Credit Risk Management

CRM Machine Learning

Home Credit data from Kaggle pre-processing

Chenyan Jiang

Supervisor:
Galina Andreeva

Sunday 13th November, 2022

Contents

1 Introduction

The report builds on coursework2 to further explore the risk of default on household loans. After a detailed analysis of the content of the documented data, a model was created to help us better analyse credit risk through previous loan records and personal information. The overall structure of the report is as follows. We **first review and refine the work of coursework2 in the first section**.Khandani et al.[1] used machine learning techniques to build a non-linear, non-parametric prediction of consumer credit risk, thus we follow similar process to build model.In the second section, we **detail the process and details of the model building** and measure the model's prediction using **evaluation criteria** such as **ROC_AUC**. In the **final third section**, we refine the **baseline model1** built in the first section and **compare it with the newly built models** (random forest, Ada Boosting, light gradient boosting).

2 Part 1 : Building & Validating Predictive Models

The data-set was gathered from Kaggle and related to home credit. In building the standard credit scoring model(referred to as the 'baseline' in short), I use the data('application_traintest.csv') after preprocessing in CW2. This dataset consists of seven CSV files with different aspects of user data. While in section 2, in order to improve the baseline and to compare different model, I merge the data-set by adding data from 'bureau.csv' and 'bureau_balance' to 'application.csv'. For data merge are followed the same primary key (unique identifier - ['SK_ID_CURR']) merging principle(Figure 1). We worked with only three of these, the master table of information provided by the customer at the time of application, the customer's past credit file from the credit bureau and the customer's past POS and cash loan files with Gitzo.

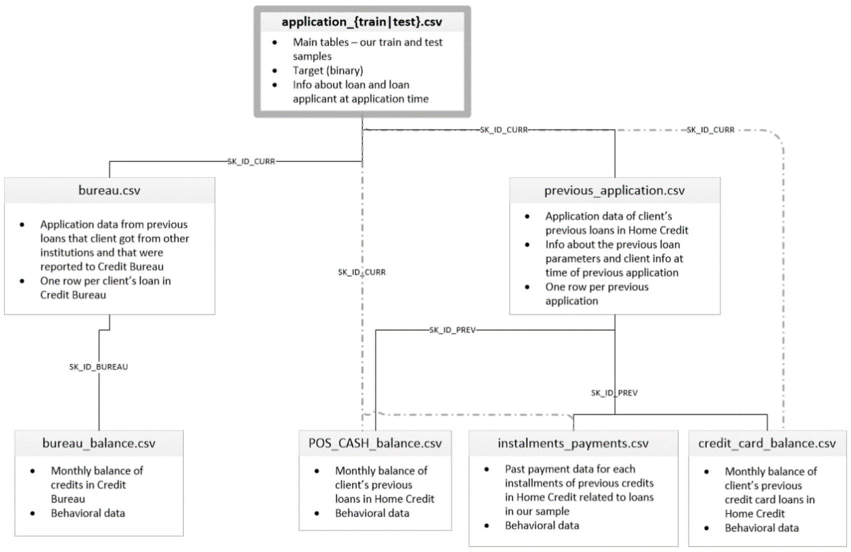
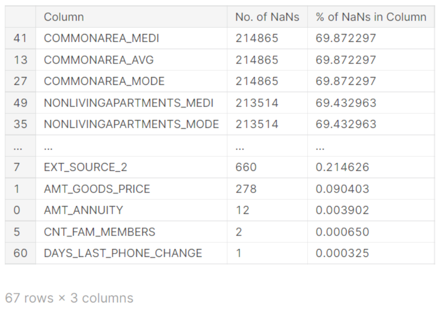

application_{train|test}.csv
Main tables – our train and test samples
Target (binary)
Info about loan and loan applicant at application time
SK_ID_CURR
SK_ID_CURR
SK_ID_CURR
SK_ID_CURR
bureau.csv
Application data from previous loans that client got from other institutions and that were reported to Credit Bureau
One row per client's loan in Credit Bureau
previous_application.csv
Application data of client's previous loans in Home Credit
Info about the previous loan parameters and client info at time of previous application
One row per previous application
SK_ID_BUREAU
SK_ID_PREV
SK_ID_PREV
bureau_balance.csv
Monthly balance of credits in Credit Bureau
Behavioral data
POS_CASH_balance.csv
Monthly balance of client's previous loans in Home Credit
Behavioral data
instalments_payments.csv
Past payment data for each installments of previous credits in Home Credit related to loans in our sample
Behavioral data
credit_card_balance.csv
Monthly balance of client's previous credit card loans in Home Credit
Behavioral data

Figure 1: data-set merge structure

Before processing, the training data-set contains **307511 rows** and **121 columns**. Testing data-set has **48744 rows** and **120 columns**.

