Prediction Lending Club Loan Status

Chenzi Zhang, NetID chenziz2; Zhaoxu Qiu, NetID zhaoxuq2

Introduction

In our project, we are interested in predicting the probability that a requested loan on LendingClub will charge off. The goal is to build a binary classification model based on the historical loan data issued by Lending Club the with 844006 rows and 30 columns in `loan_stat542.csv`. The 3 train data sets we use to build models have 590806 rows with 30 columns for each. The 3 test data sets we used to examine accuracy have 253200 rows with 30 columns. In every data set, Column 1 `id` is the ID for each loan, Column 13 `loan_status` is the binary response, and Columns 2-30 except column 13 are features that may relate to the reponse.

In this report, we try two models. The XGBoost model needs preprocess to generate new features but efficiently conduct predicting process with high accuracy, which can produce log-loss on the three test sets all lower than 0.45. The Logistic regression model with Lasso also needs hard work on generating new features before fitting a model and result in prediction not as good as lasso, whose log-losses on three test sets are all higher than 0.45 and running time is two times as long as XGBoost model. Thus, we select XGBoost as our classification method.

Method

Data Preprocess

Before calling xgboost, we need to solve missing data problem, select features and deal with features accompanied by too many levels. Here are 4 steps operated on the whole dataset from 2017 to 2018Q2:

Step 1: Set `loan_status`by factor 0 and 1. Class 1 (bad loans): 'Default' or 'Charged Off'; Class 0 (good loans): 'Fully Paid'.

Step 2: Solve missing data problem. Missing categorical data (`emp_title`, `emp_length`, `title`) is converted to `Unkown` level and missing numeric data (`dti`,`total_acc`,`mort_acc`,`pub_rec_bankruptices`)is converted to the median of the column.

Step 3: Deal with factor type features.
1) Drop `zip_code` feature. As a feature represent the location character, `zip_code` is relatively similar to `addr_state`. Since 919 levels are included in `zip_code` feature and `addr_state` only has 51 levels, we choose to remove `zip_code` rather than `addr_state`.
2) Seperate `earliest_cr_line` feature (708 levels). This feature has a format as month-year. So it can be seperated to 2 features `earliest_cr_line_month` and `earliest_cr_line_year`, resulting in less levels for each feature.
3) Convert `emp_title` (205137 levels), `title` (43275 levels) to dummy matrix by top 50 words. By constructing `ProcessLargeFactor()` function, we generate `myvocab` list – 2-element list, 50 words for each list. These 50 words are obtained by `text2vec` package and parameters are set as `term_count_min = 5, doc_proportion_max = 0.5, doc_proportion_min = 0.001`. This list is the vocab pool for further dummy matrix tranformation.

Step 4: Obtain train and test data for 3 splits. For each split:
1) Create `dtm_train` and `dtm_test` based on our vocab pool from step3 by `text2vec` package. Each dtm matrix is corresponding to `emp_title` or `title` with binary values. The columns are the 50 words existing in train or test feature value and the rows are the numbers of observation which are 590806 and 253200.
2) Combine dtm matrix with data matrix from step3. For train or test, we combine data matrix without `emp_title` and `title`, dtm matrix from `emp_title` and dtm matrix from `title`, getting half-dummy-matrix for train and test.
3) Obtain half-dummy-design matrix and reponse vector for train and test data. Drop `id` in design matrix, convert character type features into factor type, use `xgb.DMatrix()` in `xgboost` package to set external pointer.

After all the above steps, we get 590806x126 design matrix for train data and 253200x126 design matrix for test data.

Model1: Xgboost

After data preprocess, we try xgboost model with data from `xgb.DMatrix` pointer and parameter `max.depth=3, eta=0.2, nround=200, verbose = FALSE, objective="binary:logistic"`. To avoid overfitting, tree depth is set as 3. To shrink the weight on each step, we adjust eta as 0.2 to keep high learning rate. Subsampling rate is set as default 1. The loss function we refer to is logloss function. In addition, by tuning, we find that `nround = 200` gives us an efficient model buidling process with low logloss.

Model2: Logistic regression model with Lasso

Before calling Logistic regression model with Lasso, we need to change the X matrix to a numerical matrix (no factors) and do the same as XGBoost model mentioned above.

Use `cv.glmnet()` in `glmnet` packageto find the lambda that minimizes the cv error, and then we use the 1se lambda and choose the option `type = "response"` to obtain the predicted probability on test data. Finally, we get the estimated probability of having `loan_status = 1` from Lasso model and then caculate the log-loss to evaluate the performance.

