MMF2030 Course Project

Joshua Kim

Ian Lee

Garrett McNally

Michael Fecenko

Prithvi Rosunee

February 3, 2023

Contents

1 Data Processing

1.1 Null and Missing Data

The first step in data processing was to load the data in and do some exploratory data analysis. Firstly, missing data was checked. By counting the number of null values in the main training dataset and sorting columns by whichever has the highest proportion of missing, it was discovered that there are a fair number of columns that only contain 1/3 of the data. When investigating the columns themselves, it seems the majority of these features that are missing are neighborhood and living area averages and statistics and therefore are not imperative in our analysis.

```
# checking missing data
total_null = train.isnull().count()
percent = (train.isnull().sum()/train.isnull().count()*100)
missing_check  = pd.concat([total_null, percent], axis=1, keys=['Total Null', 'Percent']).
    sort_values(ascending = False, by = ['Percent'])
missing_check.head(10)
```

	Total Null	Percent
COMMONAREA_MEDI	307511	69.872297
COMMONAREA_AVG	307511	69.872297
COMMONAREA_MODE	307511	69.872297
NONLIVINGAPARTMENTS_MODE	307511	69.432963
NONLIVINGAPARTMENTS_AVG	307511	69.432963
NONLIVINGAPARTMENTS_MEDI	307511	69.432963
FONDKAPREMONT_MODE	307511	68.386172
LIVINGAPARTMENTS_MODE	307511	68.354953
LIVINGAPARTMENTS_AVG	307511	68.354953
LIVINGAPARTMENTS_MEDI	307511	68.354953

1.2 Exploratory Data Analysis

Next we examine the feature set and familiarize ourselves with the data. By utilizing the data dictionary in the appendix, intuition can be built on the variables, their meaning and how they may affect the target. For categorical and ordinal variables the distribution is analyzed through tables, and continuous variables can be investigated through KDE plots. We investigate the effect on the target of a couple of created features

further on. Below we can see an example of the makeup of a feature in a table:

	TARGET	
	pop_rate	count
NAME_TYPE_SUITE		
Children	0.073768	3267
Family	0.074946	40149
Group of people	0.084871	271
Other_A	0.087760	866
Other_B	0.098305	1770
Spouse, partner	0.078716	11370
Unaccompanied	0.081830	248526

Figure 1: Investigating the population rate for the type of suite.

From analyzing the values of population rate for the target we can see that the highest rate of default is within the "Other_B" category. However with a relatively smaller sample size of 1770, it is hard to definitely say that it deviates significantly from the population default rate. We can see from this data that the most common type of dwelling is unaccompanied, implying that that most people applying to a loan in our set live alone by themselves.

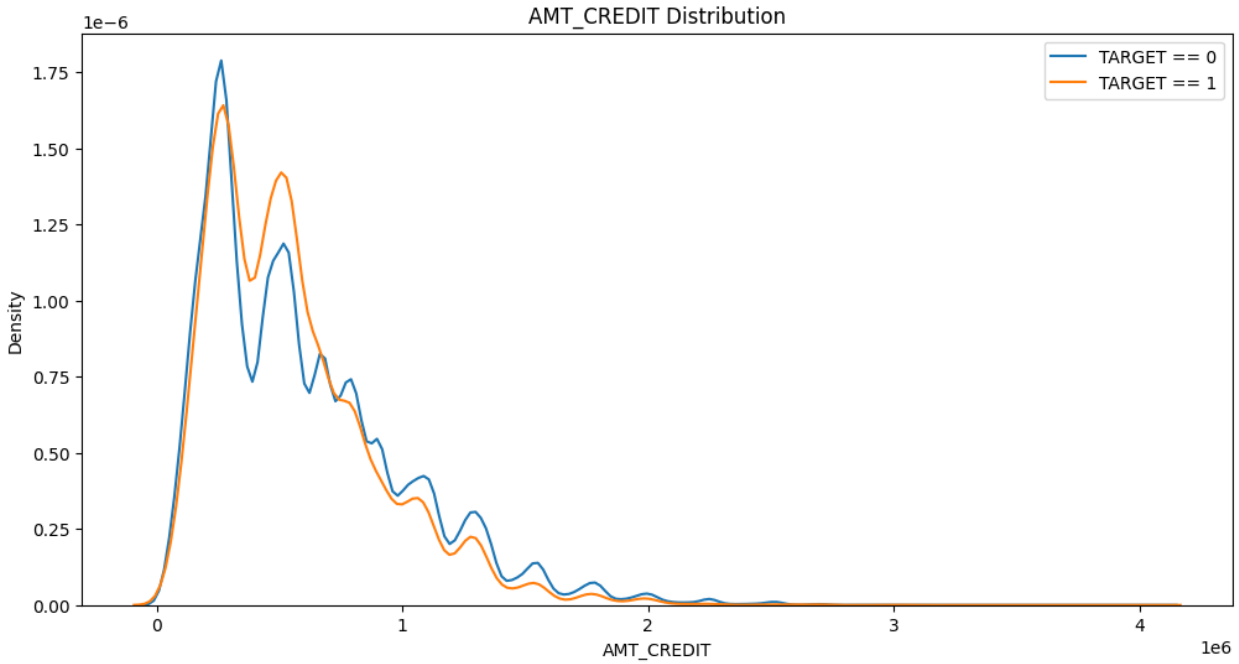


Figure 2: KDE plot for the AMT_CREDIT variable.

