ID	BALANCE	BALANCE_FREQUENCY	PURCHASES	ONEOFF_PURCHASES
0	C10001	40.900749	0.818182	95.40	0.00
1	C10002	3202.467416	0.909091	0.00	0.00
2	C10003	2495.148862	1.000000	773.17	773.17
3	C10004	1666.670542	0.636364	1499.00	1499.00
4	C10005	817.714335	1.000000	16.00	16.00

INSTALLMENTS_PURCHASES	CASH_ADVANCE	PURCHASES_FREQUENCY	ONEOFF_PURCHASES_FREQUENCY
95.4	0.000000	0.166667	0.000000
0.0	6442.945483	0.000000	0.000000
0.0	0.000000	1.000000	1.000000
0.0	205.788017	0.083333	0.083333
0.0	0.000000	0.083333	0.083333

PURCHASES_INSTALLMENTS_FREQUENCY	CASH_ADVANCE_FREQUENCY	CASH_ADVANCE_TRX
0.083333	0.000000	0
0.000000	0.250000	4
0.000000	0.000000	0
0.000000	0.083333	1
0.000000	0.000000	0

PURCHASES_TRX	CREDIT_LIMIT	PAYMENTS	MINIMUM_PAYMENTS	PRC_FULL_PAYMENT	TENURE
2	1000.0	201.802084	139.509787	0.000000	12
0	7000.0	4103.032597	1072.340217	0.222222	12
12	7500.0	622.066742	627.284787	0.000000	12
1	7500.0	0.000000	NaN	0.000000	12
1	1200.0	678.334763	244.791237	0.000000	12

Quick Descriptive Statistics

	balance	balance_frequency	purchases	oneoff_purchases
count	8950.000000	8950.000000	8950.000000	8950.000000
mean	1564.474828	0.877271	1003.204834	592.437371
std	2081.531879	0.236904	2136.634782	1659.887917
min	0.000000	0.000000	0.000000	0.000000
25%	128.281915	0.888889	39.635000	0.000000
50%	873.385231	1.000000	361.280000	38.000000
75%	2054.140036	1.000000	1110.130000	577.405000
max	19043.138560	1.000000	49039.570000	40761.250000

	installments_purchases	cash_advance	purchases_frequency
count	8950.000000	8950.000000	8950.000000
mean	411.067645	978.871112	0.490351
std	904.338115	2097.163877	0.401371
min	0.000000	0.000000	0.000000
25%	0.000000	0.000000	0.083333
50%	89.000000	0.000000	0.500000
75%	468.637500	1113.821139	0.916667
max	22500.000000	47137.211760	1.000000

	oneoff_purchases_frequency	purchases_installments_frequency
count	8950.000000	8950.000000
mean	0.202458	0.364437
std	0.298336	0.397448
min	0.000000	0.000000
25%	0.000000	0.000000
50%	0.083333	0.166667
75%	0.300000	0.750000
max	1.000000	1.000000

	cash_advance_frequency	cash_advance_trx	purchases_trx	credit_limit
count	8950.000000	8950.000000	8950.000000	8950.000000
mean	0.135144	3.248827	14.709832	4494.282473
std	0.200121	6.824647	24.857649	3638.646702
min	0.000000	0.000000	0.000000	50.000000
25%	0.000000	0.000000	1.000000	1600.000000
50%	0.000000	0.000000	7.000000	3000.000000
75%	0.222222	4.000000	17.000000	6500.000000
max	1.500000	123.000000	358.000000	30000.000000

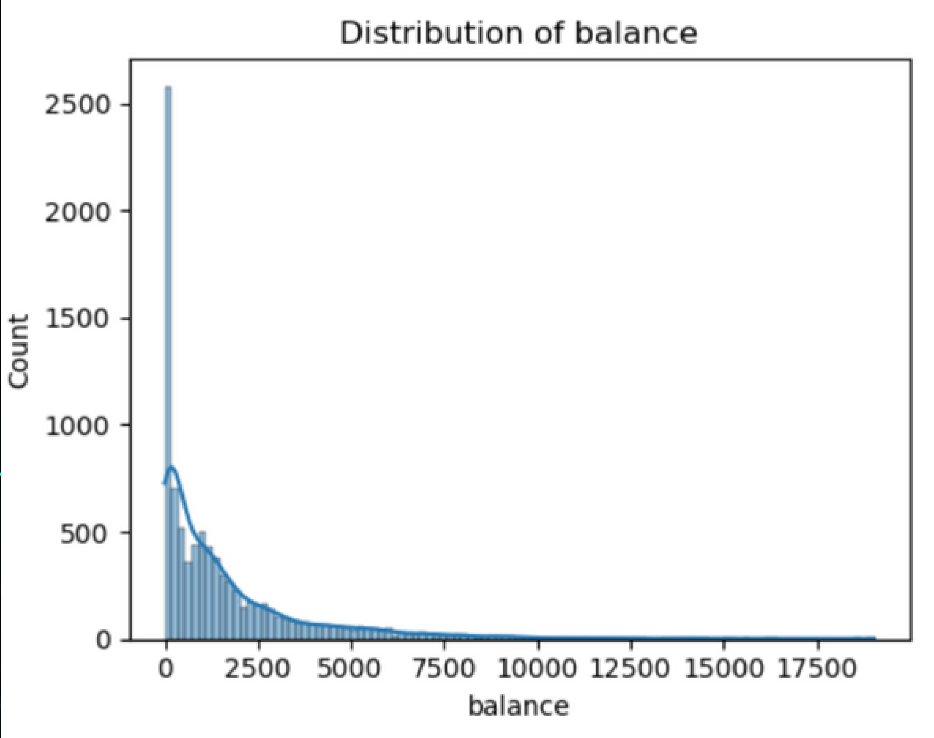
	payments	minimum_payments	prc_full_payment	tenure
count	8950.000000	8950.000000	8950.000000	8950.000000
mean	1733.143852	838.393982	0.153715	11.517318
std	2895.063757	2335.359598	0.292499	1.338331
min	0.000000	0.000000	0.000000	6.000000
25%	383.276166	164.378505	0.000000	12.000000
50%	856.901546	295.620357	0.000000	12.000000
75%	1901.134317	794.573294	0.142857	12.000000
max	50721.483360	76406.207520	1.000000	12.000000

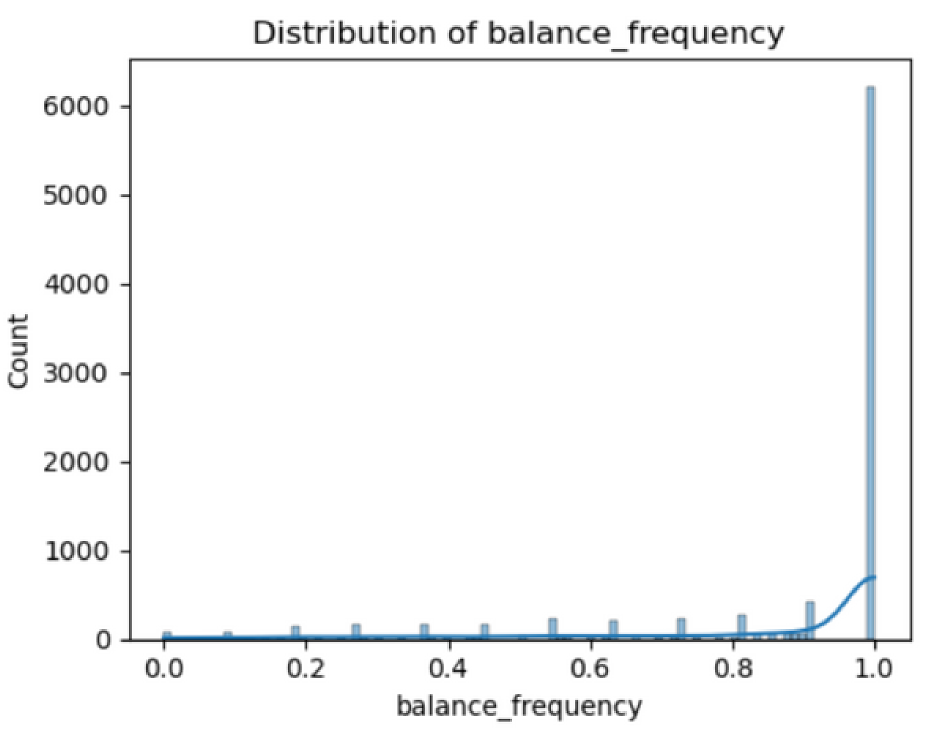
Quick Descriptive Statistics

	Skewness	Kurtosis
balance	2.393386	7.674751
balance_frequency	-2.023266	3.092370
purchases	8.144269	111.388771
oneoff_purchases	10.045083	164.187572
installments_purchases	7.299120	96.575178
cash_advance	5.166609	52.899434
purchases_frequency	0.060164	-1.638631
oneoff_purchases_frequency	1.535613	1.161846
purchases_installments_frequency	0.509201	-1.398632
cash_advance_frequency	1.828686	3.334734
cash_advance_trx	5.721298	61.646862
purchases_trx	4.630655	34.793100
credit_limit	1.522636	2.837371
payments	5.907620	54.770736
minimum_payments	13.814269	292.572223
prc_full_payment	1.942820	2.432395
tenure	-2.943017	7.694823

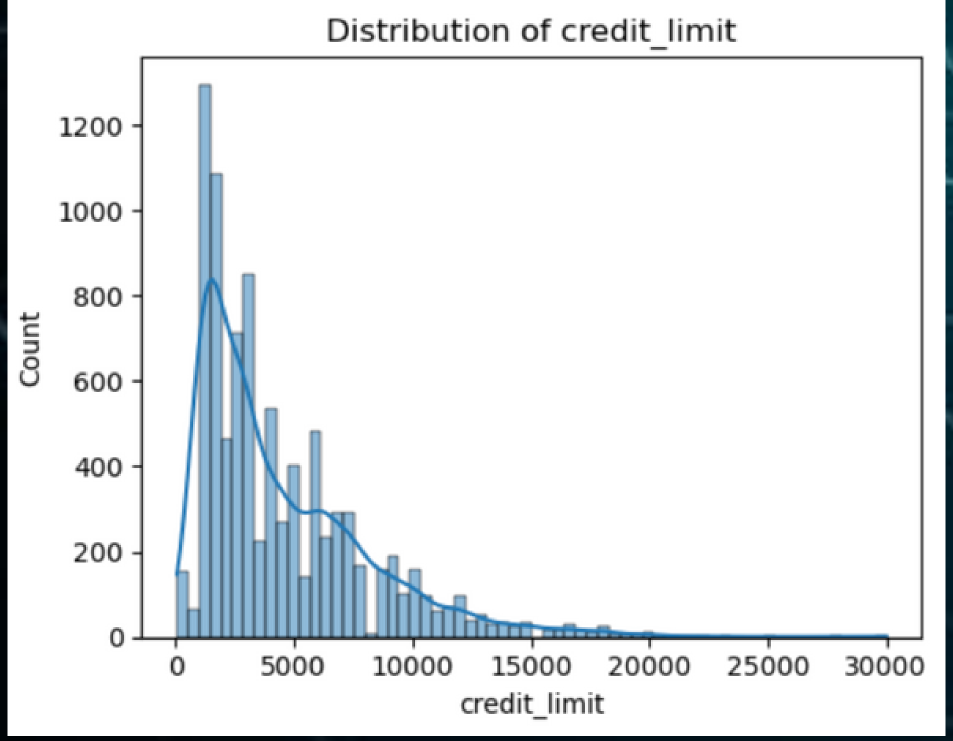
	Coefficient_of_Variation
balance	1.330499
balance_frequency	0.270047
purchases	2.129809
oneoff_purchases	2.801795
installments_purchases	2.199974
cash_advance	2.142431
purchases_frequency	0.818538
oneoff_purchases_frequency	1.473572
purchases_installments_frequency	1.090579
cash_advance_frequency	1.480799
cash_advance_trx	2.100650
purchases_trx	1.689866
credit_limit	0.809617
payments	1.670412
minimum_payments	2.785516
prc_full_payment	1.902871
tenure	0.116202

Group 1: Balance & Credit



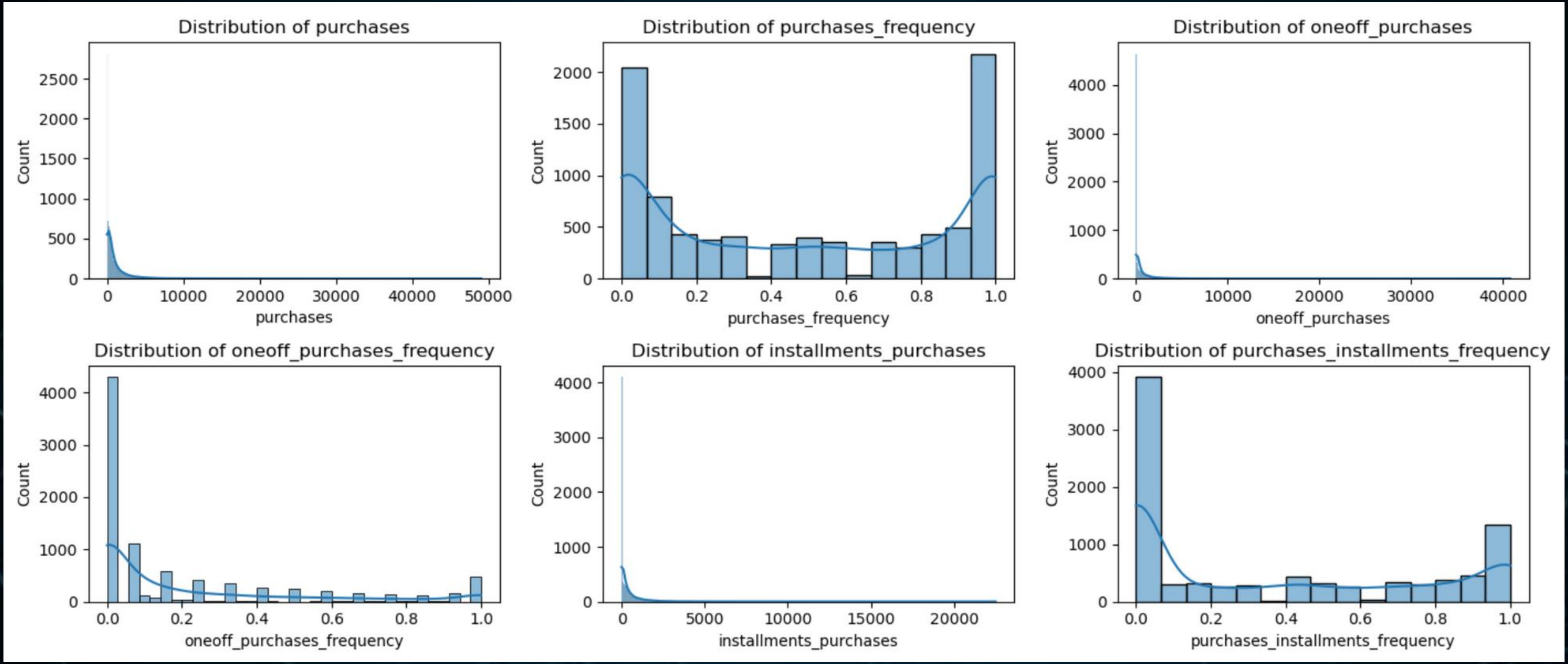


	balance	balance_frequency	credit_limit
count	8950.000000	8950.000000	8950.000000
mean	1564.474828	0.877271	4494.282473
std	2081.531879	0.236904	3638.646702
min	0.000000	0.000000	50.000000
25%	128.281915	0.888889	1600.000000
50%	873.385231	1.000000	3000.000000
75%	2054.140036	1.000000	6500.000000
max	19043.138560	1.000000	30000.000000



- Both [‘balances’] and [‘credit_limit’] show a ***strong right-skewed*** distribution.
- The majority of customers have a [‘balance_frequency’] ***very close to 1***.

