MMF2030 Machine Learning Group Component

Junhao Wang, Fei Wei, Bona Zhang

January 22, 2021

1 Data Transformations

1.1 Data

Home credit focuses on installment lending primarily to people with little or no credit history. It offers credit for a range of products including cell phones, computers, jewelry, and some other personal needs. In this project, we used three datasets from the source files provided:

1) application_train.csv: This is the main dataset we were using, which contains 307,511 samples, 1 unique ID, 1 target variable and 120 features. One sample represents one loan in this case.
2) bureau.csv: The file includes the previous credits of all clients provided by other financial institutions that were reported to Credit Bureau before the application date.
3) credit_card_balance.csv: The file provides information about the applicant's monthly balance of previous credit cards with Home Credit. Each row in the table represents a month's history of every previous credit in Home Credit (consumer credit and cash loans) related to loans in our sample.

1.2 Data Dictionary for Final Feature Table

Feature	Description
amt_balance_0_1_year	AMT balance 0-1 year
AMT_CREDIT	Credit amount of the loan
AMT_GOODS_PRICE	For consumer loans it is the price of the goods for which the loan is given
CODE_GENDER_F	Gender: Female
CODE_GENDER_M	Gender: Male
DAYS_BIRTH	Client's age in days at the time of application
DAYS_EMPLOYED	How many days before the application the person started current employment
DAYS_ID_PUBLISH	How many days before the application did client change the identity document with which he applied for the loan
DAYS_LAST_PHONE_CHANGE	How many days before application did client change phone
EXT_SOURCE_3	Normalized score from external data source
FLAG_DOCUMENT_3	Did client provide document 3
FLAG_EMP_PHONE	Did client provide work phone (1=YES, 0=NO)
NAME_EDUCATION_TYPE_Higher education	Education Type: Higher education
NAME_EDUCATION_TYPE_Secondary / secondary special	Education Type: Secondary/secondary special
NAME_INCOME_TYPE_Pensioner	Income Type: Pensioner
NAME_INCOME_TYPE_Working	Inceom Type: Working
ORGANIZATION_TYPE_XNA	Organization Type: XNA
REG_CITY_NOT_LIVE_CITY	Flag if client's permanent address does not match contact address (1=different, 0=same, at city level)
REG_CITY_NOT_WORK_CITY	Flag if client's permanent address does not match work address (1=different, 0=same, at city level)
REGION_RATING_CLIENT	Our rating of the region where client lives (1,2,3)
REGION_RATING_CLIENT_W_CITY	Our rating of the region where client lives with taking city into account (1,2,3)

Figure 1: Data Dictionary for Final Feature Table

The data dictionary for final feature table is shown in Figure 1. There were 21 features in total remained in the end.

1.3 Transformation Process

Firstly, we did exploratory data analysis (EDA) on checking the missing values.

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

Figure 2: Features with Missing Values

According to the table in Figure 2, many features have large percentages of missing values, that these features have low predicting ability and will not be helpful when building the model. Hence, we decided to remove the features that has missing values larger then 30%, where we dropped 50 features. Besides, we implemented one-hot encoding for categorical variables, which is a process where categorical variables are converted into a form that could be provided to improve ML algorithms' predicting ability. The resulting dataset has 168 features in total.

Based on the remained feature sets, we did transformation of the feature sets into Weight of Evidence (WOE) without involving new conceptual features in order to perform feature engineering and selection. The WOE_i for group i is based on the probability density functions (f_G and f_B) for two binary events (In our credit analysis, the binary events are default and non-default):

$$f_G(i) = \frac{N_i^G}{N_{Total}^G}, f_B(i) = \frac{N_i^B}{N_{Total}^B}, WOE_i = ln(\frac{f_G(i)}{f_B(i)}).$$

where N^G is the number of non-defaulted accounts and N^B is the number of defaulted accounts.

