STAT 542 Project 1 Report

Ankush Agrawal (NetID : ankusha2)

To predict the Sale Price for the Ames Housing data, I followed couple of preprocessing steps. These were done to ensure a smooth run of the models and to build a robust model. These steps are common across Part 1 and Part 2. The steps are as follows:

1. I started off with identifying the number of missing values in the entire data and to my surprise the data was relatively clean. Only 1 variable (Garage_Yr_built) had missing values, which on further investigation seemed logical as there were no garages in the houses concerned. So I removed the garage_yr_built variable. I removed Latitude and longitude as they did not make sense to me to predict Sale_Price.
2. My next step was to look into various factor and numerical variables in the data and to further understand them better:
    a. Factor Variables: Of total 83 variables, 46 or half of the variables are categorical or factor variables. I identified this through R as well as via the data dictionary I sourced from Kaggle. Upon further inspection, I found that 8 variables have 1 category dominating 98% or more of the observation. These variables are Condition_2, Heating, Pool_QC, Roof_Matl, Street and Utilities. Since there was not lot of variance in the values of these values, I removed them

       Data Description.xlsx

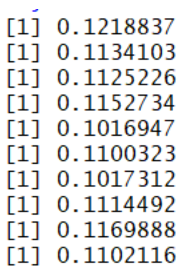
    b. Numerical Variables: I found that couple of numerical variables have extremely low number of observations present in them. This was found out when I ran a univariate (percentile/quantile) distribution over the data. Hence I removed variables such as Low_Qual_Fin_SF, Kitchen_AbvGr, Three_season_porch, Pool_Area and Misc_Val.
3. Derived variables: I created 2 new variables which made sense to me and thought could help me prop up my RMSE and also compensate for the loss of variables I dropped. These were remod_ind which is a remodeling indicator to tell if a house has been remodeled and the other is total area of the house which is a sum of all the area variables in the data. I also created another variable called "AgeAtSelling" which as the name suggests is the age of the house at selling but unfortunately I was not getting a good rmse by including this variable so I dropped it.
4. Data Transformation :
    a. I used Caret package to one-hot encode categorical variables.
    b. I used winzorisation technique to clip outliers in all the remaining numerical variables. For training data, I only provide a lower bound of 0, but for test data I provided a lower bound of 0 as well as an upper bound as the 95 percentile observed for the variable in train data. I wanted to replicate the real world scenario where we treat test data based on information from the training data.
5. Since there were more categories seen in the categorical variable of train data than test data, I had to adopt a procedure to homogenize or make the two datasets consistent after the one hot encoding. Hence, I created a list of common variables between the two datasets after one hot

encoding and there were some levels of categories that were omitted from the model development.

This finishes my data pre-processing and I moved forward with my model development.

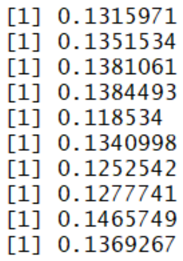
Part 1 Model 1: I tuned a Lasso regression using cross validation glmnet.

After numerous runs and multiple tweaking and adjusting, I hit the magical number of RMSE < 0.12 to gain extra credit (Attached is a snapshot). I hope the results are reproducible in the test runs by instructors. What made a real difference despite the endless data preprocessing has been the log transformation of the Y i.e. Sale_Price. It made a world of a difference and my accuracies improved significantly.



```
[1] 0.1218837
[1] 0.1134103
[1] 0.1125226
[1] 0.1152734
[1] 0.1016947
[1] 0.1100323
[1] 0.1017312
[1] 0.1114492
[1] 0.1169888
[1] 0.1102116
```

Part 1 Model 2: I ran a plain vanilla random Forest model with ntrees = 400 whose accuracies did not meet my expectation. But by this time, I had already spent close to 36 hours cumulatively trying to make Part 2 and Model 1 work. The accuracies I got were in the range of 0.13-0.14 and I am inclined to believe xgboost and gbm would have done a much better job. Hopefully, I get to leverage them in my next projects.



```
[1] 0.1315971
[1] 0.1351534
[1] 0.1381061
[1] 0.1384493
[1] 0.118534
[1] 0.1340998
[1] 0.1252542
[1] 0.1277741
[1] 0.1465749
[1] 0.1369267
```

Here are the details of my session info :

```
R version 3.5.1 (2018-07-02)
Platform: x86_64-w64-mingw32/x64 (64-bit)
Running under: Windows >= 8 x64 (build 9200)
```

I used the following packages :

1. `glmnet_2.0-16`
2. `caret_6.0-80`
3. `randomForest_4.6-14`
4. `DescTools_0.99.25`

Part 2: I used the same data preprocessing as part 1 and I am consistently seeing RMSE of 0.11.