

**QBUS6850 Machine Learning for Business**

**(2018S2)**

**Group Project (Assignment 3)**

Due date: Monday 29 October 2018

Group Number: \_\_\_\_\_\_\_\_\_\_26\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Group Members:

\_\_\_\_\_\_\_\_470042289\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_470346925\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_470513192\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_470540154\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_460461003\_\_\_\_\_\_\_\_\_\_\_

# Executive Summary

This report aims at facilitating on Home Credit’s serving of those lack of credit history, underserved clients by the traditional banks in the mortgage industry. Instead of using their historical credit information, predictions of the repayment abilities are made based on their rankings formulated using alternative data (e.g. telco and transactional information ) combined with the information of the clients’ previous credits provided by other financial institutions that were reported to Credit Bureau.

After conducting data-cleaning, 2 set of features are extracted based on either only utilising the alternative data or using both the alternative data and the Credit Bureau dataset. The effectiveness of the 2 sets of selected features are then tested by employing benchmark model – logistic regression, also KNN, Extreme Gradient Boosting and Random Forest. Finally, after model selections based on their performance scores: accuracy score, AUC and F1-score, the best performing model, which is Extreme Gradient Boosting, is the used for the final predictions.

Table of Contents

[Executive Summary 2](#_Toc528595588)

[1. Introduction 2](#_Toc528595589)

[2. Dataset and Features 3](#_Toc528595590)

[2.1. Data Pre-processing 3](#_Toc528595591)

[2.1.1. Motivation 3](#_Toc528595592)

[2.1.2. Pre-processing 3](#_Toc528595593)

[2.2. Exploratory Data Analysis 3](#_Toc528595594)

[2.2.1. Basic Information 3](#_Toc528595595)

[2.2.2. Missing Values 3](#_Toc528595596)

[2.2.3. Correlation 4](#_Toc528595597)

[2.2.4. Data distribution 5](#_Toc528595598)

[ Numerical Features 5](#_Toc528595599)

[ Categorical Features 5](#_Toc528595600)

[2.2.5. Factor Analysis and Clustering 6](#_Toc528595601)

[2.3. Data Processing 7](#_Toc528595602)

[2.3.1. Motivations 7](#_Toc528595603)

[2.3.2. Implementation of Data Processing techniques 7](#_Toc528595604)

[2.3.3. Feature Selection 7](#_Toc528595605)

[2.3.4. Missing Value Imputation 8](#_Toc528595606)

[2.3.5. Factor Analysis of Mixed Data 9](#_Toc528595607)

[3. Methodologies, Results and Interpretations 11](#_Toc528595608)

[3.2. Logistic Regression Classifier 11](#_Toc528595609)

[3.1.1. Model Descriptions 11](#_Toc528595610)

[3.1.2. Data Preparations 11](#_Toc528595611)

[3.1.3. Model Fitting 11](#_Toc528595612)

[3.1.4. Summary of findings and justifications 13](#_Toc528595613)

[3.3. K-Nearest Neighbor Classifier 14](#_Toc528595614)

[3.3.1. Model Descriptions 14](#_Toc528595615)

[3.3.2. Data Preparations 14](#_Toc528595616)

[3.3.3. Model Fitting 14](#_Toc528595617)

[3.2.4. Summary of findings and justifications 15](#_Toc528595618)

[3.3. Extreme Gradient Boosting 17](#_Toc528595619)

[3.3.1. Model Descriptions 17](#_Toc528595620)

[3.3.2. Data Preparations 17](#_Toc528595621)

[3.3.3. Model Fitting – Using Full Set of Original Features 18](#_Toc528595622)

[3.3.4. Model Fitting – Using Selected Features Without Addition Features 18](#_Toc528595623)

[3.3.5. Model Fitting – Using Selected Features With Addition Features 18](#_Toc528595624)

[3.3.6. Summary of findings and justifications 18](#_Toc528595625)

[3.4. Random Forest 19](#_Toc528595626)

[3.4.1. Model Descriptions 19](#_Toc528595627)

[3.4.2. Data Preparations 19](#_Toc528595628)

[3.4.4. Summary of findings and justifications 20](#_Toc528595629)

[4. Conclusions and Predictions 21](#_Toc528595630)

[4.1. Conclusions on Model Selections 21](#_Toc528595631)

[4.2. Prediction Outcomes 21](#_Toc528595632)

[References: 22](#_Toc528595633)

[Appendices 22](#_Toc528595634)

[Appendix – Python Code 22](#_Toc528595635)

[Appendix – Meeting Agenda 23](#_Toc528595636)

[Appendix – Meeting Template 26](#_Toc528595637)

# Introduction

Nowadays, a phenomenon of inefficiencies in mortgage industry is that lots of people are underserved by the traditional banks simply due to their lack of credit history. However, it is a quite common situation as house purchasing is a big and rare event for most of the individuals, which is not a daily routine activity such as buying coffee that we can easily get a lot of historical data. Thus, it is believed that this lack of available information does not affect the repayment abilities of the mortgagors.

Home Credit is a responsible lending company that wishes to make this world a better place by serving these clients through predicting their repayment abilities using alternative data (e.g. telco and transactional information) instead of their credit histories.

To facilitate on achieving this goal, in this report, models are built to rank Home Credit’s clients repayment abilities using these alternative information of their existing customers. The dataset contains a wide varieties of features that might have an impact on the clients’ repayment abilities, corresponding with the target results of whether the clients ending up with being default or not-default. Also, the information contains clients’ previous credits provided by other financial institutions that were reported to Credit Bureau.

After data preparations and EDA in Section 2, the data is cleaned and three sets of features were prepared to fit the models in Section 3. In Section 3, the model is then fitted using Logistic regression as benchmark model. In addition, KNN, Extreme Gradient Boosting and Random Forest are also employed for better comparability when conducting feature selection and model selections. Finally, it is concluded that Extreme Gradient Boosting is the best model overall, and thus is used for the final predictions.

# Dataset and Features

## 2.1. Data Pre-processing

### 2.1.1. Motivation

The raw data provided often contains many noise and most importantly, it has text information that cannot be used by machines or algorithms. The implementation of data pre-processing can transform the raw data and also clean unnecessary noise that can impede the performance of the models.

### 2.1.2. Pre-processing

The training data contains 274511 samples and 120 features and the extra data set which contains information from the Credit Bureau has 168455 samples and 16 extra features. Since the size of the training data set is large, we partition it into 2 parts, which are the training set with the size of approximately 60,000 samples and the validation set, with the size of 33,000 samples, aligning to the size of the test data. We will perform all the analysis as well as model fitting on the training set and evaluation the performance of the models on the validation set.

Both two raw data sets store a bunch of text information inside and this is not input-friendly for machine learning modelling techniques, thus, the Label Encoding method is employed here to conduct the numerical transformation of text columns. There two things worth noting related to the Label Encoding method. First, Label Encoding and One Hot Encoding are both general methods used to process categorical text data, however, it’s Label Encoding rather than One Hot Encoding chosen here is because that Label Encoding is handier in the following feature selection, missing value imputation and factor analysis parts. Second, in factor analysis, the package (PRINCE) we use will conduct One Hot Encoding for categorical features before the actual factorization process inside.

## 2.2. Exploratory Data Analysis

Exploratory data analysis (EDA) is a way of summarizing and visualizing the characteristics of the data, from which we gain some useful insights. In this section, we first summarize some basic information of the data, such as the data type, and check the missing values, explore the correlation within the data and also perform clustering to visualize the class distribution of the data.

### Basic Information

To utilize the original data and the extra features provided by the Credit Bureau, we merge the two data sets by the ID of Home Credit’s clients and get a new training set of the size 274511 samples and 136 features (including the target). To better understand the extra features, the following analysis will perform on the merged training data. In total, there are 81 numerical features, 54 categorical features in the new training data. Additionally, class 1 (default) only accounts for 8% of the total training samples, which means that this data set has the imbalanced problem.

### Missing Values

In total, there are 82 features containing missing values in the data, among which,45 features have missing values more than 50% and 65 features have the missing percentage more than 30%. The following plots visualize the row locations of missing values in the features containing missing values, with the blank space indicating missing values, which means that features having more blank space contain more missing values. From these two plots, we can see that some of the columns have missing values at similar rows, such as OBS\_30\_CNT\_SOCIAL\_CIRCLE, DEF\_30\_CNT\_SOCIAL\_CIRCLE, OBS\_60\_CNT\_SOCIAL\_CIRCLE and DEF\_60\_CNT\_SOCIAL\_CIRCLE. This is because some of the features are similar and thus clients who do not have record under one feature also do not have record at another similar feature. This pattern can be helpful in the process of data cleaning or imputation.

|  |  |
| --- | --- |
| Figure 1 Missing values at the first 38 features | Figure 2 Missing values at the last 44 features |

The following heatmaps further explore the structure of missing values, which describe the nullity correlation R within between features containing missing values. R=1 means the features have missing values at the exactly same rows while R=-1 means that features have missing values at the exactly different rows.

|  |  |
| --- | --- |
| Figure 3 Null correlation for the first 38 features | Figure 4 Null correlation for the last 42 features |

### Correlation

We also investigate the correlation between all the column in the new training data, including the target. As per the following correlation heatmap, most of the features have very few correlations with each other since the orange color indicates a correlation of nearly zero. Also, we observe that one variable named ‘FLAG\_MOBIL’ has zero correlation with other variables. We further checked this variable and found that it contains only one category and therefore can be counted as useless feature in this case. Hence, we drop the ‘FLAG\_MOBIL’ column from both the without- and with- extra-feature data sets.

### Data distribution

In order to further investigate the relationship between features and the target as well as the distribution of the data, we select top 20 features that have the highest correlation with the target and divide them into two groups by data types and visualize them. For numerical features we plot the conditional distributions and for categorical features we plot the bar charts.

### Numerical Features

From the conditional distribution plot, we can see that only the first 4 variables have different distribution for the two classes while that of other features are quite similar.

### Categorical Features

We use bar chart to visualize the distribution of categorical features for the two classes. For each bar chart, the y axis is the count of samples and we can see that the proportions of different categories in default class and not-default class are similar. Besides, the problem of imbalanced data set that mentioned at the beginning can be observed clearly with class 0 much larger than class 1.

|  |  |
| --- | --- |
| Figure 5 Correlation within the data | Figure 6 Conditional distribution for top 11 numerical features |

|  |  |  |
| --- | --- | --- |
|  |  |  |
|  |  |  |
|  |  |  |

*Figure 7 Bar charts of the top 9 categorical features*

### Factor Analysis and Clustering

Since we perform predictions for binary classification, we also want to see how the data are clustered. We first used factors analysis to reduce the dimensionality of the training data and perform clustering by plotting the data on the first two transformed variables. For factor analysis, we choose Factor Analysis of Mixed Data (FAMD) which performs similar function of Principle Component Analysis (PCA) except that FAMD can handle both numerical and categorical features while PCA can only deal with numerical features. As we can see from the graph, the data are not well partitioned into two clusters and are almost overlapped.

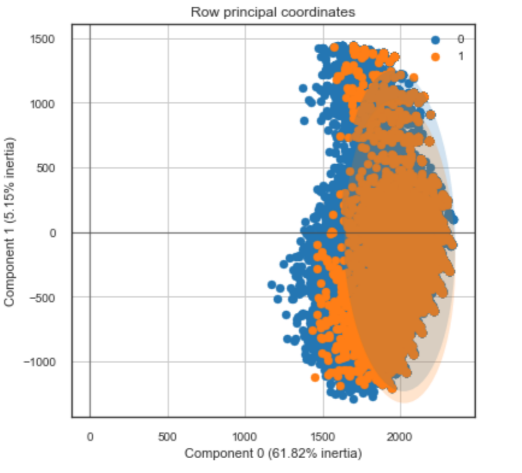


Figure 8 Clustering on the first two FAMD factors

## Data Processing

### Motivations

Data processing is the collection and manipulation of data. The reduced risk of data-overfitting and ineffective feature inputs are the main consideration that we decided to employ this method when solving this classification problem. Therefore, robust data processing procedures are of great value.

### Implementation of Data Processing techniques

Our workflow in this part can be summarised as the following three parts: data pre-processing, feature selection, missing value imputation and factor analysis. We have prepared two feature sets in this project. The first one only uses features collected by Home Credit website while the second one joins the Home Credit’s features and features offered by Credit Bureau together. And we will call the first one as without-extra-feature data set and the second with-extra-feature data set in the entire data processing part. And we decide to uniform the processing methods for both without- and with- extra feature data sets to make a fair comparison between these two data sets’ predicting power.

### Feature Selection

The logic in this data processing case is to conduct feature selection before missing value imputation and there are two supporting points for this choice. First off, any kind of imputation method will to some degree add in more or less noise to feature sets and this will unavoidable lead to biased feature selection decision. Plus, after limiting the number of features by making selection, we will efficiently lower down the missing value imputation workload and therefore more possible to generate a better performing feature set.

The generic feature selection methods include filtering, wrapper, regularization and randomized optimization. We decide to choose filtering method here since filtering will bunce the data processing part by doing selection without initial missing value imputation, effectively consider features’ relation with target and fairly low computational cost though it cannot efficiently handle multi-collinearity issue. People generally will use fold change, t test and Pearson Correlation when conducting filtering. We have tried fold change and Pearson correlation methods during feature selecting analysis.

In theory, fold change involves first calculate the mean within each class and then the log2 of the ratio of the mean between classes. The issue for fold change is that it cannot take into account of variability within the class. Thus, fold change tends to select variables without missing values, categorial and Boolean variables which will have more significant difference in mean. And we can see from **Table 1** that most of the features selected by fold change has no missing values and Boolean features dominant in both cases, except that fold change selects two continuous features from Credit Bureau in with-extra-feature data set. For entire Fold Change selected features please refer to **Appendix 1** and **Appendix 2**.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **No Extra Column Name** | **Level** | **Missing %** | **With Extra Column Name** | **Level** | **Missing %** |
| FLAG\_DOCUMENT\_4 | 2 | 0 | FLAG\_DOCUMENT\_4 | 2 | 0.000 |
| FLAG\_DOCUMENT\_10 | 2 | 0 | FLAG\_DOCUMENT\_10 | 2 | 0.000 |
| FLAG\_DOCUMENT\_12 | 2 | 0 | FLAG\_DOCUMENT\_12 | 2 | 0.000 |
| FLAG\_DOCUMENT\_2 | 2 | 0 | AMT\_CREDIT\_SUM\_OVERDUE | 211 | 0.453 |
| FLAG\_DOCUMENT\_13 | 2 | 0 | FLAG\_DOCUMENT\_2 | 2 | 0.000 |
| FLAG\_DOCUMENT\_17 | 2 | 0 | FLAG\_DOCUMENT\_13 | 2 | 0.000 |
| FLAG\_DOCUMENT\_15 | 2 | 0 | FLAG\_DOCUMENT\_17 | 2 | 0.000 |
| FLAG\_DOCUMENT\_14 | 2 | 0 | DAYS\_CREDIT\_ENDDATE | 7962 | 0.467 |
| FLAG\_DOCUMENT\_16 | 2 | 0 | FLAG\_DOCUMENT\_15 | 2 | 0.000 |
| FLAG\_DOCUMENT\_21 | 2 | 0 | FLAG\_DOCUMENT\_14 | 2 | 0.000 |

**Table 1: Top Ten Features Selected by Fold Change**

Pearson Correlation takes into account of the variability within each class and focus on the relation between a single feature and target. It will tend to select features that can well interpret the target, whereas it only considers a single correlation in each selection iteration and thus will tend to select redundant collinear features. Since Pearson Correlation remove the class variability influence, it will more evenly select features without biasing on fully filled, categorical and Boolean features. In **Table 2**, it’s easy to notice that Pearson Correlation tends to both select a variety of different feature types and include features with missing values as well from the without- and with-extra feature data sets. For entire Pearson Correlation selected features please refer to **Appendix 1** and **Appendix 2**. Our Specific feature selection strategy in Pearson Correlation Filtering is to set threshold at 1% and select features having larger than 1% of correlation with target and thus generate 79 features in total. In general, we can see from Appendix 1 and Appendix 2, the features selected in each case is similar and this filtering method doesn’t include many features from Credit Bureau in with-extra-feature data set.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **No Extra Column Name** | **Level** | **Missing%** | **With Extra Column Name** | **Level** | **Missing %** |
| EXT\_SOURCE\_3 | 809 | 19.869 | EXT\_SOURCE\_3 | 809 | 0.199 |
| EXT\_SOURCE\_2 | 114271 | 0.216 | EXT\_SOURCE\_2 | 114271 | 0.002 |
| EXT\_SOURCE\_1 | 103908 | 56.376 | EXT\_SOURCE\_1 | 103908 | 0.564 |
| DAYS\_BIRTH | 17444 | 0.000 | DAYS\_BIRTH | 17444 | 0.000 |
| REGION\_RATING\_CLIENT\_W\_CITY | 3 | 0.000 | REGION\_RATING\_CLIENT\_W\_CITY | 3 | 0.000 |
| REGION\_RATING\_CLIENT | 3 | 0.000 | REGION\_RATING\_CLIENT | 3 | 0.000 |
| DAYS\_LAST\_PHONE\_CHANGE | 3756 | 0.000 | DAYS\_CREDIT | 2922 | 0.453 |
| CODE\_GENDER | 3 | 0.000 | DAYS\_LAST\_PHONE\_CHANGE | 3756 | 0.000 |
| NAME\_EDUCATION\_TYPE | 5 | 0.000 | CODE\_GENDER | 3 | 0.000 |
| DAYS\_ID\_PUBLISH | 6158 | 0.000 | NAME\_EDUCATION\_TYPE | 5 | 0.000 |

**Table 3: Top Ten Features Selected by Pearson Correlation**

### Missing Value Imputation

During missing value imputing, we made two types assumption on missing data, missing at random (MAR) and missing not at random (MNAR). For the MAR, we use multiple Imputation by chained equation (MICE) to fill in the missing part and for MNAR, we use exotic value 100 to impute the missing records.

#### Missing at Random (MAR)

MAR is defined as the propensity for a data point to be missing is not related to the missing data, but it is related to some of the observed data, which in other words means that the actual value of missing data will not influence the current data distribution.

The assumption for MICE impute is data MAR and MICE is an unsupervised imputation method. Theoretically, MICE fills in the missing values by iteratively running regression with imputed columns as target and the rest as predictors. To be specific, linear regression is chosen for continuous data, logistic regression is selected for Boolean data and Poisson regression is used for categorical data.

The features under MAR assumption for both without- and with- extra feature data sets have been listed in **Appendix 5** and **Appendix 6** for further interest. Features assumed by MAR are found to have less proportion number of missing in the entire column most of the time and the missing reasons can basically be defined as recording error or something similar. A fairly special column named ‘EXT\_SOURCE\_1’ is under MAR assumption but have 56.376% NaN values and we don’t think MICE can precisely impute this column by running regression. Therefore, this column is deleted and excluded from both of the feature data sets.

#### Missing Not at Random (MNAR)

The two possible reasons of MNAR is that the missing value depends on the hypothetical value or missing value is dependent on some other variable’s value. For our scenario, the hypothetical value is assumed for features chosen under MNAR assumption.

For MNAR missing values, 100 is used to fill in the missing entries and the reason for doing so is because that all the MNAR missing columns are categorical or Boolean types and we want to find a new class to separate the missing values apart from those existing categories. Choosing exotic value is to make sure the separation is effective with the expectation that longer distance will perform well during modelling. We assume the same set of MNAR features for both without- and with- extra-feature data sets and the list of names has been put in **Appendix 7**. The specific reasons for MNAR in our case is mostly because applicant has no property yet and those MNAR features are statistics related to it. Take ‘FLOORSMAX\_AVG’ as an example, since the specific applicant don’t own any physical home, there is no applicable choice for them to fill in the survey and they will choose to skip this column and leave it blank.

### Factor Analysis of Mixed Data

#### Motivation

After feature selection, we now have 79 features in total. Since we use Pearson Correlation for feature selection, the selected features might be highly correlated as this filtering method do not consider the relationship within the features, which could lead to the problem of multicollinearity. To handle this potential issue, we choose Factor Analysis of Mixed Data (FAMD) to reduce redundant features as well as decreasing the number of features. As is mentioned in the EDA part, since this data set has both numerical and categorical data types, PCA is not suitable and instead, we choose FAMD which can deal with both quantitative and qualitative features. Like PCA, after FAMD, the output components are orthogonal to each other and thus, it can reduce redundant features and select important components that explain most variance of the data. FAMD is included in the *prince* package and according to the description on GitHub, it does one-hot encoding for categorical features ("MaxHalford prince", 2018) (ref)so that we do not perform one-hot encoding or create dummy variables for the categorical variables.

