Detect Potential Loan Default Customers

Home Credit Indonesia

Data Scientist Virtual Internship Program

Andi Eka Nugraha

HOME CREDIT

CONTENTS

01 BUSINESS UNDERSTANDING

PROFIL

Created by:
Andi Eka Nugraha
an.ekanugraha@gmail.com
linkedin.com/in/andi-eka-nugraha

Bachelor in Physics with major expertise in Instrumentation & Robotics and has attended a Datascience bootcamp for 4 months. Experienced in programming microcontrollers and machine learning to process data or images, as well as creating robotic systems that can support human work. Able to understand business, especially for data analysis, studying statistics and machine learning, as well as the ability to create regression models, classification, and clustering. Skills in identifying and analyzing patterns in data and presenting analytical results well.

Dataset :

https://www.kaggle.com/competitions/home-credit-default-risk

Code :

https://github.com/AnCodingML/Home-Credit-Default-Risk

Click :

Current Issue

Home Credit is currently using various statistical methods and Machine Learning to make credit score predictions from customers who apply for loans.

As a data scientist, you are tasked with creating a model that **can detect customers who are able to make payments and not be rejected when applying for a loan**

What is the Problems?

Goals

Early identification of credit risk and taking appropriate action to reduce possible losses.

Objectives

- Create machine learning that can detect defaulting customers
- Reducing company losses due to customer credit defaults

Business Metrics

Credit Loss Ratio

02

EXPLORATORY DATA ANALYSIS

Dataset

installments_payments.csv

Repayment history for the previously disbursed credits in Home Credit related to the loans.

credit_card_balance.csv

Monthly balance snapshots of previous credit cards that the applicant has with Home Credit.

POS_CASH_balance.csv

Monthly balance snapshots of previous POS (point of sales) and cash loans that the applicant had with Home Credit.

previous_application.csv

All previous applications for Home Credit loans of clients who have loans

Dataset

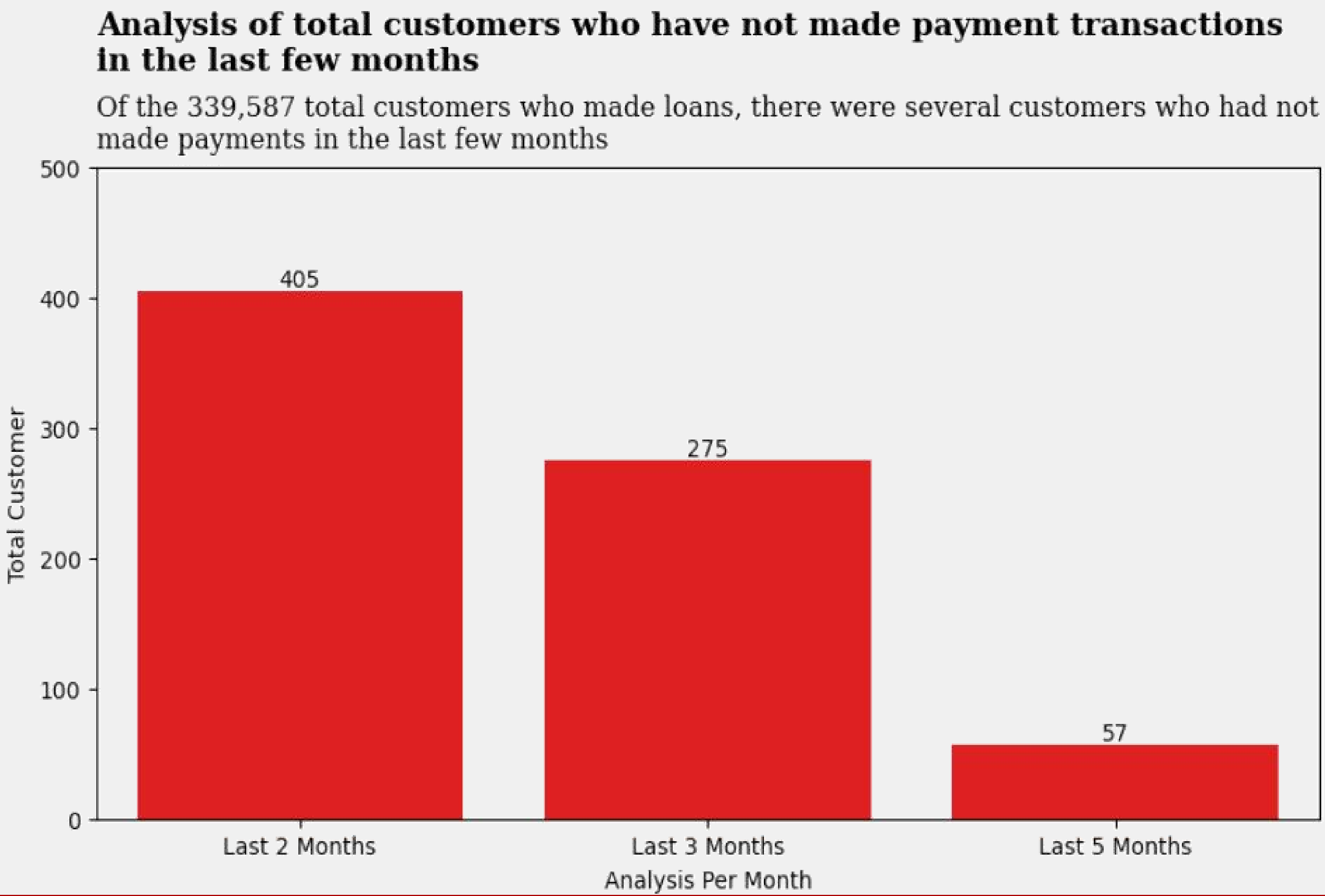
bureau.csv

All client's previous credits provided by other financial institutions that were reported to Credit Bureau

application_{train|test}.csv

Static data for all applications

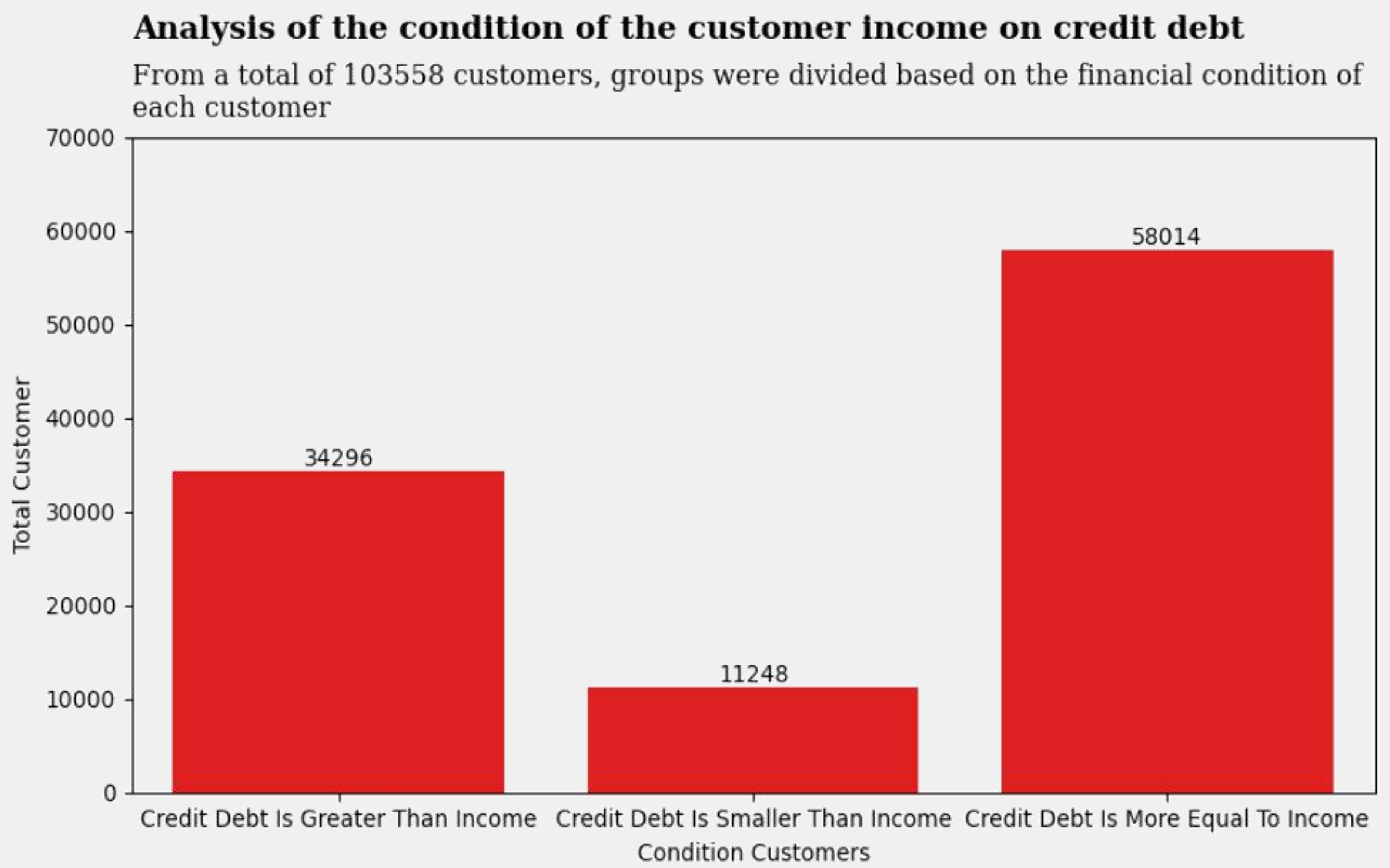
EXPLORATORY DATA ANALYSIS : installments_payments.csv



