Unlocking Insights :

Exploring Sales Trends, Customer Behavior, and Future Projections in RevoBank's Credit Card Business

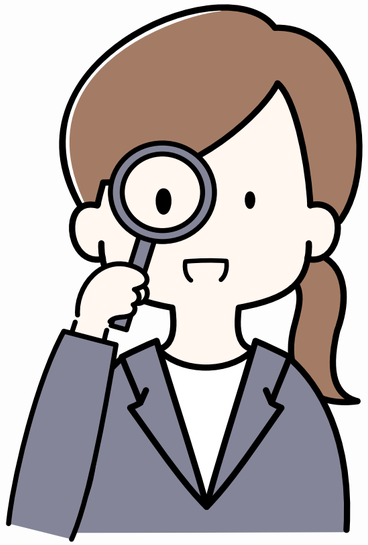
Week 6, 7, 8 - Python for Data Analytics

FSDA - Barcelona -Team 1 - Michael

Table of Contents

Analytical Objective

The goal of this analysis is to understand customer segmentation (such as aggregating by generations, gender, and frequency of promos), create recommendations based on the insights found, and identify business opportunities to raise the number of sales for credit card business of Revo Bank



Data Cleaning Summary

Data Cleaning Steps	Original Data	After Cleaning	Reason to do Cleaning Step
Treat duplicates	12487	12487	No duplicates found
Treat missing values	12487	11751	Remove missing values on avg_sales_L36M, because There is only 1 unique value in 'cnt_sales_L36M' where avg_sales_L36M is null, which is 0 (zero in count of sales, indicates no purchase made in the last 36 months. Therefore we can exlcude all these rows in the analysis)
Remove irrelevant data	11751	11750	Drops customer_value_level 'F', insignificant effects for overall of data
Remove clients under age 21	11750	11628	Remove customers with age under 21, like the question asked
Convert to correct data types	change 'birth_date' from object to datetime data type	Object data type	Datetime data type
	change identifier which is 'account_id' from integer to string data type	Integer data type	String data type
	change 'MOB' from float to integer data type	Float datatype	Integer data type

Checking Duplicated Values

The Syntax :

```
[143] cc_performance[cc_performance['account_id'].duplicated()]
```

The Result :

CHECK DUPLICATED VALUES

```
[143] cc_performance[cc_performance['account_id'].duplicated()]
```

account_id	account_activity_level	customer_value_level	MOB	flag_female	avg_sales_L36M	cnt_sales_L36M	last_sales	month_since_last_sales	count_direct_promo_L12M	birth_date

No duplicates were found

Remove Irrelevant Data

```
for column_name in cc_performance_deletedrows.columns:
  print(column_name)
  print(cc_performance_deletedrows[column_name].value_counts())
  print('\n')
```

Check every unique value in a column, and it's "count"

Finding :

```
customer_value_level
E    4205
B    2217
A    2075
C    1880
D    1373
F       1
Name: customer_value_level, dtype: int64
```

Customer_value_level **'F'**, is not defined in the data dictionary, so we can exclude this one customer from the analysis

Treat Missing Values

The Syntax :

```
[116] filtered_cc_performance_cleaning_check = cc_performance_cleaning_check[cc_performance_cleaning_check['avg_sales_L36M'].isna()]
      filtered_cc_performance_cleaning_check
```

Check how many average sales value is null.

Found 736 customers with zero sales on the last 36 months

	account_id	account_activity_level	customer_value_level	MOB	flag_female	avg_sales_L36M	cnt_sales_L36M
23	100002370	Z	A	110	0	NaN	0
26	100002547	X	A	61	0	NaN	0
37	100003731	X	A	54	1	NaN	0
59	100005474	Z	A	56	0	NaN	0
86	100008094	X	B	127	1	NaN	0
...	...	...	...	...	...	...	...
12429	101054857	Z	E	51	1	NaN	0
12429	101055421	X	E	54	0	NaN	0
12432	101055837	Z	E	51	0	NaN	0
12449	101057435	X	E	123	0	NaN	0
12461	101058331	Z	E	55	1	NaN	0

736 rows × 11 columns

The result

```
[117] filtered_cc_performance_cleaning_check = cc_performance_cleaning_check[cc_performance_cleaning_check['avg_sales_L36M'].isna()]
      unique_count_sales = filtered_cc_performance_cleaning_check ['cnt_sales_L36M'].nunique()
      print (unique_count_sales)

      1
```

See if we can remove all these rows ?

There is only 1 unique value in 'cnt_sales_L36M' where avg_sales_L36M is null, which is 0 **(zero in count of sales, indicates no purchase made in the last 36 months)**. Therefore we can **exlcude** all these rows in the analysis)

Exclude Customer Under 21 Years of Age

Calculate each customer's age by subtracting today's date with their birth date

```
cc_performance_deletedrows['client_age'] = (datetime(year=2023, month = 1, day=31) - cc_performance_deletedrows['birth_date']) // np.timedelta64(1,'Y')
cc_performance_deletedrows
```

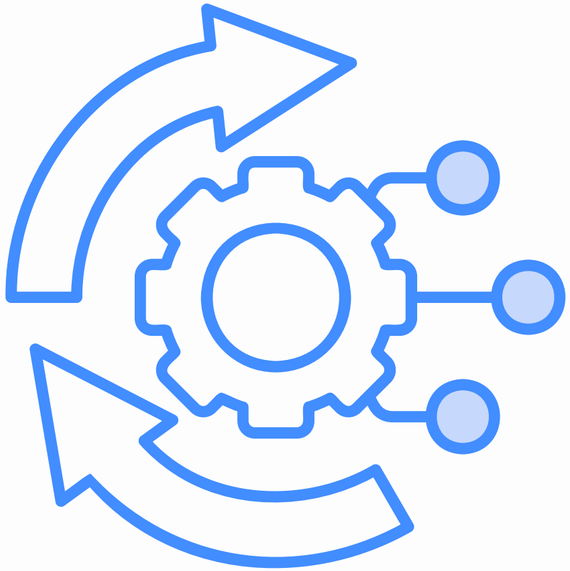
How many customers are under 21 ?

```
cc_performance_deletedrows.loc[ cc_performance_deletedrows['client_age'] < 21 ]
```

Excluding them. There 122 customers under the age of 21

```
cc_performance_deletedrows = cc_performance_deletedrows[~(cc_performance_deletedrows['client_age'] < 21) ]
cc_performance_deletedrows
```

Changing Datatypes



CHANGING ALL DATA RELATED TO MONEY TO FLOAT. ON THIS TABLE, MONEY DATA CONSISTS :

- avg_sales_L36M (already float)
- last_sales (already float)

All data related to money already in float data type, so there is no need to make any change.

CHANGING BIRTH DATE TO DATE TIME

```
[110] cc_performance_cleaning['birth_date'] = pd.to_datetime(cc_performance_cleaning['birth_date'])
      cc_performance_cleaning
```

CHANGING IDENTIFIER (ACCOUNT ID) TO STRING

```
[111] cc_performance_cleaning[['account_id']] = cc_performance_cleaning [['account_id']].astype(str)
```

CHANGING COUNT DATA TO INTEGER. ON THIS TABLE, COUNT DATA CONSISTS :

- cnt_sales_L36M (already integer)
- count_direct_promo_L12M (already integer)
- MOB (currently float)
- month_since_last_sales (already integer)

```
[113] cc_performance_cleaning[['MOB']] = cc_performance_cleaning[['MOB']].astype(int)
```

Descriptive Statistics

Calculate total sales for the last 3 years

Multiply average sales with count of sales to get total sales for every customer

```
[126] cc_performance_totalsales['total_sales'] = cc_performance_totalsales ['avg_sales_L36M'] * cc_performance_totalsales ['cnt_sales_L36M']
      cc_performance_totalsales
```

The answer : sum ‘total_sales’

```
[127] sum_of_sales = cc_performance_totalsales ['total_sales'].sum()
      formatted_sum = '{:,}'.format (sum_of_sales)
      print ("Sum of Sales :", formatted_sum)

      Sum of Sales : 402,495,520.0
```

Descriptive Statistics

Identify percentage of clients with no sales in the last 36 months (1)

