Project Report: Customer Segmentation for Cross-Selling

1. Introduction

The goal of this project is to analyze customer data to identify patterns and segment customers for cross-selling opportunities. The dataset contains information about customer transactions, balances, purchases, and other financial behaviors. By understanding customer behavior, businesses can tailor their marketing strategies to target specific customer segments effectively.

2. Dataset Overview

The dataset contains **8,950 entries** and **18 columns**, including:

- **CUST_ID**: Unique identifier for each customer.
- **BALANCE**: The balance amount in the customer's account.
- **BALANCE_FREQUENCY**: How frequently the balance is updated.
- **PURCHASES**: Total purchases made by the customer.
- **ONEOFF_PURCHASES**: One-time purchases.
- **INSTALLMENTS_PURCHASES**: Purchases made in installments.
- **CASH_ADVANCE**: Cash advances taken by the customer.
- **PURCHASES_FREQUENCY**: Frequency of purchases.
- **ONEOFF_PURCHASES_FREQUENCY**: Frequency of one-time purchases.
- **PURCHASES_INSTALLMENTS_FREQUENCY**: Frequency of installment purchases.
- **CASH_ADVANCE_FREQUENCY**: Frequency of cash advances.
- **CASH_ADVANCE_TRX**: Number of cash advance transactions.
- **PURCHASES_TRX**: Number of purchase transactions.
- **CREDIT_LIMIT**: Credit limit of the customer.
- **PAYMENTS**: Payments made by the customer.
- **MINIMUM_PAYMENTS**: Minimum payments made by the customer.
- **PRC_FULL_PAYMENT**: Percentage of full payments made.
- **TENURE**: Tenure of the customer (in months).

3. Data Exploration

3.1 Basic Information

- The dataset has **8,950 rows** and **18 columns**.
- **Missing Values**:
    - **CREDIT_LIMIT**: 1 missing value.
    - **MINIMUM_PAYMENTS**: 313 missing values.
- **Data Types**:
    - Most columns are of type float64 or int64, except for CUST_ID, which is of type object.

3.2 Summary Statistics

- **BALANCE**: The average balance is **1,564.47**, with a maximum of **19,043.14**.
- **PURCHASES**: The average total purchases are **1,003.20**, with a maximum of **49,039.57**.
- **CASH_ADVANCE**: The average cash advance is **978.87**, with a maximum of **47,137.21**.
- **CREDIT_LIMIT**: The average credit limit is **4,494.45**, with a maximum of **30,000.00**.
- **TENURE**: The average tenure is **11.52 months**, with all customers having a tenure between **6 and 12 months**.

3.3 Missing Values

- **CREDIT_LIMIT**: 1 missing value.
- **MINIMUM_PAYMENTS**: 313 missing values.
- These missing values need to be handled before further analysis.

4. Data Visualization

4.1 Distribution of Customer Balances

- The histogram of the BALANCE column shows that most customers have a balance between **0 and 5,000**.
- A small number of customers have very high balances (up to **19,043.14**).

4.2 Relationship Between Balance and Purchases

- The scatter plot between BALANCE and PURCHASES shows a weak positive correlation.
- Most customers with high balances tend to make more purchases, but there are exceptions.

4.3 Correlation Matrix

- The correlation matrix was attempted but failed due to the presence of non-numeric data (CUST_ID).
- To fix this, the CUST_ID column should be dropped before calculating the correlation matrix.

4.4 Mean Balance by Tenure

- The bar plot shows the mean balance for each tenure period.
- Customers with a tenure of **12 months** have the highest mean balance.

5. Key Findings

1. **Customer Balances**:
    - Most customers have low balances, but a small segment has very high balances.
    - High-balance customers are potential targets for premium products or services.
2. **Purchases**:
    - Customers with higher balances tend to make more purchases, but the relationship is not strong.
    - One-time purchases and installment purchases are common among customers.
3. **Cash Advances**:
    - A significant number of customers take cash advances, with some taking very large amounts.
    - Customers who frequently take cash advances may need tailored financial products.
4. **Credit Limits**:

- The average credit limit is **4,494.45**, but some customers have limits as high as **30,000.00**.
- Customers with high credit limits may be more likely to engage in large purchases.

5. **Tenure**:
   - Most customers have a tenure of **12 months**.
   - Customers with longer tenure tend to have higher balances and make more purchases.

