Individual Project Report – Data Processing

Python Programming for Data Science

Mehdi Zorkani

IÉSEG
SCHOOL OF MANAGEMENT
EMPOWERING CHANGEMAKERS FOR A BETTER SOCIETY

Table of Contents

I – Introduction

This document presents the technical documentation and preliminary analytical results of the proprietary database built from Berka Bank's financial dataset. This database serves as an essential foundation for developing predictive models aimed at improving customer risk assessment and optimizing product offering strategies.

The main object of this report is to add context to the data processing notebook used for the individual python project. We will give an overview and present the different steps of the project, describe the different variables of the basetable, clarify the methods used in data correction and analyze the most relevant variables via visualization.

The notebook already contains explanations regarding the meaning of the code, as most of the cells contain comments clarifying the meaning behind the code used and its relevancy to our project. Therefore, this report will not include a detailed description of every single step of the notebook.

II – Project Overview

Before delving into analysis, let's first give a quick and general overview of the steps taken to build the basetable.

A) Importing

The first step was to import the necessary libraries and the data files. We then had a quick look at our tables to familiarize ourselves with, looking at the shape, and checking if there are any missing values. They were none.

B) Time Windows

Afterwards, we had to deal with one of the most important steps of the project : Time Windows.

Indeed, the project had clear time windows as shown below :

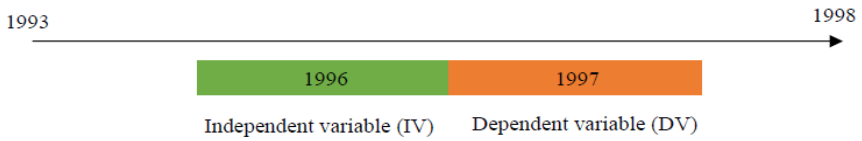


Figure 1: Illustration of the independent and dependent variables time window

The observation window contains all data up to the reference date of December 31, 1996. This ensures that all calculated characteristics, such as transaction amounts, balances, and recency measures, reflect the customer's status and behavior prior to the period being forecasted.

The forecast window (DV) measures target results exclusively during the following year, from January 1, 1997, to December 31, 1997.

The rigid separation of the two windows is technically necessary to prevent data leakage, which occurs when the training data contains information that would not actually be available at the time the prediction is made.

Thus, we had to filter the data frames that contained date info (trans and account) such as they respect the given time windows. The filtering was done for tables used for IV and DV after having converted the date to a datetime format.

C) Basetable and Features

Once the filtering was done, we started building our basetable. The first part was to create a core basetable with the level of granularity required, that is “client who is account owner”. This was done by filtering for Owner and merging the client, disp and account table. We were then able to explore other tables and add the relevant features as well as feature engineer others. Most feature engineered features were features from the transaction table as it is the largest table and the one that contains the most relevant variables given our project and context. Merging was done gradually each time we added features to our basetable.

D) Cleaning

The last step before exporting our basetable was the cleaning process. Here, we dropped the irrelevant columns, such as the id columns or the columns that were used for feature engineering only. Additionally, we handled missing values, filling them with the appropriate values. Finally, we ran a quick outlier analysis using the IQR method. However, after further reflection, we considered that in banking/demographic data, the 'outliers' represent natural variation as they reflect real population differences across districts (rural vs urban) and the transaction amounts and balance variations that naturally differ from small and large customers given their difference in income. Therefore, we decided to keep all data points as they represent valid real-world variations

E) Exporting

Afterwards, we were able to export our basetable, after indexing it by client, to a csv format in the correct data folder. A quick verification was done by importing this newly added basetable and reading the 5 five rows.

The last part of the notebook contains data description and visualization. This part will be explained in other dedicated parts of this report.

III – Variable Description

This first table describes each variable as well as its relevance in our context :

Variable	Description	Relevancy
client_id	Unique identifier of the client	Represents our granuality level
bank_district_id	ID of the district where the bank account is held	Links account to local socio-economic conditions
frequency	Type of statement frequency (weekly / monthly / after-transaction)	Indicates customer engagement level & possible transaction patterns
lor	Number of years the person has been a client	Longer relationships often correlate with loyalty/maturity
client_district_id	District where client lives	Enables matching socio-economic features affecting risk
age	Client age in 1996	Demographic factor influencing credit needs and risk tolerance
age_group	Categorical segmentation of ages	Same as age but allow for categorical modeling
Gender	Client gender	Often associated with different banking behaviors
region	Geographic region classification	Explains demographic or income-based behavior differences
nb_inhabitants	Population of client's district	Proxies economic scale and local development
nb_cities	Number of municipalities in district	Indicates urban vs rural environment
urban_inhabitants_ratio	Share of urban population	Urban clients show different spending patterns
avg_salary	Average income in district	Higher income areas correlate with lower risk
unemployment_rate_96	Local unemployment level in 1996	Indicates economic condition and potential default
nb_entrepreneurs_per_1000	Density of entrepreneurs	Measures economic dynamism
nb_crimes_96	Crime rate in district	Correlates with financial instability
unemployment_growth	Change in unemployment from 1995 to1996	Indicates worsening or improving local economy
crime_growth_rate	Change in crimefrom 1995 to1996	Can show that the district is less stable
population_per_city	District average city size	Indicates urbanization and economic activity
total_credit	Total credited money in 1996	High values reflect stable financial situation
avg_credit_amount	Average credited transaction amount	Another income proxy, shows a general image
total_withdrawal	Total withdrawals	Indicates spending / liquidity
avg_withdrawal_amount	Average withdrawal amount	Spending behavior metric
debit_count	Number of debit transactions (withdrawals)	Can show more engaged customers
credit_count	Number of credit transactions (deposits)	Can shows income flow consistency
trans_count	Total number of transactions	differnet number of transactions can show different behaviors