Steps to do :

filter customers with zero transactions

count the number of zero transactions customers

count the total number of customers

count the number of customers with transaction

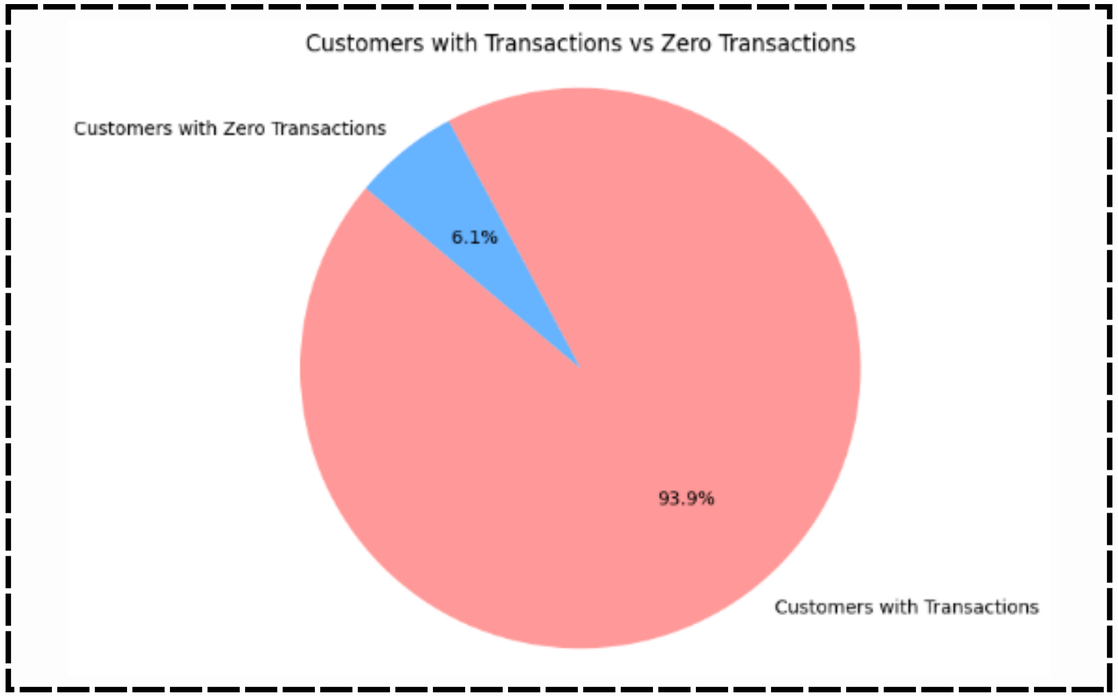
```
# Filter customers with zero transactions
zero_transaction_customers = cc_performance_nosales[cc_performance_nosales['total_sales'] == 0]

# Count the number of zero transaction customers
zero_transaction_count = len(zero_transaction_customers)

# Count the number of customers
total_customers = len(cc_performance_nosales)

# Count the number of customers with transactions
non_zero_transaction_count = total_customers - zero_transaction_count
```

Descriptive Statistics

Identify percentage of clients with no sales in the last 36 months (2)

Pie chart plotting (syntax) :

```
# Plotting the pie chart
labels = ['Customers with Transactions', 'Customers with Zero Transactions']
sizes = [non_zero_transaction_count, zero_transaction_count]
colors = ['#ff9999', '#66b3ff']

plt.figure(figsize=(6, 6))
plt.pie(sizes, labels=labels, colors=colors, autopct='%1.1f%%', startangle=140)
plt.axis('equal')  # Equal aspect ratio ensures that pie is drawn as a circle.
plt.title('Customers with Transactions vs Zero Transactions')
plt.show()
```



There is a small amount of number for customers without transaction in the last 36 months, this could also indicates good churn rate

Descriptive Statistics

Investigating activity levels by median months since last transaction (1)

The hypothesis : “X” account activity level is the most active level

```
[134] median_months_by_level = cc_performance_activity_level_analysis.groupby('account_activity_level')['month_since_last_sales'].median().sort_values()
      median_months_table = median_months_by_level.to_frame(name = 'median_months_since_last_transaction')
      print (median_months_table)

                         median_months_since_last_transaction
      account_activity_level
      Y                                                  17.0
      X                                                  18.0
      Z                                                  18.0
```

It is found that Y is the most active level by median months since last transaction, not X

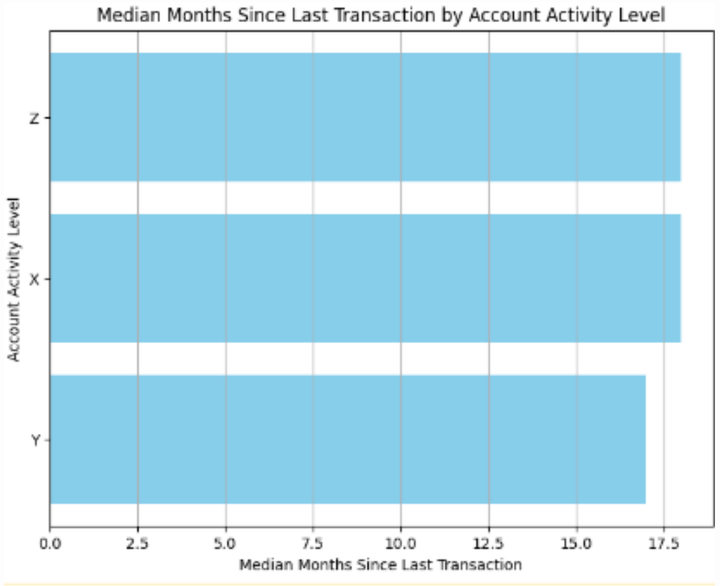
What the syntax did : calculate the median of ‘month_since_last_sales’ and group them by ‘account_activity_level’.

It is found that Y has the lowest value of median months, this shows that accounts with Y activity level is the most frequent users that used credit cards. Hence, the hypothesis stated in the question is not yet right

Descriptive Statistics

Investigating activity levels by median months since last transaction (2)

Horizontal bar chart plotting (syntax) :




```
plt.figure(figsize=(8, 6))
plt.barh(median_months_table.index, median_months_table['median_months_since_last_transaction'], color='skyblue')
plt.title('Median Months Since Last Transaction by Account Activity Level')
plt.xlabel('Median Months Since Last Transaction')
plt.ylabel('Account Activity Level')
plt.grid(axis='x')
plt.show()
```

```
[133] median_months = cc_performance_activity_level_analysis.groupby('account_activity_level')['month_since_last_sales'].median().reset_index()
      most_active_level = median_months.loc[median_months['month_since_last_sales'].idxmin(), 'account_activity_level']
      most_active_level

      'Y'
```

More direct way of showing lowest median month by account activity level

Descriptive Statistics

Sales by gender, is it true that males provide more profit to the company ? (1)

The hypothesis : Male clients has more profitability.

```
gender_sales_stats_sum_only = cc_performance_gender_agg.groupby('flag_female')['total_sales'].sum().to_frame(name = 'total_sales_sum')
gender_sales_stats_sum_only['total_sales_sum'] = gender_sales_stats_sum_only['total_sales_sum'].apply(lambda x: '{:,.0f}'.format(x))
gender_sales_stats_sum_only
```

flag_female	total_sales_sum
0	186,438,660
1	216,056,860

What the syntax did : calculate total sales and group them by gender. 1 = Females, 0 = Males

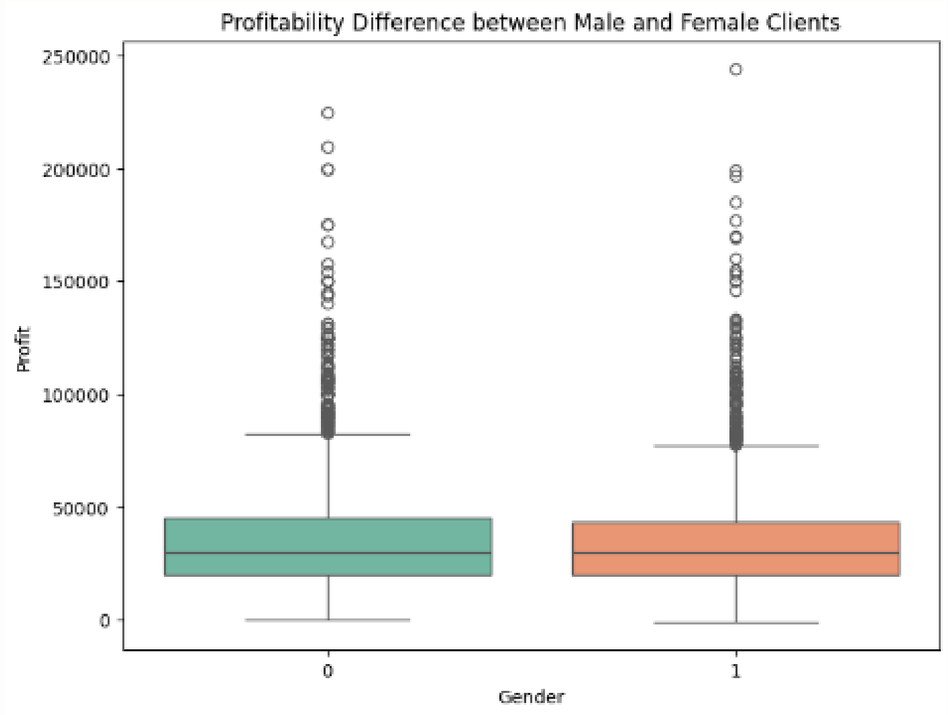
It is clear enough that females customers provides more sales to the company, hence the hypothesis stated in the question is not yet right.

Descriptive Statistics

Sales by gender, is it true that males provide more profit to the company ? (2)

Boxplotting (syntax) :

```
[137] plt.figure(figsize=(8, 6))
      sns.boxplot(x='flag_female', y='total_sales', data=cc_performance_gender_agg, palette='Set2')
      plt.title('Profitability Difference between Male and Female Clients')
      plt.xlabel('Gender')
      plt.ylabel('Profit')
      plt.show()

      print(gender_sales_stats)
```




```
[136] gender_sales_stats = cc_performance_gender_agg.groupby('flag_female')['total_sales'].describe()
      gender_sales_stats
```

flag_female	count	mean	std	min	25%	50%	75%	max
0	5335.0	34946.328022	20753.658365	0.0	20000.0	30000.0	45000.0	225000.0
1	6293.0	34332.887335	19584.114158	-1000.0	20000.0	30000.0	43000.0	244400.0

Male clients have a slightly higher mean of sales, but female clients had more transactions, that is why total sales from female clients are higher

Descriptive Statistics

Generation Segmentation (1)

```
# Calculate the year of birth
cc_performance_generation_segmentation['year_of_birth'] = cc_performance_generation_segmentation['birth_date'].dt.year

# Define the bins for different generations
bins = [1946, 1964, 1980, 1996, 2012,2023]
labels = ['Boomer', 'Gen-X', 'Millennial', 'Gen-Z', 'Gen-Alpha']

# Categorize years into generations
cc_performance_generation_segmentation['generation'] = pd.cut(cc_performance_generation_segmentation['year_of_birth'], bins=bins, labels=labels, right=False)

# Calculate total sales for each generation
generation_sales = cc_performance_generation_segmentation.groupby('generation')['total_sales'].sum()

# Calculate proportion of sales for each generation
proportion_sales = generation_sales / generation_sales.sum()
```

The steps :

 # calculate year of birth by extracting year from ‘birth_date’ column

 # define the bins for generations

 # categorize years into generations

 # calculate total sales for each generations

 # calculate proportion of sales for each generation

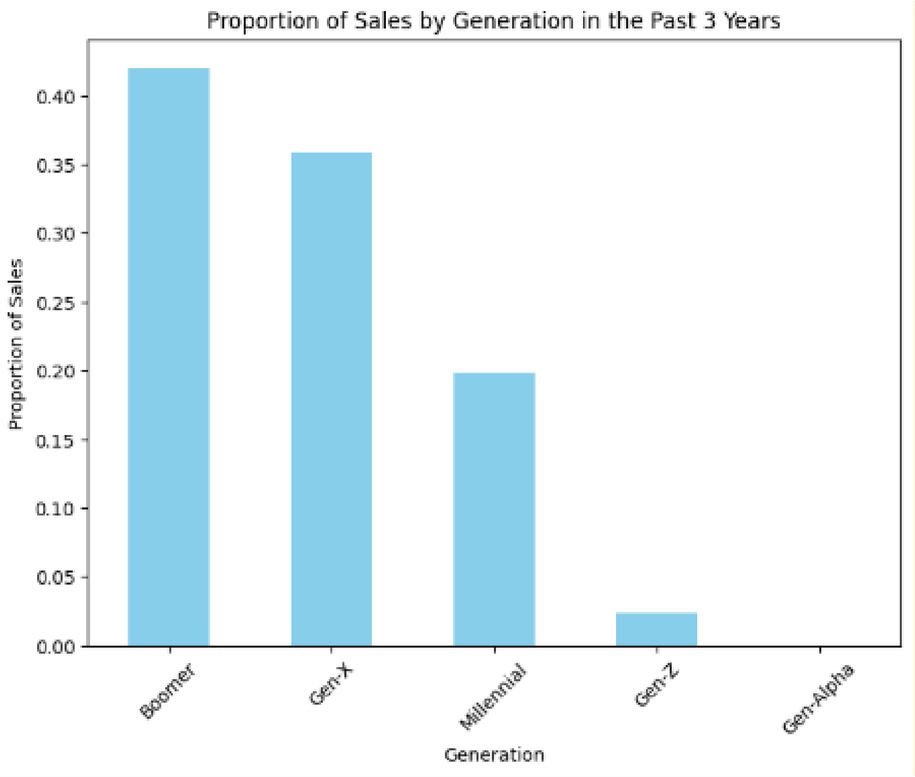
Descriptive Statistics

Generation Segmentation (2)

Bar chart plotting (syntax) :

```
# Plotting the proportion of sales for each generation
plt.figure(figsize=(8, 6))
proportion_sales.plot(kind='bar', color='skyblue')
plt.title('Proportion of Sales by Generation in the Past 3 Years')
plt.xlabel('Generation')
plt.ylabel('Proportion of Sales')
plt.xticks(rotation=45)
plt.show()

print(proportion_sales)
```

```
generation
Boomer          0.419900
Gen-X           0.358686
Millennial      0.198037
Gen-Z           0.023377
Gen-Alpha       0.000000
Name: total_sales, dtype: float64
```



‘Boomer’ generation has the highest sales proportion for credit card performance. There is no ‘Gen-Alpha’ customers detected because we already dropped customer under the age of 21

Descriptive Statistics

Promos Effect to Sales (1)

```
[142] # Group by count of direct promo and calculate the average sales per customer in each group
      cc_performance_promos_effect = cc_performance_promos_effect.groupby('count_direct_promo_L12M')['total_sales'].mean().reset_index()
      cc_performance_promos_effect.rename(columns={'total_sales': 'average_of_total_sales'}, inplace=True)
      cc_performance_promos_effect
```

What the syntax did : calculate mean of ‘total_sales’ and group them by ‘count_of_direct_promo_L12M’

```
# Plotting the relationship between count of direct promo and average sales per customer
plt.figure(figsize=(8, 6))
plt.plot(cc_performance_promos_effect['count_direct_promo_L12M'], cc_performance_promos_effect['average_of_total_sales'], marker='o', color='skyblue', linestyle='-')
plt.title('Relationship between Count of Direct Promo and Average Sales')
plt.xlabel('Count of Direct Promo')
plt.ylabel('Average Sales by Count of Promos')
plt.grid(True)
plt.show()

print(cc_performance_promos_effect)
```

