House Price Prediction

Deepti Vijay Khandagale (G01353007)

Awantika Shah (G01351499)

Group 3

Course: STAT-515: Applied Statistics and Visualization for Analytics

Spring Semester 2022

Instructor: Prof. Tokunbo Fadahunsi, PhD

PROJECT OBJECTIVE

The objective of this project is to predict the final price of the house using different statistical models.

BACKGROUND

In this project, we are using the "Ames Housing dataset" from the kaggle competition which is originally prepared by Dean De Cock. The data contains the following things:

- train.csv: the training set
- test.csv: the test set
- data_description.txt: full description of each column (columns edited)

DATASET OVERVIEW

The aim is to use statistical methods for analysis, find out the important features and perform regression techniques to build a prediction model. The data contains **81 variables** and **1,460 observations** describing the different parameters of the residential homes in Ames, Iowa. With the train dataset, we intend to fit a linear model which will predict the final price of the house.

METHODOLOGY

In order to build a reliable model, we have performed the following steps:

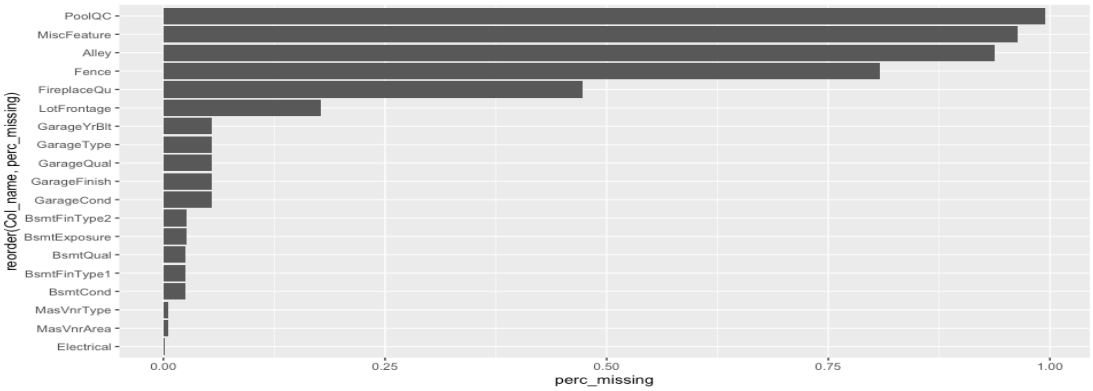
1. *Data Quality and Cleaning* : It is necessary that our data should be clean, hence we did some exploratory data analysis to check the data quality and performed data cleaning.
2. *Feature Engineering*: We have approx 81 columns to predict the house price which has mixture of numerical and categorical data. There is lot of opportunities to perform feature engineering to produce some strong features that will help in building a robust model.
3. *Training Model* : After the data cleaning and feature engineering, we will be training and tuning various linear regression models and will diagnostic the models.
4. *Prediction and Model Comparison* : Once we train our models we will compare their performances by testing it on the Test data and compare the MSE, MAE and RSME metrics.

Loading the dataset

The train data has 1,460 observations and 81 variables. In this project, our response variable is "SalePrice" which we want to predict and we'll be regressing the response variable **"SalePrice"** w.r.t to the predictors i.e the rest of the variables.

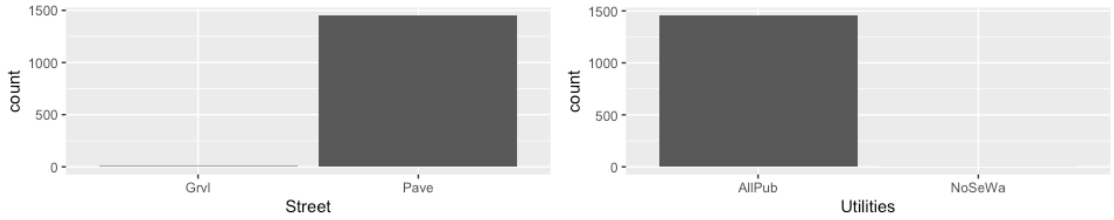
Data Quality and Cleaning

The first step for the data cleaning is to check for the missing values. Missing values can be nightmare for any analysis, hence we will be finding the missing values in the dataset.