avg_trans_amount	Average transaction size	Reflects financial capacity
max_trans_amount	Maximum single transaction	Shows the maximum capacity of the client
min_trans_amount	Minimum transaction	Could reflect a less stable situation
std_trans_amount	Standard deviation of amounts	Showss volatility in financial behavior
net_Flow	Net money flow (credit − withdrawal)	Indicates whether customer saves or spends more
last_balance	Ending account balance at end of 1996	Key financial measure that represent the current situation
avg_balance	Average balance throughout 1996	Shows a more general indication of the balance
max_balance	Highest balance reached	Shows the maximum potential
min_balance	Lowest balance	Shows the possible riksy situation
std_balance	Balance volatility	Shows volatility of the financial situation
balance_range	Max − min balance	Another measure of financial fluctuations
first_balance	First recorded balance in 1996	Shows starting point of the client
balance_growth_pct	Balance growth from first to last	Shows financial improvement/worsening
median_balance	Middle balance value	More robust measure to outliers
neg_balance_count	Number of times balance < 0	Strong risk indicator showing danger zones
days_since_first_trans	Lag since first transaction	shows activity level
transaction_span_days	Duration between first and last transaction	Can show consistent users
avg_days_between_transactions	Transaction frequency metric	Higher frequency can show more engagedment
trans_freq_per_month	Average monthly transaction count	Shows transaction intensity
credits_per_month	Monthly credits	Proxy for income flows
debits_per_month	Monthly debits	Spending behavior per month
transaction_activity_ratio	Composite RFM-like metric	Captures financial dynamism
cash_withdrawal_count	Number of cash withdrawals	Shows reliance on cash
card_withdrawal_count	ATM withdrawals using card	Can show higher activity
cash_deposit_count	Number of cash deposits	Can show surprising financial patterns
transfer_in_count	Number of incoming transfers	Can show incom flow stability
transfer_out_count	Outgoing transfers	Can show income flow risk
total_cash_withdrawal	Total cash removed from account	Useful spending metric
total_card_withdrawal	Total card withdrawals	ATM usage indicator
total_cash_deposit	Total cash deposited	Can show suspicious behavior (if too high)
card_withdrawal_ratio	Proportion of withdrawals via card	Card usage intensity
cash_deposit_ratio	Share of deposits in cash	Indicates unusual banking behavior
card_type	Type of card (classic/gold/none)	Strong predictor for credit behavior
orders_count	Number of permanent orders	Can show more stable clients
total_order_amount	Total order payments	Payment behavior measure
avg_order_amount	Mean order payment	Shows pending regularity
max_order_amount	Maximum permanent order	Can show high financial responsibility
min_order_amount	Minimum permanent order	Can show low financial responsibility
insurance_order_count	Number of insurance-related orders	Can show risk-averse clients
household_order_count	House-related payments	Indicates family or housing status
leasing_order_count	Leasing repayment orders	Shows financial commitment
loan_order_count	Permanent loan-related payments	Shows credit usage
loans_1996_count	Number of loans taken in 1996	Reflects the current debt situation
good_loan_count	Number of loans repaid successfully	Shows positive reliability
bad_loan_count	Defaulted/bad loans	Shows engative reliability
total_loan_amount	Total amount of loans in 1996	Shows exposure to risk
loan_duration	Duration (in months) of loan	Different duration could reflect different level of reliability
monthly_payments	Monthly repayment amount	Can shows payment ability
target_loan_1997	Whether client took a loan in 1997	Prediction target
target_card_1997	Whether client requested/received card in 1997	Second prediction target

Moreover, after creating our basetable, we were able to create a summary table of each variable present in, including independent variables and dependent variables.

The table list the variable name, its data type, count(non-null count, as we do not have any missing value), the number of unique values, the value range (if it is numeric) or number of categories (if it is categorical), and a summary column with basic statistics (if it is numeric) or the distribution of the 3 most common categories (if it is categorical).

The table was generated inside the notebook and will be displayed below in multiple parts due to its large size :

Variable	Data_Type	count	Unique_Values	Value_Range	Summary
bank_district_id	object	2239	77	77 categories	{1: 279, 70: 76, 74: 70}
frequency	object	2239	3	3 categories	{'POPLATEK MESICNE': 2069, 'POPLATEK TYDNE': 122, 'POPLATEK PO OBRATU': 48}
lor	int32	2239	3	3 categories	{3: 1139, 1: 661, 2: 439}
client_district_id	object	2239	77	77 categories	{1: 272, 74: 79, 70: 70}
age	int64	2239	65	[14.00, 78.00]	Mean: 42.06, Median: 41.00, Std: 17.10
age_group	object	2239	7	7 categories	{20: 433, 30: 413, 40: 398}
Gender	object	2239	2	2 categories	{'M': 1155, 'F': 1084}
region	object	2239	8	8 categories	{'south Moravia': 401, 'north Moravia': 383, 'east Bohemia': 277}
nb_inhabitants	int64	2239	77	[42821.00, 1204953.00]	Mean: 266910.44, Median: 121947.00, Std: 356612.49
nb_cities	int64	2239	11	[1.00, 11.00]	Mean: 5.54, Median: 6.00, Std: 2.94
urban_inhabitants_ratio	float64	2239	70	[33.90, 100.00]	Mean: 69.21, Median: 63.10, Std: 19.77
avg_salary	int64	2239	76	[8110.00, 12541.00]	Mean: 9516.65, Median: 8991.00, Std: 1322.93
unemployment_rate_96	float64	2239	73	[0.43, 9.40]	Mean: 3.52, Median: 3.47, Std: 2.15
nb_entrepreneurs_per_1000	int64	2239	44	[81.00, 167.00]	Mean: 121.02, Median: 116.00, Std: 22.89
nb_crimes_96	int64	2239	76	[888.00, 99107.00]	Mean: 16151.52, Median: 3839.00, Std: 31119.68
unemployment_growth	float64	2239	77	[-6.07, 148.14]	Mean: 27.28, Median: 22.50, Std: 19.51
crime_growth_rate	float64	2239	77	[-53.68, 27.89]	Mean: 3.86, Median: 2.86, Std: 11.13
population_per_city	float64	2239	77	[7619.00, 1204953.00]	Mean: 185240.39, Median: 18195.57, Std: 387817.92
total_credit	float64	2239	2236	[0.00, 1040784.70]	Mean: 204344.92, Median: 158235.70, Std: 173457.51
avg_credit_amount	float64	2239	2238	[0.00, 27118.74]	Mean: 7504.01, Median: 6017.99, Std: 5635.66
total_withdrawal	float64	2239	2223	[900.00, 1020970.20]	Mean: 202252.79, Median: 157826.00, Std: 171223.93
avg_withdrawal_amount	float64	2239	2234	[375.71, 19639.45]	Mean: 4241.20, Median: 3251.13, Std: 3269.38
debit_count	int64	2239	94	[1.00, 101.00]	Mean: 46.77, Median: 45.00, Std: 15.38
credit_count	float64	2239	54	[0.00, 58.00]	Mean: 26.40, Median: 24.00, Std: 5.67
trans_count	int64	2239	114	[5.00, 142.00]	Mean: 73.17, Median: 72.00, Std: 17.70
avg_trans_amount	float64	2239	2238	[281.41, 21887.36]	Mean: 5302.30, Median: 4189.18, Std: 3952.27
max_trans_amount	float64	2239	1913	[1100.00, 86400.00]	Mean: 24942.45, Median: 22444.00, Std: 18249.71
min_trans_amount	float64	2239	116	[0.10, 400.00]	Mean: 18.27, Median: 14.60, Std: 21.57
std_trans_amount	float64	2239	2239	[414.73, 22664.26]	Mean: 6382.11, Median: 5486.70, Std: 4612.20
net_Flow	float64	2239	2234	[-82327.80, 96138.80]	Mean: 2095.84, Median: 1294.70, Std: 18094.29
last_balance	float64	2239	2234	[-10323.30, 114883.90]	Mean: 37733.85, Median: 33325.80, Std: 19869.45
avg_balance	float64	2239	2239	[-525.73, 90196.39]	Mean: 35860.65, Median: 32808.27, Std: 15776.91
max_balance	float64	2239	2236	[9700.00, 188635.10]	Mean: 61056.02, Median: 53507.20, Std: 34053.27
min_balance	float64	2239	2230	[-12016.90, 61484.30]	Mean: 18274.15, Median: 16749.90, Std: 9748.82
std_balance	float64	2239	2239	[367.42, 37483.56]	Mean: 9796.42, Median: 7899.37, Std: 7218.20
balance_range	float64	2239	2236	[900.00, 168445.40]	Mean: 42781.86, Median: 34838.10, Std: 32463.49
first_balance	float64	2239	2234	[-5813.60, 117552.90]	Mean: 35202.62, Median: 31396.60, Std: 17909.29
balance_growth_pct	float64	2239	2104	[-10823.65, 19042.80]	Mean: 24.03, Median: 6.52, Std: 488.97
median_balance	float64	2239	2236	[-2988.20, 92092.30]	Mean: 34732.26, Median: 31646.40, Std: 15319.26
neg_balance_count	float64	2239	15	[0.00, 23.00]	Mean: 0.12, Median: 0.00, Std: 1.07
days_since_first_trans	int64	2239	32	[314.00, 365.00]	Mean: 360.75, Median: 361.00, Std: 4.47