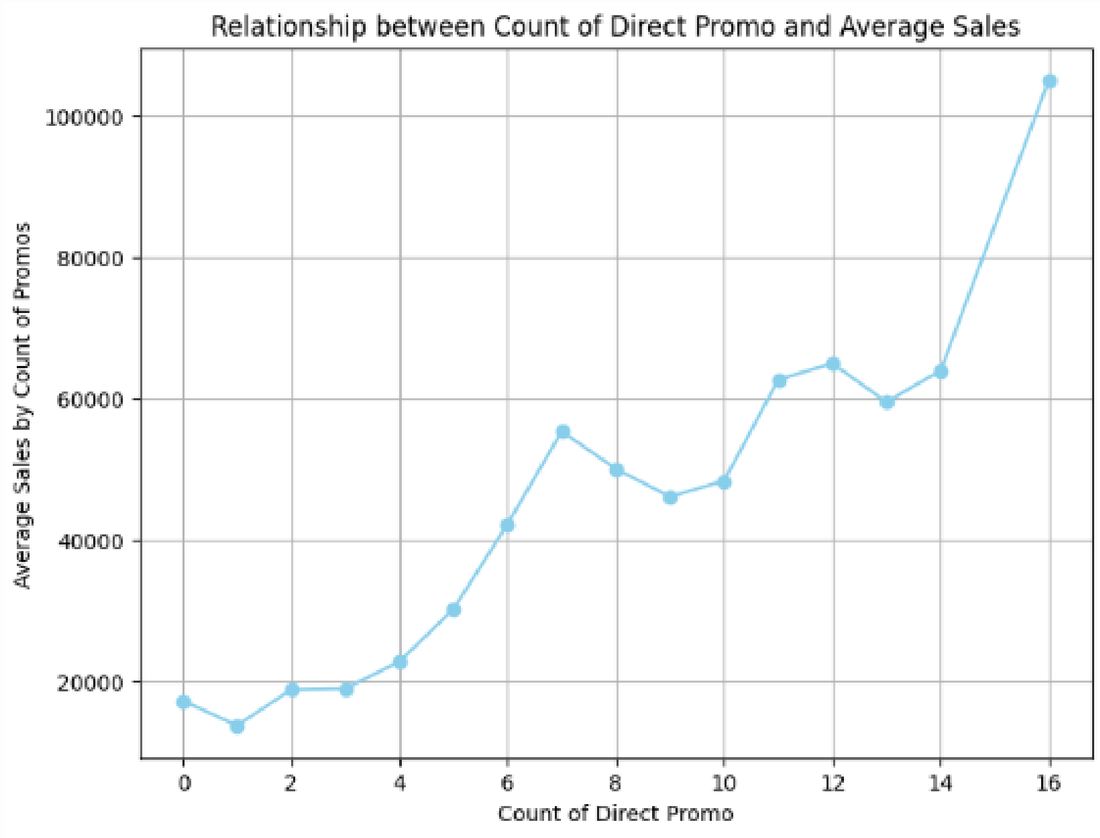
Line chart plotting

Descriptive Statistics

Promos Effect to Sales (2)

count_direct_promo_L12M	average_of_total_sales
0	17333.333333
1	13810.666667
2	18877.750000
3	19049.965870
4	22848.509154
5	30282.095142
6	42304.825442
7	55388.550420
8	50098.235294
9	46157.777778
10	48397.500000
11	62705.882353
12	64991.428571
13	59542.500000
14	64000.000000
16	105050.000000

More promos does not necessarily mean higher sales, but the trendline shows positive proportional change of sales. Therefore raising the number of direct promos is still recommended.



Customer Segmentation With K-Means Clustering

*Why K-Means Over RFM ?

- More versatile for various types of data and industry
- More efficient in terms of computation for large datasets and more linear complexity
- Dataset contains a lot of numerical data

Data Scaler : Robust Scaler

Variables to be included :

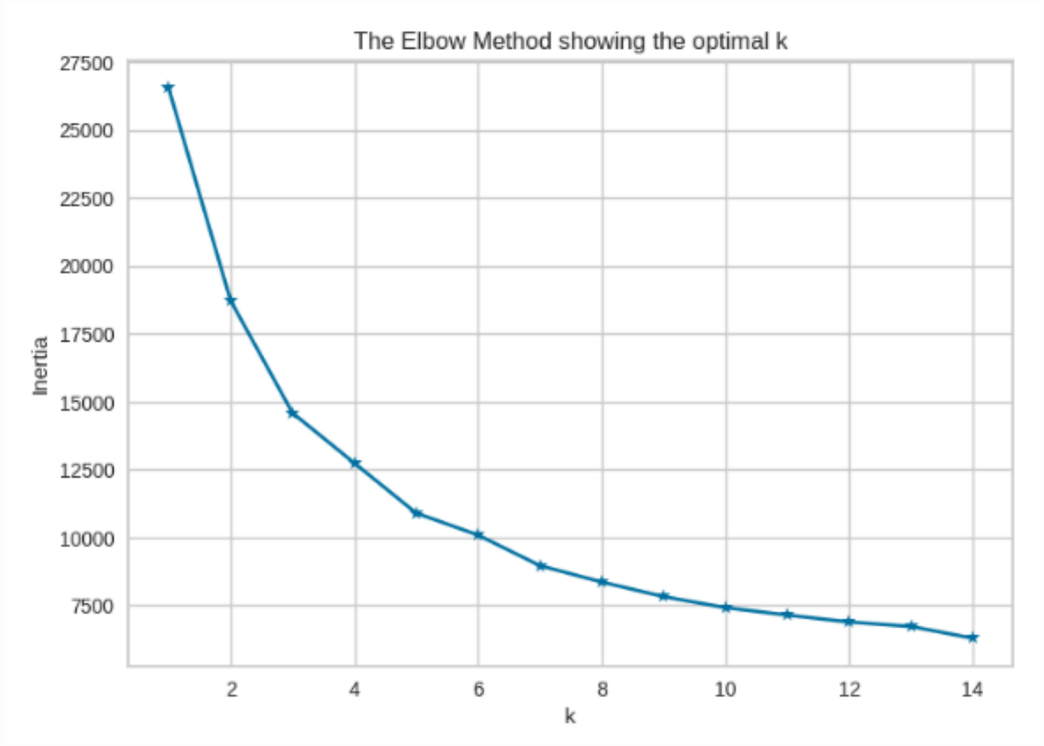
- MOB
- Average Sales Last 3 Years
- Number of Transactions Last 3 Years
- Client Age
- Total Sales

```
rs = RobustScaler()
rs_result = rs.fit_transform(kmeans_analysis)
rs_result
```

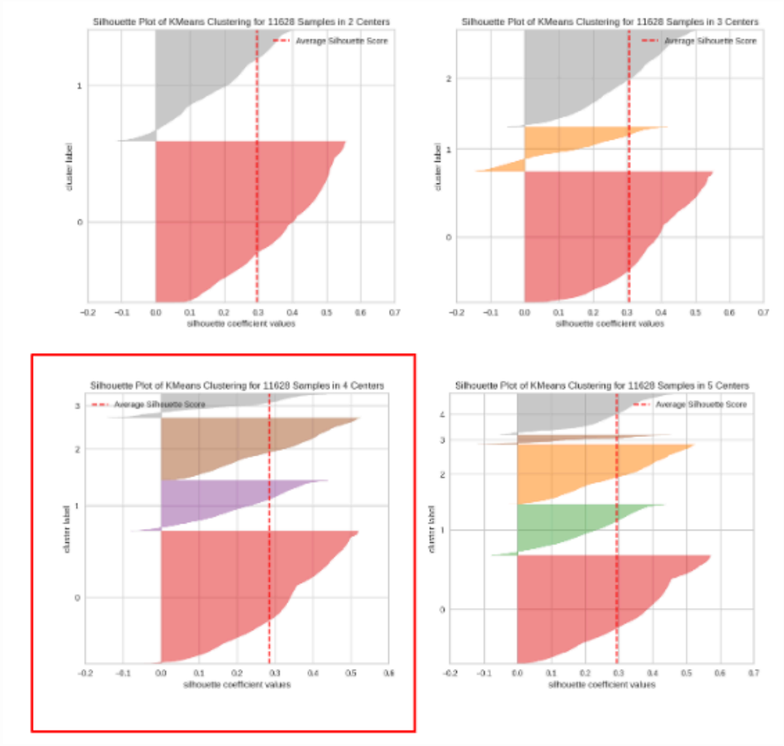
```
array([[ 0.48101266,  1.        , -0.33333333, -1.22222222, -0.21052632],
       [ 0.20253165, -0.6       ,  1.        , -0.2962963 ,  0.63157895],
       [ 0.88607595,  0.233     ,  1.33333333,  0.40740741,  3.11494737],
       ...,
       [-0.40506329,  0.5       , -0.33333333, -0.66666667, -0.42105263],
       [-0.4556962 ,  0.5       ,  0.        , -0.44444444,  0.42105263],
       [-0.40506329,  0.5       ,  0.        , -0.44444444,  0.42105263]])
```

