**Credit Scoring: Default in 2 years?**

**Abstract**

Credit is one of the most discussed and fundamental topics in the banking industry. In this project, we aim to explore how to make better prediction about whether a loan borrower would experience serious default of past due over 90 days in the next two year. The project is a classification problem and we use logistic regression as our baseline model. Other than that, eight different models are evaluated and compared (e.g. adaptive boosting, gradient boosting and decision trees) to find the optimal model. Major evaluation metrics we use include accuracy, AUC score, and precision and recall ratios. We find that adaptive boosting performs best for our credit scoring problem so far.

**1. Introduction**

In this project, we are dedicated to building an optimal model to predict whether a loan borrower would default or not in the next two years. Our goal is to help banks to improve the credit scoring mechanism and thus make better decisions on granting a loan to a particular borrower. We obtained our dataset from a Kaggle competition “Give me some credit” in 2011 with 150000 data entries and 11 variables.

Our dependent variable is “SeriousDlqin2yrs”, which indicates whether a person would experience a credit delinquency of more than or equal to 90 days past due. In the banking industry, most bank lenders would start reporting delinquency to the government’s credit bureaus when a loan is more than 30 days past due; and a credit delinquency of 90 days past due or more is usually considered as serious. Therefore, we consider this occurrence of 90-day-or-more credit delinquency worthy of studying in particular. In our dataset, this dependent variable is binary, with “1” indicating “serious 90+ day past due delinquency happens” while “0” indicating not. Overall, our question of interest is a classification problem. As for independent variables, “Line of Credit (LOC)” refers to an agreement between a financial institution and a borrower on the maximum amount of loan that could be borrowed in a certain period of time. The borrower can get access to any amount of fund under the “credit limit” at any time within the period, as long as they make timely minimum repayment. As a result, we believe the number of LOCs a borrower have and the borrower’s usage of LOCs are closely related to our question of interest. Other information about the borrower, such as data on demographics and personal or family financial conditions, is also valuable.

To address this classification problem, we apply different models, including logistic regression, trees, boosting models, Naive Bayes, K-Nearest Neighbor and neural network. In order to find the optimal model, we also use Grid Search and compare the models with the metrics of accuracy, together with AUC score and precision and recall ratios. Eventually, we find that boosting models work best for our project.

**2.Literature Review**

In order to have a basic understanding of how credit algorithm is constructed in the past, we conduct our literature review. In “What’s the Point of Credit Scoring” by Loretta J. Mester, the President and CEO of the Federal Reserve Bank of Cleveland, an explicit introduction about credit scoring is given, which helps build the conceptual knowledge of what credit scoring is and how it make sense to banks as well as other business institutions. In terms of models used in the past, “Benchmarking state-of-the-art classification algorithms for credit scoring” gives a great summary of both well-known traditional and recently used models, such as logistic regression, *k*-nearest neighbour, neural networks and decision trees. Another article which also contributes to our project is “Default Predictors and Credit Scoring Models for Retail Banking”. In the article, the author discussed their findings about the characteristics of default behavior in a pretty thorough way and emphasised the socio-demographic variables which seem to be irrelevant should not be excluded in the credit scoring models. This article offers a guideline about features we may consider to reserve.

**3.Data Description**

The original dataset from Kaggle contains 15000 entries and 11 variables in total. Table 1 in Appendix presents the details of the variables. During the data cleaning process, we aim to solve three main issues: missing values, outliers and data imbalance.

Due to the rich amount of data available, we first dropped entries with two or more columns of missing values. In addition, during the exploratory data analysis, we notice that for entries with “NumberOfTimes90DaysLate” equal to 96 or 98, their “RevolvingUtilizationOfUnsecuredLines” values are all equal to 0.99999999. In addition, they have many zero entries in other columns. We believe that these are misentries, so we dropped them as well. Before we proceed to further data cleaning procedures, we split the dataset into training set and testing set at the ratio of 0.1. The details of data cleaning for missing values and outliers of the training set is as follows:

|  |  |
| --- | --- |
| **Variable Name** | **Method** |
| RevolvingUtilizationOfUnsecuredLines | Replace values larger than 1 with 95th quantile |
| Age | Replace age under 18 with 25th quantile  Replace age over 80 with 75th quantile |
| DebtRatio | Replace ratios larger than 1 with forward filling |
| NumberOfOpenCreditLinesAndLoans | Replace lines more than 20 with 95th quantile |
| MonthlyIncome | Drop rows with missing values  Replace income less than 1000 with 5th quantile  Replace income more than 20,000 with 95th quantile |
| NumberRealEstateLoansOrLines | Replace lines more than 4 with 95th quantile |
| NumberOfDependents | Replace dependents more than 5 with 95th quantile |