#### Results

We perform FAMD on both the with- and without- extra features data sets. The following plots show the explained variance of the first 40 components for the two data sets. We can see that for both data sets, the first component explained a large proportion of the variance compared with other components and the first 40 components can explain 97% of the total variance. Since the first 40 components already explain nearly 100% of the total variance, we use 40 components as the new features for the both data sets. the first 40 components explain

|  |  |
| --- | --- |
| Figure 9 | Figure 10 |

# Methodologies, Results and Interpretations

## Logistic Regression Classifier

### Model Descriptions

A logistic regression model is built as the benchmark model to predict the probability of the clients ending up in a default situation. Logistic regression applies a sigmoid function to multi-linear regression to squeeze predicted values to range -1 to 1, and the predicted values are probability of target belonging to positive class. The default threshold of logistic regression is 0.5, which means cases with predicted probability larger than 0.5 will be classified as positive and vice versa.

However, in clients default assessment, if the client who will default is classified as none default, it will cause huge losses to Home Credit. Thus, the threshold should be set lower than 0.5 in order to predict default as accurate as possible.

### Data Preparations

After removing the rows that include missing values of more than 45% as discussed in part 2.1, the data imbalance issue that we discovered from EDA (part 2.2) was dealt with using down-sampling. Then, 2 sets of features were selected among both the original data and the expanded dataset that includes both the original ones and the extra data (part 2.3 & 2.4).

Thus, up to this point, 3 sets of features were available for the following model fitting: (1) full set of original features, (2) selected set of features from the original dataset, and (3) selected set of features from the expanded dataset. All these three sets of features are used in fitting the logistic regression model for the sake of comparing their performances.

The training data are further split into a training subset and a validation subset for cross-validating the model against whatever hyperparameters in the model that are needed to be optimized.

For Logistic Regression, there is a parameter C, which is the inverse of regularization strength. Parameter C would be tuned to find the best regularization strength to prevent overfitting problem and it is necessary to standardize features before doing regularization.

### Model Fitting

For three sets of data, the steps of fitting and assessing model are similar. Thus, only steps of implementing logistic regression to the original dataset are specified as an example.

Step 1: Import necessary libraries and data

Firstly, libraries including pandas, numpy and functions that will be used in latter parts from scikit learn library are imported. Training, validation and test data are imported separately.

Step 2: Standardize features

Since there is a regularization penalty parameter to be tuned later, features are needed to be standardized before regularization. Standard Scalar from sklearn.preprocessing is used to standardize X\_train of original data.

The validation data and test data should also be standardized by using means and standard deviations from training dataset for the same features.

scaler\_or = StandardScaler()

X\_train\_scaled\_or=scaler\_or.fit\_transform(X\_train\_or)

X\_val\_scaled\_or=scaler\_or.transform(X\_val\_or)

X\_test\_scaled\_or=scaler\_or.transform(X\_test\_or)

Step 3: Deal with imbalanced data

As shown in EDA part, the data is imbalanced with a very low proportion of positive classes. In order to avoid classifier always classifying new data as negative, which is a cheating for achieving high accuracy score, BalancedBaggingClassifier dealing with imbalanced data from the imbalanced-learn library is imported. Balanced bagging classifier draws subsamples using under-sampling method, which is the main difference to bagging classifier.

Base\_estimator: logistic regression is the base estimator of balanced bagging classifier to fit subsets randomly sampled from the dataset.

N\_estimators: the number of estimator is set to 50 in order to draw as many balanced subsamples as possible to reduce bias from sampling.

Max\_samples: the number of class 1 in the training set is calculated, and the subsample size is set to twice the number of class 1 to balance the number of two classes.

Replacement: samples are drawn with replacement.

bagging\_logit\_or = BalancedBaggingClassifier(LogisticRegression(random\_state=0),n\_estimators=50, max\_samples=samples\_number, replacement=True, random\_state=0)

Step 4: Tuning parameter

There is a parameter C indicating the inverse strength of regularization. Larger the C, weaker the regularization. To tune this parameter, grid search method is adopted. The best estimator is printed after grid search.

logit\_grid\_or = GridSearchCV(estimator=bagging\_logit\_or, param\_grid=C\_range, cv=5)

Estimator: the estimator of grid search cross validation is balanced bagging classifier specified in step 3.

Param\_grid: values of C to be searched are 0.001, 0.01, 0.1, 1, 10, 100.

Cv: 5-fold cross validation is chosen to reduce computational burden.

Step 5: Find suitable threshold

As explained in model description, the lower threshold might be preferred in this case, because we want to predict as much potential default as possible to positive class.

The predicted probabilities are recorded, and for each threshold, if the predicted probability is greater than the threshold, this case will be predicted as positive. F1-scores corresponding to each threshold are recorded. The threshold generating the highest f1-score is considered as the best one.

y\_val\_prob\_or=logit\_grid\_or.predict\_proba(X\_val\_scaled\_or)[:,1]

f1\_logit\_or=[]

threshold=[0.1,0.2,0.3,0.4,0.5]

for i in range(len(threshold)):

y\_val\_pred\_or=(y\_val\_prob\_or>=threshold[i]).astype(int)

f1\_or=f1\_score(y\_val, y\_val\_pred\_or)

f1\_logit\_or.append(f1\_or)

Step 6: Performance measurement

Finally, confusion matrix, classification report, accuracy score and roc-auc score are calculated for prediction based on the most suitable threshold decided in step 5.

### Summary of findings and justifications

Results of three logistic regression models fitting to three datasets with original features, features selected and selected features with bureau features are shown as follow:

|  |  |  |  |
| --- | --- | --- | --- |
|  | Accuracy Score | AUC Score | F1-score |
| In original data | 0.677 | 0.680 | 0.255 |
| Without additional features | 0.743 | 0.593 | 0.206 |
| With additional features | 0.743 | 0.593 | 0.206 |

Table 1 Performance Measurement for Logistic Regression

From the table, models fitting to dataset with and without additional features have the highest accuracy score 0.743 while logistic regression model fitting to the original data only has accuracy of 0.677. The higher accuracy implies that by using selected features, no matter whether additional features are used or not, the ratio of correct predictions to the total predictions is higher than using original features.

As for AUC score indicating area under the ROC curve, higher the AUC, better the performance of classification model. The AUC score for using original data is 0.68, which is roughly 0.1 higher than AUC for models with or without additional features.

Generally speaking, f1-score contains information that we are most interested in by combining precision and recall to avoid influencing of cheating. Higher the F1-score, better the performance of classifier, since we want both precision indicating proportion of true positive in positive predictions and recall representing proportion of true positive in all actual positive cases to be as high as possible. F1-score in the table shows the similar result with AUC score. Original dataset generates model with the highest f1-score 0.255 while datasets with and without additional features have f1-score only 0.206.

Performance measurements for data with and without additional features are the same since these two datasets have little difference between each other. By comparing criterion that we are more interested in—f1-score and AUC, model using original features is preferred.

## K-Nearest Neighbor Classifier

### Model Descriptions

K-Nearest Neighbor Classifier is an algorithm classifying a new object according to classes of its k nearest neighbors. Euclidean distance is commonly used to calculate distance between cases and cases with the closest distance are defined as neighbors. K-nearest neighbor classifier is a supervised learning algorithm without training phase.

In this case, according to KNN classifier, a new client will be classified as default if majority of its k neighbors are default and vice versa. Number of neighbors is the parameter to be tuned using cross validation method.

### Data Preparations

Similar to the preparations for logistic regression model, we removed rows with missing values of more than 45%, dealt with the data imbalance issue using down-sampling, and generated two more sets of features after using feature selection techniques on both original dataset and the expanded dataset.

Features in dataset have large difference in scales. Since KNN classifier requires calculation of distance, features with small scale become uninformative while features with large scale are more valued by the algorithm. Thus, before implementing KNN classifier, data normalization is applied to avoid this problem.

Finally, the training data are further split into a training subset and a validation subset for cross-validating the model against whatever hyperparameters in the model that are needed to be optimized.

### Model Fitting

For three sets of data, the steps of fitting and assessing model are similar. Thus, only steps of implementing k nearest neighbor algorithm to the original dataset are specified as an example.

Step 1: Import necessary libraries and data

Firstly, libraries including pandas, numpy and functions that will be used in latter parts from scikit learn librarys are imported. Training, validation and test data are imported separately.

Step 2: Normalize features

Features with different scale will have great impact on implementing KNN method which depends on calculating distance between data points. In this case, features have large scaling difference and there are many categorical features which are inverted to numerical by using one-hot encoding. Thus, normalization rescaling data into range [0,1] is considered more suitable than standardization in this case. For rescaled data z, the formula is:

MinMaxScaler from sklearn.preprocessing is used to normalize X\_train of original data. The validation data and test data should also be normalized by using maximums and minimums from training dataset for the same features.

mms\_or = MinMaxScaler()

X\_train\_mms\_or=mms\_or.fit\_transform(X\_train\_or)

X\_val\_mms\_or=mms\_or.transform(X\_val\_or)

Step 3: Deal with imbalanced data

Same with logistic regression part, in order to deal with imbalanced data, BalancedBaggingClassifier from imbalanced learn library is used.

However, parameters are adjusted according to KNN.

Base\_estimator: KNeighborsClassifier is the base estimator of balanced bagging classifier to fit subsamples randomly down-sampled from the dataset.

N\_estimators: the number of estimator is set to 9. Since the computational burden increases dramatically for KNN with high dimensional feature space, we can only set a relatively small number of estimators. Odd number is chosen for binary classification problem.

Max\_samples: the number of class 1 in the training set is calculated, and the subsample size is set to twice the number of class 1 to balance the number of two classes.

Replacement: samples are drawn with replacement.

bagging\_KNN\_or = BalancedBaggingClassifier(KNeighborsClassifier(), n\_estimators=9, max\_samples=samples\_number, replacement=True, random\_state=0)

Step 4: Tuning parameter

We are going to use RandomizedSearchCV from sklearn to tune parameter n\_neighbors in KNN. RandomizedSearchCV is adopted instead of GridSearchCV for the purpose of reducing computational cost.

KNN\_grid\_or = RandomizedSearchCV(estimator=bagging\_KNN\_or, param\_distributions=K\_range\_or,n\_iter=5, cv=5)

Estimator: the estimator of randomized search cross validation is balanced bagging classifier specified in step 3.

Param\_grid: values of n\_neighbors to be searched are odd numbers in the range from 3 to 23.

N\_iter: the number of iterations is only set to 5 because there are too many features in original dataset. For dataset with selected features, default option with 10 iterations is used

Cv: 5-fold cross validation is chosen to reduce computational burden.

Step 5: Performance measurement

Class of validation data is predicted by using the best model found in step 4. Finally, confusion matrix, classification report, accuracy score and roc-auc score are calculated.

### Summary of findings and justifications

Results of three K-nearest-neighbor classifiers fitting to three datasets with original features, features selected and selected features with bureau features are shown as follow:

|  |  |  |  |
| --- | --- | --- | --- |
|  | Accuracy Score | AUC Score | F1-score |
| In original data | 0.631 | 0.609 | 0.203 |
| Without additional features | 0.536 | 0.565 | 0.172 |
| With additional features | 0.534 | 0.572 | 0.176 |

Table 2 Performance Measurement for KNN

From the table above, KNN classifier fitting to dataset with original features has the highest accuracy score 0.631, followed by dataset without additional features 0.536, and dataset with additional features generates the smallest accuracy score 0.534. The lower accuracy implies that by using selected features, no matter whether additional features are used or not, the ratio of correct predictions to the total predictions is lower than using original features.

As for AUC score indicating area under the ROC curve, higher the AUC, better the performance of classification model. The AUC score for using original data is 0.609, while the other two datasets result in AUC score about 0.57, which indicates that using original features produces better prediction result in terms of AUC score.

Similar to AUC score and accuracy score, KNN with original features achieved the highest f1-score 0.203 compared to the other two datasets. KNN classifier for dataset with additional features has the second largest f1-score 0.176, which is 0.04 higher than f1-score for dataset without additional features.

Performance measurements for data with and without additional features, which are the same for logistic regression, are different to each other for KNN classifier. The difference comes from using randomized search cross validation rather than grid search cross validation when tuning parameter.

Based on accuracy score, AUC score or f1 score, the KNN model using the original features outperforms models with the other two group of features. By comparing f1-score, KNN model with additional features is preferred to model without additional features.

## Extreme Gradient Boosting

### Model Descriptions

Extreme Gradient Boosting is an ensemble technique that applies gradient boosted decision trees in the large scale with greater efficiency and accuracy. Gradient tree boosting uses a tree model hm with mth iteration. Firstly, the technique fits a tree hm to the pseudo-residuals (-gn/en) while keeping the tree structure as J regions: R1, R2, … Rj. The tree predicts value of each region with the particular inputs into the regions. The second step is to optimize learning models by minimizing the loss function with regards to β. Extreme Gradient boosting can be used in both classification and regression problems.

### Data Preparations

XGBoost only handles numerical data, to that end, converting categorical and ordinal data is essential in data preprocessing. One of the advantages of XGBoost is able to deal with missing values with spare aware implementation. Moreover, imbalanced dataset would be handled by tuning **scale\_pos\_weight** to balance the positive and negative class weights (in this case, class 1 and class 0).

Tuning Parameters with Grid Search

XGBoost is imported to solve classification problem in this case, then the model is defined as XGBClassifier with hyperparameter tuning. In the tuning parameter process, we will go through 3 steps with 3 sets of hyperparameters.

**Step 1:** Define the fixed parameters to tune tree-based parameters:

**learning\_rate =0.1**: boosting learning rate which controls the model complexity

**subsample=0.8**: XGB randomly sample 80% training data before growing trees

**colsample\_bytree=0.8**: randomly sample 80% of columns when constructing each tree

**objective= 'binary:logistic'**: specify learning objective as logistic regression for binary classification problems with probability outputs

**nthread=4:** the number of paralled threads

**scale\_pos\_weight = ratio**: to balance off imbalanced dataset in which ratio is defined as sum(negative instances) / sum(positive instances)

**seed = 30**: random number seed

n\_estimators=150

**Step 2:** Avoid overfitting by tuning Tree Booster Parameters. Due to the time and speed constraints, we subset these parameters into 2 sets:

Set A :

**'max\_depth'**:np.arange(3,10,2),

**'min\_child\_weight'**:np.arange(1,6,2)

Set B: **'gamma'**:[i/10.0 for i in range(0,5)]

**Step 3:** Tune regularization parameters: apply L2 regularisation term on weights

**'reg\_lambda'**:[1e-5, 1e-2, 0.1, 1, 100]

**Step 4:** Use gridsearch to find the best paramaters for the model and the defined parameters are:

**scoring='roc\_auc'**: use area under an ROC curve as scoring strategy for binary classification problem

**cv = 5**: use 5 folds in cross validation

### Model Fitting – Using Full Set of Original Features

After tuning highlighted parameters with grid search, the final model is:

XGBClassifier(base\_score=0.5, booster='gbtree', colsample\_bylevel=1,

colsample\_bytree=0.8, **gamma=0.4**, learning\_rate=0.1,

max\_delta\_step=0, **max\_depth=3, min\_child\_weight=3**, missing=None,

n\_estimators=150, n\_jobs=1, nthread=4, objective='binary:logistic',

random\_state=0, reg\_alpha=0, **reg\_lamda=1**,

scale\_pos\_weight=11.410717980550382, seed=30, silent=True,

subsample=0.8)

### Model Fitting – Using Selected Features Without Addition Features

The final model is shown in the following:

XGBClassifier(base\_score=0.5, booster='gbtree', colsample\_bylevel=1,

colsample\_bytree=0.8, **gamma=0.1**, learning\_rate=0.1,

max\_delta\_step=0, **max\_depth=3, min\_child\_weight=1**, missing=None,

n\_estimators=150, n\_jobs=1, nthread=4, objective='binary:logistic',

random\_state=0, reg\_alpha=0, **reg\_lamda=1**,

scale\_pos\_weight=11.410717980550382, seed=30, silent=True,

subsample=0.8)

### Model Fitting – Using Selected Features With Addition Features

The final model is

XGBClassifier(base\_score=0.5, booster='gbtree', colsample\_bylevel=1,

colsample\_bytree=0.8, gamma=0.4, learning\_rate=0.1,

max\_delta\_step=0, **max\_depth=3, min\_child\_weight=5**, missing=None,

n\_estimators=150, n\_jobs=1, nthread=4, objective='binary:logistic',

random\_state=0, reg\_alpha=0, **reg\_lamda=1**,

scale\_pos\_weight=11.410717980550382, seed=30, silent=True,

subsample=0.8)

### Summary of findings and justifications

|  |  |  |  |
| --- | --- | --- | --- |
|  | Accuracy Score | AUC Score | F1-score |
| In original data | 0.7053 | 0.7377 | 0.2618 |
| Without additional features | 0.8806 | 0.6150 | 0.0758 |
| With additional features | 0.8857 | 0.6139 | 0.0689 |

Table 3 Performance Measurement for Extreme Gradient Boosting

As shown in the table, XGB performs best on the original dataset based on F1-Score. Since the dataset is imbalanced which has large number of actual negatives, gauging performance with F1 score might be better. F1-score balances out the precision and recall where the business costs hinge on false negative and false positive, while accuracy score merely measures the large number of True Negative. The accuracy score of models with selected features outperforms the rest, but it gives a blind eye on predicting the majority class (class 0) in which would cause high variance and overfitting.

AUC score displays the performance of classification in terms of True Positive Rate and False Negative Rate. It gauges on prediction ranks not their values. In this case, XGB prediction are 73.77% correct on the original dataset.

## Random Forest

### Model Descriptions

Random Forest is an ensemble, bagging technique that consists of many decision trees and outputs the class that is the mode of the class's output by individual trees. It provides an improvement over bagged trees by way of a random small tweak that decorrelates the trees.

### Data Preparations

Similar to the preparations for logistic regression model, we removed rows with missing values of more than 45%, dealt with the data imbalance issue using down-sampling, and generated two more sets of features after using feature selection techniques on both original dataset and the expanded dataset.

Finally, the training data are further split into a training subset and a validation subset for cross-validating the model against whatever hyperparameters in the model that are needed to be optimized.

* + 1. Model fitting – Tuning Parameters with Grid Search

BalancedRandonForestClassifier is imported to deal with the imbalanced data, and RandomizedSearchCV are employed for 5-fold cross-validation.

To solve classification problem in this case, the following hyperparameter tunings are specified in the RandomizedSearchCV by the following code:

brf\_classifier = BalancedRandomForestClassifier(n\_estimators=100, n\_jobs=-1, random\_state=1)

* max\_depth: the maximum depth of the tree is specified a range from 5 to 30 with internals of one.
* min\_samples\_leaf: the minimum number of samples required to be at a leaf node is specified a range from 5 to 30 with internals of one.

The parameters that were not tuned are shown below with justifications:

* N\_estimators: the higher the better, choose 100 to trade off timing
* criterion='gini': we have gini index and cross-entropy to choose, the default gini index is chosen as they have the same effect
* Bootstrap: leave as True to do the bootstrap reshuffling
* N\_jobs is set to be -1 to maximise running cores in the computers
* Random\_state =1 get fix the result at a certain state each time running this program

Model Fitting Outcome – Using Full Set of Original Features

Model Fitting Outcome – Using Selected Features Without Addition Features

The best parameters are found by grid search to be:

• min\_samples\_leaf = 10

• max\_depth = 5

Model Fitting Outcome – Using Selected Features With Addition Features

The best parameters are found by grid search to be:

* min\_samples\_leaf = 10
* max\_depth = 5

### Summary of findings and justifications

|  |  |  |  |
| --- | --- | --- | --- |
|  | Accuracy Score | AUC Score | F1-score |
| Without additional features | 0.631 | 0.592 | 0.193 |
| With additional features | 0.635 | 0.501 | 0.192 |

Table 4 Performance Measurement for Random Forest

As shown in Table 4, Random Forest performs slightly better without the additional features according to F1-Score and AUC score, however, slightly worse in terms of accuracy.

# Conclusions and Predictions

## Conclusions on Model Selections

Overall, the decision is made based on the F1-Score performance, thus Extreme Boosting method was employed for the final prediction. Unfortunately, some information was lost in the feature selection process because our use of the unsupervised learning method, which led to unexpected feature performances.