Determining The Number Of Cluster

Inertia measuring between data



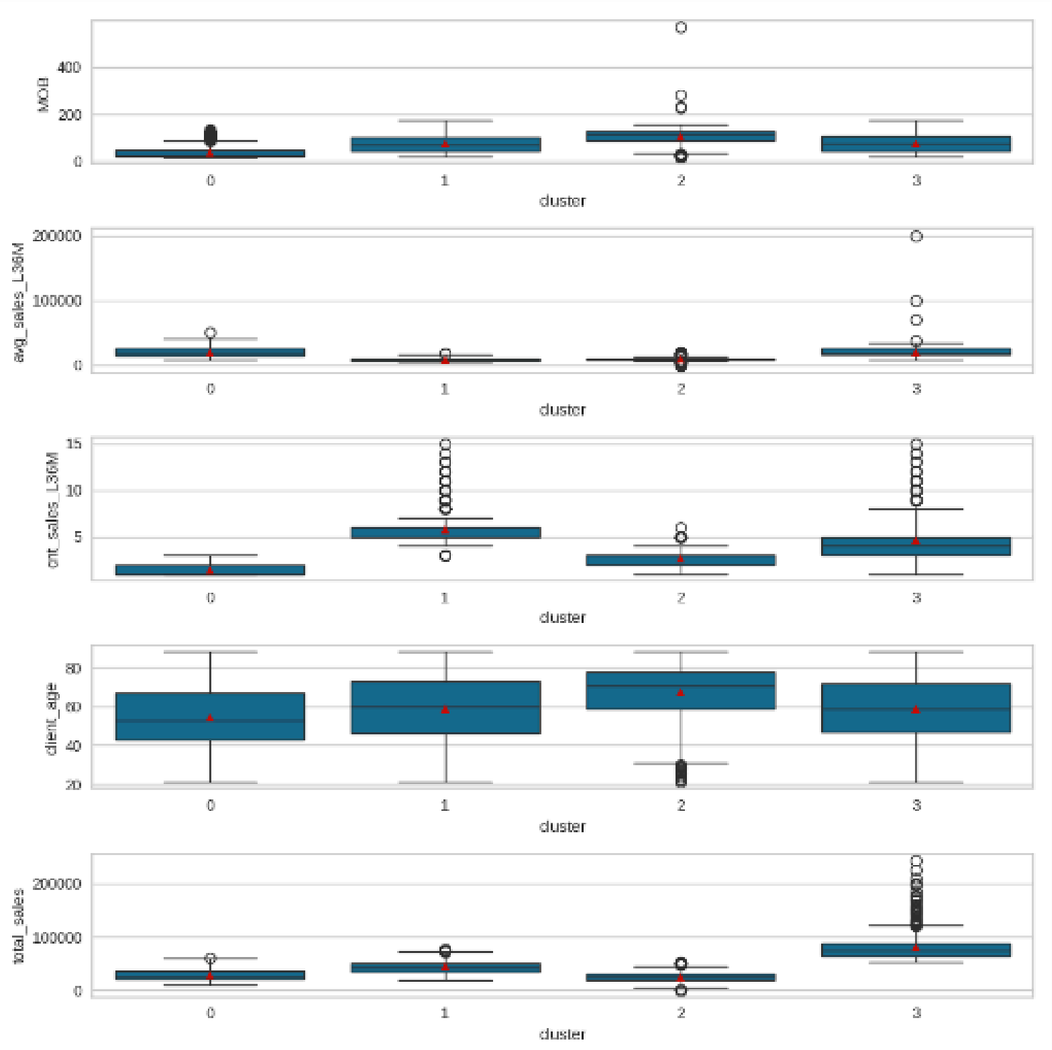
Silhouette analysis to validate clustering choice



***consideration of cluster # choice**

- Most balanced distribution of data
- Tolerable wrong-clustered data
- All clusters have proportions of data that past average silhouette score

Visualization Of The Cluster



- **Cluster 0 (Emerging Spenders)** : young & new customers, with low total sales, high average sales per customer, but low transaction

- **Cluster 1 (Frequent Flyers)** : existing customers, with medium total sales, low average sales per customer, but surprisingly very high transaction

- **Cluster 2 (Loyal Patrons)** : oldest customers (loyalty indicator), with lowest total sales, high average sales per customer, but very low on transaction

- **Cluster 3 (Big Spenders)** : existing customers, with highest total sales, medium average sales per customer, and high transaction

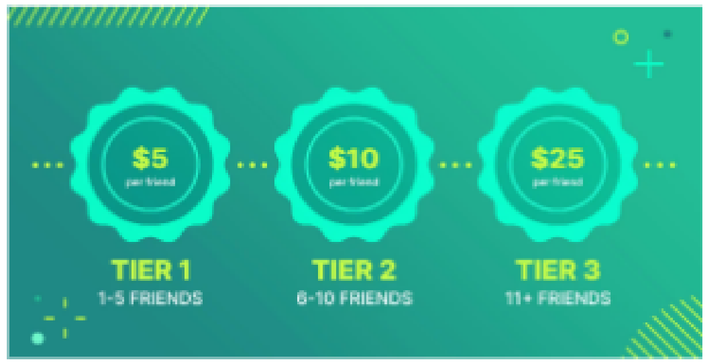
Business Recommendation

Cluster 0 (Emerging Spenders)

Implement a **loyalty program** that rewards customers for frequent purchases. Offer incentives such as discounts, freebies, or exclusive access to new products or events. This can encourage repeat purchases and increase transaction frequency.

Business Recommendation

Cluster 1 (Frequent Flyers)



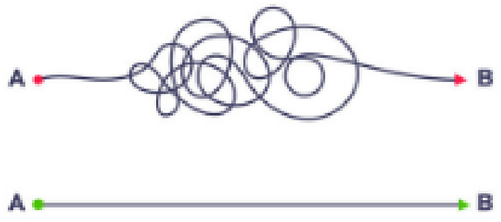
Incentivize Spending Tiers: Implement a rewards program that offers increasingly valuable benefits as customers reach higher spending tiers. Encourage customers to consolidate their smaller transactions into larger purchases by providing enhanced rewards or perks for reaching specific spending thresholds.

Other way is **Offer Cashback or Rewards on Larger Purchases**: Provide higher cashback percentages or more lucrative rewards for larger transactions. This can incentivize customers to use their credit cards for bigger purchases, thereby increasing the average sales value per transaction.

Business Recommendation

Cluster 2 (Loyal Patrons)

Reactivation Campaigns: Launch reactivation campaigns specifically targeting inactive or low-transacting customers. Send targeted emails or direct mailings with exclusive offers or bonuses to incentivize them to start using their credit cards again.



Simplify Rewards Redemption Process: Make it easy for customers to redeem their rewards and benefits earned through credit card usage. Streamline the redemption process and provide clear instructions on how to access and utilize their rewards, removing any barriers to frequent usage.

Business Recommendation

Cluster 3 (Big Spenders)

Exclusive Rewards and Benefits: Offer exclusive rewards, perks, and benefits tailored to the spending habits and preferences of high-value customers. This could include access to premium lounges at airports, concierge services, or higher cashback percentages on specific categories.

Personalized Service: Provide personalized customer service to high-value customers, including dedicated account managers or priority customer support lines. Tailor communications and offers based on their individual preferences and past behavior to enhance their experience.

Propensity Model Logistic Regression

Based on historical data, how many customers will use the pay later feature ?

Introduction

RevoBank conducted a successful pilot program for a pay later feature from February to April 2023, coupled with regular promotions, to encourage credit card usage. Following its positive reception, the team leader proposed expanding the initiative by targeting 1000 revoshop customers who have yet to utilize the pay later option. The strategy involves reaching out to them via phone and offering a 100 euro welcome bonus upon activation. To facilitate this expansion, discussions with Mr. Djoko, our manager, led to the development of a propensity model aimed at predicting which customers are most likely to embrace the pay later feature, ensuring a strategic and targeted approach to driving further adoption.