Group 2: Purchases

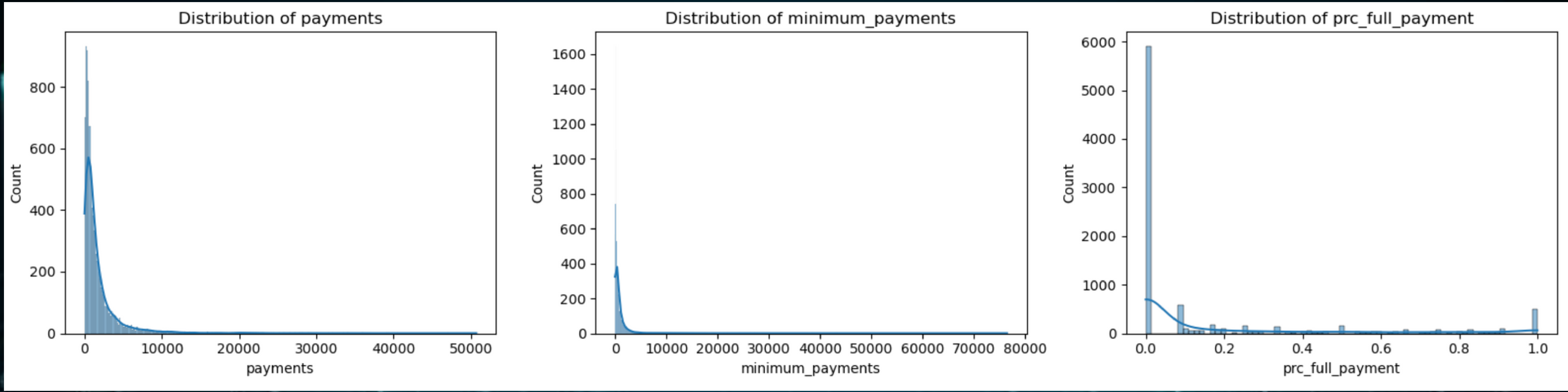


Group 2: Purchases

	purchases	purchases_frequency	oneoff_purchases	oneoff_purchases_frequency	installments_purchases	purchases_installments_frequency
count	8950.000000	8950.000000	8950.000000	8950.000000	8950.000000	8950.000000
mean	1003.204834	0.490351	592.437371	0.202458	411.067645	0.364437
std	2136.634782	0.401371	1659.887917	0.298336	904.338115	0.397448
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
25%	39.635000	0.083333	0.000000	0.000000	0.000000	0.000000
50%	361.280000	0.500000	38.000000	0.083333	89.000000	0.166667
75%	1110.130000	0.916667	577.405000	0.300000	468.637500	0.750000
max	49039.570000	1.000000	40761.250000	1.000000	22500.000000	1.000000

- ***Strongly right-skewed distributions*** for ['purchases'], ['oneoff_purchases'], and ['installments_purchases']
- ***Bimodal frequency distributions*** for ['purchases_frequency'], ['purchases_installments_frequency'] with two main customer types
- Most customers rarely make one-off purchases
- ***Highly skewed distributions*** for ['installment_frequency']

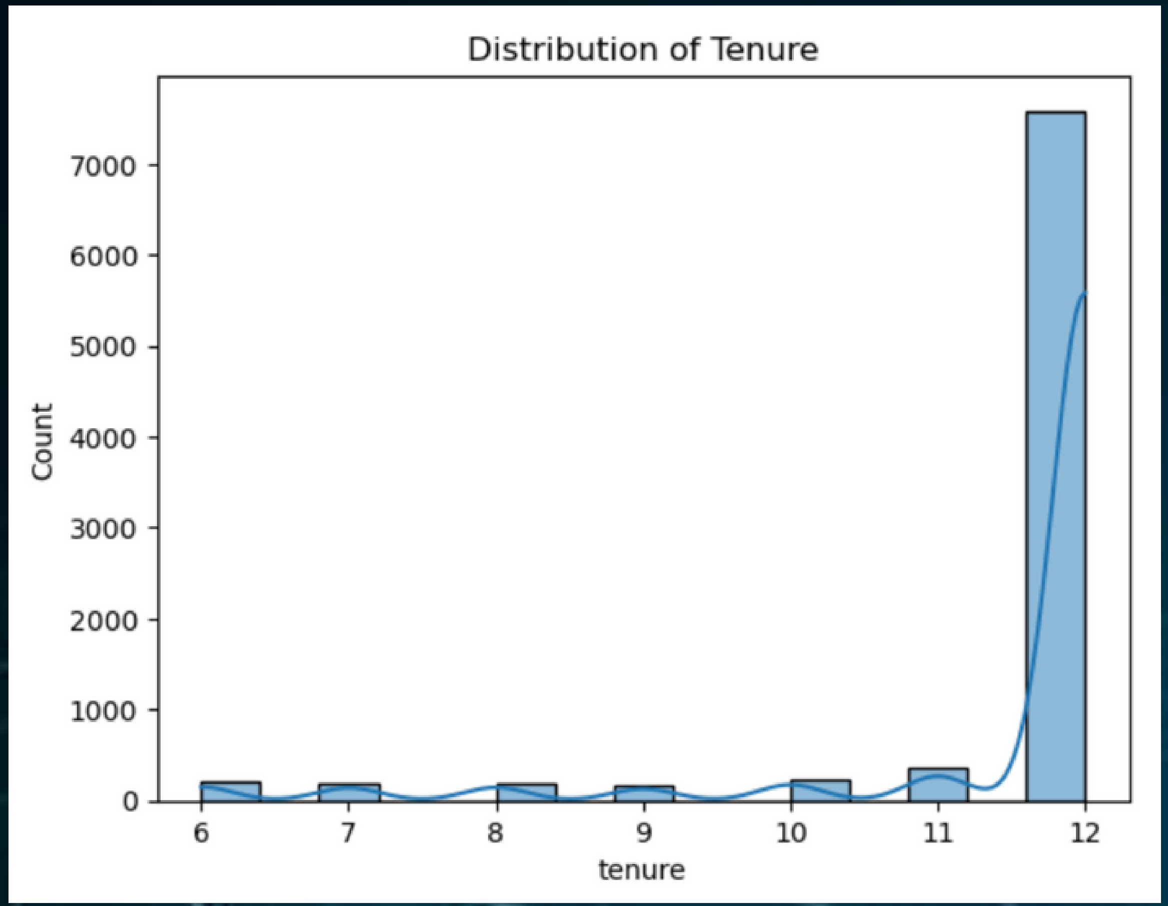
Group 3: Payment



	payments	minimum_payments	prc_full_payment
count	8950.000000	8950.000000	8950.000000
mean	1733.143852	838.393982	0.153715
std	2895.063757	2335.359598	0.292499
min	0.000000	0.000000	0.000000
25%	383.276166	164.378505	0.000000
50%	856.901546	295.620357	0.000000
75%	1901.134317	794.573294	0.142857
max	50721.483360	76406.207520	1.000000

- Both ['payments'] and ['minimum payments'] distributions are ***extremely right-skewed***
- ['prc_full_payment'] distribution is ***sharply concentrated at 0*** and also has a visible spike at 1. Most customers either never pay their balance in full (prc_full_payment ≈ 0), or always pay it off in full (prc_full_payment = 1)

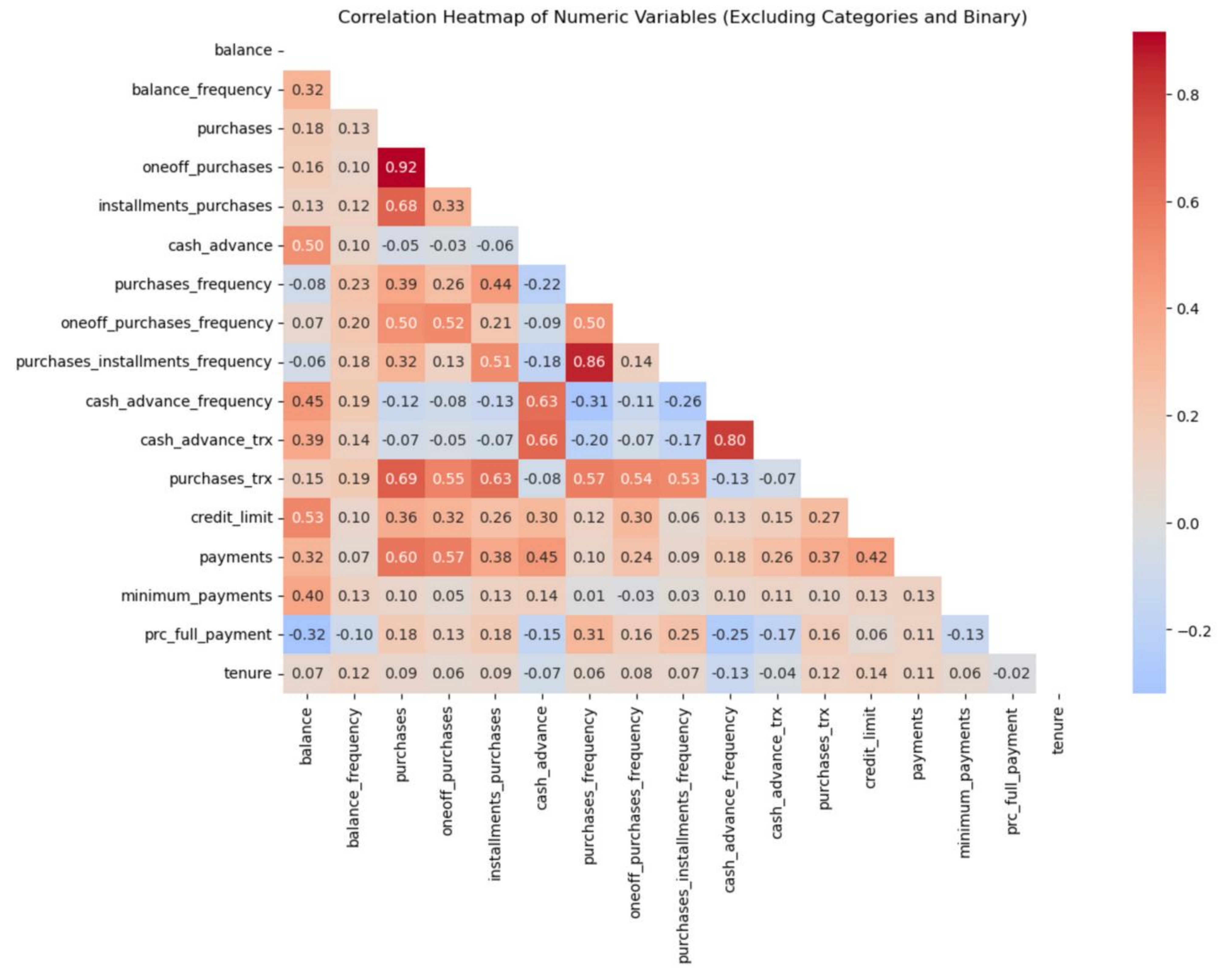
Group 4: Tenure




```
count    8950.000000
mean       11.517318
std         1.338331
min         6.000000
25%        12.000000
50%        12.000000
75%        12.000000
max        12.000000
Name: tenure, dtype: float64
```

The distribution is ***extremely right-skewed***: most customers (overwhelming majority) have a tenure of 12, while only a small number have values between 6 and 11 → most customers have reached the maximum tenure period tracked in the data, reflecting long-term engagement or a batch cohort scenario.

Relationship Detection



- Strongest Correlations:
  - Purchases & Oneoff Purchases (0.92)
  - Installments Purchases & Purchases (0.68)
- Purchase-related frequencies:
  - oneoff_purchases_frequency vs oneoff_purchases (0.52), purchases_installments_frequency vs installments_purchases (0.86)
  - Cash Advance Trx & Cash Advance Frequency (0.80)
  - Balance & Credit Limit (0.53)
- Additional Noteworthy Relationships:
  - Payments with Purchases (0.60), Oneoff (0.57), Installments (0.57):
  - Minimum Payments & Balance (0.40)
  - Cash Advance with Balance (0.50) and frequency (0.45)
- Negative or Inverse Relationships
  - Full Payment Ratio & Balance (−0.32), and Minimum Payments (−0.25):
- Weak or Uninformative Correlations
  - Tenure has no strong correlation with monetary or frequency features

Missing Values Proccessing

```
balance                              0
balance_frequency                    0
purchases                            0
oneoff_purchases                     0
installments_purchases               0
cash_advance                         0
purchases_frequency                  0
oneoff_purchases_frequency           0
purchases_installments_frequency     0
cash_advance_frequency               0
cash_advance_trx                     0
purchases_trx                        0
credit_limit                         1
payments                             0
minimum_payments                   313
prc_full_payment                     0
tenure                               0
dtype: int64
```

Missing Values Proccessing

['credit_limit']

```
balance                              0
balance_frequency                    0
purchases                            0
oneoff_purchases                     0
installments_purchases               0
cash_advance                         0
purchases_frequency                  0
oneoff_purchases_frequency           0
purchases_installments_frequency     0
cash_advance_frequency               0
cash_advance_trx                     0
purchases_trx                        0
credit_limit                         1
payments                             0
minimum_payments                   313
prc_full_payment                     0
tenure                               0
dtype: int64
```

	balance	balance_frequency	purchases	credit_limit	payments	minimum_payments
5203	18.400472	0.166667	0.0	NaN	9.040017	14.418723

Missing Values Proccessing

['credit_limit']

```
balance                             0
balance_frequency                   0
purchases                           0
oneoff_purchases                    0
installments_purchases              0
cash_advance                        0
purchases_frequency                 0
oneoff_purchases_frequency          0
purchases_installments_frequency    0
cash_advance_frequency              0
cash_advance_trx                    0
purchases_trx                       0
credit_limit                        1
payments                            0
minimum_payments                  313
prc_full_payment                    0
tenure                              0
dtype: int64
```

	balance	balance_frequency	purchases	credit_limit	payments	minimum_payments
5203	18.400472	0.166667	0.0	NaN	9.040017	14.418723

```
count     8949.000000
mean      4494.449450
std       3638.815725
min         50.000000
25%       1600.000000
50%       3000.000000
75%       6500.000000
max      30000.000000
Name: credit_limit, dtype: float64
```

Missing Values Proccessing

['credit_limit']

balance	0
balance_frequency	0
purchases	0
oneoff_purchases	0
installments_purchases	0
cash_advance	0
purchases_frequency	0
oneoff_purchases_frequency	0
purchases_installments_frequency	0
cash_advance_frequency	0
cash_advance_trx	0
purchases_trx	0
credit_limit	1
payments	0
minimum_payments	313
prc_full_payment	0
tenure	0
dtype: int64	

	balance	balance_frequency	purchases	credit_limit	payments	minimum_payments
5203	18.400472	0.166667	0.0	NaN	9.040017	14.418723

```
count     8949.000000
mean      4494.449450
std       3638.815725
min         50.000000
25%       1600.000000
50%       3000.000000
75%       6500.000000
max      30000.000000
Name: credit_limit, dtype: float64
```

```
# Median is not affacted by outliers, hence, suitable,
# because credit limit distrubition is right-skwed and the high degree in maximum value.
df.loc[5203, 'credit_limit'] = 3000
✓ 0.0s
```

Missing Values Proccessing

['minimum_payment']

```
balance                              0
balance_frequency                    0
purchases                            0
oneoff_purchases                     0
installments_purchases               0
cash_advance                         0
purchases_frequency                  0
oneoff_purchases_frequency           0
purchases_installments_frequency     0
cash_advance_frequency               0
cash_advance_trx                     0
purchases_trx                        0
credit_limit                         1
payments                             0
minimum_payments                   313
prc_full_payment                     0
tenure                               0
dtype: int64
```

Missing Values Proccessing

[‘minimum_payment’]

```
balance                              0
balance_frequency                    0
purchases                            0
oneoff_purchases                     0
installments_purchases               0
cash_advance                         0
purchases_frequency                  0
oneoff_purchases_frequency           0
purchases_installments_frequency     0
cash_advance_frequency               0
cash_advance_trx                     0
purchases_trx                        0
credit_limit                         1
payments                             0
minimum_payments                   313
prc_full_payment                     0
tenure                               0
dtype: int64
```

If [‘payment’] is 0 and [‘minimum_payment’] is missing:

It is assumed that the customer has not made any payments,

[‘minimum_payment’] is set to 0.

Missing Values Proccessing

['minimum_payment']

```
balance                              0
balance_frequency                    0
purchases                            0
oneoff_purchases                     0
installments_purchases               0
cash_advance                         0
purchases_frequency                  0
oneoff_purchases_frequency           0
purchases_installments_frequency     0
cash_advance_frequency               0
cash_advance_trx                     0
purchases_trx                        0
credit_limit                         1
payments                             0
minimum_payments                   313
prc_full_payment                     0
tenure                               0
dtype: int64
```

If ['payment'] is 0 and ['minimum_payment'] is missing:
It is assumed that the customer has not made any payments,
['minimum_payment'] is set to 0.

If ['payment'] is less than the average payment and ['minimum_payment'] is missing:
It is assumed that the minimum payment is roughly equal to the payment itself (since the customer might be paying close to the minimum).

Missing Values Proccessing

['minimum_payment']

```
balance                              0
balance_frequency                    0
purchases                            0
oneoff_purchases                     0
installments_purchases               0
cash_advance                         0
purchases_frequency                  0
oneoff_purchases_frequency           0
purchases_installments_frequency     0
cash_advance_frequency               0
cash_advance_trx                     0
purchases_trx                        0
credit_limit                         1
payments                             0
minimum_payments                   313
prc_full_payment                     0
tenure                               0
dtype: int64
```

If ['payment'] is 0 and ['minimum_payment'] is missing:
It is assumed that the customer has not made any payments,
['minimum_payment'] is set to 0.

If ['payment'] is less than the average payment and ['minimum_payment'] is missing:
It is assumed that the minimum payment is roughly equal to the payment itself (since the customer might be paying close to the minimum).

For all other cases where ['minimum_payment'] is missing:
We set it to the average of all ['payment'] to ensure that missing values are replaced with a reasonable estimate.

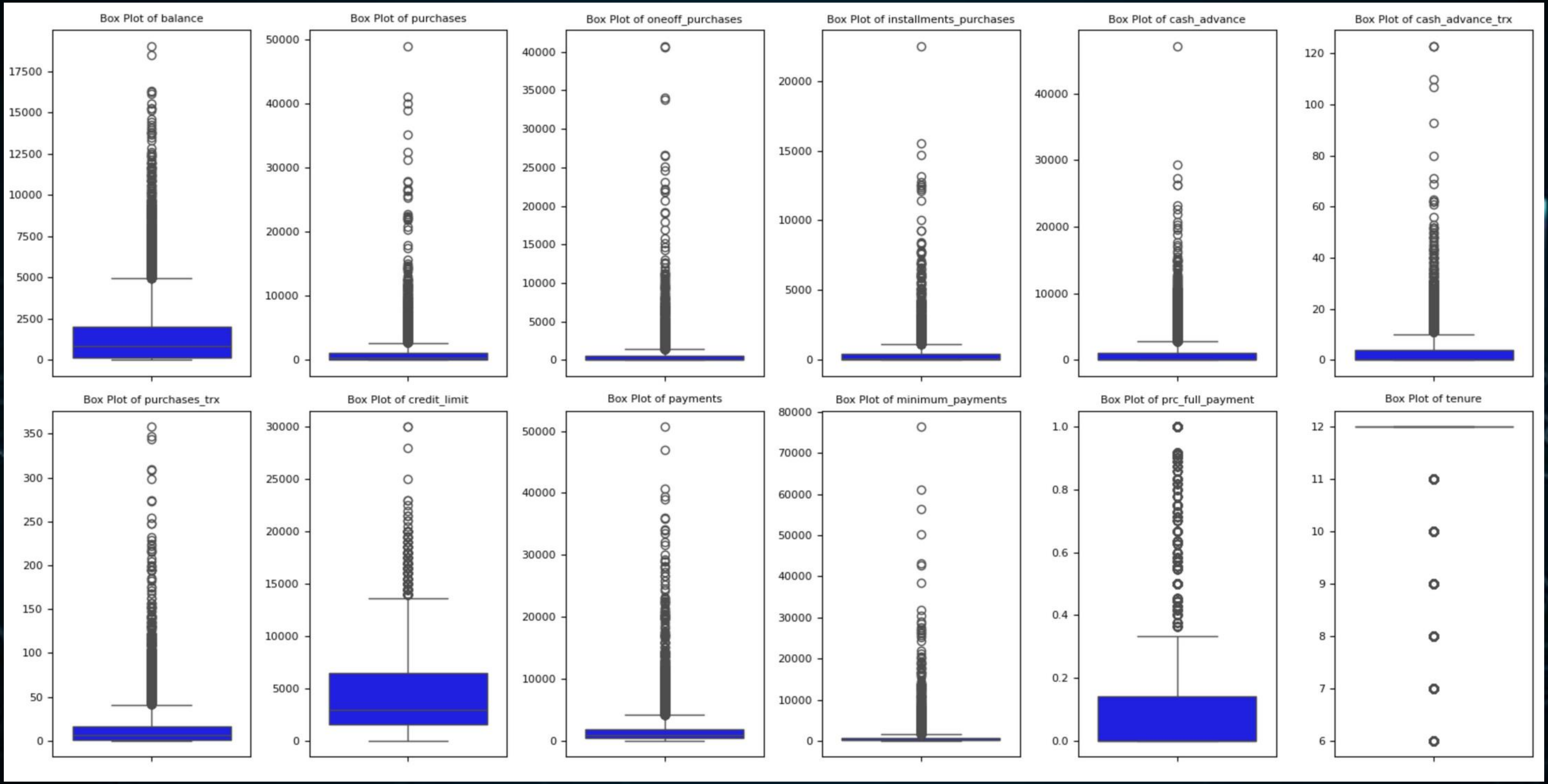
Missing Values Proccessing

['minimum_payment']

```
balance                             0
balance_frequency                   0
purchases                           0
oneoff_purchases                    0
installments_purchases              0
cash_advance                        0
purchases_frequency                 0
oneoff_purchases_frequency          0
purchases_installments_frequency    0
cash_advance_frequency              0
cash_advance_trx                    0
purchases_trx                       0
credit_limit                        1
payments                            0
minimum_payments                  313
prc_full_payment                    0
tenure                              0
dtype: int64
```

```
if min_payment_missing:
    if payment == 0:
        min_payments_filled.iloc[idx] = 0  # If no payment was made, set min_payment to 0
    elif payment < avg_payment:
        min_payments_filled.iloc[idx] = payment  # If payment is below average, use the same payment value
    else:
        min_payments_filled.iloc[idx] = avg_payment  # Otherwise, use the average payment
```

