Project 1: Predict the Housing Prices in Ames

xxx xxx (email)

October 19, 2020

Summary of Approach

In this pro udents were asked to use the Ame using Data to buil odels to meet the sp fied rements. A idering mu iple model dologies I settled on using a linear regression model with an Elasticnet penalty and Boosting Tree model. After processing the da and optimizing the model parameters these models we to achieve a minimal SE below t r ements fo 10 test data sets pr d.

Data Processing

The A ing data set is made up o me sales from 2006 to 2010. This data set was used to estimate the sale price of a home given the home's charac cs. The data include 83 columns for each home sale. The response variable used in the models was the sales pr e more data processing that was done the more processing time was required. Ultimately, I found nificant data sing was r for the linear regression, while only the transformation of the response variable ribed below was needed for the Boostin model. On average over the 10 training sets t nths of a seconds were used in processing the data for the linear model while the time processing data for the boosting tree model was so s t was hard to measure. The following processing was nsidered in building the models.

- **Reponse Variable** - xxxx
- **Removal of Variables** - xxx
- **Missing Values** - xxx
- **Winsorizing** - To eliminate e me val several variables were winsorized before being used in the model. Any value that was above the 95th quantile of that v e was set to the 95th quantile. The variables that were winsorized included 'Lot_ e', 'Lot_Area', 'Mas_Vnr_Area', 'BsmtFin_SF_2', 'Bsmt_Unf_SF', 'Total_Bsmt_SF', 'Second_Flr_S First_Fl _porch', 'Screen_Porch', 'Misc_Val'
- **Categorical Variables** - Many variables were not continuouxxiables and thus needed to be adjusted to be considered in the linear xxxession. exx ll factors in the daxxxd created new indicator variables of xx 1 for each level of the factors.
- **Ordinal Variables** - Some of thxx variables mentioned above had a logic order to them, such as overall quali hese variables could be ordered nuxx cally and used in the some time building variables including the nu al ord but t id no antly improve the model, so it was ndoned for the fi al model.
- **Additional Categorical Variables** - I spent so time grouping categorical variables together to see if it would prove the models performance. Th cluded identifying proper s with positive features, i.e. Condition 1 or Condition 2 had P N (near a park) or PosA (adjacen positive feature). Other consider luded gr verall qu and sa ondi s of good or better. O rall these added variables did not add a significant improvement so they were not included in the final model.

Model Build and Optimization

After completing the data analysis I started exploring the models. While some data processing was required such as replacing missing values and creating categorical variables for the linear model, the bulk of the model improvement came from optimizing the model parameters. I focused my efforts on two methods, linear and tree models.

Linear Methods

The first model I tried was a linear Lasso model (alpha = 1), followed by a linear Ridge model (alpha = 0), using glmnet. Neither of these methods met the minimum performance requirements for all 10 test sets. I was able to find a model that met the minimum requirements by using a Lasso (alpha = 1), and then using the significant variables from the lambda.1se for that regression in a Ridge regression. I found it was more straight forward to instead adjust the elasticnet penalty using the alpha parameter to get a good model. The final model used cv.glmnet with an alpha of 0.25.

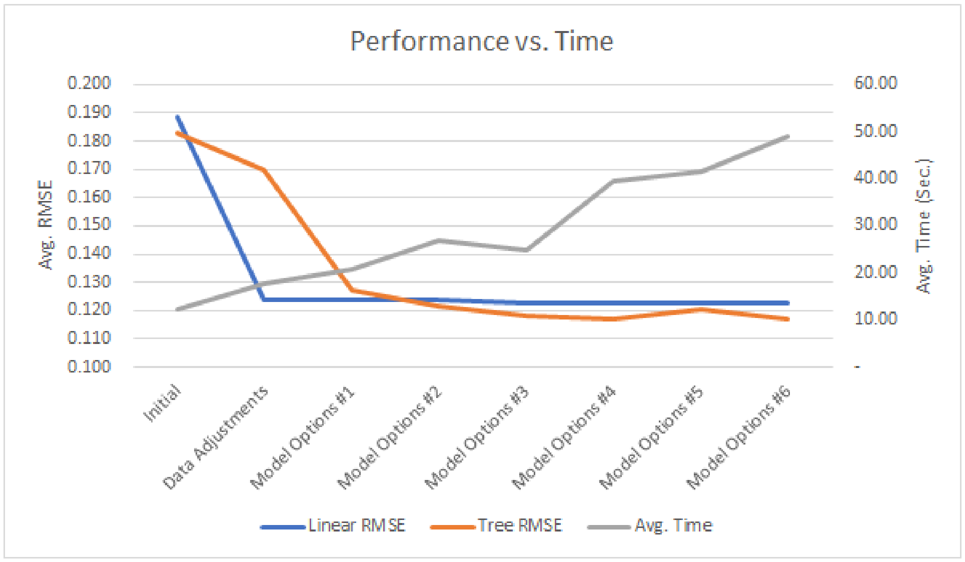
Tree Methods

The second approach was to try fitting a tree model. While the linear model was more explainable, it required a lot of data processing to find an acceptable model. The tree models did not require any data process to start modeling. The only change to the data included logging the sale price variable, which improved the model performance slightly. While time was saved on the data processing the tree models required more parameter adjustments and took longer to run. The final tree model took almost 40 seconds to train compared to about 2 seconds for the linear regression. This added up to about 7 minutes for all 10 training. These models had several parameters that need to be optimized. While I tried a few different tree methods I settled on the boosting tree using the R gbm function and focused on the n.tree, shrinkage, and interaction.depth parameter. I started with a n.tree of 100, shrinkage of 1, and interaction.depth of 3, based on an example provided in the lectures. Running with these parameters only took an average of 15 seconds, but did not meet the minimum model performance requirements. Looking at the iteration plots it was clear more trees were needed. I adjusted this parameter and settled on 1000 for the n.tree. Reducing the shrinkage also reduced the RMSE of the tests. This was reduced to 0.1 for the final model. Both of these two adjustments led to most of the increase to the processing time.

Model Accuracy

Overall the model accuracy came at the cost of performance. My work was done on a Windows laptop with an i74600U CPU @ 2.1GHz, with 8GB of memory. While the final models only took about 7 minutes to train over 10 training, multiple iterations to optimize the models added up and took some time and limited the final accuracy of the models. The below graph shows the improvement of the models for each of the following steps:

- **Data Adjustment** - Data processing
- **Model Option 1** - Linear Ridge and Tree adjusted n.tree to 200
- **Model Option 2** - Linear Lasso significant variables in Ridge and Tree shrinkage reduced to 0.1
- **Model Option 3** - Linear Alpha = .5 and Tree n.tree increased to 500
- **Model Option 4** - Linear Alpha = .25 and Tree n.tree increased to 1000
- **Model Option 5** - Linear Alpha = .2 and Tree interaction.depth = 4
- **Model Option 6** - Final model with Linear Alpha = .25 and Tree n.tree = 1000, shrinkage = 0.1, and interaction.depth 3



Conclusion

While the initial changes to the models provided big increases to the performance, many iterations were required to find models that were below the minimum RMSE. This was true for both the linear and tree models. Overall the Boosting tree provided the best performance, but took significantly longer to train. The linear model is easier to look at the parameter and coefficients to understand how it is working. Each model methods has its benefits and place depending on the need of the model.