transaction_span_days	int64	2239	38	[210.00, 365.00]	Mean: 360.39, Median: 361.00, Std: 7.36
avg_days_between_transactions	float64	2239	696	[2.57, 82.50]	Mean: 5.54, Median: 5.11, Std: 3.30
trans_freq_per_month	float64	2239	114	[0.42, 11.83]	Mean: 6.10, Median: 6.00, Std: 1.47
credits_per_month	float64	2239	54	[0.00, 4.83]	Mean: 2.20, Median: 2.00, Std: 0.47
debits_per_month	float64	2239	94	[0.08, 8.42]	Mean: 3.90, Median: 3.75, Std: 1.28
transaction_activity_ratio	float64	2239	2239	[71.55, 2061754.90]	Mean: 406366.53, Median: 315762.80, Std: 344468.56
cash_withdrawal_count	int64	2239	63	[1.00, 66.00]	Mean: 31.35, Median: 31.00, Std: 8.75
card_withdrawal_count	float64	2239	18	[0.00, 20.00]	Mean: 0.41, Median: 0.00, Std: 1.86
cash_deposit_count	float64	2239	35	[0.00, 34.00]	Mean: 9.86, Median: 12.00, Std: 8.01
transfer_in_count	float64	2239	2	[0.00, 12.00]	Mean: 4.26, Median: 0.00, Std: 5.74
transfer_out_count	float64	2239	49	[0.00, 60.00]	Mean: 15.01, Median: 12.00, Std: 12.60
total_cash_withdrawal	float64	2239	2139	[175.20, 991818.00]	Mean: 153428.76, Median: 95575.20, Std: 162744.10
total_card_withdrawal	float64	2239	123	[0.00, 44800.00]	Mean: 903.89, Median: 0.00, Std: 4061.91
total_cash_deposit	float64	2239	1551	[0.00, 1036762.00]	Mean: 152140.83, Median: 110680.00, Std: 174706.71
card_withdrawal_ratio	float64	2239	129	[0.00, 0.52]	Mean: 0.01, Median: 0.00, Std: 0.04
cash_deposit_ratio	float64	2239	77	[0.00, 1.00]	Mean: 0.36, Median: 0.50, Std: 0.26
card_type	object	2239	4	4 categories	{'None': 2133, 'classic': 75, 'junior': 26}
orders_count	float64	2239	6	[0.00, 5.00]	Mean: 1.43, Median: 1.00, Std: 1.11
total_order_amount	float64	2239	1662	[0.00, 21322.20]	Mean: 4571.18, Median: 3493.00, Std: 3879.58
avg_order_amount	float64	2239	1711	[0.00, 14658.00]	Mean: 3016.33, Median: 2559.00, Std: 2473.52
max_order_amount	float64	2239	1643	[0.00, 14658.00]	Mean: 3689.35, Median: 3159.00, Std: 2866.60
min_order_amount	float64	2239	1603	[0.00, 14658.00]	Mean: 2449.81, Median: 1996.00, Std: 2527.29
insurance_order_count	float64	2239	2	[0.00, 1.00]	Mean: 0.12, Median: 0.00, Std: 0.32
household_order_count	float64	2239	3	[0.00, 2.00]	Mean: 0.77, Median: 1.00, Std: 0.49
leasing_order_count	float64	2239	2	[0.00, 1.00]	Mean: 0.08, Median: 0.00, Std: 0.27
loan_order_count	float64	2239	2	[0.00, 1.00]	Mean: 0.15, Median: 0.00, Std: 0.36
loans_1996_count	float64	2239	2	[0.00, 1.00]	Mean: 0.04, Median: 0.00, Std: 0.19
good_loan_count	float64	2239	2	[0.00, 1.00]	Mean: 0.03, Median: 0.00, Std: 0.17
bad_loan_count	float64	2239	2	[0.00, 1.00]	Mean: 0.01, Median: 0.00, Std: 0.08
total_loan_amount	float64	2239	83	[0.00, 444864.00]	Mean: 6255.70, Median: 0.00, Std: 38727.56
loan_duration	float64	2239	6	[0.00, 60.00]	Mean: 1.53, Median: 0.00, Std: 8.34
monthly_payments	float64	2239	80	[0.00, 9268.00]	Mean: 153.49, Median: 0.00, Std: 879.35
target_loan_1997	object	2239	2	2 categories	{0: 2208, 1: 31}
target_card_1997	object	2239	2	2 categories	{0: 2119, 1: 120}

IV – Data Correction and Transformation

This section details the methodologies employed to integrate, correct, and transform some of the raw relational data into the basetable structure.

A) Data Loading

The transactional dataset, like other datasets, was loaded using the pandas read_csv() function. However, we used specific parameters to optimize data integrity. The low_memory=False

parameter was used to prevent pandas from inferring column data types in blocks, ensuring consistent data type assignment throughout the dataset.

```
# low_memory=False prevents pandas from guessing column types in chunks (better for large file)
trans = pd.read_csv('../../data/raw/data_berka/trans.asc', sep=';', low_memory=False)
```

B) Dates

As mentioned earlier in the overview, an important part of the data preparation was to deal with dates.

First, the dates columns that were present on the different tables (apart from birth_date as we will see) were objects. Hence, we needed to cast them as datetime in order to perform calculations.

```
# We created a function to convert the date column into a more suitable date format
def convert_date(series):
    date = pd.to_datetime(series, format='%y%m%d', errors='coerce')
    return date
```

0.0s

```
account['date'] = convert_date(account['date'])

account.head()
```

0.0s

	account_id	district_id	frequency	date
0	576	55	POPLATEK MESICNE	1993-01-01
1	3818	74	POPLATEK MESICNE	1993-01-01
2	704	55	POPLATEK MESICNE	1993-01-01
3	2378	16	POPLATEK MESICNE	1993-01-01
4	2632	24	POPLATEK MESICNE	1993-01-02

Thus, we created a function that converts into the desired date format and passed it into the relevant tables.

Here is an example of date casting for the account table.

For the date column in the card table and the birth_date in the client column, additional steps were required. Indeed, the number is in the card table followed the YYMMDD form, and YYMMDD (men) and YYMM+50DD (women).

```
# Before applying the function, we will first only select the date part (we exclude the timestamp)
card['issued'] = card['issued'].astype(str).str[:6]
card['issued'] = convert_date(card['issued'])

card.head()
```

0.0s

	card_id	disp_id	type	issued
0	1005	9285	classic	1993-11-07
1	104	588	classic	1994-01-19
2	747	4915	classic	1994-02-05
3	70	439	classic	1994-02-08
4	577	3687	classic	1994-02-15

For example, on the card table, we cast the date as a string, then accesses the date part with string slicing (as it also contains timestamp). Then the function was applied.