Outliers Dectection



Outliers Dectection and Log Transformation

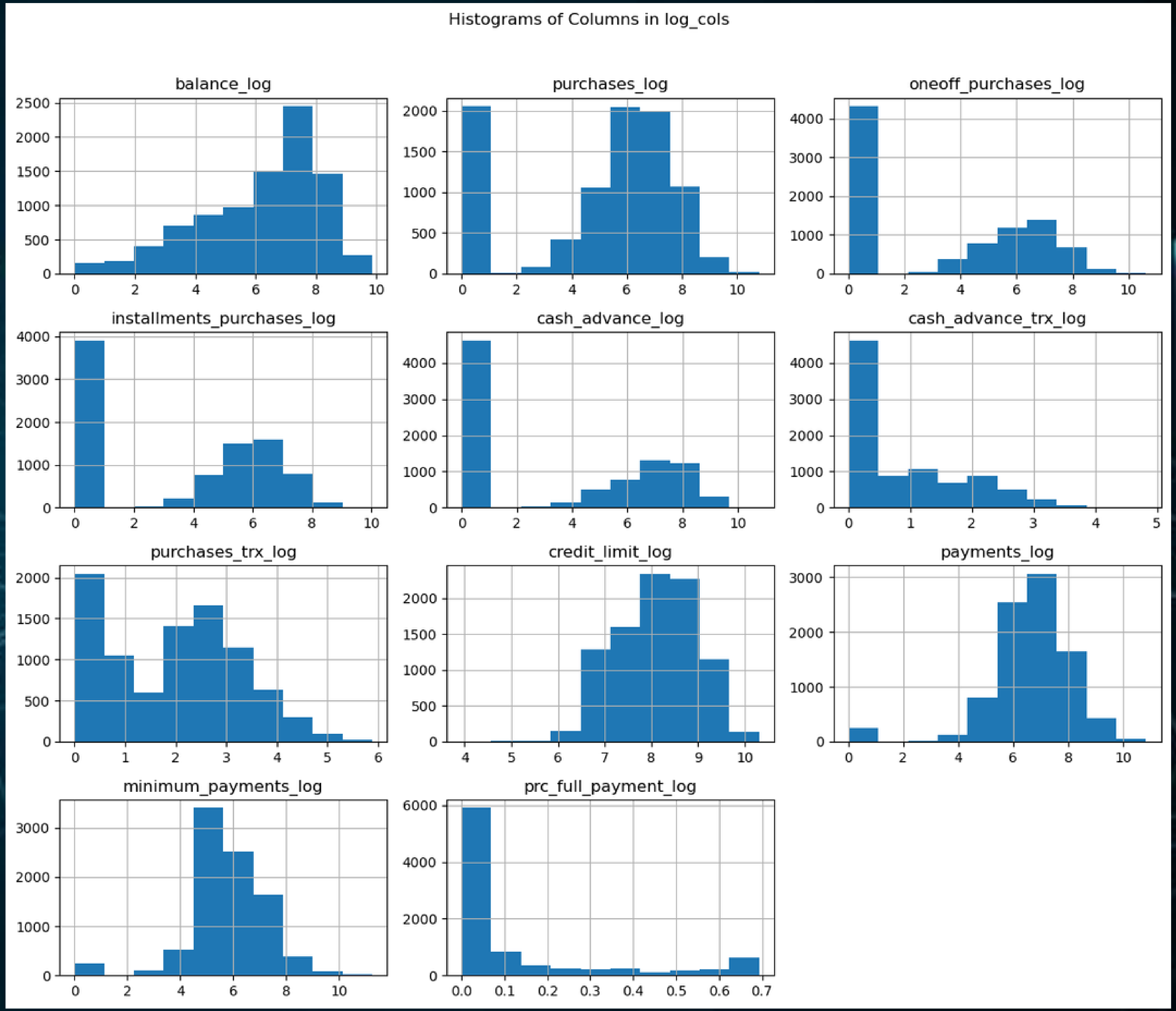
```
log_cols = []

for col in numeric_cols_filtered:
    new_col_name = f"{col}_log"
    df[new_col_name] = np.log1p(df[col])
    log_cols.append(new_col_name)
```

Outliers Dectection and Log Transformation

```python
log_cols = []

for col in numeric_cols_filtered:
    new_col_name = f"{col}_log"
    df[new_col_name] = np.log1p(df[col])
    log_cols.append(new_col_name)
```



Standard Scaling Features

Distance-based clustering methods will be deployed in the project:

1. K-Means

2.MiniBatch K-Means

3.Hierarchical

4.GMM - Gaussian Mixture Model

5.Spectral

6.DBSCAN

7.HDBSCAN

Standard Scaling Features

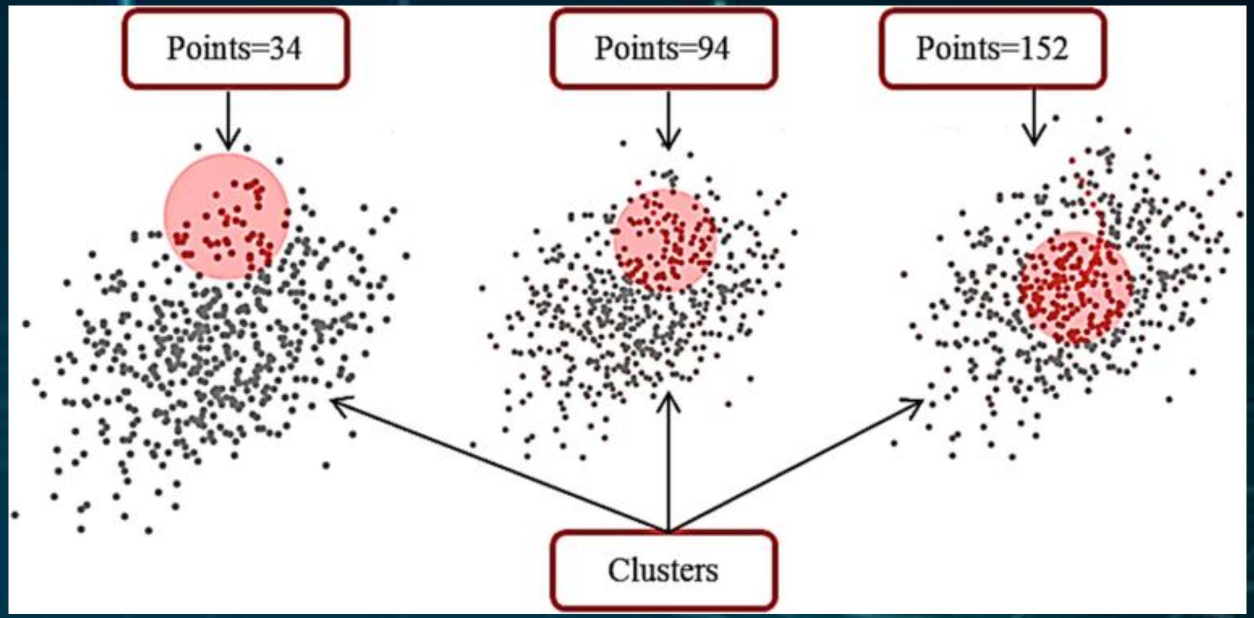
Distance-based clustering methods will be deployed in the project:

1. K-Means
2. MiniBatch K-Means
3. Hierarchical
4. GMM - Gaussian Mixture Model
5. Spectral

---

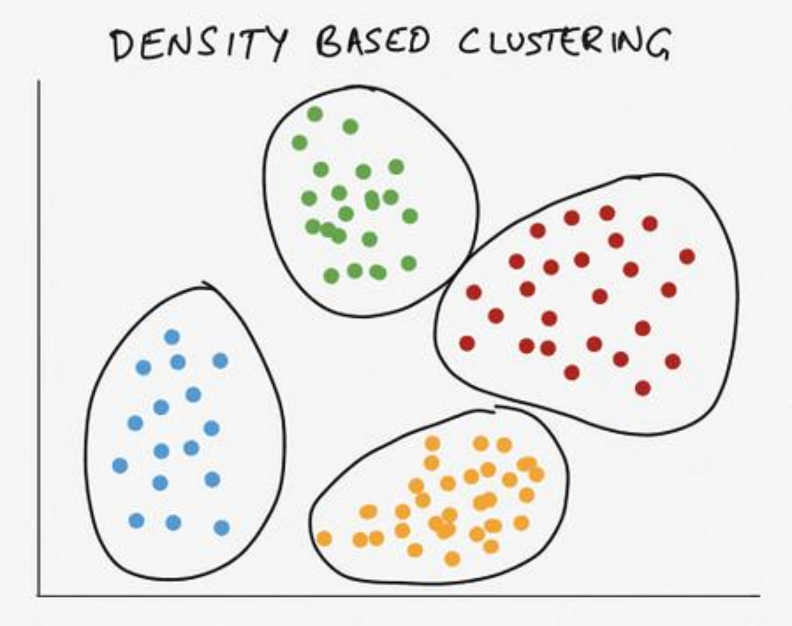
6. DBSCAN
7. HDBSCAN

Centroid-based



https://www.researchgate.net/figure/Centroid-based-clustering-algorithm_fig6_334279038

Density-based



https://www.graduatetutor.com/statistics-tutor/k-means-clustering-hierarchical-clustering-density-based-clustering-partitional-clustering/

Standard Scaling Features

1. K-Means
2. MiniBatch K-Means
3. Hierarchical
4. GMM - Gaussian Mixture Model
5. Spectral
6. DBSCAN
7. HDBSCAN

How Scaling Effects Distance based algorithms

Medium

Kushvanth Follow 2 min read · Aug 14, 2025

- **Distance Dependence:** These algorithms rely heavily on calculating distances between data points to determine similarity or make classifications.
- **Feature Dominance:** Without scaling, features with larger magnitudes or ranges can disproportionately influence distance calculations, making them seem more important than features with smaller ranges, even if the smaller-ranged features hold more predictive power. For instance, if you have features like age (0–100) and income (thousands of dollars), income's larger numerical values will dominate the distance calculations in unscaled data.
- **Skewed Results:** This dominance can lead to biased or inefficient learning, resulting in suboptimal performance and reduced accuracy.
- **Ensuring Equal Contribution:** Feature scaling techniques (like Min-Max scaling or standardization) transform features to a common scale, ensuring that each feature contributes equally to the distance calculations, regardless of its original magnitude

Standard Scaling Features

```python
# Log Transformed Columns
log_cols = ['balance_log', 'purchases_log', 'oneoff_purchases_log', 'installments_purchases_log',
            'cash_advance_log', 'cash_advance_trx_log', 'purchases_trx_log',
            'credit_limit_log', 'payments_log', 'minimum_payments_log', 'prc_full_payment_log']

# Columns of frequency and tenure (using original values)
common_cols = ['balance_frequency', 'purchases_frequency', 'oneoff_purchases_frequency',
               'purchases_installments_frequency', 'cash_advance_frequency',
               'prc_full_payment', 'tenure']

group_cols = log_cols + common_cols
```

Standard Scaling Features

```python
# Log Transformed Columns
log_cols = ['balance_log', 'purchases_log', 'oneoff_purchases_log', 'installments_purchases_log',
            'cash_advance_log', 'cash_advance_trx_log', 'purchases_trx_log',
            'credit_limit_log', 'payments_log', 'minimum_payments_log', 'prc_full_payment_log']

# Columns of frequency and tenure (using original values)
common_cols = ['balance_frequency', 'purchases_frequency', 'oneoff_purchases_frequency',
               'purchases_installments_frequency', 'cash_advance_frequency',
               'prc_full_payment', 'tenure']

group_cols = log_cols + common_cols
```

```python
scaler = StandardScaler()

df_group = df[group_cols]

# Fit and Transform
df_group_scaled = scaler.fit_transform(df_group)

df_group_scaled = pd.DataFrame(df_group_scaled, columns=group_cols, index=df.index)
```

Step 1: Run Centroid-based Clustering models

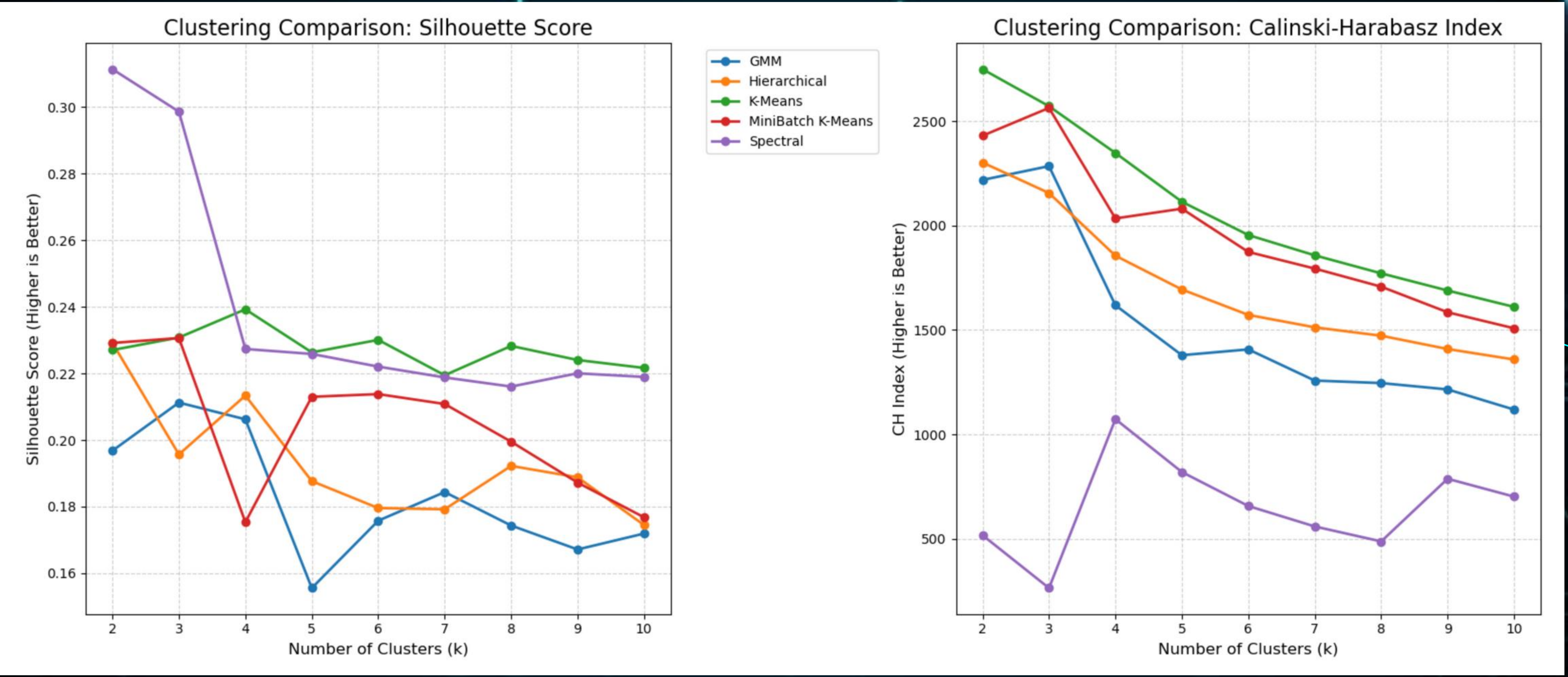
Choose the optimal one (1)

Step 2: Run Density-based Clustering models

Compare to best-centroid-based model

Step 3: Profiling Clusters and
Providing Targeted marking/loyalty compaigns for each