## Prediction Outcomes

For the prediction results, please refer to the document: ‘Group26\_Results\_Bereau’

# References:

MaxHalford/prince. (2018). Retrieved from https://github.com/MaxHalford/prince

# Appendices

## Appendix – Python Code

**Preprocessing**

# coding: utf-8

# <h1><center>Group Assignment</h1>

# ## 1. Data Preprocessing

# ### 1.1 Data Loading

# In[1]:

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

import warnings

warnings.filterwarnings('ignore')

# Plot settings

sns.set\_context('notebook')

sns.set\_style('ticks')

colours = ['#1F77B4', '#FF7F0E', '#2CA02C', '#DB2728', '#9467BD', '#8C564B', '#E377C2','#7F7F7F', '#BCBD22', '#17BECF']

crayon = ['#4E79A7','#F28E2C','#E15759','#76B7B2','#59A14F', '#EDC949','#AF7AA1','#FF9DA7','#9C755F','#BAB0AB']

sns.set\_palette(colours)

get\_ipython().run\_line\_magic('matplotlib', 'inline')

plt.rcParams['figure.figsize'] = (9, 6)

# In[255]:

train = pd.read\_csv('Data/ProjectTrain.csv',index\_col='SK\_ID\_CURR')

train\_bur = pd.read\_csv('Data/ProjectTrain\_Bureau.csv',index\_col='SK\_ID\_CURR')

test = pd.read\_csv('Data/ProjectTest.csv',index\_col='Index\_ID')

test\_bur = pd.read\_csv('Data/ProjectTest\_Bureau.csv',index\_col='Index\_ID')

# 50% + filtering numerical transform

train\_trans\_partial = pd.read\_csv('Data/train\_clean\_trans.csv', index\_col='SK\_ID\_CURR')

test\_trans\_partial = pd.read\_csv('Data/test\_clean\_trans.csv', index\_col='Index\_ID')

# # full set numerical transform

train\_trans\_full = pd.read\_csv('Data/train\_trans\_fullset.csv',index\_col='SK\_ID\_CURR')

test\_trans\_full = pd.read\_csv('Data/test\_trans\_fullset.csv',index\_col='Index\_ID')

# full imputed

X\_train\_filled = pd.read\_csv('Data/X\_train\_filled.csv',index\_col='SK\_ID\_CURR')

X\_test\_filled = pd.read\_csv('Data/X\_test\_filled.csv',index\_col='Index\_ID')

# # full imputed

X\_train\_oh = pd.read\_csv('Data/X\_train\_oh.csv',index\_col='SK\_ID\_CURR')

X\_test\_oh = pd.read\_csv('Data/X\_test\_oh.csv',index\_col='Index\_ID')

# Merged Data

train\_join = train.merge(train\_bur, how='left', left\_on='SK\_ID\_CURR', right\_on='SK\_ID\_CURR')

test\_join = test.merge(test\_bur, how='left', left\_on='Index\_ID', right\_on='Index\_ID')

train\_join\_trans = pd.read\_csv('Data/train\_join\_trans.csv',index\_col='SK\_ID\_CURR')

test\_join\_trans = pd.read\_csv('Data/test\_join\_trans.csv',index\_col='Index\_ID')

pearson\_corr\_features\_join = pd.read\_csv('Data/feature\_join\_corr.csv',index\_col='index')

test\_join.loc[test\_join['CREDIT\_ACTIVE'] == 'Bad debt', 'CREDIT\_ACTIVE'] = 'Closed'

X\_train\_famd\_ne.shape

# ### 1.2 Feature Selection

# #### 1.2.1 Filtering for Feature Selection

# ##### 1.2.1.1 Fold Change

# In[165]:

def fold\_change(train):

# data preparation

train\_target = train['TARGET'].copy # training target

train\_features = train.iloc[:,1:].copy() # training features

# partition feature into two groups by target & Mean calc

train\_feature\_oneMean = train\_features.loc[train['TARGET'] == 1,:].mean()

train\_feature\_zeroMean = train\_features.loc[train['TARGET'] == 0,:].mean()

# fold change calculation

train\_fold\_change = pd.DataFrame(np.absolute(np.log2(np.divide(train\_feature\_oneMean, train\_feature\_zeroMean))))

train\_fold\_change.columns = ['Mean'] # rename mean division result column

train\_fold\_change\_sort = train\_fold\_change.sort\_values(by=['Mean'],ascending=False) # fold change calculation

return train\_fold\_change\_sort

# ##### 1.2.1.1 Pearson's Correlation

# In[99]:

def Pearson\_corr(train):

# data preparation

train\_t = train.copy()

train\_features = train\_t.drop(columns=['TARGET']).copy() # training features

# corr calculation

train\_corr = pd.DataFrame((train\_features.corrwith(train['TARGET'])).abs())

train\_corr.columns = ['Corr']

train\_corr\_sort = train\_corr.sort\_values(by=['Corr'],ascending=False)

return train\_corr\_sort

# #### 1.2.2 Filter Out Columns with 50%+ Missing Val

# In[8]:

def MissingValFilter(train, test):

# pass value to new vars

train\_new = train.copy()

test\_new = test.copy()

# the missing proportion in each column

miss\_propor = train\_new.isna().sum()/train.shape[0]

# filter out the column if the missing proportion is larger than 45%

for i in range(0,train\_new.shape[1]):

if miss\_propor[i] > 0.45:

del train\_new[miss\_propor.index[i]]

del test\_new[miss\_propor.index[i]]

# filter out columns with only 1 level

# 1 level index retrive

level\_summary = train\_new.nunique()

one\_level\_index = level\_summary[level\_summary==1].index

# 1 level column filtering

if one\_level\_index != 'nan':

for i in range(0,one\_level\_index.shape[0]):

del train\_new[one\_level\_index[i]]

del test\_new[one\_level\_index[i]]

return train\_new, test\_new

# ### 1.3 Encoding

# #### 1.3.1 Nominal Data Encoding

# In[75]:

from sklearn.preprocessing import LabelEncoder

def NumericalTransform(train\_clean, test\_clean):

# pass value to new pars

train\_clean\_trans\_tmp = train\_clean.copy()

test\_clean\_trans\_tmp = test\_clean.copy()

# retrieve column types

types = train\_clean\_trans\_tmp.dtypes

types\_num = types[types != 'object'].index # numerical type colnames

types\_cat = types[types == 'object'].index # numerical type colnames

# numerical & categorial columns partition

# numerical cols

train\_clean\_trans\_num = train\_clean\_trans\_tmp[types\_num].copy() # train

test\_clean\_trans\_num = test\_clean\_trans\_tmp[types\_num[1:]].copy() # test

# cat cols

train\_clean\_trans\_cat = train\_clean\_trans\_tmp[types\_cat].copy() # train

test\_clean\_trans\_cat = test\_clean\_trans\_tmp[types\_cat].copy() # test

# Categorical col encoding - label encoder

for i in range(0,types\_cat.shape[0]):

le = LabelEncoder()

# fit with the desired col, col in position 0 for this example

fit\_by = pd.Series([i for i in train\_clean\_trans\_tmp[types\_cat[i]].unique() if type(i) == str]) # train

le.fit(fit\_by)

# Set transformed col leaving np.NaN as they are

train\_clean\_trans\_cat[types\_cat[i]] = train\_clean\_trans\_tmp[types\_cat[i]].apply(lambda x: le.transform([x])[0] if type(x) == str else x) # train

test\_clean\_trans\_cat[types\_cat[i]] = test\_clean\_trans\_tmp[types\_cat[i]].apply(lambda x: le.transform([x])[0] if type(x) == str else x) # test

# cocat Numerical & Categorial Cols tgt

# train

frame = [train\_clean\_trans\_num,train\_clean\_trans\_cat]

train\_clean\_trans = pd.concat(frame,axis=1)

# test

frame = [test\_clean\_trans\_num,test\_clean\_trans\_cat]

test\_clean\_trans = pd.concat(frame,axis=1)

return train\_clean\_trans, test\_clean\_trans

# #### 1.3.2 Train Validation Sample

# In[76]:

def trainValidateSample(train,y\_train):

# pass values to temp pars

train\_t = train.copy()

y\_train\_t = y\_train.copy()

# sample the selected training set

train\_ex = train\_t.sample(frac = 0.2185, replace = False, random\_state = 1) # features

y\_train\_ex = y\_train[train\_ex.index] # target

# sample the selected validation set

oppSubSample\_index = train\_t.index.isin(train\_ex.index)

subSample\_val = train\_t[~oppSubSample\_index] # rebulid validation subsample

validate\_ex = subSample\_val.sample (frac = 0.120, replace = False, random\_state = 1)

y\_validate\_ex = y\_train[validate\_ex.index]

return train\_ex, y\_train\_ex, validate\_ex, y\_validate\_ex

# #### 1.3.3 Categorical String Transformation

# In[107]:

from sklearn.preprocessing import OneHotEncoder

def oneHotEncoding(X\_train\_filled, X\_validate\_filled,X\_test\_filled, train, pearson\_corr\_features):

# pass value to new pars

X\_train\_filled\_t = X\_train\_filled.copy()

X\_validate\_filled\_t = X\_validate\_filled.copy()

X\_test\_filled\_t = X\_test\_filled.copy()

train\_t = train.copy()

pearson\_corr\_features\_t = pearson\_corr\_features.copy()

pearson\_corr\_features\_t = pearson\_corr\_features\_t.drop(labels=['EXT\_SOURCE\_1'])

# retrieve column types

types = train\_t.dtypes

types\_cat = types[types=='object'].index # categorial type colnames

unique\_count = train\_t.nunique()

types\_bin2 = unique\_count[unique\_count==2].index[1:] # binary type colnames (without nan)

types\_bin3 = unique\_count[unique\_count==3].index # binary type colnames (with nan)

pearson\_corr\_features\_t = pearson\_corr\_features\_t.loc[pearson\_corr\_features\_t['Corr']>=0.01,:].index

# extract binary and categorial features

features\_ex\_cat=[]

features\_ex\_num=[]

for i in range(0,len(pearson\_corr\_features\_t)):

if ((pearson\_corr\_features\_t[i] in types\_cat.unique())|(pearson\_corr\_features\_t[i] in types\_bin2.unique())|(pearson\_corr\_features\_t[i] in types\_bin3.unique())):

features\_ex\_cat.append(pearson\_corr\_features\_t[i])

else:

features\_ex\_num.append(pearson\_corr\_features\_t[i])

# convert to object type

for col in range(0,len(features\_ex\_cat)):

X\_train\_filled\_t[features\_ex\_cat[col]] = X\_train\_filled\_t[features\_ex\_cat[col]].astype(str)

X\_validate\_filled\_t[features\_ex\_cat[col]] = X\_validate\_filled\_t[features\_ex\_cat[col]].astype(str)

X\_test\_filled\_t[features\_ex\_cat[col]] = X\_test\_filled\_t[features\_ex\_cat[col]].astype(str)

return X\_train\_filled\_t, X\_validate\_filled\_t, X\_test\_filled\_t

# ### 1.4 Missing Value Imputation

# #### 1.4.1 MICE

# In[93]:

# MICE Imputer

from fancyimpute import IterativeImputer

def MiceImpute(train\_clean\_trans, validate\_clean\_trans, test\_clean\_trans):

imputer = IterativeImputer(n\_iter=5, sample\_posterior=True, random\_state=1)

imputer.fit(train\_clean\_trans)

X\_train\_filled\_mice = pd.DataFrame(imputer.transform(train\_clean\_trans)) # train

X\_validate\_filled\_mice = pd.DataFrame(imputer.transform(validate\_clean\_trans)) # validate

X\_test\_filled\_mice = pd.DataFrame(imputer.transform(test\_clean\_trans)) # test

# rename the column names

X\_train\_filled\_mice.columns = train\_clean\_trans.columns # train

X\_validate\_filled\_mice.columns = validate\_clean\_trans.columns # validate

X\_test\_filled\_mice.columns = test\_clean\_trans.columns # test

# rename the row

X\_train\_filled\_mice.index = train\_clean\_trans.index # train

X\_validate\_filled\_mice.index = validate\_clean\_trans.index # validate

X\_test\_filled\_mice.index = test\_clean\_trans.index # test

return X\_train\_filled\_mice, X\_validate\_filled\_mice, X\_test\_filled\_mice

# #### 1.4.2 MODE

# In[79]:

# Mode Imputer

from sklearn.preprocessing import Imputer

def SimpleImpute(train\_clean\_trans, test\_clean\_trans,stra):

imputer = Imputer(strategy=stra)

X\_train\_filled\_mode = imputer.fit\_transform(train\_clean\_trans) # train

X\_test\_filled\_mode = pd.DataFrame(imputer.transform(test\_clean\_trans)) # test

# rename the column names

X\_train\_filled\_mode.columns = train\_clean\_trans.columns # train

X\_test\_filled\_mode.columns = test\_clean\_trans.columns # test

# rename the row

X\_train\_filled\_mode.index = train\_clean\_trans.index # train

X\_test\_filled\_mode.index = test\_clean\_trans.index # test

return X\_train\_filled\_mode, X\_test\_filled\_mode

# #### 1.4.3 Final Impute

# In[96]:

def final\_impute(train, validate, test, pearson\_corr\_features):

train\_t = train.copy()

validate\_t = validate.copy()

test\_t = test.copy()

# MNAR Colnames

mnar\_colnames = pd.read\_csv('Data/col\_mnar.csv',index\_col='index')

mnar\_colnames = (mnar\_colnames.loc[mnar\_colnames['val'] == -1,:]).index

# extract feature with 1%+ features

pearson\_corr\_features = pearson\_corr\_features.loc[pearson\_corr\_features['Corr']>=0.01,:].index

# Total extracted data

train\_t\_ex = train\_t[pearson\_corr\_features].copy()

validate\_t\_ex = validate\_t[pearson\_corr\_features].copy()

test\_t\_ex = test\_t[pearson\_corr\_features].copy()

# MNAR

train\_t\_ex[mnar\_colnames] = train\_t\_ex[mnar\_colnames].fillna(100)

validate\_t\_ex[mnar\_colnames] = validate\_t\_ex[mnar\_colnames].fillna(100)

test\_t\_ex[mnar\_colnames] = test\_t\_ex[mnar\_colnames].fillna(100)

# drop EXT\_SOURCE\_1 col since too much missing

train\_t\_ex = train\_t\_ex.drop(columns=['EXT\_SOURCE\_1'])

validate\_t\_ex = validate\_t\_ex.drop(columns=['EXT\_SOURCE\_1'])

test\_t\_ex = test\_t\_ex.drop(columns=['EXT\_SOURCE\_1'])

# MICE

X\_train\_filled\_mice, X\_validate\_filled\_mice, X\_test\_filled\_mice = MiceImpute(train\_t\_ex, validate\_t\_ex, test\_t\_ex)

return X\_train\_filled\_mice, X\_validate\_filled\_mice, X\_test\_filled\_mice

# ### 1.5 Factor Analysis

# #### 1.5.1 FAMD

# In[108]:

from prince import FAMD

import fbpca

def princeFAMD (train, validate, test, n\_comp, n\_iter):

famd = FAMD(n\_components=n\_comp, n\_iter=n\_iter, copy=True, engine='auto', random\_state=4)

# fit transform

train\_trans = famd.fit\_transform(train)

validate\_trans = famd.fit\_transform(validate)

test\_trans = famd.fit\_transform(test)

return train\_trans, validate\_trans, test\_trans

# ## 2. Model Fitting

# ### 2.1 Benchmark KNN

# In[152]:

from sklearn.neighbors import KNeighborsClassifier

from sklearn.ensemble import RandomForestClassifier

from sklearn.model\_selection import cross\_val\_score

def BmModel(X\_train\_filled, y\_train):

# # # Benchmark Knn

# model\_knn = KNeighborsClassifier()

# # cross validation score

# scores\_knn = cross\_val\_score(model\_knn, X\_train\_filled, y\_train, cv=5, n\_jobs=-1, scoring = 'accuracy')

# cv\_score\_knn = scores\_knn.mean() # avg cv score

# print('Cross Validation Score in knn Benchmark:', cv\_score\_knn.round(3), '\n')

# Benchmark RandomForest

model\_rf = RandomForestClassifier()

# scores\_rf\_raw = cross\_val\_score(model\_rf, X\_train\_raw, y\_train, cv=5, n\_jobs=-1, scoring = 'roc\_auc')

scores\_rf = cross\_val\_score(model\_rf, X\_train\_filled, y\_train, cv=5, n\_jobs=-1, scoring = 'roc\_auc')

# avg cv score

# cv\_score\_rf\_raw = scores\_rf\_raw.mean()

cv\_score\_rf = scores\_rf.mean()

# print('Cross Validation Score in random forest Benchmark (Raw Data):', cv\_score\_rf\_raw.round(3), '\n')

print('Cross Validation Score in random forest Benchmark:', cv\_score\_rf.round(3), '\n')

return cv\_score\_rf

# ## 3. Main

# #### 3.1 Data Preprocessomg

# In[146]:

'''

SET 1: Original set generation

'''

## STEP 1: extract TARGET N Numerical Transform

y\_train = train['TARGET'] # extract the training target

train\_trans\_full, test\_trans\_full = NumericalTransform(train, test) #numerical transform of the whole dataset

## STEP 2: resample of the sub training n validation set

train\_sub, y\_train\_sub, validate\_sub, y\_validate\_sub = trainValidateSample(train\_trans\_full,y\_train)

## STEP 3: drop the target column in feature set

train\_sub, validate\_sub = train\_sub.drop(columns=['TARGET']), validate\_sub.drop(columns=['TARGET'])

STEP 4: impute missing value

X\_train\_filled\_original, X\_validate\_filled\_original, X\_test\_filled\_original = MiceImpute(train\_sub, validate\_sub, test\_trans\_full)

## STEP 5: output the data file

# training set

X\_train\_filled\_original.to\_csv('Data/Original\_X\_train.csv')

y\_train\_sub.to\_csv('Data/Original\_y\_train.csv')

# hold-out set

X\_validate\_filled\_original.to\_csv('Data/Original\_X\_validate.csv')

y\_validate\_sub.to\_csv('Data/Original\_y\_validate.csv')

# test set

X\_test\_filled\_original.to\_csv('Data/Original\_X\_test.csv')

'''

SET 2: Without Extra Features Set Generation

'''

## STEP 4: Pearson Correlation Calculation

pearson\_corr\_features = Pearson\_corr(train\_trans\_full)

# STEP 5: impute missing values

X\_train\_filled\_ne, X\_validate\_filled\_ne, X\_test\_filled\_ne= final\_impute(train\_sub, validate\_sub, test\_trans\_full, pearson\_corr\_features)

## STEP 6: categorial string transformation

X\_train\_oh\_ne, X\_validate\_oh\_ne, X\_test\_oh\_ne = oneHotEncoding(X\_train\_filled\_ne, X\_validate\_filled\_ne,X\_test\_filled\_ne, train, pearson\_corr\_features)

## STEP 7: FAMD factor compression

X\_train\_famd\_ne, X\_validate\_famd\_ne, X\_test\_famd\_ne = princeFAMD(X\_train\_oh\_ne, X\_validate\_oh\_ne, X\_test\_oh\_ne, 40, 5)

## STEP 8: output the data file

# training set

X\_train\_famd\_ne.to\_csv('Data/ne\_X\_train.csv')

# hold-out set

X\_validate\_famd\_ne.to\_csv('Data/ne\_X\_validate.csv')

# test set

X\_test\_famd\_ne.to\_csv('Data/ne\_X\_test.csv')

'''

SET 3: With Extra Features Set Generation

'''

# STEP 1: Numerical Transform

train\_trans\_join, test\_trans\_join = NumericalTransform(train\_join, test\_join) #numerical transform of the whole dataset

# STEP 2: pearson corr calc

pearson\_corr\_features\_join = pd.read\_csv('Data/feature\_join\_corr.csv',index\_col='index')

## STEP 3: resample of the sub training n validation set

train\_join\_sub, y\_train\_join\_sub, validate\_join\_sub, y\_validate\_sub = trainValidateSample(train\_trans\_join,y\_train)

# STEP 4: drop the target column in feature set

train\_join\_sub, validate\_join\_sub = train\_join\_sub.drop(columns=['TARGET']), validate\_join\_sub.drop(columns=['TARGET'])

## STEP 5: impute missing values

X\_train\_filled\_we, X\_validate\_filled\_we, X\_test\_filled\_we= final\_impute(train\_join\_sub, validate\_join\_sub, test\_trans\_join, pearson\_corr\_features\_join)

## STEP 6: categorial string transformation

X\_train\_oh\_we, X\_validate\_oh\_we, X\_test\_oh\_we = oneHotEncoding(X\_train\_filled\_we, X\_validate\_filled\_we,X\_test\_filled\_we, train, pearson\_corr\_features\_join)

## STEP 7: FAMD factor compression

X\_train\_famd\_we, X\_validate\_famd\_we, X\_test\_famd\_we = princeFAMD(X\_train\_oh\_we, X\_validate\_oh\_we, X\_test\_oh\_we, 40, 5)

## STEP 8: output the data file

# training set

X\_train\_famd\_we.to\_csv('Data/we\_X\_train.csv')

y\_train\_join\_sub.to\_csv('Data/we\_y\_train.csv')

## hold-out set

# X\_validate\_famd\_we.to\_csv('Data/we\_X\_validate.csv')

y\_validate\_sub.to\_csv('Data/we\_y\_validate.csv')

## test set

# X\_test\_famd\_we.to\_csv('Data/we\_X\_test.csv')

# #### 3.2 Model Fitting

# In[153]:

cv\_score\_rf = BmModel(X\_train\_famd\_ne, y\_train\_sub)

# In[154]:

cv\_score\_rf = BmModel(X\_train\_oh\_ne, y\_train\_sub)

# In[155]:

cv\_score\_rf = BmModel(X\_train\_filled\_original, y\_train\_sub)

# ## 4. FAMD Visualization

# In[163]:

famd = FAMD(n\_components=40, n\_iter=5, copy=True, engine='auto', random\_state=4)

train\_trans = famd.fit\_transform(X\_train\_oh\_we)

# In[150]:

# FAMD Training

famd = FAMD(n\_components=40, n\_iter=5, copy=True, engine='auto', random\_state=4)

train\_trans = famd.fit\_transform(X\_train\_oh\_ne)