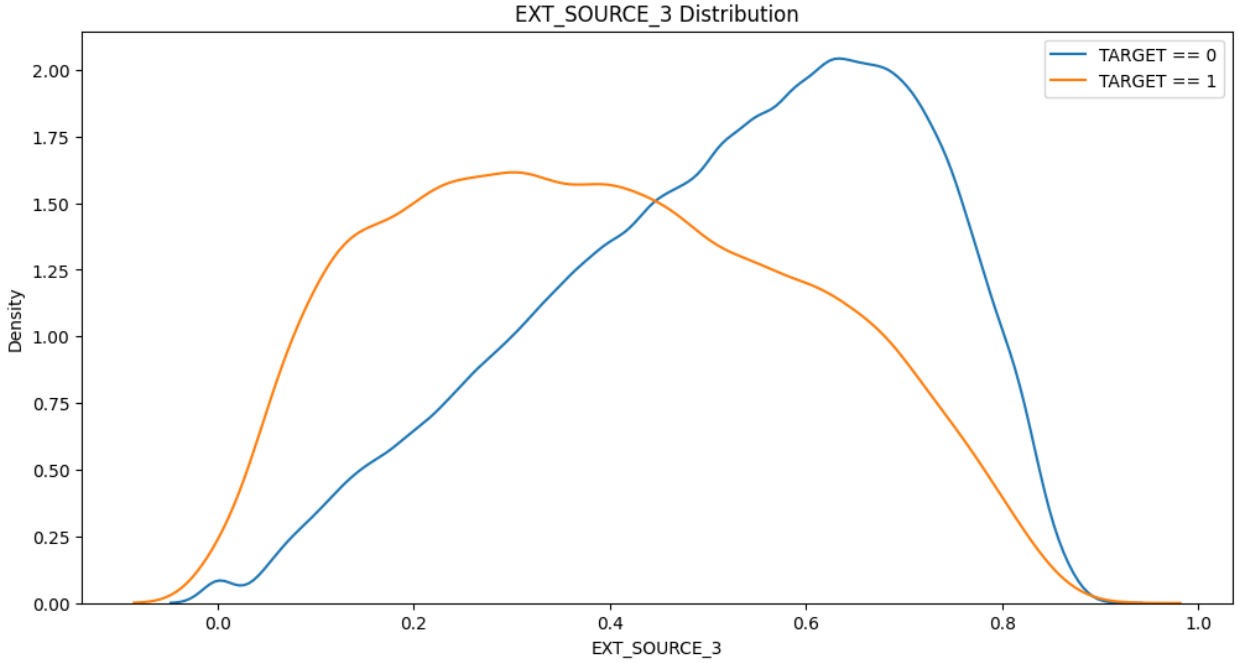


Figure 3: KDE plot for the EXT_SOURCE_3 variable.

Analyzing the two plots above we see the some variables are more effective in determine the outcome of the

target than others. In the case of the AMT_CREDIT variable the two distributions are very similar, while in the case of EXT_SOURCE_3 those who have defaulted have a much lower value on average then those who have not. This is also effective for trying to gauge the effectiveness of new features we may create, before actually filtering them in the model building process.

1.3 Feature Engineering

Another step in the data processing pipeline is to create new features. For each additional data set we employed the same approach. Firstly, using the data dictionary, we investigated the data and how it linked back to the initial train set. Second once intuition was built, features we felt might have an impact on the target were created by grouping the external data by some aggregate on the SK_CURR_ID to map back to the train. One such example is the *credit_limit* variable. To obtain this value, the maximum value of the most recent entry in was taken, and like the other variables we created, joined into our final cleaned set. Below is a snippet of the code for creating the features from the credit card balance csv:

```
cash_balance_feats = cash_balance.groupby('SK_ID_CURR') \
  .apply(lambda x: pd.Series({
      'amt_balance': x['AMT_BALANCE'].sum(),
      'amt_balance_0_1_year': x['AMT_BALANCE'][x['MONTHS_BALANCE'] >= -12].sum(),
      'amt_balance_1_2_year': x['AMT_BALANCE'][(x['MONTHS_BALANCE'] < -12) & (x['
    MONTHS_BALANCE'] >= -24)].sum(),
      'amt_recivable': x['AMT_TOTAL_RECEIVABLE'].sum(),
      'dpd': x['SK_DPD_DEF'].mean(),
      'credit_limit': x['AMT_CREDIT_LIMIT_ACTUAL'][x['MONTHS_BALANCE'].idxmax()]
  })
)
```