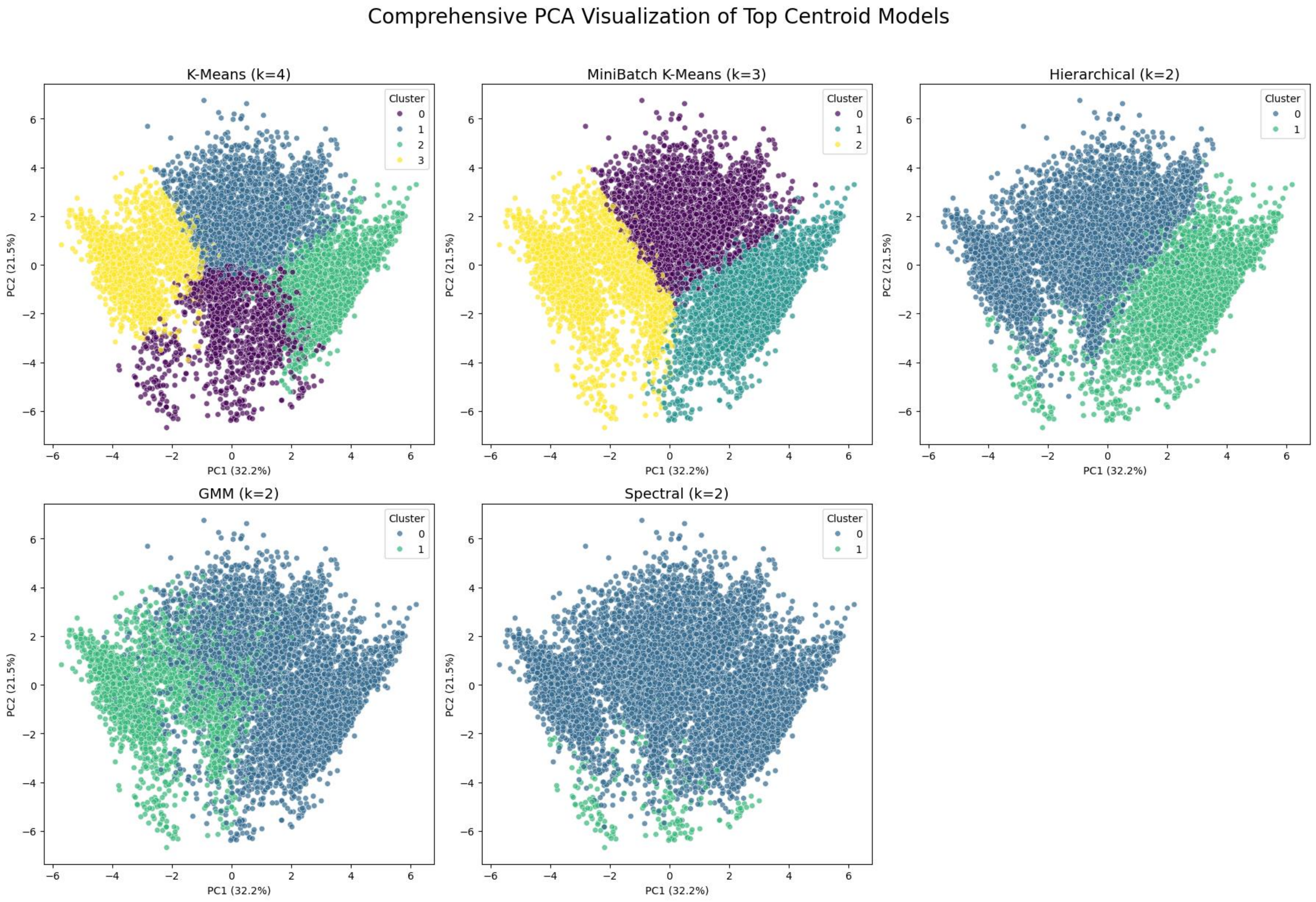
Centroid-based Clustering Models: Silhouette and Calinski-Harabasz Index



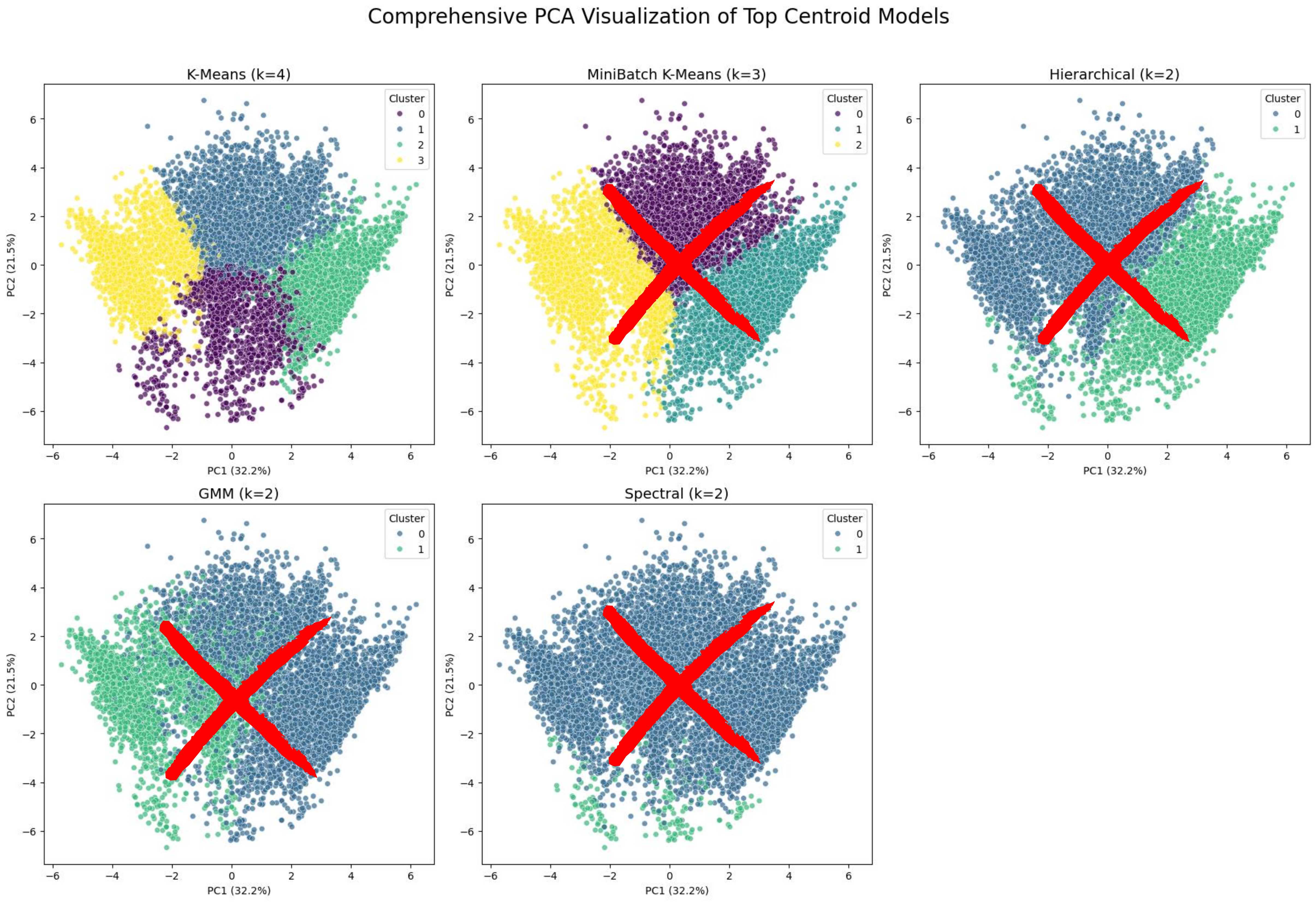
Centroid-based Clustering Models: Visualization with the Optiomal k

Model	Optimal k Chosen	Reason
K-Means	4	Local Maxima on Silhouette Score.
MiniBatch K-Means	3	Peak on Calinski-Harabasz Index.
Hierarchical	2	Highest score for both indices.
GMM	2	Highest score for both indices.
Spectral	2	Highest score for both indices.

Centroid-based Clustering Models: Visualization with the Optiomal k

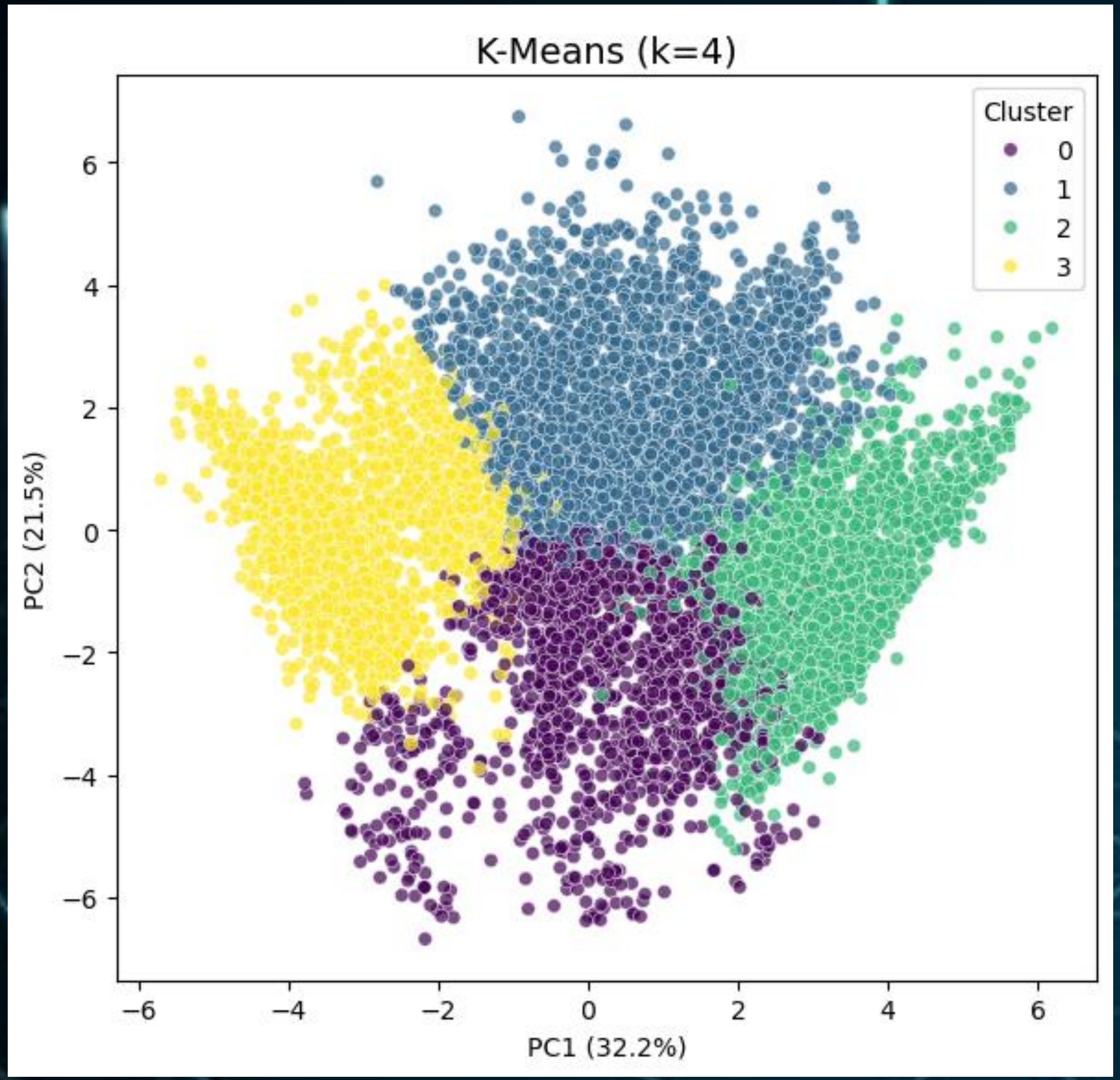


Centroid-based Clustering Models: Visualization with the Optiomal k



- **Spectral** (highest Silhouette Score at k=2, its PCA visualization reveals a chaotic and unstructured separation. The blue and green data points appear almost randomly interwoven.
- **Hierarchical** (k=2): Creates a very clear separation along the PC1 axis (left vs. right). This is the most fundamental split, based on the attribute with the highest variance.
- **GMM** (k=2): Separates the data along a more subtle, diagonal axis. This indicates that the GMM is attempting to find elliptical or density-based clusters rather than enforcing a rigid, variance-based split.

Centroid-based Clustering Models: choosing K-Means as the best method

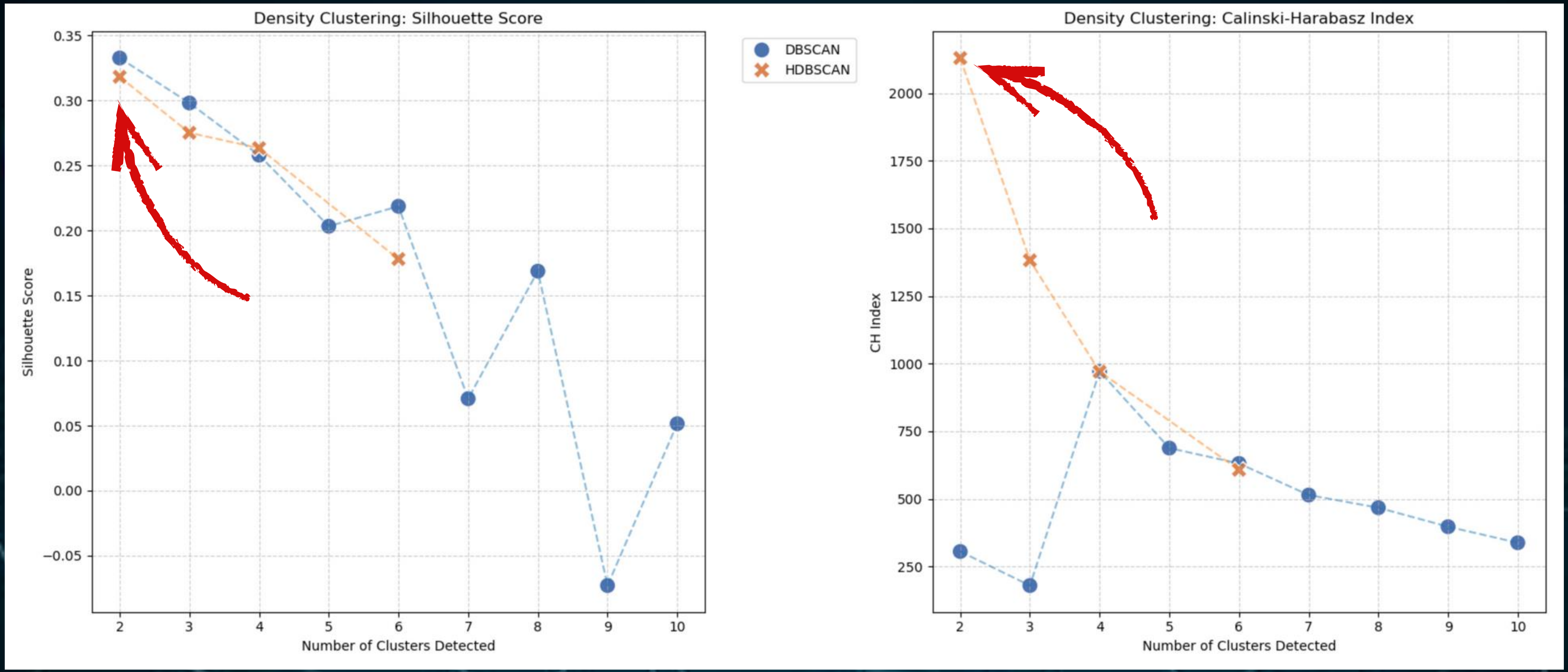


K-Means is the only model (among those with k > 2) that clearly exhibits a four-quadrant structure:

- PC1 splits the data horizontally (Left/Right).
- PC2 splits the data vertically (Up/Down).

→ Business Insight: choosing k=4 has a geometric basis. It is not an arbitrary split but a natural division along the two primary axes of data variation. This aligns perfectly with the business logic of segmenting into four groups: High-High, High-Low, Low-High, and Low-Low.

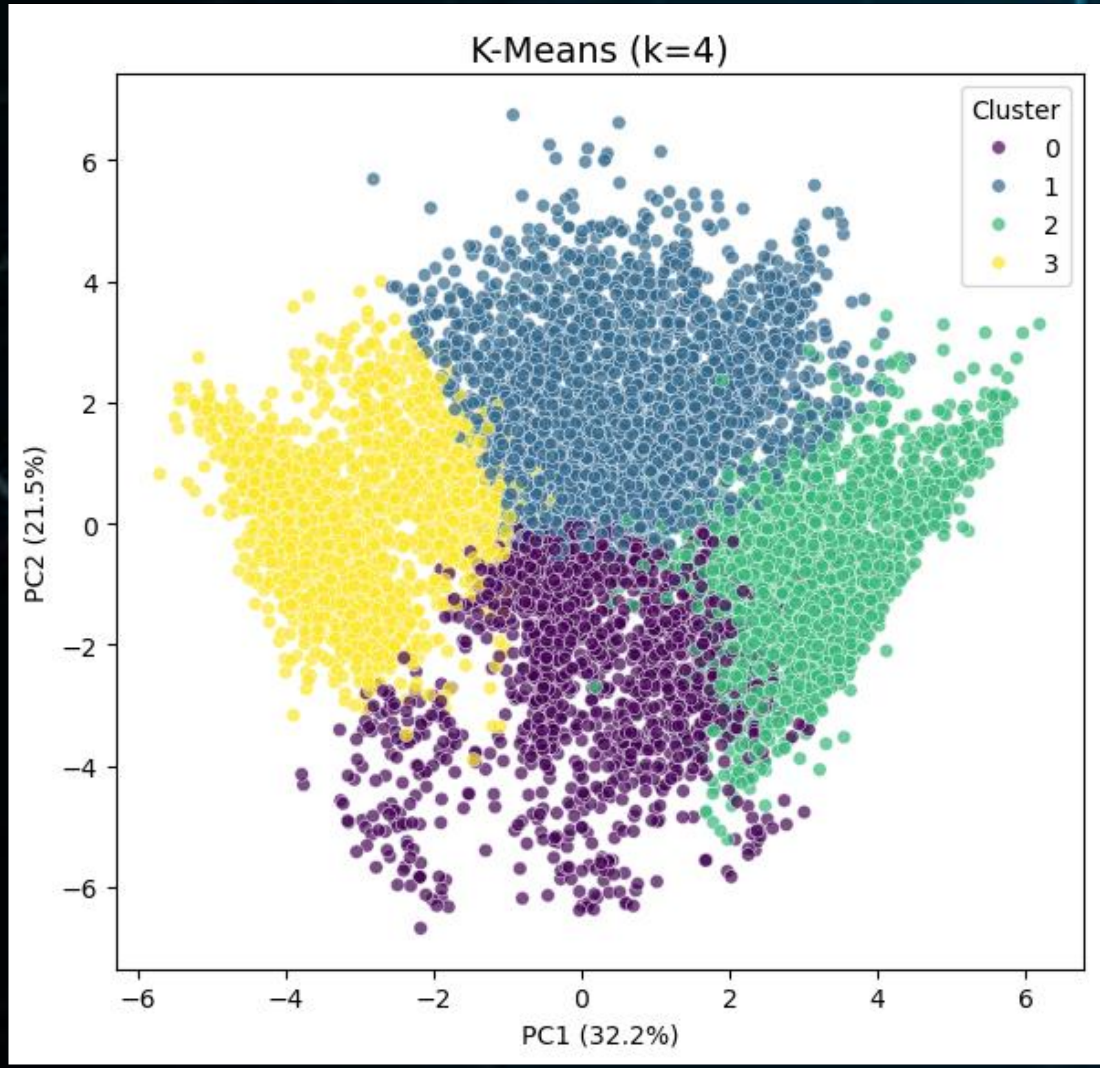
Density-based Clustering Models: Silhouette and Calinski-Harabasz Index