C) Gender and Age

For these two features, the calculation relies on the structural encoding of the birth_number which was presented in the previous section.

For the age we simply needed to extract the year using basic arithmetic, as show below, and subtracting. The '//' allows us to only keep the quotient, which represents the year.

```
# We create the Age varaible
df.loc[:, 'age'] = 1996 - (df.loc[:, 'birth_number'] // 10000 + 1900)

df.head()
```
✓ 0.0s

For the gender, we extracted the digits of the month and applied conditional logic: months greater than 12 indicate a female ('F'), while months less than or equal to 12 indicate a male ("M"). The transformation used the pandas apply() method with lambda functions.

```
# Every birth number where the month is > 12 is recorded as female 'F', else male 'M'
df.loc[:, 'Gender'] = df.loc[:, 'birth_number'].apply(lambda x : 'F' if int(str(x)[2:4]) > 12 else 'M' )

df.head()
```
✓ 0.0s

D) Renaming

Several times in the course of the project we had to rename our columns for the sake of clarity.

The general way of renaming was to use the .rename method and select the columns to rename as well as the new names.

```
# We first start by renamming the different columns with their appropriate names
district_features = district.rename(columns = {'A1' : 'district_id',
                                               'A2' : 'district_name',
                                               'A3' : 'region',
                                               'A4' : 'nb_inhabitants',
                                               'A5' : 'municipalities_0_499',
                                               'A6' : 'municipalities_500_1999',
                                               'A7' : 'municipalities_2000_9999',
                                               'A8' : 'municipalities_10000_plus',
                                               'A9' : 'nb_cities',
                                               'A10' : 'urban_inhabitants_ratio',
                                               'A11' : 'avg_salary',
                                               'A12' : 'unemployment_rate_95',
                                               'A13' : 'unemployment_rate_96',
                                               'A14' : 'nb_entrepreneurs_per_1000',
                                               'A15' : 'nb_crimes_95',
                                               'A16' : 'nb_crimes_96'})

# We verify if the columns were indeed renamed
district_features.columns
```

Here is an example with of district table that had all of its columns in a generic name and were therefore all renamed.

At the end, we check the columns in order to verify that the changes were made correctly.

E) Invalid Values

Invalid values are values that do not represent any information about the given variable and are likely cause by a human error or an error of the system. The presence of invalid values restricts our ability to feature engineer as it will raise errors.

During our data preparation, two variables (unemployment_rate_95 and nb_crimes_95) were concerned by this problem and were handled appropriately. Invalid values represented by '?' symbols were first converted to NaN, then transformed to numeric format using pandas' to_numeric() function with error coercion.

```
# Before casting we should handle the invalid values that block the casting
district_features['unemployment_rate_95'] = district_features['unemployment_rate_95'].replace('?', np.nan)
district_features['nb_crimes_95'] = district_features['nb_crimes_95'].replace('?', np.nan)

# We convert to numeric first
district_features['unemployment_rate_95'] = pd.to_numeric(district_features['unemployment_rate_95'], errors='coerce')
district_features['nb_crimes_95'] = pd.to_numeric(district_features['nb_crimes_95'], errors='coerce')

# We are going to use these variables for feature engineering
# Hence can't keep them as NaN or 0 as it does not make sense (for a growth calculation for example)
# We are therefore going to replace them by the median (handles better outliers)
district_features['unemployment_rate_95'] = district_features['unemployment_rate_95'].fillna(district_features['unemployment_rate_95'].median())
district_features['nb_crimes_95'] = district_features['nb_crimes_95'].fillna(district_features['nb_crimes_95'].median())
```

F) Missing Values

Following feature selection, feature engineering and transformation, our basetable was left with a very high number of missing values. The table below shows the columns with the most Nan's.

bad_loan_count	2225
neg_balance_count	2180
good_loan_count	2169
loans_1996_count	2155
total_loan_amount	2155
monthly_payments	2155
loan_duration	2155
card_type	2133
total_card_withdrawal	2068
card_withdrawal_count	2068
card_withdrawal_ratio	2068
leasing_order_count	2059
insurance_order_count	1972
loan_order_count	1908
transfer_in_count	1444
total_cash_deposit	623
cash_deposit_count	623
cash_deposit_ratio	622
household_order_count	578
transfer_out_count	485
orders_count	380
max_order_amount	380
total_order_amount	380
min_order_amount	380
avg_order_amount	380

The missing values were identified and handled with the following logic:

For categorical variables, in our case only the card_type column, missing values indicated absence of a card in 1996 and were filled with 'None' to represent a meaningful category rather than missing data.

For count columns (such as trans_counts), Nan's were replaced with 0, as the absence of a count logically means zero occurrences of that event.

For total columns (such as total_credit), Nan's were replaced with 0, representing no activity in that category.

For columns with averages, min and max, Nan's were replaced with 0, as these statistics cannot be calculated without transactions.

For ratio features (such as card_withdrawal_ratio), Nan's occurred when denominators were zero. These were replaced with 0 to indicate no activity rather than missing information.

For the monthly_payments feature, missing values indicated clients without loans in 1996, replaced and were thus replaced by 0.Finally, for net_Flow, missing values were replaced with 0 to indicate balanced or no cash flow. This approach ensured that 0 represented "absence of activity" rather than "unknown," preserving the logic and the meaning of the data.

G) Data Aggregation

As the desired basetable has a very specific granularity that needs to be respected, aggregation was necessary during the feature engineering process.

After having divided our dataframes by timeline (as shown below) we started the feature engineering process.

```
# We define our time windows as per project requirments

# For the independent variables
trans96 = trans[trans['date'].dt.year == 1996]
card96 = card[card['issued'].dt.year == 1996]
loan96 = loan[loan['date'].dt.year == 1996]
account96 = account[account['date'].dt.year < 1996]

# For the dependent variables
card97 = card[card['issued'].dt.year == 1997]
loan97 = loan[loan['date'].dt.year == 1997]
```

During the feature engineering process, aggregation methods were used to obtain the desired result while preserving the granularity. Here are some of the aggregation methods used :

- Counts: Used .size() or .count() to count occurrences (ex : trans_count).
- Sums: Used .sum() for amount totals (ex : total_credit)
- Averages: Used .mean() for averaging values (ex : avg_salary).
- Extremes: Used .min() and .max() to capture boundaries (ex : min_balance)
- Temporal: Used .first() and .last() for time related features (ex : first_balance).

Therefore, since accounts could have multiple transactions, orders, or cards, groupby operations on account_id were systematically applied before merging into the basetable, preserving the desired granularity.

V – Graphical Analysis

A) Top 10 Feature Selection

After serious reflection, we were able to select the 10 most important features and analyze them with the appropriate visualizations. Before looking at the graphs, let's delve quickly into the reasons behind our selection.

Demographics

- Age : a key factor in financial behavior and creditworthiness
- Gender : can influence preferences for banking products and financial decisions.

