House Prices – Advanced Regression Techniques
Link to data: https://www.kaggle.com/c/house-prices-advanced-regression-techniques/data

Kaggle name: Aemen Sultan
Total no. of teams: 5165

House Prices – Advanced Regression Techniques

Background

In this competition, we are assessing what factors impact house prices in order to generate a model that predicts these values. Some of these factors include what year house was built, external condition, number of bedrooms, etc. As a data scientist in the making, my goal is to apply feature engineering and other mechanisms in developing a useful model to assist in prediction submission.

Tackling this project as a data scientist in the making, below I will demonstrate the approach I took in manipulating data that was involved in establishing final model. As mentioned before, much of this process encompasses data preparation. Through this competition, I was allowed the opportunity to challenge my analytical skills and enhance my exposure to concepts like predictive modelling.

Datasets

Utilizing R language for my data manipulation process, two data sets for *Train* and *Test* were imported from the House Price data (link above). One of the primary reasons for selecting this competition was data size. Both train and test csv files enabled an easy import to R which was a major problem in my attempt to tackle some of the other competitions mentioned above. The entire dataset set will be cleaned and formatted to generate the best fitted model that we can use to predict prices on Test dataset.

Exploratory Data Analysis

EDA is a data analysis process that makes use of different graphical and non-graphical techniques to identify anomalies and patterns in data set. The process of identifying these components aids in building more reliable models. Moreover, identifying patterns can support understanding how to process different types of columns (does variable have integer values or string values?).

1. **Understanding the dataset** – below is a screenshot of R code used to identify what category of variables that exist in our data frame. This is the structure of our data set after combining Train and Test files and removing variables unnecessary/repetitive to the model. For example, ExterQual (quality of material on exterior) was removed due to

the fact Exterior1st is also an indicator of material used on exterior. Moreover, new column for “SalePrice” was also created in the Test dataset as it does not exist.

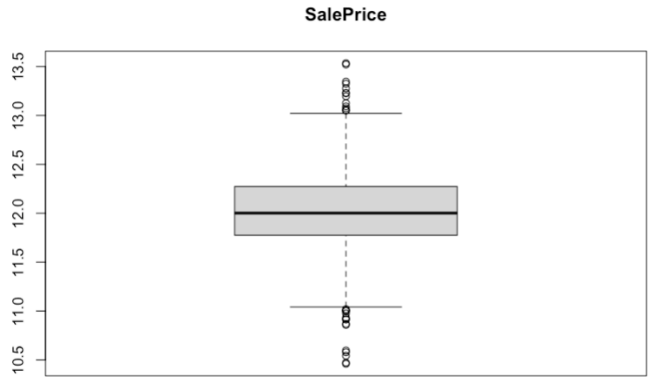
```
> ##Check structure of the new data frame
> str(New_Data_DF)
'data.frame':   2919 obs. of  69 variables:
 $ Id           : int  1 2 3 4 5 6 7 8 9 10 ...
 $ MSSubClass   : int  60 20 60 70 60 50 20 60 50 190 ...
 $ MSZoning     : chr  "RL" "RL" "RL" "RL" ...
 $ LotFrontage  : int  65 80 68 60 84 85 75 NA 51 50 ...
 $ LotArea      : int  8450 9600 11250 9550 14260 14115 10084 10382 6120 7420 .
 $ Street       : chr  "Pave" "Pave" "Pave" "Pave" ...
 $ Alley        : chr  NA NA NA NA ...
 $ LotShape     : chr  "Reg" "Reg" "IR1" "IR1" ...
 $ LandContour  : chr  "Lvl" "Lvl" "Lvl" "Lvl" ...
 $ Utilities    : chr  "AllPub" "AllPub" "AllPub" "AllPub" ...
 $ LotConfig    : chr  "Inside" "FR2" "Inside" "Corner" ...
 $ LandSlope    : chr  "Gtl" "Gtl" "Gtl" "Gtl" ...
 $ Neighborhood : chr  "CollgCr" "Veenker" "CollgCr" "Crawfor" ...
 $ Condition1   : chr  "Norm" "Feedr" "Norm" "Norm" ...
 $ Condition2   : chr  "Norm" "Norm" "Norm" "Norm" ...
 $ BldgType     : chr  "1Fam" "1Fam" "1Fam" "1Fam" ...
 $ HouseStyle   : chr  "2Story" "1Story" "2Story" "2Story" ...
 $ YearBuilt    : int  2003 1976 2001 1915 2000 1993 2004 1973 1931 1939 ...
 $ YearRemodAdd : int  2003 1976 2002 1970 2000 1995 2005 1973 1950 1950 ...
 $ RoofStyle    : chr  "Gable" "Gable" "Gable" "Gable" ...
 $ RoofMatl     : chr  "CompShg" "CompShg" "CompShg" "CompShg" ...
```

2. **Formatting** – as displayed above, our data contains string and integer data types. All categorical and ordinal variables (e.g. MSZoning) will need to be factorized to allow proper ingestion into our model.