After cleaning missing values and outliers, we observe that the majority class (dependent variable = 0) has 100683 entries whereas the minority class (dependent variable = 1) only has 7450 rows. We attempt to address the imbalance with resampling method. We first use SMOTE to oversample the minority class with a ratio of 0.4. Then, we undersample the majority class to achieve a well-balanced dataset.

For the testing set, we use the same method to deal with the missing values and outliers. The value we imputed in the testing set is from the training set statistics. Moreover, in order to better evaluate the model performance, we apply the same resampling method to the testing set. After data preprocessing, we have a training set with 80546 entries and a testing set with 10890 entries.

**4.Methodology**

In this project, we apply a variety of machine learning models to compare and discuss their performance, including logistic regressions, K Nearest Neighbors, Naive Bayes, Neural Network, Classification Tree, and ensemble models such as Random Forest, Gradient Boosting, Adaptive Boosting, and Stacking model combining the top performers. The major model evaluation metrics are overall accuracy score, confusion matrix, AUC score, and precision and recall ratios. Some of the key metrics are defined as follows:

Notice that the overall accuracy score and precision and recall ratios are measures with a specific threshold. Precision focuses on false positives (Type II error), while recall focuses on false negatives (Type I error). For any specific threshold, there exists a tradeoff between false positives and false negatives. In comparison, the AUC score demonstrates the average proportion of time one can predict “which is which” correctly, given a random positive example and a random negative example, over all threshold levels. AUC is more robust than accuracy when the data is imbalanced. During the entire training and testing process, overall accuracy is the main performance measure that are intended to be optimized, while AUC, precision and recall also serve as references.

**5. Model**

**5.1. Baseline model: Logistic Regressions**

Without any hyperparameter tuning, we set off by using the basic logistic regression model with default L2 regularization in the scikit-learn library. We get a 0.7934 training accuracy score, and a 0.5274 testing accuracy score, which is slightly better than random guess. Based on the confusion matrix on testing data, the precision is 0.4793 and the recall is 0.1942. Similar to testing accuracy score, the testing AUC score here is 0.5262. After scaling the x covariates with Robust Scaler, we try parameter tuning for L1-regularized and L2-regularized logistic regression models respectively: by using GridSearchCV on a 10-based log-scaled parameter grid of “C” , which is defined as the inverse of the regularization term, we obtain the optimal “C” of 0.0158 for L1-regularized logistic regression and 0.0019 for L2-regularized logistic regression. However, these models do not perform very well. Their testing accuracy score is less than 50%, and AUC scores and precision and recall ratios close to the first default L2-regularized model. Since a small “C” term suggests a large regularization term correspondingly, this probably indicates that logistic regressions are too simple in this context so that a large regularization term dampens the model performance further. Overall, we can see obviously that logistic regressions have a poor performance, so here we use logistic regression models as the baseline for our project.

**5.2. K-Nearest Neighbors(KNN)**

For K-Nearest Neighbors model, its performance could be seriously affected by abnormal large scales, causing all of neighbour points to crowdedly align together in the dimensions with smaller range, therefore, we normalize independent variables with Robust Scaler. We obtain a training accuracy of 0.905 and testing accuracy of 0.5446. The precision and recall ratios, as well as training and testing AUC scores, slightly exceed those of logistic regressions. Through plotting and manual parameter tuning, we find that the nearest-neighbor model (KNN with number of neighbors set to 1) is the optimal estimator. After tuning, the training and testing accuracy, and precision and recall improve a little, but still do not seem very good, yet require much time for training. The KNN algorithm is not working very well possibly because the “distance” among points might not be a most appropriate measure of similarity here.