Above we can see in the graph that there are 19 variables which have missing values and 5 of them have more than ~ 50% of missing values. We decided to drop all the columns which has more than 40% of missing data (eliminating 5 variables : PoolQC, MiscFeature, Alley, Fence, FireplaceQu). For the rest of the variables we have performed data imputation as mentioned below.

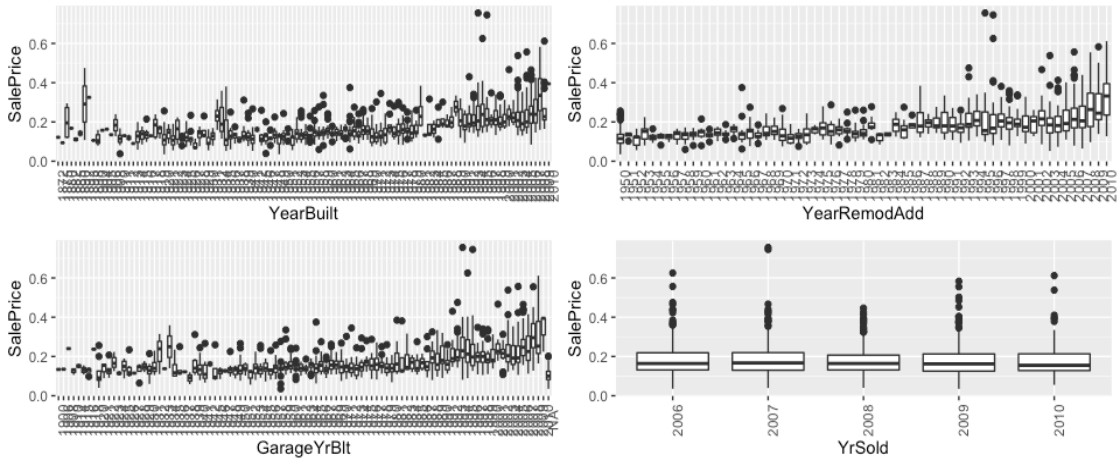
1. *Numerical Variables* : All the numerical variables which had missing values were replaced by the median of the variable as some of the variables had skewed distribution.
2. *Categorical Variables* : All the Categorical variables which had missing values were replaced by the Mode of the variable.
3. *Miscellaneous Variables* : There were some variables which had meaning for the missing values, example : Categorical columns related to Basement had missing values which is associated with the No Basement. Similarly we had missing values in the Garage columns which means there is no garage for that observation.
4. There are columns like **Street** and **Utilities** which had only 2 categories and were highly skewed. We decided to drop these two columns as they hold no great value here. Below is the Bar plot of these two columns.



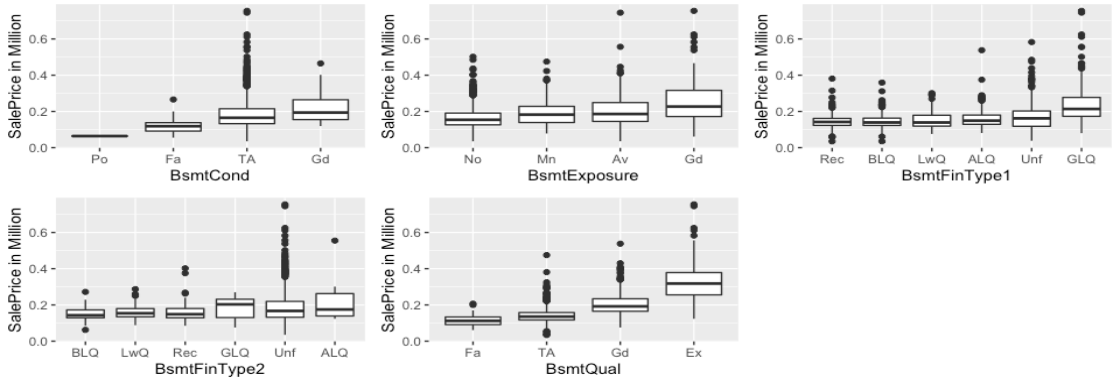
Feature Engineering

Feature engineering is one of the most important steps in any Machine Learning process. We can improve the performance of any model by creative feature engineering. As we know the variables are treated differently depending on their datatype we must make sure to prepare these columns for training our models. Hence we will first tackle all the categorical variables first to see if there is any opportunity for feature engineering. We will be categorizing all the categorical variables based on their type which are as follows :

