IÉSEG
SCHOOL OF MANAGEMENT

Financial Programming Group Project

AMBLARD Léo - FLORES MORAN Alvaro - KODANGADA Ketan Bopanna

Contents

Data Cleaning

The very first step in this project was to take analysis the provided datasets and understand the connection between the variables of the different tables. In order to for each table using python we applied the '.describe()' method to have a first look at the tables.

We also checked the statistical characteristic of the data set using a for loop to check the number of observation and columns for each table of the dataset. We used the following code to achieve this:

```
dflist = [account, card, client, disp, district, loan, order, trans, account]
for i, dfr in enumerate(dflist):
 print(dflist[i].shape)
```

After taking a an in-depth look at the variables, the first step of the data cleaning was to change datatypes for various using the '.astype(int)', '.astype(float)', '.astype(str)' for certain variables that would be needed for further computation forth-coming feature engineering part.

The pattern to in our data cleaning process was the following for most of the variables across the different tables:

- Count the number of missing values in each columns of the data set:

```
# Count the number of missing values in the column
print(trans['amount'].isna().sum())
```

- Count the number of values equal to zero:

```
# Count the number of zero values in the column
zero_values_mask = trans['amount'] ==0
print(zero_values_mask.sum())
```

- Check unique values in the each columns

```
# Check unique values
```

```
print(trans['type'].unique())
```

Feature Engineering

Creation of the Dependent variables

DV1

As the topic demands to create two dependant variables based on the guidelines, we decided to start by creating them to have a better overview and idea of the feature engineering we would create based on those DV's. To create the first dependent variable which regards to clients who had loan granted for the 1997 time period, we first started by filtering in account_id in 'loan' table that had loan in the year 1997. The following code was used to do that:

```
# Change date formats
loan['date'] = pd.to_datetime(loan['date'], format='%y%m%d')

# Select rows where the year is equal to 1997
loan_1997 = loan[loan['date'].dt.year == 1997]
```

The next step was to join the filtered loan (for 1997) table with the 'disp' table to connect the account_id to the client_id. We also dropped unnecessary columns for the computation of the DV, keeping only the client_id, account_id and loan_id in the data frame of the DV.

```
# Drop unecessary columns
loan_1997.drop(columns=['date', 'amount','duration','payments','status'],
inplace = True)

# Merge disp table with dv1 to get the number of client who have loans
(merging on account_id)
dv1 = pd.merge(disp, loan_1997, on = 'account_id', how = 'left')
dv1.head()

dv1.drop(columns=['disp_id','type'], inplace = True)
del dv1['account_id']
dv1.head()
```

At this stage of the creation of the first DV, we can see which client_id has a loan (loan_id) for the 1997 time period. The final step was to change was to change in the loan_id column the NaN

values to zero (that represent the client_id not having any loan for the time period) and all other values (that represent the loan_ID for the client in 1997) to 1. The following code:

```
dv1['loan_id'] = dv1['loan_id'].fillna(0).replace()
dv1['loan_id'] = dv1['loan_id'].astype(int)
dv1["loan_id"] = dv1["loan_id"].where(dv1["loan_id"] != 0, dv1["loan_id"]
```