# FAMD Visualization

ax = famd.plot\_row\_coordinates(X\_train\_oh\_ne, ax=None, figsize=(6, 6), x\_component=0,...

y\_component=1, labels=None, color\_labels=y\_train\_sub, ellipse\_outline=False, ...

ellipse\_fill=True, show\_points=True)

# ### 5. Random Forest Model Fitting - Without Extra Features

# In[256]:

from sklearn.model\_selection import GridSearchCV

from sklearn.model\_selection import RandomizedSearchCV

from sklearn.metrics import accuracy\_score, confusion\_matrix, classification\_report, f1\_score,roc\_auc\_score

from sklearn.preprocessing import MinMaxScaler

# from imblearn.ensemble import BalancedBaggingClassifier

from imblearn.ensemble import BalancedRandomForestClassifier

brf\_classifier = BalancedRandomForestClassifier(n\_estimators=100, n\_jobs=-1, random\_state=1)

tuning\_parameters = {

'max\_depth': list(range(5,20,5)),

'min\_samples\_leaf' : list(range(5,20,5)),

# 'sampling\_strategy': ['majority','not minority'],

# 'replacement': [True, False],

# 'class\_weight': [None, 'balanced']

}

brf\_cls\_search = RandomizedSearchCV(brf\_classifier, tuning\_parameters, cv= 5, return\_train\_score=True, scoring='f1')

brf\_cls\_search.fit(X\_train\_famd\_ne, y\_train\_sub)

brf\_cls\_search\_best = brf\_cls\_search.best\_estimator\_

print('Best parameters found by grid search:', brf\_cls\_search.best\_params\_, '\n')

# #### Validation Set Evaluation

# In[266]:

# Prediction and scoring

y\_validate\_sub\_pred = brf\_cls\_search\_best.predict(X\_validate\_famd\_ne)

# y\_test\_sub\_pred.index = X\_test\_famd\_ne.index

# # print(brf\_cls\_search\_best.score(X\_validate\_famd\_ne, y\_validate\_sub))

# #confusion matrix

# confusion\_brf =confusion\_matrix(y\_validate\_sub, y\_validate\_sub\_pred)/y\_validate\_sub.shape

# print(confusion\_brf)

# #performance score

# performance\_brf =classification\_report(y\_validate\_sub, y\_validate\_sub\_pred, digits = 3)

# print(performance\_brf)

# # accuracy score

# accuracy\_brf =round(accuracy\_score(y\_validate\_sub, y\_validate\_sub\_pred),3)

# print(accuracy\_brf)

# auc

# roc\_auc\_score(y\_validate\_sub, y\_validate\_sub\_pred)

# In[265]:

y\_test\_sub\_pred.to\_csv('Data/y\_test\_ne.csv')

# ### 6. Random Forest Model Fitting - With Extra Features

# In[ ]:

brf\_classifier\_we = BalancedRandomForestClassifier(n\_estimators=100, n\_jobs=-1, random\_state=1)

tuning\_parameters = {

'max\_depth': list(range(5,20,5)),

'min\_samples\_leaf' : list(range(5,20,5)),

# 'sampling\_strategy': ['majority','not minority'],

# 'replacement': [True, False],

'class\_weight': [None, 'balanced']

}

brf\_cls\_search\_we = RandomizedSearchCV(brf\_classifier\_we, tuning\_parameters, cv= 5, return\_train\_score=True, scoring='f1')

brf\_cls\_search\_we.fit(X\_train\_famd\_we, y\_train\_join\_sub)

brf\_cls\_search\_we\_best = brf\_cls\_search\_we.best\_estimator\_

print('Best parameters found by grid search:', brf\_cls\_search\_we.best\_params\_, '\n')

# #### Validation Set Evaluation

# In[183]:

# Prediction and scoring

y\_train\_join\_sub\_pred = brf\_cls\_search\_best.predict(X\_validate\_famd\_we)

# print(brf\_cls\_search\_best.score(X\_validate\_famd\_ne, y\_validate\_sub))

#confusion matrix

confusion\_brf\_we =confusion\_matrix(y\_train\_join\_sub, y\_train\_join\_sub\_pred)

print(confusion\_brf\_we)

#performance score

performance\_brf\_we =classification\_report(y\_train\_join\_sub, y\_train\_join\_sub\_pred, digits = 3)

print(performance\_brf\_we)

# accuracy score

accuracy\_brf\_we =round(accuracy\_score(y\_train\_join\_sub, y\_train\_join\_sub\_pred),3)

print(accuracy\_brf\_we)

# ### 7. Random Forest Model Fitting - Original Features

# In[ ]:

brf\_classifier\_original = BalancedRandomForestClassifier(n\_estimators=100, n\_jobs=-1, random\_state=1)

tuning\_parameters = {

'max\_depth': list(range(5,20,5)),

'min\_samples\_leaf' : list(range(5,20,5)),

# 'sampling\_strategy': ['majority','not minority'],

# 'replacement': [True, False],

'class\_weight': [None, 'balanced']

}

brf\_cls\_search\_original = RandomizedSearchCV(brf\_classifier\_original, tuning\_parameters, cv= 5, return\_train\_score=True, scoring='f1')

brf\_cls\_search\_original.fit(X\_train\_famd\_we, y\_train\_join\_sub)

brf\_cls\_search\_original\_best = brf\_cls\_search\_we.best\_estimator\_

print('Best parameters found by grid search:', brf\_cls\_search\_we.best\_params\_, '\n')

# #### Validation Set Evaluation

# In[183]:

# Prediction and scoring

y\_train\_join\_sub\_pred = brf\_cls\_search\_best.predict(X\_validate\_famd\_we)

# print(brf\_cls\_search\_best.score(X\_validate\_famd\_ne, y\_validate\_sub))

#confusion matrix

confusion\_brf\_we =confusion\_matrix(y\_train\_join\_sub, y\_train\_join\_sub\_pred)

print(confusion\_brf\_we)

#performance score

performance\_brf\_we =classification\_report(y\_train\_join\_sub, y\_train\_join\_sub\_pred, digits = 3)

print(performance\_brf\_we)

# accuracy score

accuracy\_brf\_we =round(accuracy\_score(y\_train\_join\_sub, y\_train\_join\_sub\_pred),3)

print(accuracy\_brf\_we)

**EDA**

# coding: utf-8

# <h1><center>Group Assignment</h1>

# ## Data Preparation

# In[2]:

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

import missingno as msno

import warnings

warnings.filterwarnings('ignore')

# Plot settings

sns.set\_context('notebook')

sns.set\_style('ticks')

colours = ['#1F77B4', '#FF7F0E', '#2CA02C', '#DB2728', '#9467BD', '#8C564B', '#E377C2','#7F7F7F', '#BCBD22', '#17BECF']

crayon = ['#4E79A7','#F28E2C','#E15759','#76B7B2','#59A14F', '#EDC949','#AF7AA1','#FF9DA7','#9C755F','#BAB0AB']

sns.set\_palette(colours)

get\_ipython().run\_line\_magic('matplotlib', 'inline')

# plt.rcParams['figure.figsize'] = (9, 6)

# In[4]:

train = pd.read\_csv('Datasets/ProjectTrain.csv',index\_col='SK\_ID\_CURR')

train\_bur = pd.read\_csv('Datasets/ProjectTrain\_Bureau.csv',index\_col='SK\_ID\_CURR')

test = pd.read\_csv('Datasets/ProjectTest.csv',index\_col='Index\_ID')

test\_bur = pd.read\_csv('Datasets/ProjectTest\_Bureau.csv',index\_col='Index\_ID')

# merge

train\_join = train.merge(train\_bur, how='left', left\_on='SK\_ID\_CURR', right\_on='SK\_ID\_CURR')

test\_join = test.merge(test\_bur, how='left', left\_on='Index\_ID', right\_on='Index\_ID')

X\_train\_join = train\_join.iloc[:,1:]

X\_train = train.iloc[:,1:]

y\_train = train.iloc[:,0]

y\_train\_join = train\_join.iloc[:,0]

# joined datasets after labelencoding

train\_join\_trans = pd.read\_csv('Datasets/train\_join\_trans.csv',index\_col='SK\_ID\_CURR')

test\_join\_tans = pd.read\_csv('Datasets/test\_join\_trans.csv',index\_col='Index\_ID')

# Pearson correlation

features\_join\_corr = pd.read\_csv('Datasets/feature\_join\_corr.csv', index\_col='Features')

features\_join\_corr\_20 = pd.read\_csv('Datasets/feature\_join\_corr\_top20.csv', index\_col='Features')

# ### Train Validation Random Sampling

# In[11]:

def trainValidateSample(train,y\_train):

# pass values to temp pars

train\_t = train.copy()

y\_train\_t = y\_train.copy()

# sample the selected training set

train\_ex = train\_t.sample(frac = 0.2185, replace = False, random\_state = 1) # features

y\_train\_ex = y\_train[train\_ex.index] # target

# sample the selected validation set

oppSubSample\_index = train\_t.index.isin(train\_ex.index)

subSample\_val = train\_t[~oppSubSample\_index] # rebulid validation subsample

validate\_ex = subSample\_val.sample (frac = 0.120, replace = False, random\_state = 1)

y\_validate\_ex = y\_train[validate\_ex.index]

return train\_ex, y\_train\_ex, validate\_ex, y\_validate\_ex

# # EDA Reuqirements:

#

# - NA plots

# - Corr plots

# - Selected numerical features' impact for y: conditional distribution

# - Selected numerical features' impact for y: Bar chart

# - Clusters (after FAMD)

# ## 1. Missng value percentage plot

# In[30]:

train\_join.shape

# In[29]:

train\_bur.shape

# In[27]:

train.shape

# In[28]:

test.shape

# In[12]:

# Sample the data

## Exclude extra features

X\_train, y\_train, X\_val, y\_val = trainValidateSample(train, y\_train)

# Include the extra features

X\_train\_join, y\_train, X\_val\_join, y\_val = trainValidateSample(train\_join, y\_train\_join)

## join after transform

X\_train\_join\_trans, y\_train, X\_val\_join\_trans, y\_val = trainValidateSample(train\_join\_trans, y\_train\_join)

# In[33]:

missingdata\_col = X\_train\_join.columns[X\_train\_join.isnull().any()].tolist()

# In[34]:

len(missingdata\_col)

# In[36]:

len(missingdata\_col\_1)

# In[15]:

# select columns that have missing values

missingdata\_col\_1 = X\_train\_join.iloc[:,1:72].columns[X\_train\_join.iloc[:,1:72].isnull().any()].tolist()

missingdata\_col\_2 = X\_train\_join.iloc[:,72:].columns[X\_train\_join.iloc[:,72:].isnull().any()].tolist()

# plot first 40 colums with NA

msno.matrix(X\_train\_join[missingdata\_col\_1])

# Also, the sparkline on the right gives you a summary of the general shape of the data completeness and an indicator of the rows with maximum and minimum rows.

# In[49]:

msno.matrix(X\_train\_join[missingdata\_col\_2])

# In[35]:

missingdata\_col\_2

# ## 2. Correlation plot

# Features with no missing value are excluded in the heatmap. If the nullity correlation is very close to zero (-0.05 < R < 0.05), no value will be displayed.

#

# Also, a perfect positive nullity correlation (R=1) indicates when the first feature and the second feature both have corresponding missing values while a perfect negative nullity correlation (R=-1) means that one of the features is missing and the second is not missing.

# ### 2.1 Nullity relationship

# In[18]:

msno.heatmap(X\_train\_join[missingdata\_col\_1], figsize=(15,15))

# In[19]:

msno.heatmap(X\_train\_join[missingdata\_col\_2], figsize=(15,15))

# ### 2.2 Feature Correlation

# In[20]:

# X\_train\_join\_trans.shape # without target

# ada target

X\_train\_join\_trans['TARGET'] = y\_train\_join

# In[39]:

missingdata\_col\_1

# In[41]:

mask = np.zeros\_like(corr)

mask[np.triu\_indices\_from(mask)] = True

fig = plt.figure(figsize=(20,20))

ax = sns.heatmap(corr, mask=mask, vmax=.3, square=True, cmap='coolwarm',)

fig.savefig('feature\_corr1.png')

# with sns.axes\_style("white"):

# ax = sns.heatmap(corr, mask=mask, vmax=.3, square=True)

# In[23]:

corr = X\_train\_join\_trans.corr()

mask = np.zeros\_like(corr)

mask[np.triu\_indices\_from(mask)] = True

fig = plt.figure(figsize=(20,20))

ax = sns.heatmap(corr[(corr >= 0.5) | (corr <= -0.5)], cmap='coolwarm',mask=mask, vmax=1.0, vmin=-1.0, linewidths=0.1,square=True);

fig.savefig('feature\_corr.png')

# ## 3. Relationship of highly correlated features with response variable

# ### 3.1 Numerical features

# - Inspect outliers

# - Relationship with response variable

# In[8]:

# select numerical feature names

features\_num = features\_join\_corr\_20.loc[features\_join\_corr\_20['Numerical']==0, 'Numerical'].index

# In[9]:

from statlearning import plot\_conditional\_distributions

fig = plt.figure()

fig, ax = plot\_conditional\_distributions(X\_train\_join[features\_num], y\_train\_join, labels=['Not default', 'Default'])

fig.savefig('conditional\_distribution\_numerical.png')

# ### 3.2 Categorical features

# In[133]:

# select numerical feature names

features\_cate = features\_join\_corr\_20.loc[features\_join\_corr\_20['Numerical']==1, 'Numerical'].index

# In[176]:

ct\_region = pd.crosstab(X\_train\_join[features\_cate[0]], X\_train\_join.TARGET).transpose()

ct\_region.plot.bar(stacked=True, figsize=(7,5), fontsize=14, colormap='viridis', alpha=0.7)

plt.show()

# In[163]:

ct\_region = pd.crosstab(X\_train\_join[features\_cate[1]], X\_train\_join.TARGET).transpose()

ct\_region.plot.bar(stacked=True, figsize=(7,5), fontsize=14, colormap='viridis', alpha=0.7)

plt.show()

# In[164]:

ct\_region = pd.crosstab(X\_train\_join[features\_cate[2]], X\_train\_join.TARGET).transpose()

ct\_region.plot.bar(stacked=True, figsize=(7,5), fontsize=14, colormap='viridis', alpha=0.7)

plt.show()

# In[166]:

ct\_region = pd.crosstab(X\_train\_join[features\_cate[3]], X\_train\_join.TARGET).transpose()

ct\_region.plot.bar(stacked=True, figsize=(7,5), fontsize=14, colormap='viridis', alpha=0.7)

plt.show()

# In[167]:

ct\_region = pd.crosstab(X\_train\_join[features\_cate[4]], X\_train\_join.TARGET).transpose()

ct\_region.plot.bar(stacked=True, figsize=(7,5), fontsize=14, colormap='viridis', alpha=0.7)

plt.show()

# In[169]:

ct\_region = pd.crosstab(X\_train\_join[features\_cate[5]], X\_train\_join.TARGET).transpose()

ct\_region.plot.bar(stacked=True, figsize=(7,5), fontsize=14, colormap='viridis', alpha=0.7)

plt.show()

# In[171]:

ct\_region = pd.crosstab(X\_train\_join[features\_cate[6]], X\_train\_join.TARGET).transpose()

ct\_region.plot.bar(stacked=True, figsize=(7,5), fontsize=14, colormap='viridis', alpha=0.7)

plt.show()

# In[172]:

ct\_region = pd.crosstab(X\_train\_join[features\_cate[7]], X\_train\_join.TARGET).transpose()

ct\_region.plot.bar(stacked=True, figsize=(7,5), fontsize=14, colormap='viridis', alpha=0.7)

plt.show()

# In[173]:

ct\_region = pd.crosstab(X\_train\_join[features\_cate[8]], X\_train\_join.TARGET).transpose()

ct\_region.plot.bar(stacked=True, figsize=(7,5), fontsize=14, colormap='viridis', alpha=0.7)

plt.show()

# ## FAMD

# FAMD on the original data and plot first 2 or 3 comps to see clusters

# In[ ]:

famd = FAMD(n\_components=40, n\_iter=5, copy=True, engine='auto', random\_state=4)

train\_trans = famd.fit\_transform(X\_train\_oh\_ne)

# In[ ]:

ax = famd.plot\_row\_coordinates(X\_train\_oh\_ne, ax=None, figsize=(6, 6), x\_component=0, y\_component=1, labels=None, color\_labels=y\_train\_sub, ellipse\_outline=False, ellipse\_fill=True, show\_points=True)

# In[25]:

var\_type = pd.read\_csv('Datasets/variable\_type.csv', index\_col='Row')

# In[26]:

var\_type.head()

**Logistic Regression**

#%%

# Import libraries

import pandas as pd

import numpy as np

from sklearn.model\_selection import train\_test\_split

from sklearn.model\_selection import GridSearchCV

from sklearn.linear\_model import LogisticRegression

from sklearn import ensemble

from sklearn.metrics import accuracy\_score, confusion\_matrix, classification\_report, f1\_score, roc\_auc\_score

from sklearn.preprocessing import StandardScaler

from imblearn.ensemble import BalancedBaggingClassifier

#%%

"""

ORIGINAL DATA

"""

# "or"--original

X\_train\_or\_df=pd.read\_csv('Original\_X\_train.csv')

X\_val\_or\_df=pd.read\_csv('Original\_X\_validate.csv')

X\_test\_or\_df=pd.read\_csv('Original\_X\_test.csv')

X\_train\_or=X\_train\_or\_df.iloc[:,1:]

X\_val\_or=X\_val\_or\_df.iloc[:,1:]

X\_test\_or=X\_test\_or\_df.iloc[:,1:]

Original\_y\_train=pd.read\_csv('Original\_y\_train.csv',header=None)

Original\_y\_val=pd.read\_csv('Original\_y\_validate.csv',header=None)

y\_train=Original\_y\_train.iloc[:,1]

y\_val=Original\_y\_val.iloc[:,1]

# count number of each class

class\_cnt=y\_train.value\_counts()

samples\_number=2\*class\_cnt[1]

# Normalization

scaler\_or = StandardScaler()

X\_train\_scaled\_or=scaler\_or.fit\_transform(X\_train\_or)

X\_val\_scaled\_or=scaler\_or.transform(X\_val\_or)

X\_test\_scaled\_or=scaler\_or.transform(X\_test\_or)

# define the parameter values that should be searched

# range of C searched:

bagging\_logit\_or = BalancedBaggingClassifier(LogisticRegression(random\_state=0),n\_estimators=50,

max\_samples=samples\_number, replacement=True, random\_state=0)

C\_range = {'base\_estimator\_\_C': np.power(10.0, np.arange(-3,3))}

# A dictionary is created

# key: parameter name

# value: list of values that should be searched for that parameter

# single key-value pair for param\_grid

# define grid, with 5-fold cross validation

logit\_grid\_or = GridSearchCV(estimator=bagging\_logit\_or, param\_grid=C\_range, cv=5)

# fit the grid with training data

logit\_grid\_or.fit(X\_train\_scaled\_or, y\_train)

# print the best estimator for logistic regression

print(logit\_grid\_or.best\_estimator\_)

# Predict and f1\_score

# loop for changing threshold

y\_val\_prob\_or=logit\_grid\_or.predict\_proba(X\_val\_scaled\_or)[:,1]

f1\_logit\_or=[]

threshold=[0.1,0.2,0.3,0.4,0.5]

for i in range(len(threshold)):

y\_val\_pred\_or=(y\_val\_prob\_or>=threshold[i]).astype(int)

f1\_or=f1\_score(y\_val, y\_val\_pred\_or)

f1\_logit\_or.append(f1\_or)

print("\n The threshold resulting the largest f1\_score on validation data is:")

print (threshold[f1\_logit\_or.index(max(f1\_logit\_or))])

print("\n The largest f1\_score on validation data is:")

print (round(max(f1\_logit\_or),3))

y\_val\_pred\_best\_or=(y\_val\_prob\_or>=threshold[f1\_logit\_or.index(max(f1\_logit\_or))]).astype(int)

#confusion matrix

confusion\_logit\_or=confusion\_matrix(y\_val, y\_val\_pred\_best\_or)

print(confusion\_logit\_or)

#performance score

performance\_logit\_or=classification\_report(y\_val, y\_val\_pred\_best\_or, digits = 3)

print(performance\_logit\_or)

# accuracy score

accuracy\_logit\_or=round(accuracy\_score(y\_val, y\_val\_pred\_best\_or),3)

print("\n Accuracy Score = ")

print (accuracy\_logit\_or)

auc\_logit\_or=round(roc\_auc\_score(y\_val, y\_val\_pred\_best\_or),3)

print("\n ROC\_AUC\_SCORE = ")

print(auc\_logit\_or)

#%%

"""

WITHOUT EXTRA FEATURES

"""

X\_train\_ne\_df=pd.read\_csv('ne\_X\_train.csv')

X\_val\_ne\_df=pd.read\_csv('ne\_X\_validate.csv')

X\_test\_ne\_df=pd.read\_csv('ne\_X\_test.csv')

X\_train\_ne=X\_train\_ne\_df.iloc[:,1:]