We utilized the package `xverse` to achieve this transformation. This package is useful for machine learning in the space of feature engineering, feature transformation and feature selection. Once we finished the transformation, we could also be access to the Information Value (IV). IV measures the predictive power of a single variable, which can be calculated as weighted sum of WOE:

$$IV = \sum_{i=1}^{n} [f_G(i) - f_B(i)] \times WOE_i.$$

To select the features, we use the rule of thumb and code below:

< 0.02	very weak
0.02 - 0.1	weak
0.1 - 0.3	medium
$0.3 <$	strong

Variable_Name	Information_Value
EXT_SOURCE_2	0.246709
EXT_SOURCE_3	0.227621
DAYS_BIRTH	0.074309
NAME_EDUCATION_TYPE_Higher education	0.050990
NAME_INCOME_TYPE_Working	0.049861
REGION_RATING_CLIENT_W_CITY	0.045145
DAYS_LAST_PHONE_CHANGE	0.042736
DAYS_EMPLOYED	0.042685
REGION_RATING_CLIENT	0.042037
CODE_GENDER_M	0.040438
CODE_GENDER_F	0.040408
NAME_EDUCATION_TYPE_Secondary / secondary special	0.038651
amt_balance_0_1_year	0.036210
NAME_INCOME_TYPE_Pensioner	0.035595
FLAG_EMP_PHONE	0.035277
ORGANIZATION_TYPE_XNA	0.035255
DAYS_ID_PUBLISH	0.033313
AMT_CREDIT	0.032570
AMT_GOODS_PRICE	0.031889
REG_CITY_NOT_WORK_CITY	0.031836
FLAG_DOCUMENT_3	0.026392
REG_CITY_NOT_LIVE_CITY	0.020642

Figure 3: Final Features with Corresponding IVs

After dropping the features with IV less than 0.02 in our data, 146 features have been removed in total. The remaining 21 features' IV values are shown in Figure 3. Eventually, we could have the final feature table in Section 1.2 with all features been WOE transformed.

1.4 Quality Assurance (QA)

Quality assurance (QA) is an important component in machine learning. It is an essential contributor to safety, and it has become necessary to develop QA and regulatory frameworks for higher risk artificial intelligence applications.

We firstly did a spot check on the calculation we had. Figure 4 below shows part of the calculated values we had, and we made sure that everything looks reasonable.

	SK_ID_PREV	SK_ID_CURR	MONTHS_BALANCE	AMT_BALANCE	AMT_CREDIT_LIMIT_ACTUAL	AMT_DRAWINGS_ATM_CURRENT
617886	1914954	100028	-1	37335.915	225000	0.0
1660670	1914954	100028	-2	24710.085	225000	0.0
1835811	1914954	100028	-3	20221.830	225000	0.0
1561465	1914954	100028	-4	0.000	225000	0.0
1460388	1914954	100028	-5	0.000	225000	0.0
1272681	1914954	100028	-6	0.000	225000	0.0
1498585	1914954	100028	-7	0.000	225000	0.0
1384907	1914954	100028	-8	0.000	225000	0.0
3060632	1914954	100028	-9	6438.330	225000	0.0
2466412	1914954	100028	-10	7888.545	225000	0.0
3017565	1914954	100028	-11	18248.895	225000	0.0
3749066	1914954	100028	-12	16717.410	225000	0.0

Figure 4: Spot Check the Calculation

Secondly, we looked at the data on an aggregation level, where the important parameters for part of the features are shown in Figure 5. We observed that there are potential imbalance data issues for some of the features such as "AMT_BALANCE" that need to be dealt with later.

	SK_ID_PREV	SK_ID_CURR	MONTHS_BALANCE	AMT_BALANCE	AMT_CREDIT_LIMIT_ACTUAL	AMT_DRAWINGS_ATM_CURRENT	AMT_DRAWINGS_CURRENT
count	3.84E+06	3.84E+06	3.84E+06	3.84E+06	3.84E+06	3.09E+06	3.84E+06
mean	1.90E+06	2.78E+05	-3.45E+01	5.83E+04	1.54E+05	5.96E+03	7.43E+03
std	5.36E+05	1.03E+05	2.67E+01	1.06E+05	1.65E+05	2.82E+04	3.38E+04
min	1.00E+06	1.00E+05	-9.60E+01	-4.20E+05	0.00E+00	-6.83E+03	-6.21E+03
25%	1.43E+06	1.90E+05	-5.50E+01	0.00E+00	4.50E+04	0.00E+00	0.00E+00
50%	1.90E+06	2.78E+05	-2.80E+01	0.00E+00	1.13E+05	0.00E+00	0.00E+00
75%	2.37E+06	3.68E+05	-1.10E+01	8.90E+04	1.80E+05	0.00E+00	0.00E+00
max	2.84E+06	4.56E+05	-1.00E+00	1.51E+06	1.35E+06	2.12E+06	2.29E+06

