Name: Caleb Johnson
UTEID: cdj2273
Kaggle UserName: calebJ
Public Leaderboard: Score: .883365 Pos: 47/57
Private Leaderboard: Score: .87643 Pos: 46/57

Overview

As someone with no prior experience with data science, this competition was a wonderful learning experience. I tried many things that didn't work well, read a great amount of documentation and forum opinions on what problems different models are effective at solving and how best to implement those models. In this report, I will explain what I worked on with regards to pre-processing, model choice and parameterization choices, and post-processing. While I did not do everything in the most efficient order, I will make this document flow in a logical order from pre-proc to post-proc for the sake of clarity.

The 3 most successful models I trained during this competition were a neural net, a random forest, and an XG Boost model. In each section, I will focus on those 3 models to explain my process.

Major Mistake

I was evaluating my models using roc_auc_score but only submitting 1s and 0s rather than the probabilities roc_auc_score needs. This led me down some pre-processing steps that were probably unnecessary, but I am glad I made this mistake, because I was forced to manipulate the data set in creative ways out of desperation (*I couldn't validate higher than high 60s*).

Pre-Processing and Feature Engineering

- The first thing I did was generate a correlation matrix for my features. I didn't find that any features were strongly correlated, so I decided not to combine any features.
- Y=1 for about 95% of the training set. I struggled early on to get my model to do anything other than predict all 1's (*see Major Mistake Section).* To remedy this, I made a training set containing every 0 I had in the training set and an equivalent number of 1s – a 50/50 training set. This boosted my score significantly on the validation set and on the Kaggle test set. I went from validating in mid 60s to high 70s. My Kaggle score jumped from high 50s to mid 60s.
- I got another significant bump in score by removing the features which correlated the least with Y. I used my Random Forest model to score each feature on how correlated it was with Y. I noticed that the features with the lowest scores seemed to be categorical, as they took specific integer values within a small range. I tried one-hot-encoding these

features, but it made my feature space too large and bogged down my model. I found that removing these features made my model the strongest. My Kaggle score went from mid-60s to low-70s from this.

- Next I realized that I had been erroneously and manually turning my model outputs to 1s and 0s before submitting them. Submitting the raw outputs of my Random Forest and XG Boost models made my score jump from low 70s to low 80s.
- The RF feature scores revealed that one feature, **f14**, was MUCH more correlated than any other feature (*over 10x more correlated*). I tried making extra features out of this feature but did not have any success. I tried a square, sqrt, log, and cube feature.
- I noticed that when f14 was either 0 or 1, Y was the same values 100% of the time. This allowed me to grab about 3000 more samples from the test set and add them to the training set by manually adding the labels. I was not sure if this was good practice, but it seemed reasonable. Training on these expanded set gave me meaningful gains in Kaggle, taking me from mid 80s to high 80s.

Model Selection and Parameterization

- **Neural Net**
  A neural net probably seems like a silly place to start on this problem, but I have a strong GPU in my desktop at home, and I wanted to learn more about neural nets, so it's where I started. Most of my gains while I worked on this model came from feature engineering and training set manipulation. I did find that a NN with 2 hidden layers consistently outperformed ones with 1 hidden layer for this problem. This may suggest a relatively complicated underlying function.

  I found that a hidden layer size of around 80% of the feature space performed the best. I used relu activations on my hidden layers, and a sigmoid activation on my output layer. I found that a larger batch size was best to reduce training time. When a particularly strong set of parameters was found, the batch size could be turned far down to give a slight boost in model performance.

  Overall it was fun playing with the neural net and my GPU for a day, but this did not seem to be a good model for the problem. It was meaningfully outperformed by both the Random Forest and XG Boost.

- **Random Forest**
  While this model was slightly outperformed by XG Boost, it was invaluable, as it gave me my list of feature importance scores. Using those scores to remove excess features gave me one of my biggest jumps in score on Kaggle.

I used RandomizedSearchCV to parameterize my Random Forest. While I believe it was tuned well, I think I wasted too much time by allowing my SearchCV to iterate hundreds of time. I probably could have been more efficient with less iterations and more manual tuning.

- **XG Boost**
  This was my best-performing model by a very slight margin over the Random Forest. It seemed to respond to changes in features and training data in the same ways as the Random Forest. I found the colsample_bytree and learning_rate parameters to be the most sensitive. Small changes in them could lead to relatively large differences in score. Due to this, I allowed my RandomizedSearchCV to iterate for quite a long time on this model as well. While using the suggested parameters yielded my best results, I still feel that my time could have been better spent than waiting for my model to parameterize for nearly an hour at a time.

Post-Processing

- Originally I was manually changing all my model outputs (probabilities) to binary. While I was eventually training in the 70s by submitting only 1s and 0s, this post-processing step was obviously counter-productive and I got a big jump when I submitted in the correct format
- I got my final bump from mid 80s to high 80s by averaging my Random Forest and XG Boost models together and submitting them.

Takeaways

- I spent most of my weekend learning and playing around. I eventually got a respectable score, but I left myself with little time during the week to do more work on the model because I had to catch up in other classes. I believe this hurt my overall ranking in the competition, as people starting late were able to get advice from peers that had already started. I went from #22 to #47 within a day or so after school started on Monday. I should have continued fine-tuning my model until the end of the competition, as there was very little difference across the leaderboard, and incremental increases in score could result in large jumps in ranking.
- This is one of the best projects I have done at UT. I enjoyed the fact that there is not necessarily a right and wrong answer. It removed the pressure of finding the exact right answer and allowed to us just play around and explore the data. I found this very enlightening.