**5.3. Neural Network**

We run a basic neural network model utilizing the sequential model from the Keras library. With one input layer and three hidden layers with ReLU (Rectified Linear Unit) activation function and 64 input nodes, the Sequential model uses L2 regularization and sigmoid output function. However, like the KNN algorithm, the neural network model demands more computing power and takes exceptionally longer time to process. What’s more, it turns out to be performing badly, only with a 0.5185 testing accuracy and a 0.5485 testing AUC in this case.

**5.4. Naive Bayes**

For the Gaussian Naive Bayes, we run the model after the Robust Scaler standardization. The training accuracy is 0.7452, while the testing accuracy is only 0.4952; the training and testing AUC scores are 0.8672 and 0.5405. At the same time, the precision ratio of 0.4781 and the recall ratio of 0.1051 appear to be particularly low as compared to any other model. One explanation could be the underlying assumption that the likelihood of the independent features follows the Gaussian distribution, which very likely does not conform to our situation. In addition, we cannot resort to other principal Naive Bayes models including Multinomial Naive Bayes and Bernoulli Naive Bayes here. Multinomial Naive Bayes is intended to to be used when it comes to discrete counts of outcomes and does not allow negative covariate values; Bernoulli Naive Bayes works with binary-valued features essentially. Therefore, the Naive Bayes models do not seem to be useful here.

**5.5. Classification Tree**

Classification trees categorize data points according to levels of different conditions. We scale the covariates with Robust Scaler first. By applying a decision tree classifier with a maximum depth of 5 in the beginning, we attain an initial visualization of the tree structure, illustrating the importance of features (Appendix Graph 3). Here the number of times the borrower paid back 30-59 days past due in the past two years is the dominant feature; the number of times of 90 days past due previously and the revolving utilization rate of unsecured credit lines are also quite significant. After drawing the graph, we attempt to run the model without tuning first (with limitless depth of tree and minimum samples at leaves of 2), which gives us a 0.99996 training accuracy and a 0.7436 testing accuracy, similar training and testing AUC scores, along with a precision of 0.8032 and a recall of 0.4339. Then we take three hyperparameters for tuning - max\_depth, min\_samples\_leaf, and min\_samples\_split. Before doing GridSearchCV, we first explore the trend of accuracy score over a range of the parameters by plots (Appendix Graph 4). For example, in the graph for exploring the relationship between maximum depth of tree and accuracy, we can observe a turning point at around 12-14 where the gap between training and testing accuracy is the narrowest, balancing bias and variance. Therefore, a closer examination within this specific range should be conducted during GridSearchCV.

For individual parameter tuning, we get an optimal model with depth of 12, minimum samples required at leaves of 50, and samples required for splits of 80; for tuning of all the three parameters combined, the optimum is achieved when these parameters are 14, 50 and 70 respectively. Among all, we obtain the best performance of classification tree when it comes to tuning of the single parameter of “min\_samples\_split” of 80, showing over 76% testing accuracy and AUC scores and a 0.4087 recall ratio. This has been a big improvement over the baseline, and a slight enhancement over the default model without tuning.

**5.6. Boosting**

“Boosting” refers to a family of algorithms that converts weak learner to strong learners. In this project, we implement adaptive boosting and gradient boosting. Since there is no specific assumption for boosting models, we directly fit them with our training data. After parameters tuning, their performance improves a little. For adaptive boosting, we tune the learning rate and n\_estimators and end up with the optimal model with a learning rate of 1.3 and 2000 estimators. It has a testing accuracy of 0.8570 and AUC score of 0.8923. For gradient boosting, we tune the min\_samples\_split, max\_depth, subsample and n\_estimators. Since there are more parameters here, this tuning process take more time than adaboosting. The best parameters are max depth of 5, minimum samples of 4, 391 estimators and 1 subsample. The optimal model reaches performance of 0.8464 for testing accuracy and 0.8831 for AUC score. We can see that boosting models perform well both in accuracy and AUC score. Overall, the training process for boosting models is efficient.