6. Recommendations

1. **Target High-Balance Customers**:
   - Offer premium products or services to customers with high balances.
   - Provide personalized financial advice to retain these customers.
2. **Cross-Sell to Frequent Purchasers**:
   - Identify customers who make frequent purchases and offer them complementary products.
   - Use loyalty programs to encourage repeat purchases.
3. **Cash Advance Customers**:
   - Offer financial products like low-interest loans or credit cards to customers who frequently take cash advances.
   - Educate these customers on better financial management.
4. **Credit Limit Utilization**:
   - Encourage customers with high credit limits to utilize their limits by offering discounts or rewards.
   - Monitor customers who are close to their credit limits and offer credit limit increases.
5. **Tenure-Based Marketing**:
   - Reward long-tenure customers with exclusive offers or discounts.
   - Focus on retaining customers who have been with the company for a shorter duration.

7. Next Steps

1. **Handle Missing Values**:
    - Impute missing values in CREDIT_LIMIT and MINIMUM_PAYMENTS using appropriate methods (e.g., mean, median, or predictive modeling).
2. **Customer Segmentation**:
    - Use clustering algorithms (e.g., K-Means) to segment customers based on their behavior (e.g., balance, purchases, cash advances).
    - Analyze each segment to identify specific cross-selling opportunities.
3. **Predictive Modeling**:
    - Build predictive models to identify customers likely to make large purchases or take cash advances.
    - Use these models to target customers with personalized offers.
4. **A/B Testing**:
    - Test different marketing strategies on customer segments to determine the most effective approach.
    - Measure the impact of cross-selling campaigns on customer behavior.

8. Conclusion

This project provides valuable insights into customer behavior and identifies key segments for cross-selling opportunities. By leveraging these insights, businesses can optimize their marketing strategies, improve customer retention, and increase revenue. Further analysis and modeling will help refine these strategies and drive better results.

```python
import pandas as pd
import matplotlib.pyplot as plt

# Load the dataset
data = pd.read_csv('Customer Data.csv')

# Display the first few rows of the dataset
print(data.head())

# Basic information about the dataset
print(data.info())

# Summary statistics
print(data.describe())

# Check for missing values
print(data.isnull().sum())

# Visualize the distribution of the 'BALANCE' column
plt.figure(figsize=(10, 6))
plt.hist(data['BALANCE'], bins=50, color='blue', alpha=0.7)
plt.title('Distribution of Customer Balances')
plt.xlabel('Balance')
plt.ylabel('Frequency')
plt.show()

# Visualize the relationship between 'BALANCE' and 'PURCHASES'
plt.figure(figsize=(10, 6))
plt.scatter(data['BALANCE'], data['PURCHASES'], alpha=0.5)
plt.title('Balance vs Purchases')
plt.xlabel('Balance')
plt.ylabel('Purchases')
plt.show()

# Correlation matrix
correlation_matrix = data.corr()
print(correlation_matrix)

# Visualize the correlation matrix using a heatmap
import seaborn as sns

plt.figure(figsize=(12, 8))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm')
plt.title('Correlation Matrix')
plt.show()

# Group by 'TENURE' and calculate the mean balance
tenure_balance = data.groupby('TENURE')['BALANCE'].mean()
print(tenure_balance)
```