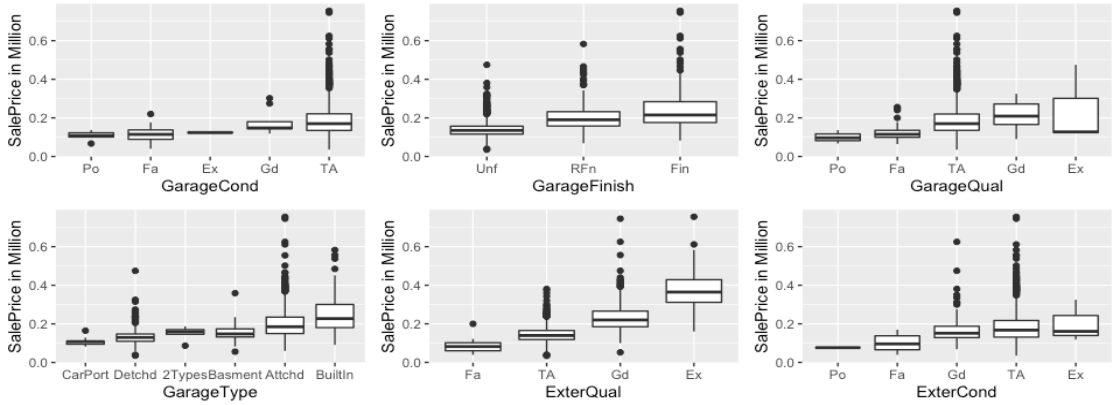
Year of Construction: There are 4 variables with years (yearBuilt, yearRemodled, GarageYrBlt and YrSold) each ploted against the Sale price. One can notice that the sale price are increasing with the years for the yearBuilt, yearRemodled, GarageYrBlt columns and there is no differnce in the Sale price for different YrSold. Hence we will be dropping YrSold and it's associated MoSold column as it shows no impact on the final Saleprice. As the Sale price is increasing with the years in other three variables, it is natural to keep these variables as numerical. We can expect some collinearity between these variable and should consider only one out of all, but for now we will keep all of them and let our models do the magic.



Basement: In this category, we included all the variables related to basements like BsmtCond, BsmtExposure, BsmtFinType1, BsmtFinType2 and BsmtQual. We have created the boxplot of sale Price against all these variables and we observed that the sale price is affected by BsmtCond, BsmtExposure and BsmtQual, i.e better the quality better the price. Hence it is safe to assign ordinal values to these categorical values which shall be treated as numerical value in our model. This way we will be able to reduce dimensions in our model to some extent. BsmtfinType1 and BsmtfinType1 doesn't affect much so we will leave it as it is.

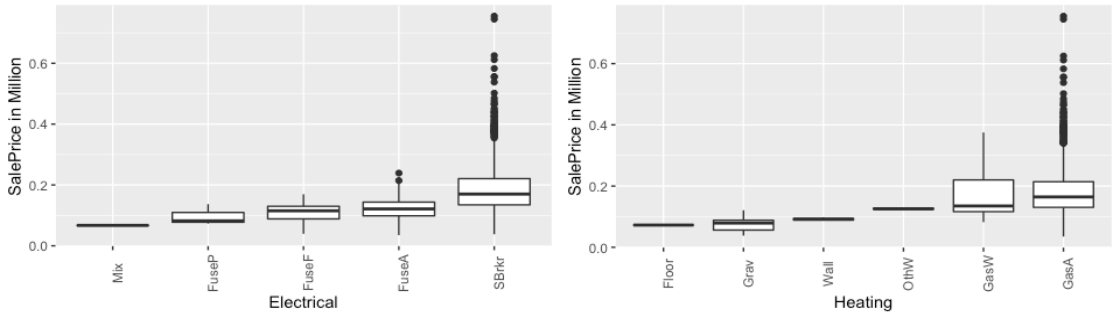


Garage & Exterior: Just like in Basement, we did similar exercise for the Garage and Exterior, we found some relation with the GarageQual, GarageCond and GarageFinish; ExterCond and ExterQual. Hence it is safe to assign ordinal values to these categorical values which shall be treated as numerical value in our model. For others, it doesn't affect much so we will leave it as it is.