**5.7. Random Forests**

Random forests are ensemble learning method for classification. Similar to boosting models, there is no specific restriction or assumption for applying random forests. We first start out with a basic model (n\_estimators=300, max\_depth=5), and get a testing accuracy of 0.5967 and AUC score of 0.6885. At first sight, this seems not working well. However, after parameters tuning, our optimal model reaches an accuracy of 0.7858 and AUC score of 0.8495 on the test set. Although random forests and boosting models have similarly good performances, the tuning process for random forest is much slower. Due to the tree structure and number of parameters, this takes more computing power, making the Gridsearch process less efficient. Another thing worth noticing is that random forests achieve 0.9999 accuracy on the training set, so there seems to be overfitting a little.

**5.8. Stacking**

Finally, given the two top-performing models so far--the tuned Adaptive Boosting and Gradient Boosting models, we try out stacking these two models. By integrating predictions from the two models regarding both training and testing data as new inputs and outputs, together with a default logistic regression as the meta training model, we arrive at a 0.9385 training accuracy score and a 0.8510 testing accuracy score, as well as a 0.4531 recall and a 0.8618 testing AUC. It is slimly lower than the top Adaptive Boosting model regarding overall testing accuracy in spite of a stronger training accuracy, which may probably signal minor overfitting issue. Upon inspecting this stacking method, we decide that the single Adaptive Boosting model is the best one for this project.

**6. Results and Interpretations**

Based on our analytical methodologies and models, we pick out the Adaptive Boosting algorithm as our optimal model. We recommend banks considering using a tuned AdaBoost model, perhaps combined with other boosting models, in order to advance loan default prediction, control credit risk and prevent potential loss.

We come to the conclusion that tree-based models are powerful for our project. On the one hand, when trees are united together, they could achieve much better performance than other models we try out. On the other hand, individual trees are easy to visualize and interpret, so single tree models could be a good starting point for prediction tasks.

Throughout the research process, we get to better realize the relative importance of different features. Historical data on varying degrees of loan default behavior in the past might be the most substantial factors to be considered; Interestingly, banks may need to not only focus on similar serious defaults previously (i.e. 90 days past due), but also take records of less significant defaults into consideration. Possibilities of escalation of default may worth attention here: for instance, moderate default behaviors observed in the past may deteriorate into more severe noncompliance as time passes, especially on condition of worsening financial positions or possible accidents happening to the borrower or the borrower’s household. In this way, the numbers of previous defaults of different degrees, as well as other information capturing the borrower’s and the household’s financial situations, might be of great value.

At the same time, some features seem to have lower significance in this project, such as debt ratio, monthly income and the number of open credit lines. Consequently, we become aware that the key issue here might not be one-sided focus on either income or expenditure, but rather the internal relationships among presence of different features for detecting potential abnormal patterns that reflect discrepancies between the borrower’s or the household’s income level and consumption style, which might lead to insolvency.

We notice in particular that in this project, precision appears to be a lot higher than recall. This means that Type II error is more easily committed than Type I error in our current methodologies. In practices of credit scoring, however, Type I error (“predicting non-default when actually there is default”) usually incurs a higher stake than Type II error (“predicting default when actually there is none”), so we hope to further reduce Type II error. The original data set we use may still miss some useful information to be captured, such as country, gender, education and occupation. One of the possible ways that might help is to search other relevant sources and collect more data on useful features, such as the borrower’s or his/her family’s health conditions. Another crucial way comes down to setting the objective scoring function for model training as custom recall ratios when programming, instead of using the default accuracy score in most scikit-learn classification models.

**7. Discussion**

Reviewing from our approaches, we analyze our limitation and potential improvements below. To begin with, although we use eight types of models in total, there are still some models waiting to be examined. Therefore, if more time was given, we were supposed to try more models such as support vector models and discriminant analysis. In addition, in terms of the features we use, instead of directly using the covariates from the data set alone to do the prediction, feature engineering may also be an important part that requires further consideration. To be more specific, we may try to add interaction terms, combine and transform some features mathematically, or try to create some new features by ourselves via extraction of information from existing variables to make prediction.

In addition, for data preprocessing, better performance may also be achieved through improving the methods for cleaning the data. For example, due to the time constraint, we currently use the forward numbers to fulfill the missing values of DebtRatio, which is thought to be relatively random. Alternatively, other methods such as k-nearest neighbors for missing value imputation is worth trying. Another thing worth mentioning is that when we test our models on the test data without SMOTE or any resampling (i.e. imbalanced test set), we find that our top models have very poor performance in AUC scores. More research into this issue could be conducted in the future. Understanding the limitations remaining in our project enables us to reflect the whole project progress in a thorough and critical way, which offers us more inspiration for future improvements afterwards.

