Survey on several methods of reject inference

Yusheng, Ni
2016110731
714366596@qq.com

Yang, Xia
2015120263
317654943@qq.com

December 3, 2018

Abstract

1 Background issues and research significance

In financial credit scenario, the lending institution filters the applicants by model scoring. Users with better scoring are approved of getting the loan, while those with poor scoring are directly rejected. Thus, lending institutions can only get loan records of applicants with good scoring and repayments of the rejected applicants are not available which results in the problem of sample bias. With the trend moving on, training samples in the hands of the organization are all “scores better” through the applicants, but there is no “poorly rated” rejection applicants, and the trained model focus more and more on the “scoring better” applicants. However it can’t get any verification in the "poorly rated" users,which will cause sample bias.This bias is even more pronounced in online lending, where over 90% of total loan requests are rejected.

The traditional way to build credit score card for applicants is to use Logistic Regression because of its interpretability and there have been attempts to deal with the sample bias problem including the traditional heuristic reject inference methods[4][5],semi-supervised learning related methods[1][6]. The core idea of these methods is to make full use of information of both the accepted and rejected applicants(namely both labelled and unlabelled data) to improve the performance of scoring models on new applicants which contains both ’rejected’ ones and ’accepted’ ones. Fuzzy Augmentation[4] and Parcelling[4] are two methods which intends to assign labels to the ’rejected applicants’ according to credit score(the probability to be ’bad’ applicants) of by the current classifier. Re-classification and Re-weighting intend to use the information that the applicants will first go through another classifier(called) to be assigned ’rejected’ or ’accepted’ and try to train such a classifier from the data in hand. Then use the output probability of being ’rejected’/’accepted’ to re-weight the samples or assign ’good/bad’ labels to a small amount of ’rejected’ applicants. Semisupervised SVM add two constraints to the traditional SVM formulation for each point in the working set. One constraint calculates the misclassification error as if the point were in positive class and the other constraint calculates the misclassification error as if the point were in negative class. The objective function calculates the minimum of the two possible misclassification errors.

However Logistic Regression may not perform well when the data is extremely dirty and when the data is high dimensional, it may lose the interpretability as well. There also have been empirical studies[7] which deny the supremacy of the reject inference methods and suggest the sample bias is not as serious as people worry. And for semisupervised SVM, solving the problem requires solving an integer programming which is too expansive for large scale data.

In our case, we conduct different methods for dealing with the problem of sample bias, examining the performance improvement based on distinctive methods, including Parcelling, Fuzzy Augmentation, SMOTE, oversampling and evaluating the feasibility on the specific problem. We take high dimensional features into consideration and seek for methods for dealing with them since these can encumber the robustness of classification accuracy. We made comparison between Lasso, Adaptive Lasso with SIS and feature filtering based on feature scores given by XGBoost[8]. As for the model compiling, we figure out a specific method which can improve the classification robustness, where XGBoost is chosen as our basic algorithm for its considerable interpretability and excellent computational efficiency.

2 Approaches

Step 1 —— Establishing Baseline:

Use the simplest Logistic Regression to fit the labelled data(accepted applicants). This model constitutes our benchmark, against which the performance of other models will be compared.

Step 2 —— Feature Engineering:

At the stage of data preprocessing, we take focus on the missing value for each sample and remove samples with over 2000 missing values for decreasing the noise of data which account for about 0.01. Missing values of data are taken as features and put into model for training.

Considering the feature dimension is ultra high(with 6784 different features) and most credit cards need only hundreds of features, we adopt some variable selection and dimension reduction techniques involving the SIS[2]+Adaptive Lassos[3] for generalized linear models and variable selection by feature importance given by boosting algorithm to reduce over-fitting and improve model interpretability.

The reason why we choose these method is as follows: The SIS method guarantees that with probability one the features selected will contain the true model and the Adaptive Lasso is proved to have the oracle property. Besides, use feature importance output by boosting algorithm is a well-known technique in competitions like Kaggle.