X\_val\_ne=X\_val\_ne\_df.iloc[:,1:]

X\_test\_ne=X\_test\_ne\_df.iloc[:,1:]

# Normalization

scaler\_ne = StandardScaler()

X\_train\_scaled\_ne=scaler\_ne.fit\_transform(X\_train\_ne)

X\_val\_scaled\_ne=scaler\_ne.transform(X\_val\_ne)

X\_test\_scaled\_ne=scaler\_or.transform(X\_test\_ne)

# define the parameter values that should be searched

bagging\_logit\_ne = BalancedBaggingClassifier(LogisticRegression(random\_state=0),n\_estimators=50,

max\_samples=samples\_number, replacement=True,random\_state=0)

# A dictionary is created

# key: parameter name

# value: list of values that should be searched for that parameter

# single key-value pair for param\_grid

# define grid, with 5-fold cross validation

logit\_grid\_ne = GridSearchCV(estimator=bagging\_logit\_ne, param\_grid=C\_range, cv=5)

# fit the grid with training data

logit\_grid\_ne.fit(X\_train\_scaled\_ne, y\_train)

# print the best estimator for logistic regression

print(logit\_grid\_ne.best\_estimator\_)

# Predict and f1\_score

# loop for changing threshold

y\_val\_prob\_ne=logit\_grid\_ne.predict\_proba(X\_val\_scaled\_ne)[:,1]

f1\_logit\_ne=[]

for i in range(len(threshold)):

y\_val\_pred\_ne=(y\_val\_prob\_ne>=threshold[i]).astype(int)

f1\_ne=f1\_score(y\_val, y\_val\_pred\_ne)

f1\_logit\_ne.append(f1\_ne)

print("\n The threshold resulting the largest f1\_score on validation data is:")

print (threshold[f1\_logit\_ne.index(max(f1\_logit\_ne))])

print("\n The largest f1\_score on validation data is:")

print (round(max(f1\_logit\_ne),3))

y\_val\_pred\_best\_ne=(y\_val\_prob\_ne>=threshold[f1\_logit\_ne.index(max(f1\_logit\_ne))]).astype(int)

#confusion matrix

confusion\_logit\_ne=confusion\_matrix(y\_val, y\_val\_pred\_best\_ne)

print(confusion\_logit\_ne)

#performance score

performance\_logit\_ne=classification\_report(y\_val, y\_val\_pred\_best\_ne, digits = 3)

print(performance\_logit\_ne)

# accuracy score

accuracy\_logit\_ne=round(accuracy\_score(y\_val, y\_val\_pred\_best\_ne),3)

print("\n Accuracy Score = ")

print (accuracy\_logit\_ne)

auc\_logit\_ne=round(roc\_auc\_score(y\_val, y\_val\_pred\_best\_ne),3)

print("\n ROC\_AUC\_SCORE = ")

print(auc\_logit\_ne)

#%%

"""

With EXTRA FEATURES

"""

X\_train\_we\_df=pd.read\_csv('we\_X\_train.csv')

X\_val\_we\_df=pd.read\_csv('we\_X\_validate.csv')

X\_test\_we\_df=pd.read\_csv('we\_X\_test.csv')

X\_train\_we=X\_train\_we\_df.iloc[:,1:]

X\_val\_we=X\_val\_we\_df.iloc[:,1:]

X\_test\_we=X\_test\_we\_df.iloc[:,1:]

# Normalization

scaler\_we = StandardScaler()

X\_train\_scaled\_we=scaler\_we.fit\_transform(X\_train\_we)

X\_val\_scaled\_we=scaler\_we.transform(X\_val\_we)

X\_test\_scaled\_we=scaler\_we.transform(X\_test\_we)

# define the parameter values that should be searched

bagging\_logit\_we = BalancedBaggingClassifier(LogisticRegression(random\_state=0),n\_estimators=50,

max\_samples=samples\_number, replacement=True,random\_state=0)

# A dictionary is created

# key: parameter name

# value: list of values that should be searched for that parameter

# single key-value pair for param\_grid

# define grid, with 5-fold cross validation

logit\_grid\_we = GridSearchCV(estimator=bagging\_logit\_we, param\_grid=C\_range, cv=5)

# fit the grid with training data

logit\_grid\_we.fit(X\_train\_scaled\_we, y\_train)

# print the best estimator for logistic regression

print(logit\_grid\_we.best\_estimator\_)

# Predict and f1\_score

# loop for changing threshold

y\_val\_prob\_we=logit\_grid\_we.predict\_proba(X\_val\_scaled\_we)[:,1]

f1\_logit\_we=[]

for i in range(len(threshold)):

y\_val\_pred\_we=(y\_val\_prob\_we>=threshold[i]).astype(int)

f1\_we=f1\_score(y\_val, y\_val\_pred\_we)

f1\_logit\_we.append(f1\_we)

print("\n The threshold resulting the largest f1\_score on validation data is:")

print (threshold[f1\_logit\_we.index(max(f1\_logit\_we))])

print("\n The largest f1\_score on validation data is:")

print (round(max(f1\_logit\_we),3))

y\_val\_pred\_best\_we=(y\_val\_prob\_we>=threshold[f1\_logit\_we.index(max(f1\_logit\_we))]).astype(int)

#confusion matrix

confusion\_logit\_we=confusion\_matrix(y\_val, y\_val\_pred\_best\_we)

print(confusion\_logit\_we)

#performance score

performance\_logit\_we=classification\_report(y\_val, y\_val\_pred\_best\_we, digits = 3)

print(performance\_logit\_we)

# accuracy score

accuracy\_logit\_we=round(accuracy\_score(y\_val, y\_val\_pred\_best\_we),3)

print("\n Accuracy Score = ")

print (accuracy\_logit\_we)

auc\_logit\_we=round(roc\_auc\_score(y\_val, y\_val\_pred\_best\_we),3)

print("\n ROC\_AUC\_SCORE = ")

print(auc\_logit\_we)

**KNN**

#%%

# Import libraries

import pandas as pd

import numpy as np

from sklearn.model\_selection import train\_test\_split

from sklearn.model\_selection import GridSearchCV

from sklearn.model\_selection import RandomizedSearchCV

from sklearn.neighbors import KNeighborsClassifier

from sklearn import ensemble

from sklearn.metrics import accuracy\_score, confusion\_matrix, classification\_report, f1\_score,roc\_auc\_score

from sklearn.preprocessing import MinMaxScaler

from imblearn.ensemble import BalancedBaggingClassifier

"""

Original Features

"""

# "or"--original

X\_train\_or\_df=pd.read\_csv('Original\_X\_train.csv')

X\_val\_or\_df=pd.read\_csv('Original\_X\_validate.csv')

X\_test\_or\_df=pd.read\_csv('Original\_X\_test.csv')

X\_train\_or=X\_train\_or\_df.iloc[:,1:]

X\_val\_or=X\_val\_or\_df.iloc[:,1:]

X\_test\_or=X\_test\_or\_df.iloc[:,1:]

Original\_y\_train=pd.read\_csv('Original\_y\_train.csv',header=None)

Original\_y\_val=pd.read\_csv('Original\_y\_validate.csv',header=None)

y\_train=Original\_y\_train.iloc[:,1]

y\_val=Original\_y\_val.iloc[:,1]

# count number of each class

class\_cnt=y\_train.value\_counts()

samples\_number=2\*class\_cnt[1]

# Normalization

mms\_or = MinMaxScaler()

X\_train\_mms\_or=mms\_or.fit\_transform(X\_train\_or)

X\_val\_mms\_or=mms\_or.transform(X\_val\_or)

X\_test\_mms\_or=mms\_or.transform(X\_test\_or)

# define the parameter values that should be searched

# range of n\_neighbors searched:

bagging\_KNN\_or = BalancedBaggingClassifier(KNeighborsClassifier(), n\_estimators=9,

max\_samples=samples\_number, replacement=True, random\_state=0)

K\_range\_or = {'base\_estimator\_\_n\_neighbors': list(range(3,23,2))}

# A dictionary is created

# key: parameter name

# value: list of values that should be searched for that parameter

# single key-value pair for param\_grid

# define grid, with 5-fold cross validation

KNN\_grid\_or = RandomizedSearchCV(estimator=bagging\_KNN\_or, param\_distributions=K\_range\_or,n\_iter=5, cv=5)

# fit the grid with training data

KNN\_grid\_or.fit(X\_train\_mms\_or, y\_train)

# print the best estimator for logistic regression

print(KNN\_grid\_or.best\_estimator\_)

y\_pred\_knn\_or=KNN\_grid\_or.predict(X\_val\_mms\_or)

#confusion matrix

confusion\_knn\_or=confusion\_matrix(y\_val, y\_pred\_knn\_or)

print(confusion\_knn\_or)

#performance score

performance\_knn\_or=classification\_report(y\_val, y\_pred\_knn\_or, digits = 3)

print(performance\_knn\_or)

# accuracy score

accuracy\_knn\_or=round(accuracy\_score(y\_val, y\_pred\_knn\_or),3)

print("\n Accuracy Score = ")

print (accuracy\_knn\_or)

auc\_knn\_or=round(roc\_auc\_score(y\_val, y\_pred\_knn\_or),3)

print("\n ROC\_AUC\_SCORE = ")

print(auc\_knn\_or)

#%%

"""

WITHOUT EXTRA FEATURES

"""

X\_train\_ne\_df=pd.read\_csv('ne\_X\_train.csv')

X\_val\_ne\_df=pd.read\_csv('ne\_X\_validate.csv')

X\_test\_ne\_df=pd.read\_csv('ne\_X\_test.csv')

X\_train\_ne=X\_train\_ne\_df.iloc[:,1:]

X\_val\_ne=X\_val\_ne\_df.iloc[:,1:]

X\_test\_ne=X\_test\_ne\_df.iloc[:,1:]

# Normalization

mms\_ne = MinMaxScaler()

X\_train\_mms\_ne=mms\_ne.fit\_transform(X\_train\_ne)

X\_val\_mms\_ne=mms\_ne.transform(X\_val\_ne)

X\_test\_mms\_ne=mms\_ne.transform(X\_test\_ne)

bagging\_KNN\_ne = BalancedBaggingClassifier(KNeighborsClassifier(), n\_estimators=9,

max\_samples=samples\_number, replacement=True, random\_state=0)

K\_range = {'base\_estimator\_\_n\_neighbors': list(range(3,33,2))}

# A dictionary is created

# key: parameter name

# value: list of values that should be searched for that parameter

# single key-value pair for param\_grid

# define grid, with 5-fold cross validation

KNN\_grid\_ne = RandomizedSearchCV(estimator=bagging\_KNN\_ne, param\_distributions=K\_range, cv=5)

# fit the grid with training data

KNN\_grid\_ne.fit(X\_train\_mms\_ne, y\_train)

# print the best estimator for logistic regression

print(KNN\_grid\_ne.best\_estimator\_)

y\_pred\_knn\_ne=KNN\_grid\_ne.predict(X\_val\_mms\_ne)

#confusion matrix

confusion\_knn\_ne=confusion\_matrix(y\_val, y\_pred\_knn\_ne)

print(confusion\_knn\_ne)

#performance score

performance\_knn\_ne=classification\_report(y\_val, y\_pred\_knn\_ne, digits = 3)

print(performance\_knn\_ne)

# accuracy score

accuracy\_knn\_ne=round(accuracy\_score(y\_val, y\_pred\_knn\_ne),3)

print("\n Accuracy Score = ")

print (accuracy\_knn\_ne)

auc\_knn\_ne=round(roc\_auc\_score(y\_val, y\_pred\_knn\_ne),3)

print("\n ROC\_AUC\_SCORE = ")

print(auc\_knn\_ne)

#%%

"""

WITH EXTRA FEATURES

"""

X\_train\_we\_df=pd.read\_csv('we\_X\_train.csv')

X\_val\_we\_df=pd.read\_csv('we\_X\_validate.csv')

X\_test\_we\_df=pd.read\_csv('we\_X\_test.csv')

X\_train\_we=X\_train\_we\_df.iloc[:,1:]

X\_val\_we=X\_val\_we\_df.iloc[:,1:]

X\_test\_we=X\_test\_we\_df.iloc[:,1:]

# Normalization

mms\_we = MinMaxScaler()

X\_train\_mms\_we=mms\_we.fit\_transform(X\_train\_we)

X\_val\_mms\_we=mms\_we.transform(X\_val\_we)

X\_test\_mms\_we=mms\_we.transform(X\_test\_we)

bagging\_KNN\_we = BalancedBaggingClassifier(KNeighborsClassifier(), n\_estimators=9,

max\_samples=samples\_number, replacement=True, random\_state=0)

# A dictionary is created

# key: parameter name

# value: list of values that should be searched for that parameter

# single key-value pair for param\_grid

# define grid, with 5-fold cross validation

KNN\_grid\_we = RandomizedSearchCV(estimator=bagging\_KNN\_we, param\_distributions=K\_range, cv=5)

# fit the grid with training data

KNN\_grid\_we.fit(X\_train\_mms\_we, y\_train)

# print the best estimator for logistic regression

print(KNN\_grid\_we.best\_estimator\_)

y\_pred\_knn\_we=KNN\_grid\_we.predict(X\_val\_mms\_we)

#confusion matrix

confusion\_knn\_we=confusion\_matrix(y\_val, y\_pred\_knn\_we)

print(confusion\_knn\_we)

#performance score

performance\_knn\_we=classification\_report(y\_val, y\_pred\_knn\_we, digits = 3)

print(performance\_knn\_we)

# accuracy score

accuracy\_knn\_we=round(accuracy\_score(y\_val, y\_pred\_knn\_we),3)

print("\n Accuracy Score = ")

print (accuracy\_knn\_we)

auc\_knn\_we=round(roc\_auc\_score(y\_val, y\_pred\_knn\_we),3)

print("\n ROC\_AUC\_SCORE = ")

print(auc\_knn\_we)

"""

#%% Prediction

X\_train\_val\_we=pd.concat([X\_train\_we,X\_val\_we],axis=0)

y\_train\_val=pd.concat([y\_train,y\_val],axis=0)

mms\_train\_val\_we = MinMaxScaler()

X\_train\_val\_mms\_we=mms\_train\_val\_we.fit\_transform(X\_train\_val\_we)

X\_test\_mms\_we=mms\_train\_val\_we.transform(X\_test\_we)

# count number of each class

class\_cnt\_train\_val=y\_train\_val.value\_counts()

samples\_number\_train\_val=2\*class\_cnt\_train\_val[1]

#%%

knn\_we= BalancedBaggingClassifier(KNeighborsClassifier(), n\_estimators=5,

max\_samples=samples\_number\_train\_val, replacement=True, random\_state=0)

KNN\_search\_we = RandomizedSearchCV(estimator=knn\_we, param\_distributions=K\_range\_or, n\_iter=5, cv=5)

KNN\_search\_we.fit(X\_train\_val\_mms\_we, y\_train\_val)

print(KNN\_search\_we.best\_estimator\_)

#%%

y\_pred\_knn\_we\_test=KNN\_search\_we.predict(X\_test\_mms\_we)

test\_Bureau=pd.read\_csv('ProjectTest\_Bureau.csv')

Predict\_index=test\_Bureau['Index\_ID']

Predict\_values=pd.Series(y\_pred\_knn\_we\_test,index=['TARGET'])

Predict\_test=pd.concat([Predict\_index,Predict\_values],axis=1)

Predict\_test.to\_csv('Group26\_Results\_Bureau.csv')

"""

**XGBOOST**

**Original**

import numpy as np

import pandas as pd

import xgboost as xgb

from xgboost.sklearn import XGBClassifier

from sklearn import cross\_validation, metrics

from sklearn.grid\_search import GridSearchCV

import matplotlib.pylab as plt

get\_ipython().run\_line\_magic('matplotlib', 'inline')

from matplotlib.pylab import rcParams

rcParams['figure.figsize'] = 12, 4

# ### Import original dataset

# In[7]:

og\_X\_train = pd.read\_csv('Original\_X\_train.csv')

og\_X\_test = pd.read\_csv('Original\_X\_test.csv')

og\_X\_validate = pd.read\_csv('Original\_X\_validate.csv')

og\_y\_train = pd.read\_csv('Original\_y\_train.csv', header = None)

og\_y\_validate = pd.read\_csv('Original\_y\_validate.csv', header = None)

og\_X\_train.head(5)

# In[32]:

predictors = og\_X\_train.drop('SK\_ID\_CURR', axis = 1)

X\_val = og\_X\_validate.drop('SK\_ID\_CURR', axis = 1)

y\_val = og\_y\_validate.iloc[:,1]

target = og\_y\_train.iloc[:,1]

ratio = float(np.sum(target.values == 0)) / np.sum(target.values == 1)

# ### Tune max\_depth and min\_child\_weight

# In[8]:

param\_test1 = {

'max\_depth':np.arange(3,10,2),

'min\_child\_weight':np.arange(1,6,2),

}

gsearch1 = GridSearchCV(estimator = XGBClassifier( learning\_rate =0.1, n\_estimators=150, max\_depth=5,

min\_child\_weight=1, gamma=0, subsample=0.8, colsample\_bytree=0.8,

objective= 'binary:logistic', nthread=4, scale\_pos\_weight=ratio, seed=30),

param\_grid = param\_test1, scoring='roc\_auc',n\_jobs=4,iid=False, cv=5)

gsearch1.fit(predictors,target)

gsearch1.grid\_scores\_, gsearch1.best\_params\_, gsearch1.best\_score\_

# ### Tune gamma

# In[10]:

param\_test2 = {

'gamma':[i/10.0 for i in range(0,5)]

}

gsearch2 = GridSearchCV(estimator = XGBClassifier( learning\_rate =0.1, n\_estimators=150, max\_depth=3,

min\_child\_weight=3, gamma=0, subsample=0.8, colsample\_bytree=0.8,

objective= 'binary:logistic', nthread=4, scale\_pos\_weight=ratio,seed=30),

param\_grid = param\_test2, scoring='roc\_auc',n\_jobs=4,iid=False, cv=5)

gsearch2.fit(predictors,target)

gsearch2.grid\_scores\_, gsearch2.best\_params\_, gsearch2.best\_score\_

# ### Tune regularization parameter

# In[11]:

param\_test3 = {

'reg\_lambda':[1e-5, 1e-2, 0.1, 1, 100]

}

gsearch3 = GridSearchCV(estimator = XGBClassifier( learning\_rate =0.1, n\_estimators=150, max\_depth=3,

min\_child\_weight=3, gamma=0.4, subsample=0.8, colsample\_bytree=0.8,

objective= 'binary:logistic', nthread=4, scale\_pos\_weight=ratio,seed=30),

param\_grid = param\_test3, scoring='roc\_auc',n\_jobs=4,iid=False, cv=5)

gsearch3.fit(predictors,target)

gsearch3.grid\_scores\_, gsearch3.best\_params\_, gsearch3.best\_score\_

# ### Fit final model

# In[17]:

xgb = XGBClassifier(learning\_rate =0.1, n\_estimators=150, max\_depth=3, min\_child\_weight=3, gamma=0.4,

subsample=0.8, colsample\_bytree=0.8, objective= 'binary:logistic', nthread=4,

reg\_lamda=1, scale\_pos\_weight=ratio, seed=30)

xgb.fit(predictors,target, eval\_metric = 'auc')

# ### Prediction

# In[18]:

#Predict validation set:

dval\_predictions = xgb.predict(X\_val)

dval\_predprob = xgb.predict\_proba(X\_val)[:,1]

#Print model report:

print ("Accuracy: %.4g" % metrics.accuracy\_score(y\_val, dval\_predictions))

print ("AUC Score: %f" % metrics.roc\_auc\_score(y\_val, dval\_predprob))

print("F1-score: %.4g" % metrics.f1\_score(y\_val, dval\_predictions))

# In[44]:

#Predict test set:

X\_test = og\_X\_test.drop('Index\_ID', axis = 1)

dtest\_predictions = xgb.predict(X\_test)

final\_predictions = pd.DataFrame(dtest\_predictions, columns = ['TARGET'])

final\_predictions.head(5)

# In[46]:

df\_test\_bureau = pd.read\_csv('ProjectTest\_Bureau.csv')

df\_test\_id = pd.DataFrame(df\_test\_bureau['Index\_ID'])

df\_test\_id.head()

# In[52]:

df = pd.concat([df\_test\_id, final\_predictions], axis=1)

df.to\_csv('Group26\_Results\_Bureau.csv')

# In[53]:

df.head()

**XGBOOST**

**With extra features:**

import numpy as np

import pandas as pd

import xgboost as xgb

from xgboost.sklearn import XGBClassifier

from sklearn import cross\_validation, metrics

from sklearn.grid\_search import GridSearchCV

import matplotlib.pylab as plt

get\_ipython().run\_line\_magic('matplotlib', 'inline')

from matplotlib.pylab import rcParams

rcParams['figure.figsize'] = 12, 4

# ### Import dataset with extra features

# In[52]:

we\_X\_train = pd.read\_csv('we\_X\_train.csv')

we\_X\_test = pd.read\_csv('we\_X\_test.csv')

we\_X\_validate = pd.read\_csv('we\_X\_validate.csv')

we\_y\_train = pd.read\_csv('we\_y\_train.csv', header = None)

we\_y\_validate = pd.read\_csv('we\_y\_validate.csv', header = None)

we\_X\_train.head(5)