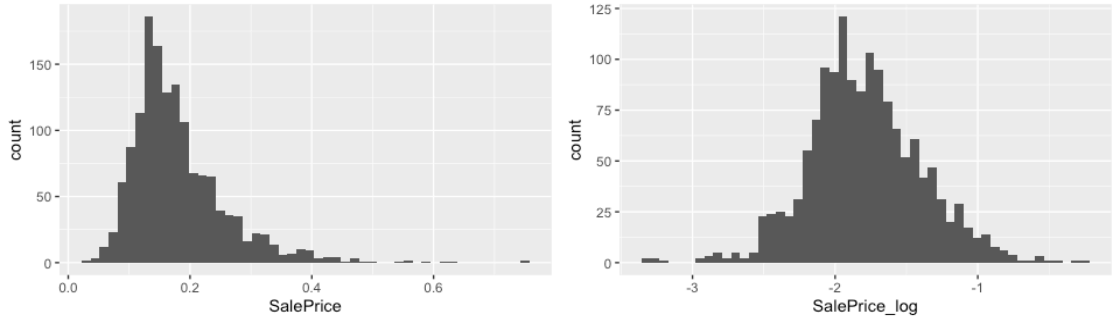
Construction: For the construction, in the variables like Foundation, LandContour, LandSlope, NldgType, MasvnrType, Lot Shape, KitchenQual, RoofStyl and RoofMatl, there is no ordinal data and no pattern was found. So we are keeping it as Categorical

Appliance: In appliance, Heating and Electrical showed some pattern. Hence we assign ordinal values to these categorical values which shall be treated as numerical value. Rest we will leave as it is.



Miscellaneous: Other Columns on Miscellaneous include HeatingQC, HouseStyle, LotConfig, SaleCondition, SaleType, Functional and CentralAc doesn't show any pattern, hence we treat it as categorical only.

SalePrice: Let us check the linearity assumption by plotting a histogram of our target variable sale price. One the left graph we can see that the sale price distribution is rightly skewed which means that the assumption of linearity may not be true here. In order to mitigate this we can take the natural log of the sale price and check the distribution. On the right we can see that the log distribution looks normally distributed, hence we shall be training our model using the Log of the target variable.



