Ömer Faruk Akgün

54076

ENGR421 – Homework 7

I began by doing some explanatory work on data. Got the number of NAs, looked at summary statistics. 2 important observations:

1) Many NA data, we need to do some preliminary work to prepare the data
2) High multicollinearity in data. Three options to overcome: We can use some feature selection based on a correlation or entropy-based measure (e.g. Mutual information) We can do feature extraction (PCA brings orthogonal PCs so by nature they are independent) or we can select a training model that does not necessarily be hurt by collinearity.

To overcome first problem, I prepared my data such way. First, I used dummyVars to one-hot encode categorical variables. (dummyVars does automatic one-hot encoding for categoricals and slightly faster than onehot function). Then, I have done some data imputation using MICE package. Since the number of NAs is great, more than 50% for some variables, empirical methods such as mean imputation, random imputation etc. would introduce too much bias into data. Therefore, imputation problem itself can be threated as a ML subproblem. MICE offers, many options, some work for categoricals, some work for continuous values, some work for any variable. The problem is many stochastic approach needs the data matrix to be inverted (which is not an option for us). That's why I proceeded with a tree based approach "cart" (Classification and regression trees).

To overcome second issue, I evaluated two options. Using PCA beforehand to reduce dimensionality and work with independent features or directly working with an algorithm that is less affected by collinearity. I performed PCA over the datasets and the problem with PCA was this. Features with eigenvalues >1 were only able to explain 70% of the variance. (Features with eigenvalues<1 do not have explanatory power more than the initial variables.

Therefore, I proceeded with the second option. The beauty of the forest based algorithms is that they pick features randomly and this means correlated features are less likely to be in the same set in the training and in the optimal classifiers they are less likely to be represented. This is why, multicollinearity is not a huge issue for forests.

I used XGBoost since it is known for its speed and performance in tabular and structured data. Two issues regarding XGBoost: 1) requires all numeric input (which is not a problem for our case) 2) parameter tuning (I tried to overcome this by using a cross-validated grid search)

First, we set up our libraries xgboost for the extreme gradient boosting and caret for training and parameter tuning. One important observation to make is that, our data is high dimensional and we are using a model that inherently overfits. Two important factors that makes overfitting easier. We can try to overcome this with cross validation. Cross-validation is a technique that randomly splits the training data into train and test sets and train a model that works well among each partition. That way we can increase our accuracy without overfitting. I am doing a 5-fold cross validation. A commonly used value in ML applications.

Then I defined my parameter grid, sensible values that also includes the default parameters. I am doing binary logistic regression with evaluation parameter AUROC. (Used auc package) I set the number of rounds to 30 to limit the training time since the algorithm is going to train a model for

each combination in the grid it might take a bit of time. Parameters in the grid are (max_depth, gamma, colsample_bytree, min_child_weight, subsample) Note that I am not testing for eta since and the one that I used is 0.3, which is the default rate. Lower values might increase the auc but I am iterating enough so it should not be a huge problem.

Then I trained all three of my model using hyperparameters found above but I increased the number of folds to 20 and number of rounds to 150. So the naming might be confusing in the code so let me recap that.

1) Doing 5-fold cross validated gird search on set 1
2) Using the hyperparameters found on grid search on set 1, I train my models using higher number of folds (20) and rounds (150)

I predict the values using predict function for training sets and printed their AUROCs. Using the trained models created predictions for the test sets and printed them in csv files. By increasing the number of folds in cross validation and the number of rounds I could get a better AUC. However, that would, increase the training time drastically. This could have been overcame by a) using a better computer b) using a responsive surface design instead of a grid search. Since grid search tries all alternatives.

Again, I am doing a grid search for only set#1. For the other sets I am just using the best parameters in training one. This is a quite naïve assumption (I am thinking since the data sets, objectives and the nature of the problem is the same, it may work). A better way would be doing a grid search for all.