Name: Wenyang Fu

Kaggle Username: Zephyr

Public LB: 0.86714 (rank 9) ¶

Private LB: 0.85837 (rank 8)

What did I try?

Correlated Features

I began by looking at the features and seeing if there was any interesting information. I looked at the number of unique values for each feature, and tried to see if there were any highly correlated features. It seemed like F2, F14, and F25 were highly correlated with each other, so I decided to drop F2 and F25 (since they had less categories), thinking that this would help with the prediction. Unfortunately, this made my predictions worse by around .03 (measured via area under ROC) during cross-validation, so I kept F2 and F25.

Missing Value Imputation

I also looked for missing values, and I noticed that F6 and F19 both contained a number of missing values. After visualizing the amount of missing data in F6 and F19, I decided to impute the missing values via a mean() approach. For F19 (with a tail-heavy distribution), perhaps a most frequent value imputation would've worked better, but I wasn't sure how to apply two separate strategies to the dataset using scikit-learn's Imputer() class.

Feature Scaling, Normalization

I decided to skip feature scaling and normalization, since tree-based models should be invariant to any form of normalization.

Model Fitting

Broadly speaking, I tried the following techniques:

- Random Forests
- XGBoost
- hyperopt for hyperparameter selection
- Logistic Regression w/ and w/out feature scaling
- MLP w/ feature scaling
- Voting Ensemble of Classifiers over RF, XGB, and MLP

- Deep Pyramidal MLP on Keras

I decided to go with tree-based models and Logistic Regression. I used the 90% of the training data as the training set and 10% as the holdout set. I then performed 5-fold cross-validation on the 90% of the training data. I used this approach for my first few models, before switching to pure 5-fold cross validation on the training data, and simply trusting my cross-validation scores for the best models. I aggressively optimized hyperparameters for my base XGBoost and Random Forest models using hyperopt (https://github.com/hyperopt/hyperopt), a Bayesian hyperparameter optimization framework.

After picking the "best" hyperparameter combos out of 250 rounds of sampling, I created a bagging meta-estimator containing 20 XGBoost models and created a Voting Ensemble (utilizing soft probability voting), which performed better than the default.

What didn't work

Unfortunately, Logistic regression with no feature scaling performed very poorly, sitting at only 0.68. Deep Pyramidal MLP on Keras did slightly worse than the sklearn MLP, sitting at around 0.815. Sklearn MLP performed around .820. Adding the MLP to my Voting Ensemble (which feature scaling) dragged my score down, so I omitted the MLP.

What I wish I had the time to do:

I would've liked to try Stacking my models together and run logistic regression on new features generated from pairs/triplets of existing features. I would've also liked to investigate alternate neural net architectures that would work better on this dataset. I think stacking my models would've improved my score for me to place in the top 5.

In [ ]: