GB 656 – Machine Learning - Final Project

Reported by Kalyani Patil

Team Members

Kalyani Patil, Jillian Wedin, and Elizabeth Deneff.

Business Context

State Farm Insurance is a large group of mutual insurance companies throughout the United States. It has several insurance products including but not limited to Homeowners insurance, Auto Insurance, Life Insurance, Flood insurance, and Renters insurance.

Computation of insurance premium is a key function for state farm. The Actuarial team determines insurance premium rates for any insurance product. The 3 biggest factors determining the home insurance premium are **location, policy deductible and home's market value**.

The home insurance department services customers all over the country. There is immense variation in the features of homes that State Farm insures. We are a team of analysts at State Farm tasked with **predicting the market value aka sale price of homes insured**.

Problem Statement

- The market value of a home at the time of sale is the sale price of the property. Several factors contribute to deciding the sale price like appraisals, estimates, negotiations etc. Therefore, there is no question of estimating the market value for new customers.
- However, with passage of time the market value of the same property changes, and real estate can be a very volatile market. In case of continuing customers, we need to determine the current market value without an appraisal or any negotiations taking place.
- Each year State Farm continues to gain new home insurance customers, leading to creation of a data set that has explanatory feature variables describing aspects of residential homes along with the sale price.
- The analytics team wants to use this data of new customers to build model that will predict the market value aka sale prices of residential homes of continuing customers; which will then be used by the actuarial team as an input for determining insurance premium of continuing customers.
- Sale Price of a home is not just determined by the height of the basement ceiling or the proximity to an east-west railroad. Data proves that much more influences prices than the number of bedrooms or a white-picket fence.
- To being modeling we have been given a train dataset with 79 explanatory variables describing (almost) every aspect of residential homes in Ames, Iowa, along with their Sale Price and we are to predict the final price of each home in the test dataset.
- This is an opportunity for us to perform creative feature engineering and advanced regression techniques to build a model that predicts sale prices.

Data Management Approach

- Data provided: Train (1460 rows and 81 columns), Test (1459 rows and 80 columns) and feature description.
- The train set has 79 feature variables and 1 outcome variable, and the test set just has the 79 feature variables.
- We first combine the train and test data into a single file and create a data frame in R. Data cleaning and feature engineering will be performed on this data frame.
- After data cleaning and feature engineering, we duplicate this data frame. One (dat1) will be used to build random forest model and another (dat2) will be used for gradient boosting model.
- As we begin to build our model/s, we will split this data frame into smaller data frames containing original data as given to us, namely train and test. Let's call them subset train and subset test respectively, for the purpose of this report.
- We will use the 'subset train data' to build (train) our predictive model, apply the model on the 'subset test data' and finally apply the model the main train data to get predictions.

Data Cleaning and Feature Engineering

- Descriptive Statistics: There are 2919 rows and 81 columns. This means that there are 79 feature variables, 1 Sale Price Column and 1 ID Column. The Average Sale Price in the Train dataset is $180921.
- Checked for structural issues in the data to make sure that data types are accurate.
- Changed 'MSSubClass' to categorical variable with factor datatype as it is a categorical variable in the variable description.
- Checked if there were any duplicate ID, to remove duplicate rows – non were removed.
- Replaced 'NA' with 'none' or '0' depending on the variable type for the following feature variables:

  MSZoning, LotFrontage, Exterior1st, Exterior2nd, MasVnrType, MasVnrArea, BsmtQual, BsmtCond, BsmtExposure, BsmtFinType1, BsmtFinType2, BsmtFinF1, BsmtFinF2, BsmtUnfSF, TotalBsmtSF, Electrical, BsmtFullBath, BsmtHalfBath, KitchenQual, Functional, FireplaceQu, GarageType, GarageYrBlt, GarageFinish, GarageCars, GarageArea, GarageQual, GarageCond, PoolQC, Fence, MiscFeature, and SaleType.