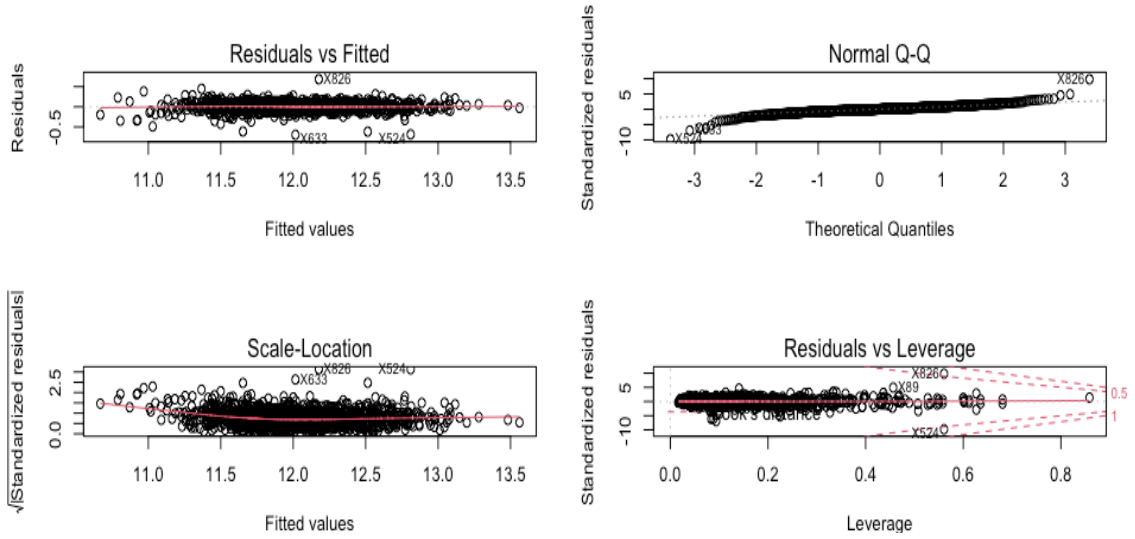
TRAINING MODEL

Now that we have done the heavy lifting of data cleaning and feature engineering, it's time to train our model and get some prediction. For this project we will be focusing on the Linear regression techniques we have learned in the Class and shall be implementing 4 type of linear regression which are as follows:

Multiple Linear Regression (with Cross Validation)

```
# knitr::opts_chunk$set(warning = FALSE, message = FALSE)
custom <- trainControl(method = "repeatedcv",number = 10, repeats = 5, verboseIter = F)
lm <- train(log(SalePrice)~. , train_house,  method = 'lm',  trControl = custom)

summary(lm)
lm$results

par(mfrow=c(2,2))
plot(lm$finalModel)
```

[4]