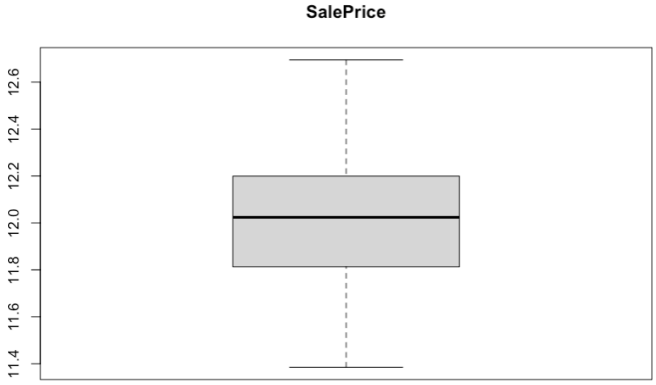
```
##Creating factors for all categorical variables
New_Data_DF1 <- New_Data_DF %>%
  mutate(MSZoning <- fct_recode(MSZoning, c="c(all)"),
         Street <- factor(Street),
         Alley <- factor(Alley),
         LotShape <- factor(LotShape),
```

3. **Outliers** – an outlier is any value in our dataset that extensively differs from other values. Due to the rules of this competition, we cannot drop outliers as it would result in less predictions than the requirement. In order to deal with outliers, we utilize the mean and median to calculate missing values.

   Boxplot containing Outliers:



Boxplot after outliers are addressed:



4. **Missing Values** – the dataset, after removing outliers, still contains missing values that make it difficult in performing regression. Due to the fact columns Alley, FireplaceQu, PoolQC and Fence contained a significant portion of missing values, I had to remove these.

```
> print (MV_Count)
          Id    MSSubClass      MSZoning   LotFrontage       LotArea        Street         Alley
           0             0             4           486             0             0          2721
  Condition1    Condition2     BldgType    HouseStyle     YearBuilt  YearRemodAdd     RoofStyle
           0             0             0             0             0             0             0
    BsmtQual      BsmtCond  BsmtExposure  BsmtFinType1    BsmtFinSF1  BsmtFinType2    BsmtFinSF2
          81            82            82            79             1            80             1
   X1stFlrSF     X2ndFlrSF  LowQualFinSF     GrLivArea  BsmtFullBath  BsmtHalfBath      FullBath
           0             0             0             0             2             2             0
 FireplaceQu    GarageType  GarageFinish    GarageCars    GarageArea    GarageQual    GarageCond
        1420           157           159             1             1           159           159
    PoolArea        PoolQC         Fence     SalePrice
           0          2909          2348          1459
> |
```

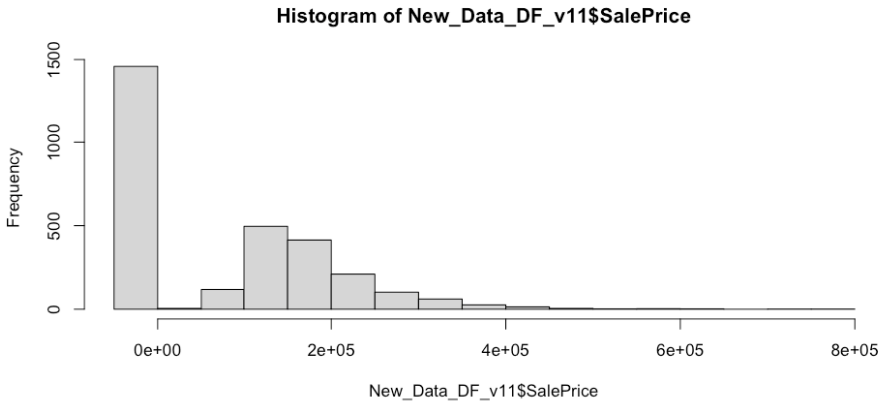
To deal with other missing values such as in BsmtQual, I used imputation method in R provided by the “MICE” package. MICE function works on data that is missing at random. Using this function, I was able to impute missing data variable by variable with

5 iteration and using “cart” method which is for classification and regression. As per below, all missing values are now gone.