Figure 5: Aggregation Check

The third method we were using is to check the shapes before and after downsampling our data (stratified) for speed. As shown in Figure 6, we had 307,511 samples before, and after the downsampling, there were only 53,094 samples remained.

```
print(train_factorized.shape, train_compute_mi_sample.shape)
(307511, 170) (53094, 170)
```

Figure 6: Shape Check

2 Models

2.1 Performance Measures & Rationale

In order to evaluate and validate how good the machine learning models we have built in code as candidates for the best model, we chose **AUC-ROC Curve** and **Confusion Matrix** with several important metrics such as recall, precision and f1-score to measure the model performance.

The AUC-ROC curve helps us visualize how well our machine learning models as classifiers are performing. **Receiver Operator Characteristic (ROC)** curve is one of the validation metrics for binary classification. It is a probability curve that graphs the True Postitive Rate (TPR) against False Positive Rate (FPR) and fundamentally splits the 'signal' from the 'noise'. ROC curve can be summarized by the area under the ROC curve (AUC) Statistics. **AUC** Statistics represents degree of separability. It tells the measure of the ability of a model to distinguish between classes. The higher the AUC, the better the performance of the model is at predicting 0s as 0s and 1s as 1s. By analogy, the higher the AUC, the better the model is at distinguishing between clients without payment difficulties and clients with payment difficulties correspondingly.

The metrics in the **Confusion Matrix**, namely recall, precision and f1-score are the crucial components behind the AUC-ROC curve and evaluate the machine learning models with a specific threshold. The confusion matrix contains the following elements:

$$\begin{bmatrix} \text{True Positive} & \text{False Positive} \\ \text{False Negative} & \text{True Negative} \end{bmatrix}$$

which gives

- **True Positive (TP)** for correctly predicted event values, where data points with actual positive values are classifed as positive by the model;

- **False Positive (FP)** for incorrectly predicted event values, where data points with actual negative values are classified as positive by the model;
- **True Negative (TN)** for correctly predicted no-event value, where data points with actual negative values are classified as negative by the model;
- **False Negative (FN)** for incorrectly predicted no-event values, where data points with actual positive values are classified as positive by the model.

From the confusion matrix, we can derive **precision** and **recall (TRP)**, which have the following formulas:

$$Precision = \frac{TP}{TP + FN}$$

$$Recall = \frac{TP}{TP + FP}$$

In our dataset, TP cases are correctly detected clients with payment difficulties, while FP cases indicate the customers that the model classifies as clients without payment difficulties that actually have difficulties. Besides, FP occurs when customers the model detects as clients with payment difficulties that are actually not.

We realized that if we increased recall to make it a desirable classifier, the model would suffer from low precision. Therefore, there is a trade-off between recall and precision. As a result, we wanted to find an optimal combination of precision and recall by maximizing f1-score. The calculation of **f1-score** is formulated as follows:

$$F\text{-}1\ Score = 2 \times \frac{Precision \times Recall}{Precision + Recall}$$

Apart from calculating the corresponding f1-scores directly through changing thresholds and finding the specific threshold that maximizes the f1-score, instead, we knew by default that the threshold is 0.5 where the values for recall and precision close to each other maximize the value of f1-score. Therefore the overall best model performance occurs when threshold is 0.5 because of the close similarity on the values of recall and precision to be discussed in Section 2.2.3 later.

2.2 Best Model Search

2.2.1 Approaches/Models

The first approach we used is random forest. It is an ensemble learning method for classification, regression and other tasks that is operated by constructing a multitude of decision trees at training time and outputting the class that is the mode of the classes (classification) or mean/average prediction (regression) of the individual trees. Random forest corrects for decision trees' habit of overfitting to their training set. We firstly implemented random forest on our dataset with k-fold where k equals 5 for the best speed. To improve the performance, we used K-means clustering. K-means clustering is a type of unsupervised learning, which is used when the data is without defined categories or groups. The goal of this algorithm is to find groups in the data, with the number of groups represented by the variable K. The algorithm works iteratively to assign each data point to one of K groups based on the features that are provided. Rather than defining groups before looking at the data, clustering allows us to find and analyze the groups that have formed organically. We chose K equals to 5 again in this case.