**Appendix**

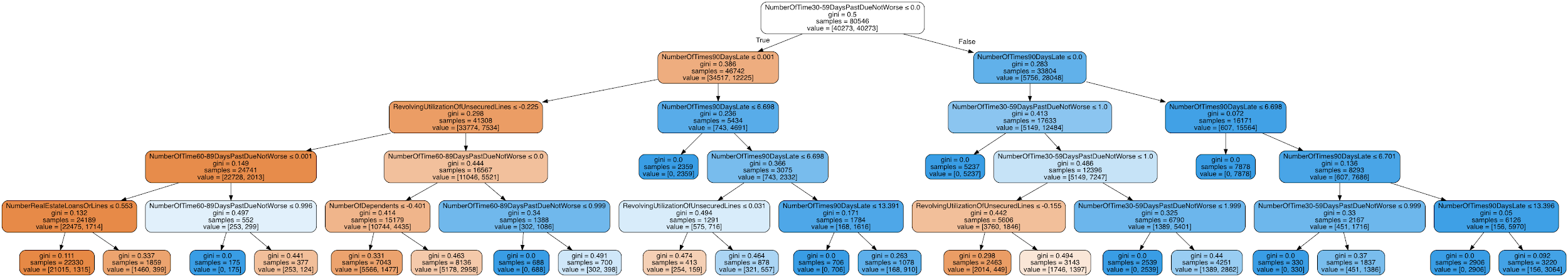
**Table 1. Variables Description**

|  |  |  |
| --- | --- | --- |
| **Variable Name** | **Description** | **Type** |
| SeriousDlqin2yrs | Person experienced 90 days past due delinquency or worse | binary |
| RevolvingUtilizationOfUnsecuredLines | Total balance on credit cards and personal lines of credit except real estate and no installment debt like car loans divided by the sum of credit limits | percentage |
| Age | Age of borrower in years | integer |
| NumberOfTime30-59DaysPastDueNotWorse | Number of times borrower has been 30-59 days past due but no worse in the last 2 years. | integer |
| DebtRatio | Monthly debt payments, alimony,living costs divided by monthy gross income | percentage |
| MonthlyIncome | Monthly income | real |
| NumberOfOpenCreditLinesAndLoans | Number of Open loans (installment like car loan or mortgage) and Lines of credit (e.g. credit cards) | integer |
| NumberOfTimes90DaysLate | Number of times borrower has been 90 days or more past due. | integer |
| NumberRealEstateLoansOrLines | Number of mortgage and real estate loans including home equity lines of credit | integer |
| NumberOfTime60-89DaysPastDueNotWorse | Number of times borrower has been 60-89 days past due but no worse in the last 2 years. | integer |
| NumberOfDependents | Number of dependents in family excluding themselves (spouse, children etc.) | integer |

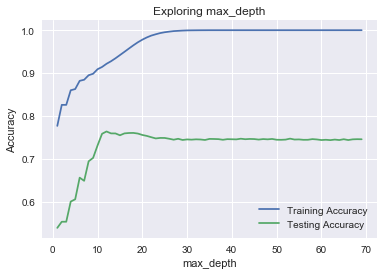
**Graph 2. Correlation Matrix**



**Graph 3. Classification Tree**



**Graph 4. Fitting curve for “max\_depth”**



**Bibliography**

B Baesens, T Van Gestel, S Viaene, M Stepanova, J Suykens & J Vanthienen

(2003) “Benchmarking state-of-the-art classification algorithms for credit scoring”, *Journal of the Operational Research Society*, 54:6, 627-635, DOI:

10.1057/palgrave.jors.2601545

Mester, Loretta J. “What's the Point of Credit Scoring.” *Business Review*, Sept. 1997, econpapers.repec.org/article/fipfedpbr/y\_3a1997\_3ai\_3asep\_3ap\_3a3-16.htm.

Kocenda, Evzen, and Martin Vojtek. “Default Predictors and Credit Scoring Models for Retail Banking.” *SSRN*, 7 Dec. 2009, papers.ssrn.com/sol3/papers.cfm?abstract\_id=1519792.