Matthew Zlotnik

Mcz334

Business Data Science

10/31/2019

Kaggle Competition Writeup

This Kaggle competition was unquestionably the most unique and effective method of testing/knowledge checking I have seen in my entire time as a student. I thoroughly enjoyed evaluating different models and conducting exploratory data analyses and data engineering. With that being said, I was relatively unsuccessful in deriving meaningful models for this data set, as I finished in last place with a .85 AUROC.

I focused much of my analysis on analyzing each of the independent variables and trying to ensure that I derived as much value as possible. In my initial analysis, I checked each X variable to see if any null values existed in the data, and if there were any collinearity issues. With every variable passing both of these tests, I dove into each variable to ensure I was properly using the variables in the most predictive and powerful method possible. For example, feature ‘f2’ is classified in the data as a continuous variable, yet it exists only in discrete values 1-7. As such, I created dummy variables for these values. However, the percentage of Y=0 values (the more rare class), was effectively the same in each dummy variable. I conducted analyses similar to this, including log-transforming data, and classifying data into entirely new data based on distributions of Y=0 within each X variable. In doing these analyses, I increased the AUROC of my model fairly significantly. So, even though the models ultimately were not effective, I would classify my feature engineering and extraction as an ultimately successful process.

The modeling process was far less successful. Initially, I fit ridge and lasso regression models, hoping that they could help with feature reduction in addition to prediction. However, when implementing these models, I got my lowest public scores on the Kaggle site, as evidenced by my first submission. Despite their ultimate failures, the predictions from these ridge and lasso models would be used again. Next, I fit a logistic regression model with gridsearchcv to test various hyperparameters and solver methods. Ultimately, my logistic regression models were unsuccessful in creating meaningful predictions, though these predictions would also be utilized again later. Subsequently, I tried to fit an elementary random forest model to the data, using a manually created cross-validation to test for depth and width of the forest. As I ultimately elected to use XGBoost with gridsearchcv, I knew that it was extremely unlikely that the random forest model would be more powerful, so I scrapped it before even submitting it to the Kaggle. Finally, I used this aforementioned combination of techniques to generate my most accurate predictions. My initial model created by these tactics was my most effective, which used only my engineered features to predict the response class based on a combination of XGBoost and gridsearchcv using online research to choose the hyperparameter ranges from which to test.

Next, I tried to stack this XGBoost model on top of the other models I had trained. To do this, I added the predictions generated by logistic, lasso, and ridge regressions as features to be used by the XGBoost. Ultimately, this change increased my test AUC and performance on the public Kaggle leaderboard, but did not produce my most effective model on the private data on Kaggle. I did try to do one more application of XGBoost and gridsearchcv using many more options of parameters, featuring L1 and L2 regularization in the model. I thought that giving this model six hours to run would be enough. However, this was based on an extremely quick and uneducated estimation which drew from the times required to fit previous models. At the end of the six hour allotted period, the model had not finished fitting the data, and I was out of time. This error was partially caused by a lack of forethought and a misunderstanding of the order of calculation time needed by the gridsearchcv method, but ultimately cost me a large chunk of possible extra model-fitting time. I believe that my models suffered because I did not allow them to be robust enough in estimating hyperparameters, which kept my model AUROCs in the range of .80-.85. In conclusion, I think my lack of time towards the end of the submission window prevented me from properly validating the hyperparameters of the XGBoost model, which kept my AUC scores lower than the rest of the class.