Total losses due to customers making loan payments amounted to $47322631.35, and losses in the last month amounted were $11504.25.

There were a total of 1149574 late payments from all customers and there were 26450 late payments in the last month.

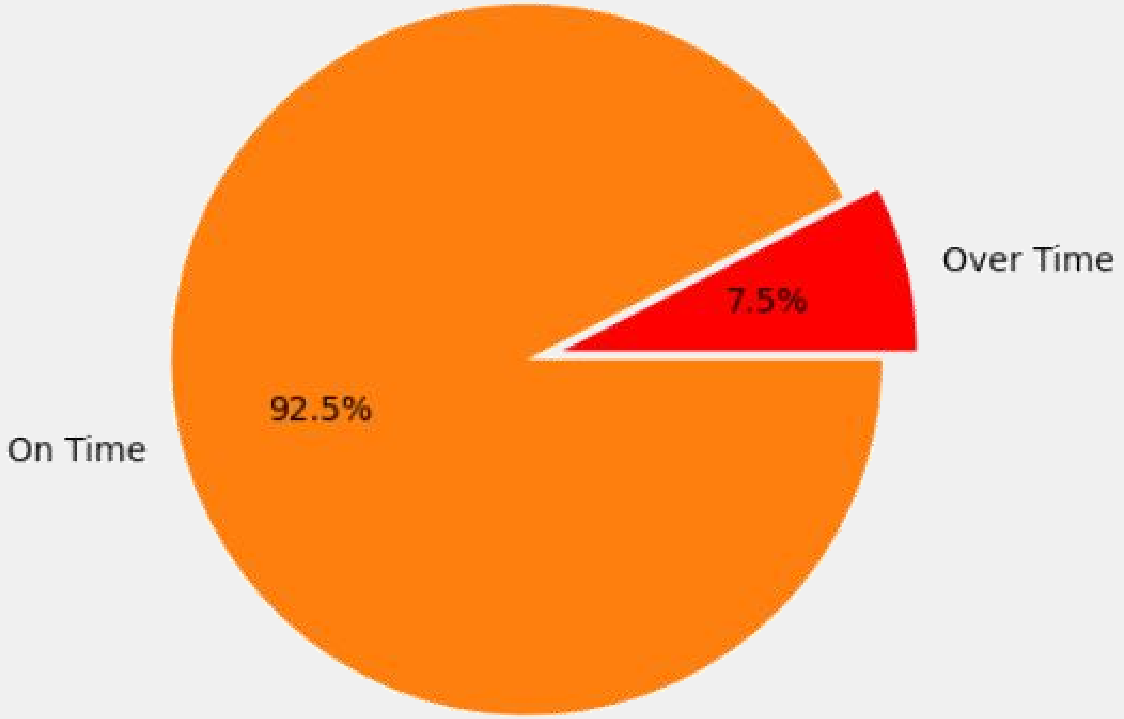
EXPLORATORY DATA ANALYSIS : credit_card_balance.csv

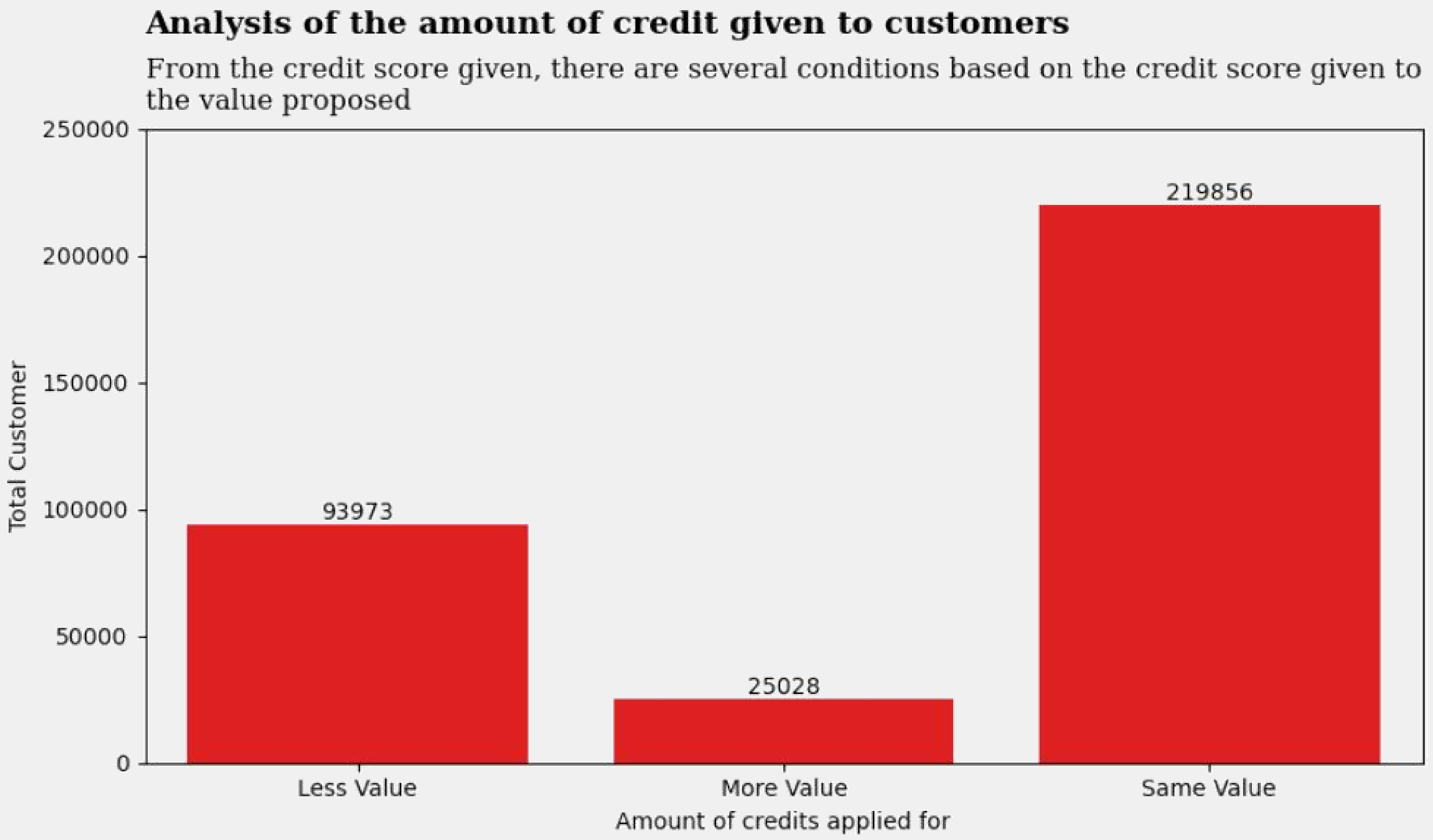


There are **11248** customers who must receive special attention because of their financial condition whose income is lower than credit debt