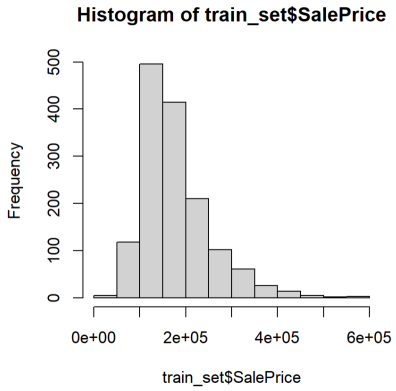
- Removed the following columns:

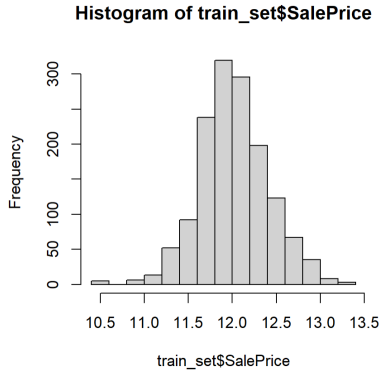
  'Alley' and 'Utilities' as it had several 'NA' values.

- Binning variables
  Binning is a technique for reducing the cardinality of continuous and discrete data. We have binned related values together in bins to reduce the number of distinct values in following feature variables:
  MasVnrArea, X2ndFlrSF, WoodDeckSF, EnclosedPorch, ScreenPorch, PoolArea, GarageYrBlt, BsmtFinSF2, LowQualFinSF, X3SsnPorch
- Since the 'subset test data' (in both dat1 and dat2) does not have any values for feature variable 'SalePrice', we remove it.
- For 'Subset train data' (in both dat1 and dat2):
  - ♦ We do not need ID column to build our predictive model, so we remove that.
  - ♦ We check the outcome variable 'SalePrice' for any Outliers. Since, we have only 4 observations with 'SalePrice' greater than 600000, we remove these observations.

- Log transformation of the outcome variable

  It is used to transform skewed data to approximately conform to normality. After plotting a histogram for 'SalePrice' we observe that the data is slightly right skewed, so we perform log transformation of the outcome variable.





  Note: After the predictions are made, we will un-transform predictions from log.

- Scaling

  It means that you transform your data so that it fits a specific scale like 0-100 or 0-1. Scaling is only needed for models that use regularization (ridge, LASSO, etc.), so we decided to skip this step as we are using random forest.

Splitting the data

Dividing data into training and test set helps in effective training and evaluating machine learning algorithms. It is typically done to avoid overfitting. The training set is typically larger than the testing set because we want to feed the model with as much data as possible to find and learn meaningful patterns. Therefore, we allocate 80% of the data to train set and 20% to test set. We do this for both models.

Performance Evaluation function

Root Mean Squared Error gives an idea of how much error the system typically makes in its predictions, with a higher weight for larger errors. So, we set up a RMSE function for the train and test dataset to do performance evaluation of the model. *The Lower the value of RMSE, the better the model is.* We do this for both models.

```
> #Create functions to find the RMSE for the train and test data
> do.RMSE.trn <- function(yhat)  sqrt( mean( (Y-yhat)^2 ) )
> do.RMSE.tst <- function(yhat)  sqrt( mean( (Y.tst-yhat)^2 ) )
> #Create Y and Y.tst
> Y <- train$SalePrice
> Y.tst <- test$SalePrice
```

Modeling - Bootstrap Aggregating (Bagging) and Boosting

The fundamental concept behind ensemble by aggregation is a simple but powerful one- the wisdom of crowds: Many relatively uncorrelated models (trees) operating as a committee will outperform any of the individual constituent models. Bagging and Boosting are two examples of such ensembles.

- ***Random Forest***

  Bootstrap Aggregating creates multiple bootstrap samples out of the original dataset so that each new bootstrap sample will act as another (almost) independent dataset. Then, we fit a model for each of these samples and

finally aggregate by taking the average of their outputs. Random Forest is an extension of Bagging that also randomly selects subsets of features used in each data sample.