Account feature

Here, we have only selected the length of the relationship. This feature can show that longer banking relationships are corelated with customer loyalty and stability, two factors that banks value when granting new products.

Financial behavior

- Average account balance : reflects financial capacity and stability
- Number of transactions: indicates account activity and engagement
- Total credit amount: Represents income/cash inflows

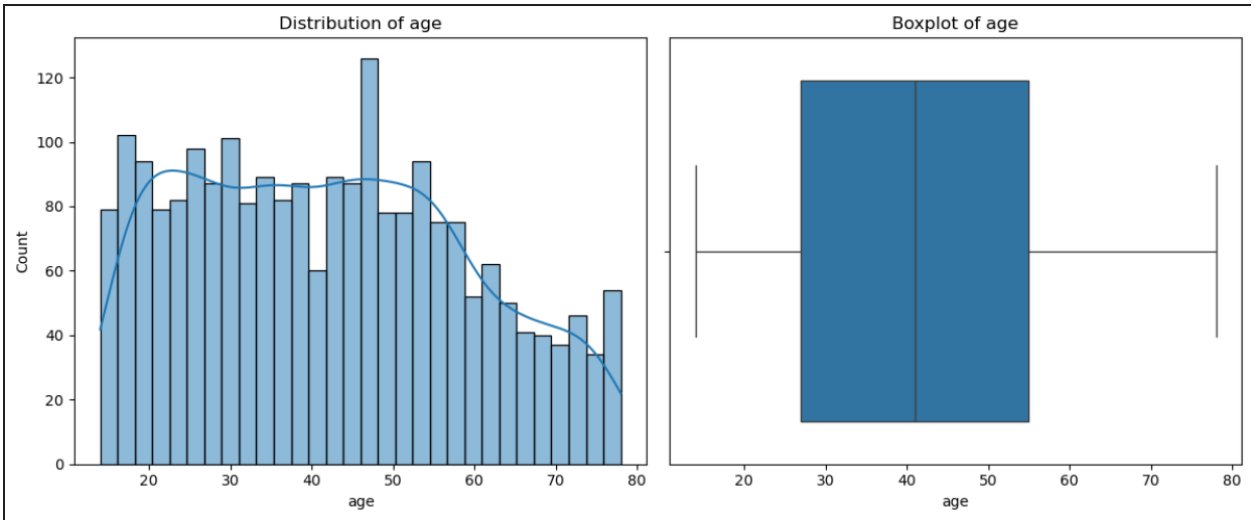
Total withdrawal amount: Reflect spending habits

Socio-economic features

- Average salary: Reflects local economic conditions.
- Unemployment rate in 96: The rate in the district reflects economic health.
- Crime growth rate: Crime trends can indicate the stability and the risk of the district.

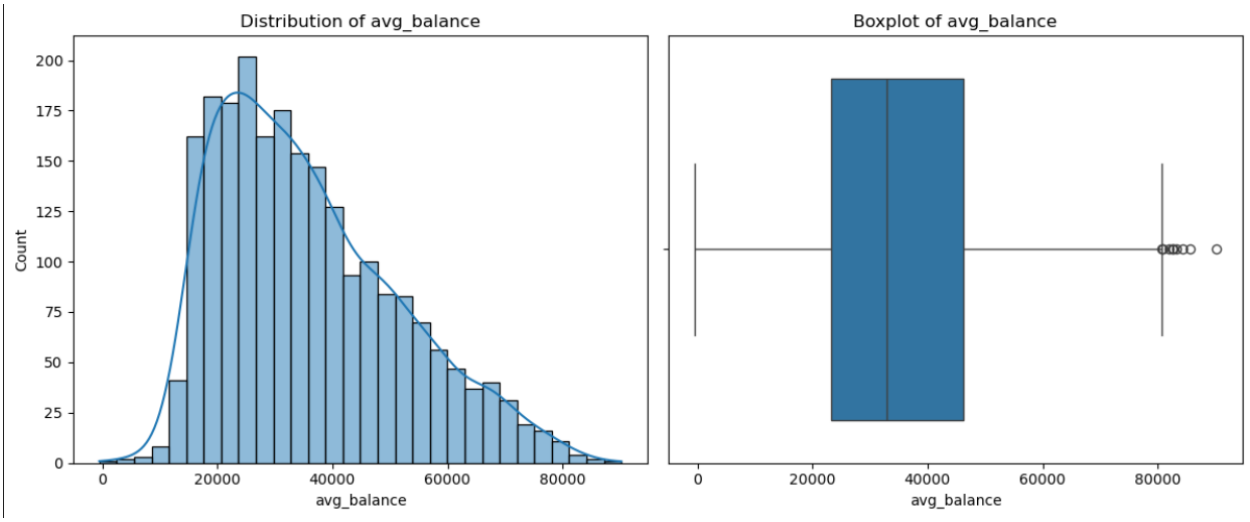
B) Graphical analysis – Top 10 Features

Age



The age distribution shows a relatively even spread across the age groups, with a notable peak around 46-48 years old. The box plot reveals no significant outliers, with a median around 41. The distribution suggests a mature customer base, which is typical for banking products such as loans and credit cards.

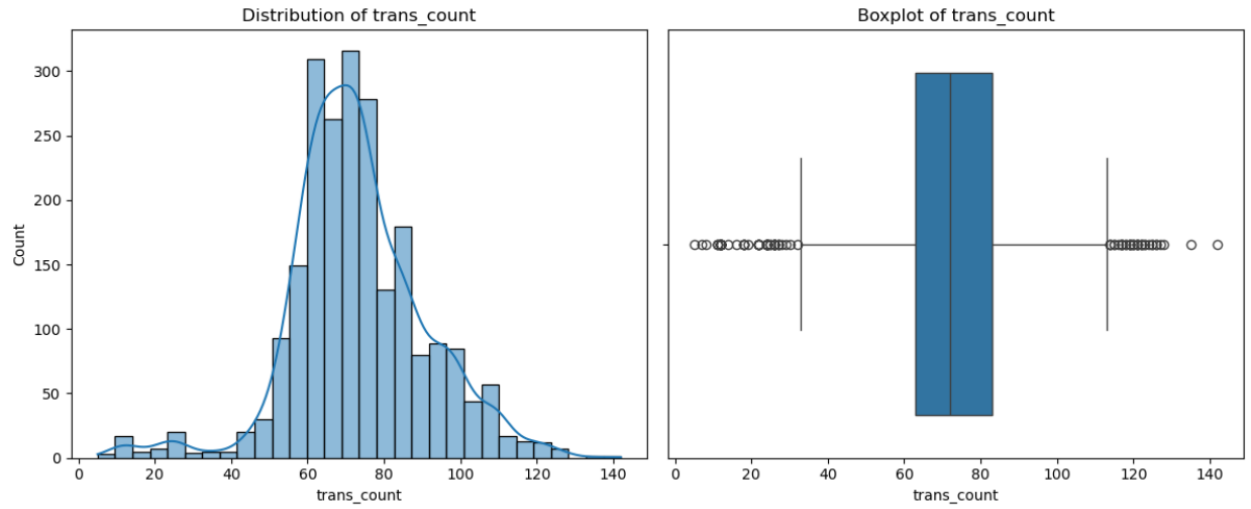
Average balance



The distribution of the average balance is rightly skewed, with most customers having a balance between 10,000 and 40,000. The box plot shows numerous outliers above 80,000 , indicating the presence of wealthy customers. The median (32,808) is slightly lower than the average (35,860),