# In[75]:

predictors = we\_X\_train.drop('SK\_ID\_CURR', axis = 1)

X\_val = we\_X\_validate.drop('SK\_ID\_CURR', axis = 1)

y\_val = we\_y\_validate.iloc[:,1]

target = we\_y\_train.iloc[:,1]

ratio = float(np.sum(target.values == 0)) / np.sum(target.values == 1)

# ### Tune max\_depth and min\_child\_weight

# In[55]:

param\_test1 = {

'max\_depth':np.arange(3,10,2),

'min\_child\_weight':np.arange(1,6,2),

}

gsearch1 = GridSearchCV(estimator = XGBClassifier( learning\_rate =0.1, n\_estimators=150, max\_depth=5,

min\_child\_weight=1, gamma=0, subsample=0.8, colsample\_bytree=0.8,

objective= 'binary:logistic', nthread=4, scale\_pos\_weight=ratio, seed=30),

param\_grid = param\_test1, scoring='roc\_auc',n\_jobs=4,iid=False, cv=5)

gsearch1.fit(predictors,target)

gsearch1.grid\_scores\_, gsearch1.best\_params\_, gsearch1.best\_score\_

# ### Tune gamma

# In[58]:

param\_test2 = {

'gamma':[i/10.0 for i in range(0,5)]

}

gsearch2 = GridSearchCV(estimator = XGBClassifier( learning\_rate =0.1, n\_estimators=150, max\_depth=3,

min\_child\_weight=5, gamma=0, subsample=0.8, colsample\_bytree=0.8,

objective= 'binary:logistic', nthread=4, scale\_pos\_weight=ratio,seed=30),

param\_grid = param\_test2, scoring='roc\_auc',n\_jobs=4,iid=False, cv=5)

gsearch2.fit(predictors,target)

gsearch2.grid\_scores\_, gsearch2.best\_params\_, gsearch2.best\_score\_

# ### Tune regularization parameter

# In[59]:

param\_test3 = {

'reg\_lambda':[1e-5, 1e-2, 0.1, 1, 100]

}

gsearch3 = GridSearchCV(estimator = XGBClassifier( learning\_rate =0.1, n\_estimators=150, max\_depth=3,

min\_child\_weight=5, gamma=0.4, subsample=0.8, colsample\_bytree=0.8,

objective= 'binary:logistic', nthread=4, scale\_pos\_weight=ratio,seed=30),

param\_grid = param\_test3, scoring='roc\_auc',n\_jobs=4,iid=False, cv=5)

gsearch3.fit(predictors,target)

gsearch3.grid\_scores\_, gsearch3.best\_params\_, gsearch3.best\_score\_

# ### Fit final model

# In[76]:

xgb = XGBClassifier(learning\_rate =0.1, n\_estimators=150, max\_depth=3, min\_child\_weight=5, gamma=0.4,

subsample=0.8, colsample\_bytree=0.8, objective= 'binary:logistic', nthread=4,

reg\_lamda=1, scale\_pos\_weight=ratio, seed=30)

xgb.fit(predictors,target, eval\_metric = 'auc')

# ### Prediction

# In[77]:

#Predict validation set:

dval\_predictions = xgb.predict(X\_val)

dval\_predprob = xgb.predict\_proba(X\_val)[:,1]

#Print model report:

print ("Accuracy: %.4g" % metrics.accuracy\_score(y\_val, dval\_predictions))

print ("AUC Score: %f" % metrics.roc\_auc\_score(y\_val, dval\_predprob))

print("F1-score: %.4g" % metrics.f1\_score(y\_val, dval\_predictions))

**XGBOOST**

**Without Extra Features**

import numpy as np

import pandas as pd

import xgboost as xgb

from xgboost.sklearn import XGBClassifier

from sklearn import cross\_validation, metrics

from sklearn.grid\_search import GridSearchCV

import matplotlib.pylab as plt

get\_ipython().run\_line\_magic('matplotlib', 'inline')

from matplotlib.pylab import rcParams

rcParams['figure.figsize'] = 12, 4

# ### Import dataset without extra features

# In[2]:

ne\_X\_train = pd.read\_csv('ne\_X\_train.csv')

ne\_X\_test = pd.read\_csv('ne\_X\_test.csv')

ne\_X\_validate = pd.read\_csv('ne\_X\_validate.csv')

og\_y\_train = pd.read\_csv('Original\_y\_train.csv', header = None)

og\_y\_validate = pd.read\_csv('Original\_y\_validate.csv', header = None)

og\_y\_train.head(5)

# In[9]:

predictors = ne\_X\_train.drop('SK\_ID\_CURR', axis = 1)

X\_val = ne\_X\_validate.drop('SK\_ID\_CURR', axis = 1)

y\_val = og\_y\_validate.iloc[:,1]

target = og\_y\_train.iloc[:,1]

ratio = float(np.sum(target.values == 0)) / np.sum(target.values == 1)

# ### Tune max\_depth and min\_child\_weight

# In[4]:

param\_test1 = {

'max\_depth':np.arange(3,10,2),

'min\_child\_weight':np.arange(1,6,2),

}

gsearch1 = GridSearchCV(estimator = XGBClassifier( learning\_rate =0.1, n\_estimators=150, max\_depth=5,

min\_child\_weight=1, gamma=0, subsample=0.8, colsample\_bytree=0.8,

objective= 'binary:logistic', nthread=4, scale\_pos\_weight=ratio, seed=30),

param\_grid = param\_test1, scoring='roc\_auc',n\_jobs=4,iid=False, cv=5)

gsearch1.fit(predictors,target)

gsearch1.grid\_scores\_, gsearch1.best\_params\_, gsearch1.best\_score\_

# ### Tune gamma

# In[5]:

param\_test2 = {

'gamma':[i/10.0 for i in range(0,5)]

}

gsearch2 = GridSearchCV(estimator = XGBClassifier( learning\_rate =0.1, n\_estimators=150, max\_depth=3,

min\_child\_weight=1, gamma=0, subsample=0.8, colsample\_bytree=0.8,

objective= 'binary:logistic', nthread=4, scale\_pos\_weight=ratio,seed=30),

param\_grid = param\_test2, scoring='roc\_auc',n\_jobs=4,iid=False, cv=5)

gsearch2.fit(predictors,target)

gsearch2.grid\_scores\_, gsearch2.best\_params\_, gsearch2.best\_score\_

# ### Tune regularization parameter

# In[6]:

param\_test3 = {

'reg\_lambda':[1e-5, 1e-2, 0.1, 1, 100]

}

gsearch3 = GridSearchCV(estimator = XGBClassifier( learning\_rate =0.1, n\_estimators=150, max\_depth=3,

min\_child\_weight=1, gamma=0.1, subsample=0.8, colsample\_bytree=0.8,

objective= 'binary:logistic', nthread=4, scale\_pos\_weight=ratio,seed=30),

param\_grid = param\_test3, scoring='roc\_auc',n\_jobs=4,iid=False, cv=5)

gsearch3.fit(predictors,target)

gsearch3.grid\_scores\_, gsearch3.best\_params\_, gsearch3.best\_score\_

# ### Fit final model

# In[10]:

xgb = XGBClassifier(learning\_rate =0.1, n\_estimators=150, max\_depth=3, min\_child\_weight=1, gamma=0.1,

subsample=0.8, colsample\_bytree=0.8, objective= 'binary:logistic', nthread=4,

reg\_lamda=1, scale\_pos\_weight=ratio, seed=30)

xgb.fit(predictors,target, eval\_metric = 'auc')

# ### Prediction

# In[11]:

#Predict validation set:

dval\_predictions = xgb.predict(X\_val)

dval\_predprob = xgb.predict\_proba(X\_val)[:,1]

#Print model report:

print ("Accuracy: %.4g" % metrics.accuracy\_score(y\_val, dval\_predictions))

print ("AUC Score: %f" % metrics.roc\_auc\_score(y\_val, dval\_predprob))

print("F1-score: %.4g" % metrics.f1\_score(y\_val, dval\_predictions))

## Appendix – Meeting Agenda

**TEAM TASK MEETING AGENDA**

|  |
| --- |
| **TEAM MEETING AGENDA**  **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Project Group 26\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**  Meeting to be held \_\_\_\_\_\_\_\_\_Law Library Group Study Room M108\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  \_\_\_\_\_\_\_\_\_\_\_\_Oct 18, 2018\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ (Date)  \_\_\_\_\_\_\_\_\_\_\_\_\_14.00pm \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_(Time)  Chairperson: \_\_\_\_\_\_\_\_Di Xu\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  Minute-Taker: \_\_\_\_\_\_\_ Anh Mai Phuong\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |
| 1. Attendance: All attended 2. Discussion Point and Actions Decided   (1) Go through the requirements of the assignment  (2) Discuss the expectation from the professor  (3) Set up a plan for completing each part  (4) Allocate tasks to each individual  (5) Discuss any remaining issues  (6) Determine the next meeting time and date   1. Any other business – No 2. Next meeting – Oct 21, 2018 |

**TEAM TASK MEETING AGENDA**

|  |
| --- |
| **TEAM MEETING AGENDA**  **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Project Group 26\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**  Meeting to be held \_\_\_\_\_\_\_\_\_Law Library Group Study Room M104\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  \_\_\_\_\_\_\_\_\_\_\_\_Oct 21, 2018\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ (Date)  \_\_\_\_\_\_\_\_\_\_\_\_\_14.00pm \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_(Time)  Chairperson: \_\_\_\_\_\_\_\_\_Manjing Fang \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  Minute-Taker: \_\_\_\_\_\_\_\_\_ Anh Mai Phuong\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |
| 1. Attendance: All attended 2. Discussion Point   (1) Everyone presents his / her work  (2) Discuss ways of selecting features  (4) Update new thoughts of the assignment requirements  (5) Update our plan and work allocations  (6) Determine the next meeting time and date   1. Any other business – No 2. Next meeting – Oct 24, 2018 |

**TEAM TASK MEETING AGENDA**

|  |
| --- |
| **TEAM MEETING AGENDA**  **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Project Group 26\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**  Meeting to be held \_\_\_\_\_\_\_\_\_Law Library Group Study Room M108\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  \_\_\_\_\_\_\_\_\_\_\_\_Oct 24, 2018\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ (Date)  \_\_\_\_\_\_\_\_\_\_\_\_\_14.00pm \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_(Time)  Chairperson: \_\_\_\_\_\_\_\_\_ Danlu Liang \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  Minute-Taker: \_\_\_\_\_\_\_\_\_Chen Chen\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |
| 1. Attendance: All attended 2. Discussion Point   (1) Everyone presents his / her work  (2) Discussion about model selection  (3) Decide on specific issues (how to set parameters, etc.)  (4) Determine the next meeting time and date   1. Any other business – No 2. Next meeting – Oct 27, 2018 |

## Appendix – Meeting Template

**MINUTES TEMPLATE**

**Minutes of meeting for** \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Project Group 26\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ (**Company Name**)

Date: \_\_\_\_\_\_\_\_\_Oct 18,2018\_\_\_\_\_\_\_\_\_\_\_ Time: \_\_\_\_\_\_\_\_\_\_14.00pm\_\_\_\_\_\_ Location: Law Library Group Study Room M108\_\_\_\_\_\_\_\_\_\_\_

Chairperson: \_\_\_\_\_\_\_\_\_Di Xu\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Minute-Taker: \_\_\_\_\_\_\_\_\_\_ Anh Mai Phuong \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Agenda Item | Key Points | Action | By Whom | When | Communication Strategy |
| 1. Understand Requirement  2. Task Allocation | Go through the requirements of the assignment  \* Discuss the expectation from the professor  \* Set up a plan for completing each part  \* Allocate tasks to each individual |  | All team members | 14.00—16.00 |  |

Souce: TAFE Access Division “Communication for Business”, 2000

**MINUTES TEMPLATE**

**Minutes of meeting for** \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Project Group 26\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ (**Company Name**)

Date: \_\_\_\_\_\_\_\_\_Oct 21,2018\_\_\_\_\_\_\_\_\_\_\_ Time: \_\_\_\_\_\_\_\_\_\_14.00pm\_\_\_\_\_\_ Location: Law Library Group Study Room M104\_\_\_\_\_\_\_\_\_\_\_

Chairperson: \_\_\_\_\_\_\_\_\_Manjing Fang\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Minute-Taker: \_\_\_\_\_\_\_\_\_\_ Anh Mai Phuong \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Agenda Item | Key Points | Action | By Whom | When | Communication Strategy |
| 1. Present own works  2. Update information | \* Everyone presents his / her work  \* Discuss ways of selecting features  \* Update new thoughts of the assignment requirements  Update our plan and work allocations |  | All team members | 14.00—16.00 |  |

Souce: TAFE Access Division “Communication for Business”, 2000

**MINUTES TEMPLATE**

**Minutes of meeting for** \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Project Group 26\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ (**Company Name**)

Date: \_\_\_\_\_\_\_\_\_Oct 24,2018\_\_\_\_\_\_\_\_\_\_\_ Time: \_\_\_\_\_\_\_\_\_\_14.00pm\_\_\_\_\_\_ Location: Law Library Group Study Room M108\_\_\_\_\_\_\_\_\_\_\_

Chairperson: \_\_\_\_\_\_\_\_\_Danlu Liang\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Minute-Taker: \_\_\_\_\_\_\_\_\_\_Chen Chen\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Agenda Item | Key Points | Action | By Whom | When | Communication Strategy |
| 1. Present own works  2. Discuss specific issues | \* Everyone presents his / her work  \* Discussion about model selection  \* Decide on specific issues (how to set parameters, etc.) |  | All team members | 14.00—16.00 |  |

Souce: TAFE Access Division “Communication for Business”, 2000

**Appendix 1: Ascending Sorted Features for Without- extra Feature Set Selected By Fold Change**

|  |  |  |
| --- | --- | --- |
| **Column Name** | **Level** | **Missing Value** |
| FLAG\_DOCUMENT\_4 | 2.000 | 0.000 |
| FLAG\_DOCUMENT\_10 | 2.000 | 0.000 |
| FLAG\_DOCUMENT\_12 | 2.000 | 0.000 |
| FLAG\_DOCUMENT\_2 | 2.000 | 0.000 |
| FLAG\_DOCUMENT\_13 | 2.000 | 0.000 |
| FLAG\_DOCUMENT\_17 | 2.000 | 0.000 |
| FLAG\_DOCUMENT\_15 | 2.000 | 0.000 |
| FLAG\_DOCUMENT\_14 | 2.000 | 0.000 |
| FLAG\_DOCUMENT\_16 | 2.000 | 0.000 |
| FLAG\_DOCUMENT\_21 | 2.000 | 0.000 |
| REG\_CITY\_NOT\_LIVE\_CITY | 2.000 | 0.000 |
| DAYS\_EMPLOYED | 12310.000 | 0.000 |
| NAME\_CONTRACT\_TYPE | 2.000 | 0.000 |
| FLAG\_DOCUMENT\_6 | 2.000 | 0.000 |
| EMERGENCYSTATE\_MODE | 2.000 | 47.337 |
| FLAG\_DOCUMENT\_7 | 2.000 | 0.000 |
| DEF\_60\_CNT\_SOCIAL\_CIRCLE | 9.000 | 0.333 |
| FLAG\_DOCUMENT\_18 | 2.000 | 0.000 |
| FLAG\_DOCUMENT\_9 | 2.000 | 0.000 |
| DEF\_30\_CNT\_SOCIAL\_CIRCLE | 10.000 | 0.333 |
| REG\_CITY\_NOT\_WORK\_CITY | 2.000 | 0.000 |
| EXT\_SOURCE\_3 | 809.000 | 19.869 |
| EXT\_SOURCE\_1 | 103908.000 | 56.376 |
| ELEVATORS\_AVG | 254.000 | 53.220 |
| ELEVATORS\_MEDI | 46.000 | 53.220 |
| ELEVATORS\_MODE | 26.000 | 53.220 |
| CODE\_GENDER | 3.000 | 0.000 |
| HOUSETYPE\_MODE | 3.000 | 50.113 |
| EXT\_SOURCE\_2 | 114271.000 | 0.216 |
| LIVE\_CITY\_NOT\_WORK\_CITY | 2.000 | 0.000 |
| FLAG\_DOCUMENT\_20 | 2.000 | 0.000 |
| FLAG\_DOCUMENT\_11 | 2.000 | 0.000 |
| FLAG\_WORK\_PHONE | 2.000 | 0.000 |
| DAYS\_LAST\_PHONE\_CHANGE | 3756.000 | 0.000 |
| AMT\_REQ\_CREDIT\_BUREAU\_DAY | 9.000 | 13.529 |
| AMT\_REQ\_CREDIT\_BUREAU\_MON | 24.000 | 13.529 |
| REG\_REGION\_NOT\_LIVE\_REGION | 2.000 | 0.000 |
| FLAG\_PHONE | 2.000 | 0.000 |
| LIVINGAREA\_AVG | 5141.000 | 50.142 |
| LIVINGAREA\_MEDI | 5212.000 | 50.142 |
| TOTALAREA\_MODE | 5056.000 | 48.204 |
| LIVINGAREA\_MODE | 5229.000 | 50.142 |
| OWN\_CAR\_AGE | 60.000 | 65.985 |
| COMMONAREA\_AVG | 3116.000 | 69.832 |
| COMMONAREA\_MEDI | 3139.000 | 69.832 |
| NONLIVINGAREA\_AVG | 3218.000 | 55.119 |
| NONLIVINGAREA\_MEDI | 3249.000 | 55.119 |
| NONLIVINGAREA\_MODE | 3261.000 | 55.119 |
| FLAG\_OWN\_CAR | 2.000 | 0.000 |
| FLOORSMAX\_AVG | 396.000 | 49.698 |
| FLOORSMAX\_MEDI | 49.000 | 49.698 |
| FLOORSMAX\_MODE | 25.000 | 49.698 |
| CNT\_CHILDREN | 15.000 | 0.000 |
| DAYS\_REGISTRATION | 15571.000 | 0.000 |
| COMMONAREA\_MODE | 3062.000 | 69.832 |
| REG\_REGION\_NOT\_WORK\_REGION | 2.000 | 0.000 |
| APARTMENTS\_AVG | 2307.000 | 50.677 |
| APARTMENTS\_MEDI | 1139.000 | 50.677 |
| APARTMENTS\_MODE | 757.000 | 50.677 |
| AMT\_GOODS\_PRICE | 950.000 | 0.091 |
| FLAG\_DOCUMENT\_8 | 2.000 | 0.000 |
| FLOORSMIN\_AVG | 300.000 | 67.798 |
| FLOORSMIN\_MEDI | 47.000 | 67.798 |
| FLOORSMIN\_MODE | 25.000 | 67.798 |
| FLAG\_DOCUMENT\_3 | 2.000 | 0.000 |
| DAYS\_ID\_PUBLISH | 6158.000 | 0.000 |
| LIVINGAPARTMENTS\_AVG | 1843.000 | 68.312 |
| REGION\_POPULATION\_RELATIVE | 81.000 | 0.000 |
| LIVINGAPARTMENTS\_MEDI | 1089.000 | 68.312 |
| NAME\_INCOME\_TYPE | 8.000 | 0.000 |
| LIVINGAPARTMENTS\_MODE | 734.000 | 68.312 |
| NAME\_HOUSING\_TYPE | 6.000 | 0.000 |
| BASEMENTAREA\_AVG | 3741.000 | 58.469 |
| BASEMENTAREA\_MEDI | 3734.000 | 58.469 |
| ORGANIZATION\_TYPE | 58.000 | 0.000 |
| DAYS\_BIRTH | 17444.000 | 0.000 |
| NAME\_EDUCATION\_TYPE | 5.000 | 0.000 |
| AMT\_CREDIT | 5451.000 | 0.000 |
| FLAG\_EMP\_PHONE | 2.000 | 0.000 |
| BASEMENTAREA\_MODE | 3800.000 | 58.469 |
| AMT\_REQ\_CREDIT\_BUREAU\_YEAR | 25.000 | 13.529 |
| FLAG\_DOCUMENT\_19 | 2.000 | 0.000 |
| LANDAREA\_MEDI | 3507.000 | 59.323 |
| LIVE\_REGION\_NOT\_WORK\_REGION | 2.000 | 0.000 |
| LANDAREA\_AVG | 3472.000 | 59.323 |
| REGION\_RATING\_CLIENT\_W\_CITY | 3.000 | 0.000 |
| OBS\_30\_CNT\_SOCIAL\_CIRCLE | 33.000 | 0.333 |
| LANDAREA\_MODE | 3514.000 | 59.323 |
| REGION\_RATING\_CLIENT | 3.000 | 0.000 |
| OBS\_60\_CNT\_SOCIAL\_CIRCLE | 33.000 | 0.333 |
| AMT\_INCOME\_TOTAL | 2347.000 | 0.000 |
| ENTRANCES\_AVG | 279.000 | 50.283 |
| ENTRANCES\_MEDI | 46.000 | 50.283 |
| NONLIVINGAPARTMENTS\_AVG | 375.000 | 69.386 |
| ENTRANCES\_MODE | 30.000 | 50.283 |
| OCCUPATION\_TYPE | 18.000 | 31.401 |
| AMT\_REQ\_CREDIT\_BUREAU\_HOUR | 5.000 | 13.529 |
| NONLIVINGAPARTMENTS\_MEDI | 209.000 | 69.386 |
| AMT\_ANNUITY | 13414.000 | 0.004 |
| FLAG\_DOCUMENT\_5 | 2.000 | 0.000 |
| HOUR\_APPR\_PROCESS\_START | 24.000 | 0.000 |
| FLAG\_EMAIL | 2.000 | 0.000 |
| AMT\_REQ\_CREDIT\_BUREAU\_QRT | 11.000 | 13.529 |
| FLAG\_OWN\_REALTY | 2.000 | 0.000 |
| CNT\_FAM\_MEMBERS | 17.000 | 0.001 |
| WALLSMATERIAL\_MODE | 7.000 | 50.766 |
| YEARS\_BUILD\_AVG | 149.000 | 66.452 |
| NAME\_FAMILY\_STATUS | 6.000 | 0.000 |
| YEARS\_BUILD\_MEDI | 151.000 | 66.452 |
| YEARS\_BUILD\_MODE | 154.000 | 66.452 |
| WEEKDAY\_APPR\_PROCESS\_START | 7.000 | 0.000 |
| NAME\_TYPE\_SUITE | 7.000 | 0.419 |
| NONLIVINGAPARTMENTS\_MODE | 165.000 | 69.386 |
| FONDKAPREMONT\_MODE | 4.000 | 68.341 |
| YEARS\_BEGINEXPLUATATION\_MEDI | 235.000 | 48.719 |
| YEARS\_BEGINEXPLUATATION\_AVG | 272.000 | 48.719 |
| YEARS\_BEGINEXPLUATATION\_MODE | 215.000 | 48.719 |
| AMT\_REQ\_CREDIT\_BUREAU\_WEEK | 9.000 | 13.529 |
| FLAG\_CONT\_MOBILE | 2.000 | 0.000 |
| FLAG\_MOBIL | 1.000 | 0.000 |