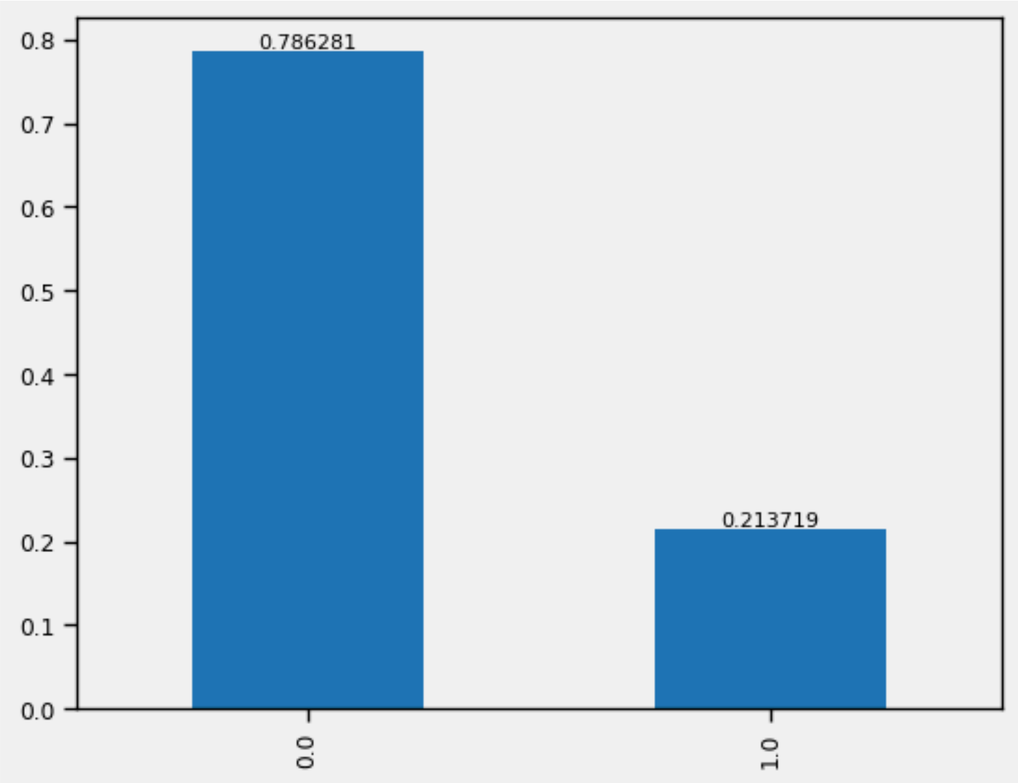
- Introduction :
  - RevoBank piloted a pay later feature from Feb-Apr 2023 alongside regular promotions.
  - Pay later converts e-commerce transactions into fixed installments with low interest rates.
  - Aim: Boost credit card usage.
- Pilot Success:
  - Positive results from the pilot program.
  - Request from team leader to assess project expansion.
- Expansion Plan:
  - Target: 1000 revoshop customers who never used the pay later feature.
  - Approach: Contact them by phone.
  - Incentive: Offer a 100 euro welcome bonus upon activation.
- Propensity Model Development:
  - Collaboration with Mr. Djoko, our manager.
  - Goal: Develop a predictive model to identify customers likely to use pay later.

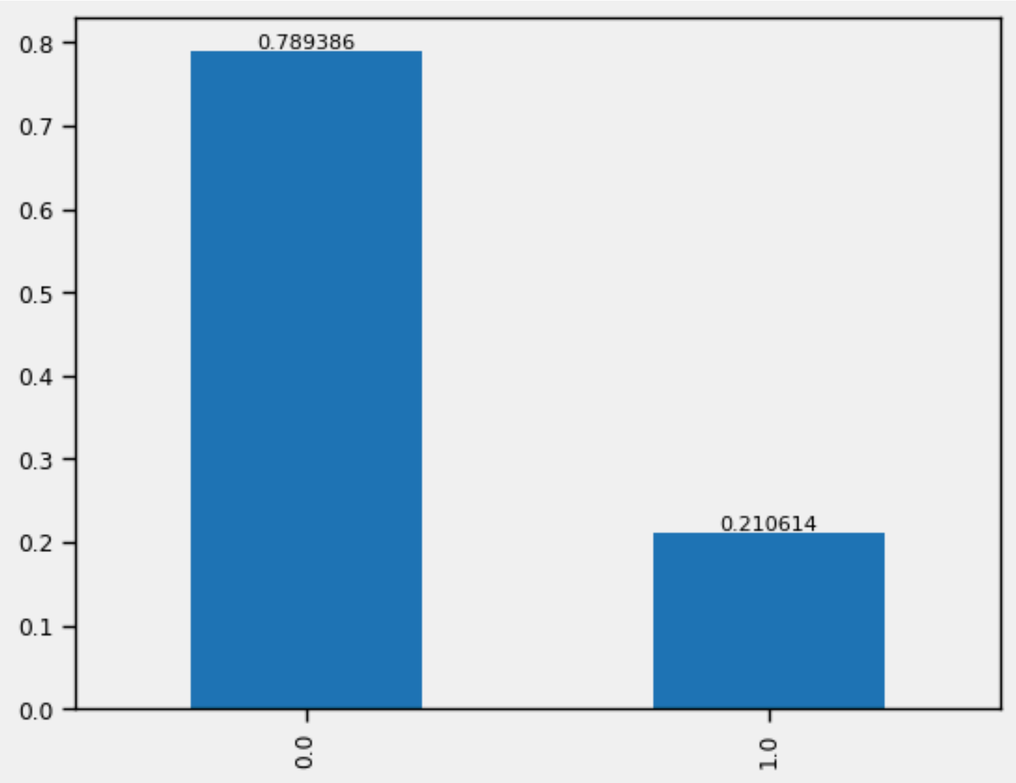
Data Cleaning & Preparation

Data Cleaning Steps	Before	After	Reasoning
Change Data Type	flag_female data is in integer	Change into object	Ensure functionality for python operation
	birth_date data is in object	Change into datetime	
	Xsell_count data is in float	Change into integer	
Impute Missing Values	Missing values in avg_sales_L36M	Imputed with "0"	missing values or "0" means no purchase for the last 36 months
	Missing values in avg_sales_L36M_promo	Imputed with "0"	
Removing Irrelevant Data	"XYZ" in account_activity_level	Removed	Not defined in the data dictionary
	"F" in customer_value_level	Removed	
	"-1000" value in avg_sales_L36M	Imputed with cnt_sales_L36M * last_sales	Could be calculated with reverse mathematics, because cnt_sales_L36M value is available which is "1"
Remove Duplicate Data	No duplicates are found	-	-
Eliminate Outliers	Some outliers found in avg_sales, cnt_sales, and other features	Removed	Outliers can directly affected our propensity model accuracy, so it is best to remove them