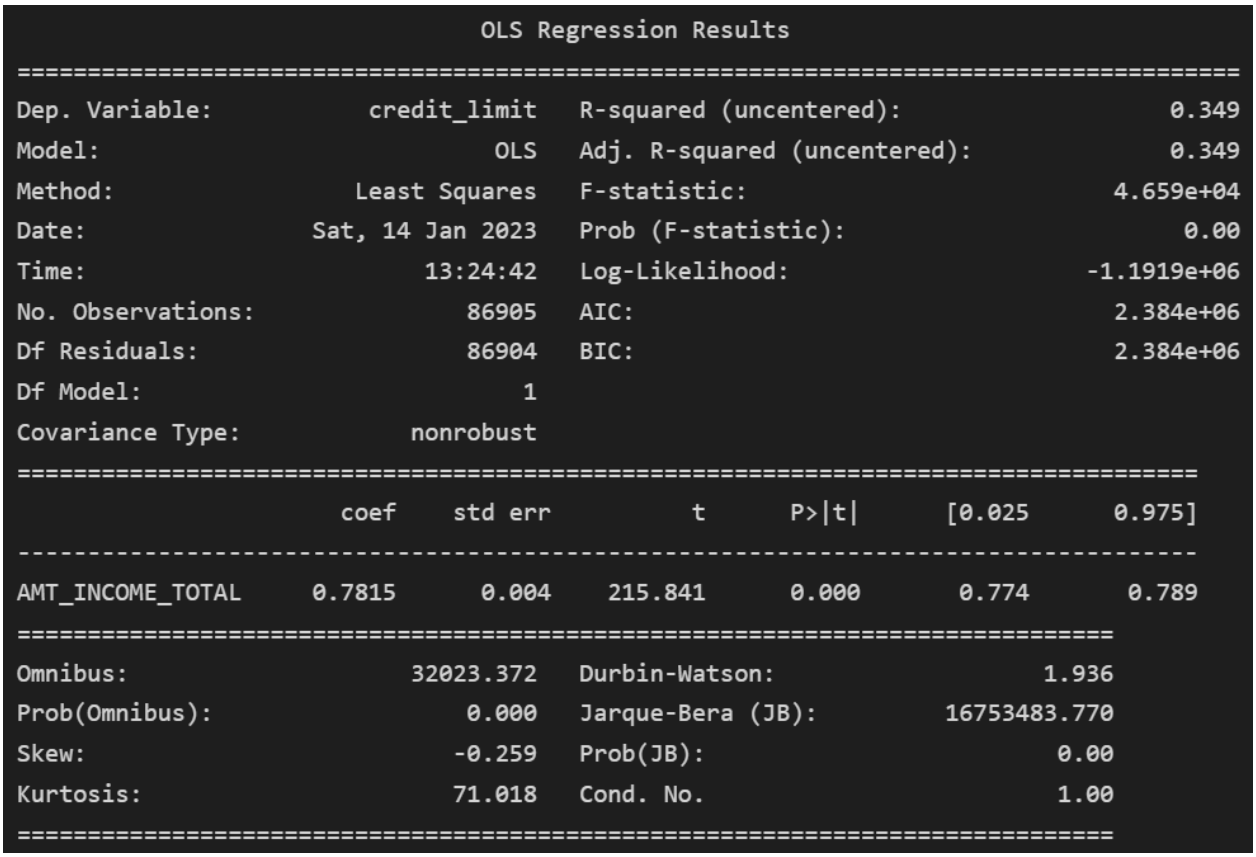
For variables created this way, there was not data to join for every record in the set. Thus, the method for imputing for each of our features needed to be determine. At first, all missing data was going to be imputed with zero. Going over the new features once more, this made sense as a logical fit, except for the aforementioned "credit limit" column. In this case, to impute for missing data, we trained an OLS model on the data that we did had and used prediction from that model to fill in the rest. At this point we have a full data set, with no missing or NA values.

1.4 Imputing Missing Data

When dealing with imputing missing data, we considered each feature that was created to understand the strategic use case for each. Mostly, we found that imputing missing data with 0 was a reasonable choice. However, for *credit_limit* we took a different approach. With this feature being the most recent credit limit that the client has on file, we found that imputing values of 0 for all missing data of *credit_limit* would be a poor choice. Since we found that *income_amount* has complete data, we build a simple linear regression model using all the available data for *credit_limit* as the train set and using the model to fill in missing values of *credit_limit* as a linear function of *income_amount*. The code can be seen below.

```
#build OLS model Y~X where credit limit ~ amt income total
data = df[["SK_ID_CURR","credit_limit","AMT_INCOME_TOTAL"]]
train = data[data["credit_limit"].notnull()]
x = train["AMT_INCOME_TOTAL"]
y = train["credit_limit"]
model = sm.OLS(y,x)
model_result = model.fit()

#predict results on main dataset
df["credit_limit"] = df.apply(lambda row: model_result.predict(row['AMT_INCOME_TOTAL'])[0]
    if row["credit_limit"] np.isnull() else row["credit_limit"], axis=1)
```



OLS Regression Results

Dep. Variable:	credit_limit	R-squared (uncentered):	0.349
Model:	OLS	Adj. R-squared (uncentered):	0.349
Method:	Least Squares	F-statistic:	4.659e+04
Date:	Sat, 14 Jan 2023	Prob (F-statistic):	0.00
Time:	13:24:42	Log-Likelihood:	-1.1919e+06
No. Observations:	86905	AIC:	2.384e+06
Df Residuals:	86904	BIC:	2.384e+06
Df Model:	1		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
AMT_INCOME_TOTAL	0.7815	0.004	215.841	0.000	0.774	0.789

Omnibus:	32023.372	Durbin-Watson:	1.936
Prob(Omnibus):	0.000	Jarque-Bera (JB):	16753483.770
Skew:	-0.259	Prob(JB):	0.00
Kurtosis:	71.018	Cond. No.	1.00

Figure 4: Regression results for imputing.

1.5 Documentation of Quality Assurance

In terms of Quality assurance techniques and steps taken, for each of the groupings on external data sets, we isolated for a specific ID as a sample and ran the calculations ourselves to make sure our data transforms were performing as expected. By doing this we protect against model risk and save time by not having to go back to fix odd data or features. Another step done throughout is using the pandas function .describe() when importing new data or after our aggregations have completed. At each point we can check percentile metrics as well as count, mean and standard deviation to ensure that everything is working as intended.

	paid_in_full	pmt_tardiness
count	3.395870e+05	339587.000000
mean	7.496631e+03	-351.988292
std	1.777886e+05	508.732820
min	-3.037736e+06	-10529.000000
25%	-1.570462e+04	-472.000000
50%	0.000000e+00	-242.000000
75%	0.000000e+00	-117.000000
max	4.417384e+06	82905.000000

Figure 5: Describe after creating features.

In the above figure, we see get descriptive statistics for the two features created from the installment payments external data. We can see for the paid in full that most of the data is at zero, indicating loans are fully paid back. In the payment tardiness feature we can see that most of the data indicates that customers consistently pay off their balances after they are due, thus accumulating a higher negative value. What we also can see is that the min and max figure for both features seem odd, so we investigate those cases in more detail.

	pmt	time
SK_ID_PREV		
1319813	-1673353.845	-35.0
1613832	0.000	-5.0
1833116	0.000	-294.0
2105653	-142145.100	-237.0
2210909	-329278.950	-30.0
2776764	0.000	-24.0
2833195	-892957.680	-81.0

Figure 6: Checking minimum data validity

From this figure we take the current ID with the lowest paid in full value and investigate by grouping the data by each previous ID. All of these values point to the individual paying late. The very large negative values for the paid in full raise some red flags. One last step we can take is to look at individual entries for one previous ID.