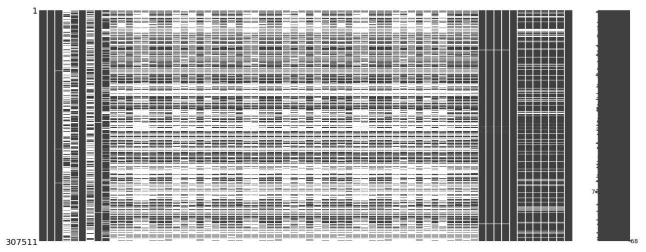
We count the the **NaNs** values and its' percent(a) and then visualize the columns with the **distribution** of NaNs values based on target variable(b).



	Column	No. of NaNs	% of NaNs in Column
41	COMMONAREA_MEDI	214865	69.872297
13	COMMONAREA_AVG	214865	69.872297
27	COMMONAREA_MODE	214865	69.872297
49	NONLIVINGAPARTMENTS_MEDI	213514	69.432963
35	NONLIVINGAPARTMENTS_MODE	213514	69.432963
...	...	...	...
7	EXT_SOURCE_2	660	0.214626
1	AMT_GOODS_PRICE	278	0.090403
0	AMT_ANNUITY	12	0.003902
5	CNT_FAM_MEMBERS	2	0.000650
60	DAYS_LAST_PHONE_CHANGE	1	0.000325

67 rows × 3 columns

(a) NaNs values



(b) Distribution of NaNs

Figure 2: Analysis the NaNs

Check **duplication** of record in training data-set: There is no duplicated record in this data-set.

```
train_df.duplicated().sum()
# result = 0
```

Check the target value in counting by using a bar chart: Then from the result we find most people has the ability to pay the bill, while few can not pay. We should bear in mind the **target variable is not balanced, but we need to balance them**.

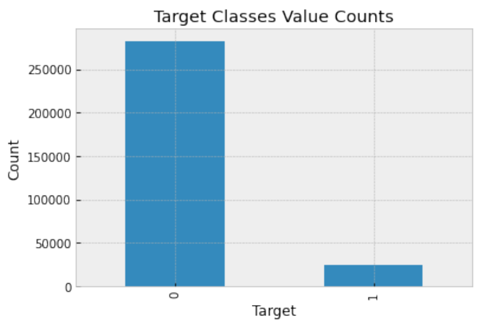
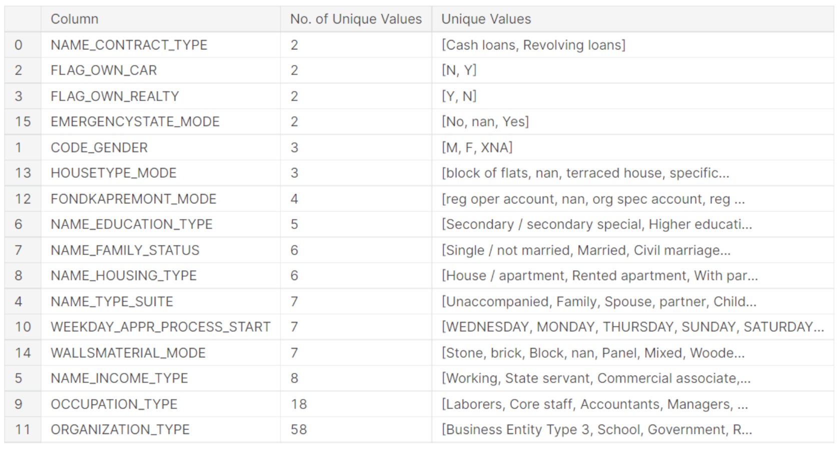


Figure 3: Target Class's Value Count

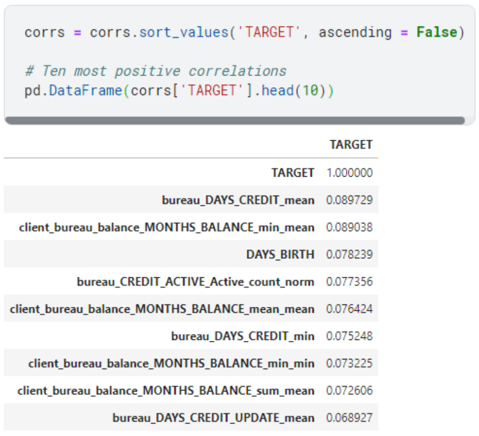
We then give a overall looking on unique classes in each column:



	Column	No. of Unique Values	Unique Values
0	NAME_CONTRACT_TYPE	2	[Cash loans, Revolving loans]
2	FLAG_OWN_CAR	2	[N, Y]
3	FLAG_OWN_REALTY	2	[Y, N]
15	EMERGENCYSTATE_MODE	2	[No, nan, Yes]
1	CODE_GENDER	3	[M, F, XNA]
13	HOUSETYPE_MODE	3	[block of flats, nan, terraced house, specific...
12	FONDKAPREMONT_MODE	4	[reg oper account, nan, org spec account, reg ...
6	NAME_EDUCATION_TYPE	5	[Secondary / secondary special, Higher educati...
7	NAME_FAMILY_STATUS	6	[Single / not married, Married, Civil marriage...
8	NAME_HOUSING_TYPE	6	[House / apartment, Rented apartment, With par...
4	NAME_TYPE_SUITE	7	[Unaccompanied, Family, Spouse, partner, Child...
10	WEEKDAY_APPR_PROCESS_START	7	[WEDNESDAY, MONDAY, THURSDAY, SUNDAY, SATURDAY...
14	WALLSMATERIAL_MODE	7	[Stone, brick, Block, nan, Panel, Mixed, Woode...
5	NAME_INCOME_TYPE	8	[Working, State servant, Commercial associate,...
9	OCCUPATION_TYPE	18	[Laborers, Core staff, Accountants, Managers, ...
11	ORGANIZATION_TYPE	58	[Business Entity Type 3, School, Government, R...

Figure 4: extract unique classes in each column

We have calculated WOE and IV in coursework 2, we find the most important variables that choose to have deeper analysis.

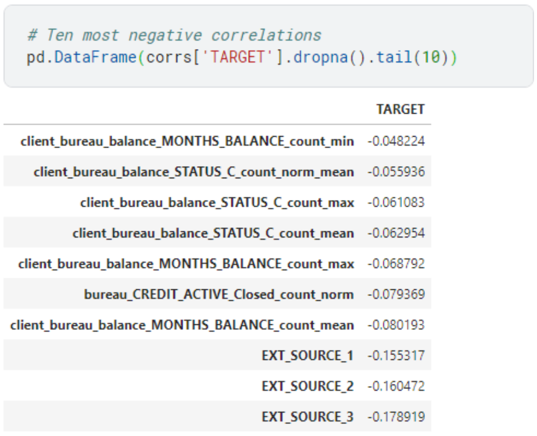


```
corrs = corrs.sort_values('TARGET', ascending = False)

# Ten most positive correlations
pd.DataFrame(corrs['TARGET'].head(10))
```

	TARGET
TARGET	1.000000
bureau_DAYS_CREDIT_mean	0.089729
client_bureau_balance_MONTHS_BALANCE_min_mean	0.089038
DAYS_BIRTH	0.078239
bureau_CREDIT_ACTIVE_Active_count_norm	0.077356
client_bureau_balance_MONTHS_BALANCE_mean_mean	0.076424
bureau_DAYS_CREDIT_min	0.075248
client_bureau_balance_MONTHS_BALANCE_min_min	0.073225
client_bureau_balance_MONTHS_BALANCE_sum_mean	0.072606
bureau_DAYS_CREDIT_UPDATE_mean	0.068927

(a) top 10 positive variables related to TARGET



```
# Ten most negative correlations
pd.DataFrame(corrs['TARGET'].dropna().tail(10))
```

	TARGET
client_bureau_balance_MONTHS_BALANCE_count_min	-0.048224
client_bureau_balance_STATUS_C_count_norm_mean	-0.055936
client_bureau_balance_STATUS_C_count_max	-0.061083
client_bureau_balance_STATUS_C_count_mean	-0.062954
client_bureau_balance_MONTHS_BALANCE_count_max	-0.068792
bureau_CREDIT_ACTIVE_Closed_count_norm	-0.079369
client_bureau_balance_MONTHS_BALANCE_count_mean	-0.080193
EXT_SOURCE_1	-0.155317
EXT_SOURCE_2	-0.160472
EXT_SOURCE_3	-0.178919

(b) top 10 negative variables related to TARGET

Figure 5: The most important variables that choose to analysis

In additional, we use the following code to delete the outline:

```
# Find continuous columns
all_numerical_columns = list(training_df.select_dtypes(exclude='object').columns)
cont_cols = [col for col in all_numerical_cols if col!=="TARGET" and col[:5]!='FLAG__']
```

After we find all continues variable(number = 78) in the above step, we draw box-plot(figure 6) for each of these variables.

As the figure 6 shows, almost all continuous features are outlier-sensitive, so if we use models sensitive to outliers, such as Lovesic regression models, or less influential and powerful models sensitive to outliers in tree models or deep neural networks, then these features should be normalised.Functions with dubious data can be well analysed to clean up erroneous data.