Step 3 – Model Fine Tuning

Step 4 – Model ensemble:

Model ensemble is claimed to lower the variance and improve performance. So we try three method : Averaging, Bagging and Stacking

Step 5 – Reject Inference Method:

Parcelling and fuzzy augmentation are traditional reject interference methods for credit scoring problem, which using the known application's results to interfere the unknown performance of the rejects needs to be inferred. Reject

inference is a form of missing values treatment where the outcomes are "missing not at random", but are heuristic methods. We conduct them to examine the performance on realistic problems There are two broad approaches used to infer the missing values: assignment and augmentation. We choose two augmentation techniques to examine the performance: fuzzy augmentation and parcelling.

Fuzzy augmentation assumes scoring of the rejects using the base model. Each record is effectively duplicated contains weighted "bad" and weighted "good" components. The weights with the weights equal to "1" for all the accepts, are use in the final model.

Parcelling: Parcelling is a hybrid method. The method first trains a base model and bin known samples into different bins and uses the trained model to predict unknown samples and classify them into different bins according to their probability scores. In each bin, there are specific distribution for "0,1" labels according to the empirical distribution based on known sample bins. Those unknown samples are tagged with "0,1" labels and put together with the known samples. Finally, a new model is trained based on the new dataset.

Step 6 – Error Analysis:

After getting the best model, we carefully inspect the mistakes by our model. One thing to do is descriptive statistics on the misclassified data which aims to find whether these samples bear some resemblance in their features and get intuition for a better feature engineering. Another thing to do is to carefully consider the confusion matrix which indicates what is weakness of our model and whether the model complexity is too large or too small.

3 Experiment

3.1 Data Set Description

The data set we use comes from the Rong360 competition.

It contains totally 100,000 samples and 6784 features. 33,000 samples are labelled and the rest are unlabelled. Among the labelled ones, nearly 2200 belongs to positive class and the rest belongs to negative.

The main criterion to evaluate models is AUC value on online validate set(with unknown labels).

To write this report,We take 20% of the labelled as our test set and the rest for training and validation. This part of data contains 7786 negative samples and 581 positive samples.

3.2 Results

Throughout the experiment, the data preprocessing is very simple: using mean value to fill the missing value(when needed) and using minmax scaler to do standardization. In the first step we implement the simplest LogisticRegression to establish a baseline for comparison.(The Logistic Regression function we use here is 'sklearn.linear_models.LogisticRegression' with parameter 'class weight' set to 'balanced' and other parameter deafault.) The result is shown in the first row in Table 1. As it can be seen, the result is not very good.

After variable selection, total 2179 features survived in the end. We use the selected features to re-run the same Logistic Regression to get the result in

the second row in Table 1.

It can be seen that there is big leap in the result. The AUC and F1 score increase about 30% and 33% respectively. And there is also a little improvement in the Recall score.The Precision score is almost unchanged, which means it's not easy for Logistic Regression to predict the positive samples very well.

We also use xgboost for comparison. We omit the paramter tuning process and only report the best result here. Also, since xgboost has the ability to deal with missing values itself(recoginzed as np.nan), we skip the step of filling the missing values.

One thing worth talking about is that the feature importance output by xgboost of variables selected by SIS+Adaptive lasso is not very high. So we use all features to train Xgboost instead. As the result in the third row in Table 1 shows,the performance of Xgboost is much better than we originally think. Then we use model ensemble method to see whether it can be further improved. The best result comes from averaging. We only report the best result of the three ensemble method (bagging,stacking,averaging)since their difference is very tiny. The result in the forth row in Table 1 shows ensemble result. There is exactly a nice improvement.

We have also done several other things such as cutting values into bins, counting the number of missing values as feature of applicants and etc. However these tricks didn't bring visible improvements.

	Accuracy	Precision	Recall	F1 score	AUC score
LR	0.7248	0.11	0.5	0.1922	0.688
LR(2179 variables)	0.767	0.170	0.61	0.2666	0.777
Xgboost	0.932	0.57	0.080	0.141	0.814
Xgboost(ensemble)	0.940	0.89	0.158	0.269	0.843

Table 1: Results

After applying Parcelling and Fuzzy Augmentation, we get the following result as Table 2 shows. The result is not as good as we expect. The possible underlying reason may be that these two method both use the information learned by the classifier from the labelled data to assign labels to the unlabelled data. So this action would probably cause overfitting on the orginal data set.