	DAYS_INSTALMENT	DAYS_ENTRY_PAYMENT	AMT_INSTALMENT	AMT_PAYMENT
3800113	-19.0	-53.0	72754.515	729.270
4016874	-109.0	-140.0	72754.515	67.815
4162543	-289.0	-288.0	72754.515	31657.905
4196365	-559.0	-554.0	72754.515	18753.975
4213831	-379.0	-378.0	72754.515	55048.950
4232172	-319.0	-343.0	72754.515	12.420
4275261	-289.0	-285.0	72754.515	41096.610
4311577	-559.0	-557.0	72754.515	54000.000
4631647	-349.0	-343.0	72754.515	18460.080
4686792	-199.0	-207.0	72754.515	72754.515

Figure 7: Checking minimum data for single previous ID

We can see that from a sample of 10 rows for this specific previous ID, the customer on paid the full installment once. As well was either late or on time, and never early for a payment. These values point to the grouped values being correct. One final approach would be to remove the a top and bottom percentile of the data for the continuous variables to help deal with outliers. In the sake of time and for other steps of the model process this exercise was omitted.

Finally, in line with many jurisdictions, we remove demographic data from our data set. While demographics data do improve the performance of the model, this discriminates against some applicants unfairly. In a pure business standpoint, there can be blow backs and repercussions from building a model with discrimination built into it. More importantly, from a ethical standpoint, this is not right and thus we remove data such as marital status and gender.

1.6 Final Data Dictionary

In this section we include a list of all created features. A complete list of all 262 features used in our models is provided later in the appendix.

Feature	Description
NAME_CONTRACT_TYPE_Cash loans	Identification if loan is cash or revolving
NAME_CONTRACT_TYPE_Revolving loans	Identification if loan is cash or revolving
CODE_GENDER_F	Gender of the client
CODE_GENDER_M	Gender of the client
CODE_GENDER_XNA	Gender of the client
FLAG_OWN_CAR_N	Flag if the client owns a car
FLAG_OWN_CAR_Y	Flag if the client owns a car
FLAG_OWN_REALTY_N	Flag if client owns a house or flat
FLAG_OWN_REALTY_Y	Flag if client owns a house or flat
NAME_TYPE_SUITE_Children	Normalized information about building where the client lives, What is average (_AVG suffix), modus (_MODE suffix), median (_MEDI suffix) apartment size, common area, living area, age of building, number of elevators, number of entrances, state of the building, number of floor
NAME_TYPE_SUITE_Family	Normalized information about building where the client lives, What is average (_AVG suffix), modus (_MODE suffix), median (_MEDI suffix) apartment size, common area, living area, age of building, number of elevators, number of entrances, state of the building, number of floor

NAME_TYPE_SUITE_Group of people	Normalized information about building where the client lives, What is average (_AVG suffix), modus (_MODE suffix), median (_MEDI suffix) apartment size, common area, living area, age of building, number of elevators, number of entrances, state of the building, number of floor
NAME_TYPE_SUITE_Other_A	Normalized information about building where the client lives, What is average (_AVG suffix), modus (_MODE suffix), median (_MEDI suffix) apartment size, common area, living area, age of building, number of elevators, number of entrances, state of the building, number of floor
NAME_TYPE_SUITE_Other_B	Normalized information about building where the client lives, What is average (_AVG suffix), modus (_MODE suffix), median (_MEDI suffix) apartment size, common area, living area, age of building, number of elevators, number of entrances, state of the building, number of floor
NAME_TYPE_SUITE_Spouse, partner	Normalized information about building where the client lives, What is average (_AVG suffix), modus (_MODE suffix), median (_MEDI suffix) apartment size, common area, living area, age of building, number of elevators, number of entrances, state of the building, number of floor
NAME_TYPE_SUITE_Unaccompanied	Normalized information about building where the client lives, What is average (_AVG suffix), modus (_MODE suffix), median (_MEDI suffix) apartment size, common area, living area, age of building, number of elevators, number of entrances, state of the building, number of floor

NAME_INCOME_TYPE_Businessman	Normalized information about building where the client lives, What is average (_AVG suffix), modus (_MODE suffix), median (_MEDI suffix) apartment size, common area, living area, age of building, number of elevators, number of entrances, state of the building, number of floor
NAME_INCOME_TYPE_Commercial associate	Normalized information about building where the client lives, What is average (_AVG suffix), modus (_MODE suffix), median (_MEDI suffix) apartment size, common area, living area, age of building, number of elevators, number of entrances, state of the building, number of floor
NAME_INCOME_TYPE_Maternity leave	Normalized information about building where the client lives, What is average (_AVG suffix), modus (_MODE suffix), median (_MEDI suffix) apartment size, common area, living area, age of building, number of elevators, number of entrances, state of the building, number of floor
NAME_INCOME_TYPE_Pensioner	Normalized information about building where the client lives, What is average (_AVG suffix), modus (_MODE suffix), median (_MEDI suffix) apartment size, common area, living area, age of building, number of elevators, number of entrances, state of the building, number of floor
NAME_INCOME_TYPE_State servant	Normalized information about building where the client lives, What is average (_AVG suffix), modus (_MODE suffix), median (_MEDI suffix) apartment size, common area, living area, age of building, number of elevators, number of entrances, state of the building, number of floor