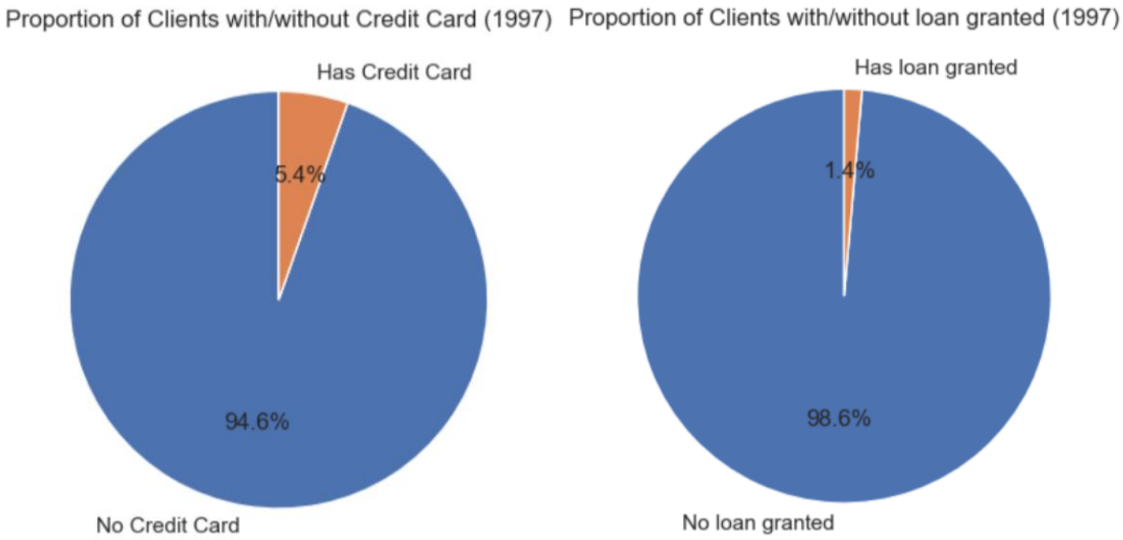
To explain the resonnning behind this, the condition dv1["loan_id"] != 0 checks if the value in the loan_id column is not equal to 0. If the condition is true, the value is kept unchanged; otherwise, the original value is replaced with 1. Essentially, this code only replaces the values in the loan_id column with 1 if they are currently equal to 0. If the value is already not equal to 0, it is left unchanged. In the end this code allows to change loan_id table from having NaN values is the client doesn't have a loan to zero and changes the loan_id of the client in the varible (which means that the client has a loan granted) to 1.

DV2

The process for creating the second dependant variables (Client had credit card issued in 1997) is essentially the same. We started by filtering the card table by date (select rows where the date issued of the card is equal to 1997) and change the date formats of the 'issued' variable to pandas date time using the 'pd.datetime()' method.

As the card issued in the card table are only connected to the disp_id. We merged (through a left join on disp_id) the 'disp' table and filter card table (for 1997) in order to connect which the card_id variable to the client_id. We figured out that client_id and disp_id are exactly the same variables (all values are the same in both variables, each client_id equals the same disp_id) so we dropped this column (using the del method in python). We also dropped all unused columns for the creation of the DV2 in order to keep only the necessary variables (client_id and card_id). Using the '.drop' pandas method. Same as DV1, we changed the all NaN values to 0 and all other values in the card_id column of the data frame to 1. The coding part for DV2 is essentially the same as DV1 but working with different variables and tables so we decided not to include it in the report. For more detail please check the code in the Jupyter Notebook file of the project, a commented section is available for the code of the DV2. For both DV's, data types are integer in order to perform a predictive model on this variable

Using the matplotlib library, we displayed the proportion and distribution of values (1 and 0) for both dependant variables's. We can see that in 1997, 5.4% of the clients in the bank had credit cards issued (for both owners and disponents). Additionally, only 1.4% of clients had loans granted for the