Analysis of customers making credit bill payments beyond the time limit/tolerance limit

out of a total of 337252 customer payments, there were 7% of transactions that exceeded the time limit or tolerance period

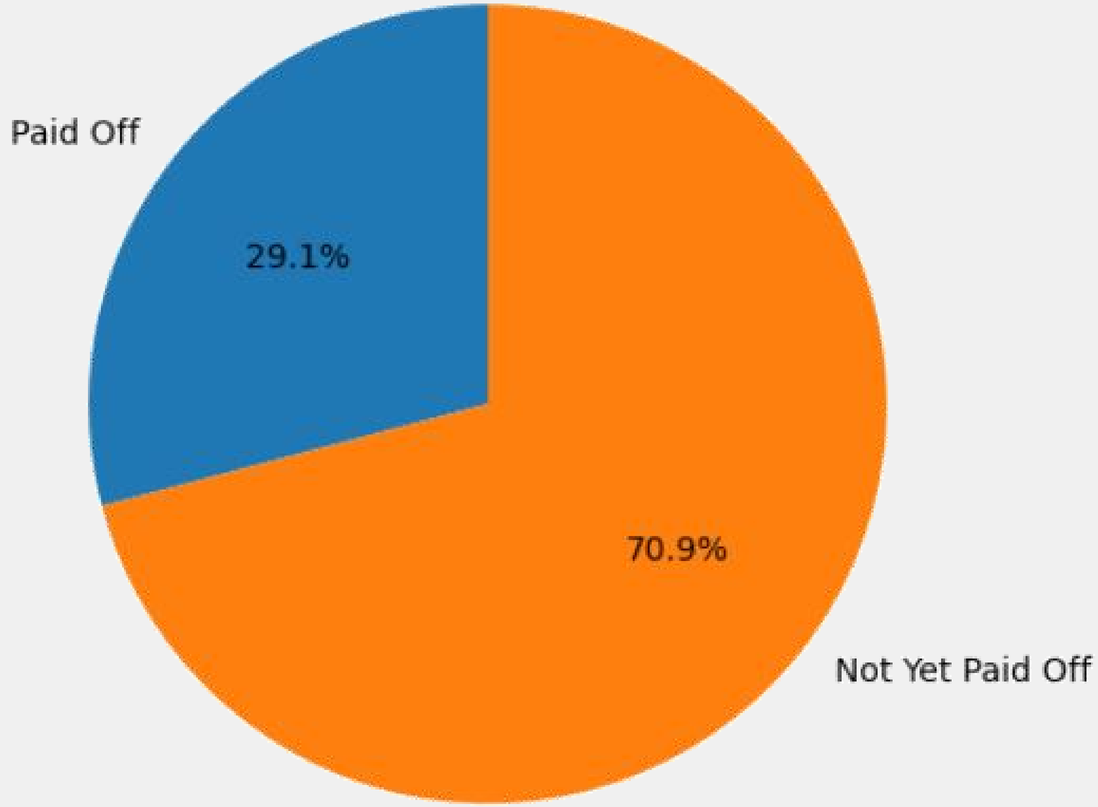



Analysis of the amount of credit given to customers
From the credit score given, there are several conditions based on the credit score given to the value proposed
250000
200000
150000
100000
50000
0
Total Customer
93973
25028
219856
Less Value
More Value
Same Value
Amount of credits applied for

EXPLORATORY DATA ANALYSIS : bureau.csv

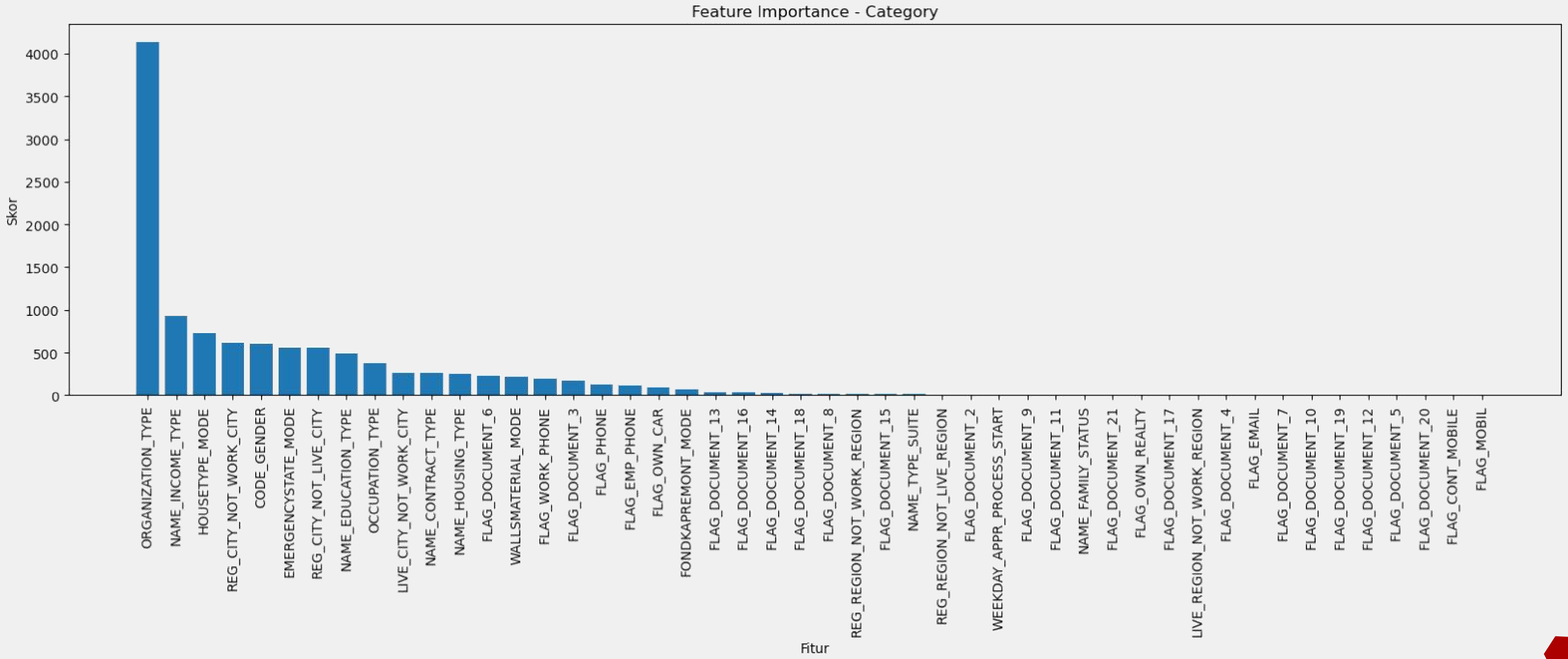
Analysis of customer payment installment conditions

out of a total of 305811 customers, 29.1% of them have completed their installment payments



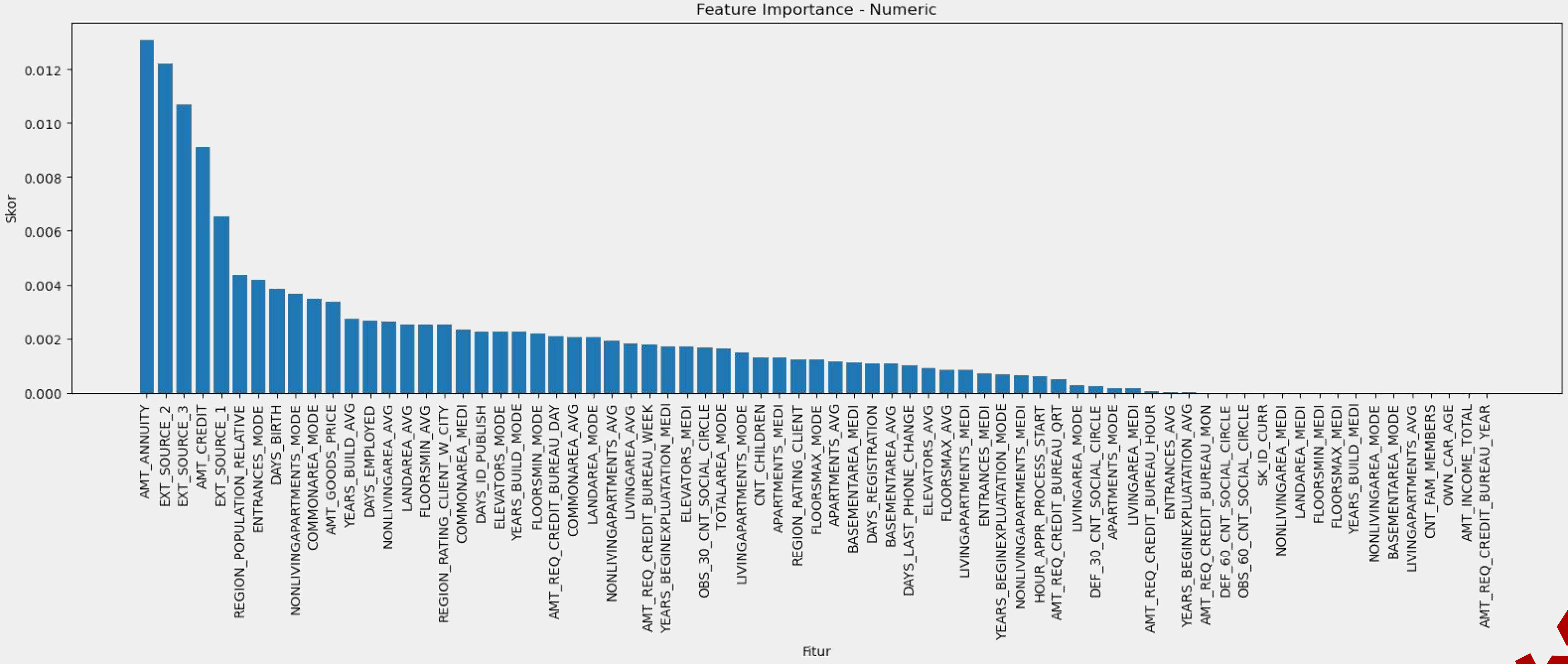
EXPLORATORY DATA ANALYSIS : application_train.csv

Measure the feature importance of the category column using chi-square



EXPLORATORY DATA ANALYSIS : application_train.csv

Measure the feature importance of the numeric column using regression

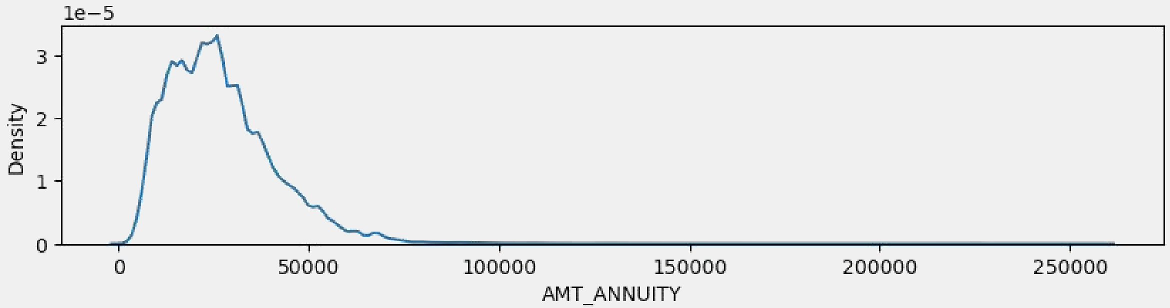


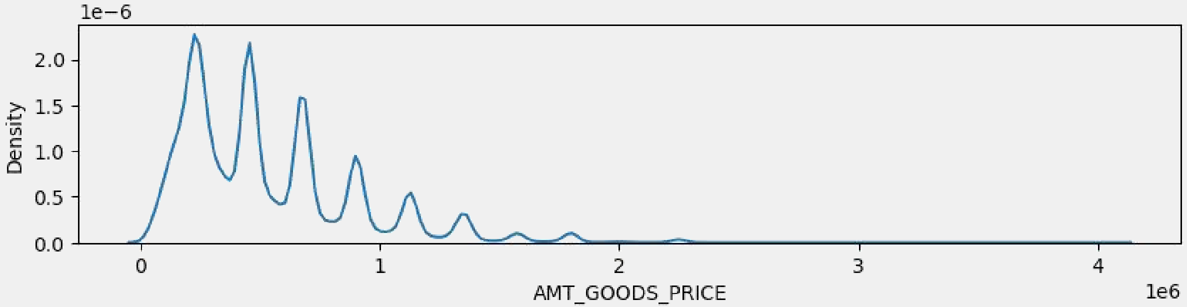
EXPLORATORY DATA ANALYSIS : application_train.csv

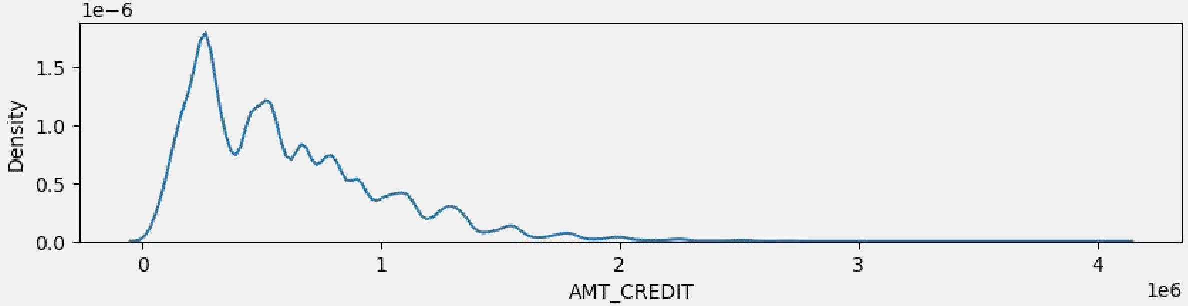
Features that are selected and considered influential based on select k-best and dataset analysis :

```
data = pd.DataFrame()
feature_importance = ['SK_ID_CURR','TARGET',
                      'ORGANIZATION_TYPE','NAME_INCOME_TYPE','HOUSETYPE_MODE','CODE_GENDER','NAME_EDUCATION_TYPE','REG_CITY_NOT_WORK_CITY',
                      'DAYS_BIRTH','EXT_SOURCE_1','EXT_SOURCE_2','EXT_SOURCE_3','AMT_ANNUITY','AMT_CREDIT','AMT_GOODS_PRICE']
data = application_train[feature_importance]
```

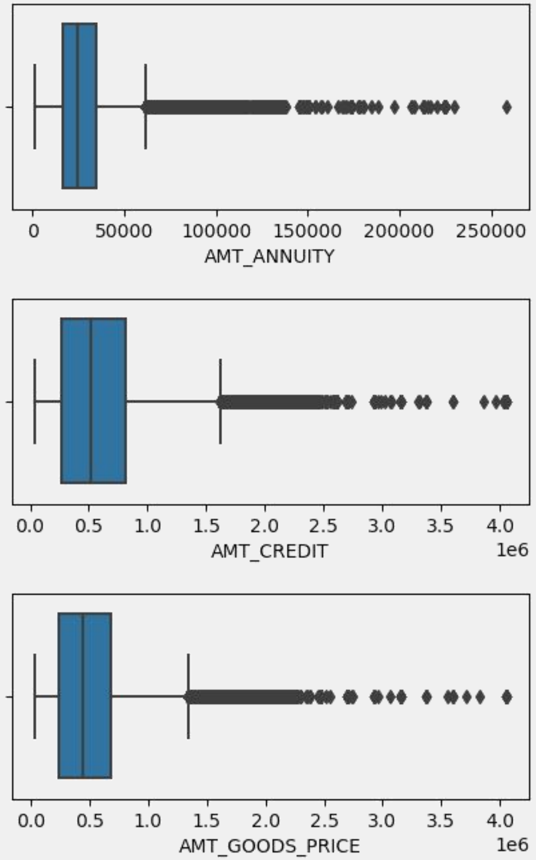
The AMT_ANNUITY, AMT_CREDIT, AMT_GOODS_PRICE features have a left skew data distribution indicating that there is data accumulation in a low class. This type of data has the potential to contain outliers.

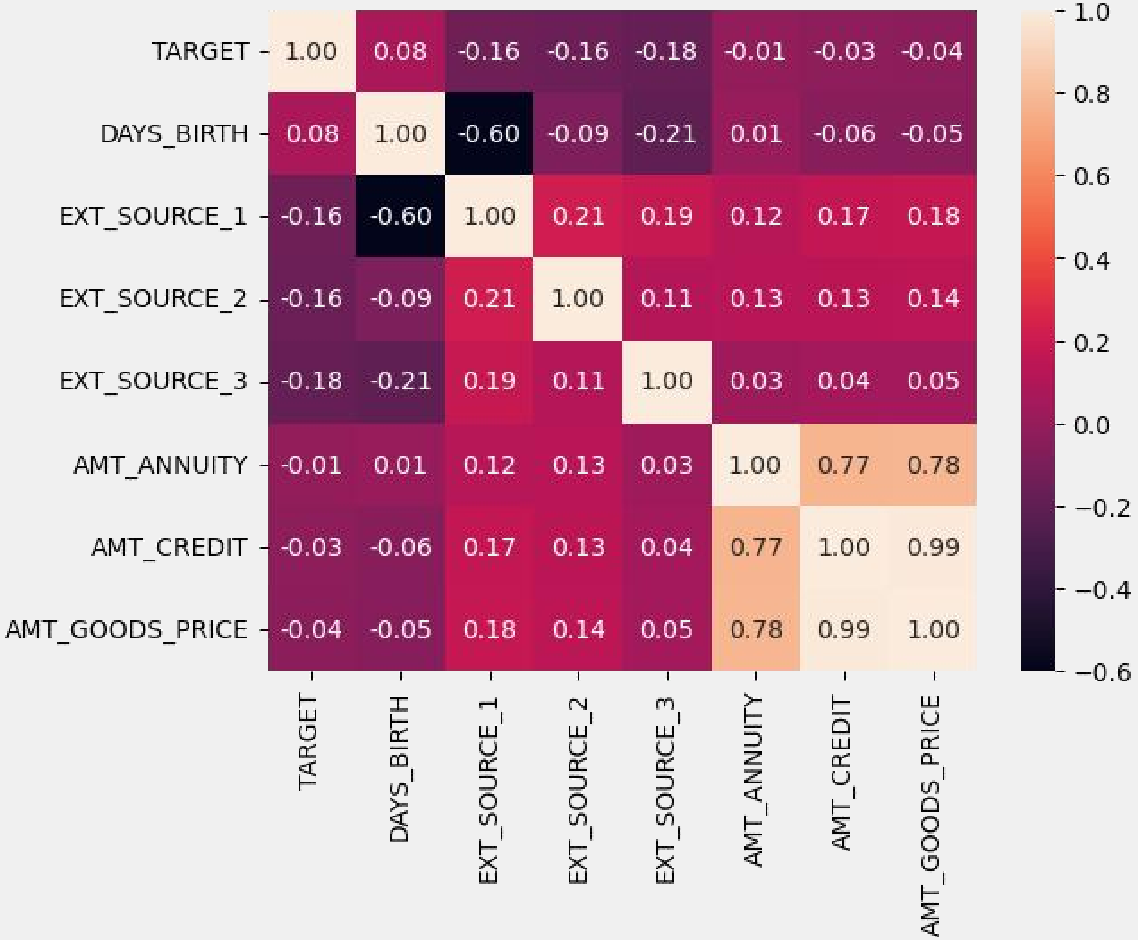






EXPLORATORY DATA ANALYSIS : application_train.csv





The AMT_ANNUITY, AMT_CREDIT, AMT_GOODS_PRICE features have outlier values so they need further processing

Features AMT_ANNUITY, AMT_CREDIT, AMT_GOODS_PRICE have a high correlation so they are potentially redundant

03

DATA PREPROCESSING

DATA PREPROCESSING

Feature Enginering installments_payments.csv

```
installment_agg_last5 = installment_payments.groupby('SK_ID_CURR').tail(5).groupby('SK_ID_CURR').agg({'NUM_INSTALMENT_VERSION': 'mean',
                                                                                                     'DAYS_INSTALMENT': 'max', 'DAYS_ENTRY_PAYMENT': 'max', 'AMT_INSTALMENT': 'sum',
                                                                                                     'AMT_PAYMENT': 'sum'}).reset_index()
```

	SK_ID_CURR	NUM_INSTALMENT_VERSION	DAYS_INSTALMENT	DAYS_ENTRY_PAYMENT	AMT_INSTALMENT	AMT_PAYMENT
0	100001	1.200000	2886.0	2875.0	33262.875	33262.875
1	100002	1.000000	475.0	498.0	46258.875	46258.875
2	100003	1.000000	2310.0	2324.0	216925.920	216925.920
3	100004	1.333333	784.0	795.0	21288.465	21288.465
4	100005	1.000000	676.0	711.0	24066.000	24066.000

Aggregate based on the last 5 data on each customer to get data distribution and insight in a wider timeframe but still efficient

DATA PREPROCESSING

Feature Enginering credit_card_balance.csv

```
credit_card_balance.fillna(0)
```

```
credit_card_balance['max_current'] = credit_card_balance['AMT_BALANCE'] + credit_card_balance['AMT_CREDIT_LIMIT_ACTUAL']
credit_card_balance
```

```
pd.set_option('display.max_columns', None)
x = ['SK_ID_PREV','SK_ID_CURR','MONTHS_BALANCE','CNT_INSTALMENT_MATURE_CUM','AMT_BALANCE',
     'AMT_CREDIT_LIMIT_ACTUAL','max_current','AMT_INST_MIN_REGULARITY','AMT_PAYMENT_TOTAL_CURRENT',
                                        'AMT_TOTAL_RECEIVABLE','CNT_DRAWINGS_CURRENT','SK_DPD','SK_DPD_DEF']
credit_card_balance = credit_card_balance[x]
```

```
credit_card_balance_agg = credit_card_balance.groupby('SK_ID_CURR').tail(1).groupby('SK_ID_CURR').agg({'MONTHS_BALANCE': 'min','CNT_INSTALMENT_MATURE_CUM':'max',})
```

```
sorted_data = credit_card_balance.sort_values('MONTHS_BALANCE')

# Group by 'SK_ID_CURR' and calculate minimum 'MONTHS_BALANCE' and maximum 'CNT_INSTALMENT_MATURE_CUM'
grouped_data = sorted_data.groupby('SK_ID_CURR').agg({
    'MONTHS_BALANCE': 'min',
    'CNT_INSTALMENT_MATURE_CUM': 'max'
})

# Get the last row for each group based on the sorted order
last_row_indices = sorted_data.groupby('SK_ID_CURR').tail(1).index
# Get the last values for other columns based on the last row indices
last_values = sorted_data.loc[last_row_indices]
# Merge the grouped data and last values
credit_card_balance_agg = pd.merge(grouped_data, last_values, on='SK_ID_CURR').abs()
```

```
drop = ['MONTHS_BALANCE_y','CNT_INSTALMENT_MATURE_CUM_y']
credit_card_balance_agg = credit_card_balance_agg.drop(drop, axis=1)
```

```
credit_card_balance_agg['unpaid_invoice_amount'] = credit_card_balance_agg['AMT_TOTAL_RECEIVABLE'] - credit_card_balance_agg['AMT_PAYMENT_TOTAL_CURRENT']
```

#	Column	Non-Null Count	Dtype
0	SK_ID_CURR	103558 non-null	int64
1	MONTHS_BALANCE_x	103558 non-null	int64
2	CNT_INSTALMENT_MATURE_CUM_x	103558 non-null	float64
3	SK_ID_PREV	103558 non-null	int64
4	AMT_BALANCE	103558 non-null	float64
5	AMT_CREDIT_LIMIT_ACTUAL	103558 non-null	int64
6	max_current	103558 non-null	float64
7	AMT_INST_MIN_REGULARITY	103558 non-null	float64
8	AMT_PAYMENT_TOTAL_CURRENT	103558 non-null	float64
9	AMT_TOTAL_RECEIVABLE	103558 non-null	float64
10	CNT_DRAWINGS_CURRENT	103558 non-null	int64
11	SK_DPD	103558 non-null	int64
12	SK_DPD_DEF	103558 non-null	int64
13	unpaid_invoice_amount	103558 non-null	float64

