Credit Default Prediction

Kiyoon Jeong, Yankun Xu, Minjie Dong

April 10, 2018

Abstract

In order to investigate a well-performed statistical model for credit default detection, the article compares traditional classification methods such as logistic regression, linear discriminant analysis (LDA), K-nearest neighbors (KNN), Support vector machine (SVM), Random forest, XGBoost, and LightGBM. LightGBM outperforms the rest methods, with a relatively low misclassification error rate, strong sensitivity and competitive computation time. The article also introduces stacking model, which integrates the advantages of each classification model. Consequently, the result of stacking model performs better than that of any traditional model.

Keywords: credit default, classification methods, sensitivity, stacking model

1 Introduction

Credit default is always a serious issue for banks. In order to detect the default beforehand, many banks collect their clients' information and use them to determine which clients are credible. Therefore, a feasible and credible statistical model becomes a cornerstone for banks' decision. In the circumstance, the article demonstrates traditional classification methods' performances by comparing misclassification errors. Considering each method's pro and con, it also introduces stacking model to predict credit default in a comprehensive way.

The article investigates the credit default in Taiwan from April 2005 to September 2005. The model study in the article consists of four main aspects: preprocessing, variable selection, traditional classification methods, and stacking model. In the process of variable selection, it implements subset selection, LASSO, and PCA, with the results implying involve all variables. Classification methods, e.g. logistic regression, LDA, KNN, SVM, Random forest, XGBoost, and LightGBM, are gradually applied to model. The performance measurements for each model are misclassification error rate, sensitivity, and computation time. When it comes to stacking model, it takes advantages of different traditional models. In the process, all the meta features from traditional models above tend to fit in a logistic model, to predict credit default.

In the following, the article demonstrates data description, methodology and result, and conclusion.

2 Data Description

This default dataset is collected in Taiwan from April 2005 to September 2005, posted on UCI Machine Learning Repository. It provides clients' basic information, such as sex and education, history of payment, bill statement, and credit data (limit balance). Default payment (Yes = 1, No = 0) is the response variable. Detail of data is displayed in Appendix.

3 Methodology and Result

3.1 Preprocess

According to data description, both level 5 and 6 of variable 'EDUCATION' 'represent unknown. In order to avoid ambiguity, level 5 represents both unknown education background.

Some of demographic factors (e.g. 'MARRIAGE', and 'EDUCATION') are categorical variables, in order to fit machine learning models(like 'KNN'), they are processed to be dummy varaibles by one-hot encoding. The value for binary variable like 'SEX' is maintained since it would not

have much influence on the final result. Another issue is that 'PAY' variables are also categorical variables, however, they have relationship between the scale -1 to 9 (-1 means pay duly, and 9 means payment delay more than 9 months). Thus, they are maintained as numerical variables with the assumption that each scale has linear increase.

In order to avoid post selection bias, data structure is 30% data for variable selection, 40% for training set and 30% for test set.

```
Stepwise Model Path
Analysis of Deviance Table

Initial Model:
dpnm ~ LIMIT_BAL + SEX + AGE + PAY_0 + PAY_2 + PAY_3 + PAY_4 +
    PAY_5 + PAY_6 + BILL_AMT1 + BILL_AMT2 + BILL_AMT3 + BILL_AMT4 +
    BILL_AMT5 + BILL_AMT6 + PAY_AMT1 + PAY_AMT2 + PAY_AMT3 +
    PAY_AMT4 + PAY_AMT5 + PAY_AMT6 + EDUCATION_0 + EDUCATION_1 +
    EDUCATION_2 + EDUCATION_3 + EDUCATION_4 + MARRIAGE_0 + MARRIAGE_1 +
    MARRIAGE_2

Final Model:
dpnm ~ LIMIT_BAL + SEX + AGE + PAY_0 + PAY_2 + PAY_3 + PAY_4 +
    PAY_5 + PAY_6 + BILL_AMT1 + BILL_AMT2 + BILL_AMT3 + BILL_AMT4 +
    BILL_AMT5 + BILL_AMT6 + PAY_AMT1 + PAY_AMT2 + PAY_AMT3 +
    PAY_AMT4 + PAY_AMT5 + PAY_AMT6 + EDUCATION_0 + EDUCATION_1 +
    EDUCATION_2 + EDUCATION_3 + EDUCATION_4 + MARRIAGE_0 + MARRIAGE_1 +
    MARRIAGE_2


  Step Df Deviance Resid. Df Resid. Dev      AIC
1                      8970   8274.919 8334.919
```

