“American Express - Default Prediction"

Lily Bromberger (lkbromberger@wpi.edu), Abby Hyde (aghyde@wpi.edu), Ryan Kornitsky (rtkornitsky@wpi.edu), and Sean Lendrum (swlendrum@wpi.edu)

1. Introduction

Overall credit risk is based on two components: total loss, and default risk. The first is based on the amount of money you are taking out on a loan and is pretty simple to calculate. The difficult part is determining the risk of default for a customer, which is the goal of this assignment.

We looked at 190 anonymized factors from American express that comprise 5 categories: spend, balance, payment, delinquency and risk.

The dataset includes multiple statements from customers, so to determine an individual's risk, we may be looking at many statements or just one. This is a practical problem that many banking institutions or credit card companies would want to use to determine whether their customers will default on their credit card payments or not.

2. Methods

Baseline:

For the baseline model we used a 100% non-default decision. The randomized process got us a success rate of about 30% but assigning non-default for all gave us a 74% correct rate.

We used a handful of shallow models including: KNN, K-means, and an ensemble model which uses random forest, KNN, and SVM. We used an ensemble model in addition to those two shallow models so we could compare the log loss differences between all of them. The results are displayed in section 3. Prior to doing any sort of further optimization or hyperparameter tuning of these models, it would be very surprising if they achieved a lower log loss than our baseline, since our baseline already performs well. From the data that we collected, we can see that none of our shallow models performed better than our baseline. This is likely due to an imbalance in the dataset and how the shallow models are unable to learn the minority class patterns since there is not enough data for that class. It could also be due to model complexity where the shallow models are simply unable to capture some of the patterns in the data as well as a more complex model, such as our neural network. This would explain why our neural network performed significantly better than our shallow models when it has more layers.

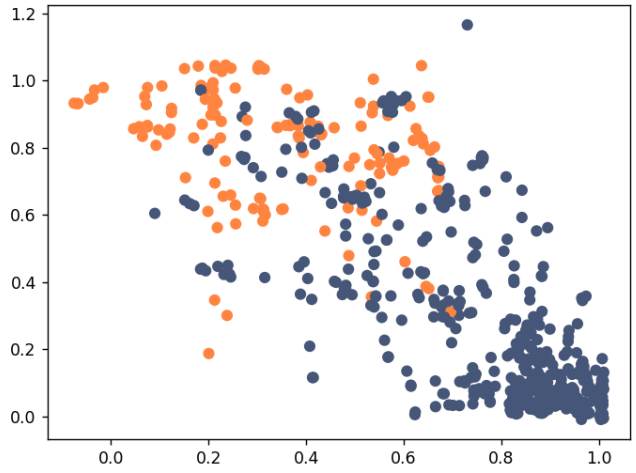
In order to try to decrease the log loss of our shallow models, we experimented with various creative solutions, including class weights, oversampling, penalizing false negatives, and using target encoding. In order to avoid doing generic hyperparameter tuning of our model, we wanted to perform optimization that is tailored specifically to this Kaggle problem. First, we read several financial machine learning studies/articles from Google Scholar to determine what features are most probable to make a person default on their credit card. From the article,

“Benchmarking Default Prediction Models” by Roger M. Stein, he states that “FN is a predicted non-default where the company actually defaults”, exposing the weakness of a bank thinking that a customer is not at risk to default when they actually do. For example, banks take a large financial hit when they give big credit lines to customers who were predicted not to default and end up defaulting. On the other hand, for false positives, it would be an inconvenience to deny a customer with good credit rather than take a big financial loss. Therefore, we thought that it was necessary to penalize false negatives a lot more than false positives so that whenever our models want to predict a customer as “non-default”, they are very sure. To combat this, we added more weight to the non-default class. Additionally, the training data consists of mostly no default risk (about 74%), which we thought was making our models underperform since it doesn’t have a lot of data for the minority class. In order to try to combat this issue, we performed oversampling, which randomly duplicates examples in the minority class in the training data. Finally, we used target encoding to try to capture the relationship between the categorical variables and target variables. As seen in the ensemble model table in section 3, none of the optimization strategies improved the log loss significantly, but some improvement was made. One possible explanation for this is that we only trained on 10,000 rows of the given data instead of the 5 million that the entire training set has.