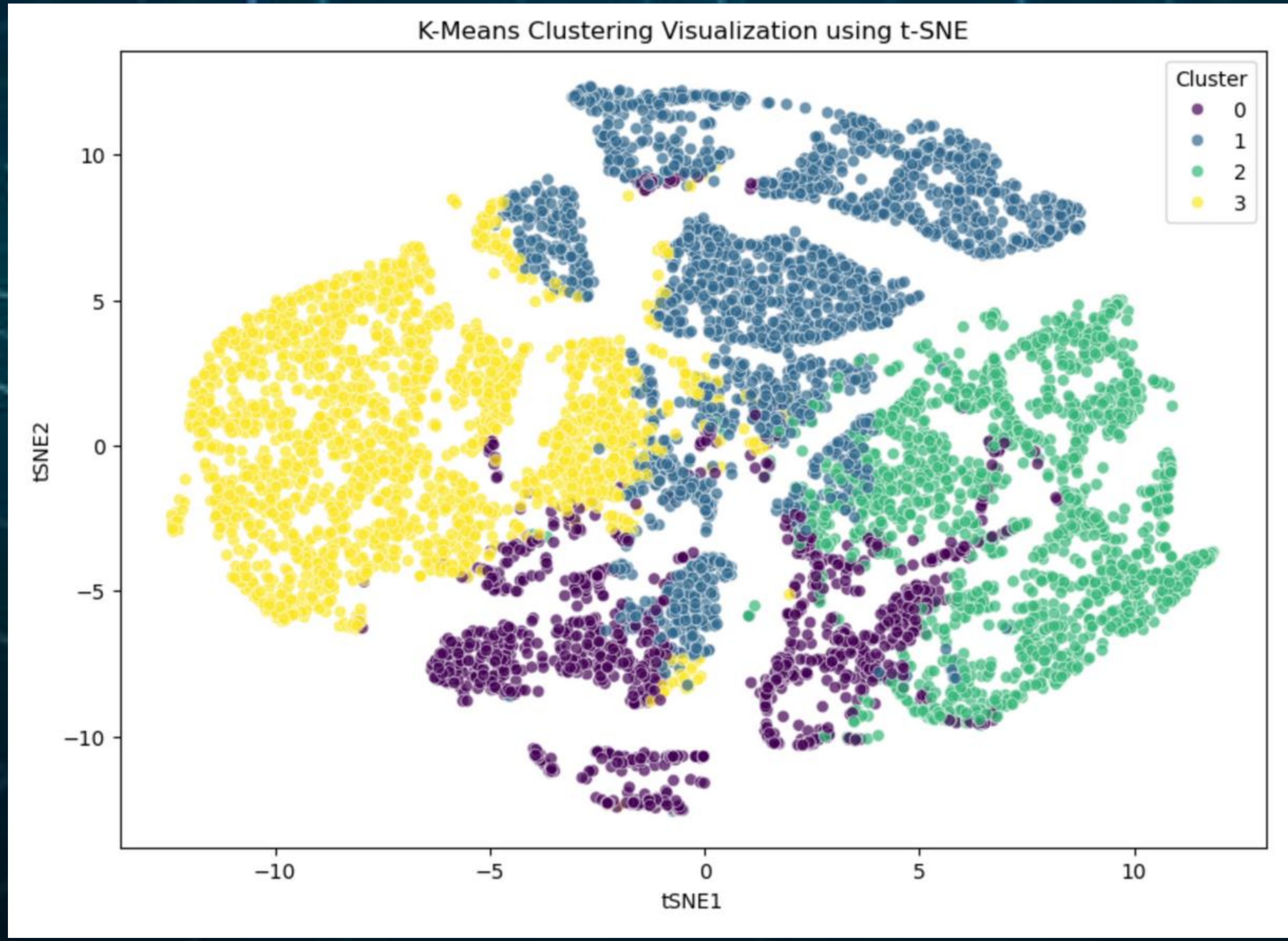


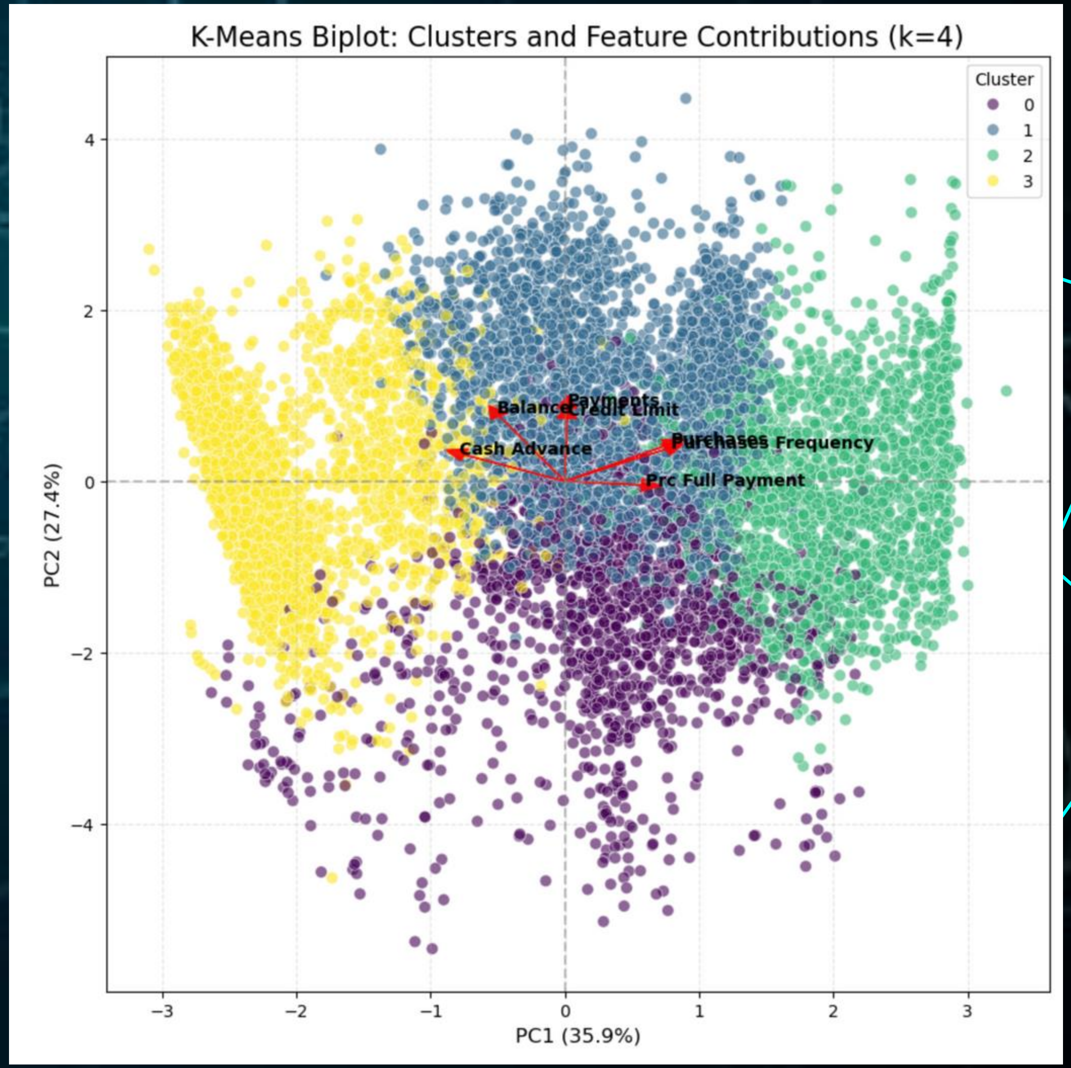
- **Silhouette Score**: k=2 with highest scores. These algorithms focus on local density and remove noise (-1), they discover two or three extremely cohesive and well-separated clusters.
- **Calinski-Harabasz Index**: HDBSCAN with highest score at k=2. Both models show a rapid declining trend, and their scores at larger k values are very low.
- The performance of two models is very volatile, especially beyond k=4. The Silhouette Score drops sharply and even becomes negative at k=9 for DBSCAN, indicating a complete breakdown of the cluster structure.

Best Clustering Model

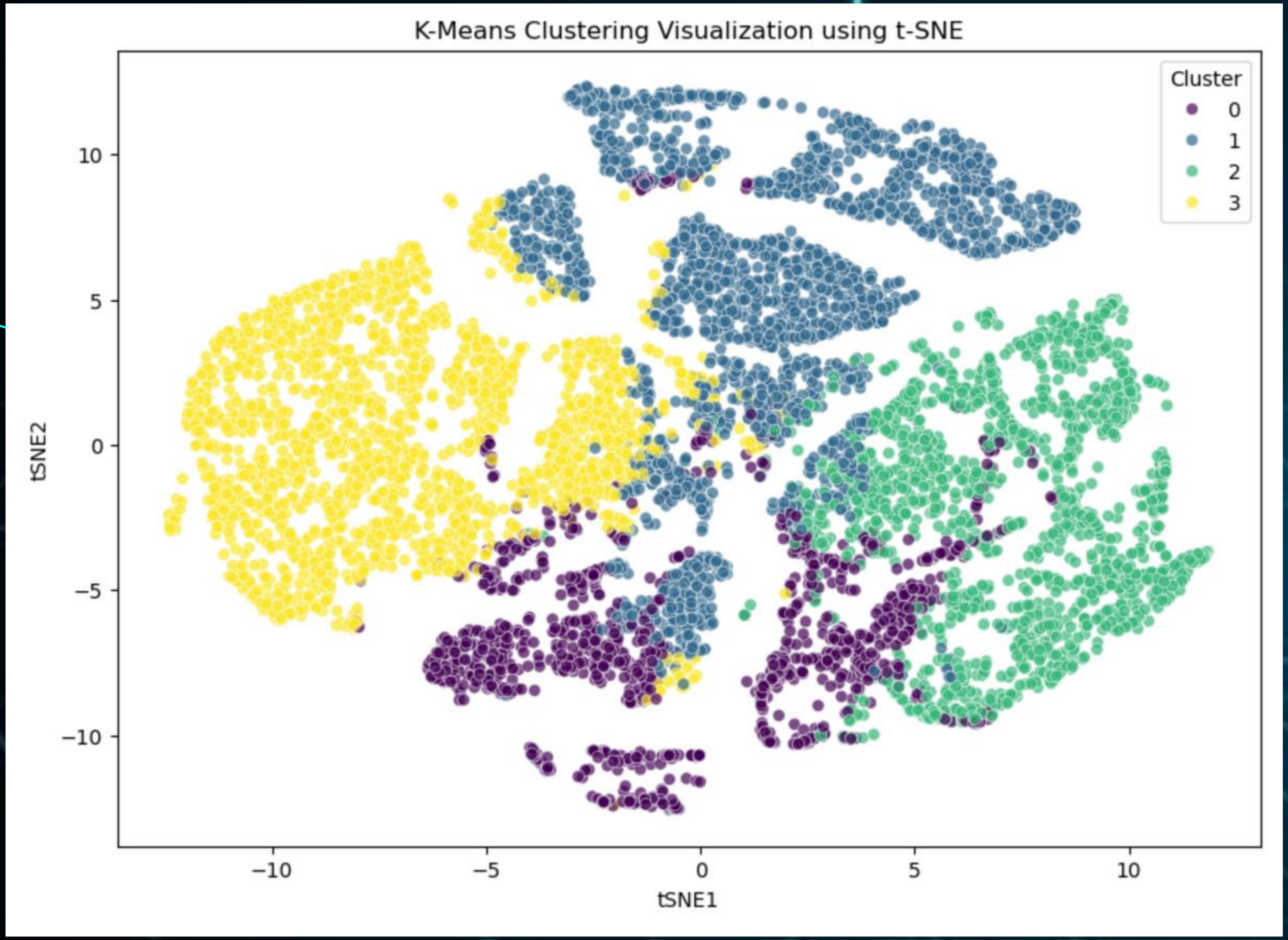
- Based on the balance between technical performance and feasibility in credit business operations, we choose **K-Means with k=4**.
- **Technical Reason:** Although Density-based models achieved higher Silhouette Scores, K-Means excels in CH Index, demonstrating better overall dispersion and separation, while being more stable and easier to interpret.
- **Business Reason:** In the Credit sector, we need to know who the "Noise" customers are (as they could represent either risks or potential opportunities). K-Means assigns labels to everyone, providing 4 actionable segments that you require.







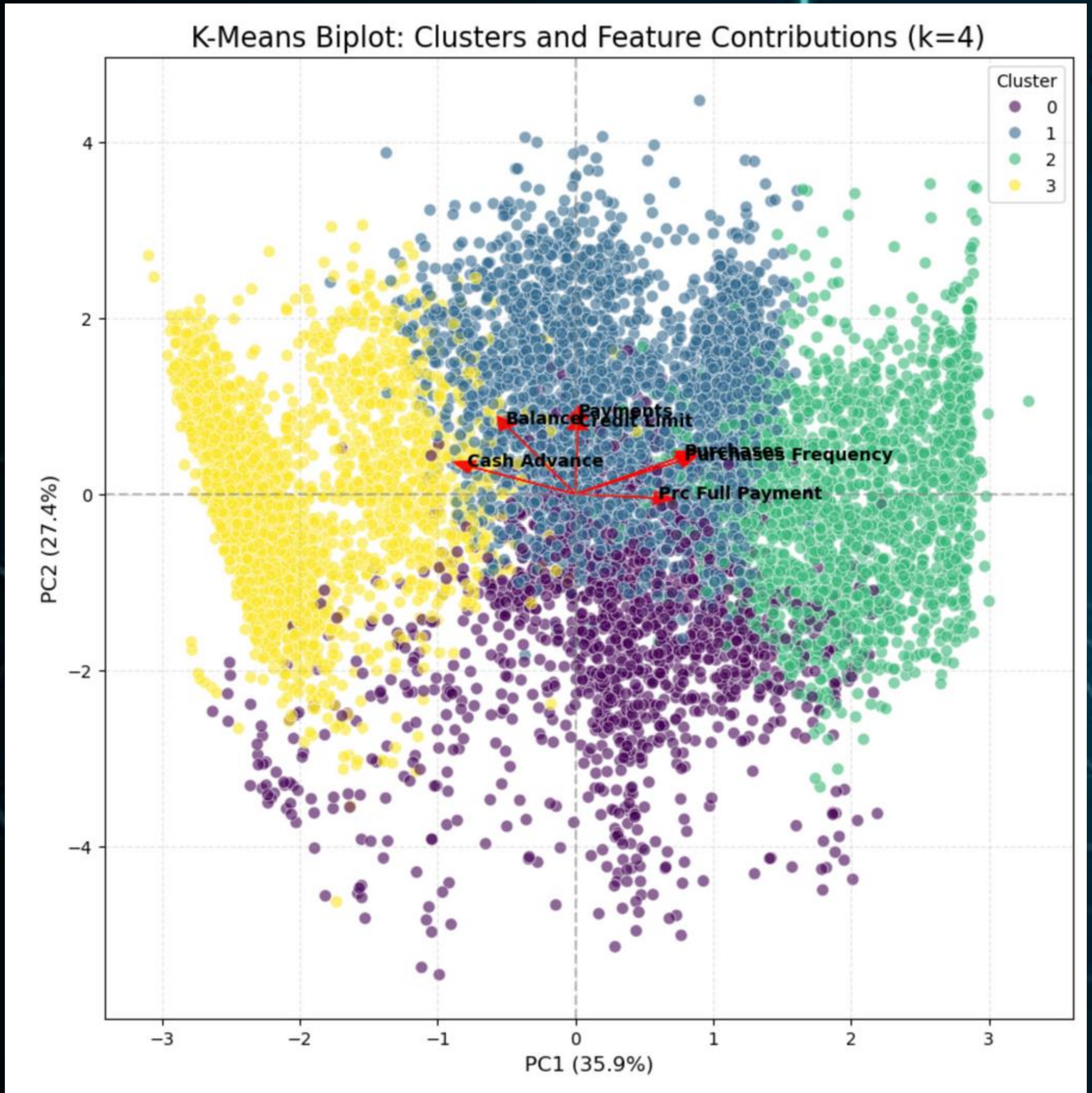
K-Means Cluster Visual Insights: t-SNE



- Cluster 3 (**Yellow**) and Cluster 1 (**Blue**) form two large, almost opposing blocks.
- Cluster 0 (**Purple**) and Cluster 2 (**Green**) are located close to each other and have more intermingled points.

→ Conclusion: This overlap reinforces the point of the low Silhouette Score (k=4, ≈0.24), indicating these two groups have the most similar characteristics and are the most difficult to distinguish. This is an area where the business team needs to pay special attention when designing campaigns.

