American Express Default Prediction

Benjamin Shaw
College of Science
Utah State University
Logan, UT, United States
ben.shaw@usu.edu

***Abstract*—An overview of an approach to working through a particular Kaggle competition is given. In total, six submissions are made to the competition, with varying levels of success.**

***Index Terms*—Random Forests, Neural Networks, Classification**

I. Introduction

This paper is a summary of an approach to the problem of predicting default probabilities for American Express Customers. The problem has been posed as a Kaggle competition [1]. The Kaggle competition was made available in May of 2022, and the submission deadline was in August of 2022. Thus, the submissions made in this project were made after the deadline. In fact, cash prizes and full-time employment were among the rewards for scoring well in the competition: it is thus regrettable that the timeline for the competition was not better aligned with the timeline for this paper.

The evaluation metric M is calculated as follows:

$$M = 0.5(G + D), \tag{1}$$

where G is the normalized Gini coefficient, and D is the default rate captured at 4%: D is the percentage of the positive labels (defaults) captured withing the highest-ranked 4% of the predictions, representing a Sensitivity/Recall statistic. For G and D, the negative labels are given a weight of 20 to adjust for "downsampling." The metric M has a maximum value of 1.0.

II. Overview of Data and Preprocessing

The data is offered by means of three .csv files, the first containing the training data, the second containing the training labels, and the third containing the test data (no labels). Each of the training and test datapoints have 190 features, including the "Customer ID," the statement date, and two categorical features, whose values are strings. The meaning of each feature is not shared with the competition participants.

The training data consists of over 458,000 rows, and the test data consists of over 924,000 rows. As a complication, the number of statements per customer varies between one and 13: that is, the same customer ID can be found on a number of rows between 1 and 13, inclusive.

The goal of the competition is to make predictions for a particular customer: that is, per customer ID. Therefore, as a first preprocessing step, the original dataset was split into many .csv files, with a .csv file for each customer ID. This was completed using the "groupby" method for pandas dataframes. Understandably, this step is very time consuming for the amount of data, taking several days to complete. Time taken by this step is perhaps exacerbated by grouping by customer ID, the values of which are long strings. In any case, the split was complete, with over 80,000 .csv files in the new training file location.

When data is loaded for a particular customer, we ignore the (generate) information in the customer ID column. We also ignore the statement date column, as well as the two categorical columns. This leaves us with 186 features for every row. Some entries are missing values: for the purposes of this project, the missing values were set to 0.0.

III. Early models

The very first submission to the competition was merely a "dummy submission," consisting of prediction probabilities of 0 for each customer. While the accuracy score reflected that of the majority class, the private score using (1) was a mere 0.017. For reference, the evaluation score of the benchmark random forest model is approximately 0.75. The highest score in the competition is approximately 0.809.

The next set of early models analyzed only the last statement for each customer. The first such model was a Random forest model implemented using sklearn [2]. At first, the default arguments were used–in particular, the number of trees was set to 100. Applying a training/validation split of 0.33–that is, reserving roughly $1/3$ of the data for validation, this model earned an accuracy score of 88%, and an evaluation score of 0.768. When the number of trees was increased to 1000, the accuracy was boosted to 89.5% and the evaluation score rose to 0.774. With the large amount of data, it was particularly helpful to use the sklearn implementation, which allowed for simultaneous building of trees.

The next early model was a support vector machine, again implemented with sklearn [2]. The chosen kernel was a radial basis function. The support vector machine does not seem amenable to parallel computation, as was the case with the random forest algorithm: thus, the SVM classifier took a few days to train. At the completion of training, the validation accuracy was found to be 87.7%, and the evaluation score was found to be 0.71. Due to the lengthy nature of the training of the support vector machine, and due to the apparent out-of-the-box underperforming when compared to the random forest approach, variants of our SVM were not explored.

The next model was a fully connected neural network. For this approach, the labels were 1-hot encoded. The architecture of the neural network was as follows. After the input layer, a layer of size 400 followed, after which a layer of size 100 followed, and the output had size 2. The final output was a softmax activation function, but the first two activation functions were ReLU activation functions. The proportion of validation data was 0.2. The Adam optimizer was used along with the negative log likelihood loss, and the learning rate was chosen to be 0.001. The model was trained for 1000 epochs. A plot of the loss function values as a function of the number of epochs is depicted in Fig. 1.