NAME_INCOME_TYPE_Student	Normalized information about building where the client lives, What is average (_AVG suffix), modus (_MODE suffix), median (_MEDI suffix) apartment size, common area, living area, age of building, number of elevators, number of entrances, state of the building, number of floor
NAME_INCOME_TYPE_Unemployed	Normalized information about building where the client lives, What is average (_AVG suffix), modus (_MODE suffix), median (_MEDI suffix) apartment size, common area, living area, age of building, number of elevators, number of entrances, state of the building, number of floor
NAME_INCOME_TYPE_Working	Normalized information about building where the client lives, What is average (_AVG suffix), modus (_MODE suffix), median (_MEDI suffix) apartment size, common area, living area, age of building, number of elevators, number of entrances, state of the building, number of floor
NAME_EDUCATION_TYPE_Academic degree	Normalized information about building where the client lives, What is average (_AVG suffix), modus (_MODE suffix), median (_MEDI suffix) apartment size, common area, living area, age of building, number of elevators, number of entrances, state of the building, number of floor
NAME_EDUCATION_TYPE_Higher education	Normalized information about building where the client lives, What is average (_AVG suffix), modus (_MODE suffix), median (_MEDI suffix) apartment size, common area, living area, age of building, number of elevators, number of entrances, state of the building, number of floor

NAME_EDUCATION_TYPE_Incomplete higher	Normalized information about building where the client lives, What is average (_AVG suffix), modus (_MODE suffix), median (_MEDI suffix) apartment size, common area, living area, age of building, number of elevators, number of entrances, state of the building, number of floor
NAME_EDUCATION_TYPE_Lower secondary	Normalized information about building where the client lives, What is average (_AVG suffix), modus (_MODE suffix), median (_MEDI suffix) apartment size, common area, living area, age of building, number of elevators, number of entrances, state of the building, number of floor
NAME_EDUCATION_TYPE_Secondary / secondary special	Normalized information about building where the client lives, What is average (_AVG suffix), modus (_MODE suffix), median (_MEDI suffix) apartment size, common area, living area, age of building, number of elevators, number of entrances, state of the building, number of floor
NAME_FAMILY_STATUS_Civil marriage	Normalized information about building where the client lives, What is average (_AVG suffix), modus (_MODE suffix), median (_MEDI suffix) apartment size, common area, living area, age of building, number of elevators, number of entrances, state of the building, number of floor
NAME_FAMILY_STATUS_Married	Normalized information about building where the client lives, What is average (_AVG suffix), modus (_MODE suffix), median (_MEDI suffix) apartment size, common area, living area, age of building, number of elevators, number of entrances, state of the building, number of floor

NAME_FAMILY_STATUS_Separated	Normalized information about building where the client lives, What is average (_AVG suffix), modus (_MODE suffix), median (_MEDI suffix) apartment size, common area, living area, age of building, number of elevators, number of entrances, state of the building, number of floor
NAME_FAMILY_STATUS_Single / not married	Normalized information about building where the client lives, What is average (_AVG suffix), modus (_MODE suffix), median (_MEDI suffix) apartment size, common area, living area, age of building, number of elevators, number of entrances, state of the building, number of floor
NAME_FAMILY_STATUS_Unknown	Normalized information about building where the client lives, What is average (_AVG suffix), modus (_MODE suffix), median (_MEDI suffix) apartment size, common area, living area, age of building, number of elevators, number of entrances, state of the building, number of floor
NAME_FAMILY_STATUS_Widow	Normalized information about building where the client lives, What is average (_AVG suffix), modus (_MODE suffix), median (_MEDI suffix) apartment size, common area, living area, age of building, number of elevators, number of entrances, state of the building, number of floor
NAME_HOUSING_TYPE_Co-op apartment	Normalized information about building where the client lives, What is average (_AVG suffix), modus (_MODE suffix), median (_MEDI suffix) apartment size, common area, living area, age of building, number of elevators, number of entrances, state of the building, number of floor

NAME_HOUSING_TYPE_House / apartment	Normalized information about building where the client lives, What is average (_AVG suffix), modus (_MODE suffix), median (_MEDI suffix) apartment size, common area, living area, age of building, number of elevators, number of entrances, state of the building, number of floor
NAME_HOUSING_TYPE_Municipal apartment	Normalized information about building where the client lives, What is average (_AVG suffix), modus (_MODE suffix), median (_MEDI suffix) apartment size, common area, living area, age of building, number of elevators, number of entrances, state of the building, number of floor
NAME_HOUSING_TYPE_Office apartment	Normalized information about building where the client lives, What is average (_AVG suffix), modus (_MODE suffix), median (_MEDI suffix) apartment size, common area, living area, age of building, number of elevators, number of entrances, state of the building, number of floor
NAME_HOUSING_TYPE_Rented apartment	Normalized information about building where the client lives, What is average (_AVG suffix), modus (_MODE suffix), median (_MEDI suffix) apartment size, common area, living area, age of building, number of elevators, number of entrances, state of the building, number of floor
NAME_HOUSING_TYPE_With parents	Normalized information about building where the client lives, What is average (_AVG suffix), modus (_MODE suffix), median (_MEDI suffix) apartment size, common area, living area, age of building, number of elevators, number of entrances, state of the building, number of floor
OCCUPATION_TYPE_Accountants	What kind of occupation does the client have
OCCUPATION_TYPE_Cleaning staff	What kind of occupation does the client have
OCCUPATION_TYPE_Cooking staff	What kind of occupation does the client have