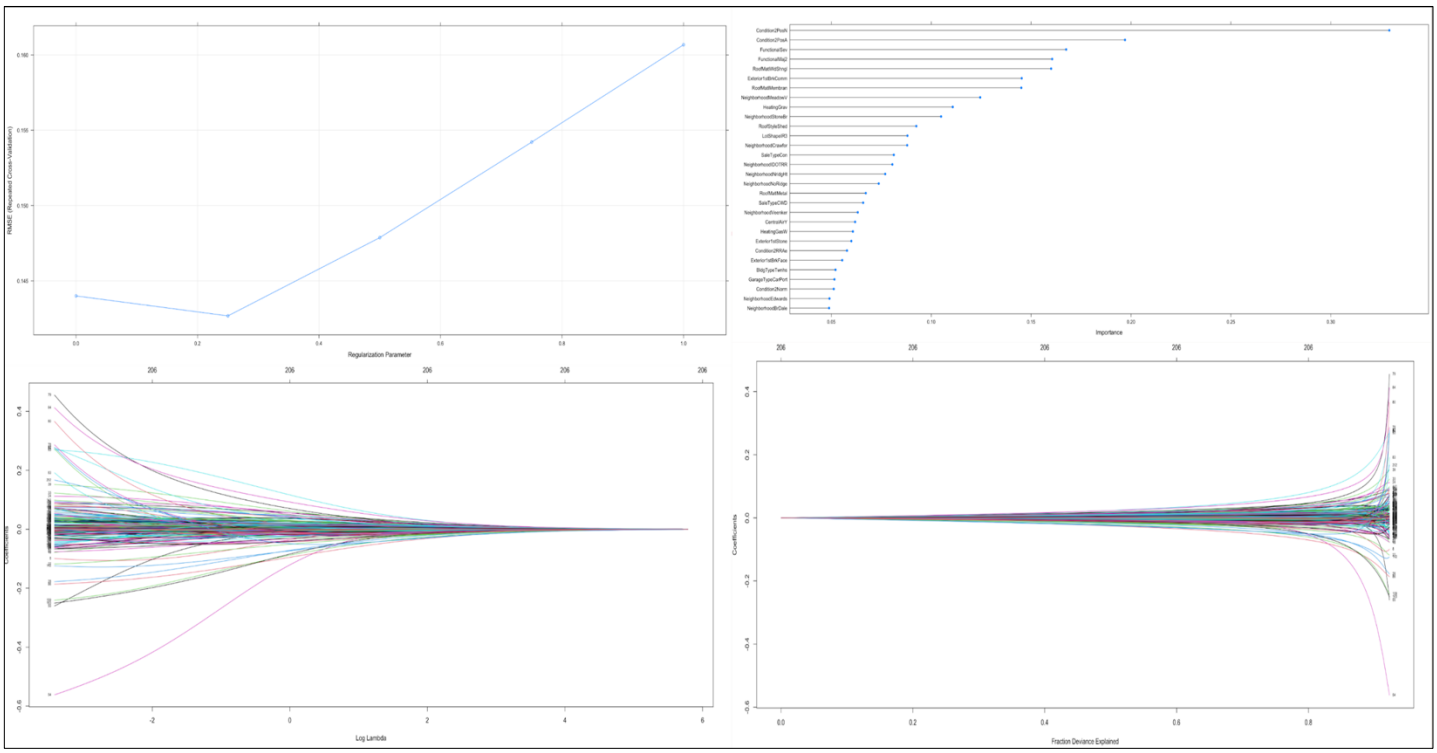


In multiple linear regression, the output of the data is interpreted using the diagnostics plot: **Residual v/s fitted graph** doesn't show any pattern in the graph, which means that there is no non-linear relationship between the predictor and outcome variables. **Normal Q-Q**, the residuals are normally distributed. The points forming but there are few outliers at the lower side. **Scale-Location**, this graph checks the assumption of homoscedasticity and as seen the residuals are scattered randomly which satisfies assumption. **Residual vs Leverage**, this graph helps us to identify the influential outliers that might affect the analysis and here outside Cook's distance (red dotted) we find two outliers. [2]

Ridge Linear Regression (with Cross Validation)

```
set.seed(1234)
custom <- trainControl(method = "repeatedcv", number = 10, repeats = 5, verboseIter = F)
ridge <- train(log(SalePrice)~. ,train_house,method = 'glmnet',tuneGrid = expand.grid(al
pha = 0,lambda = seq(0.0001,1,length = 5)),trControl = custom)
ridge

The final values used for the model were alpha = 0 and lambda = 0.250075.

ridge_plot <- plot(ridge)
ridge_lamda <- plot(ridge$finalModel , xvar = 'lambda' , label = T)

ridge_dev <- plot(ridge$finalModel , xvar = 'dev' , label = T)

ridge_30 <- plot(varImp(ridge , scale = F) , top=30)
ridge_30

## glmnet
##
## 1460 samples
##   70 predictor
##
## No pre-processing
## Resampling: Cross-Validated (10 fold, repeated 5 times)
## Summary of sample sizes: 1315, 1313, 1314, 1313, 1314, 1315, ...
## Resampling results across tuning parameters:
##
##   lambda    RMSE       Rsquared   MAE
##   0.000100  0.1443477  0.8697884  0.09162363
##   0.250075  0.1427452  0.8745107  0.09241939
##   0.500050  0.1478353  0.8716688  0.09720715
##   0.750025  0.1540901  0.8679603  0.10262491
```