Feature engineering and analysis of the variables

Variables Average Credit, Average Withdrawal, Average Net Financial Activity

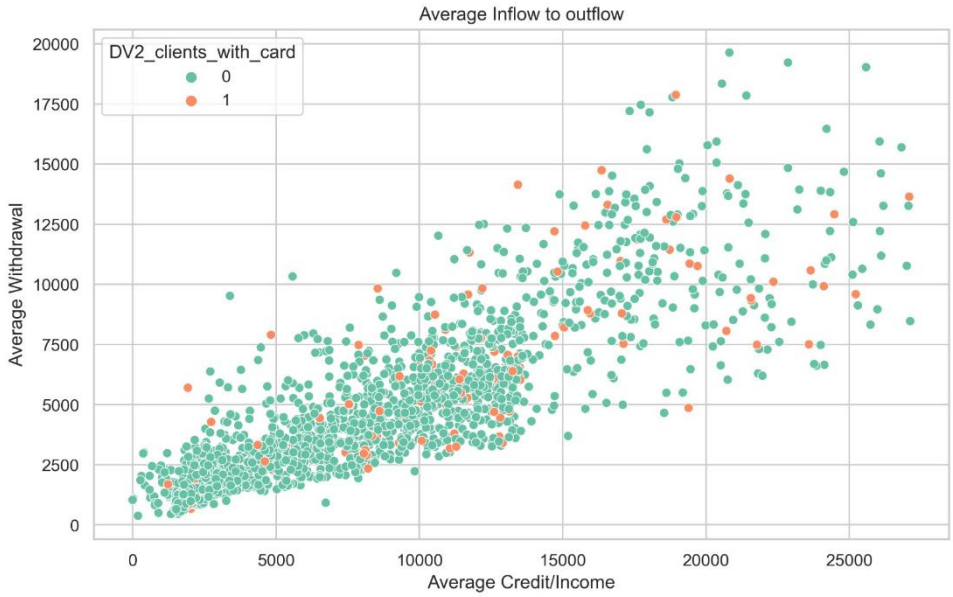
To create these variables, we used the data frame of transactions filtered by 1996. First the means of the amount variable were calculated grouped by 'account_id' and 'type'. Then these calculations were pivoted to have them in columns and at this point it was important to replace missing values by zeros because not all the accounts had credits or withdrawals and their respective average. With the avg_credit and avg_withdrawal set, it was possible to create an additional variable called avg_net_financial_activity that is the subtraction between the avg_credit and the avg_withdrawal.

```
# Creating average Average_Net_Financial_Activity
averages_by_account['avg_net_financial_activity'] = averages_by_account['PRIJEM']-averages_by_account['VYDAJ']
averages_by_account = averages_by_account.rename(columns={'PRIJEM': 'avg_credit', 'VYDAJ': 'avg_withdrawal'})
print(averages_by_account)
```

These variables are important because they allow to display (in average) during 1996 how much money was taken in and out of the account, and the net financial activity. With

them it is possible to analyse the financial behaviour of account holders, for example, if they spend more money than they have in their accounts.

The following graphic shows the scatter plot between the average credit and average withdrawal. In general, it is possible to see that the highest the average income, the highest the average withdrawal



Variables junior_count, classic_count, gold_count:

These variables tell us how many credit cards of different types the bank has given to its clients. There are three types of cards: junior, classic, and gold. Each type has different benefits and requirements. For example, junior cards are for young people who are starting to build their credit history, classic cards are for regular customers who have a stable income and credit score, and gold cards are for premium customers who have a high income and credit score.

The bank may have different strategies for offering these cards to different groups of clients, depending on their financial needs and creditworthiness. By looking at the counts of each card type, we can get an idea of how the bank is distributing its credit cards among its clients.