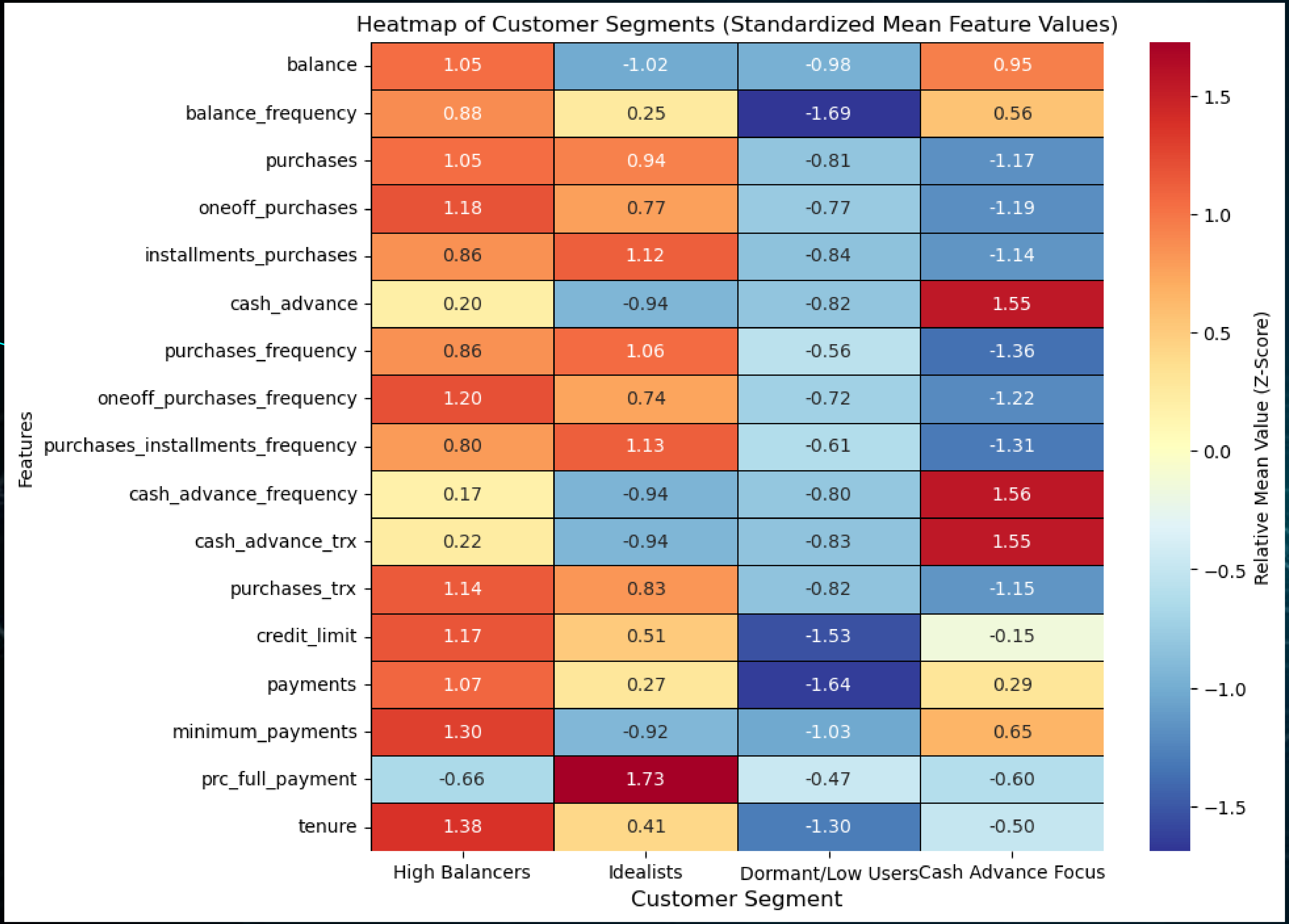
K-Means Cluster Visual Insights: Biplot



- Cluster 0 (Purple) and Cluster 2 (Green) have significant overlap, explaining the low Silhouette Score and requiring attention for campaign deployment.
- **Two main opposing customer groups:**
  - **Risk Group (Cluster 3 - Yellow):** Characterized by high Cash_Advance and Balance. This group uses cash frequently and is considered higher risk.
  - **Ideal Group (Cluster 2 - Green):** Characterized by high Purchases_Frequency and Pre_Full_Payment. This group shops frequently and has responsible payment habits.
- A clear negative correlation exists between Cash_Advance and Pre_Full_Payment. Customers who frequently use cash advances tend to rarely make full payments, and vice versa.

Introduction EDA Preprocessing **Unsupervised Learning** Supervised Learning

Profiling



Profiling

Mean Value Heatmap (Original Values, Row-wise Scaling)

Features	High Balancers	Idealists	Dormant/Low Users	Cash Advance Focus
balance	2,481.11	214.32	256.13	2,380.50
balance_frequency	0.986	0.886	0.578	0.934
purchases	1,766.47	1,681.94	352.89	74.07
oneoff_purchases	1,071.07	897.13	241.15	65.87
installments_purchases	695.59	785.19	112.55	8.25
cash_advance	1,006.30	34.99	139.80	2,158.85
purchases_frequency	0.765	0.831	0.309	0.052
oneoff_purchases_frequency	0.352	0.292	0.101	0.036
purchases_installments_frequency	0.576	0.664	0.201	0.014
cash_advance_frequency	0.136	0.007	0.024	0.297
cash_advance_trx	3.39	0.121	0.421	7.13
purchases_trx	27.21	23.65	4.59	0.795
credit_limit	5,217.22	4,740.86	3,278.25	4,271.29
payments	2,270.61	1,783.30	619.43	1,792.97
minimum_payments	1,431.10	217.82	160.34	1,076.65
prc_full_payment	0.019	0.582	0.063	0.033
tenure	11.77	11.57	11.22	11.38

Customer Segment

Mean Value Heatmap (Original Values, Row-wise Scaling)

Features	High Balancers	Idealists	Dormant/Low Users	Cash Advance Focus
balance	2,481.11	214.32	256.13	2,380.50
balance_frequency	0.986	0.886	0.578	0.934
purchases	1,766.47	1,681.94	352.89	74.07
oneoff_purchases	1,071.07	897.13	241.15	65.87
installments_purchases	695.59	785.19	112.55	8.25
cash_advance	1,006.30	34.99	139.80	2,158.85
purchases_frequency	0.765	0.831	0.309	0.052
oneoff_purchases_frequency	0.352	0.292	0.101	0.036
purchases_installments_frequency	0.576	0.664	0.201	0.014
cash_advance_frequency	0.136	0.007	0.024	0.297
cash_advance_trx	3.39	0.121	0.421	7.13
purchases_trx	27.21	23.65	4.59	0.795
credit_limit	5,217.22	4,740.86	3,278.25	4,271.29
payments	2,270.61	1,783.30	619.43	1,792.97
minimum_payments	1,431.10	217.82	160.34	1,076.65
prc_full_payment	0.019	0.582	0.063	0.033
tenure	11.77	11.57	11.22	11.38

Customer Segment

Profiling

Based on the behavioral analysis:

- Cluster 1: **HIGH BALANCERS** (High Spending + High Interest Debt)
- Cluster 2: **IDEALISTS** (Good Payment + Good Spending)
- Cluster 0: **DORMANT** (Inactive/Low Usage)
- Cluster 3: **CASH ADVANCE FOCUS** (Cash Withdrawal + Low Purchases)

Business Interpretation: Profiling Analysis and Recommended Actions

Final Customer Segmentation Profile

Cluster ID	Segment Name	Proportion	Key Characteristics	Risk & Profitability Profile
2	Idealists	~22%	High full payment rate, Highest purchase frequency, Healthy usage	Sustainable Profit: Lowest risk, healthy spending behavior
1	High Balancers	~31%	Highest average balance ($1,648), High purchases	Maximum Profit: High interest revenue but highest credit risk
3	Cash Advance Focus	~29%	Highest cash advances ($2,159), Very low purchases	Urgent Risk: Liquidity stress signals, highest default risk
0	Dormant/Low Users	~18%	Lowest balance and purchases	Churn Risk: Low profitability, high competitive vulnerability

Strategic Recommendations & Campaigns

Segment	Strategic Objective	Recommended Campaigns
2: Idealists	Retention & Spending Growth: Increase LTV through brand advocacy	**Proactive Upgrades**: Platinum/Signature cards (no/low annual fees), higher limits, exclusive benefits
1: High Balancers	Profit Maximization & Debt Stabilization: Maintain interest revenue while controlling debt	**Loan Product Cross-sell**: Balance transfer offers, personal loans to reduce interest burden and stabilize debt
3: Cash Advance Focus	Risk Mitigation & Behavior Change: Shift from cash advances to purchases	**Risk Management**: Close DPD monitoring, cash advance limits, cashback/discount offers for first purchases
0: Dormant/Low Users	Reactivation & Re-engagement: Bring customers back to card usage	**"Re-use" Campaign**: Low-barrier offers like cashback after 3 small transactions, fee waivers

feature	description
custid	Unique identification of the credit cardholder (categorical)
balance	Balance amount left in the account for purchases
balancefrequency	Frequency of balance updates (0 = rarely, 1 = frequently)
purchases	Total amount spent on purchases
oneoffpurchases	Maximum single transaction purchase amount
installmentspurchases	Purchases made in installments
cash_advance	Cash advances taken by the user
purchasesfrequency	Frequency of purchases (0 = rarely, 1 = frequently)
oneoffpurchasesfrequency	Frequency of one-time purchases (0 = rarely, 1 = frequently)
purchasesinstallmentsfrequency	Frequency of installment purchases (0 = rarely, 1 = frequently)
cashadvancefrequency	Frequency of cash advances taken
cashadvancetrx	Number of cash advance transactions
purchasestrx	Number of purchase transactions
credit_limit	User's credit card limit
payments	Amount of payments made by the user
minimum_payments	Minimum payment amount made by the user
prcfullpayment	Percentage of the full payment made
tenure	Duration of credit card usage (in months)

High-Risk Customers Prediction

Feature Engineering

Target Variable

Relevant Features

Feature Engineering

Target Variable

Relevant Features

High-Risk Customer Identification Logic:

- Condition 1: **minimum_payments > payments**: Customers are paying only the minimum required amount instead of the actual payment due. This behavior indicates potential financial distress, as customers who can only afford minimum payments may be struggling to manage their debt obligations and are at higher risk of default.
- Condition 2: **balance > credit_limit * 0.8**: Customers are utilizing over 80% of their available credit limit. This creates high over-limit risk, as customers approaching their credit ceiling have limited financial flexibility and are more likely to exceed their credit limits, indicating potential cash flow problems.
- Condition 3: **High cash advance frequency** (for example, >=50% of the time) is an important risk indicator in credit. This behavior often signals customer liquidity problems, where the credit card is used as a short-term loan, serving as a strong warning sign of potential impending financial distress.

Feature Engineering

Target Variable

```
df_sl['is_high_risk'] = (
    (df_sl['minimum_payments'] > df_sl['payments']) |
    (df_sl['payments'] <= 1.0) |
    (df_sl['balance'] > df_sl['credit_limit'] * 0.8)
).astype(int)
```

```
TARGET VARIABLE DISTRIBUTION:
is_high_risk
0    5707
1    3243
Name: count, dtype: int64
Total samples: 8950
High Risk ratio (1): 36.23%
```

```
y = df_sl['is_high_risk']
```

Relevant Features

Feature Engineering

Target Variable

```
y = df_sl['is_high_risk']
```

Relevant Features

Feature Engineering

Target Variable

Relevant Features

```
X = df_sl[[
    'prc_full_payment',
    'cash_advance_frequency',
    'purchases_frequency',
    'tenure',
    'cash_advance_trx',
    'purchases_trx']]
```

Feature Engineering

Target Variable

```
y = df_sl['is_high_risk']
```

Relevant Features

```
X = df_sl[[
    'prc_full_payment',
    'cash_advance_frequency',
    'purchases_frequency',
    'tenure',
    'cash_advance_trx',
    'purchases_trx']]
```

Correlation with y

cash_advance_frequency	0.192777
cash_advance_trx	0.113223
tenure	-0.073761
purchases_trx	-0.144155
purchases_frequency	-0.236698
prc_full_payment	-0.363532

Feature Engineering

Target Variable

```
y = df_sl['is_high_risk']
```

Relevant Features

```
X = df_sl[[
    'prc_full_payment',
    'cash_advance_frequency',
    'purchases_frequency',
    'tenure',
    'cash_advance_trx',
    'purchases_trx']]
```

Correlation with y

cash_advance_frequency	0.192777
cash_advance_trx	0.113223
tenure	-0.073761
purchases_trx	-0.144155
purchases_frequency	-0.236698
prc_full_payment	-0.363532

All features' correlation coefficients: -0.05 < and < 0.05

Step to Run Models and Compare Results

XGBoost - Extreme Gradient Boosting

↓

Random Forest

↓

Logistic Regression

↓

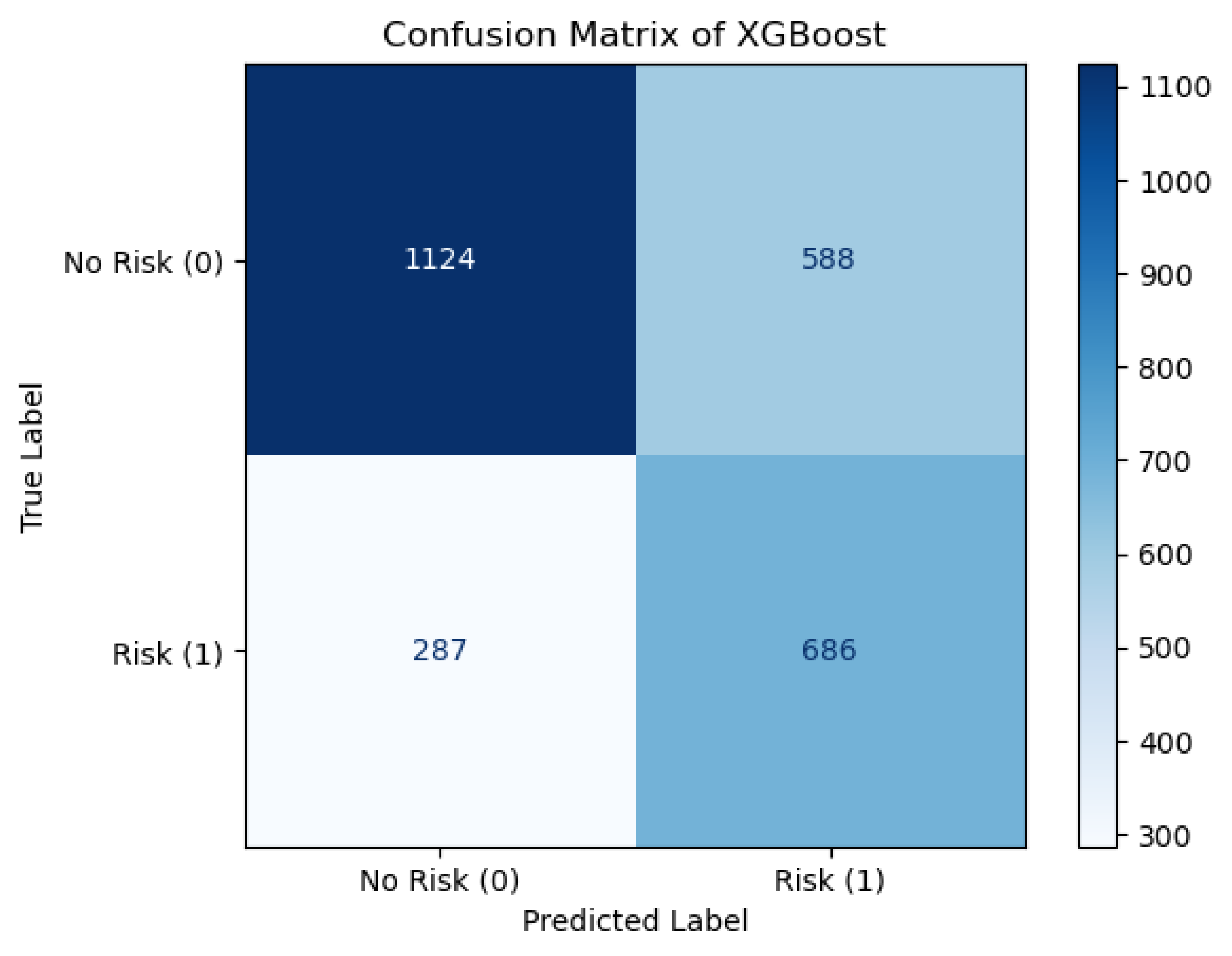
SVM - Support Vector Machine

Model: XGBoost - Extreme Gradient Boosting

Metric	Value	Business Meaning
ROC AUC Score	0.7787	Strongest indicator. Shows the model has a 77.87% ability to correctly distinguish a risky customer from a randomly selected non-risky one. This is the most reliable metric.
Recall (R) for Risk (Class 1)	0.85	Very High. The model successfully found 85% of all truly risky customers. This significantly helps the bank minimize the most dangerous type of error (missing a customer about to default - False Negative).
Precision (P) for Risk (Class 1)	0.54	Weak. When the model predicts "Risk," it is correct only 54% of the time. This means 46% (718 samples) are false alarms (False Positives).

Model: XGBoost - Extreme Gradient Boosting

Metric	Value	Business Meaning
ROC AUC Score	0.7787	Strongest indicator. Shows the model has a 77.87% ability to correctly distinguish a risky customer from a randomly selected non-risky one. This is the most reliable metric.
Recall (R) for Risk (Class 1)	0.85	Very High. The model successfully found 85% of all truly risky customers. This significantly helps the bank minimize the most dangerous type of error (missing a customer about to default - False Negative).
Precision (P) for Risk (Class 1)	0.54	Weak. When the model predicts "Risk," it is correct only 54% of the time. This means 46% (718 samples) are false alarms (False Positives).