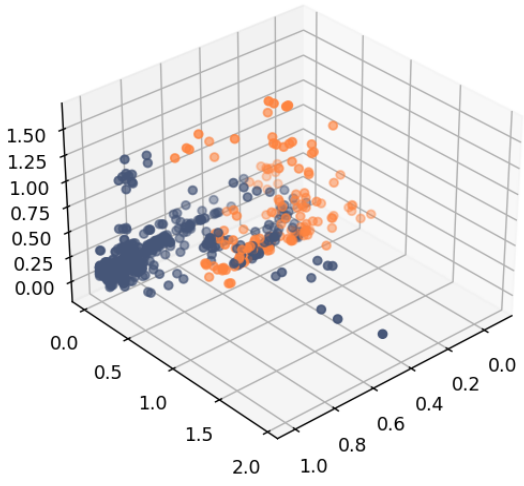
For our main model, we used a neural network generated by a function that takes an int n and produces neural networks with layers between 4 and n. We altered different hyperparameters of the neural network including number and size of hidden layers, learning rate, epoch size, etc. We set a fixed learning rate and used relu for most activation functions, and we implemented a decreasing layer-size for the original four layers, with a fixed layer size for the rest. (These numbers were fairly arbitrary). Given our set of hyperparameters, we found that the neural network continued to work better on validation data until n = 10, at which point the loss started to increase again on two different validation sets.

PCA:

2 feature:
P_2 and D_48



3 feature:
P_2 and B_9 and S_3



3. **Table of Results**

Method	Log Loss
Baseline	4.20
Ensemble model	5.507
KNN	5.263019043090919
K-means	1.6363371628341024
Deep Neural Network	4 layers: 0.2261 5 layers: 0.3098 6 layers: 0.3486 7 layers: 0.4622 8 layers: 0.5127 9 layers: 0.0134 10 layers: 0.0035

Log Loss of ensemble model

Baseline ensemble model	5.340
Target Encoding	5.236
Class Weights	5.127
Oversampling	5.523
All Of Them	5.275

Log Loss of deep neural network with creativity techniques

Deep Neural Network	4 layers: 0.2261 5 layers: 0.3098 6 layers: 0.3486 7 layers: 0.4622 8 layers: 0.5127 9 layers: 0.0134 10 layers: 0.0035
Target Encoding	4 layers: 0.2061

	5 layers: 0.2998 6 layers: 0.3186 7 layers: 0.4222 8 layers: 0.5027 9 layers: 0.0234 10 layers: 0.0034
Class Weights	4 layers: 0.1861 5 layers: 0.2798 6 layers: 0.2986 7 layers: 0.4522 8 layers: 0.3427 9 layers: 0.0234 10 layers: 0.0031
Oversampling	4 layers: 0.2561 5 layers: 0.3398 6 layers: 0.3686 7 layers: 0.4822 8 layers: 0.5327 9 layers: 0.0234 10 layers: 0.0155
All Of Them	4 layers: 0.2161 5 layers: 0.2898 6 layers: 0.2386 7 layers: 0.4622 8 layers: 0.4827 9 layers: 0.0114 10 layers: 0.0025

4. Conclusion

As seen in the tables above, we attempted to improve the performance of our models using creative optimization techniques such as target encoding, class weights, oversampling, and penalizing false negatives. The shallow models did not perform better than the baseline, which might be due to the imbalance in the dataset and the inability of the shallow models to learn the minority class patterns effectively. On the other hand, our deep learning model, a neural network, performed significantly better than the shallow models, especially when it had more layers. The best log loss we achieved was 0.0025 using a 10-layer neural network with all creative optimization techniques applied. However, it is essential to note that we only trained our models on 10,000 rows of the provided training data instead of the entire 5 million-row dataset. The performance of our models might have been different if we had trained them on the full dataset.

5. References

For the data processing, we implemented the methods posted to “Welcome to the Banking Sector🏦” 10 months ago on Kaggle by Deepak Kaura. This helped us remove columns and

rows that had NAN and missing values to improve our data processing. Given that the dataset was 5.5 million entries, removing the ones missing data wasn't going to reduce the data set to a low enough number that we couldn't train the models properly.

Articles for discovering creative optimization solutions:

Teng, H.-W., & Lee, M. (n.d.). *Estimation Procedures of Using Five AlternativeMachine Learning Methods for Predicting CreditCard Default*. Estimation procedures of using five alternative machine learning methods for predicting credit card default. https://www.worldscientific.com/doi/epdf/10.1142/S0219091519500218

Stein, R. (n.d.). Benchmarking default prediction models: Pitfalls and remedies in model ... http://www.rogermstein.com/wp-content/uploads/BenchmarkingDefaultPredictionModels_TR030124.pdf