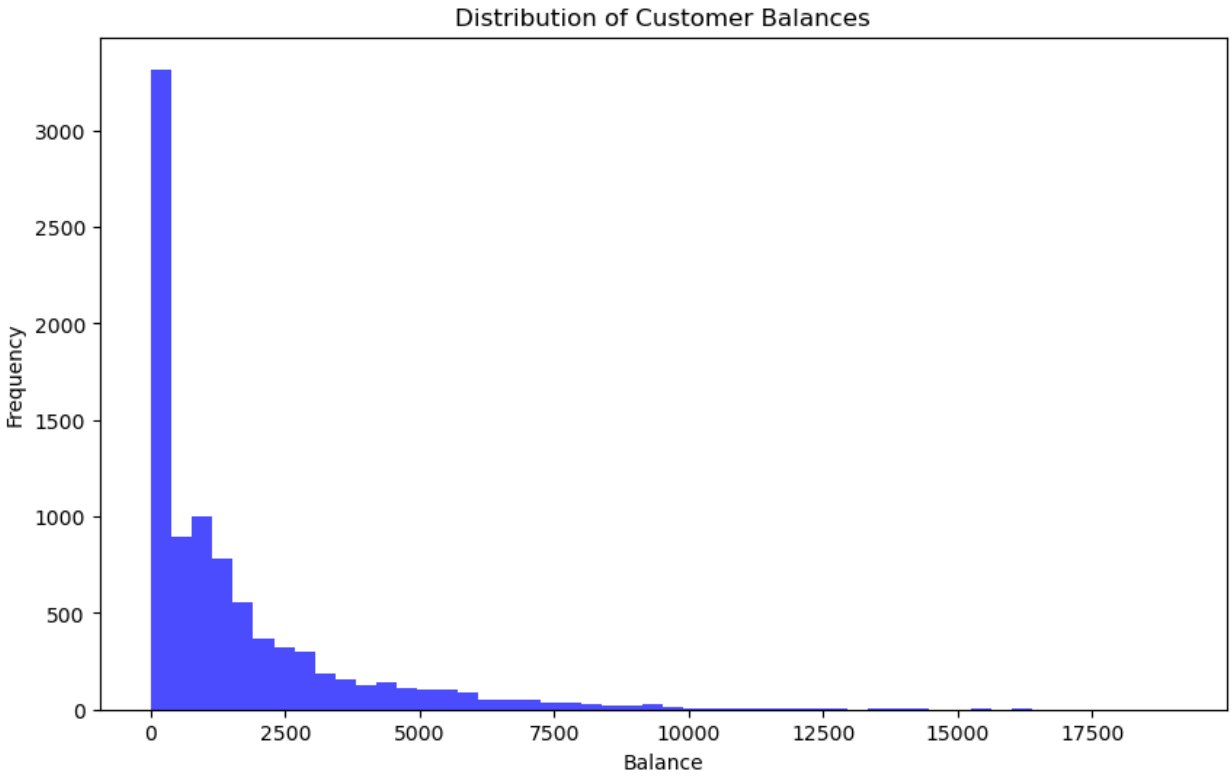
```
# Plot the mean balance by tenure
plt.figure(figsize=(10, 6))
tenure_balance.plot(kind='bar', color='green')
plt.title('Mean Balance by Tenure')
plt.xlabel('Tenure (Months)')
plt.ylabel('Mean Balance')
plt.show()
```