OCCUPATION_TYPE_Core staff	What kind of occupation does the client have
OCCUPATION_TYPE_Drivers	What kind of occupation does the client have
OCCUPATION_TYPE_HR staff	What kind of occupation does the client have
OCCUPATION_TYPE_High skill tech staff	What kind of occupation does the client have
OCCUPATION_TYPE_IT staff	What kind of occupation does the client have
OCCUPATION_TYPE_Laborers	What kind of occupation does the client have
OCCUPATION_TYPE_Low-skill Laborers	What kind of occupation does the client have
OCCUPATION_TYPE_Managers	What kind of occupation does the client have
OCCUPATION_TYPE_Medicine staff	What kind of occupation does the client have
OCCUPATION_TYPE_Private service staff	What kind of occupation does the client have
OCCUPATION_TYPE_Realty agents	What kind of occupation does the client have
OCCUPATION_TYPE_Sales staff	What kind of occupation does the client have
OCCUPATION_TYPE_Secretaries	What kind of occupation does the client have
OCCUPATION_TYPE_Security staff	What kind of occupation does the client have
OCCUPATION_TYPE_Waiters/barmen staff	What kind of occupation does the client have
WEEKDAY_APPR_PROCESS_START_FRIDAY	On which day of the week did the client apply for the loan
WEEKDAY_APPR_PROCESS_START_MONDAY	On which day of the week did the client apply for the loan
WEEKDAY_APPR_PROCESS_START_SATURDAY	On which day of the week did the client apply for the loan
WEEKDAY_APPR_PROCESS_START_SUNDAY	On which day of the week did the client apply for the loan
WEEKDAY_APPR_PROCESS_START_THURSDAY	On which day of the week did the client apply for the loan
WEEKDAY_APPR_PROCESS_START_TUESDAY	On which day of the week did the client apply for the loan
WEEKDAY_APPR_PROCESS_START_WEDNESDAY	On which day of the week did the client apply for the loan
ORGANIZATION_TYPE_Advertising	Type of organization where client works

ORGANIZATION_TYPE_Agriculture	Type of organization where client works
ORGANIZATION_TYPE_Bank	Type of organization where client works
ORGANIZATION_TYPE_Business Entity Type 1	Type of organization where client works
ORGANIZATION_TYPE_Business Entity Type 2	Type of organization where client works
ORGANIZATION_TYPE_Business Entity Type 3	Type of organization where client works
ORGANIZATION_TYPE_Cleaning	Type of organization where client works
ORGANIZATION_TYPE_Construction	Type of organization where client works
ORGANIZATION_TYPE_Culture	Type of organization where client works
ORGANIZATION_TYPE_Electricity	Type of organization where client works
ORGANIZATION_TYPE_Emergency	Type of organization where client works
ORGANIZATION_TYPE_Government	Type of organization where client works
ORGANIZATION_TYPE_Hotel	Type of organization where client works
ORGANIZATION_TYPE_Housing	Type of organization where client works
ORGANIZATION_TYPE_Industry: type 1	Type of organization where client works
ORGANIZATION_TYPE_Industry: type 10	Type of organization where client works
ORGANIZATION_TYPE_Industry: type 11	Type of organization where client works
ORGANIZATION_TYPE_Industry: type 12	Type of organization where client works
ORGANIZATION_TYPE_Industry: type 13	Type of organization where client works
ORGANIZATION_TYPE_Industry: type 2	Type of organization where client works

ORGANIZATION_TYPE_Industry: type 3	Type of organization where client works
ORGANIZATION_TYPE_Industry: type 4	Type of organization where client works
ORGANIZATION_TYPE_Industry: type 5	Type of organization where client works
ORGANIZATION_TYPE_Industry: type 6	Type of organization where client works
ORGANIZATION_TYPE_Industry: type 7	Type of organization where client works
ORGANIZATION_TYPE_Industry: type 8	Type of organization where client works
ORGANIZATION_TYPE_Industry: type 9	Type of organization where client works
ORGANIZATION_TYPE_Insurance	Type of organization where client works
ORGANIZATION_TYPE_Kindergarten	Type of organization where client works
ORGANIZATION_TYPE_Legal Services	Type of organization where client works
ORGANIZATION_TYPE_Medicine	Type of organization where client works
ORGANIZATION_TYPE_Military	Type of organization where client works
ORGANIZATION_TYPE_Mobile	Type of organization where client works
ORGANIZATION_TYPE_Other	Type of organization where client works
ORGANIZATION_TYPE_Police	Type of organization where client works
ORGANIZATION_TYPE_Postal	Type of organization where client works
ORGANIZATION_TYPE_Realtor	Type of organization where client works
ORGANIZATION_TYPE_Religion	Type of organization where client works
ORGANIZATION_TYPE_Restaurant	Type of organization where client works
ORGANIZATION_TYPE_School	Type of organization where client works
ORGANIZATION_TYPE_Security	Type of organization where client works
ORGANIZATION_TYPE_Security Ministries	Type of organization where client works
ORGANIZATION_TYPE_Self-employed	Type of organization where client works

ORGANIZATION_TYPE_Services	Type of organization where client works
ORGANIZATION_TYPE_Telecom	Type of organization where client works
ORGANIZATION_TYPE_Trade: type 1	Type of organization where client works
ORGANIZATION_TYPE_Trade: type 2	Type of organization where client works
ORGANIZATION_TYPE_Trade: type 3	Type of organization where client works
ORGANIZATION_TYPE_Trade: type 4	Type of organization where client works
ORGANIZATION_TYPE_Trade: type 5	Type of organization where client works
ORGANIZATION_TYPE_Trade: type 6	Type of organization where client works
ORGANIZATION_TYPE_Trade: type 7	Type of organization where client works
ORGANIZATION_TYPE_Transport: type 1	Type of organization where client works
ORGANIZATION_TYPE_Transport: type 2	Type of organization where client works
ORGANIZATION_TYPE_Transport: type 3	Type of organization where client works
ORGANIZATION_TYPE_Transport: type 4	Type of organization where client works
ORGANIZATION_TYPE_University	Type of organization where client works
ORGANIZATION_TYPE_XNA	Type of organization where client works
FONDKAPREMONT_MODE_not specified	Normalized information about building where the client lives, What is average (_AVG suffix), modus (_MODE suffix), median (_MEDI suffix) apartment size, common area, living area, age of building, number of elevators, number of entrances, state of the building, number of floor