Loss Function

We use Log Loss to evaluate the performace of our model. Log Loss quantifies the accuracy of a classifier by penalising false classifications. Minimising the Log Loss is basically equivalent to maximising the accuracy of the classifier. A perfect classifier would have a Log Loss of precisely zero. Less ideal classifiers have progressively larger values of Log Loss. If there are only two classes then the expression simplifies to

$$-\frac{1}{N}\sum_{i=1}^{N}[y_i log p_i + (1-y_i) log(1-p_i)]$$

For each instance only the term for the correct class actually contributes to the sum.

Results

Evalution on Prediction Accuracy and Efficiency

Table 1: Log-loss, Running Time (Unit: Mins)

model	split1	split2	split3	split1	split2	split3
XGBoost	0.4471349	0.4486607	0.4472586	8.847513	8.838648	8.916858
Lasso	0.4546815	0.4554209	0.4544172	18.59224	18.73294	18.90183

In general, XGBoost model has better prediction performance and can produce log-loss less than 0.45 over all three test data that that of logistic regression model with Lasso, while the log-loss of Lasso model can't satisfy the benchmark 0.45.

My computer system is iMac which has 3 GHz Intel Core i5 Processor, 16 GB 2400 MHz DDR4 Memory. After comparing running time of these two models, we find Lasso has longer running time, which is is more than two times as long as XGBoost model. Combined with prediction accuracy, we can conclude that XGBoost method has the best efficiency – better performance and shorter running time.

Final Classifier

According to the results shown above, We select XGBoost method. Before calling xgboost, we need to preprocess data sets `LoanStats_2007_to_2018Q2.csv`, `LoanStats_2018Q3.csv` and `LoanStats_2018Q4.csv`:

Step 1: Select same 30 features from data `LoanStats_2007_to_2018Q2.csv`.

Step 2: Remove observations whose reponses are not `Default`, `Charged Off` or `Fully Paid` and set `loan_status`by factor 0 and 1.

Step 3: Solve missing data problem same as mentioned above.

Step 4: Deal with factor type features.
1) Drop `zip_code` feature.
2) Seperate `earliest_cr_line` feature.
3) Convert `emp_title` and `title` to dummy matrix by top 50 words same as mentioned above.

Step 5: Drop `id` in design matrix, convert `int_rate` and `revol_util` into numeric type, use `xgb.DMatrix()` in `xgboost` package to set external pointer. Besides that, for `data3.x` and `data4.x`, we need to select columns included in `data0.x` and set 0 to missing columns.

After data preprocess, we try xgboost model with data `dtrain0` and parameter `max.depth=3, eta=0.2, nround=200, verbose = FALSE, objective="binary:logistic"` mentioned above to build the final classifier. Finally, we use the model built on preprocesswd new data `data3.x` and `data4.x` to predict and calculate the log-loss.

Table 2: log-loss

model	2018Q3	2018Q4
XGBoost	0.3037739	0.1958872

Model Interpret

By calling `xgb.importance()`, we get the importance percentage of features in our xgboost model. The figure1 below shows 10 most important features. Importance means the trend for feature to be taken as root or at least non-leaf nodes. Since our model has 200 trees with depth 3, the trees are different, forming the importance percentage.

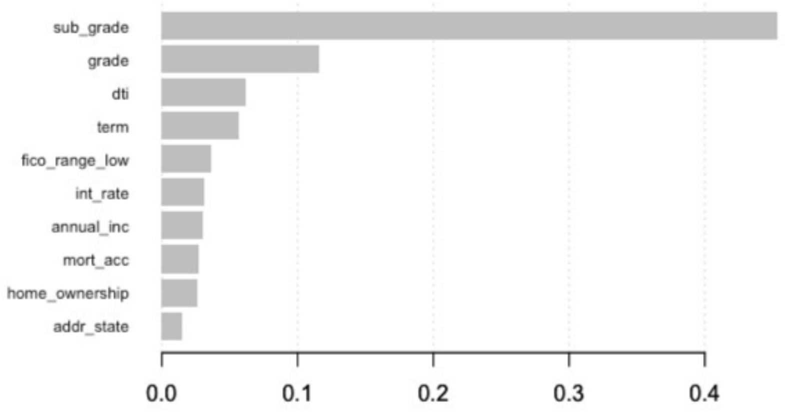
The figure2 indicates that sub_grade is the most important feature and grade follows. Then, we go back to the tree structure in xgboost and call `xgb.dump()` with `xgb.plot.tree()` to explore the trees. We find leaf 7 in tree 2 and leaf 8 in tree 8 contrain high percentage(>0.5) bad loan status obervation. They contrain features `sub_grade`, `term`, `mort_acc` and `dti`. Besides, in average, we find that the leaf from worse `sub_grade`, shorter `term`, more `mort_acc` and larger `dti` tend to classify observation to bad loan status. Therefore, it is reasonable that top 10 features take that important role.