**Appendix 2: Ascending Sorted Features for With- extra Feature Set Selected By Fold Change**

|  |  |  |
| --- | --- | --- |
| **Column Name** | **Level** | **Missing Value** |
| FLAG\_DOCUMENT\_4 | 2 | 0.000 |
| FLAG\_DOCUMENT\_10 | 2 | 0.000 |
| FLAG\_DOCUMENT\_12 | 2 | 0.000 |
| AMT\_CREDIT\_SUM\_OVERDUE | 211 | 0.453 |
| FLAG\_DOCUMENT\_2 | 2 | 0.000 |
| FLAG\_DOCUMENT\_13 | 2 | 0.000 |
| FLAG\_DOCUMENT\_17 | 2 | 0.000 |
| DAYS\_CREDIT\_ENDDATE | 7962 | 0.467 |
| FLAG\_DOCUMENT\_15 | 2 | 0.000 |
| FLAG\_DOCUMENT\_14 | 2 | 0.000 |
| FLAG\_DOCUMENT\_16 | 2 | 0.000 |
| FLAG\_DOCUMENT\_21 | 2 | 0.000 |
| REG\_CITY\_NOT\_LIVE\_CITY | 2 | 0.000 |
| DAYS\_EMPLOYED | 12310 | 0.000 |
| NAME\_CONTRACT\_TYPE | 2 | 0.000 |
| FLAG\_DOCUMENT\_6 | 2 | 0.000 |
| EMERGENCYSTATE\_MODE | 2 | 0.473 |
| AMT\_CREDIT\_MAX\_OVERDUE | 5821 | 0.869 |
| FLAG\_DOCUMENT\_7 | 2 | 0.000 |
| DEF\_60\_CNT\_SOCIAL\_CIRCLE | 9 | 0.003 |
| FLAG\_DOCUMENT\_18 | 2 | 0.000 |
| FLAG\_DOCUMENT\_9 | 2 | 0.000 |
| DEF\_30\_CNT\_SOCIAL\_CIRCLE | 10 | 0.003 |
| REG\_CITY\_NOT\_WORK\_CITY | 2 | 0.000 |
| CREDIT\_CURRENCY | 4 | 0.453 |
| EXT\_SOURCE\_3 | 809 | 0.199 |
| EXT\_SOURCE\_1 | 103908 | 0.564 |
| ELEVATORS\_AVG | 254 | 0.532 |
| ELEVATORS\_MEDI | 46 | 0.532 |
| ELEVATORS\_MODE | 26 | 0.532 |
| CODE\_GENDER | 3 | 0.000 |
| HOUSETYPE\_MODE | 3 | 0.501 |
| EXT\_SOURCE\_2 | 114271 | 0.002 |
| LIVE\_CITY\_NOT\_WORK\_CITY | 2 | 0.000 |
| FLAG\_DOCUMENT\_20 | 2 | 0.000 |
| AMT\_CREDIT\_SUM | 49050 | 0.453 |
| FLAG\_DOCUMENT\_11 | 2 | 0.000 |
| FLAG\_WORK\_PHONE | 2 | 0.000 |
| DAYS\_LAST\_PHONE\_CHANGE | 3756 | 0.000 |
| DAYS\_CREDIT\_UPDATE | 2871 | 0.453 |
| AMT\_REQ\_CREDIT\_BUREAU\_DAY | 9 | 0.135 |
| AMT\_REQ\_CREDIT\_BUREAU\_MON | 24 | 0.135 |
| CNT\_CREDIT\_PROLONG | 6 | 0.453 |
| AMT\_CREDIT\_SUM\_LIMIT | 3435 | 0.595 |
| DAYS\_CREDIT | 2922 | 0.453 |
| REG\_REGION\_NOT\_LIVE\_REGION | 2 | 0.000 |
| FLAG\_PHONE | 2 | 0.000 |
| LIVINGAREA\_AVG | 5141 | 0.501 |
| LIVINGAREA\_MEDI | 5212 | 0.501 |
| TOTALAREA\_MODE | 5056 | 0.482 |
| LIVINGAREA\_MODE | 5229 | 0.501 |
| OWN\_CAR\_AGE | 60 | 0.660 |
| COMMONAREA\_AVG | 3116 | 0.698 |
| COMMONAREA\_MEDI | 3139 | 0.698 |
| NONLIVINGAREA\_AVG | 3218 | 0.551 |
| NONLIVINGAREA\_MEDI | 3249 | 0.551 |
| NONLIVINGAREA\_MODE | 3261 | 0.551 |
| FLAG\_OWN\_CAR | 2 | 0.000 |
| FLOORSMAX\_AVG | 396 | 0.497 |
| FLOORSMAX\_MEDI | 49 | 0.497 |
| FLOORSMAX\_MODE | 25 | 0.497 |
| CNT\_CHILDREN | 15 | 0.000 |
| DAYS\_REGISTRATION | 15571 | 0.000 |
| COMMONAREA\_MODE | 3062 | 0.698 |
| REG\_REGION\_NOT\_WORK\_REGION | 2 | 0.000 |
| APARTMENTS\_AVG | 2307 | 0.507 |
| APARTMENTS\_MEDI | 1139 | 0.507 |
| CREDIT\_ACTIVE | 3 | 0.453 |
| AMT\_ANNUITY\_y | 10008 | 0.872 |
| APARTMENTS\_MODE | 757 | 0.507 |
| DAYS\_ENDDATE\_FACT | 2869 | 0.626 |
| AMT\_GOODS\_PRICE | 950 | 0.001 |
| FLAG\_DOCUMENT\_8 | 2 | 0.000 |
| FLOORSMIN\_AVG | 300 | 0.678 |
| FLOORSMIN\_MEDI | 47 | 0.678 |
| FLOORSMIN\_MODE | 25 | 0.678 |
| FLAG\_DOCUMENT\_3 | 2 | 0.000 |
| DAYS\_ID\_PUBLISH | 6158 | 0.000 |
| LIVINGAPARTMENTS\_AVG | 1843 | 0.683 |
| REGION\_POPULATION\_RELATIVE | 81 | 0.000 |
| LIVINGAPARTMENTS\_MEDI | 1089 | 0.683 |
| NAME\_INCOME\_TYPE | 8 | 0.000 |
| LIVINGAPARTMENTS\_MODE | 734 | 0.683 |
| NAME\_HOUSING\_TYPE | 6 | 0.000 |
| BASEMENTAREA\_AVG | 3741 | 0.585 |
| BASEMENTAREA\_MEDI | 3734 | 0.585 |
| ORGANIZATION\_TYPE | 58 | 0.000 |
| DAYS\_BIRTH | 17444 | 0.000 |
| NAME\_EDUCATION\_TYPE | 5 | 0.000 |
| AMT\_CREDIT | 5451 | 0.000 |
| FLAG\_EMP\_PHONE | 2 | 0.000 |
| BASEMENTAREA\_MODE | 3800 | 0.585 |
| AMT\_REQ\_CREDIT\_BUREAU\_YEAR | 25 | 0.135 |
| FLAG\_DOCUMENT\_19 | 2 | 0.000 |
| CREDIT\_DAY\_OVERDUE | 189 | 0.453 |
| LANDAREA\_MEDI | 3507 | 0.593 |
| LIVE\_REGION\_NOT\_WORK\_REGION | 2 | 0.000 |
| LANDAREA\_AVG | 3472 | 0.593 |
| REGION\_RATING\_CLIENT\_W\_CITY | 3 | 0.000 |
| OBS\_30\_CNT\_SOCIAL\_CIRCLE | 33 | 0.003 |
| LANDAREA\_MODE | 3514 | 0.593 |
| REGION\_RATING\_CLIENT | 3 | 0.000 |
| OBS\_60\_CNT\_SOCIAL\_CIRCLE | 33 | 0.003 |
| AMT\_INCOME\_TOTAL | 2347 | 0.000 |
| ENTRANCES\_AVG | 279 | 0.503 |
| ENTRANCES\_MEDI | 46 | 0.503 |
| NONLIVINGAPARTMENTS\_AVG | 375 | 0.694 |
| ENTRANCES\_MODE | 30 | 0.503 |
| OCCUPATION\_TYPE | 18 | 0.314 |
| AMT\_REQ\_CREDIT\_BUREAU\_HOUR | 5 | 0.135 |
| NONLIVINGAPARTMENTS\_MEDI | 209 | 0.694 |
| AMT\_ANNUITY\_x | 13414 | 0.000 |
| FLAG\_DOCUMENT\_5 | 2 | 0.000 |
| HOUR\_APPR\_PROCESS\_START | 24 | 0.000 |
| FLAG\_EMAIL | 2 | 0.000 |
| AMT\_REQ\_CREDIT\_BUREAU\_QRT | 11 | 0.135 |
| FLAG\_OWN\_REALTY | 2 | 0.000 |
| CREDIT\_TYPE | 9 | 0.453 |
| AMT\_CREDIT\_SUM\_DEBT | 27510 | 0.498 |
| CNT\_FAM\_MEMBERS | 17 | 0.000 |
| WALLSMATERIAL\_MODE | 7 | 0.508 |
| YEARS\_BUILD\_AVG | 149 | 0.665 |
| NAME\_FAMILY\_STATUS | 6 | 0.000 |
| YEARS\_BUILD\_MEDI | 151 | 0.665 |
| YEARS\_BUILD\_MODE | 154 | 0.665 |
| WEEKDAY\_APPR\_PROCESS\_START | 7 | 0.000 |
| NAME\_TYPE\_SUITE | 7 | 0.004 |
| NONLIVINGAPARTMENTS\_MODE | 165 | 0.694 |
| FONDKAPREMONT\_MODE | 4 | 0.683 |
| SK\_ID\_BUREAU | 150265 | 0.453 |
| YEARS\_BEGINEXPLUATATION\_MEDI | 235 | 0.487 |
| YEARS\_BEGINEXPLUATATION\_AVG | 272 | 0.487 |
| YEARS\_BEGINEXPLUATATION\_MODE | 215 | 0.487 |
| AMT\_REQ\_CREDIT\_BUREAU\_WEEK | 9 | 0.135 |
| FLAG\_CONT\_MOBILE | 2 | 0.000 |
| FLAG\_MOBIL | 1 | 0.000 |

**Appendix 3: Ascending Sorted Features for Without- extra Feature Set Selected by Pearson Correlation**

|  |  |  |
| --- | --- | --- |
| **Column Name** | **Level** | **Missing Value** |
| EXT\_SOURCE\_3 | 809 | 19.869 |
| EXT\_SOURCE\_2 | 114271 | 0.216 |
| EXT\_SOURCE\_1 | 103908 | 56.376 |
| DAYS\_BIRTH | 17444 | 0.000 |
| REGION\_RATING\_CLIENT\_W\_CITY | 3 | 0.000 |
| REGION\_RATING\_CLIENT | 3 | 0.000 |
| DAYS\_LAST\_PHONE\_CHANGE | 3756 | 0.000 |
| CODE\_GENDER | 3 | 0.000 |
| NAME\_EDUCATION\_TYPE | 5 | 0.000 |
| DAYS\_ID\_PUBLISH | 6158 | 0.000 |
| REG\_CITY\_NOT\_WORK\_CITY | 2 | 0.000 |
| NAME\_INCOME\_TYPE | 8 | 0.000 |
| FLAG\_EMP\_PHONE | 2 | 0.000 |
| FLOORSMAX\_AVG | 396 | 49.698 |
| DAYS\_EMPLOYED | 12310 | 0.000 |
| FLOORSMAX\_MEDI | 49 | 49.698 |
| FLAG\_DOCUMENT\_3 | 2 | 0.000 |
| FLOORSMAX\_MODE | 25 | 49.698 |
| REG\_CITY\_NOT\_LIVE\_CITY | 2 | 0.000 |
| DAYS\_REGISTRATION | 15571 | 0.000 |
| AMT\_GOODS\_PRICE | 950 | 0.091 |
| REGION\_POPULATION\_RELATIVE | 81 | 0.000 |
| OWN\_CAR\_AGE | 60 | 65.985 |
| ELEVATORS\_AVG | 254 | 53.220 |
| ELEVATORS\_MEDI | 46 | 53.220 |
| FLOORSMIN\_AVG | 300 | 67.798 |
| FLOORSMIN\_MEDI | 47 | 67.798 |
| FLOORSMIN\_MODE | 25 | 67.798 |
| NAME\_HOUSING\_TYPE | 6 | 0.000 |
| ELEVATORS\_MODE | 26 | 53.220 |
| LIVINGAREA\_AVG | 5141 | 50.142 |
| LIVINGAREA\_MEDI | 5212 | 50.142 |
| LIVE\_CITY\_NOT\_WORK\_CITY | 2 | 0.000 |
| TOTALAREA\_MODE | 5056 | 48.204 |
| DEF\_30\_CNT\_SOCIAL\_CIRCLE | 10 | 0.333 |
| DEF\_60\_CNT\_SOCIAL\_CIRCLE | 9 | 0.333 |
| NAME\_CONTRACT\_TYPE | 2 | 0.000 |
| LIVINGAREA\_MODE | 5229 | 50.142 |
| ORGANIZATION\_TYPE | 58 | 0.000 |
| AMT\_CREDIT | 5451 | 0.000 |
| APARTMENTS\_AVG | 2307 | 50.677 |
| APARTMENTS\_MEDI | 1139 | 50.677 |
| FLAG\_DOCUMENT\_6 | 2 | 0.000 |
| FLAG\_WORK\_PHONE | 2 | 0.000 |
| APARTMENTS\_MODE | 757 | 50.677 |
| LIVINGAPARTMENTS\_AVG | 1843 | 68.312 |
| LIVINGAPARTMENTS\_MEDI | 1089 | 68.312 |
| HOUR\_APPR\_PROCESS\_START | 24 | 0.000 |
| LIVINGAPARTMENTS\_MODE | 734 | 68.312 |
| FLAG\_PHONE | 2 | 0.000 |
| BASEMENTAREA\_AVG | 3741 | 58.469 |
| YEARS\_BUILD\_MEDI | 151 | 66.452 |
| YEARS\_BUILD\_AVG | 149 | 66.452 |
| FLAG\_OWN\_CAR | 2 | 0.000 |
| YEARS\_BUILD\_MODE | 154 | 66.452 |
| BASEMENTAREA\_MEDI | 3734 | 58.469 |
| AMT\_INCOME\_TOTAL | 2347 | 0.000 |
| OCCUPATION\_TYPE | 18 | 31.401 |
| AMT\_REQ\_CREDIT\_BUREAU\_YEAR | 25 | 13.529 |
| CNT\_CHILDREN | 15 | 0.000 |
| BASEMENTAREA\_MODE | 3800 | 58.469 |
| ENTRANCES\_AVG | 279 | 50.283 |
| COMMONAREA\_AVG | 3116 | 69.832 |
| COMMONAREA\_MEDI | 3139 | 69.832 |
| ENTRANCES\_MEDI | 46 | 50.283 |
| ENTRANCES\_MODE | 30 | 50.283 |
| COMMONAREA\_MODE | 3062 | 69.832 |
| EMERGENCYSTATE\_MODE | 2 | 47.337 |
| AMT\_ANNUITY | 13414 | 0.004 |
| NONLIVINGAREA\_AVG | 3218 | 55.119 |
| AMT\_REQ\_CREDIT\_BUREAU\_MON | 24 | 13.529 |
| NONLIVINGAREA\_MEDI | 3249 | 55.119 |
| FLAG\_DOCUMENT\_13 | 2 | 0.000 |
| LANDAREA\_MEDI | 3507 | 59.323 |
| FLAG\_DOCUMENT\_16 | 2 | 0.000 |
| NONLIVINGAREA\_MODE | 3261 | 55.119 |
| WALLSMATERIAL\_MODE | 7 | 50.766 |
| LANDAREA\_AVG | 3472 | 59.323 |
| LANDAREA\_MODE | 3514 | 59.323 |
| YEARS\_BEGINEXPLUATATION\_MEDI | 235 | 48.719 |
| CNT\_FAM\_MEMBERS | 17 | 0.001 |
| YEARS\_BEGINEXPLUATATION\_AVG | 272 | 48.719 |
| FLAG\_DOCUMENT\_14 | 2 | 0.000 |
| NAME\_TYPE\_SUITE | 7 | 0.419 |
| OBS\_30\_CNT\_SOCIAL\_CIRCLE | 33 | 0.333 |
| HOUSETYPE\_MODE | 3 | 50.113 |
| OBS\_60\_CNT\_SOCIAL\_CIRCLE | 33 | 0.333 |
| YEARS\_BEGINEXPLUATATION\_MODE | 215 | 48.719 |
| FLAG\_DOCUMENT\_8 | 2 | 0.000 |
| REG\_REGION\_NOT\_WORK\_REGION | 2 | 0.000 |
| FLAG\_DOCUMENT\_18 | 2 | 0.000 |
| FLAG\_OWN\_REALTY | 2 | 0.000 |
| FLAG\_DOCUMENT\_15 | 2 | 0.000 |
| FLAG\_DOCUMENT\_2 | 2 | 0.000 |
| REG\_REGION\_NOT\_LIVE\_REGION | 2 | 0.000 |
| WEEKDAY\_APPR\_PROCESS\_START | 7 | 0.000 |
| FLAG\_DOCUMENT\_9 | 2 | 0.000 |
| NAME\_FAMILY\_STATUS | 6 | 0.000 |
| FLAG\_DOCUMENT\_21 | 2 | 0.000 |
| AMT\_REQ\_CREDIT\_BUREAU\_DAY | 9 | 13.529 |
| FLAG\_DOCUMENT\_17 | 2 | 0.000 |
| LIVE\_REGION\_NOT\_WORK\_REGION | 2 | 0.000 |
| FLAG\_DOCUMENT\_11 | 2 | 0.000 |
| FONDKAPREMONT\_MODE | 4 | 68.341 |
| FLAG\_DOCUMENT\_4 | 2 | 0.000 |
| NONLIVINGAPARTMENTS\_AVG | 375 | 69.386 |
| NONLIVINGAPARTMENTS\_MEDI | 209 | 69.386 |
| AMT\_REQ\_CREDIT\_BUREAU\_QRT | 11 | 13.529 |
| FLAG\_DOCUMENT\_10 | 2 | 0.000 |
| FLAG\_EMAIL | 2 | 0.000 |
| FLAG\_DOCUMENT\_20 | 2 | 0.000 |
| FLAG\_DOCUMENT\_7 | 2 | 0.000 |
| FLAG\_DOCUMENT\_5 | 2 | 0.000 |
| FLAG\_DOCUMENT\_12 | 2 | 0.000 |
| AMT\_REQ\_CREDIT\_BUREAU\_HOUR | 5 | 13.529 |
| FLAG\_DOCUMENT\_19 | 2 | 0.000 |
| NONLIVINGAPARTMENTS\_MODE | 165 | 69.386 |
| FLAG\_CONT\_MOBILE | 2 | 0.000 |
| AMT\_REQ\_CREDIT\_BUREAU\_WEEK | 9 | 13.529 |
| FLAG\_MOBIL | 1 | 0.000 |

