Improvement of Classification in Credit Default

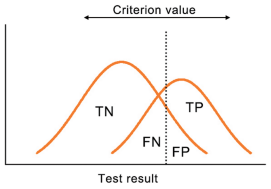
Yuehua Duan, Yunfei Xia, Hongyuan Yang, Ali Mahzarnia
University of North Carolina at Charlotte

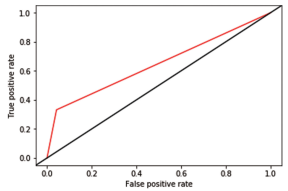
Abstract

In this project, we are interested in addressing this issue by predicting if a new customer should be granted credit card or not based on the information of the past customer whose credit card has or has not become default. That's means we need to improve the model, which has much higher accuracy rate. Before doing this project, we do the surveys and find the Random Forest is the best machine learning model when giving prediction with the customer default problem. At the same time, we find a new method of classification, which is NP-Umbrella algorithm. We try to compare it with Random Forest model, then based on the better model to do variable selection to improve the accuracy rate and get the best model to predict the customer default problem.

Experiment NP Random Forest

	Model gives the best overall accuracy	Overall accuracy
Baseline model	Random Forest	0.707467
NP-Umbrella-Version1	NP-Logistic-Version1	0.8216
NP-Umbrella-Version2	NP-Random Forest Version2	**0.8233**





Reference

[1] Xiong, Tengke, et al. "Personal bankruptcy prediction by mining credit card data.".
[2] Liang, Deron, et al. "A novel classifier ensemble approach for financial distress prediction." .
[3] Lu, Hongya, Haifeng Wang, and Sang Won Yoon. "Real Time Credit Card Default Classification Using Adaptive Boosting-Based Online Learning Algorithm."
[4] Islam, S. R., Eberle, W., & Ghafoor, S. K. (2018). Credit Default Mining Using Combined Machine Learning and Heuristic Approach.
[5] Yeh, I-Cheng, and Che-hui Lien. "The comparisons of data mining techniques for the predictive accuracy of probability of default of credit card clients."
[6] Tong, X., Feng, Y., & Li, J. J. (2018). Neyman-Pearson classification algorithms and NP receiver operating characteristics. Science advances.
[7] Tong, X, Xia, L Wang, J (2018). Neyman-Pearson classification: parametrics and power enhancement.

Feature Selection

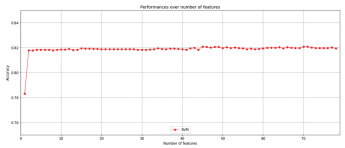
Stepwise method is to marry the feature selection process to the type of model being built, evaluating feature subsets in order to detect the model performance between features, and subsequently select the best performing subset. It attempts to optimize feature selection process for a given machine learning algorithm.

In our project, the stepwise feature selection is based on Recursive Feature Elimination (RFE), we know that RFE is to select features by recursively considering smaller and smaller sets of features.

Feature selection for SVM

After data preprocessing, we have 78 features in total, we rank the feature importance with different sets of features and compare the prediction accuracy of SVM based on picking only one feature to picking all the 78 features.

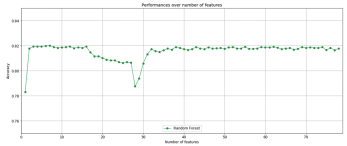
From the experiment result, we can see that when picking 45 or 70 features, SVM gives the best prediction accuracy.



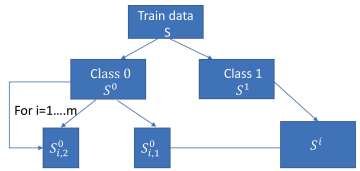
Feature selection for Random Forest

After data preprocessing, we have 78 features in total, we rank the feature importance with different sets of features and compare the prediction accuracy of Random Forest model based on picking only one feature to picking all the 78 features.

From the experiment result, we can see that when picking 7 features, Random Forest gives the best prediction accuracy.



NP Umbrella



Algorithm: An NP-Umbrella algorithm

1. **Input**:
   Training data: A mixed i.i.d. sample $S = S^0 \cup S^1$, where S^0 and S^1 are class 0 and 1 samples, respectively
   α: Type I error upper bound, $0 \le \alpha \le 1$; (default α = 0.05)
   δ: A small tolerance level, $0 \le \delta \le 1$; (default δ = 0.05)
   M: Number of random splits on S^0 ; (default M = 1)
2. **Function** Rank (n, α, δ)

3. **For** k in $\{1,\dots,n\}$ **do**	◂ For each rank threshold candidate k
4. $V(k) \leftarrow \sum_{j=k}^{n} C_j^n (1-\alpha)^j \alpha^{n-j}$	◂ Calculate the violation rate upper bound
5. $k^* \leftarrow \min\{k \in \{1,\dots,n\} : V(k) \le \delta\}$	◂ Pick the rank threshold
6. **Return** k^*	
7. **Procedure** NP $(Sample, \alpha, \delta, M)$	
8. $n = \lvert S^0 \rvert / 2$	◂ Denote half of the size of |S0|as n
9. $k^* \leftarrow$ Rank (n, α, δ)	◂ Find the rank threshold
10. **For** i in $\{1,\dots,M\}$ **do**	◂ Randomly split S0 for M times
11. $S^0_{L1}, S^0_{L2} \leftarrow$ random split on S^0	◂ Each time randomly split S0 into two halves with equal sizes
12. $S_i \leftarrow S^0_{L1} \cup S^1$	◂ Combine S^0_{L1} and S^1
13. $S^0_{L2} = \{x_1,\dots,x_n\}$	◂ Write S^0_{L2} as a set of n data points
14. $f_i \leftarrow$ ClassificationAlgorithm(S_i)	◂ Train a scoring function f_i on S_i
15. $\tau_i = \{t_{0,1},\dots,t_{i,n}\} \leftarrow \{f_i(x_1),\dots,f_i(x_n)\}$	◂ Apply the scoring function f_i to S^0_{L2} to obtain a set of score threshold candidates
16. $\{t_{i,(1)},\dots,t_{i,(n)}\} \leftarrow \text{sort}(\tau)$	◂ Sort elements of τ_i in an increasing order
17. $t_i^* \leftarrow t_{i,(k^*)}$	◂ Find the score threshold corresponding to the rank threshold k^*
18. $\Phi_i(X) = I(f_i(X) > t_i^*)$	◂ Construct an NP classifier based on the scoring function f_i and the threshold t_i^*
19. **Output**: an ensemble NP classifier $\hat{\Phi}_n(X) = I(\frac{1}{M}\sum_{i=1}^{M} \Phi_i(X) \ge \frac{1}{2})$	◂ By majority vote