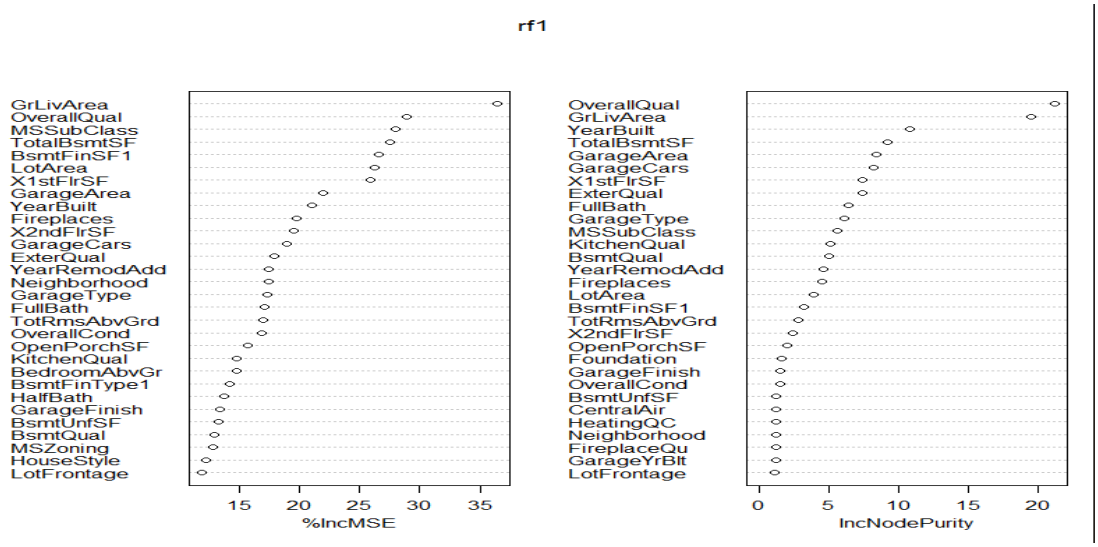
Let's set the parameters. One nice aspect about random forests is that there aren't many "knobs" – we must set the number of variables to try in each tree 'mtry', we have to tell the implementation whether to assess variable importance, and the number bootstrap resamples 'ntree'.

```
#create parameter for number of features to try in each tree
mtry <- round(ncol(train)^.5); mtry
#create a parameter for the number of bootstrap samples
ntree <- 1000
```

Build the model

```
rf1 <- randomForest(SalePrice ~ ., data = train, ntree=ntree, mtry=mtry, importance=TRUE)
```

After running the model, we use the 'importance' function that gives the influence of each feature, as measured by aggregate reduction in the predictive performance. The variable important plots provide a nice way to understand what drives the predictions.



Evaluating Predictions

We evaluate our model by doing out of sample prediction accuracy and finding out 'out of sample' RMSE.

```
RMSE.trn_RF <- do.RMSE.trn(predict(rf1, data = train))
RMSE.trn_RF #0.1416503
RMSE.tst_RF <- do.RMSE.tst(predict(rf1, data = test))
RMSE.tst_RF #0.5099826
```

The in sample RMSE is 0.1416503 and out of sample RMSE is 0.5099826

- ***Gradient Boosting Model***

Boosting is a technique that consists in fitting models sequentially, in an adaptive way: each model in the sequence is fitted giving more importance to observations in the dataset that were badly handled by the previous models in the sequence. Intuitively, each new model focuses its effort on residuals of the last model. Gradient boosting uses a gradient descent algorithm to minimize loss when adding models to the ensemble. Unlike random forests, the decision trees in gradient boosting are built additively; in other words; each decision tree is built one after another. Subsequent models are fit to pseudo-residuals instead of adjusting weights.

For this model, we use dat2 as we had to put the data in a matrix form.

Creating dummy variables (columns) for feature variables with character data type. And then removed the character variables.