	Accuracy	Precision	Recall	F1 score	AUC score
Fuzzy Augmentation	0.8	0.49	0.108	0.177	0.813
Parcelling	0.767	0.55	0.133	0.214	0.778

Table 2: Results

The result of Rong360 competition is shown in Table 3. The model we submit is the ensembled xgboost.After conducting model ensembling, we have results in a 0.002 improvement in auc score on the online validation set. With all the process above, we made 0.11 improvement compared with the benchmark given by Logistic Regression in online auc score.

	online validation set	online test set
AUC score	0.8317	0.8171
Ranking	None	25

Table 3: Online Results

4 Conclusion

From table 1 we can easily see that xgboost performs better than LR in terms of Accuracy and AUC. However, the Precision and Recall of these two model is totally different. The Recall score of LR is higher than xgboost, and the precision of xgboost is higher than recall. In credit score scenario, it's more important to identify those 'bad' applicants. The recall score is therefore more important.Besides, the high accuracy of xgboost is a sign of overfitting although we have added penalty term in the training process. So we can't conclude that xgboost is better than LR because the recall score of xgboost is extremely low. Maybe with more delicate feature engineering, LR has more potiential than xgboost.

5 Discussion and future improvements

There are two things we want to try but we couldn't yet find a delicate way to apply them to this task to get a higher online AUC score. One is the semisupervised algorithms and the other is the techniques of feature engineering such as feature crosses.

In our experiment,a difficulty that haven't been well solved is the large amount of false negative error. Our guess is this problem is caused by the sample imbalance so we tried several method such as oversampling and scaled loss function to control the false positive error. These methods work,but the price is the sharply increasing number of the false negative error. It needs to be further studied whether our model can be optimized with respect to this problem and if not how to make this tradeoff. Because in real world, this scenario is cost sensitive,i.e. the false positive error will lead to far more loss to company than the the false negative error.

Although xgboost with no preprocessing of the data perform best in this task, we still have a question whether we can use low dimensional features to get nearly the same result because 6874 such a high dimension is totally of no interpretability and in real world especially in the credit score scenario, model interpretabiliy is important. Credit institution usually won't allow a 'black box' to decided whether to give a loan to a person and 6874 features is far too much for portraiting a person.

References

[1] Bennett, K. P., & Demiriz, A. (1999). Semi-supervised support vector machines. In Advances in Neural Information processing systems (pp. 368-374).

[2] Fan J, Lv J. Sure independence screening for ultrahigh dimensional feature space[J]. Journal of the Royal Statistical Society: Series B (Statistical Methodology), 2008, 70(5): 849-911.

[3] Zou H. The adaptive lasso and its oracle properties[J]. Journal of the American statistical association, 2006, 101(476): 1418-1429.

[4] Hand D J, Henley W E. Can reject inference ever work?[J]. IMA Journal of Management Mathematics, 1993, 5(1): 45-55.

[5] Joanes D N. Reject inference applied to logistic regression for credit scoring[J]. IMA Journal of Management Mathematics, 1993, 5(1): 35-43.

[6] Maldonado S, Paredes G. A semi-supervised approach for reject inference in credit scoring using SVMs[C]//Industrial Conference on Data Mining. Springer, Berlin, Heidelberg, 2010: 558-571.

[7] Crook J, Banasik J. Does reject inference really improve the performance of application scoring models?[J]. Journal of Banking & Finance, 2004, 28(4): 857-874.

[8] Chen T, Guestrin C. Xgboost: A scalable tree boosting system[C]//Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining. ACM, 2016: 785-794.

[9] Banasik J, Crook J. Reject inference, augmentation, and sample selection[J]. European Journal of Operational Research, 2007, 183(3): 1582-1594.

6 Appendix

Hardware:
CPU: Intel(R) Core(TM) i5-6300HQ @2.30
Memory: Kingston DDR3 1600 8GB
No GPU sopport
Main Environment:
Python 3.71
sklearn 0.20
xgboost 0.80
numpy 1.15.3
pandas 0.23.4

The code file is organized as follows:

(1) LR base.py is the baseline model

(2) LR after Variable selection.py is the LR model after variable selection

(3) SIS.py is used to do SIS

(4) Apative Lasso.rar is Adaptive lasso package downloaded from github

(5) xgboost.py is used to train xgboost and do model ensemble

(6) .m files are models trained to do ensemble

(7) parcelling and fuzzy.py is used for reject inference methods.