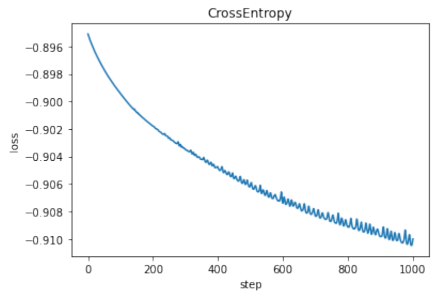

Fig. 1. A plot of the loss function values as a function of the number of epochs for the simple, fully connected neural network.

The validation accuracy was found to be 89.7%, which is only slightly higher than the validation accuracy obtained using random forests. However, the evaluation metric for this model was a disappointing 0.698. At first glance, this discrepancy is quite unaccountable. However, we direct our attention to the output predictions of this model. The outputs are very close to either 0 or 1, perhaps representative of overconfidence in the model. The previous algorithms may have outperformed the neural network simply due to the more moderate probability predictions. This leads us to believe that the model complexity may be too high, or that the model was over-trained.

IV. The Failed LSTM

The next model attempted to make use of an LSTM. The motivation behind this attempt was the notion that the data is sequential in nature, at least on a per-customer basis. When examining the data corresponding to a single customer, we see a sequence of data, approximately evenly spaced in time (one statement every month): a sequential model is believed to be appropriate.

The data was prepared in the following manner. For a customer with 13 statements, a sequence of length 13 was constructed, with each item in the sequence coming from a row of data–in order. For customers with $n < 13$ statements, an in-order sequence of length n was constructed. Subsequently, the sequence was padded with rows of zeros, bringing the length of the sequence to 13. Having prepared the data, great hope was had for the success of the LSTM.

However, once a simple LSTM model was deployed, the optimization of the parameters was unsuccessful: the loss function value remained almost unchanging throughout the duration of the training. The architecture was adjusted so that the LSTM layer was followed by fully connected layers, and yet this availed us but little. Due to the lack of success in training this model, this model was not used to make predictions on the test dataset and thus did not result in a competition submission. Although it is quite possible that more success could have been obtained using an LSTM, the constraints of the project dictated that more methods be applied, so that, for the sake of time, the LSTM model was abandoned.

V. Later models

Having made submissions based on the predictions of four models–the “dummy submission,” (constant prediction of 0), the random forest model, the SVM, and the simple neural network–attention was turned to other models. Specifically, the question became how to use the training data in its entirety, rather than just using the last statement data. The attempt to use the LSTM had failed, and while other sequential models were tempting to pursue, the project deadline loomed, and more familiar models were pursued.

As a given pair of customers may have an unequal amount of data, it was thought that a model consisting of 13 different classifiers would have success. The rows of the customer data could be appended, giving a single row of length $186n$, where n is the number of statements. A classifier could then be trained for each statement size, and a model evaluation of a datapoint would be evaluated by one of the 13 different classifiers, depending on the size of the datapoint. A total of two models were constructed in conjunction with this thought process.

In the first model, the 13 classifiers were chosen to be random forest classifiers, owing not only to the early success of the random forest model, but also to the fact that the random forest model is amenable to parallel training, thus promising to deliver results in a timely fashion (several hours). The number of trees for each classifier was chosen to be $100n$, where n is the number of statements.

It was at this point that it was discovered that a commanding majority of the customers had precisely 13 statements associated to their customer ID’s. The number of customers with n statements was about $3,000 - 5,000$ for each n with $1 \leq n \leq 12$, while the rest of the customers had precisely 13 statements.

The validation accuracy of the 13 Random forests model was approximately 89%, while the evaluation score was 0.759, notably less than the first random forest model. Interestingly, the validation accuracies for the customers with n statements for $1 \leq n \leq 11$ was approximately 80%, the validation accuracy for the customers with 12 statements was approximately 84%, and the validation accuracy for customers with 13 statements was approximately 90%.

The final model for this project employed 13 different neural networks. The architectures for the networks consisted of fully connected layers of the following sizes, including the input and output layers: $186, 186n, 186n/2, 186n/4$, and 2, where division in this case is integer division, and where n is (again) the number of statements. As with the earlier neural network model, the activation functions were ReLU, ReLU, and softmax, respectively. The same loss function and optimizer were used for each neural network, which optimizers and loss functions matched those of the earlier neural network model. Fig. 2 depicts the graphs of the loss function for each network as a function of the number of epochs.

