Ethan Wong (erw956)
Intro to Machine Learning - STA S380

Individual Prediction Project Write Up

For the prediction contest, I decided to use my prior experience as an undergraduate business analytics student at UT Austin and through my time as a teaching assistant for BAX 357: Predictive Analytics with Dr. Mark Tsechansky in order to structure my approach for the contest. First, I will outline the data pre-processing steps I used for both the models requested in the project instructions (from take-home problem question four) and the new models I introduced. Next, I will discuss the process I used to fit and optimize several models for the competition. Finally, I will report the model I ended up selecting to create predictions based on the holdout data.

To prepare my data for modeling, as per the instructions, I first removed the street address and description as predictors from the dataset. After analyzing the data types of each column as read in by pandas, I re-encoded *hasAssociation*, *hasGarage*, *hasSpa*, *hasView*, amd *latest_salemonth*, as category data types to be dummy-encoded and dropped *zipcode*, *homeType*, and *latest_saledate* from the dataset. I dropped *homeType* from the dataset because the entire dataset consisted of only single-family homes, and the *latest_saledate* would only be useful for time series modeling but is otherwise too granular to be used as a predictor. As for *zipcode*, while I believe geographic area could absolutely play a role in dictating *latestPrice*, I did not want to dummy encode every *zipcode* and significantly increase the number of columns in my dataset. So, I decided to aggregate the zipcodes into broader regions in Austin so I could still take advantage of the *zipcode* predictor and replaced this predictor with *zipcode_aggregated*. All of these changes were applied to the holdout dataset as well.

With my data prepared for modeling, I used ten-fold GridSearch cross-validation to evaluate various configurations, select the best set of parameters for several model types (bagging, random forest, k-nearest neighbors, boosting, and lasso/ridge/partial least squares regression) during training and report the out-of-sample mean-squared error (MSE).

Once I had the best parameters for each model type, I fed the models (with their corresponding parameters. with the lower out-of-sample MSEs at this stage into the sequential feature selection (SFS) wrapper method function in scikit-learn. The wrapper method is a more computationally costly feature (predictor) selection methodology as it uses a specific machine learning model to evaluate the effectiveness of subsets of features. As a result, the selected features are specific to the model type being evaluated. Through the SFS function, for each model, I specified the function to automatically determine the ideal numberof features, assess each feature subset by the negative mean-squared error, use ten folds to cross-validate each feature subset, and utilize backward elimination wherein for each step, the least significant feature (the one whose removal causes the least deterioration in the model performance) is removed. The goal of my feature selection process was to select a subset of the available features for each model that could potentially lower the out-of-sample MSE (generalizability) of my model and ultimately, create a simpler but more robust model. I ended up selecting a Gradient Boosting Regressor with a learning rate of 0.1, a maximum depth of five for the individual regression estimators, the square root of the total number of features in the data to consider when looking for the best split, a minimum of two samples required to be at a leaf mode, 200 boosting stages, and to use all samples to fit the individual base learners. This model in conjunction with a subset of the available predictors resulted in an out-of-sample MSE of 25,401.5143. I used this model to create predictions of the *latestPrice* for each record in the holdout data set.