Variable DispCount, Issued_Year, Issued_Month, Issued_Date

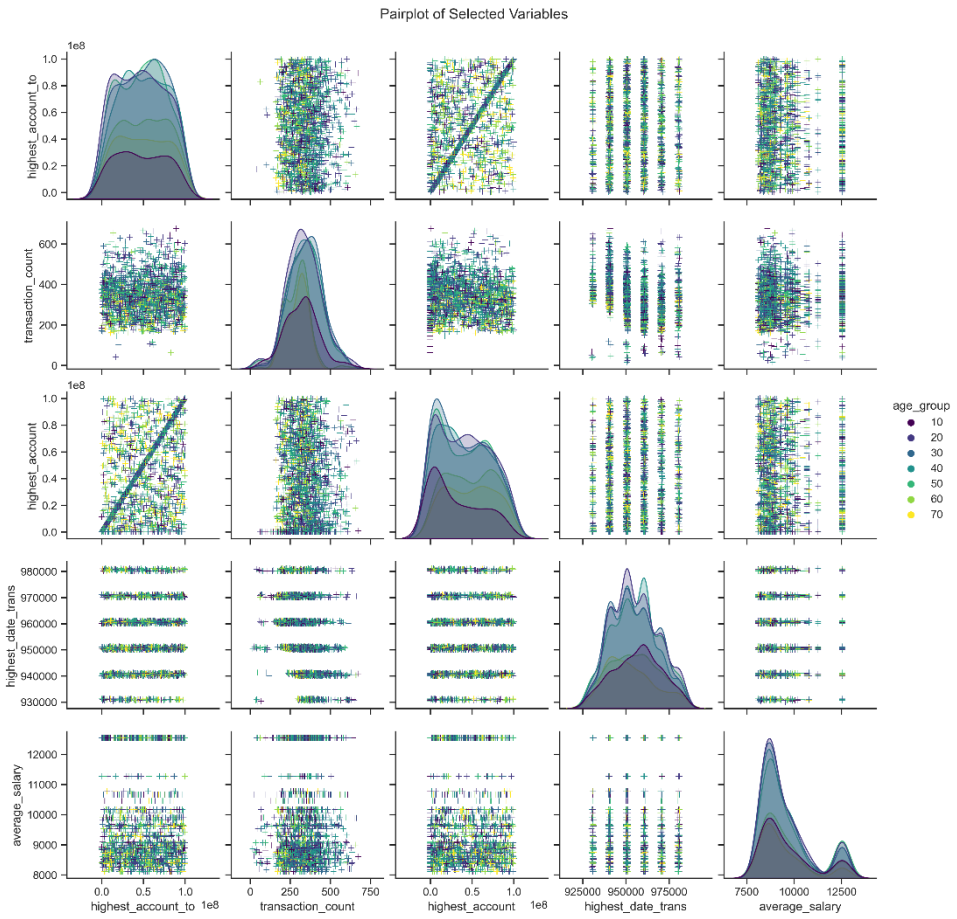
DispCount is a variable that tells us how many Disp cards the bank has given to its clients. Disp cards are a special type of credit cards that have some unique features and benefits. For example, Disp cards have no annual fees, no foreign transaction fees, and a low interest rate. Disp cards are also compatible with mobile payments and online banking. By looking at the DispCount variable, we can see how many clients have chosen this card type and how popular it is among the bank's customers.

Issued_Year, Issued_Month, and Issued_Date are variables that tell us the exact date when a card was issued. We had to format the column ISSUED, to make it more readable. These variables help us to track the trends and patterns of card issuance over time. For example, we can see if there are certain months or seasons when the bank issues more cards than usual, or if there are any changes in the card issuance over the years. This information can help us to understand the bank's performance, customer behaviour, and market conditions.

Variables average_salary, unemployment_rate:

Average Salary is a variable that tells us how much money people in a district make from their work or business. It is important for the bank to know this because it helps them to understand how well-off the people in that area are. The bank can use this information to figure out how much money their clients can afford to spend, save, or pay back their loans. It also helps the bank to offer financial products that suit the income levels of the people in that area

Unemployment Rate is a variable that tells us how many people in a district are looking for a job but can't find one. It is a very important measure of the economy for the bank. A high unemployment rate means that people in that area might have trouble finding a job or keeping a steady income. This could affect their ability to manage their debts or use credit products. A low unemployment rate means that the economy is more stable and people have more job opportunities. This could lead to better chances of paying back their loans.



Pairplot Explanation and Analysis

1. Highest Amounts vs. Age Group: The diagonal KDE plots show the distribution of each variable. For example, the distribution of highest_account_to, transaction_count, highest_account, and average_salary for different age groups.

2. Scatter Plots: The scatter plots represent the relationships between pairs of variables. For instance: There seems to be a positive correlation between highest_account and transaction_count.

3. highest_account_to has a varied relationship with other variables across different age groups. Color Coded by Age Group: The plots are color-coded by age group, allowing you to see if there are any distinct patterns or differences between age groups.

4.Distribution of Average Credit: The distribution of avg_credit for different age groups is shown in the diagonal KDE plots.

5.Outliers: Identify any potential outliers in the scatter plots.