DATA PREPROCESSING

Feature Enginering POS_CASH_balance.csv

```
POS_CASH_balance_agg = POS_CASH_balance.groupby(['SK_ID_CURR']).agg({'SK_DPD': 'sum','SK_DPD_DEF':'sum',}).reset_index()
POS_CASH_balance_agg['DIF_SK_DPD_SK_DPD_DEF'] = POS_CASH_balance_agg['SK_DPD'] - POS_CASH_balance_agg['SK_DPD_DEF']
```

	SK_ID_CURR	SK_DPD	SK_DPD_DEF	DIF_SK_DPD_SK_DPD_DEF
0	100001	7	7	0
1	100002	0	0	0
2	100003	0	0	0
3	100004	0	0	0
4	100005	0	0	0

Feature Enginering previous_application.csv

```
previous_application['PREV_COUNT'] = previous_application.groupby('SK_ID_CURR').cumcount() + 1

# Menggabungkan data untuk 5 aplikasi terakhir
df_last_5 = previous_application[previous_application['PREV_COUNT'] <= 5].groupby('SK_ID_CURR').agg({
    'AMT_ANNUITY': 'mean',
    'AMT_APPLICATION': 'sum',
    'AMT_CREDIT': 'sum',
    'NAME_CONTRACT_STATUS': lambda x: x.mode().iat[0]
}).reset_index()
```

	SK_ID_CURR	AMT_ANNUITY	AMT_APPLICATION	AMT_CREDIT	NAME_CONTRACT_STATUS
0	100001	3951.0000	24835.5	23787.0	Approved
1	100002	9251.7750	179055.0	179055.0	Approved
2	100003	56553.9900	1306309.5	1452573.0	Approved
3	100004	5357.2500	24282.0	20106.0	Approved
4	100005	4813.2000	44617.5	40153.5	Approved

Feature Enginering bureau.csv

```
bureau['AMT_CREDIT_SUM_DEBT'] = bureau['AMT_CREDIT_SUM_DEBT'].fillna(0)
bureau['AMT_CREDIT_SUM'] = bureau['AMT_CREDIT_SUM'].fillna(bureau['AMT_CREDIT_SUM_DEBT'])
bureau_agg = bureau.groupby('SK_ID_CURR').agg({'AMT_CREDIT_SUM':'sum',
                                                'AMT_CREDIT_SUM_DEBT':'sum',
                                                'DAYS_CREDIT_UPDATE':'max'}).reset_index()
```

	SK_ID_CURR	AMT_CREDIT_SUM	AMT_CREDIT_SUM_DEBT	DAYS_CREDIT_UPDATE
0	100001	1453365.000	596686.500	-6
1	100002	865055.565	245781.000	-7
2	100003	1017400.500	0.000	-43
3	100004	189037.800	0.000	-382
4	100005	657126.000	568408.500	-11

Merge Datasets

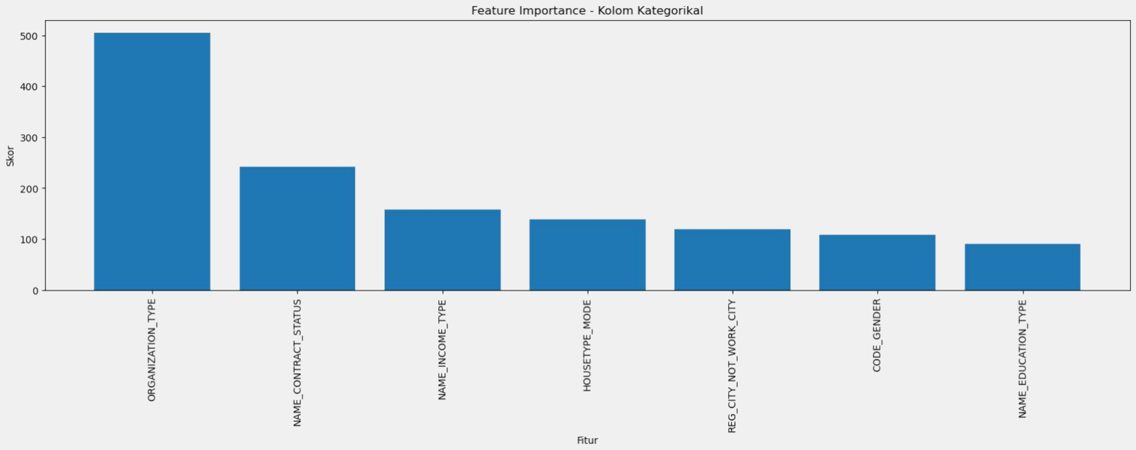
```
data = pd.DataFrame()
feature_importance = ['SK_ID_CURR','TARGET',
                      'ORGANIZATION_TYPE','NAME_INCOME_TYPE','HOUSETYPE_MODE','CODE_GENDER','NAME_EDUCATION_TYPE','REG_CITY_NOT_WORK_CITY',
                      'DAYS_BIRTH','EXT_SOURCE_1','EXT_SOURCE_2','EXT_SOURCE_3','AMT_ANNUITY','AMT_CREDIT','AMT_GOODS_PRICE']
data = application_train[feature_importance]
```

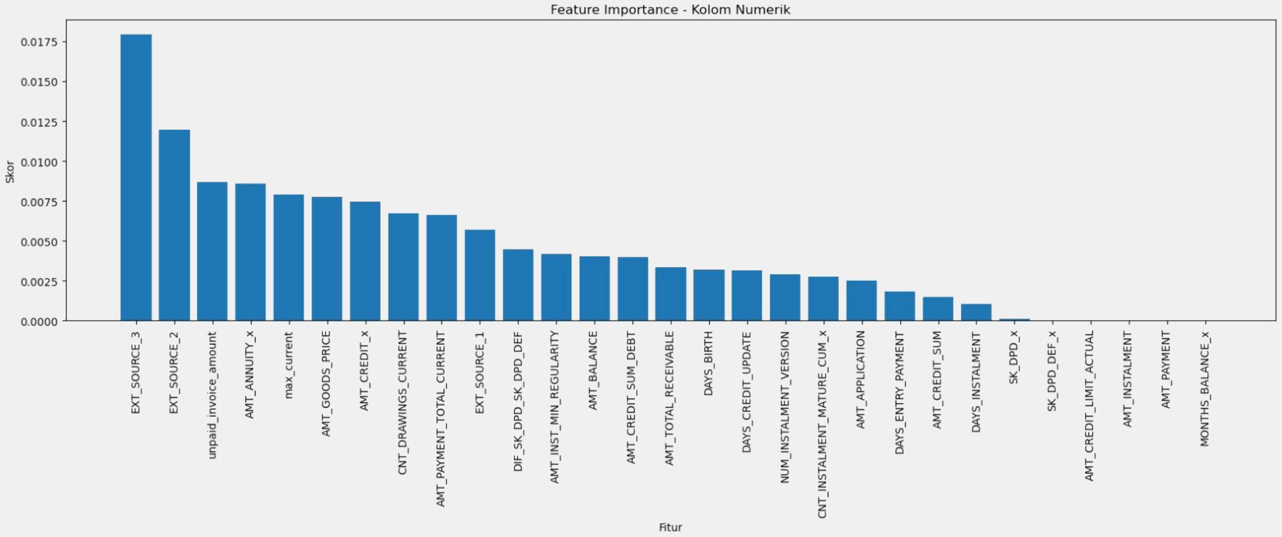
```
# Daftar semua dataset
datasets = [data, credit_card_balance_agg, installment_agg_last5, POS_CASH_balance_agg, df_last_5, bureau_agg]

# Lakukan inner join berdasarkan SK_ID_CURR
merged_df = reduce(lambda left, right: pd.merge(left, right, on='SK_ID_CURR', how='inner'), datasets)

# merged_df akan berisi hasil inner join dari semua dataset berdasarkan SK_ID_CURR
merged_df
```

```
drop = ['AMT_ANNUITY_y','AMT_CREDIT_y','SK_DPD_y','SK_DPD_DEF_y','SK_ID_PREV']
merged_df = merged_df.drop(drop, axis =1)
```