Training & Test Data Balance Checking

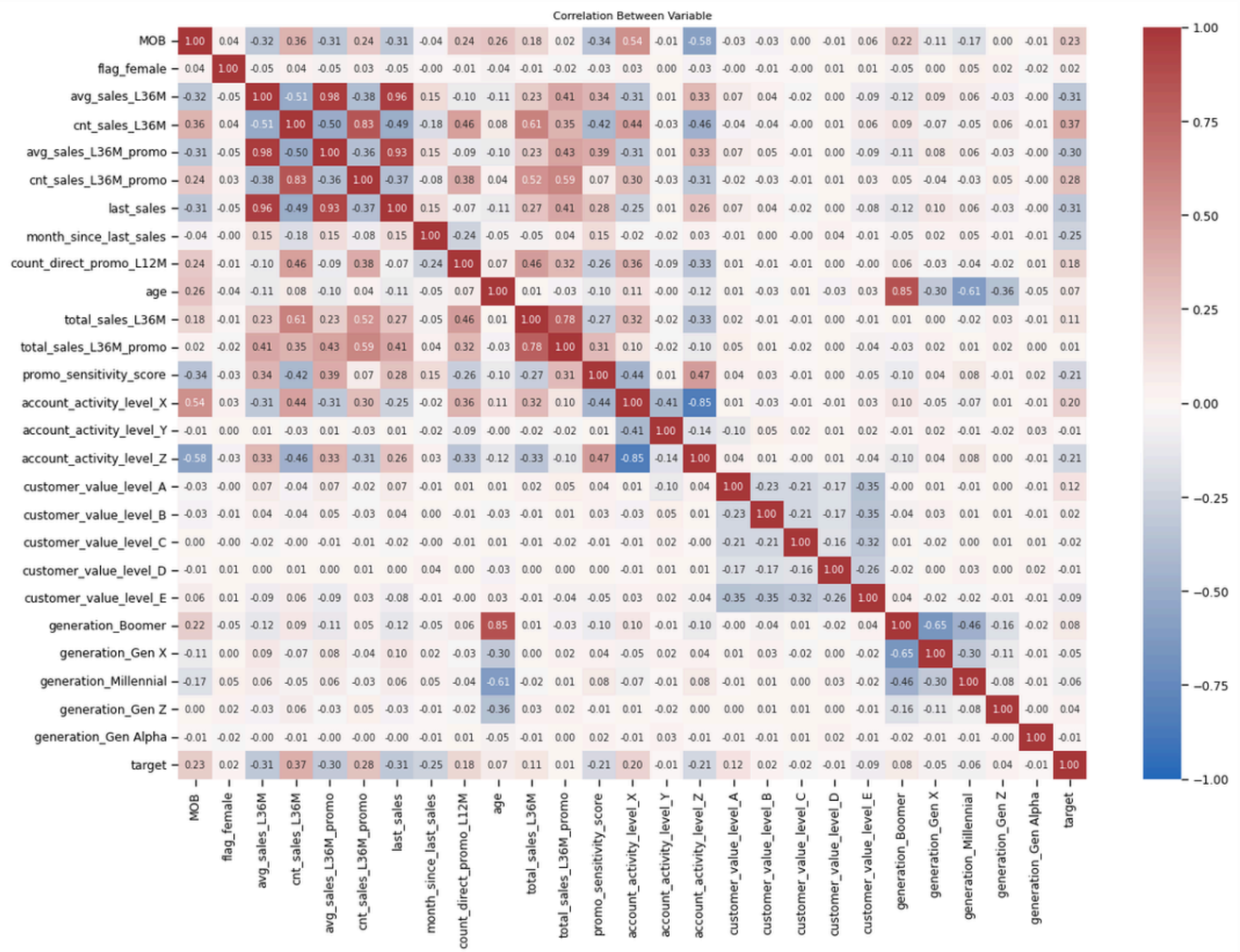
TRAINING → **Unbalanced** ← TEST





We will do “oversampling” where we make dummies data to fill up the unbalanced gap between prediction answers to avoid bias prediction from the maching learning

Features / Variable Correlation Checking



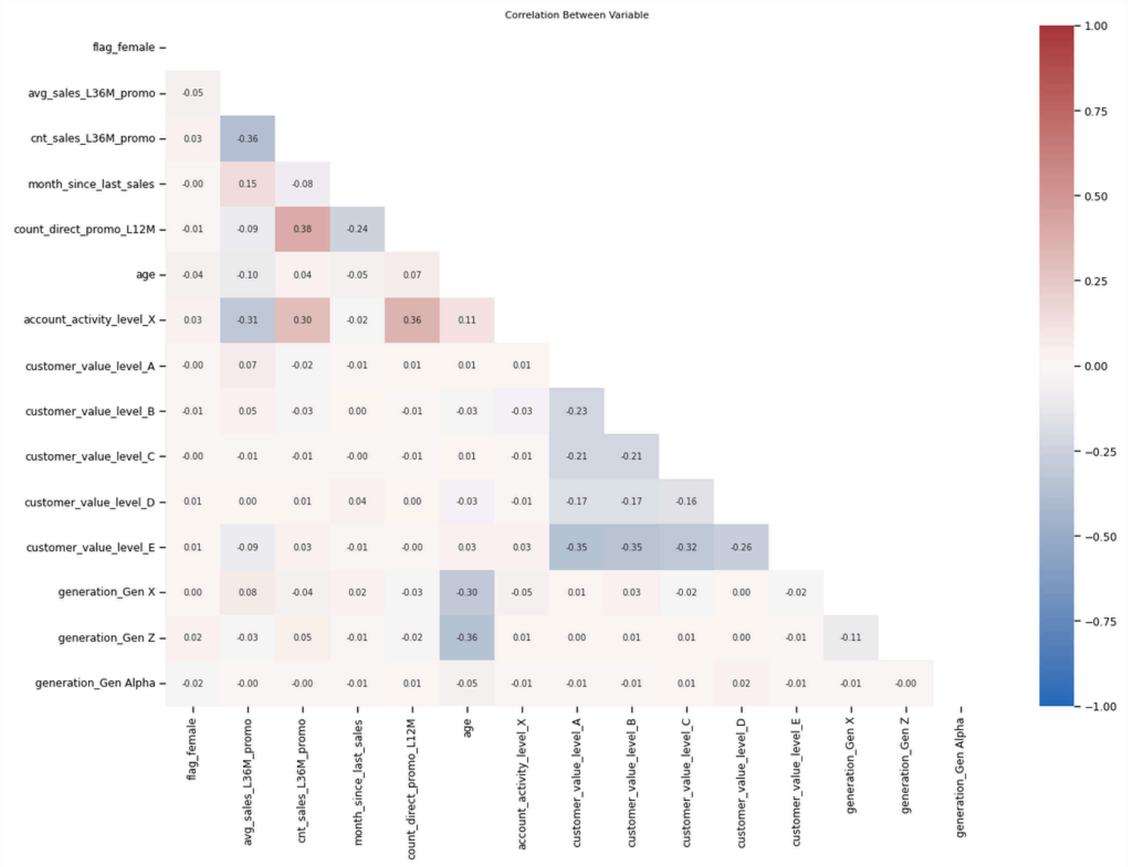
Features with high correlation (> 0.5 or < -0.5) with other features, and features with low correlation (0) with "target"

Removed features / variables :

- last_sales
- total_sales_L36M_promo
- avg_sales_L36M
- cnt_sales_L36M
- account_activity_level_Z
- MOB
- generation_Boomer
- account_activity_level_Y
- generation_Millenial
- total_sales_L36M
- promo_sensitivity_score

After features removal.

No more high or low correlations



account_id	prob_takers	prediction
100638029	0.534613	1.0
100786007	0.367895	0.0
100136859	0.298775	0.0
100142213	0.399690	0.0
100434021	0.133766	0.0
...	...	...
100749351	0.440467	0.0
100177744	0.607819	1.0
100685632	0.667666	1.0
100458549	0.667566	1.0
100023975	0.490466	0.0

8441 rows × 2 columns

Training Data :

Based on the model fit result, there are 3256 customers that will take the pay later features (38,57% out of 8441)

```
[ ] # Filter the DataFrame to include only rows where prediction is 1.0
    prediction_x = x_training2[x_training2['prediction'] == 1.0]

    # Count the number of rows in the filtered DataFrame
    count_prediction_x = len(prediction_x)

    print("Number of account_id with prediction = 1:", count_prediction_x)

    Number of account_id with prediction = 1: 3256
```

account_id	prob_takers	prediction
100158635	0.619895	1.0
100025376	0.689058	1.0
100266214	0.881349	1.0
100598155	0.188807	0.0
100895809	0.572642	1.0
...	...	...
101034497	0.790254	1.0
101005076	0.115742	0.0
100885175	0.686913	1.0
100202883	0.055353	0.0
100878266	0.554546	1.0

3618 rows × 2 columns

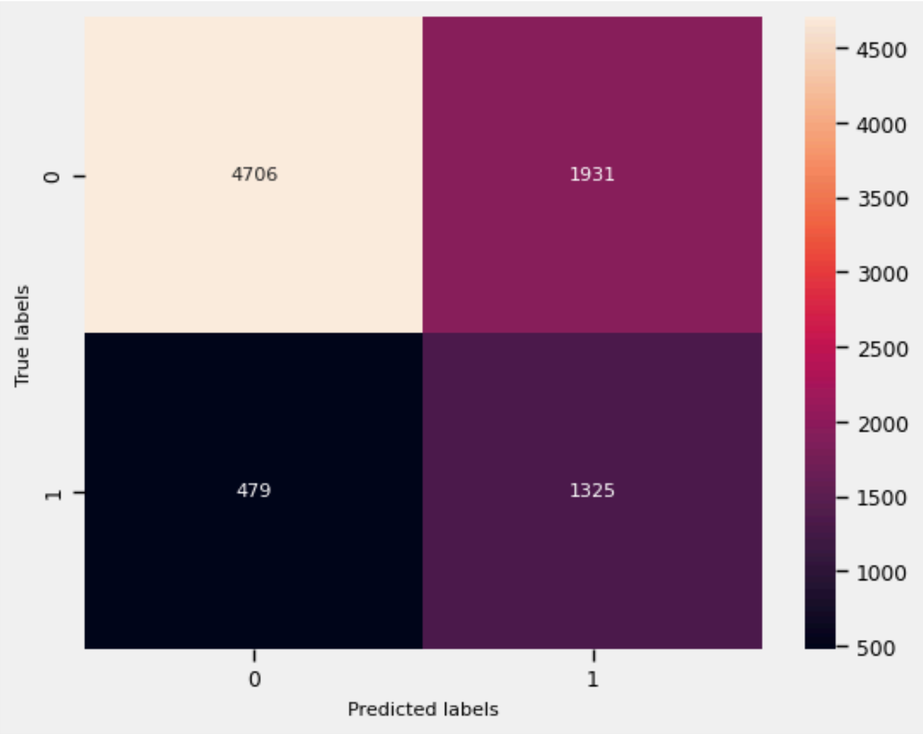
Test Data :