FONDKAPREMONT_MODE_org spec account	Normalized information about building where the client lives, What is average (_AVG suffix), modus (_MODE suffix), median (_MEDI suffix) apartment size, common area, living area, age of building, number of elevators, number of entrances, state of the building, number of floor
FONDKAPREMONT_MODE_reg oper account	Normalized information about building where the client lives, What is average (_AVG suffix), modus (_MODE suffix), median (_MEDI suffix) apartment size, common area, living area, age of building, number of elevators, number of entrances, state of the building, number of floor
FONDKAPREMONT_MODE_reg oper spec account	Normalized information about building where the client lives, What is average (_AVG suffix), modus (_MODE suffix), median (_MEDI suffix) apartment size, common area, living area, age of building, number of elevators, number of entrances, state of the building, number of floor
HOUSETYPE_MODE_block of flats	Normalized information about building where the client lives, What is average (_AVG suffix), modus (_MODE suffix), median (_MEDI suffix) apartment size, common area, living area, age of building, number of elevators, number of entrances, state of the building, number of floor
HOUSETYPE_MODE_specific housing	Normalized information about building where the client lives, What is average (_AVG suffix), modus (_MODE suffix), median (_MEDI suffix) apartment size, common area, living area, age of building, number of elevators, number of entrances, state of the building, number of floor

HOUSETYPE_MODE_terraced house	Normalized information about building where the client lives, What is average (_AVG suffix), modus (_MODE suffix), median (_MEDI suffix) apartment size, common area, living area, age of building, number of elevators, number of entrances, state of the building, number of floor
WALLSMATERIAL_MODE_Block	Normalized information about building where the client lives, What is average (_AVG suffix), modus (_MODE suffix), median (_MEDI suffix) apartment size, common area, living area, age of building, number of elevators, number of entrances, state of the building, number of floor
WALLSMATERIAL_MODE_Mixed	Normalized information about building where the client lives, What is average (_AVG suffix), modus (_MODE suffix), median (_MEDI suffix) apartment size, common area, living area, age of building, number of elevators, number of entrances, state of the building, number of floor
WALLSMATERIAL_MODE_Monolithic	Normalized information about building where the client lives, What is average (_AVG suffix), modus (_MODE suffix), median (_MEDI suffix) apartment size, common area, living area, age of building, number of elevators, number of entrances, state of the building, number of floor
WALLSMATERIAL_MODE_Others	Normalized information about building where the client lives, What is average (_AVG suffix), modus (_MODE suffix), median (_MEDI suffix) apartment size, common area, living area, age of building, number of elevators, number of entrances, state of the building, number of floor

WALLSMATERIAL_MODE_Panel	Normalized information about building where the client lives, What is average (_AVG suffix), modus (_MODE suffix), median (_MEDI suffix) apartment size, common area, living area, age of building, number of elevators, number of entrances, state of the building, number of floor
WALLSMATERIAL_MODE_Stone, brick	Normalized information about building where the client lives, What is average (_AVG suffix), modus (_MODE suffix), median (_MEDI suffix) apartment size, common area, living area, age of building, number of elevators, number of entrances, state of the building, number of floor
WALLSMATERIAL_MODE_Wooden	Normalized information about building where the client lives, What is average (_AVG suffix), modus (_MODE suffix), median (_MEDI suffix) apartment size, common area, living area, age of building, number of elevators, number of entrances, state of the building, number of floor
EMERGENCYSTATE_MODE_No	Normalized information about building where the client lives, What is average (_AVG suffix), modus (_MODE suffix), median (_MEDI suffix) apartment size, common area, living area, age of building, number of elevators, number of entrances, state of the building, number of floor
EMERGENCYSTATE_MODE_Yes	Normalized information about building where the client lives, What is average (_AVG suffix), modus (_MODE suffix), median (_MEDI suffix) apartment size, common area, living area, age of building, number of elevators, number of entrances, state of the building, number of floor
active_credit	Amount of active credit the client has
active_debt	Amount of active debt the client has
active_products	Number of active products the client has

credit_card	Number of active credit cards the client has
mortgage	Number of mortgages the client has
days_overdue	The number of days overdue the client was in total.
credit_sum_overdue	The sum of all credits that were overdue for the client
paid_in_full	Whether all payments where made in full, leftover value. Note that a negative value is paid early, 0 is paid on time and positive value is paid late
pmt_tardiness	Total value how early or late the client made payments
amt_balance_0_1_year	Balance during the month of previous credit, summed over the last 1 year
amt_balance_1_2_year	Balance during the month of previous credit, summed over the period between 1-2 years ago
dpd	Average DPD (Days past due) for all client records with tolerance (debts with low loan amounts are ignored)
credit_limit	The last reported credit limit for the client

2 Model Development

2.1 Model Training

The model selection process was facilitated by *H2O*, a Python library offering an open-source automated machine learning framework, i.e. *AutoML*. AutoML is a process of automating various stages of machine learning from data preparation to model deployment. Since the project boils down to a classic binary classification problem using tabular data, a heavily studied topic in the field of supervised learning, such task is where AutoML excels. The pool of machine learning algorithms from which *H2O* AutoML selected are:

- generalized linear model with regularization (GLM)
- distributed random forest (DRF)

- extremely randomized trees (XRT)
- gradient boosting machines (GBM)
- extreme gradient boosting (XGBoost)

While the library also provides deep neural network and stacked ensemble, they were manually excluded from the pool because the two models lack interpretability and require excessive training time compared to the other linear or tree-based algorithms. A convenient feature of *H2O* AutoML is the hyperparameter optimization using random grid search. The library documentation lists the hyperparameters of each base model, along with all potential values that can be randomly chosen in the search. For example, *Figure 8* is an exhaustive list of XGBoost hyperparameters and their corresponding searchable values.