DATA PREPROCESSING

Merge Datasets Feature Importance





Merge Datasets Feature Importance

```
drop = ['CNT_DRAWINGS_CURRENT','DAYS_ENTRY_PAYMENT','AMT_CREDIT_SUM_DEBT','SK_DPD_x','DAYS_CREDIT_UPDATE','MONTHS_BALANCE_x','DIF_SK_DPD_SK_DPD_DEF',
        'AMT_CREDIT_SUM','SK_DPD_DEF_x','NUM_INSTALMENT_VERSION','AMT_INSTALMENT','AMT_PAYMENT']

merged_df = merged_df.drop(drop, axis =1)
```

Merge Datasets Feature Enginering

Fill in the empty category data with the mode and numeric data with the average.

```
for col in num:
    mean_value = merged_df[col].mean()
    merged_df[col].fillna(mean_value, inplace=True)

for col in cat:
    mode_value = merged_df[col].mode()[0]
    merged_df[col].fillna(mode_value, inplace=True)
```

Redundant data aggregation

```
merged_df['DAYS_BIRTH'] = merged_df['DAYS_BIRTH']/-365

merged_df['credit_downpayment'] = merged_df['AMT_GOODS_PRICE'] - merged_df['AMT_CREDIT_x']
merged_df['credit_goods_price_ratio'] = merged_df['AMT_CREDIT_x']/merged_df['AMT_GOODS_PRICE']
merged_df['credit_annuity_ratio'] = merged_df['AMT_CREDIT_x']/merged_df['AMT_ANNUITY_x']

drop = ['AMT_GOODS_PRICE','AMT_CREDIT_x','AMT_ANNUITY_x']
merged_df = merged_df.drop(drop, axis = 1)
```

DATA PREPROCESSING

Label Encoding

```
label_col = ['REG_CITY_NOT_WORK_CITY','NAME_EDUCATION_TYPE','CODE_GENDER','ORGANIZATION_TYPE']
label_encoder = LabelEncoder()
for col in label_col:
    merged_df[col] = label_encoder.fit_transform(merged_df[col])
    label_names = label_encoder.classes_  # Mendapatkan nama label yang diubah
    print(f"Nama label yang diubah pada kolom {col}:")
    print(label_names)
    print()
```

One Hot Encoding

```
x = pd.get_dummies(merged_df['NAME_INCOME_TYPE'], prefix = 'INCOME_TYPE_')
y = pd.get_dummies(merged_df['HOUSETYPE_MODE'], prefix = 'HOUSETYPE_')
z = pd.get_dummies(merged_df['NAME_CONTRACT_STATUS'], prefix = 'STATUS')
```

```
merged_df = pd.concat([merged_df, x], axis=1)
merged_df = pd.concat([merged_df, y], axis=1)
merged_df = pd.concat([merged_df, z], axis=1)
drop = ['NAME_INCOME_TYPE','HOUSETYPE_MODE','NAME_CONTRACT_STATUS']
merged_df = merged_df.drop(drop, axis = 1)
```

DATA PREPROCESSING

Transform data

```
dataset_train = pd.DataFrame()
# Mengambil nama kolom dalam dataset
merged = merged_df.drop('SK_ID_CURR', axis =1)
columns = merged.columns

# Membuat objek MinMaxScaler
scaler = MinMaxScaler()

# Melakukan normalisasi pada semua kolom
dataset_train[columns] = scaler.fit_transform(merged_df[columns])
```

```
---  ------                           --------------  -----
0    TARGET                           72793 non-null  float64
1    ORGANIZATION_TYPE                72793 non-null  float64
2    CODE_GENDER                      72793 non-null  float64
3    NAME_EDUCATION_TYPE              72793 non-null  float64
4    REG_CITY_NOT_WORK_CITY           72793 non-null  float64
5    DAYS_BIRTH                       72793 non-null  float64
6    EXT_SOURCE_1                     72793 non-null  float64
7    EXT_SOURCE_2                     72793 non-null  float64
8    EXT_SOURCE_3                     72793 non-null  float64
9    CNT_INSTALMENT_MATURE_CUM_x      72793 non-null  float64
10   AMT_BALANCE                      72793 non-null  float64
11   AMT_CREDIT_LIMIT_ACTUAL          72793 non-null  float64
12   max_current                      72793 non-null  float64
13   AMT_INST_MIN_REGULARITY          72793 non-null  float64
14   AMT_PAYMENT_TOTAL_CURRENT        72793 non-null  float64
15   AMT_TOTAL_RECEIVABLE             72793 non-null  float64
16   unpaid_invoice_amount            72793 non-null  float64
17   DAYS_INSTALMENT                  72793 non-null  float64
18   AMT_APPLICATION                  72793 non-null  float64
19   credit_downpayment               72793 non-null  float64
20   credit_goods_price_ratio         72793 non-null  float64
21   credit_annuity_ratio             72793 non-null  float64
22   INCOME_TYPE__Commercial associate 72793 non-null  float64
23   INCOME_TYPE__Pensioner           72793 non-null  float64
24   INCOME_TYPE__State servant       72793 non-null  float64
25   INCOME_TYPE__Student             72793 non-null  float64
26   INCOME_TYPE__Working             72793 non-null  float64
27   HOUSETYPE__block of flats        72793 non-null  float64
28   HOUSETYPE__specific housing      72793 non-null  float64
29   HOUSETYPE__terraced house        72793 non-null  float64
30   STATUS_Approved                  72793 non-null  float64
31   STATUS_Canceled                  72793 non-null  float64
32   STATUS_Refused                   72793 non-null  float64
33   STATUS_Unused offer              72793 non-null  float64
```