(a) Forward

```
Stepwise Model Path
Analysis of Deviance Table

Initial Model:
dpnm ~ LIMIT_BAL + SEX + AGE + PAY_0 + PAY_2 + PAY_3 + PAY_4 +
    PAY_5 + PAY_6 + BILL_AMT1 + BILL_AMT2 + BILL_AMT3 + BILL_AMT4 +
    BILL_AMT5 + BILL_AMT6 + PAY_AMT1 + PAY_AMT2 + PAY_AMT3 +
    PAY_AMT4 + PAY_AMT5 + PAY_AMT6 + EDUCATION_0 + EDUCATION_1 +
    EDUCATION_2 + EDUCATION_3 + EDUCATION_4 + MARRIAGE_0 + MARRIAGE_1 +
    MARRIAGE_2

Final Model:
dpnm ~ LIMIT_BAL + SEX + AGE + PAY_0 + PAY_3 + PAY_4 + BILL_AMT1 +
    BILL_AMT3 + BILL_AMT6 + PAY_AMT1 + PAY_AMT2 + PAY_AMT4 +
    PAY_AMT5 + EDUCATION_1 + EDUCATION_2 + EDUCATION_3 + MARRIAGE_2


           Step Df   Deviance Resid. Df Resid. Dev      AIC
1                                  8970   8274.919 8334.919
2     - PAY_AMT3  1 0.01103788      8971   8274.930 8332.930
3  - EDUCATION_4  1 0.05746051      8972   8274.987 8330.987
4   - MARRIAGE_1  1 0.10317931      8973   8275.091 8329.091
5        - PAY_5  1 0.17339118      8974   8275.264 8327.264
6        - PAY_6  1 0.18371669      8975   8275.448 8325.448
7    - BILL_AMT5  1 0.31243807      8976   8275.760 8323.760
8    - BILL_AMT4  1 0.11966891      8977   8275.880 8321.880
9  - EDUCATION_0  1 0.49547245      8978   8276.375 8320.375
10   - BILL_AMT2  1 0.82738751      8979   8277.203 8319.203
11    - PAY_AMT6  1 1.11671725      8980   8278.319 8318.319
12  - MARRIAGE_0  1 1.13038964      8981   8279.450 8317.450
13       - PAY_2  1 1.47989969      8982   8280.930 8316.930
```

(b) Backward

Figure 1: Detail of forward and backward selection

3.2 Variable Selection

Figure 2: PCA

	Eigenvalue	Proportion	Cumulative
dim 1	6.50050603	22.41553804	22.41554
dim 2	4.23225176	14.59397160	37.00951
dim 3	2.41667318	8.33335580	45.34287
dim 4	1.64464037	5.67117370	51.01404
dim 5	1.38957975	4.79165432	55.80569
dim 6	1.22683076	4.23045089	60.03614
dim 7	1.03264921	3.56085935	63.59700
dim 8	1.00846483	3.47746495	67.07447
dim 9	0.99889393	3.44446183	70.51893
dim 10	0.97473657	3.36116058	73.88009
dim 11	0.96212952	3.31768801	77.19778
dim 12	0.90502430	3.12077345	80.31855
dim 13	0.87208202	3.00717938	83.32573
dim 14	0.85192743	2.93768080	86.26341
dim 15	0.75253629	2.59495272	88.85837
dim 16	0.66326113	2.28710733	91.14547
dim 17	0.61515761	2.12123313	93.26671
dim 18	0.51934812	1.79085558	95.05756
dim 19	0.39794800	1.37223447	96.42980
dim 20	0.25906002	0.89331041	97.32311
dim 21	0.25422248	0.87662926	98.19974
dim 22	0.18441682	0.63592006	98.83566
dim 23	0.13094034	0.45151840	99.28717
dim 24	0.06729010	0.23203483	99.51921

There are 29 variables in our dataset. The methods for variable selection can mainly be subset selection, shrinkage, and dimension reduction. We successively apply typical methods in each area to examine which variables are important and select them if possible.

To apply subset selection, a preliminary logistic regression model is established. Best subset selection is not feasible due to more than 30 minutes operating time. The result of forward selection implies that all the variables should be involved in the model. Backward selection decreases the variables to 17 (Figure 1).

LASSO chooses the most of variables except ḂILL_AMT4ánd ḾARRIAGE_1ź. We decide to include them too since the P-values are not significant.

For PCA result, it can explain 90% of variance only when the dimension of PCA is over 16, which does not reduce the dimension at all. As a matter of fact, we continue to use 29 basic variables as our input(Figure2).

3.3 Traditional Classification Method

In our dataset, the default rate is only 0.22, which means it possess possibility of trivial case. The trivial case means classifying all observations as non-default. To prevent and deal with this case, we use different thresholds(from 0.5 and 0.1).