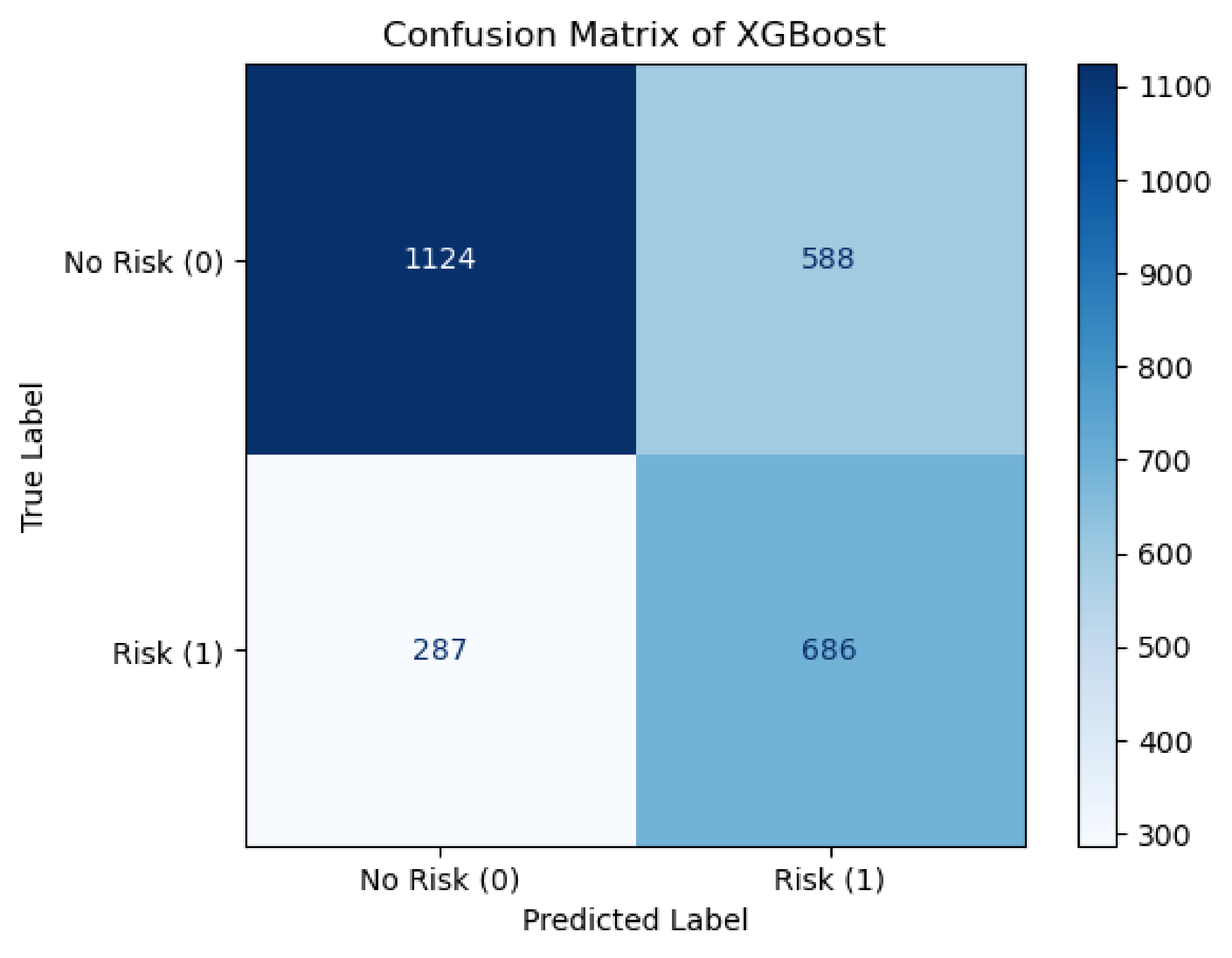


Cost/Benefit Analysis (Error Costs)

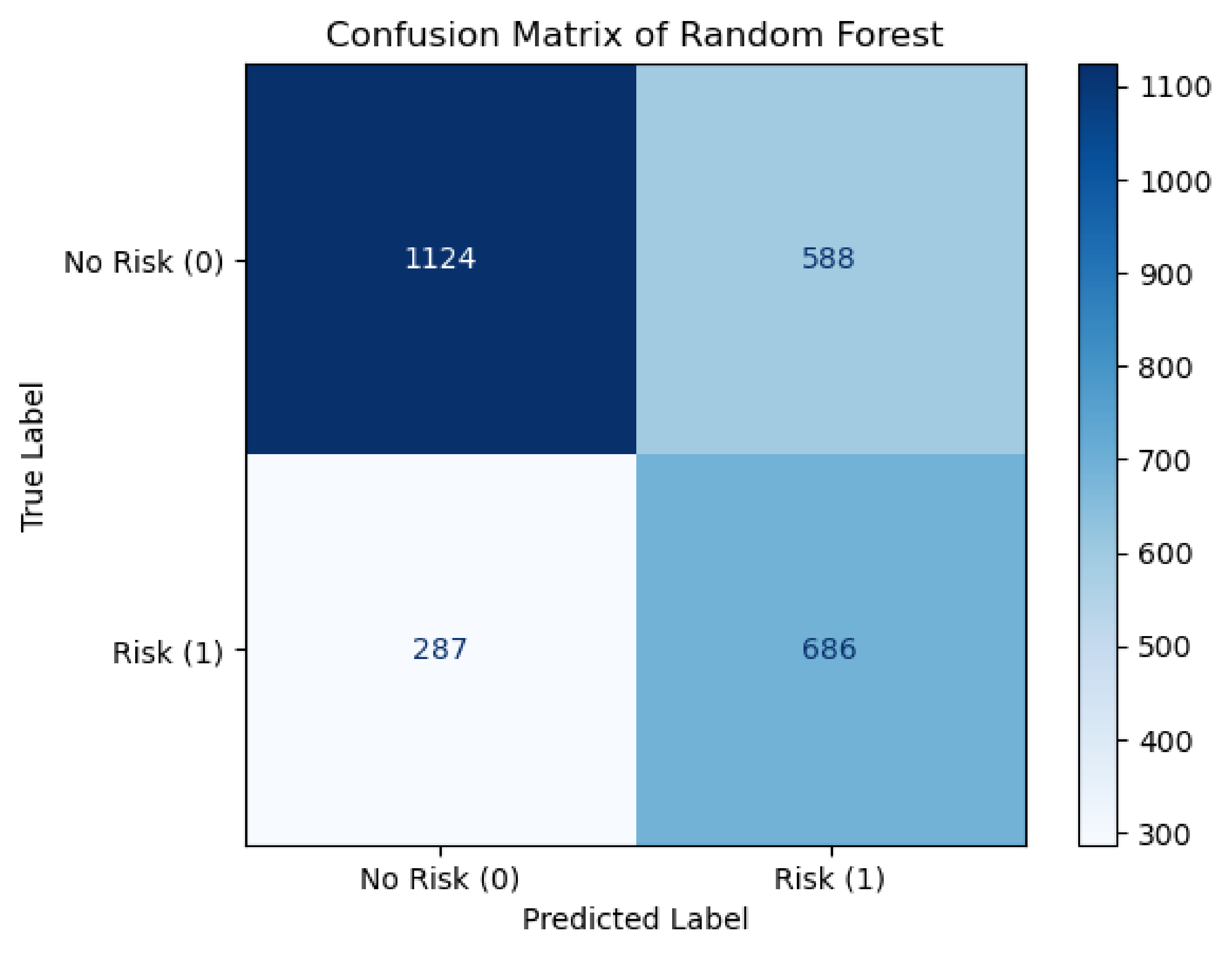
- TN (Safe Found): 994 → Success.
- TP (Risks Found): 831 → Success.
- FN (Risks Missed): 142 → Dangerous error. Thanks to scale_pos_weight, this number was successfully minimized to only 142.
- FP (False Alarms): 718 → Resource drain. This is the cost the customer service or debt collection department will incur by contacting 718 non-risky customers.

Conclusion: The model has been successfully optimized to maximize risk detection (Recall 85%)—fulfilling the primary objective of this credit risk task.

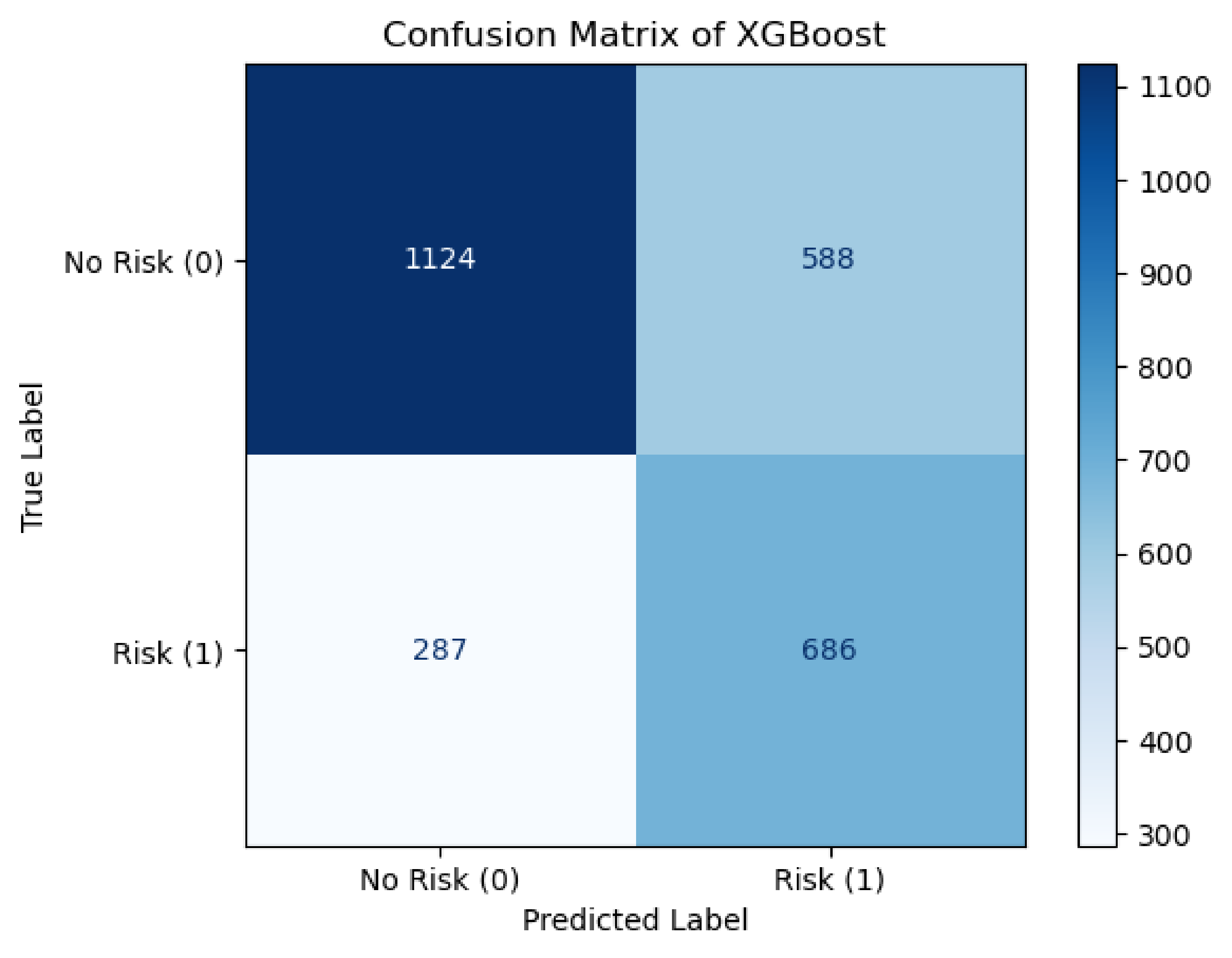
Model: RF - Random Forest



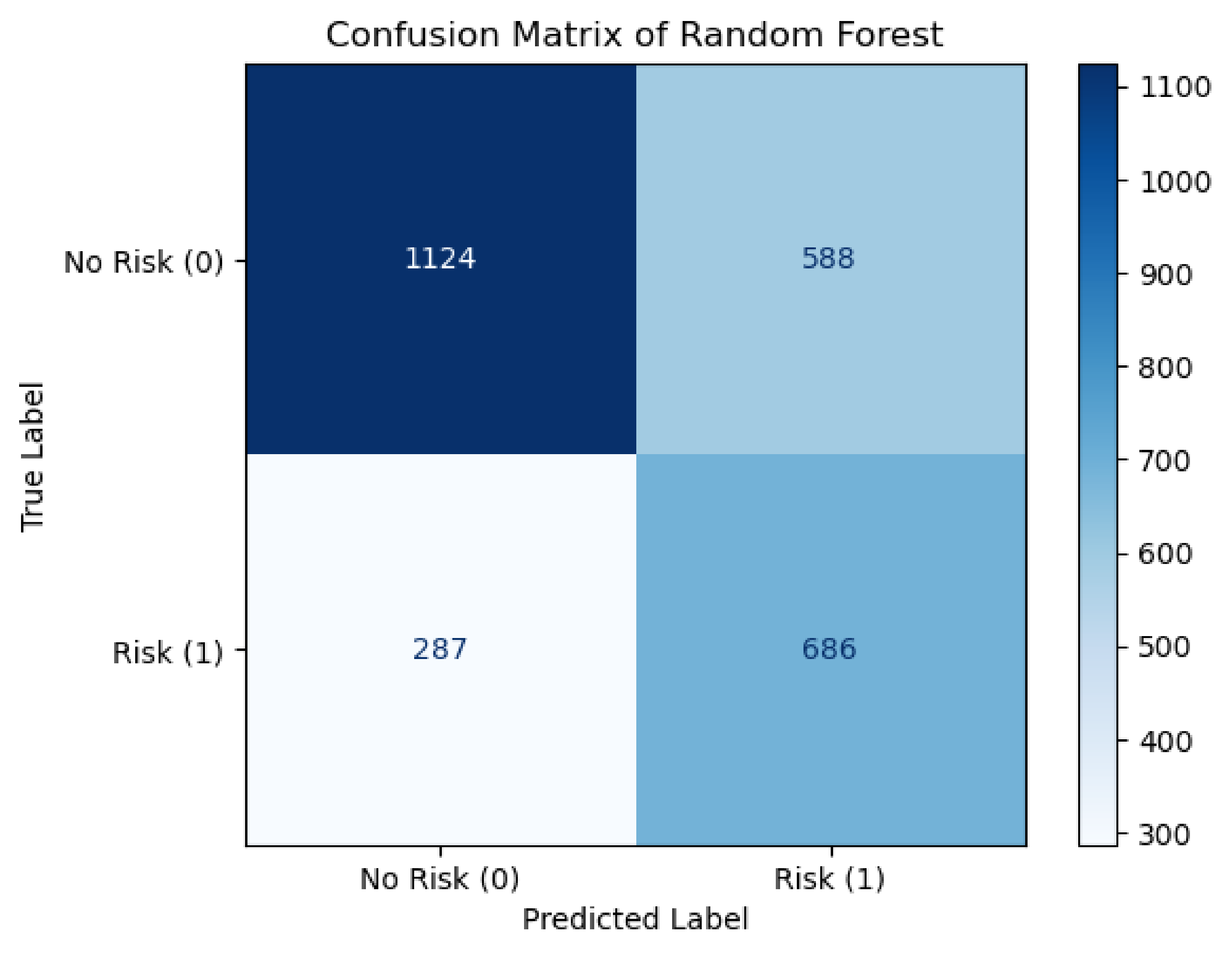
vs.



Model: RF - Random Forest

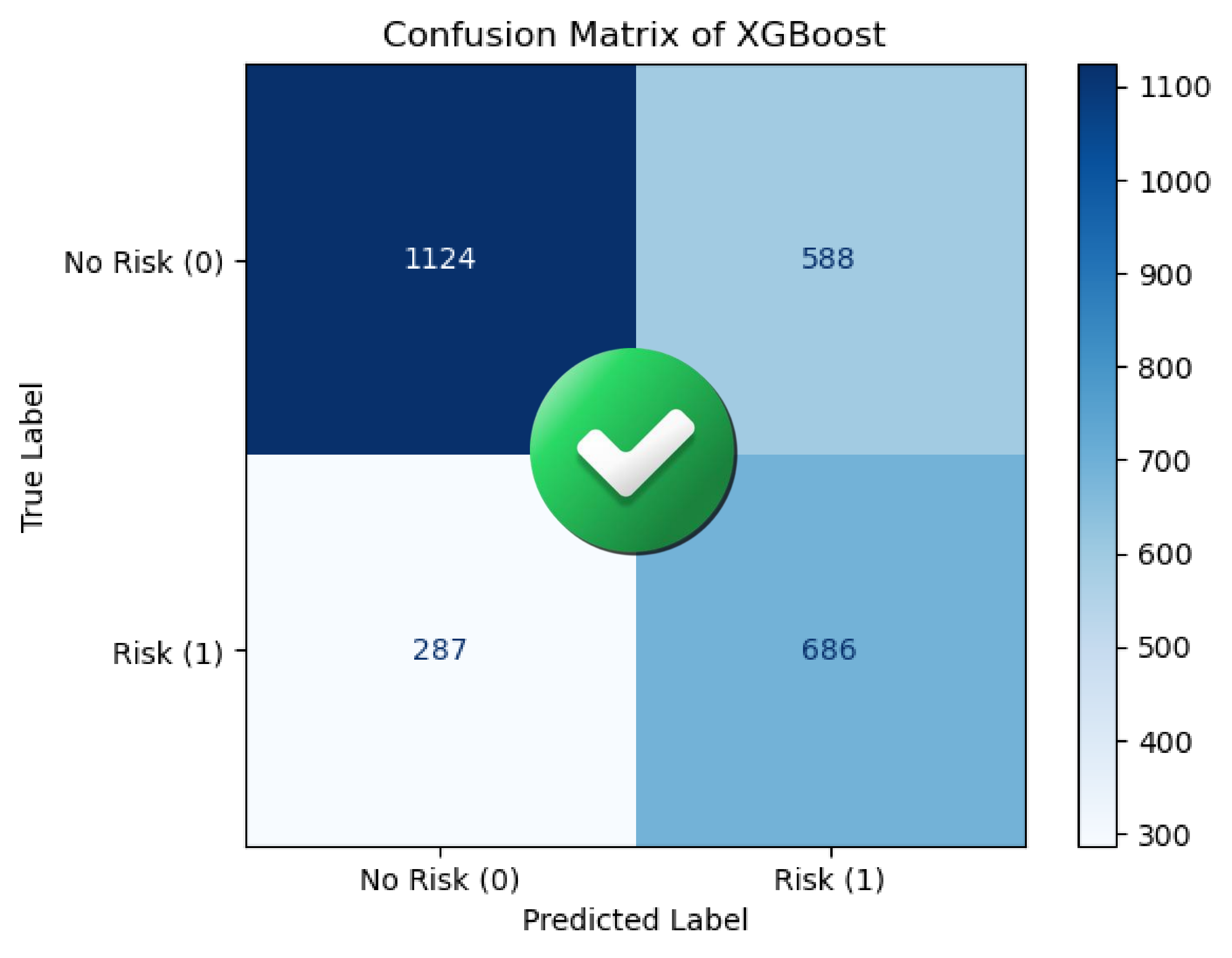


vs.

