Steve Su, Kristian Montoya

Kaggle King’s County Home Prices

LINK TO GITHUB: <https://github.com/steve303/kaggle01>

**EDA**

Our first step was to examine the data by performing EDA. We plotted the response variable, price, to each predictor variable. We noticed some linear relation with some predictors but not with others. Where we could, we transformed the data to something more meaningful relative to price. For example, the variables, lat and long, themselves are not particularly good predictors themselves but combined together they determine where a house is located and any realtor will tell you location, location, location. Knowing that house prices tend to be highest towards the city center, we combined lat and long to calculate distance from the most expensive neighborhoods in King County. We then plotted this distance against home prices to get a better linear relation compared to lat and long on their own. Correlation coefficients were also computed between predictors where we noticed a few with high correlation coefficients.

Other engineered features include: ‘sqft\_house’, that simply adds up the total square footage of the house being analyzed (living + basement + attic); ‘renovation\_score’, where we try to normalize ‘yr\_renovated’, ‘yr\_built’, ‘bedrooms’, ‘bathrooms’, and ‘floors’ into a normalized dataset; and ‘landvalue’, where we merge ‘view’, ‘waterfront’, ‘condition’, and ‘grade’ arithmetically to create a stronger correlation to said features and the expected price. In the end, the engineered features DID normalize the overall data fairly well (the engineered features had a stronger relationship to ‘price’). However, we believe that our list of engineered features could probably be altered further to create a stronger relation to price, and in turn, reduce the overall MSE of our model.

It is important to note how we originally selected which features to engineer as well. At first, we simply included all the features in our model and evaluated the associated p-scores/feature-correlation to see which features had the strongest relation to ‘price’. From there, we used a for loop to eliminate features from the model one by one, until we were left with a specific number of features or were left with a “low” MSE (we used backwards selection to thin our features down).

Lastly, as part of our EDA we created box plots and threw out some of the outliers from our training data because we didn’t want our model to train on these unlikely observations.

**Linear Regression Model**

Now with our data prepared we split the data, 70% for training and 30% for test. We used multiple linear regression as our technique using libraries from statsmodel. To evaluate the performance of different models we used RMSE. For our first model, we ran all predictors but took out lat and long and replaced with distance (as described above). Our Rsqared value was 0.72 with a RMSE of 199305. With room for improvement, we tried other models. From the EDA, we noticed that our distance predictor seemed to have a squared relationship with prices, so in the model we added a distance squared term which dropped the RMSE by 10000. Smaller gains were made by taking out predictors which had high correlation with another or taking their product and putting it into the model. This only accounted for a drop of RMSE of 2000.

**Random Forest Model**

Similar to our Linear Regression Model, we split our original dataset into training and validation sets to create/test our model with. From there, we ran our models with differing combinations of features to attempt to find the lowest MSE (using backwards selection and manually rearranging a feature list to test performance). Once we were “satisfied” with the MSE generated by our selected feature list, we turned our attention to tuning the model’s hyper parameters to lower our MSE further. We originally started by finding the “optimal” tree depth and number of trees to include in our forest. We started with a max depth of 25 and a total of 25 trees in the forest. From there, we introduced a min leaf constraint, hoping our trees would fit better to the data when more “options/paths” were available in said tree. In the end, however, we decided that pruning our trees was the best approach for optimizing our model. Because we decided to use pruning, we removed the max depth, min leaf, and num trees limit and let pruning do the work for us. We set our ccp\_alpha to 1e9, a small enough “decision boundary” that still captured important decisions throughout the tree. Overall, the average MSE of our random forest was around 180k-190k (not a significant improvement when compared to our linear regression model).

**Summary**

Overall, our Random Forest and Linear Regression models performed similarly to one another; meaning, we don’t think we normalized/altered our original data/features “correctly” to induce better performance from EITHER model. Because they perform the same essentially, our problems (more than likely) lie within the data we’re feeding our models. Perhaps if we took a different approach to feature engineering we would have had a better performing model.