Variable Statement Frequency:

In order to create this variable it was important to validate that there were not outliers in the data frame.

```
# Count the number of missing values in the column
print(df['frequency'].isna().sum())

# Check unique values
print(df['frequency'].unique())
```

```
0
['POPLATEK MESICNE' 'POPLATEK PO OBRATU' 'POPLATEK TYDNE']
```

It was possible to validate that only these strings were present ['POPLATEK MESICNE' 'POPLATEK PO OBRATU' 'POPLATEK TYDNE']

After this we replaced this names by numbers as following: "POPLATEK MESICNE": 1,"POPLATEK TYDNE": 2,"POPLATEK PO OBRATU": 3

This variable was necessary as it is important to have the variables as numeric.

Variables: Number of permanent orders and Total permanent order amount

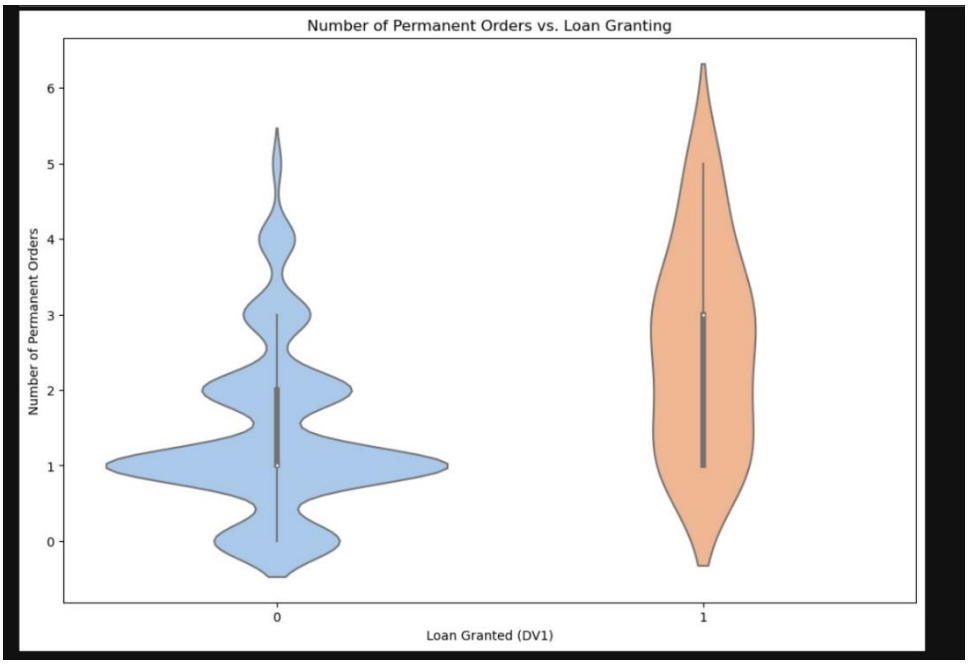
For the creation of these variables, after the data validation and cleaning, it was important to count order ids and sum amounts grouped by account id.

```
# Count the number of unique order_id values for each account_id
order_counts = order.groupby('account_id')['order_id'].nunique().reset_index()

# Sum the amount column by account_id
amount_sum = order.groupby('account_id')['amount'].sum().reset_index()
```

These variables are useful because they permit to identify how many regular, recurring transactions an account holder has set up and how much money is consistently allocated for fixed expenses. Both help to see how well planned the payments are for each account holder, and give an overview of financial stability.

For example, in the graphic created with the data frame. It is possible to conclude that people who have less number permanent orders don't tend to receive loans. By contrary the people who have more than one permanent orders, they do receive loans



Variable: N of card issued in 1996

This variable was created first by filtering the cards table by the year 1996. Then it was possible to count the card_id by the account_id.

```
card1=card.copy()
card1['issued'] = pd.to_datetime(card1['issued'], format='%y%m%d %H:%M:%S')
card1 = card1[card1['issued'].dt.year == 1996]
merge = pd.merge(disp, card1, how='left', on='disp_id')

# Group by 'account_id' and cound card_id to have the number of cards issued in 1996
countcards = merge.groupby(['account_id'])['card_id'].count().reset_index()
```

This variable permit to know how many cards were issued for an account during 1996. This historical data can serve as a source to identify trends and as a predictor for future years.