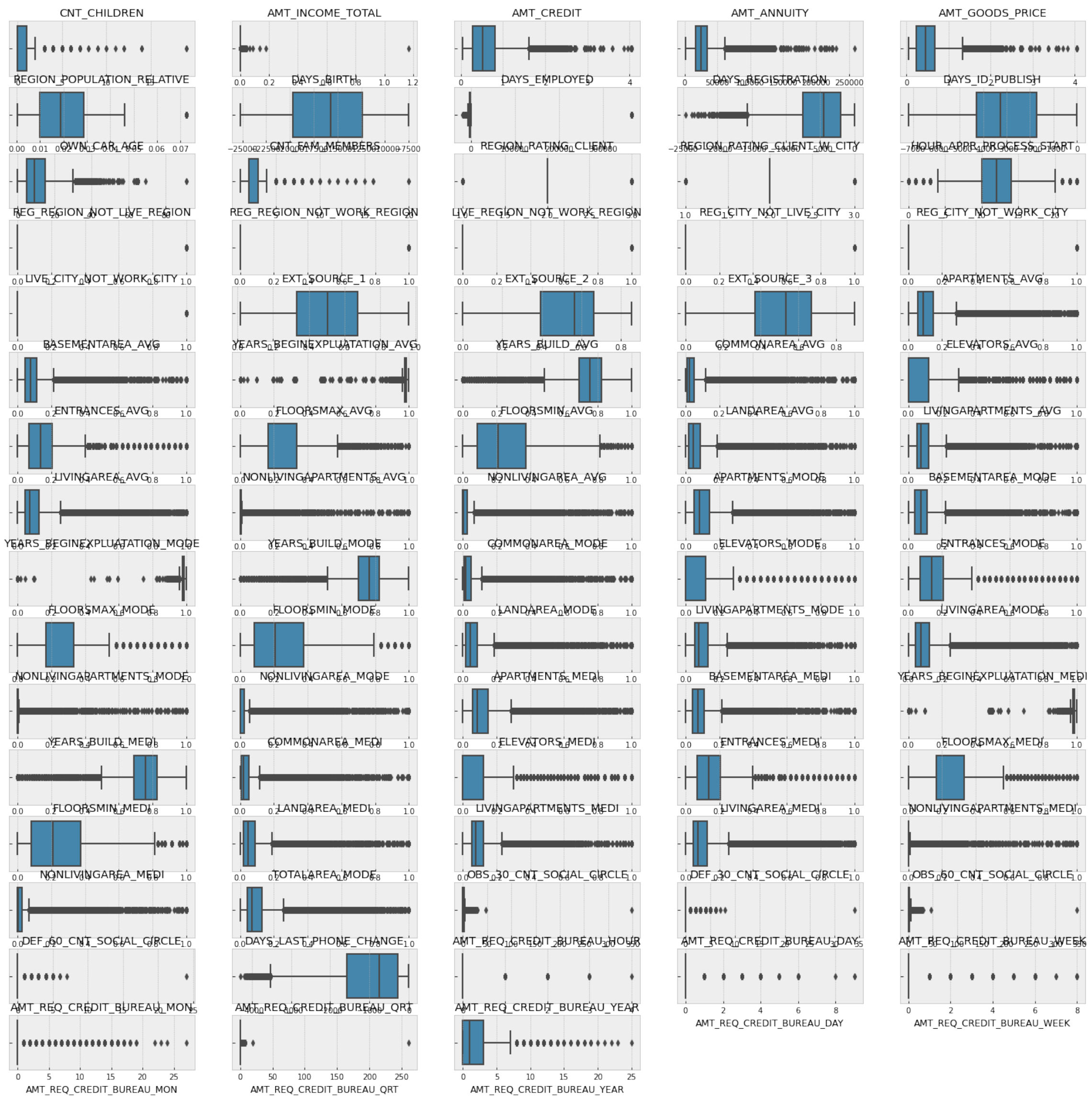


Figure 6: box-plot for all continue variable

Our records include older users, but they may not have had a previous habit of using a mobile phone, so it is necessary to check the distribution.(Figure 7)

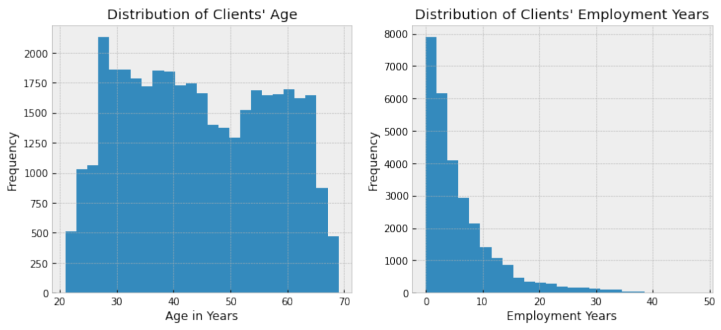


Figure 7: Age in yearemployment years

In Figure 7, you can see that our concern is correct; up to 12% of the data will have the kind of situation where the majority of people fall in the age range between 27 and 65, and many of them can earn money by working, and we can deal with the average.