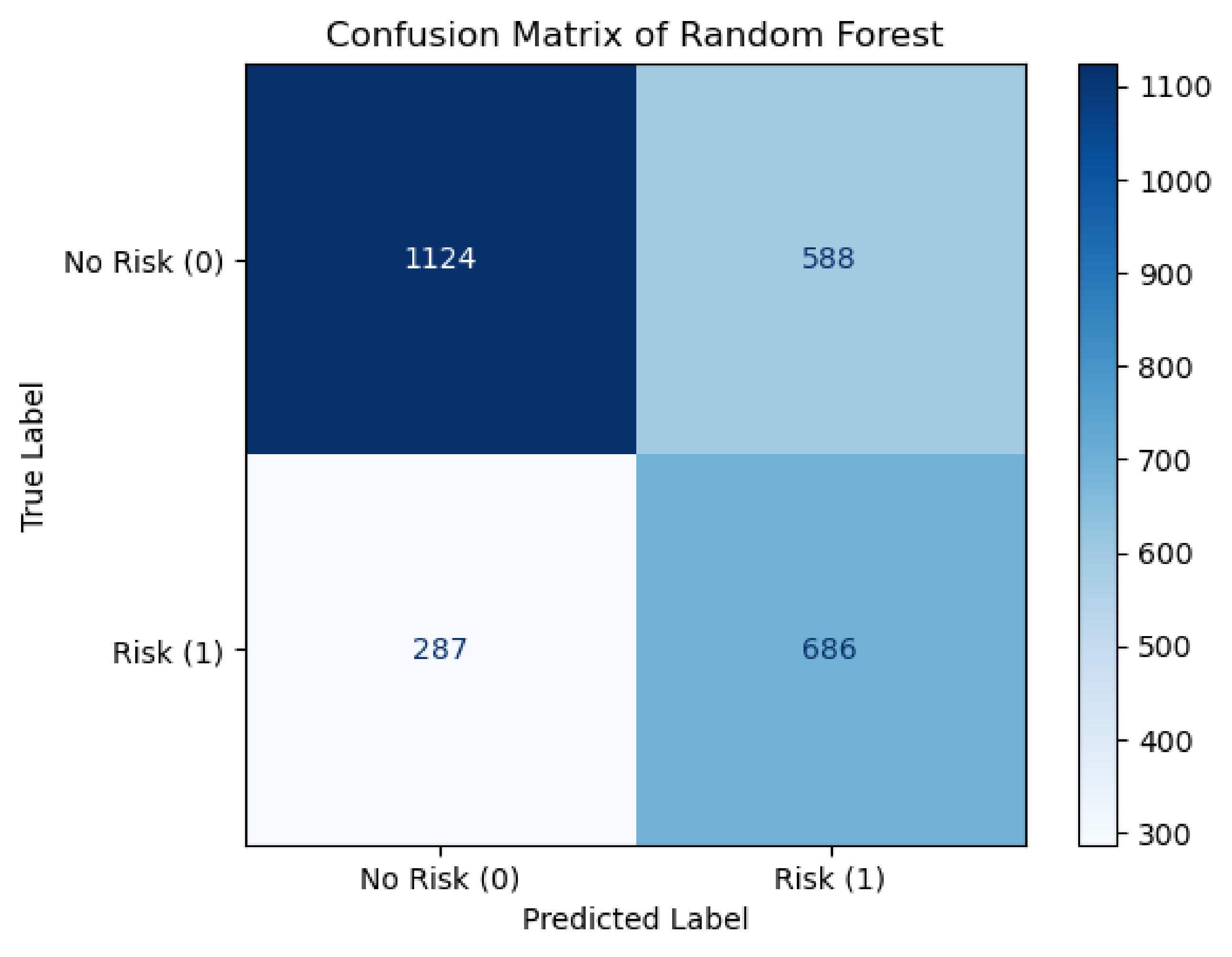


Metric (Risk Class 1)	XGBoost (Image 1)	Random Forest (Image 2)	Evaluation
True Positives (TP)	831	686	XGBoost identifies more actual risks.
False Negatives (FN)	142	287	XGBoost misses fewer risks (Most important).
Recall (Sensitivity)	85.4%	70.5%	XGBoost is significantly superior.
False Positives (FP)	718	588	XGBoost generates more false alarms.
Precision	53.6%	53.9%	Nearly equivalent.
True Negatives (TN)	994	1124	Random Forest is better at identifying safe customers.

Model: RF - Random Forest

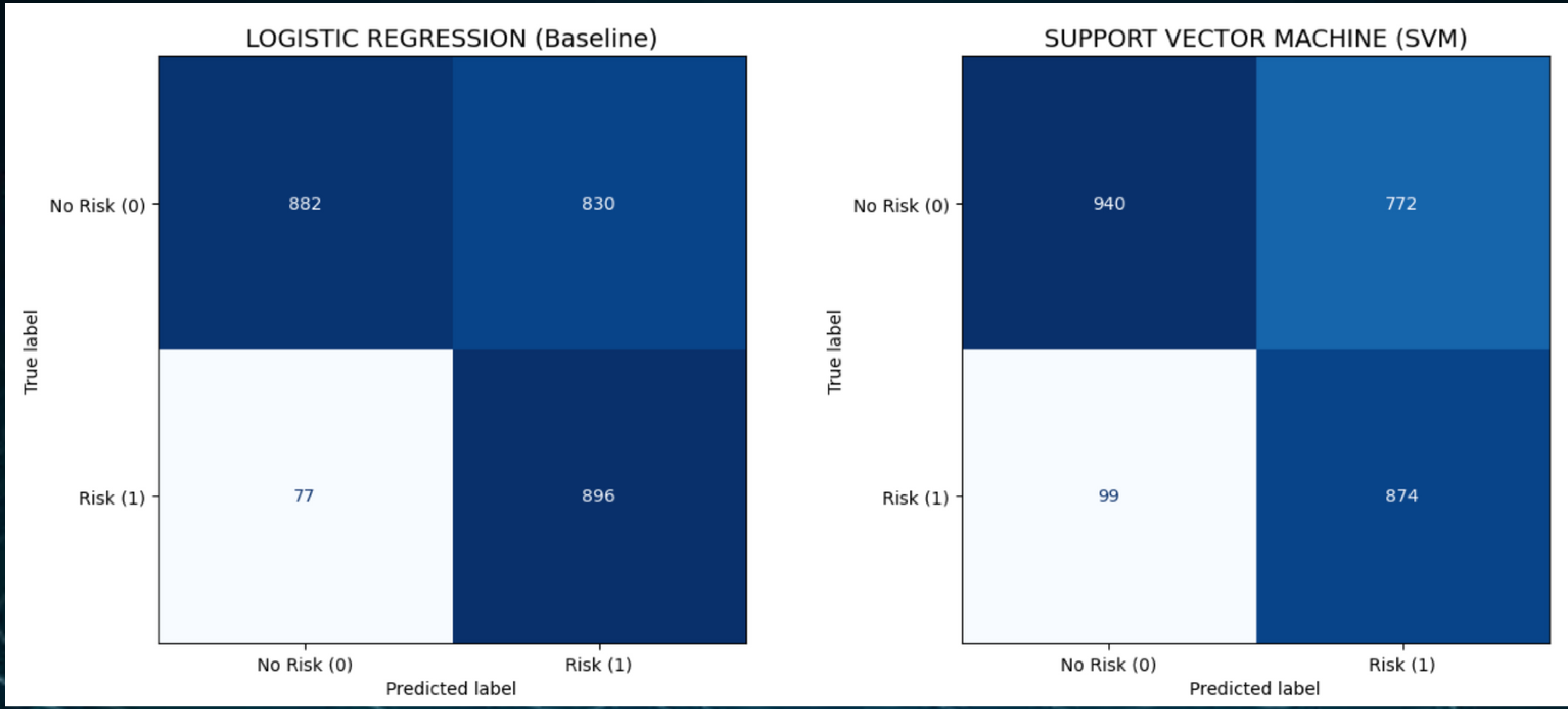


vs.

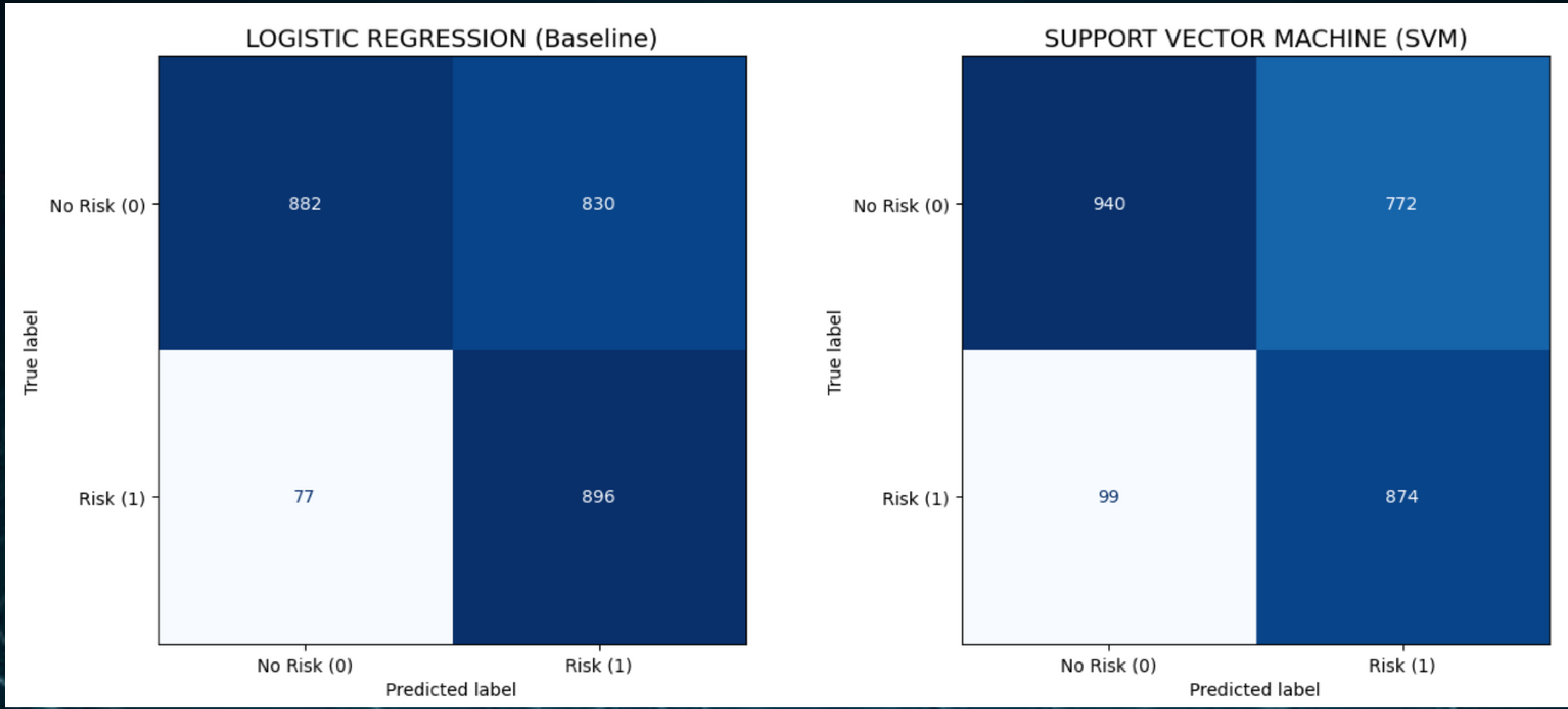


Metric (Risk Class 1)	XGBoost (Image 1)	Random Forest (Image 2)	Evaluation
True Positives (TP)	831	686	XGBoost identifies more actual risks.
False Negatives (FN)	142	287	XGBoost misses fewer risks (Most important).
Recall (Sensitivity)	85.4%	70.5%	XGBoost is significantly superior.
False Positives (FP)	718	588	XGBoost generates more false alarms.
Precision	53.6%	53.9%	Nearly equivalent.
True Negatives (TN)	994	1124	Random Forest is better at identifying safe customers.

Model: Logistic Regression and SVM - Support Vector Machine



Model: Logistic Regression and SVM - Support Vector Machine



Metric (Risk Class 1)	Logistic Regression	SVM	Analysis
True Positives (TP)	896	874	Logistic Regression finds more risks.
False Negatives (FN)	77	99	Logistic Regression misses fewer risks.
False Positives (FP)	830	772	SVM generates fewer false alarms.
True Negatives (TN)	882	940	SVM is more accurate at identifying safe customers.
Recall (Risk/Class 1)	**92.1%**	**89.8%**	Logistic Regression has higher risk detection capability.
Precision (Risk/Class 1)	**51.9%**	**53.1%**	SVM has higher prediction accuracy for risks.
Accuracy	~66.2%	~67.5%	Nearly equivalent overall accuracy.

Model: Logistic Regression and SVM - Support Vector Machine

Metric (Risk Class 1)	Logistic Regression	SVM	Analysis
True Positives (TP)	896	874	Logistic Regression finds more risks.
False Negatives (FN)	77	99	Logistic Regression misses fewer risks.
False Positives (FP)	830	772	SVM generates fewer false alarms.
True Negatives (TN)	882	940	SVM is more accurate at identifying safe customers.
Recall (Risk/Class 1)	**92.1%**	**89.8%**	Logistic Regression has higher risk detection capability.
Precision (Risk/Class 1)	**51.9%**	**53.1%**	SVM has higher prediction accuracy for risks.
Accuracy	~66.2%	~67.5%	Nearly equivalent overall accuracy.

XGBoost vs. Logisic Regression, which is the best?

Model	True Positives (TP)	False Negatives (FN - Missed Risks)	Recall (Detection Rate)	Recommendation
XGBoost	831	142	85.4%	**Best Choice**: Lowest FN, ideal for risk prioritization
Logistic Regression	896	77	92.1%	Highest recall, but very high FP (830), costly operations

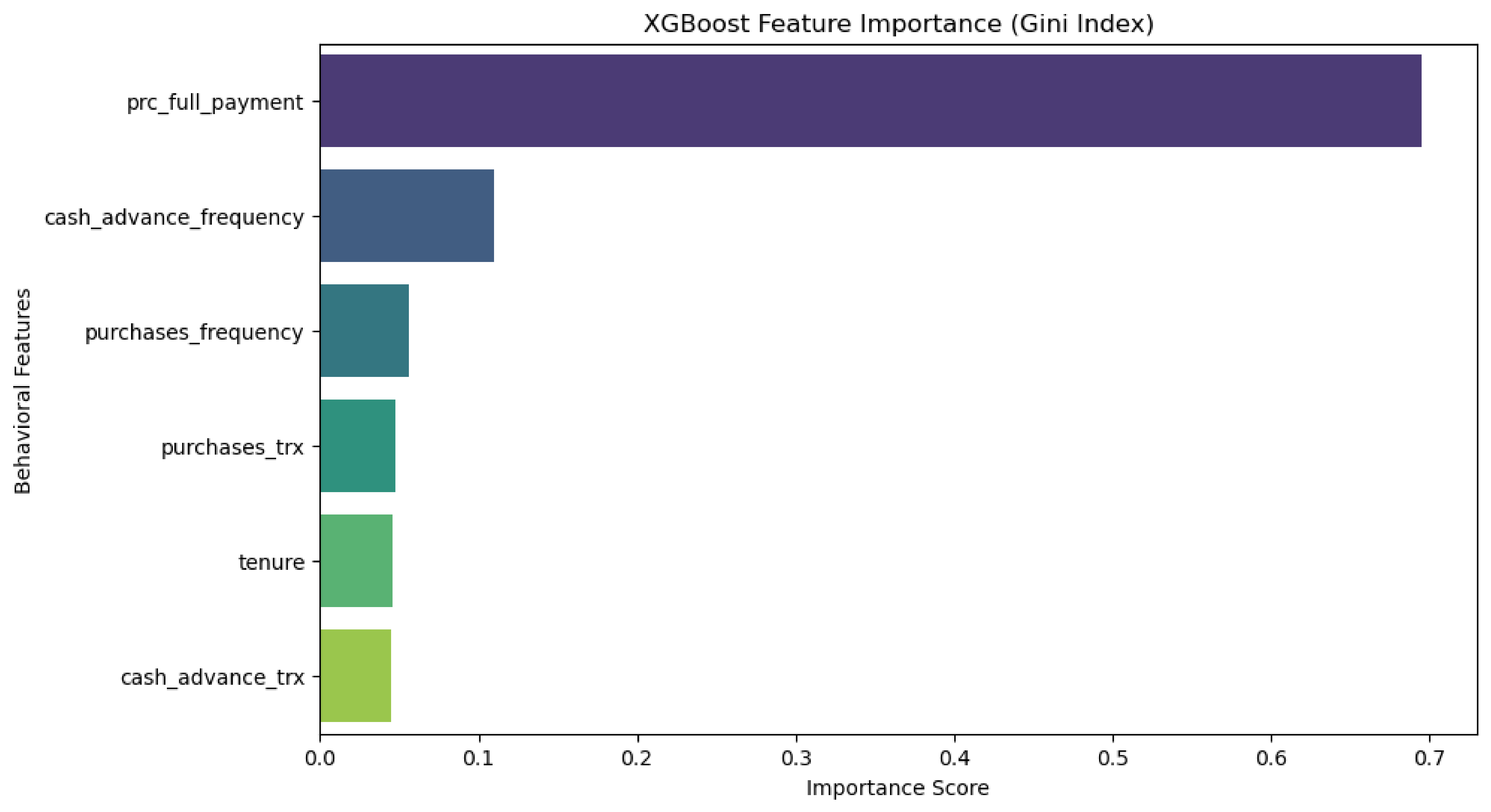
Reason for Choosing XGBoost:

- XGBoost provides the best balance for business objectives. Although Logistic Regression has a higher Recall (92.1%), XGBoost still achieves a very high Recall (85.4%) while better controlling the number of false alarms compared to Logistic Regression.
- In the context of credit risk, the cost of missing an actual risk (142 FN) is typically much higher than the cost of a false alarm. XGBoost achieves the lowest FN among the tree-based models.
- Through Feature Importance and techniques like SHAP analysis, we can interpret why XGBoost makes specific predictions. This explainability is crucial and offers a significant advantage over the more opaque "black box" nature of Logistic Regression in complex scenarios.

Model: XGBoost - Extreme Gradient Boosting

FEATURE IMPORTANCE RANKING:

	importance
prc_full_payment	0.695576
cash_advance_frequency	0.109516
purchases_frequency	0.056370
purchases_trx	0.047773
tenure	0.045892
cash_advance_trx	0.044873



Model: XGBoost - Extreme Gradient Boosting

FEATURE IMPORTANCE RANKING:

	importance
prc_full_payment	0.695576
cash_advance_frequency	0.109516
purchases_frequency	0.056370
purchases_trx	0.047773
tenure	0.045892
cash_advance_trx	0.044873

1. Core Insight: Over 80% of the models predictive power is determined by two factors:
   - Payment Habit (**prc_full_payment**): the habit of paying in full (as opposed to making minimum or insufficient payments) is the number one factor in predicting risk. The model learned that debt repayment behavior is more important than spending behavior.
   - Risky Behavior (**cash_advance_frequency**): Contributes an additional 11%. Customers who frequently use cash advances (a sign of cash shortage) have significantly higher risk.

2. Business Implications and Policy Recommendations
   - Credit Approval/Credit Limit Increase Policy: Decisions to **approve or increase credit limits** should be **based primarily on customers full payment history**, **not just their spending volume.**
   - Early Warning System: Its crucial to establish an automated system to monitor two key metrics:
     - Customers showing a continuous decrease in prc_full_payment.
     - Customers with a sudden spike in cash_advance_frequency.
   - Features like tenure and purchases_trx show low importance: how long a customer has been with the bank or how frequently they spend is less important than how they repay their debts.