**Appendix 4: Ascending Sorted Features for With- extra Feature Set Selected by Pearson Correlation**

|  |  |  |
| --- | --- | --- |
| **Column Name** | **Level** | **Missing Value** |
| EXT\_SOURCE\_3 | 809 | 0.199 |
| EXT\_SOURCE\_2 | 114271 | 0.002 |
| EXT\_SOURCE\_1 | 103908 | 0.564 |
| DAYS\_BIRTH | 17444 | 0.000 |
| REGION\_RATING\_CLIENT\_W\_CITY | 3 | 0.000 |
| REGION\_RATING\_CLIENT | 3 | 0.000 |
| DAYS\_CREDIT | 2922 | 0.453 |
| DAYS\_LAST\_PHONE\_CHANGE | 3756 | 0.000 |
| CODE\_GENDER | 3 | 0.000 |
| NAME\_EDUCATION\_TYPE | 5 | 0.000 |
| DAYS\_ID\_PUBLISH | 6158 | 0.000 |
| REG\_CITY\_NOT\_WORK\_CITY | 2 | 0.000 |
| NAME\_INCOME\_TYPE | 8 | 0.000 |
| FLAG\_EMP\_PHONE | 2 | 0.000 |
| FLOORSMAX\_AVG | 396 | 0.497 |
| DAYS\_EMPLOYED | 12310 | 0.000 |
| FLOORSMAX\_MEDI | 49 | 0.497 |
| FLAG\_DOCUMENT\_3 | 2 | 0.000 |
| FLOORSMAX\_MODE | 25 | 0.497 |
| REG\_CITY\_NOT\_LIVE\_CITY | 2 | 0.000 |
| DAYS\_REGISTRATION | 15571 | 0.000 |
| CREDIT\_ACTIVE | 3 | 0.453 |
| AMT\_GOODS\_PRICE | 950 | 0.001 |
| DAYS\_CREDIT\_UPDATE | 2871 | 0.453 |
| REGION\_POPULATION\_RELATIVE | 81 | 0.000 |
| DAYS\_ENDDATE\_FACT | 2869 | 0.626 |
| OWN\_CAR\_AGE | 60 | 0.660 |
| ELEVATORS\_AVG | 254 | 0.532 |
| ELEVATORS\_MEDI | 46 | 0.532 |
| FLOORSMIN\_AVG | 300 | 0.678 |
| FLOORSMIN\_MEDI | 47 | 0.678 |
| FLOORSMIN\_MODE | 25 | 0.678 |
| NAME\_HOUSING\_TYPE | 6 | 0.000 |
| ELEVATORS\_MODE | 26 | 0.532 |
| LIVINGAREA\_AVG | 5141 | 0.501 |
| LIVINGAREA\_MEDI | 5212 | 0.501 |
| LIVE\_CITY\_NOT\_WORK\_CITY | 2 | 0.000 |
| TOTALAREA\_MODE | 5056 | 0.482 |
| DEF\_30\_CNT\_SOCIAL\_CIRCLE | 10 | 0.003 |
| DEF\_60\_CNT\_SOCIAL\_CIRCLE | 9 | 0.003 |
| NAME\_CONTRACT\_TYPE | 2 | 0.000 |
| LIVINGAREA\_MODE | 5229 | 0.501 |
| ORGANIZATION\_TYPE | 58 | 0.000 |
| AMT\_CREDIT | 5451 | 0.000 |
| APARTMENTS\_AVG | 2307 | 0.507 |
| APARTMENTS\_MEDI | 1139 | 0.507 |
| FLAG\_DOCUMENT\_6 | 2 | 0.000 |
| DAYS\_CREDIT\_ENDDATE | 7962 | 0.467 |
| FLAG\_WORK\_PHONE | 2 | 0.000 |
| APARTMENTS\_MODE | 757 | 0.507 |
| LIVINGAPARTMENTS\_AVG | 1843 | 0.683 |
| LIVINGAPARTMENTS\_MEDI | 1089 | 0.683 |
| HOUR\_APPR\_PROCESS\_START | 24 | 0.000 |
| LIVINGAPARTMENTS\_MODE | 734 | 0.683 |
| FLAG\_PHONE | 2 | 0.000 |
| BASEMENTAREA\_AVG | 3741 | 0.585 |
| YEARS\_BUILD\_MEDI | 151 | 0.665 |
| YEARS\_BUILD\_AVG | 149 | 0.665 |
| FLAG\_OWN\_CAR | 2 | 0.000 |
| YEARS\_BUILD\_MODE | 154 | 0.665 |
| BASEMENTAREA\_MEDI | 3734 | 0.585 |
| AMT\_INCOME\_TOTAL | 2347 | 0.000 |
| OCCUPATION\_TYPE | 18 | 0.314 |
| AMT\_REQ\_CREDIT\_BUREAU\_YEAR | 25 | 0.135 |
| BASEMENTAREA\_MODE | 3800 | 0.585 |
| ENTRANCES\_AVG | 279 | 0.503 |
| COMMONAREA\_AVG | 3116 | 0.698 |
| COMMONAREA\_MEDI | 3139 | 0.698 |
| ENTRANCES\_MEDI | 46 | 0.503 |
| ENTRANCES\_MODE | 30 | 0.503 |
| COMMONAREA\_MODE | 3062 | 0.698 |
| EMERGENCYSTATE\_MODE | 2 | 0.473 |
| AMT\_ANNUITY\_x | 13414 | 0.000 |
| NONLIVINGAREA\_AVG | 3218 | 0.551 |
| AMT\_REQ\_CREDIT\_BUREAU\_MON | 24 | 0.135 |
| CREDIT\_TYPE | 9 | 0.453 |
| NONLIVINGAREA\_MEDI | 3249 | 0.551 |
| FLAG\_DOCUMENT\_13 | 2 | 0.000 |
| LANDAREA\_MEDI | 3507 | 0.593 |
| FLAG\_DOCUMENT\_16 | 2 | 0.000 |
| NONLIVINGAREA\_MODE | 3261 | 0.551 |
| WALLSMATERIAL\_MODE | 7 | 0.508 |
| LANDAREA\_AVG | 3472 | 0.593 |
| AMT\_CREDIT\_SUM | 49050 | 0.453 |
| LANDAREA\_MODE | 3514 | 0.593 |
| YEARS\_BEGINEXPLUATATION\_MEDI | 235 | 0.487 |
| SK\_ID\_BUREAU | 150265 | 0.453 |
| CNT\_FAM\_MEMBERS | 17 | 0.000 |
| YEARS\_BEGINEXPLUATATION\_AVG | 272 | 0.487 |
| FLAG\_DOCUMENT\_14 | 2 | 0.000 |
| NAME\_TYPE\_SUITE | 7 | 0.004 |
| OBS\_30\_CNT\_SOCIAL\_CIRCLE | 33 | 0.003 |
| HOUSETYPE\_MODE | 3 | 0.501 |
| OBS\_60\_CNT\_SOCIAL\_CIRCLE | 33 | 0.003 |
| YEARS\_BEGINEXPLUATATION\_MODE | 215 | 0.487 |
| FLAG\_DOCUMENT\_8 | 2 | 0.000 |
| REG\_REGION\_NOT\_WORK\_REGION | 2 | 0.000 |
| FLAG\_DOCUMENT\_18 | 2 | 0.000 |
| AMT\_CREDIT\_SUM\_OVERDUE | 211 | 0.453 |
| FLAG\_OWN\_REALTY | 2 | 0.000 |
| FLAG\_DOCUMENT\_15 | 2 | 0.000 |
| FLAG\_DOCUMENT\_2 | 2 | 0.000 |
| REG\_REGION\_NOT\_LIVE\_REGION | 2 | 0.000 |
| AMT\_CREDIT\_MAX\_OVERDUE | 5821 | 0.869 |
| WEEKDAY\_APPR\_PROCESS\_START | 7 | 0.000 |
| FLAG\_DOCUMENT\_9 | 2 | 0.000 |
| NAME\_FAMILY\_STATUS | 6 | 0.000 |
| AMT\_CREDIT\_SUM\_LIMIT | 3435 | 0.595 |
| FLAG\_DOCUMENT\_21 | 2 | 0.000 |
| AMT\_REQ\_CREDIT\_BUREAU\_DAY | 9 | 0.135 |
| FLAG\_DOCUMENT\_17 | 2 | 0.000 |
| LIVE\_REGION\_NOT\_WORK\_REGION | 2 | 0.000 |
| AMT\_ANNUITY\_y | 10008 | 0.872 |
| FLAG\_DOCUMENT\_11 | 2 | 0.000 |
| FONDKAPREMONT\_MODE | 4 | 0.683 |
| FLAG\_DOCUMENT\_4 | 2 | 0.000 |
| NONLIVINGAPARTMENTS\_AVG | 375 | 0.694 |
| CNT\_CREDIT\_PROLONG | 6 | 0.453 |
| CREDIT\_CURRENCY | 4 | 0.453 |
| NONLIVINGAPARTMENTS\_MEDI | 209 | 0.694 |
| AMT\_REQ\_CREDIT\_BUREAU\_QRT | 11 | 0.135 |
| FLAG\_DOCUMENT\_10 | 2 | 0.000 |
| FLAG\_EMAIL | 2 | 0.000 |
| FLAG\_DOCUMENT\_20 | 2 | 0.000 |
| FLAG\_DOCUMENT\_7 | 2 | 0.000 |
| FLAG\_DOCUMENT\_5 | 2 | 0.000 |
| FLAG\_DOCUMENT\_12 | 2 | 0.000 |
| AMT\_REQ\_CREDIT\_BUREAU\_HOUR | 5 | 0.135 |
| AMT\_CREDIT\_SUM\_DEBT | 27510 | 0.498 |
| FLAG\_DOCUMENT\_19 | 2 | 0.000 |
| CREDIT\_DAY\_OVERDUE | 189 | 0.453 |
| NONLIVINGAPARTMENTS\_MODE | 165 | 0.694 |
| FLAG\_CONT\_MOBILE | 2 | 0.000 |
| AMT\_REQ\_CREDIT\_BUREAU\_WEEK | 9 | 0.135 |
| FLAG\_MOBIL | 1 | 0.000 |

**Appendix 5: Features Under MAR Assumption for Without-Extra-Feature Data Set**

|  |
| --- |
| EXT\_SOURCE\_3 |
| EXT\_SOURCE\_2 |
| EXT\_SOURCE\_1 |
| DAYS\_BIRTH |
| REGION\_RATING\_CLIENT\_W\_CITY |
| REGION\_RATING\_CLIENT |
| DAYS\_LAST\_PHONE\_CHANGE |
| CODE\_GENDER |
| NAME\_EDUCATION\_TYPE |
| DAYS\_ID\_PUBLISH |
| REG\_CITY\_NOT\_WORK\_CITY |
| NAME\_INCOME\_TYPE |
| FLAG\_EMP\_PHONE |
| DAYS\_EMPLOYED |
| FLAG\_DOCUMENT\_3 |
| REG\_CITY\_NOT\_LIVE\_CITY |
| DAYS\_REGISTRATION |
| AMT\_GOODS\_PRICE |
| REGION\_POPULATION\_RELATIVE |
| NAME\_HOUSING\_TYPE |
| LIVE\_CITY\_NOT\_WORK\_CITY |
| DEF\_30\_CNT\_SOCIAL\_CIRCLE |
| DEF\_60\_CNT\_SOCIAL\_CIRCLE |
| NAME\_CONTRACT\_TYPE |
| ORGANIZATION\_TYPE |
| AMT\_CREDIT |
| FLAG\_DOCUMENT\_6 |
| FLAG\_WORK\_PHONE |
| HOUR\_APPR\_PROCESS\_START |
| FLAG\_PHONE |
| FLAG\_OWN\_CAR |
| AMT\_INCOME\_TOTAL |
| AMT\_REQ\_CREDIT\_BUREAU\_YEAR |
| CNT\_CHILDREN |
| AMT\_ANNUITY |
| AMT\_REQ\_CREDIT\_BUREAU\_MON |
| FLAG\_DOCUMENT\_13 |
| FLAG\_DOCUMENT\_16 |
| YEARS\_BEGINEXPLUATATION\_MEDI |
| CNT\_FAM\_MEMBERS |
| YEARS\_BEGINEXPLUATATION\_AVG |
| FLAG\_DOCUMENT\_14 |
| NAME\_TYPE\_SUITE |
| OBS\_30\_CNT\_SOCIAL\_CIRCLE |
| HOUSETYPE\_MODE |
| OBS\_60\_CNT\_SOCIAL\_CIRCLE |
| YEARS\_BEGINEXPLUATATION\_MODE |
| FLAG\_DOCUMENT\_8 |
| REG\_REGION\_NOT\_WORK\_REGION |
| FLAG\_DOCUMENT\_18 |
| FLAG\_OWN\_REALTY |
| FLAG\_DOCUMENT\_15 |
| FLAG\_DOCUMENT\_2 |
| REG\_REGION\_NOT\_LIVE\_REGION |
| WEEKDAY\_APPR\_PROCESS\_START |
| FLAG\_DOCUMENT\_9 |
| NAME\_FAMILY\_STATUS |
| FLAG\_DOCUMENT\_21 |
| AMT\_REQ\_CREDIT\_BUREAU\_DAY |
| FLAG\_DOCUMENT\_17 |
| LIVE\_REGION\_NOT\_WORK\_REGION |
| FLAG\_DOCUMENT\_11 |
| FONDKAPREMONT\_MODE |
| FLAG\_DOCUMENT\_4 |
| NONLIVINGAPARTMENTS\_AVG |
| NONLIVINGAPARTMENTS\_MEDI |
| AMT\_REQ\_CREDIT\_BUREAU\_QRT |
| FLAG\_DOCUMENT\_10 |
| FLAG\_EMAIL |
| FLAG\_DOCUMENT\_20 |
| FLAG\_DOCUMENT\_7 |
| FLAG\_DOCUMENT\_5 |
| FLAG\_DOCUMENT\_12 |
| AMT\_REQ\_CREDIT\_BUREAU\_HOUR |
| FLAG\_DOCUMENT\_19 |
| NONLIVINGAPARTMENTS\_MODE |
| FLAG\_CONT\_MOBILE |
| AMT\_REQ\_CREDIT\_BUREAU\_WEEK |
| FLAG\_MOBIL |

**Appendix 6: Features Under MAR Assumption for With-Extra-Feature Data Set**

|  |
| --- |
| CREDIT\_ACTIVE |
| CREDIT\_CURRENCY |
| DAYS\_CREDIT |
| CREDIT\_DAY\_OVERDUE |
| DAYS\_CREDIT\_ENDDATE |
| DAYS\_ENDDATE\_FACT |
| AMT\_CREDIT\_MAX\_OVERDUE |
| CNT\_CREDIT\_PROLONG |
| AMT\_CREDIT\_SUM |
| AMT\_CREDIT\_SUM\_DEBT |
| AMT\_CREDIT\_SUM\_LIMIT |
| AMT\_CREDIT\_SUM\_OVERDUE |
| CREDIT\_TYPE |
| DAYS\_CREDIT\_UPDATE |
| AMT\_ANNUITY |
| EXT\_SOURCE\_3 |
| EXT\_SOURCE\_2 |
| EXT\_SOURCE\_1 |
| DAYS\_BIRTH |
| REGION\_RATING\_CLIENT\_W\_CITY |
| REGION\_RATING\_CLIENT |
| DAYS\_LAST\_PHONE\_CHANGE |
| CODE\_GENDER |
| NAME\_EDUCATION\_TYPE |
| DAYS\_ID\_PUBLISH |
| REG\_CITY\_NOT\_WORK\_CITY |
| NAME\_INCOME\_TYPE |
| FLAG\_EMP\_PHONE |
| DAYS\_EMPLOYED |
| FLAG\_DOCUMENT\_3 |
| REG\_CITY\_NOT\_LIVE\_CITY |
| DAYS\_REGISTRATION |
| AMT\_GOODS\_PRICE |
| REGION\_POPULATION\_RELATIVE |
| NAME\_HOUSING\_TYPE |
| LIVE\_CITY\_NOT\_WORK\_CITY |
| DEF\_30\_CNT\_SOCIAL\_CIRCLE |
| DEF\_60\_CNT\_SOCIAL\_CIRCLE |
| NAME\_CONTRACT\_TYPE |
| ORGANIZATION\_TYPE |
| AMT\_CREDIT |
| FLAG\_DOCUMENT\_6 |
| FLAG\_WORK\_PHONE |
| HOUR\_APPR\_PROCESS\_START |
| FLAG\_PHONE |
| FLAG\_OWN\_CAR |
| AMT\_INCOME\_TOTAL |
| AMT\_REQ\_CREDIT\_BUREAU\_YEAR |
| CNT\_CHILDREN |
| AMT\_ANNUITY |
| AMT\_REQ\_CREDIT\_BUREAU\_MON |
| FLAG\_DOCUMENT\_13 |
| FLAG\_DOCUMENT\_16 |
| YEARS\_BEGINEXPLUATATION\_MEDI |
| CNT\_FAM\_MEMBERS |
| YEARS\_BEGINEXPLUATATION\_AVG |
| FLAG\_DOCUMENT\_14 |
| NAME\_TYPE\_SUITE |
| OBS\_30\_CNT\_SOCIAL\_CIRCLE |
| HOUSETYPE\_MODE |
| OBS\_60\_CNT\_SOCIAL\_CIRCLE |
| YEARS\_BEGINEXPLUATATION\_MODE |
| FLAG\_DOCUMENT\_8 |
| REG\_REGION\_NOT\_WORK\_REGION |
| FLAG\_DOCUMENT\_18 |
| FLAG\_OWN\_REALTY |
| FLAG\_DOCUMENT\_15 |
| FLAG\_DOCUMENT\_2 |
| REG\_REGION\_NOT\_LIVE\_REGION |
| WEEKDAY\_APPR\_PROCESS\_START |
| FLAG\_DOCUMENT\_9 |
| NAME\_FAMILY\_STATUS |
| FLAG\_DOCUMENT\_21 |
| AMT\_REQ\_CREDIT\_BUREAU\_DAY |
| FLAG\_DOCUMENT\_17 |
| LIVE\_REGION\_NOT\_WORK\_REGION |
| FLAG\_DOCUMENT\_11 |
| FONDKAPREMONT\_MODE |
| FLAG\_DOCUMENT\_4 |
| NONLIVINGAPARTMENTS\_AVG |
| NONLIVINGAPARTMENTS\_MEDI |
| AMT\_REQ\_CREDIT\_BUREAU\_QRT |
| FLAG\_DOCUMENT\_10 |
| FLAG\_EMAIL |
| FLAG\_DOCUMENT\_20 |
| FLAG\_DOCUMENT\_7 |
| FLAG\_DOCUMENT\_5 |
| FLAG\_DOCUMENT\_12 |
| AMT\_REQ\_CREDIT\_BUREAU\_HOUR |
| FLAG\_DOCUMENT\_19 |
| NONLIVINGAPARTMENTS\_MODE |
| FLAG\_CONT\_MOBILE |
| AMT\_REQ\_CREDIT\_BUREAU\_WEEK |
| FLAG\_MOBIL |

**Appendix 7: Column Names Under MNAR for Both Without- and With- Extra Feature Data Set**

|  |
| --- |
| FLOORSMAX\_AVG |
| FLOORSMAX\_MEDI |
| FLOORSMAX\_MODE |
| OWN\_CAR\_AGE |
| ELEVATORS\_AVG |
| ELEVATORS\_MEDI |
| FLOORSMIN\_AVG |
| FLOORSMIN\_MEDI |
| FLOORSMIN\_MODE |
| ELEVATORS\_MODE |
| LIVINGAREA\_AVG |
| LIVINGAREA\_MEDI |
| TOTALAREA\_MODE |
| LIVINGAREA\_MODE |
| APARTMENTS\_AVG |
| APARTMENTS\_MEDI |
| APARTMENTS\_MODE |
| LIVINGAPARTMENTS\_AVG |
| LIVINGAPARTMENTS\_MEDI |
| LIVINGAPARTMENTS\_MODE |
| BASEMENTAREA\_AVG |
| YEARS\_BUILD\_MEDI |
| YEARS\_BUILD\_AVG |
| YEARS\_BUILD\_MODE |
| BASEMENTAREA\_MEDI |
| OCCUPATION\_TYPE |
| BASEMENTAREA\_MODE |
| ENTRANCES\_AVG |
| COMMONAREA\_AVG |
| COMMONAREA\_MEDI |
| ENTRANCES\_MEDI |
| ENTRANCES\_MODE |
| COMMONAREA\_MODE |
| EMERGENCYSTATE\_MODE |
| NONLIVINGAREA\_AVG |
| NONLIVINGAREA\_MEDI |
| LANDAREA\_MEDI |
| NONLIVINGAREA\_MODE |
| WALLSMATERIAL\_MODE |
| LANDAREA\_AVG |
| LANDAREA\_MODE |