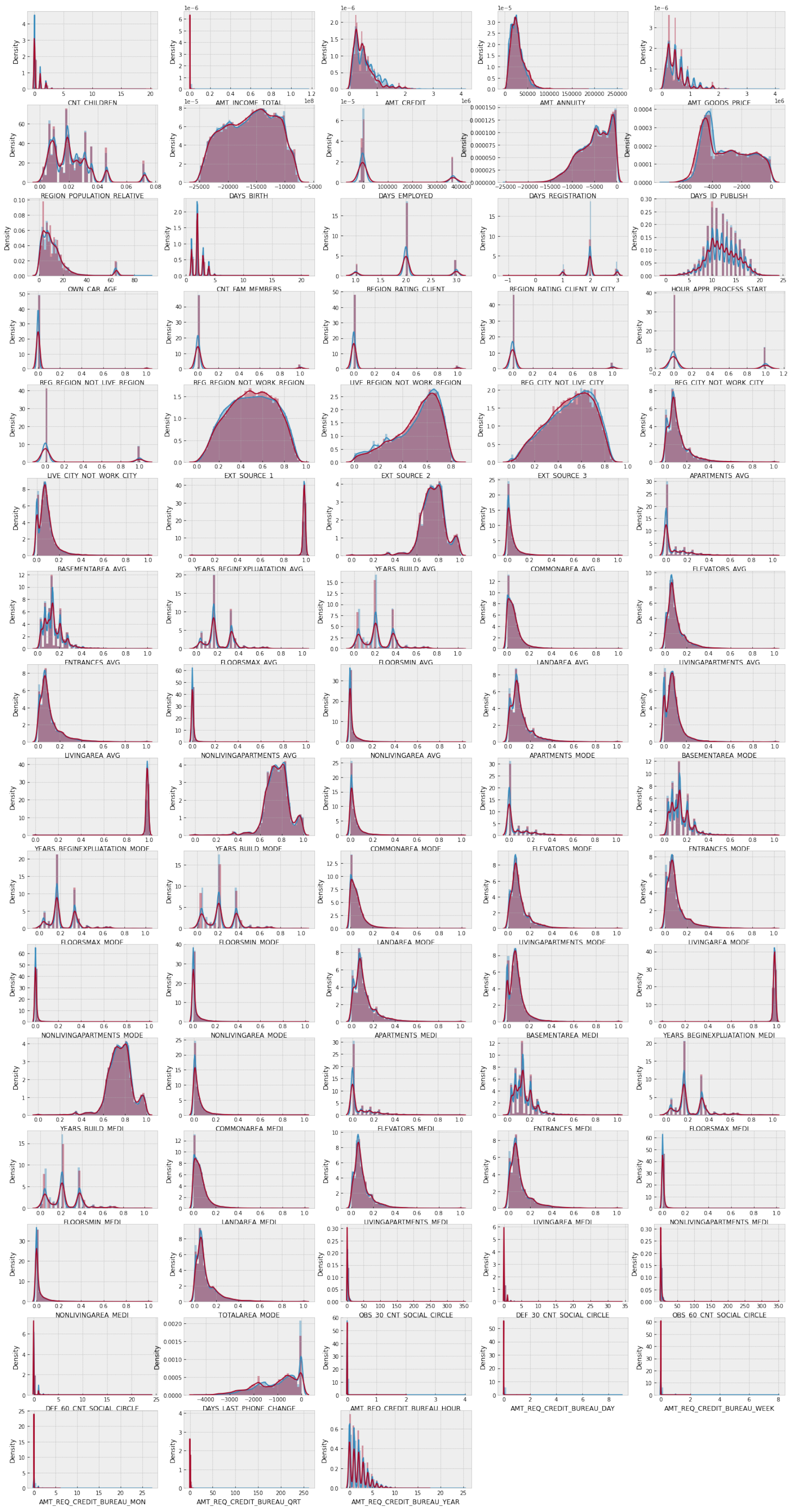
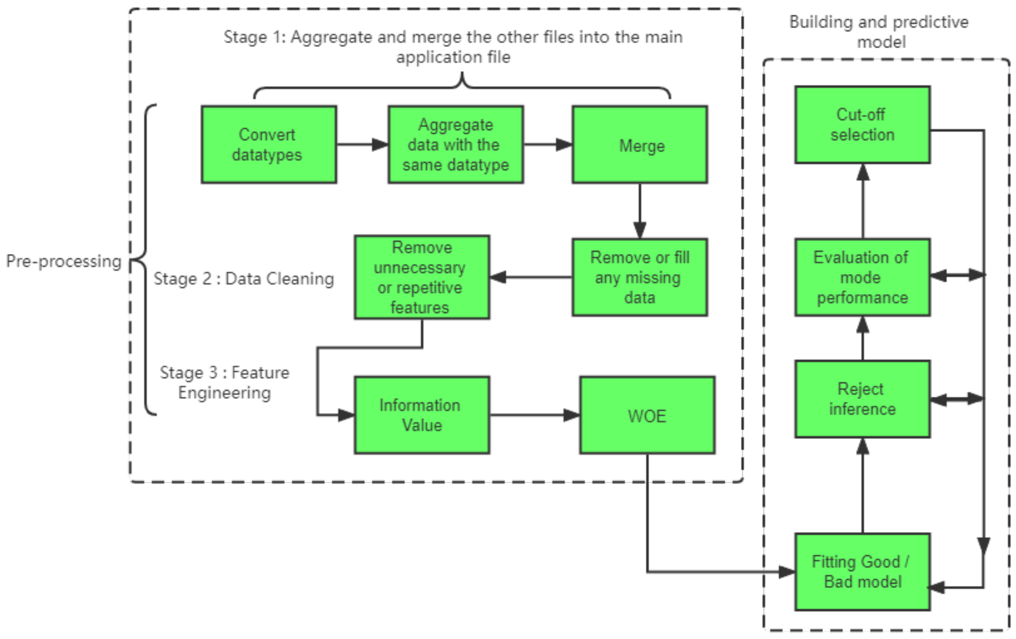