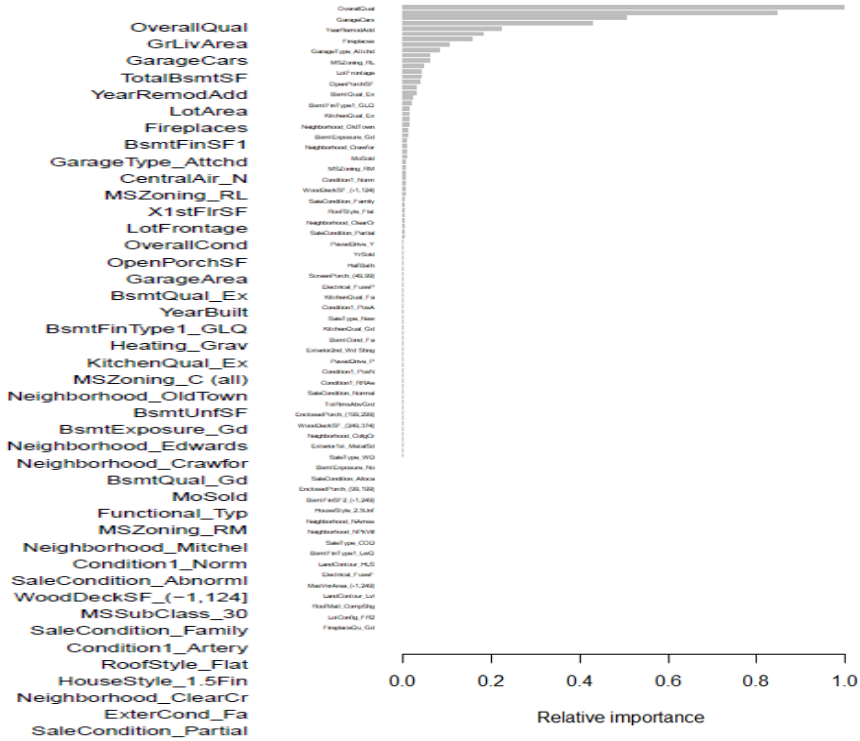
Setting Parameters

```
parm <- list(nthread=2, max_depth=5, eta=0.10, gamma=3.5, min_child_weight=20, subsample=.67)
```

Build the model

```
bt <- xgboost(parm, data=X, label=Y2, verbose=2, objective='reg:squarederror', nrounds=20)
```

Evaluating the influence of features using a variable importance plot:



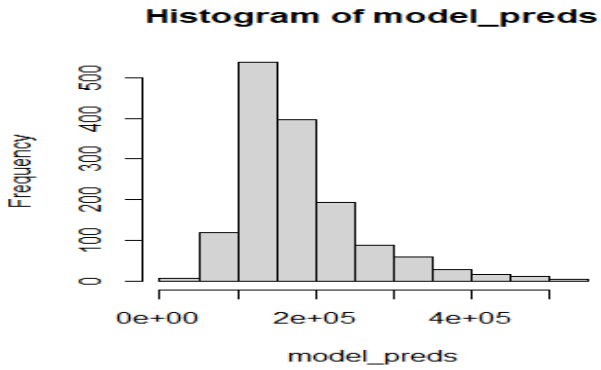
Evaluating Predictions:

```
#Make predictions and find out of sample RMSE
RMSE.trn_BT <- do.RMSE.trn2(predict(bt, newdata = X))
RMSE.trn_BT #0.06958905
#Convert test data to matrix
RMSE.tst_BT <- do.RMSE.tst2(predict(bt, newdata = X.tst))
RMSE.tst_BT #0.1385241
```

Since the out of sample RMSE of the gradient boost model is the lowest, that is the better model.

Results

Finally applying the Gradient Boosting Model to the main test data. Following histogram captures the frequency distribution of the predicted sale prices.


Histogram of model_preds
Frequency
0
100
200
300
400
500
0e+00
2e+05
4e+05
model_preds