Data Science and Strategic Pricing

Instructor: Jacob LaRiviere, Director of Economics and Data Science at Amazon

Email: jlarivi1@uw.edu, lghhager@uw.edu (TA)

Course Assignments & Reading

Course assignments should be knitted Rmarkdown file and turned in at the start of class unless otherwise noted. Feel free to work in groups but everyone is required to turn in their own work with answers written in your own words. In both calculations and complex ideas, write down each step of logic used in reaching your conclusion. Keep in mind that in most cases a good answer is one precise sentence; quality is heavily favored over quantity. This will be graded on a full credit, half credit and no credit basis. All work must be typed

Discussion questions do not need be written out ahead of time. Students will be called on, potentially at random, to add their insight. This part of class will contribute heavily to your course participation grade.

Week 5, due October 30

Assignment to be turned in. Please turn in your Rmarkdown file (as HTML) with answers embedded.

In this assignment we're going to use a regression tree to break stores into bins or types. This will facilitate pricing differently for each store type. Our target variable will be sales weighted price.

1. We're going to do some basic exploration with `xgboost`.
    a. Install the package `xgboost` and library it.
    b. Divide the data into a training set (80% of the data) and a hold-out set (20% of the data).
    c. We're going to train a model to predict logmove. To do this, we're going to create a training and testing matrix that we can give to the package to do cross validation on.
        i. Use the `xgb.DMatrix` function to create a train and test matrix. This function takes arguments "data" (must be a matrix, so consider using the model.matrix command) and "label" (the outcome, logmove in our case).
        ii. Use the `xgb.cv` function to do 5-fold cross-validation on our training data. We'll just use the defaults for most of the hyperparameters. A few useful arguments:
            1. `nfold`: number of folds for cross-validation
            2. `nrounds`: number of training rounds (generally, we want this to be a very large number since we don't want to be artificially stopped short of achieving a minimum)
            3. `early_stopping_rounds`: if this argument is set, XGBoost will stop training if the testing error does not improve in whatever number the user puts here. This should be our stopping criterion (as opposed to hitting nrounds)
            4. `print_every_n`: if you set this to, say, 100, XGBoost will report its progress every 100 iterations, instead of each iteration.
            5. Important note: we're not actually cross-validating or setting any of the hyperparameters that make XGBoost a powerful algorithm. If

you're curious about what other parameters you can set, inspect the documentation for this function or for the function `xgboost`.

iii. Report the training RMSE (root mean squared error) and testing RMSE from the best model. How does this compare to previous models that we've used (remember that you should square this to get MSE)?

iv. Use the `xgboost` function to train a model on the full training data using our one cross-validated hyperparameter (the number of training iterations). To do this, find the best iteration of the cross validated model and set that as `nrounds` for the `xgboost` function.

v. Use the predict command (the same way that we do in regression) and your testing `xgb.DMatrix` to assess the fit of the model on the held out data. How does the MSE compare to the MSE from cross-validation? How does it compare to prior models?