Figure 8: distribution for all continue variable

2.1 Overview of the model building process

In this section, the overall process of building baseline model is shown in Figure 2.



Stage 1: Aggregate and merge the other files into the main application file
Building and predictive model
Convert datatypes
Aggregate data with the same datatype
Merge
Cut-off selection
Pre-processing
Stage 2 : Data Cleaning
Remove unnecessary or repetitive features
Remove or fill any missing data
Evaluation of mode performance
Stage 3 : Feature Engineering
Information Value
WOE
Reject inference
Fitting Good / Bad model

Figure 9: Overview of the model building process

2.1.1 Fitting Good / Bad model

(1) Logistic Regression model with balanced classes

In Logistic regression, we use three data sets which is training, validation and testing by function called train_value_size.

```
def train_value_size(data, value_size, test_size):
    y = data["TARGET"]]
    data_train, data_valuetest, y_train, y_valuetest = train_test_split(data, y,
                                                       test_size = value_size+ test_size)
    .........
    return {'data_train:' data_train, 'data_value':data_value, 'data-set':data_test}
```

We first use the function developed in coursework 2 called mis_value_treatment.The aim of this function is to treat missing value. Then we use Regression model with balanced weighted classes classification which have the same effect of 'Coarse – classification of numeric characteristics'

```
model = LogisticRegression(class_weight = 'balanced')
```

We will first calculate class probabilities. Then, we predicted class labels using Built-in methods in model package(predict_proba(data_test)).

The previous created statistics data and the value counts in 4.1.2 need to be merged to the main application training file.

2.1.2 Reject inference

In the data-set our data use the target variable 0,1. We use simple augmentation, i.e. we assign rejections below and above the cut-off value to the ”bad” or ”good” category respectively.

2.1.3 Evaluation of Model Performance

For model that we designed in section 1.2.1:

(1) We calculate the Confusion Matrix:

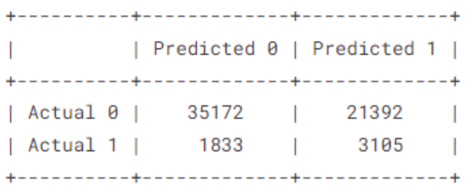
```
ConfusionMatrix = metrics.confusion_matrix(y_test,y_predicted)
```

Then Visualise our matrix:

```
def confusion_matrix(cm):
    tab = PrettyTable([' ', 'Predicted 0', 'Predicted 1'])
    tab.add_row(["Actual 0", cm[0][0], cm[0][1]])
    tab.add_row(["Actual 1", cm[1][0], cm[1][1]])
    print(tab)
```

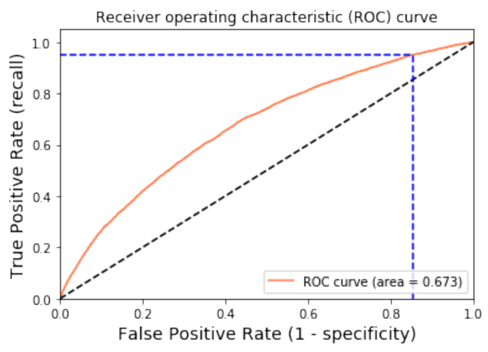
(2) we calculate the ROC_CURVE and plot:

```
def roc_curve(model,data_test,y_test):
.......
    [fpr, tpr, thr] = metrics.roc_curve(y_test, y_pred_proba)
    plt.plot(fpr, tpr, color='coral', label='ROC curve (area = %0.3f)' % metrics.auc(fpr
                                            , tpr))
```



```
+----------+-------------+-------------+
|          | Predicted 0 | Predicted 1 |
+----------+-------------+-------------+
| Actual 0 |    35172    |    21392    |
| Actual 1 |     1833    |     3105    |
+----------+-------------+-------------+
```

]



(b) ROC_CURVE for model1

Figure 10: Confusion Matrix ROC_CURVE for model 1

As the figure shown, the ROC CURVE score for model one = 0.673(baseline model) is better than random model(value = 0.5). In the section 3, we will keep comparing with different models.

2.1.4 Cut-off selection

We set the filiter at 0.95 for the value of tpr.

```
...
idx = np.min(np.where(tpr > 0.95))  # index of the first threshold for which the
                                        sensibility > 0.95
...
```

3 Part 2: Detailed Investigation of Modeling

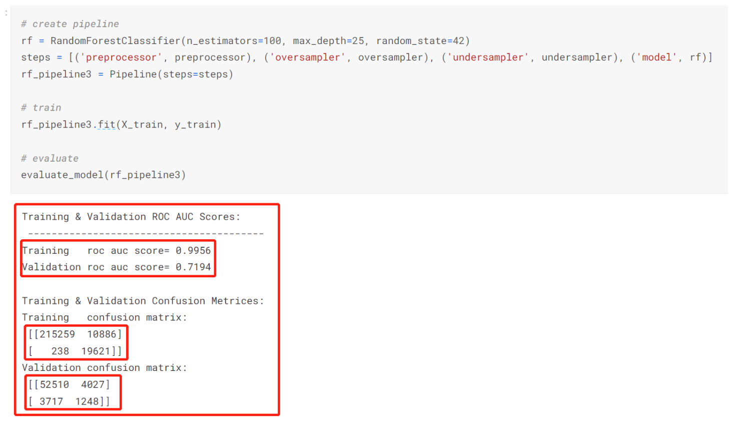
3.1 Compare model 1 - (Baseline) with model 2 - Random Forest / model 3 - Light GBM

(1) For model two(Random Forest):

To improve our baseline model, we use a new algorithm, a standard random forest algorithm for machine learning, where we first train on the original pre-processed dataset without the addition of unknown variables. Based on Bharathidason's random forest seed selection, we chose to use ten trees for our predictive model building [2].

A similar methodology was used by Butaru et al.(2016)[3], whereby it investigated consumer delinquency using C4.5 decision trees, logistic regression and random forests and used data from six different banks.

We calculate the **Confusion Matrix** and **ROC** value,then we Visualise our matrix the curve.



```
# create pipeline
rf = RandomForestClassifier(n_estimators=100, max_depth=25, random_state=42)
steps = [('preprocessor', preprocessor), ('oversampler', oversampler), ('undersampler', undersampler), ('model', rf)]
rf_pipeline3 = Pipeline(steps=steps)

# train
rf_pipeline3.fit(X_train, y_train)

# evaluate
evaluate_model(rf_pipeline3)
```

```
Training & Validation ROC AUC Scores:
 ---------------------------------------
Training   roc auc score= 0.9956
Validation roc auc score= 0.7194

Training & Validation Confusion Metrices:
Training   confusion matrix:
 [[215259  10886]
 [    238  19621]]
Validation confusion matrix:
 [[52510   4027]
 [ 3717   1248]]
```

Figure 11: Confusion Matrix and ROC for model 2

We conjecture that the expressiveness of the random forest model is **negatively correlated with the size of the number of features**. It may be that the feature vectors are sparse at high latitudes, so this leads to an imbalance problem in the data. This leads to an unbalanced model tree of the random forest, so the prediction results drop.

As the figure shown, the **ROC CURVE** score for model two (random forest) . In the section 3, we will keep comparing with different models.