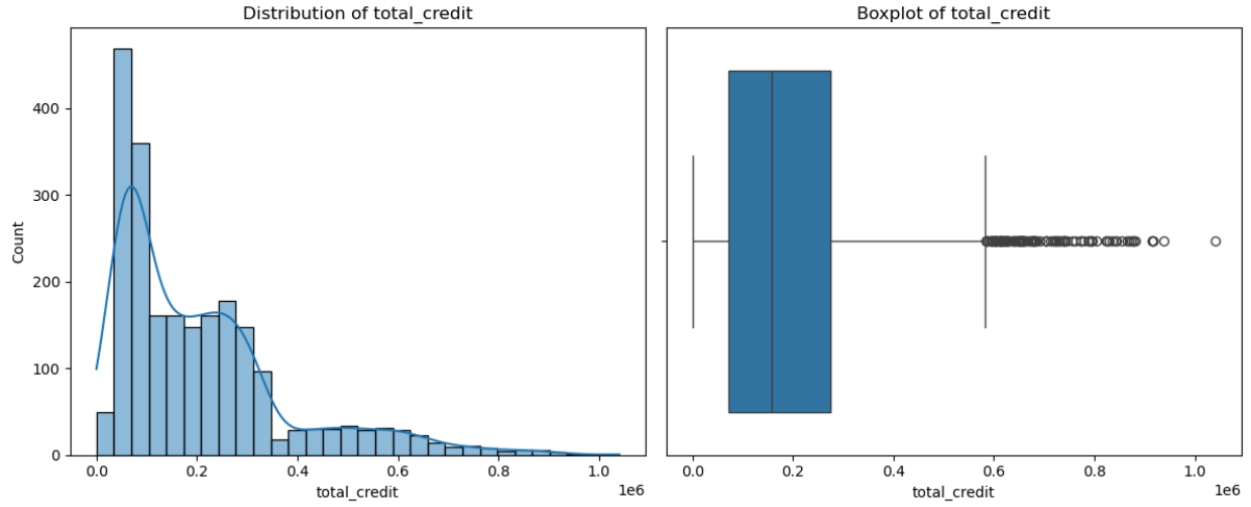
confirming the positive skew. This distribution is also typical in the banking sector, where most customers maintain modest balances while a minority hold significantly larger amounts.

Transaction count



The number of transactions follows an approximately normal distribution centered around 60 to 80 transactions per year in 1996. The box plot shows a median of 72 transactions, with an IQR extending from approximately 37 to 115. Some outliers exist beyond these limits, representing very active and very inactive accounts. This pattern suggests that most customers have regular but moderate activity on their accounts, which is important for assessing their commitment.

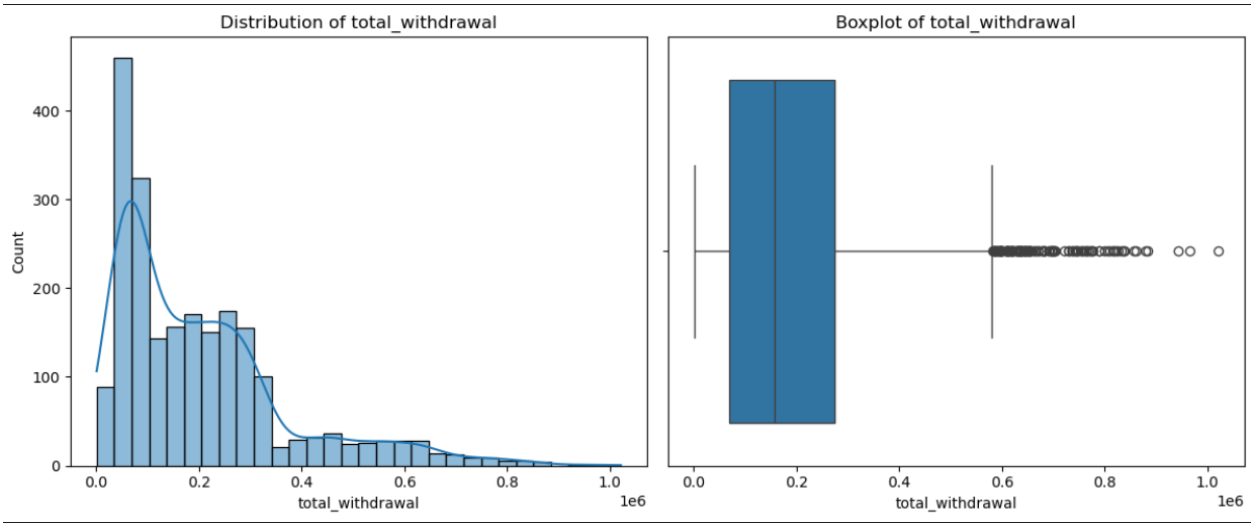
Total credit



Total credit shows a strong right skewness, with the majority of customers having less than 200,000 in credit in 1996. The box plot reveals numerous outliers above 600,000, representing high-income

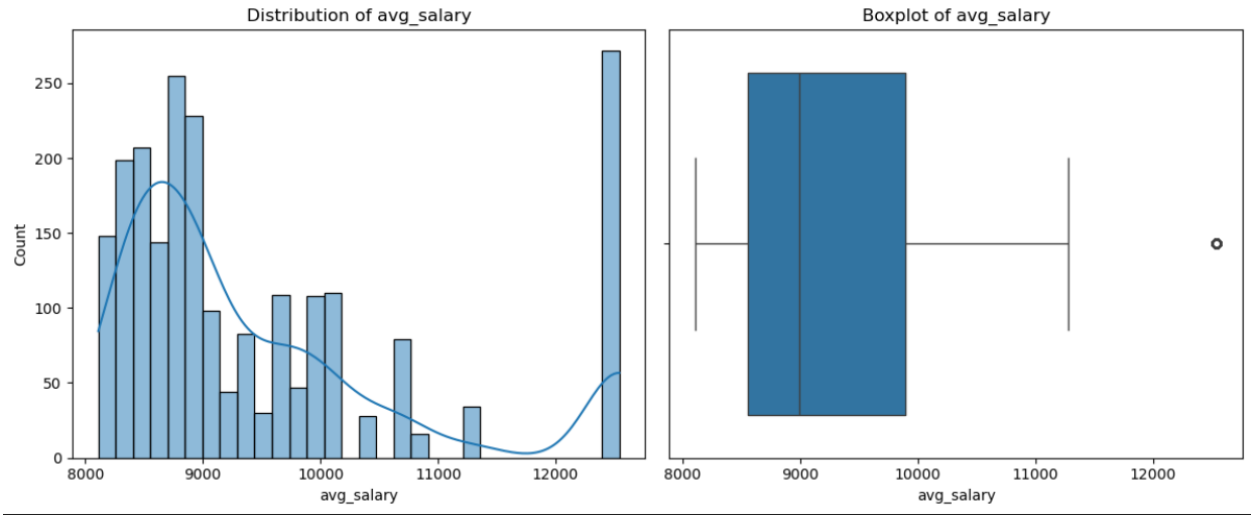
customers. The median (158,235) is significant lower than the mean (204,344) which indicates more credit among a small part of clients.