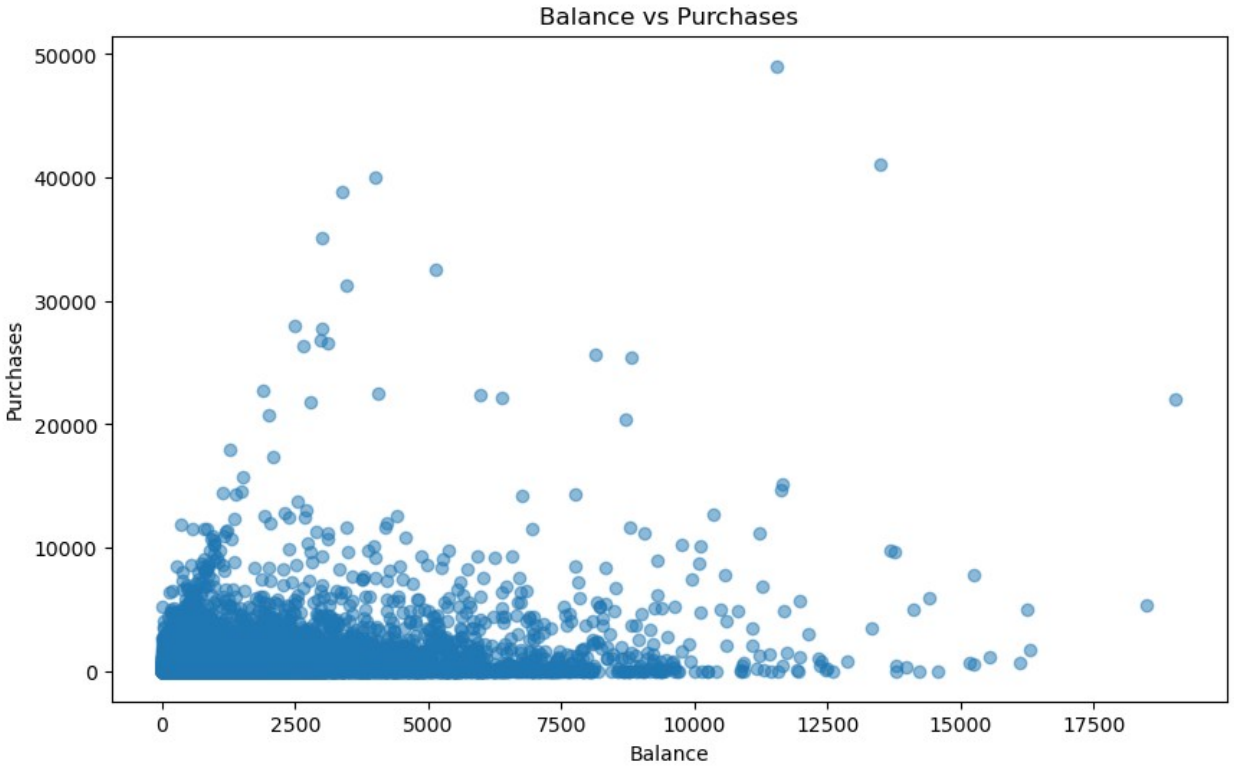
```
  CUST_ID      BALANCE  BALANCE_FREQUENCY  PURCHASES  ONEOFF_PURCHASES
\
0  C10001    40.900749           0.818182      95.40              0.00

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  \
0                    95.4      0.000000             0.166667
1                     0.0   6442.945483             0.000000
2                     0.0      0.000000             1.000000
3                     0.0    205.788017             0.083333
4                     0.0      0.000000             0.083333

   ONEOFF_PURCHASES_FREQUENCY  PURCHASES_INSTALLMENTS_FREQUENCY  \
0                    0.000000                          0.083333
1                    0.000000                          0.000000
2                    1.000000                          0.000000
3                    0.083333                          0.000000
4                    0.083333                          0.000000

   CASH_ADVANCE_FREQUENCY  CASH_ADVANCE_TRX  PURCHASES_TRX
CREDIT_LIMIT  \
0                0.000000                 0              2
1000.0
1                0.250000                 4              0
7000.0
2                0.000000                 0             12
7500.0
3                0.083333                 1              1
7500.0
4                0.000000                 0              1
1200.0

      PAYMENTS  MINIMUM_PAYMENTS  PRC_FULL_PAYMENT  TENURE
0   201.802084        139.509787          0.000000      12
```

```
1  4103.032597       1072.340217          0.222222      12
2   622.066742        627.284787          0.000000      12
3     0.000000               NaN          0.000000      12
4   678.334763        244.791237          0.000000      12
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 8950 entries, 0 to 8949
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)
memory usage: 1.2+ MB
None
            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
8949.000000
mean                 0.135144          3.248827      14.709832
4494.449450
std                  0.200121          6.824647      24.857649
3638.815725
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       8637.000000       8950.000000  8950.000000
mean    1733.143852        864.206542          0.153715    11.517318
std     2895.063757       2372.446607          0.292499     1.338331
min        0.000000          0.019163          0.000000     6.000000
25%      383.276166        169.123707          0.000000    12.000000
50%      856.901546        312.343947          0.000000    12.000000
75%     1901.134317        825.485459          0.142857    12.000000
max    50721.483360      76406.207520          1.000000    12.000000
CUST_ID                             0
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
```






```
---------------------------------------------------------------------------
-----
ValueError                                Traceback (most recent call
last)
Cell In[1], line 36
     33 plt.show()
     35 # Correlation matrix
---> 36 correlation_matrix = data.corr()
     37 print(correlation_matrix)
     39 # Visualize the correlation matrix using a heatmap

File ~\anaconda3\Lib\site-packages\pandas\core\frame.py:11049, in
DataFrame.corr(self, method, min_periods, numeric_only)
  11047 cols = data.columns
  11048 idx = cols.copy()
> 11049 mat = data.to_numpy(dtype=float, na_value=np.nan, copy=False)
  11051 if method == "pearson":
  11052     correl = libalgos.nancorr(mat, minp=min_periods)

File ~\anaconda3\Lib\site-packages\pandas\core\frame.py:1993, in
DataFrame.to_numpy(self, dtype, copy, na_value)
   1991 if dtype is not None:
   1992     dtype = np.dtype(dtype)
-> 1993 result = self._mgr.as_array(dtype=dtype, copy=copy,
na_value=na_value)
   1994 if result.dtype is not dtype:
```

```
   1995     result = np.asarray(result, dtype=dtype)

File ~\anaconda3\Lib\site-packages\pandas\core\internals\
managers.py:1694, in BlockManager.as_array(self, dtype, copy,
na_value)
   1692         arr.flags.writeable = False
   1693 else:
-> 1694     arr = self._interleave(dtype=dtype, na_value=na_value)
   1695     # The underlying data was copied within _interleave, so no
need
   1696     # to further copy if copy=True or setting na_value
   1698 if na_value is lib.no_default:

File ~\anaconda3\Lib\site-packages\pandas\core\internals\
managers.py:1753, in BlockManager._interleave(self, dtype, na_value)
   1751     else:
   1752         arr = blk.get_values(dtype)
-> 1753     result[rl.indexer] = arr
   1754     itemmask[rl.indexer] = 1
   1756 if not itemmask.all():

ValueError: could not convert string to float: 'C10001'
```