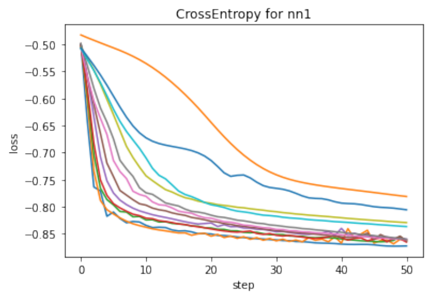

Fig. 2. A plot of loss function values as functions of the number of epochs for the 13 NN model.

Interestingly, and as a general rule, lower loss function values were obtained as n increased. Applying lessons from the previous neural network architecture, the networks were trained for fewer epochs: 50, in fact. This approach resulted in an accuracy of 89% and an evaluation score of 0.72, an improvement over the previous neural network architecture. However, the accuracy is no better than that of the 13 random forests model, and the evaluation score is worse. Upon inspection, the class probability predictions of the 13 Neural Network model are typically close to either 0 or 1, as was the case with the early neural network model. Thus, it is reasonable to conclude that the reason for the under-performance with respect to the evaluation metric is the overconfidence in the class predictions.

VI. Reflection and Future Work

This project, it seems, was no simple walk in the park. Due to the sheer volume of data, any data handling steps were time consuming. Greater transparency as to the meaning of each of the features may have aided the analysis: some features may have been less important than others. Of course feature importance could also have been the subject of study in this project; however, given the need to deploy several successful models, and due to the large amount of time needed to accomplish anything with this dataset, time was the inhibiting constraint.

On the subject of more that could be done with the project given time is the deployment of additional machine learning models. In this project, neural networks and random forest-based models were preferred for computational reasons: neural networks and the random forest algorithm are amenable to parallel computation. However, it would have been interesting to try training with a Gradient Boosting Machine, for example.

At the very least, it would be interesting to try variations of the SVM, possibly using alternate kernel functions. The amount of time needed to train an SVM was too great to make this possible. In fact, on that note, the need to deploy several models made any fine tuning of any models particularly difficult, as even the random forest and neural network models took a surprisingly large amount of computational time.

Also of interest is the failure to break the 90% validation accuracy barrier. The metric for this competition is not accuracy, however it would be of interest to explore other models in pursuit of obtaining a higher accuracy score.

It was also interesting to note that the accuracies and the evaluation scores did not tend to improve as the models increased in complexity. A simple random forest, trained and evaluated only on the last statement data, was the best performing model of this project in terms of the evaluation metric, and was on par with the best accuracies obtained.

In light of this observation, as well as the failure to break the 90% validation accuracy barrier, it is possible that all of the models are overfitting, and that a simpler model should be deployed, notwithstanding the enormity of data. In fact, it is quite possible that the data features themselves are quite degenerate, so that a simpler model applied to significantly fewer features would merit more success.

Then again, it is conceivable that the opposite approach is needed: that an approach such as the 13 neural network model is revisited so that each of the 13 classifiers are made to be more robust models. Perhaps the 90% barrier could be broken, and perhaps a better evaluation score would follow.

Perhaps the most disappointing aspect of this project was the inability to get a sequential-based classifier off the ground. Intuitively, it seems that the sequential aspect of this problem could lend itself to divulging more information: if a feature corresponds to a customer's credit card balance, and that balance suddenly increases sharply, the probability of a payment default may increase. Conversely, a decrease in the credit card balance may decrease the probability of a payment default. Therefore, a sequential model remains of interest. Besides an LSTM, other neural network-based architectures exist, such as transformers and, perhaps, pretrained models. Perhaps methods from time series data analysis could be of use, specifically methods for analyzing multivariate time series.

Lastly, work should be done so as to subdue the overconfidence of the probability predictions of the neural network models. Perhaps this can be addressed with regularization, or perhaps the use of an alternate loss function. Without addressing this issue, a neural network model will struggle to obtain convincing evaluation scores.

In conclusion, it is believed that more could be done to improve upon the results of this project, notwithstanding the success of the relatively simple models deployed. In particular, the deployment of sequential models is believed to have the potential to outperform the models given in this project. It is also believed that neural networks have greater potential, which potential can be unlocked by addressing the over-confidence in the class probability predictions. It may also be necessary, in order to improve computational speed, to apply training on only a subset of the data or to apply training on a machine with a large amount of RAM which is also GPU-equipped.

References

[1] A. Express, "American express - default prediction," May 2022. [Online]. Available: https://www.kaggle.com/competitions/amex-default-prediction/

[2] F. Pedregosa *et al.*, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.