04

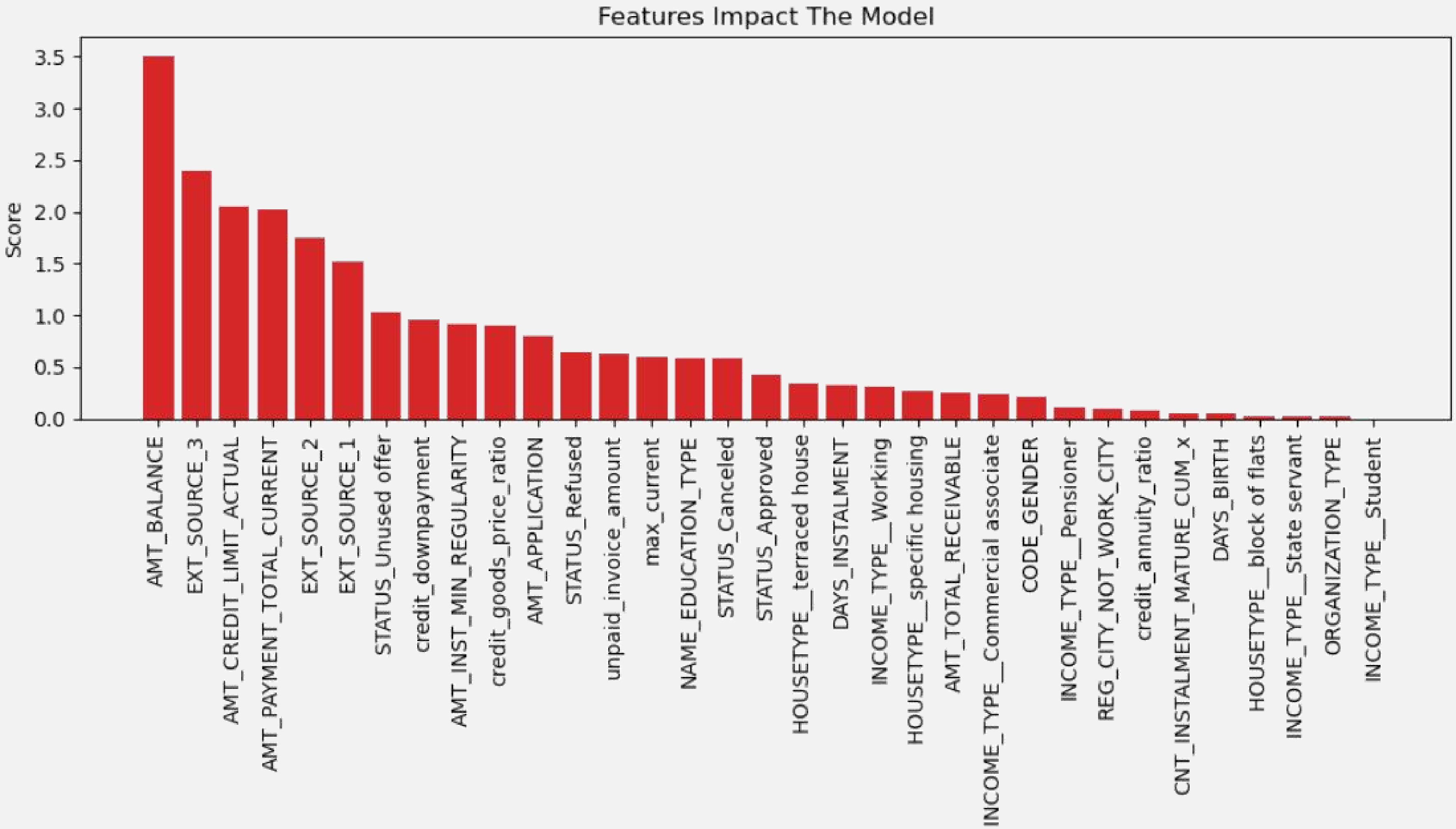
DATA MODELING

Evaluation Metrics : Receiver Operating Characteristic Area Under the Curve

Ket.	Logistic Regression	Light Gradien Boosting Machine
AUC	0.75	0.77
roc_auc (crossval train)	0.74	0.86
roc_auc (crossval test)	0.74	0.75

Based on testing result, model that has best performance is **Logistic Regression**

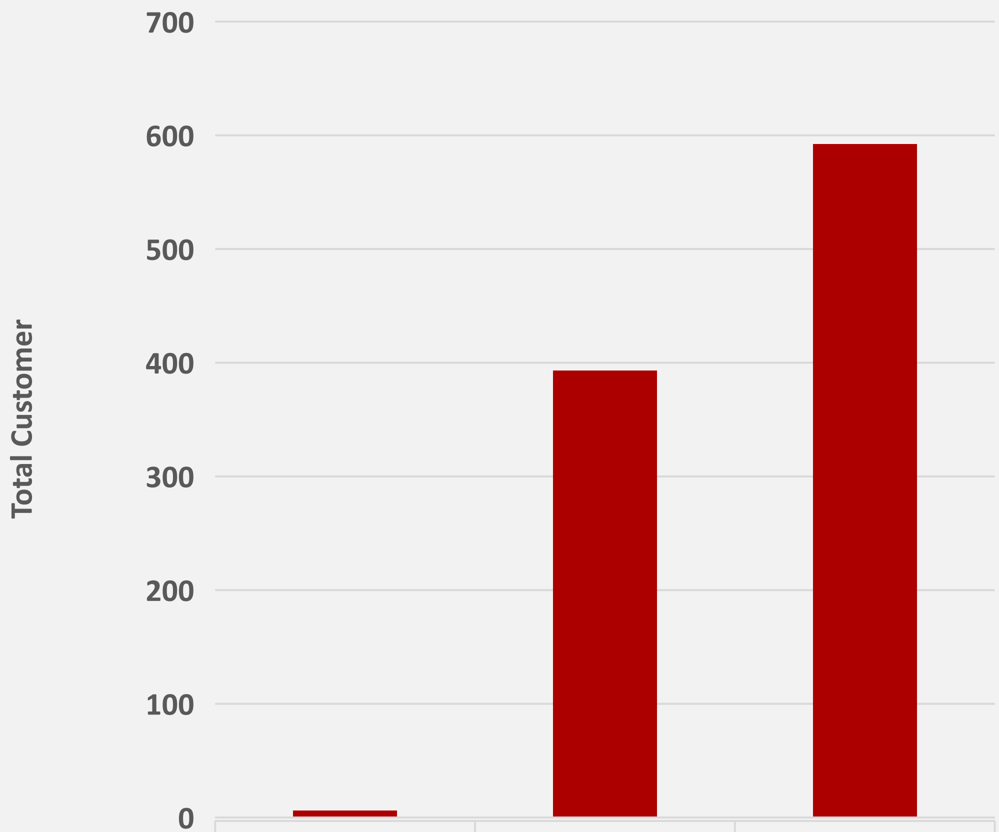
DATA MODELING



The features that have the most impact on machine learning models are **AMT_BALANCE, EXT_SOURCE_3,** and **AMT_CREDIT_LIMIT_ACTUAL**

MODEL TESTING

Comparative analysis of the results of predicting the total customers at risk of default for each model with real data



	LGBM	LOGREG	REAL
Total Customers	6	393	592

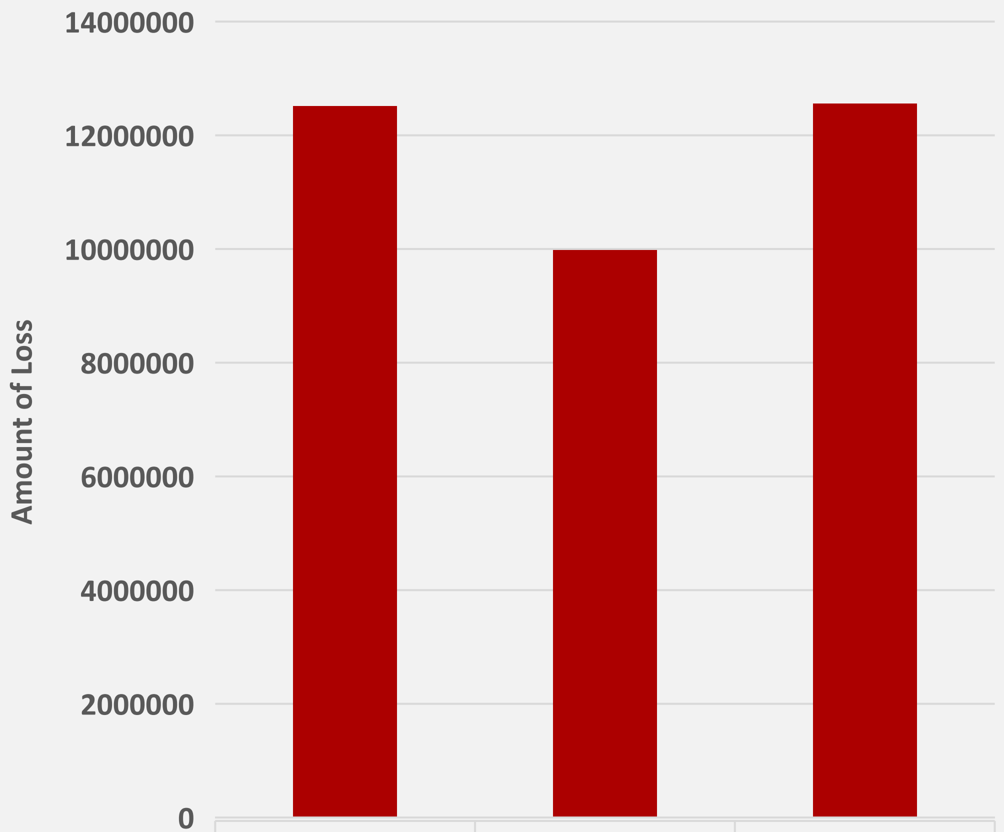
The logistic regression model can detect 393 people who are now defaulted.

logistic regression can reduce default rating by:

44.8%

MODEL TESTING

Comparative analysis of the results of predicting the total customers at risk of default for each model with real data



	LGBM	LOGREG	REAL
Total Loss ($)	12513933.9	9982558.98	12557193.66

The logistic Regression Model reduced the total loss of $2574634.68 from defaulted customers.

Logistic regression can reduce rating losses by:

20.6%

05

BUSINESS RECOMMENDATiON

BUSINESS RECOMMENDATION

- Pay special attention to the AMT_BALANCE, AMT_PAYMENT_TOTAL_CURRENT, AMT_LIMIT_CREDIT_ACTUAL features for the amount of funds submitted.

- Provide notification of payment deadlines to customers to avoid customers being late in making payments.

- Provide warnings to customers who cross the payment limit to be followed up on discussing payment scheme solutions.

Thanks

HOME CREDIT