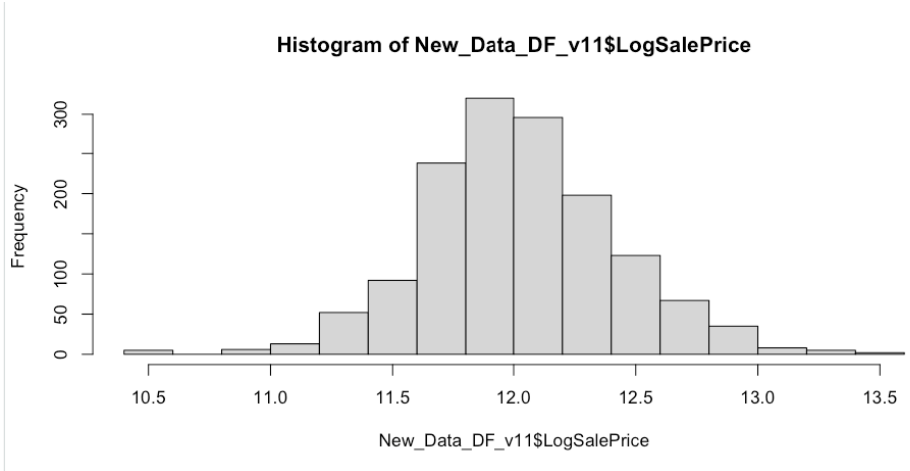
```
> sapply(New_Data_DF_v4, function(New_Data_DF_v4) sum(is.na(New_Data_DF_v4)))
Outliers_Data            Id    MSSubClass      MSZoning   LotFrontage       LotArea        Street      LotShape   LandContour     Utilities     LotConfig 
            0             0             0             0             0             0             0             0             0             0             0 
    LandSlope  Neighborhood    Condition1    Condition2      BldgType    HouseStyle     YearBuilt  YearRemodAdd     RoofStyle       RoofMatl   Exterior1st 
            0             0             0             0             0             0             0             0             0             0             0 
  Exterior2nd    MasVnrType    MasVnrArea    Foundation      BsmtQual      BsmtCond  BsmtExposure  BsmtFinType1    BsmtFinSF1  BsmtFinType2    BsmtFinSF2 
            0             0             0             0             0             0             0             0             0             0             0 
   BsmtUnfSF   TotalBsmtSF       Heating     HeatingQC    CentralAir    Electrical     X1stFlrSF     X2ndFlrSF  LowQualFinSF     GrLivArea  BsmtFullBath 
            0             0             0             0             0             0             0             0             0             0             0 
BsmtHalfBath      FullBath      HalfBath  BedroomAbvGr  KitchenAbvGr  TotRmsAbvGrd    Functional    Fireplaces    GarageType  GarageFinish    GarageCars 
            0             0             0             0             0             0             0             0             0             0             0 
  GarageArea    GarageQual    GarageCond    PavedDrive    WoodDeckSF   OpenPorchSF EnclosedPorch    X3SsnPorch   ScreenPorch      PoolArea     SalePrice 
            0             0             0             0             0             0             0             0             0             0             0 
LogSalePrice 
            0 
```

5. **Log Transform –** as a part of preliminary assessment, taking log of SalePrice eliminates the “skewness” of the distribution. As taking the log improves our ability to linearize a model, I have used LogSalePrice in the final model.

Histogram of SalePrice (skewed):



Histogram of the Log SalePrice:



Model Development & Validation

Stepwise regression is a technique that runs model through multiple iterations by adding and removing variables in order to select the best one (i.e. one with lowest prediction error). The Akaike Information Criterion (AIC) is a good indicator of the prediction error so we select the model with the lowest AIC value. I used "step" function in R to run this regression on my cleaned data and selected the best model.

Cross Validation (CV) is used to test our model before we decide to finalize it. I was able to perform CV on my model to evaluate the Mean Absolute Error (MAE) which is calculated by taking difference between our observed and predicted values. We can see a value for Root Mean Square Error (RMSE) below as well.

```
> print(Model_Train_v1)
Linear Regression

2919 samples
  66 predictor

No pre-processing
Resampling: Bootstrapped (25 reps)
Summary of sample sizes: 2919, 2919, 2919, 2919, 2919, 2919, ...
Resampling results:

  RMSE      Rsquared   MAE
  0.379173  0.8219918  0.1206279

Tuning parameter 'intercept' was held constant at a value of TRUE
```