3.3.1 Logistic Regression, LDA, KNN

The results of logistic regression and LDA are pretty similar with misclassification error around 0.188 and sensitivity around 0.27. Since the assumption of LDA that observations are drawn from Gaussian distribution may be violated because some categorical variables are involved in the model, it may result in outperformance of logistic regression over LDA. The application of QDA failed,

with the model's error implying of existence of high collinearity. It may be attributed to the same violation against assumption as LDA.

In order to set value for k in KNN model, we applied 5-fold cross validation to training set with k from 1 to 100, and the optimal k reached to 89. The error rate under KNN model (k = 89) is around 0.2243 and sensitivity is around 0.03, with longer computation time than previous two methods.

3.3.2 SVM and Random Forest

Compared to the previous methods, SVM and Random Forest's error rate is less than 0.18. There were several issues in these models. For SVM, the tuning function use the 10-fold cross validation to search the best tuning factors, the running process is time consuming and keeping failed. Thus, we set the several lambda values 0.01, 0.1, 0.5, 1, 5, 10, degrees 2,3,4 (polynomial), and gamma values 0.1, 0.5, 1, 5, 10 (radial) by using training and test set instead of cross validation. When the model is linear and the lambda cost was 5, it gives best result.

For random forest, we fit the model with default values first. The plot of this model shows that even if the ntree is more than 300, there is no big changes. So, we set ntree as 300 and try the several mtree values (number of variables available for splitting at each tree node). As a result, the default value (square root of the number of predictors = 6) gives us the minimum error rate, which is 0.178.

3.3.3 XGBoost and LightGBM

XGBoost is short term for "Extreme Gradient Boosting" (Friedman, 2001). It is a tool motivated by the boosting tree with both deep consideration in terms of systems optimization and principles in machine learning. The goal of this library is to push the extreme of the computation limits of machines to provide a scalable, portable and accurate library.

During the parameter adjustment process, there are two candidate metrics, one is 'error' which stands for misclassification error rate and the other one is 'auc' which is for sensitivity evaluation. After fitting the model, by looking into Table 1, model performs much better in term of sensitivity when metrics is set to be 'auc'.

	Misclassification	Sensitivity
Error	0.1767	0.5397
AUC	0.1773	0.5501

Table 1: Performance Table

LightGBM model is the last meta model. LightGBM is a boosting method developed by Microsoft. Unlike many boosting tools using pre-sorted based algorithms for decision tree learning, which have simple solutions but not easy to optimize, LightGBM uses the histogram based algorithms, which bucketing continuous feature (attribute) values into discrete bins, to speed up training procedure and reduce memory usage.

Compared to XGBoost, the biggest advantage of LightGBM is computation time, which takes only 6.09 seconds to complete the process. Considering its high efficiency and relatively low misclassification error and strong sensitivity, it is safe to say that LightGBM is the best model among all traditional models.

3.4 Stacking Model

Stacking (also called meta ensembling) is a model ensembling technique used to combine information from multiple predictive models to generate a new model. It highlights each base model where it performs best and discredit each base model where it performs poorly.

There are two layers in stacking model. In the first layer, the training set is divided into 5 folds randomly. Then a model is fitted on 4 folds of training set, here model is chosen from traditional classification methods (Logistic, LDA, KNN, SVM, Random Forest, XGBoost and LightGBM) which are mentioned in the previous session. After fitting model, the left one fold of training

set and testset are used to do prediction. The prediction results are named meta1 and test1 individually. Then several models have been implemented in the traditional models part. The same process is repeated for the other four folds of training set and stack them together to form the training meta feature, the only difference is for testset is after getting all the prediction result, the average is used to form the testing meta feature.

In the second layer, final model is fitted on all the meta features. In this article logistic model is used as the final model. Figure 4 is the process of stacking model.