Appendix

The following table details all the variables created for our final base table:

Item	Meaning	Remark
gender	Gender of the client	Categorical variable (M/F)
age	Age of the client	Numeric values representing age
age_group	Age group of the client	Categorical variable representing age groups
avg_credit	Average credit amount associated with the account	Numeric values representing the average credit

avg_withdrawal	Average withdrawal amount associated with the account	Numeric values representing the average withdrawal
avg_net_financial_activity	Average net financial activity associated with the account	Numeric values representing the average net financial activity
number_of_permanent_orders	Number of permanent orders associated with the account	Numeric values representing the number of permanent orders
total_permanent_order_amount	Total amount of permanent orders associated with the account	Numeric values representing the total permanent order amount
trans_count	Number of transactions associated with the account	Numeric values representing the transaction count
withdrawal_count	Number of withdrawal transactions associated with the account	Numeric values representing the withdrawal transaction count
district_name	Name of the district where the client resides	Categorical variable representing different district names
region_name	Name of the region where the client resides	Categorical variable representing different region names
average_salary	Average salary in the client's district	Numeric values representing average salary
unemployment_rate	Unemployment rate in the client's district	Numeric values representing unemployment rates
DispCount	Count of Disp card types	Numeric values representing counts
major_region	Major region classification	Categorical variable representing different major regions
Issued_Year	Year when an issuance occurred	Numeric values representing years
Issued_Month	Month when an issuance occurred	Numeric values representing months
Issued_Day	Day when an issuance occurred	Numeric values representing days
junior_count	Count of junior card types	Numeric values representing counts
classic_count	Count of classic card types	Numeric values representing counts
gold_count	Count of gold card types	Numeric values representing counts
total_credit	Total credit amount associated with the account	Numeric values representing total credit
N_Issuedcards1996	Number of issued cards in 1996	Numeric values representing counts
DV1 (loan granted)	Binary variable indicating whether a loan was granted (1) or not (0)	Binary variable
DV2_clients_with_card	Binary variable indicating whether clients have a card (1) or not (0)	Binary variable
loan_id	Identifier for loans	Numeric values representing different loans
date	Date associated with a transaction or event	Numeric values representing dates
amount	Amount associated with a transaction or event	Numeric values representing amounts

duration	Duration of a loan	Numeric values representing durations
payments	Number of payments associated with a loan	Numeric values representing payments
status	Status of a transaction or event	Numeric values representing status
order_count	Count of orders associated with the account	Numeric values representing counts
total_order_amount	Total amount of orders associated with the account	Numeric values representing total order amounts
highest_bank_to	Bank code for the highest transaction destination	Categorical variable representing different bank codes
highest_account_to	Account number for the highest transaction destination	Numeric values representing account numbers
highest_k_symbol	Highest transaction symbol	Categorical variable representing different symbols
transaction_count	Number of transactions associated with the account	Numeric values representing the transaction count
highest_type_trans	Type of the highest transaction	Categorical variable representing different transaction types
highest_operation	Highest transaction operation	Categorical variable representing different transaction operations
highest_k_symbol_trans	Highest transaction symbol (additional)	Categorical variable representing different symbols
highest_bank	Bank code for the highest transaction bank	Categorical variable representing different bank codes
highest_account	Account number for the highest transaction bank	Numeric values representing account numbers
highest_date_trans	Date of the highest transaction	Numeric values representing dates
no._of_inhabitants	Number of inhabitants in the district	Numeric values representing counts
no._of_munis_habs<499	Number of municipalities with inhabitants less than 499	Numeric values representing counts
no._of_munis_habs_500-1999	Number of municipalities with inhabitants between 500 and 1999	Numeric values representing counts
no._of_munis_habs_2000-9999	Number of municipalities with inhabitants between 2000 and 9999	Numeric values representing counts
no._of_munis_habs_10000	Number of municipalities with inhabitants greater than 10000	Numeric values representing counts
ratio_of_urban_inhabitants	Ratio of urban inhabitants in the district	Numeric values representing ratios
no._of_enterpreneurs_per_1000_inhabitants	Number of entrepreneurs per 1000 inhabitants	Numeric values representing counts
no._of_commited_crimes_96	Number of committed crimes in 1996	Numeric values representing counts