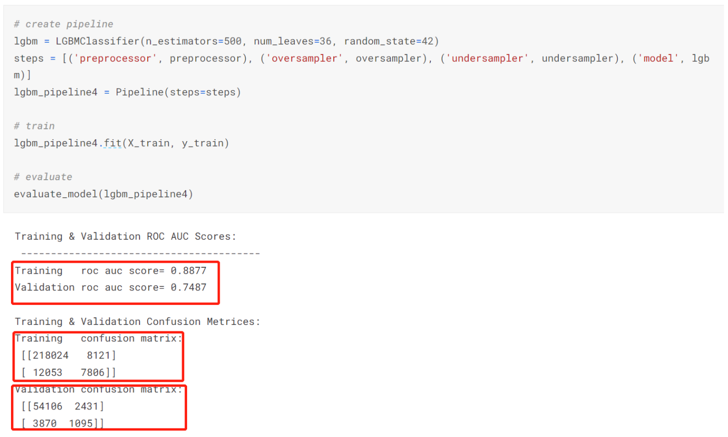
(2) For model three (Light GBM):

Our data needs to address the following issues, **Issue 1:** Our model 3 needs to be easy to interpret, and it needs to be ranked in importance based on the calculated **IV** values. **Problem 2:** is the presence of noise that needs to be considered. So the **lightGBM model** is our choice. This is because the model has some generalisation capability[4]. The formula below is used to fit the regression values, where the importance of **softmax** indicates the probability of the corresponding classification.

$$P_m^t(x_i) = \frac{e^{f_m^{t-1}(x_i)}}{\sum_{p=1}^{K} f_p^{t-1}(x_i)}$$

Figure 12: Formula of LightGBM model

[4] We calculate the Confusion Matrix and ROC value,then we Visualise our matrix the curve.



```
# create pipeline
lgbm = LGBMClassifier(n_estimators=500, num_leaves=36, random_state=42)
steps = [('preprocessor', preprocessor), ('oversampler', oversampler), ('undersampler', undersampler), ('model', lgb
m)]
lgbm_pipeline4 = Pipeline(steps=steps)

# train
lgbm_pipeline4.fit(X_train, y_train)

# evaluate
evaluate_model(lgbm_pipeline4)
```

```
Training & Validation ROC AUC Scores:
 ----------------------------------------
Training   roc auc score= 0.8877
Validation roc auc score= 0.7487

Training & Validation Confusion Metrices:
Training   confusion matrix:
 [[218024   8121]
 [ 12053   7806]]
Validation confusion matrix:
 [[54106  2431]
 [ 3870  1095]]
```

Figure 13: Confusion Matrix and ROC for model 3

(3) Compare the upper three model, we can finding:

1) performance ranking (based on ROC_AUC score) :

model 1 - baseline(regression) = 0.673 <model 2 - improved with more data (Random Forest) = 0.7194 <model 3 - Light GBM model = 0.7487

Though the result may be changed because of different hardware condition in machine, the model 3 has the best performance in the three models.

The model one - **logistic regression model** has performed best when trained on a data-set generated by a polynomial approach. **LightGBM** outperforms both the **random forest** and **logistic regression models**, whether the model is trained on the original dataset or a **pre-processed** or **balanced unbalanced** dataset.

4 Conclusion

The report presents data processing and innovations regarding credit scoring modelling, while based on the processing of our three models, we find that model three performs the best, while the **auc scores** are high close to **80%**. Also when we replaced many of the model **features selection and construction methods**, the performance was consistent as described in section 3.1. For the original dataset we also used the **random forest algorithm** and also found that the performance of the **pre-processed data** species decreased, but the linear model scores increased.

There are still imperfections in our project such as our **lack of a suitable time window**, which makes it difficult to use some useful tools to help us with our analysis, but this is also an inherent weakness of the dataset, and we only considered three models in our modelling, but there are many **Gaussian or stacked models** that could be used, and perhaps **PCA** could also be considered. For our dataset, the use of more tables for experiments may have improved the performance and accuracy of the predictions; we did not analyse the robustness of the models, i.e. their stability in different environments. This is a direction for our future improvement.

References

[1] Amir E Khandani, Adlar J Kim, and Andrew W Lo. Consumer credit-risk models via machine-learning algorithms. *Journal of Banking & Finance*, 34(11):2767–2787, 2010.

[2] S Bharathidason and C Jothi Venkataeswaran. Improving classification accuracy based on random forest model with uncorrelated high performing trees. *Int. J. Comput. Appl*, 101(13):26–30, 2014.

[3] Florentin Butaru, Qingqing Chen, Brian Clark, Sanmay Das, Andrew W Lo, and Akhtar Siddique. Risk and risk management in the credit card industry. *Journal of Banking & Finance*, 72:218–239, 2016.

[4] Ziyue Qiu, Yuming Li, Pin Ni, and Gangmin Li. Credit risk scoring analysis based on machine learning models. In *2019 6th International Conference on Information Science and Control Engineering (ICISCE)*, pages 220–224. IEEE, 2019.