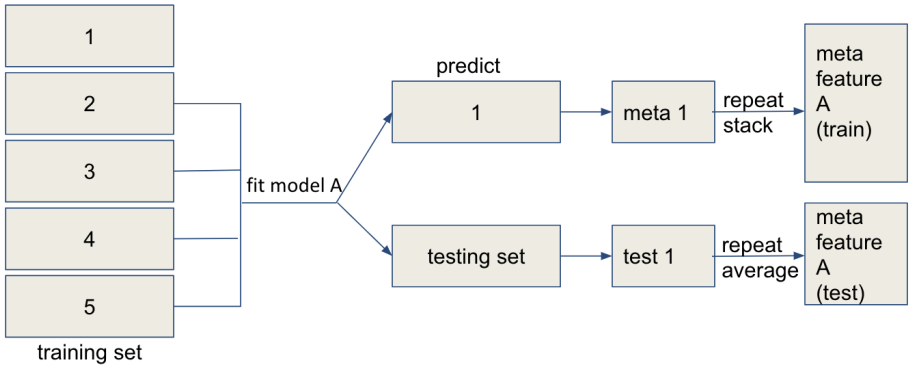


Figure 3: Process of stacking

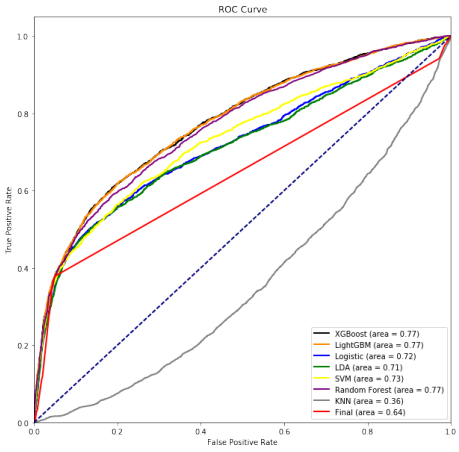
4 Conclusion

By looking into the results of both traditional models and stacking model, there are three criterons to evaluate: misclassification error rate, sensitivity, and computation time. In Table 2, final stacking model, lightGBM and XGBoost are the best choices in terms of error rate. XGBoost, lightGBM and final stacking model are the best for sensitivity evaluation. LDA comes first when it comes to computation time.

KNN performs terrible in both error rate and sensitivity, the main reason here is KNN uses distance as measurement, however since all the categorical variables are treated as dummy variables by one-hot encoding, which means the relation between different level is assumed to be either 0 or 1. This relation may not always be true in real life.

Meta Feature	Coefficient	Meta Feature	Coefficient
LGB	0.638612	Random Forest	-0.011443
Logistic	0.461583	SVM	-0.025302
LDA	-0.52347	XGB	1.833813
KNN	-0.030646		

(a) Coefficients of Stacking model



(b) ROC

Figure 4: Stacking Model Performance

Which one is the best?			
Method	Error Rate	Sensitivity	Computation Time
Logistic Regression	0.1882	0.2613	3.47 sec
LDA	0.1884	0.2733	**1.58 sec**
KNN	0.2243	0.0314	49.47 sec
SVM	0.1799	0.3910	127.96 sec
Random Forest	0.1788	0.4653	103.39 sec
XGBoost	0.1773	**0.5501**	11.8 sec
LightGBM	**0.1770**	0.5476	6.09 sec
Final (Logistic)	**0.1767**	**0.5352**	0.132 sec

Table 2: Performance Table

In Figure 3(a), the coefficients of different variables are displayed. XGB and lightGBM have the biggest coefficients which means they play the most important roles in the final stacking model.

Finally, Figure 3(b) is the ROC curve of different methods. Besides KNN, all methods seem to have a pretty solid performance. However, the final stacking model performs worse than any other traditional method, the phenomenon can be explained by trade-off. Except XGBoost using 'auc' as metrics, other methods all use misclassification error rate as metrics to build models. By combining them together, the final stacking model has the smallest error rate but relatively bad performance in terms of sensitivity.

Reference

Friedman, J. H. (2001). Greedy function approximation: a gradient boosting machine. Annals of statistics, 1189-1232.

Appendix

Data Description in detail

- Following 23 variables as explanatory variables:

- X1: Amount of the given credit (NT dollar): it includes both the individual consumer credit and his/her family (supplementary) credit.
- X2: Gender (1 = male; 2 = female).
- X3: Education (1 = graduate school; 2 = university; 3 = high school; 4 = others).
- X4: Marital status (1 = married; 2 = single; 3 = others).
- X5: Age (year).
- X6 - X11: History of past payment. We tracked the past monthly payment records (from April to September, 2005) as follows: X6 = the repayment status in September, 2005; X7 = the repayment status in August, 2005; . . .;X11 = the repayment status in April, 2005. The measurement scale for the repayment status is: -1 = pay duly; 1 = payment delay for one month; 2 = payment delay for two months; . . .; 8 = payment delay for eight months; 9 = payment delay for nine months and above.
- X12-X17: Amount of bill statement (NT dollar). X12 = amount of bill statement in September, 2005; X13 = amount of bill statement in August, 2005; . . .; X17 = amount of bill statement in April, 2005.
- X18-X23: Amount of previous payment (NT dollar). X18 = amount paid in September, 2005; X19 = amount paid in August, 2005; . . .;X23 = amount paid in April, 2005.