Besides, we also implemented XGBoost to the dataset. XGBoost is a popular and efficient open-source implementation of the gradient boosted trees algorithm. Gradient boosting is a supervised learning algorithm that attempts to accurately predict a target variable by combining the estimates of a set of simpler, weaker models. When using gradient boosting for regression, the weak learners are regression trees, and each regression tree maps an input data point to one of its leafs that contains a continuous score. XGBoost

minimizes a regularized objective function that combines a convex loss function based on the difference between the predicted and target outputs, and a penalty term for model complexity. The training proceeds iteratively, adding new trees that predict the residuals or errors of prior trees that are then combined with previous trees to make the final prediction. It is called gradient boosting because it uses a gradient descent algorithm to minimize the loss when adding new models.

Another model we used is based on Multi-Layer Perceptron (MLP). MLP is a class of feedforward artificial neural network (ANN). It consists of at least three layers of nodes: an input layer, a hidden layer and an output layer. Except for the input nodes, each node is a neuron that uses a nonlinear activation function. MLP utilizes a supervised learning technique called backpropagation for training. Its multiple layers and non-linear activation distinguish MLP from a linear perceptron. MLP can distinguish data that is not linearly separable.

Last but not least, we fitted a logistic regression model to our data. Logistic regression is a machine learning algorithm which is used for the classification problems, it is a predictive analysis algorithm and based on the concept of probability. We firstly fitted a logistic regression model on the data with k-fold where k equals 5.

2.2.2 Hyperparameter Tuning

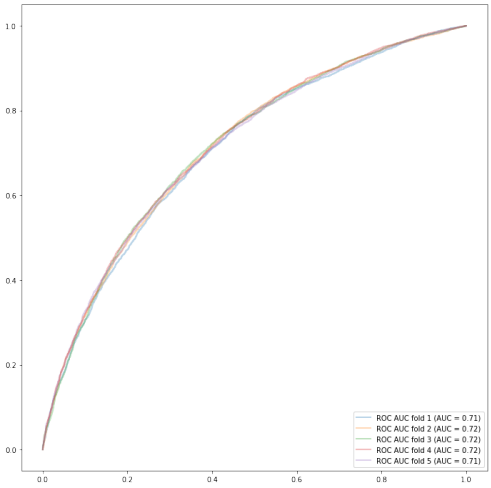
A hyperparameter is a parameter whose value is used to control the learning process. Hyperparameter tuning is the process of choosing a set of optimal hyperparameters for a machine learning algorithm.

In XGBoost model, we tuned the hyperparameters of `'max_depth'`, `'min_child_weight'`, `'subsample'`, and `'colsample_bytree'`. We firstly set the hyperparameters to their base values, and used grid search for tuning and helping compute the optimum values of hyperparameters. Grid search is an exhaustive search that is performed on a the specific parameter values of a model, where the model is also known as an estimator. It will build a model for every combination of hyperparameters specified and evaluate each model and find the best parameter values. After tuning, we found the most optimal values for the parameters are: `'max_depth'` $= 4$, `'min_child_weight'` $= 1$, `'subsample'` $= 0.8$, and `'colsample_bytree'` $= 1$.

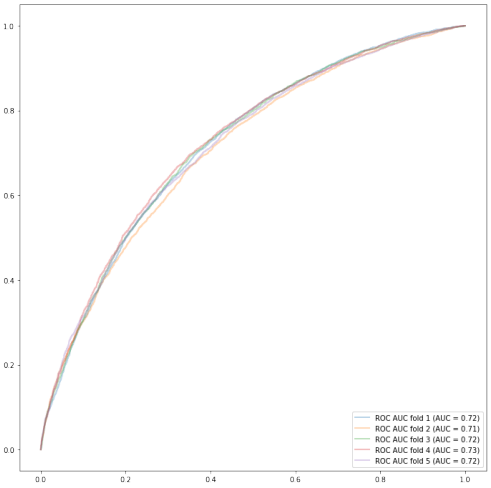
In MLP model, we wanted to tune the hyperparameters of `'hidden_layer_sizes'` and `'activation'`. Instead of implementing grid search which takes long time to run, we conducted `RandomizedSearchCV` to perform cross validation and find the optimal hyperparameters. `RandomizedSearchCV` implements a "fit" method and a "predict" method like any classifier except that the parameters of the classifier used to predict is optimized by cross-validation. In contrast to grid search, not all parameter values are tried out, but rather a fixed number of parameter settings is sampled from the specified distributions. The number of parameter settings that are tried is given by `n_iter`. After tuning, we had the best hyperparameters that `'hidden_layer_sizes' = (70,)` and `'activation' = 'logistic'`.