Final model based on best R-squared value:

```
> summary(Model_Final_v2)

Call:
lm(formula = LogSalePrice ~ GrLivArea + YearRemodAdd + LotArea +
    PavedDrive + X3SsnPorch + TotalBsmtSF + TotRmsAbvGrd + BedroomAbvGr +
    X1stFlrSF + Fireplaces, data = Data.Train)

Residuals:
     Min       1Q   Median       3Q      Max
-2.86900 -0.09009  0.02098  0.11582  0.58503

Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept)  -1.710e+00  5.550e-01  -3.081 0.002100 **
GrLivArea     2.849e-04  2.088e-05  13.642  < 2e-16 ***
YearRemodAdd  6.417e-03  2.817e-04  22.779  < 2e-16 ***
LotArea       1.753e-06  5.635e-07   3.111 0.001904 **
PavedDriveP   1.629e-01  4.254e-02   3.829 0.000134 ***
PavedDriveY   2.412e-01  2.259e-02  10.677  < 2e-16 ***
X3SsnPorch    2.467e-04  1.793e-04   1.376 0.169123
TotalBsmtSF   2.333e-04  2.148e-05  10.861  < 2e-16 ***
TotRmsAbvGrd  1.953e-02  6.725e-03   2.904 0.003735 **
BedroomAbvGr -2.552e-02  9.233e-03  -2.764 0.005776 **
X1stFlrSF    -3.131e-05  2.609e-05  -1.200 0.230355
Fireplaces    9.342e-02  9.639e-03   9.692  < 2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.2001 on 1448 degrees of freedom
Multiple R-squared:  0.7509,    Adjusted R-squared:  0.749
F-statistic: 396.7 on 11 and 1448 DF,  p-value: < 2.2e-16
```

LASSO/Ridge Regression

As an attempt to score better in the leaderboard of this competition, I developed a LASSO and Ridge regression model by performing below steps:

- Splitting Train data set into *Lasso Train* and *Lasso Test*
- Creating log y on *Lasso Train* and x matrix on original cleaned data set
- Further splitting our x matrix into train (first 1100 rows), test (rows 1101 to 1460) and predict (last 1459 rows)
- Running regression for alpha = 1 (LASSO) and alpha = 0 (Ridge)
- Selected LASSO model as it had a better MAPE score

Due to competition policies, I was not able to generate submission file that adhered to requirements and therefore, used "predict" function in R to generate sale prices for test data.

Predicting Sale Prices

There are less steps involved in simply using the predict function in R and the first step entails splitting final data set back into Train (first 1460 rows) and Test data sets (last 1459 rows). Next, I use selected final model (from model development) to predict prices on the test data. This generates a Root Mean Squared Logarithmic error of 0.19794.

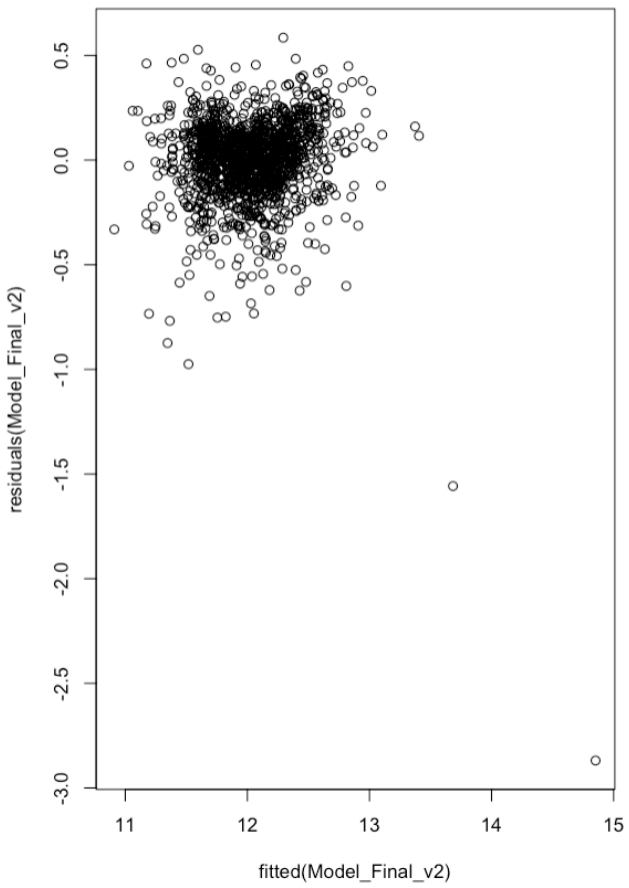
The final model is contained in Assignment 1 - Final.R

```
##Running prediction on test data
Data.Train <- subset(New_Data_DF_v4, New_Data_DF_v4$Id <= 1460)
Data.Test <- subset(New_Data_DF_v4, New_Data_DF_v4$Id > 1460)

Model_Final_v2 <- lm(LogSalePrice ~ GrLivArea + YearRemodAdd + LotArea + PavedDrive +
                       X3SsnPorch + TotalBsmtSF + TotRmsAbvGrd + BedroomAbvGr +
                       X1stFlrSF + Fireplaces, Data.Train)
summary(Model_Final_v2)

Predict_SalePrice <- exp(predict(Model_Final_v2, Data.Test))
write.csv(Predict_SalePrice, file = "SalePrice_Predictions.csv")
```

Residuals vs Fitted plot:


residuals(Model_Final_v2)
0.5
0.0
-0.5
-1.0
-1.5
-2.0
-2.5
-3.0
11
12
13
14
15
fitted(Model_Final_v2)