Total withdrawal



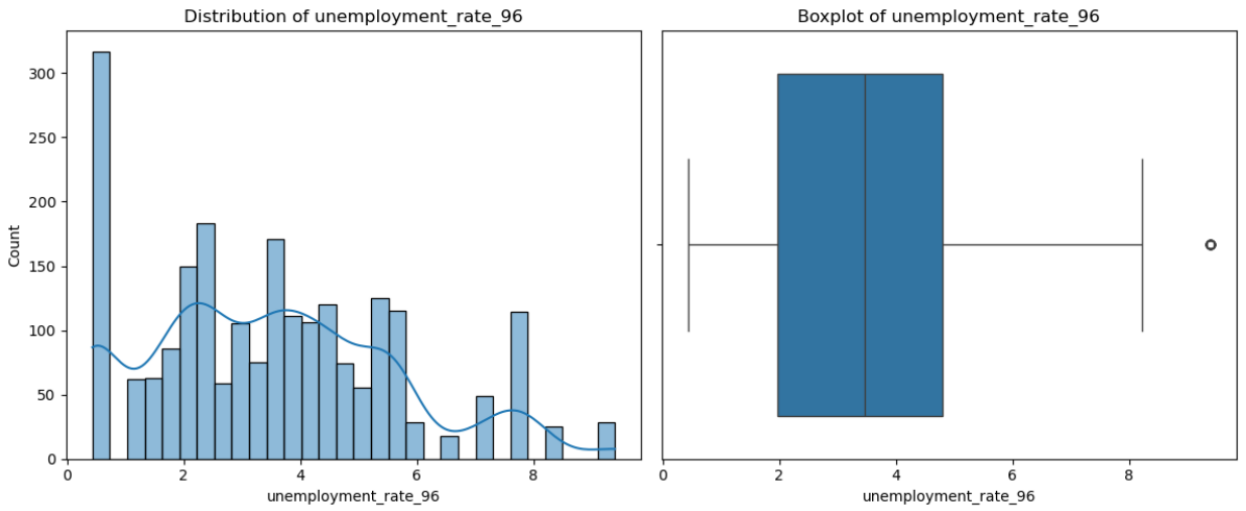
The total withdrawal is similar to the total credit with a heavy right skewness. Most customers withdrew less than 200,000 in 1996. The box plot shows significant outliers above 600,000. The similar trend to that of total credit suggests balanced cash flows for most customers, meaning that in general the money inflow equals money outflow. High withdrawal amounts may indicate high spending power which is relevant for credit assessment.

Average salary



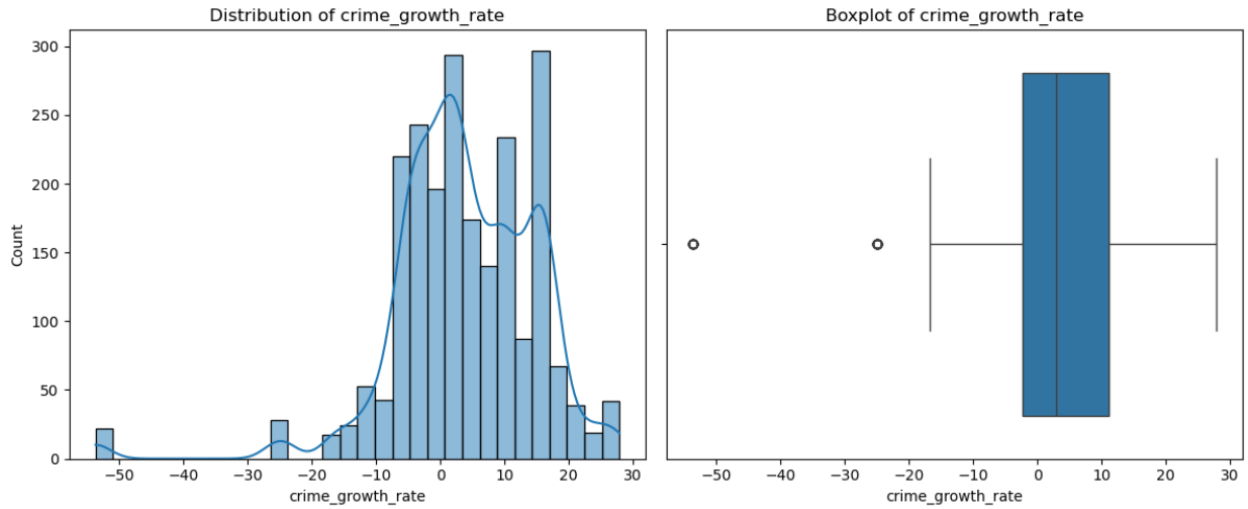
The average salary has two peaks, one around 8,500 and the other around 125,000. This is called bimodularity, which, in our context, could reflect two different economic regions, one with low-income rural areas and one with higher-income urban areas. We could conclude from this analysis that customers in higher-wage districts have a better chance of obtaining a loan, regardless of their individual characteristics.

Unemployment rate 1996



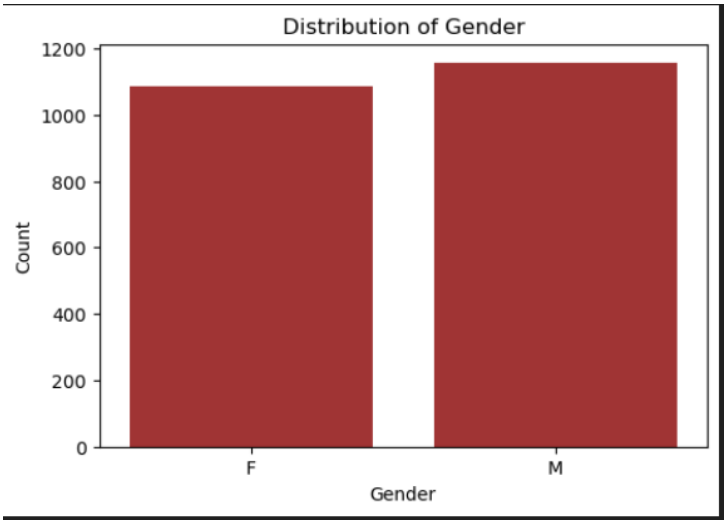
The unemployment rate is also right skewed, with a high concentration of customers in neighborhoods where the rate is is very low. The distribution shows also some secondary concentrations around 2% and 4%. The box plot reveals one neighborhood with an unemployment rate of around 9%. This distribution suggests that most customers live in economically sound neighborhoods, which could positively influence credit decisions.

Crime growth rate



The crime growth rate shows an approximately normal distribution, with a mean of 3.86%. However, we can notice that the distribution is clearly left-skewed distribution, suggesting that crime rates generally increased in most districts between 1995 and 1996. Districts where crime increased significantly may present a higher risk for lending.