Based on the model fit result, there are 1377 customers that will take the pay later features (38,05% out of 3618)

```
[ ] # Filter the DataFrame to include only rows where prediction is 1.0
    prediction_1 = x_test2[x_test2['prediction'] == 1.0]

    # Count the number of rows in the filtered DataFrame
    count_prediction_1 = len(prediction_1)

    print("Number of account_id with prediction = 1:", count_prediction_1)

    Number of account_id with prediction = 1: 1377
```

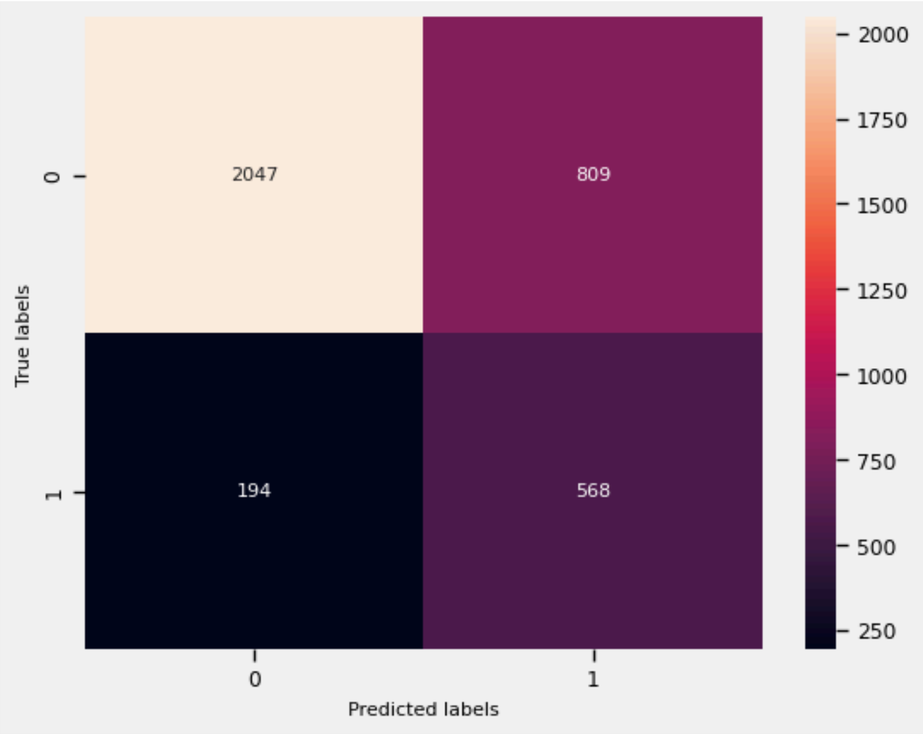
Accuracy & Confusion Matrix of Model Prediction

TRAINING



Accuracy : 71,4 %

TEST



Accuracy : 72,3 %

*Accuracy is based on the True Positive + True Negative compared to the overall prediction result

Classification Report

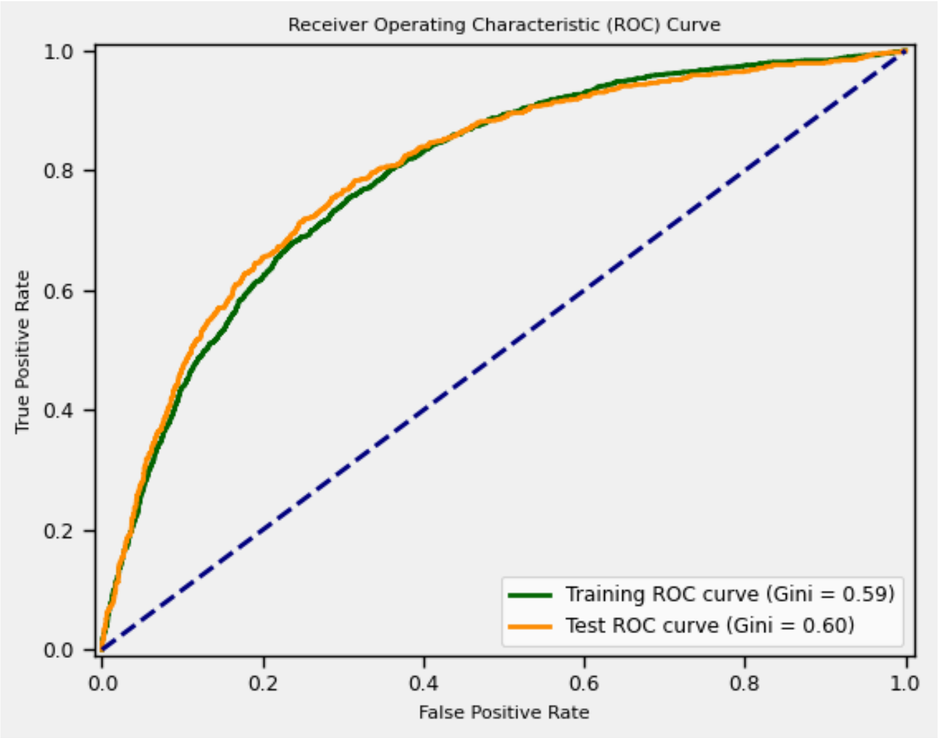
```
y_predict = model.predict(x_training_model)
print(classification_report(y_training, y_predict))
```

```
              precision    recall  f1-score   support

         0.0       0.91      0.71      0.80      6637
         1.0       0.41      0.73      0.52      1804

    accuracy                           0.71      8441
   macro avg       0.66      0.72      0.66      8441
weighted avg       0.80      0.71      0.74      8441
```

```
print(classification_report(y_test, model.predict(x_test_model)))
```

```
              precision    recall  f1-score   support

         0.0       0.91      0.72      0.80      2856
         1.0       0.41      0.75      0.53       762

    accuracy                           0.72      3618
   macro avg       0.66      0.73      0.67      3618
weighted avg       0.81      0.72      0.75      3618
```

- **Recall** which is used to predict True Positive is 73% for training data and 75% for test data
- **Precision** which used to determine False Positive is 41% for both training and test data
- **The statistic (f1) score** between test & training data does not deviate more than 5%, that mean that our model is fit



The ROC curve between training and test data does not deviate to each other, that means our model is fit

Decile Performance

binning	actual_#non_takers	actual_#takers	total_obs	prob_takers	%non_takers	%takers	cumm%_non_takers	cumm%_takers
(-9.999997168600002e-06, 0.12459]	816	29	845	0.034320	0.122947	0.016075	0.122947	0.016075
(0.12459, 0.18218]	816	28	844	0.033175	0.122947	0.015521	0.245894	0.031596
(0.18218, 0.24923]	800	44	844	0.052133	0.120536	0.024390	0.366431	0.055987
(0.24923, 0.32272]	764	81	845	0.095858	0.115112	0.044900	0.481543	0.100887
(0.32272, 0.40541]	733	110	843	0.130486	0.110441	0.060976	0.591984	0.161863
(0.40541, 0.48849]	687	157	844	0.186019	0.103511	0.087029	0.695495	0.248891
(0.48849, 0.56906]	633	211	844	0.250000	0.095374	0.116962	0.790869	0.365854
(0.56906, 0.66176]	581	263	844	0.311611	0.087540	0.145787	0.878409	0.511641
(0.66176, 0.76421]	458	386	844	0.457346	0.069007	0.213969	0.947416	0.725610
(0.76421, 0.99687]	349	495	844	0.586493	0.052584	0.274390	1.000000	1.000000

- We will observe the probability binning above **0,569** (top 3 deciles)
- The top 3 deciles of probability from the training datasets determined that these customers has a 54,8% chance of taking the pay later feature

KS SCORE : 44,66

- Based on the prediction model, there will be **1377 customer out of 3618 who will use the pay later feature**
- Prediction model result with training and test data performed consistenly, sugesting good generalization

*KS Score benchmark is 40, so our prediction model is considered fit

COST

- Cost of contacting customers
- Cost of contacting customers (dropped off)
- Cost of welcome bonus
- Total cost = **€ 12,990**

BENEFIT

- Expected Revenue : **€ 6.594,86**
- Total number of contacted customers : **150**
- Total number of dropped calls : **850**

B/C RATIO : 0.51

Project definition : contacting top 100 customers to pursue for using the pay later feature, by gifting welcome bonus

this project is not recommended to be executed as it will only return 51% of our investment, re-adjusting cost would be recommended to gain more profittability such as reducing welcome bonus cost

THANK YOU!