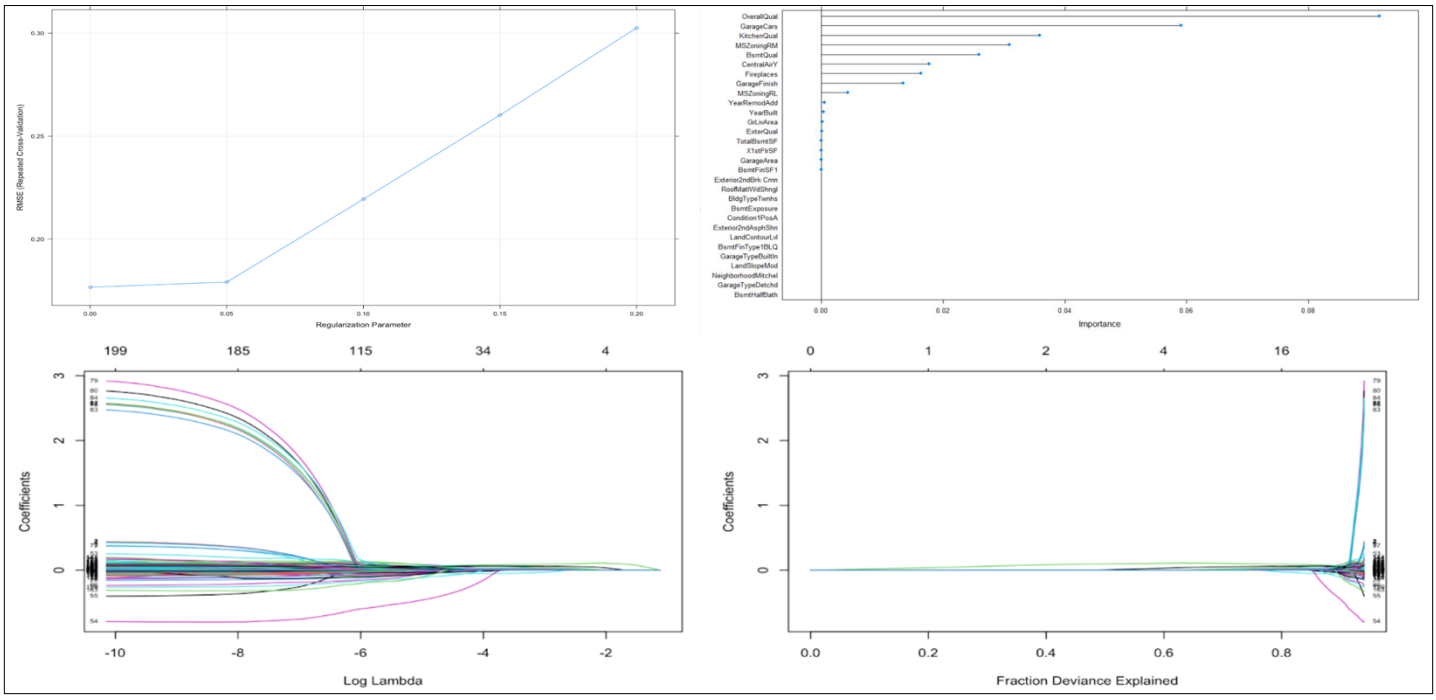
```
##   1.000000  0.1604961  0.8643436  0.10812448
##
## Tuning parameter 'alpha' was held constant at a value of 0
## RMSE was used to select the optimal model using the smallest value.
## The final values used for the model were alpha = 0 and lambda = 0.250075.

ridge_plot <- plot(ridge)
ridge_lamda <- plot(ridge$finalModel , xvar = 'lambda' , label = T)
```



In Ridge linear regression, it penalizes using Lamda on the co-efficient, if lamda is increasing, the co-efficients tends towards zero. We find the best lamda, which gives us the important variables for our model. So here, the best lamda is 0.25 and it provides the top 30 important variables as Condition2PosN, Condition2PosA, FunctionalSev, FunctionalMaj2, RoofMatWdshngl, ExtFirstBrkComm and so on.[2]

Lasso Linear Regression (with Cross Validation)

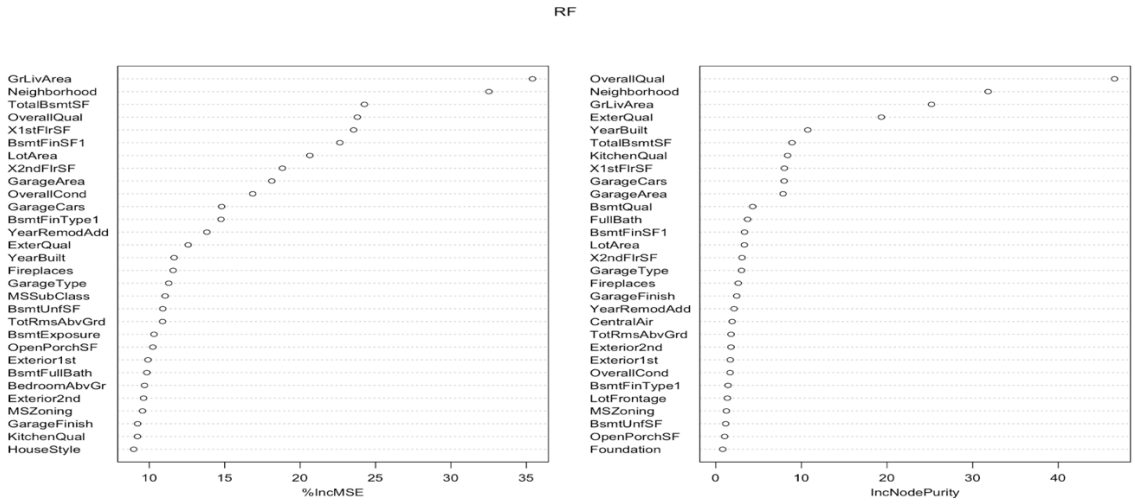
```
set.seed(1234)
custom <- trainControl(method = "repeatedcv",number = 10, repeats = 5, verboseIter = T)

lasso <- train(log(SalePrice)~. ,train_house, method = 'glmnet',tuneGrid = expand.grid(a
lpha = 1, lambda = seq(0.0001,0.2,length = 5)),trControl = custom)

par(mfrow=c(2,2))
plot(lasso)

plot(lasso$finalModel , xvar = 'lambda' , label = T)
plot(lasso$finalModel , xvar = 'dev' , label = T)
plot(varImp(lasso , scale = FALSE) , top=30 )
```