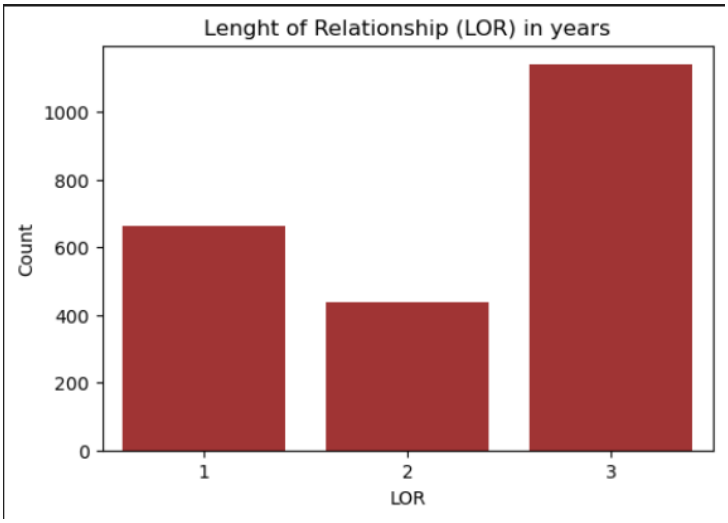
Gender



The gender distribution is almost perfectly balanced, with 1,084 female customers and 1,155 male customers.

This balance is ideal for modeling, as it avoids gender bias in the dataset. The slight male majority is negligible.

Length of Relationship

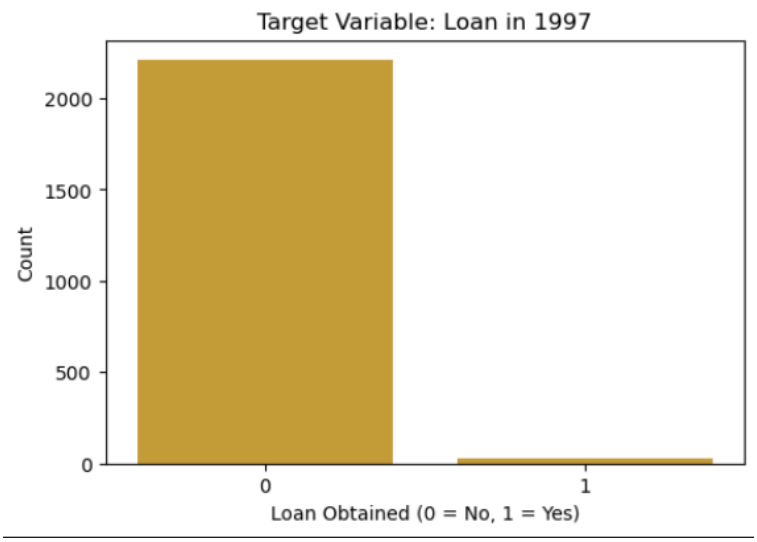


The LOR distribution shows three groups representing the age of accounts in years. The largest group consists of customers with a 3-year relationship (1,139 accounts, 50.9%), the second largest group has a one-year relationship (661 accounts, 29.5) and the smallest group has two-year relationships (439 accounts, 19.6%).

From a credit perspective customers with a LOR of 3 have the longest track record, which banks generally favor when granting loans or credit cards.

C) Graphical analysis – Target Variables

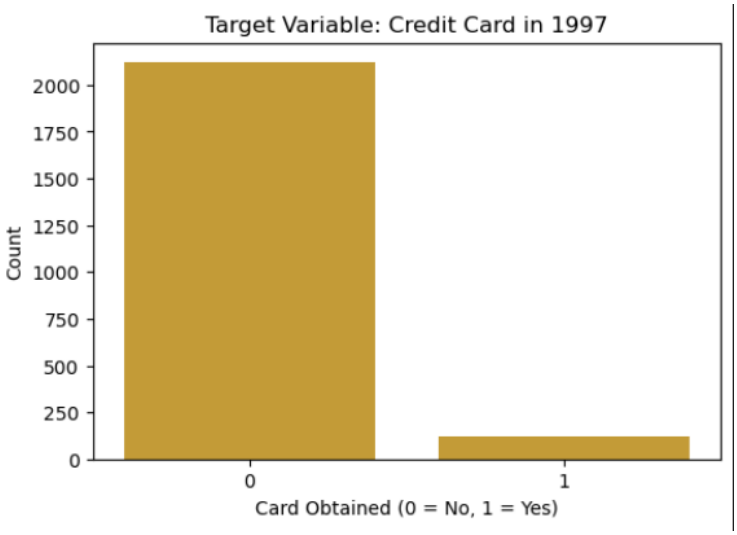
Loan in 1997



The target variable is highly imbalanced: 2,208 customers (98%) did not obtain a loan in 1997, compared to only 31 customers (2%) who did. This significant imbalance between classes poses a

challenge for modeling, as algorithms may tend to predict “0” for all cases in order to achieve high accuracy. It is essential to remedy this imbalance in order to build effective predictive models.

Credit card in 1997



The credit cards target variable shows a similar imbalance, but slightly less severe, with approximately 2,119 customers (95%) who did not obtain a card, compared to 120 customers (5%) who did. The higher adoption rate of credit cards compared to loans is consistent with typical banking habits, as cards have fewer barriers to entry than loans.

VI - References

- cs95. (2018, December 6). *Pandas Merging 101*. Stack Overflow. https://stackoverflow.com/questions/53645882/pandas-merging-101
- GeeksforGeeks. (2020, February 28). *Pandas Tutorial*. GeeksforGeeks. https://www.geeksforgeeks.org/pandas/pandas-tutorial/
- https://plus.google.com/u/0/+Datacamp. (2024). *Get Started*. DataCamp. https://app.datacamp.com/learn/courses/cleaning-data-in-python
- https://plus.google.com/u/0/+Datacamp. (2025). *Get Started*. DataCamp. https://app.datacamp.com/learn/courses/introduction-to-data-visualization-with-matplotlib
- *matplotlib.pyplot.subplots — Matplotlib 3.6.0 documentation*. (n.d.). Matplotlib.org. https://matplotlib.org/stable/api/_as_gen/matplotlib.pyplot.subplots.html
- *matplotlib.pyplot.tight_layout — Matplotlib 3.5.1 documentation*. (n.d.). Matplotlib.org. https://matplotlib.org/stable/api/_as_gen/matplotlib.pyplot.tight_layout.html
- McKinney, W. (2022, August). *Python for data analysis, 3E*. Wesmckinney.com. https://wesmckinney.com/book/
- *pandas.DataFrame.select_dtypes — pandas 1.4.2 documentation*. (n.d.). Pandas.pydata.org. https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.select_dtypes.html
- *pandas.DataFrame.to_csv — pandas 1.1.3 documentation*. (n.d.). Pandas.pydata.org. https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.to_csv.html
- W3Schools. (2019). *Python Lambda*. W3schools.com. https://www.w3schools.com/python/python_lambda.asp
- Astanin, S. (2022, October 6). tabulate: Pretty-print tabular data. PyPI. https://pypi.org/project/tabulate/
- Zach. (2020, June 24). *What is a Bimodal Distribution?* Statology. https://www.statology.org/bimodal-distribution/