Parameter	Searchable Values
`booster`	`gbtree` , `dart`
`col_sample_rate`	`{0.6, 0.8, 1.0}`
`col_sample_rate_per_tree`	`{0.7, 0.8, 0.9, 1.0}`
`max_depth`	`{5, 10, 15, 20}`
`min_rows`	`{0.01, 0.1, 1.0, 3.0, 5.0, 10.0, 15.0, 20.0}`
`ntrees`	Hard coded: `10000` (true value found by early stopping)
`reg_alpha`	`{0.001, 0.01, 0.1, 1, 10, 100}`
`reg_lambda`	`{0.001, 0.01, 0.1, 0.5, 1}`
`sample_rate`	`{0.6, 0.8, 1.0}`

Figure 8: *XGBoost hyperparameters and their searchable values under H2O AutoML grid search*

Prior to training a classification model, additional training data preparation was carried out beyond the feature engineering steps described in *Section 1*. Most importantly, data was stored as *H2OFrame* objects to be compatible with the rest of the *H2O* framework, shown in *Figure 9*. We then dropped any constant variables, i.e. columns with one unique value, to reduce potential overhead by including these uninformative features in the training process. For the project, we chose 80:20 split (80% of the data for training, 20% for testing), a commonly used split ratio in practice. The purpose of train-test split is to obtain an unbiased evaluation of a trained model with a dataset that was unused during training. If the training set were to be used to measure model performance, the evaluation would be overly optimistic since the model has already

seen the training set before. See *Figure 10* for Python code used to prepare training data.

```
# initialize H2O instance
h2o.init()

# set up H2O dataframe objects
train_hf = h2o.H2OFrame(pd.concat([X_train, y_train], axis=1))
train_hf["TARGET"] = train_hf["TARGET"].asfactor()
test_hf = h2o.H2OFrame(pd.concat([X_test, y_test], axis=1))
test_hf["TARGET"] = test_hf["TARGET"].asfactor()
```

Figure 9: *code for storing data in H2OFrame objects*

```
# load the cleaned dataset
df = pd.read_csv("MMF-ML-main/cleaned_train.csv", index_col=0)

# drop ID and target variable
X = df.drop(columns=["SK_ID_CURR", "TARGET"])

# drop constant columns
X = X.drop(columns=['FLAG_DOCUMENT_10', 'NAME_INCOME_TYPE_Businessman', 'FLAG_DOCUMENT_12', 'NAME_INCOME_TYPE_Student', 'FLAG_MOBIL'])

# drop demographic features (most of them have already been dropped during data preprocessing anyways)
X = X.drop(columns=['DAYS_BIRTH'])

# separate target variable
y = df["TARGET"]

# 8:2 train-test split
X_train, X_test, y_train, y_test = train_test_split(X, y, stratify=y, test_size=0.2, random_state=2030)
```

Figure 10: *training data preparation code in Python*

We ran the experiment for 20 model iterations, followed by the model selection stage where the AutoML framework chose the model with the highest 5-fold cross-validation ROC AUC as the final classifier. The `balance_classes` feature was enabled to ensure that each target class (0 and 1) are equally represented during the training of the model, shown in *Figure 11*. The final model is an XGBoost with the following hyperparameters:

- $number\_of\_trees = 76$
- $max\_depth = 3$
- $min\_rows = 1$
- $min\_child\_weight = 1$
- $learn\_rate = 0.3$

The log-loss value of the selected model evaluated using 5-fold cross validation was 0.2462.

```
# train using AutoML
aml = h2o.automl.H2OAutoML(max_models=20, seed=2030, balance_classes=True, exclude_algos=["DeepLearning", "StackedEnsemble"])
aml.train(training_frame=train_hf, y="TARGET")
```

Figure 11: *H2O AutoML training code in Python*

2.2 Model Evaluation

To evaluate the model, we measured ROC AUC (area under the ROC curve) calculated using 5-fold cross validation. ROC AUC assesses how well a binary classification model is able to distinguish between true and false positives. The ROC is a curve representing the change in TPR (true positive rate) according to the change in FPR (false positive rate). This curve visualizes the change in values of FPR and TPR, whose AUC is often used as a performance indicator of the classification model. An AUC of 1 indicates a perfect classifier, while an AUC of .5 indicates a poor classifier, whose performance is no better than random guessing. The AUC of 0.7582 suggests the our model is reasonably capable of classifying between high and non-high risk customers.

Summarized in *Table 2*, the four metrics - accuracy, precision, recall, and F_1-score - are measured to evaluate the quality of our model predictions, where they range between 0 (poor) and 1 (perfect). Accuracy is a performance metric that determines how close the predicted data is to the actual data. It is a very intuitive evaluation that displays the predictive power of a model. In the case of binary classification however, it can be unreasonable to evaluate accuracy alone because the performance of the model may be distorted depending on the structure of the data, namely the class imbalance.

To complement it, we also compare precision and recall to evaluate our model performance. Precision, also called positive predictive value, is the fraction of relevant instances among the retrieved instances, while recall, also known as sensitivity, is the fraction of relevant instances that were retrieved. Depending on the task, the relative importance of recall and precision may differ, If precision or recall needs special emphasis due to the nature of the task to be classified, the value of either metric can be increased by adjusting the classification decision threshold. Since they are complementary indicators however, if one is forcibly raised, the value of the other tends to drop. Hence we should not aim to maximize the precision (or recall) of a model because one can simply achieve 100% precision (or recall) by labelling everything as negatives (or positives). The F_1-score, calculated as the harmonic mean of the precision and recall, resolves this issue by striking the balance between the two metrics. It has a relatively high value when precision and recall are

not biased in either direction.

Table 2: *final model performance evaluation calculated using 5-fold cross validation*

model	ROC AUC	accuracy	precision	recall	F_1-score
baseline	0.6386	0.6471	0.6921	0.4402	0.5474
final	0.7582	0.8445	0.2503	0.4643	0.3252