In logistic regression, grid search was also used when tuning the hyperparameter of `'C'`. We got the most optimal value of 0.01274 after tuning.

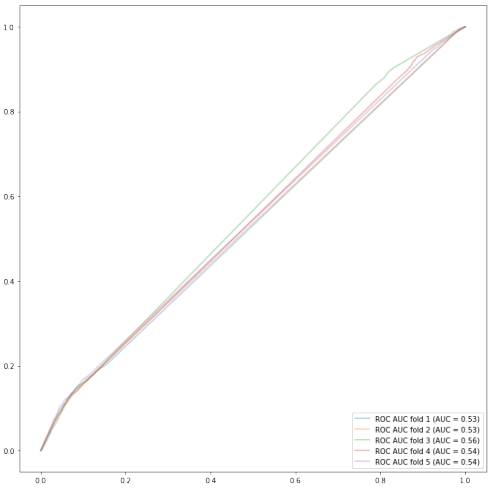
2.2.3 Evaluation Loss



(a) Random Forest



(b) XGBoost



(c) MLP

Figure 7: AUC-ROC Curve

Firstly, we compare the models from Task 1. Figure 7 plots the AUC-ROC curves for the three models. According to the terminology, the closer the AUC is to 1, the better the model performance. As the figures illustrate, XGBoost model has the best performance with the highest average AUC value of 0.72.

```
[[4750  970]
 [2683 2216]]
```

(a) Random Forest

```
[[2188 3532]
 [ 569 4330]]
```

(b) XGBoost

```
[[ 176 5544]
 [  55 4844]]
```

(c) MLP

Figure 8: Confusion Matrix

	precision	recall	f1-score	support
0	0.639	0.830	0.722	5720
1	0.696	0.452	0.548	4899
accuracy			0.656	10619
macro avg	0.667	0.641	0.635	10619
weighted avg	0.665	0.656	0.642	10619

(a) Random Forest

	precision	recall	f1-score	support
0	0.794	0.383	0.516	5720
1	0.551	0.884	0.679	4899
accuracy			0.614	10619
macro avg	0.672	0.633	0.597	10619
weighted avg	0.682	0.614	0.591	10619

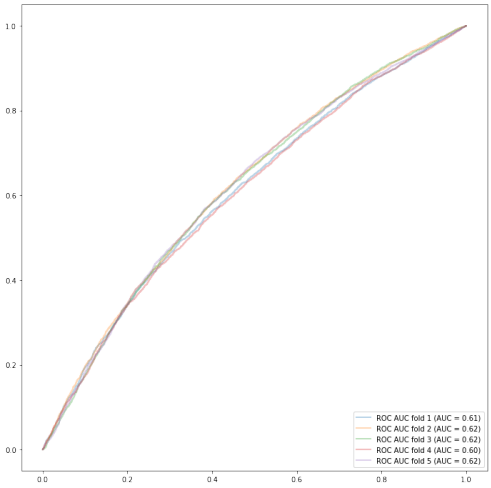
(b) XGBoost

	precision	recall	f1-score	support
0	0.762	0.031	0.059	5720
1	0.466	0.989	0.634	4899
accuracy			0.473	10619
macro avg	0.614	0.510	0.346	10619
weighted avg	0.626	0.473	0.324	10619

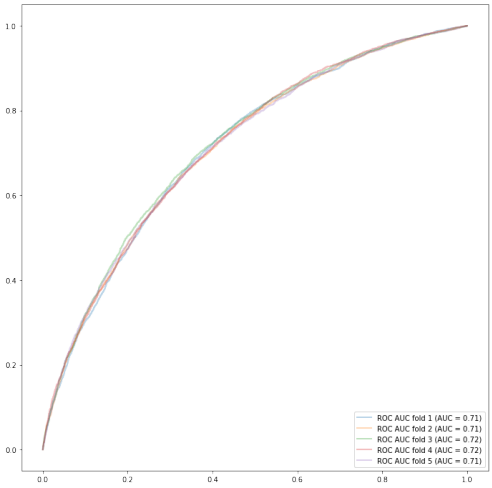
(c) MLP

Figure 9: Precision, Recall and F-1 Score

Figure 8 shows the confusion matrices for the three models. To have a clearer comparison, we derived precision, recall and f-1 score from the matrix for each model, as shown in Figure 9. As stated earlier, to get rid of the trade-off between precision and recall, we should look for the model whose f-1 score is close to the most optimal value of 0.5. Since the f-1 scores for '0' and '1' in XGBoost model are 0.516 and 0.679 respectively, which are really close to 0.5. So XGBoost model also performs the best based on this metric.