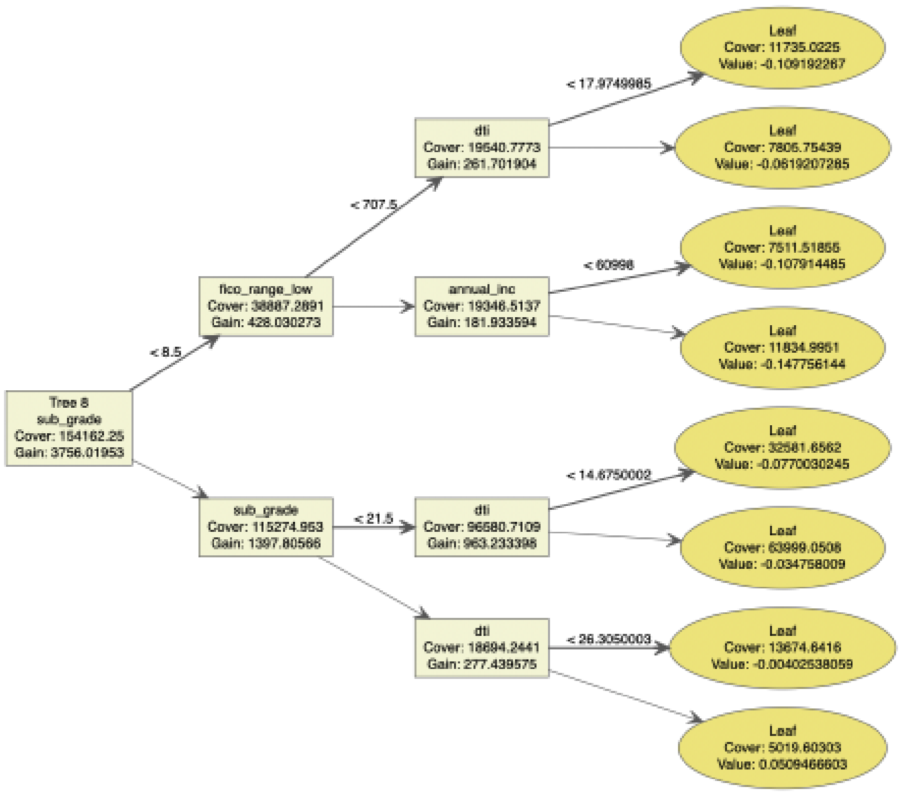
For observation whose `id=140330058` in data set `LoanStats_2018Q3.csv`, features `sub_grade=35(G5)`, `term=2(60 months)`, `mort_acc=0` and `dti=39.07`, which means this loan has high loan subgrade assigned by LendingClub, many mortgage accounts, much monthly debt or low monthly income and long payment term. All these values satisfy the cutoff in tree8 above, which indicates it has high probability to default the loan, so the predcition result our model is 1.

For observation whose `id=139802285` in data set `LoanStats_2018Q3.csv`, features `sub_grade=11.71(A5)`, `term=1(36 months)`, `mort_acc=2` and `dti=11.71`, which means this loan has low loan subgrade assigned by LendingClub, few mortgage accounts, little monthly debt or high monthly income and short payment term. This indicates it has low probability to default the loan, so the predcition result our model is 0.

For observation whose `id=142844535` in data set `LoanStats_2018Q4.csv`, features `sub_grade=8(E5)`, `term=1(60 months)`, `mort_acc=6` and `dti=6.72`, which means this loan has high loan subgrade assigned by LendingClub, many mortgage accounts, much monthly debt or low monthly income and long payment term. All these values satisfy the cutoff in tree8 above, which indicates it has high probability to default the loan, so the predcition result our model is 1.

For observation whose `id=145558098` in data set `LoanStats_2018Q4.csv`, features `sub_grade=11.71(B3)`, `term=1(36 months)`, `mort_acc=2` and `dti=11.71`, which means this loan has low loan subgrade assigned by LendingClub, few mortgage accounts, little monthly debt or high monthly income and short payment term. This indicates it has low probability to default the loan, so the predcition result our model is 0.





Conclusion

We applied machine learning methods to predict the probability that a loan status on LendingClub will turn out Default or Change Off. After training and evaluating two different models (Xgboost, Logistic Regression with Lasso), we find that Xgboost performs better than Logistic with shorter running time and better accuracy. Therefore, we use Xgboost method to build a model to predcit loan data in 2018Q3 and 2018Q4. The Log Loss function value of prediction is both small which are around 0.3 or 0.2, meaning our model has high accuracy in probability prediction. This model, while far from perfect, can provide a somewhat informed prediction of the probability that a loan will charge off or default, using only data available to potential investors before the loan is fully funded.

We also found that, according to the multi-tree structure of our model, the most important variables for predicting Default or Change Off are the subgrade from LendingClub, the loan term, number of mortgage accounts and the borrower's total monthly debt payments on the total debt obligations, excluding mortgage and the requested LC loan, divided by the borrower's self-reported monthly income.