In lasso regression, the lamda shrinks coefficient completely to zero and removes the unnecessary predictors. So here, the best lamda is very small 0.05 and it provides the top 30 important variables as . The topmost variables are OverallQual, GarageCars, KitchenQual, MSZoningRM and BsmtQual.

Random Forest Regression

```
set.seed(123)
RF = randomForest(log(train_house$SalePrice)~., data=train_house[-71],
                          mtry=20, ntree=500,importance=TRUE, na.action = na.omit)
RF


## Call:
##  randomForest(formula = log(train_house$SalePrice) ~ ., data = train_house[-71],
mtry = 20, ntree = 500, importance = TRUE, na.action = na.omit)
##                Type of random forest: regression
##                      Number of trees: 500
## No. of variables tried at each split: 20
##
##           Mean of squared residuals: 0.01841201
##                     % Var explained: 88.45

##               %IncMSE IncNodePurity
## MSSubClass  11.051471     0.6466833
## MSZoning     9.548788     1.2390741
## LotArea     20.643507     3.3392820
## LotShape     4.447814     0.2433614
## LandContour  1.973690     0.3203438
## LotConfig   -1.815598     0.2393249
```



In random forest, mean of squared residuals: 0.01841201 means the prediction error. 88.45% variance is explained by the variables. In the first plot, the higher the increase in MSE, the important is the variable. we see that GrLivArea, Neighborhood, TotalBsmiSF and so on are most important. In second, node purity is depended on the Gini Index and here the important variables are Overall Qual, Neighborhood, GrLivArea etc.

PREDICTION AND MODEL COMPARISON

Now we have trained our models on the train data, it is time to test it on the test data. We have done the same level of data quality checking and cleaning as it was applied to train data.

```
pred.lm = predict(lm,newdata=test_house)
rmse(log(test_Saleprice$SalePrice), pred.lm )

## [1] 2.701563

pred.ridge = predict(ridge,newdata=test_house)
rmse(log(test_Saleprice$SalePrice), pred.ridge)

## [1] 0.3617194

pred.lasso = predict(lasso,newdata=test_house)
rmse(log(test_Saleprice$SalePrice), pred.lasso )

## [1] 0.3011523

pred.RF = predict(RF,newdata=test_house)
rmse(log(test_Saleprice$SalePrice), pred.RF )

## [1] 0.3411514
```

CONCLUSION

As per our analysis on the test data, it is seen that Root Mean Square error of Lasso regression is lowest and hence we select the **Lasso Regression** the best fit for our prediction of sale price.The topmost variables are OverallQual, GarageCars, KitchenQual, MSZoningRM and BsmtQual.

REFERENCES

[1] House Price Prediction Data. Retrieved from: https://www.kaggle.com/competitions/house-prices-advanced-regression-techniques/overview

[2] James, G., Witten, D., Hastie, T., & Tibshirani, R. (2013). An introduction to statistical learning: With applications in R.

[3] R Codes. retrieved from: https://www.rdocumentation.org/

[4] MLR Output Summary: https://github.com/Deepti1206/House_Price_Prediction_Project/blob/main/MLR_Output_Summary.pdf