(a) Logistic Regression without Feature Selection



(b) Logistic Regression with Feature Selection

Figure 10: Logistic Regression Model AUC-ROC Curves

```
[[4515 1205]
 [3180 1719]]
```

(a) Logistic Regression without Feature Selection

```
[[3997 1723]
 [1821 3078]]
```

(b) Logistic Regression with Feature Selection

Figure 11: Logistic Regression Model Confusion Matrix

	precision	recall	f1-score	support
0	0.587	0.789	0.673	5720
1	0.588	0.351	0.439	4899
accuracy			0.587	10619
macro avg	0.587	0.570	0.556	10619
weighted avg	0.587	0.587	0.565	10619

(a) Logistic Regression without Feature Selection

	precision	recall	f1-score	support
0	0.687	0.699	0.693	5720
1	0.641	0.628	0.635	4899
accuracy			0.666	10619
macro avg	0.664	0.664	0.664	10619
weighted avg	0.666	0.666	0.666	10619

(b) Logistic Regression with Feature Selection

Figure 12: Logistic Regression Model Precision, Recall and F-1 Score

Next we want to compare the models we have in Task 2, which are the logistic regression models without and with feature selection. The AUC-ROC curves, confusion matrix, and precision, recall and f-1 score values are displayed in Figures 10, 11 and 12 respectively. It is clearly to see that after feature selection via IV, the AUC-ROC curves become more curved up, and the average AUC values increased from 0.62 to 0.72, which is much closer to 1. Besides, the predicted TP and TN values become larger and more balanced after doing feature selection. By looking at the f-1 score, it becomes closer to 0.5 and more balanced. Hence, the logistic regression model after feature selection outperforms the one before conducting feature selection.

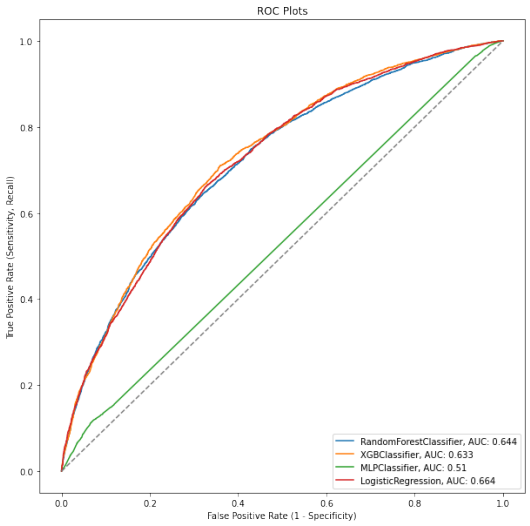


Figure 13: AUC-ROC Curves

In addition, to compare the models together in a easier way, we implemented the model `classifier` object, where the AUC-ROC curves are plotted in Figure 13. As illustrated in the values of AUC, logistic regression model after feature selection has largest value of AUC of 0.664, thus having the best performance.

3 Conclusion

In this project, we built four models on our data using random forest, XGBoost, MLP, and logistic regression. By transforming the feature sets to WOE and exclude features with IV values less than 0.2, we were able to narrow down to our final 21 features. When evaluating the model performance, we used the rationales of AUC-ROC curve, confusion matrix, precision, recall, and f-1 score. We noticed that there is a trade-off between precision and recall. Therefore, a better way is to search for models whose f-1 score is closest to the threshold of 0.5. In addition, we also did hyperparameter tuning using the methods of grid search and `RandomizedSearchCV` to improve the learning process. By comparing the models of random forest, XGBoost, and MLP, we found that XGBoost model has the best performance. In the comparison

of logistic regression models without and with feature selection, it was observed that feature selection improved the model performance a lot. Furthermore, we also compared the four models (i.e. random forest, XGBoost, MLP, logistic regression with feature selection) together via the model `classifier` object and found out the logistic regression model with feature selection has the best performance among all.