Houseing Price Prediction

Ramesh Mitawa

2021-05-12

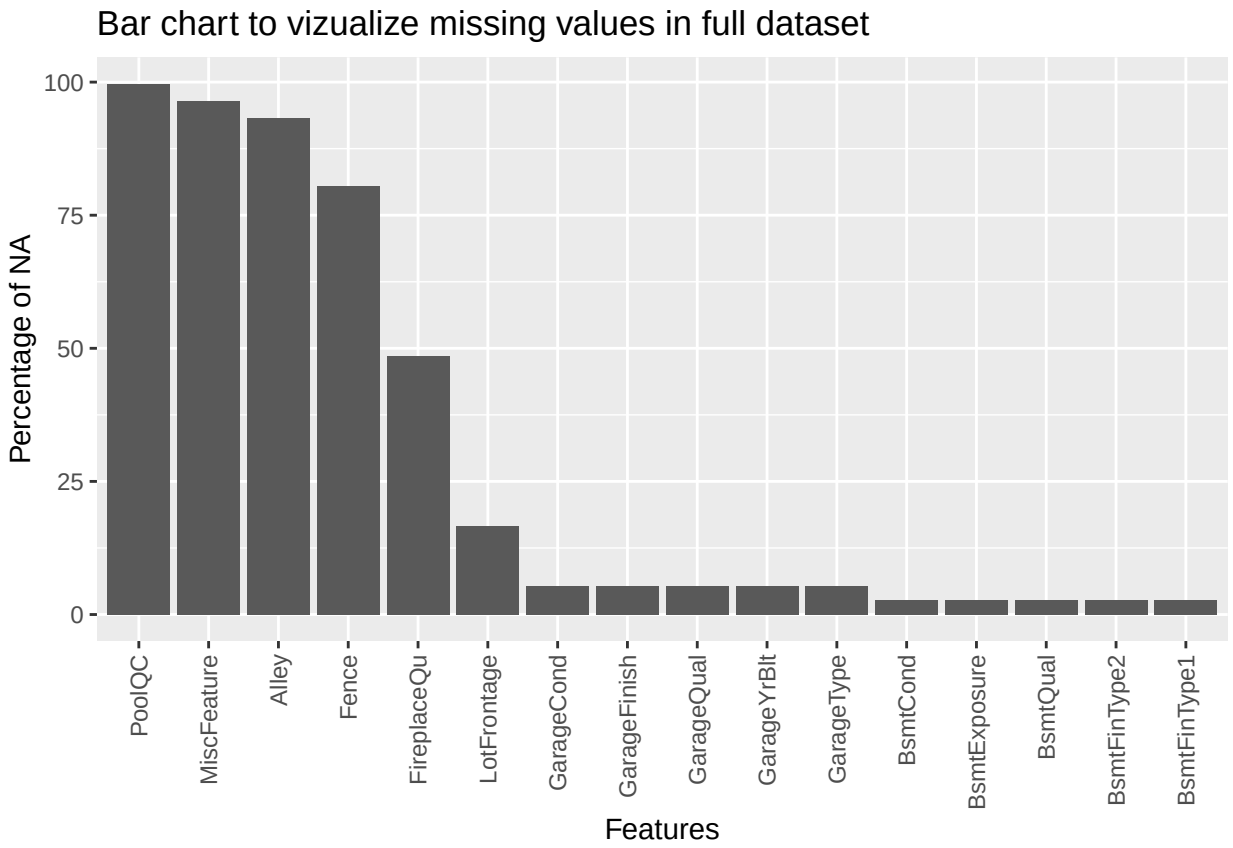
The Goal of this project is to perform an analysis on the houses data, and build a machine learning model to predict the price of houses based on historic data present in the trainging dataset. The performance of the model will be evaludated on the testing set of the dataset.

```
# Data Exploration and ETL ------------------------------------------------

# Reading the datafiles from data directory
train_df <- read.csv("data/train.csv", stringsAsFactors = TRUE)
test_df <- read.csv("data/test.csv", stringsAsFactors = TRUE)

# Combining traing and test data for analysis purposes
x_train_df <- select(train_df, -SalePrice)
y_train_df <- train_df[,"SalePrice"]

full_df <- bind_rows(x_train_df, test_df)
```

Exploratory Data Analysis (EDA)

We can start the analysis of the dataset by performing an EDA on the whole combined dataset. We will find out the missing values present in the dataset and comes up with a plan to handle them with an appropriate method suggested by the data analysis.

```
# Calculate NA's percentages in each column
na_percentages <- data.frame(Feature = colnames(full_df), Percentage_of_NA = colSums(is.na(full_df))/nr
rownames(na_percentages) <- NULL
na_percentages <- na_percentages %>% dplyr::filter(Percentage_of_NA > 1)

# Missing data vizualization
na_percentages <- na_percentages %>% arrange(-Percentage_of_NA)
ggplot(data = na_percentages, aes(x = reorder(Feature, -Percentage_of_NA), y = Percentage_of_NA)) +
  geom_bar(stat = "identity") +
  theme(axis.text.x = element_text(angle = 90, vjust = 0.5, hjust=1)) +
  labs(x = "Features", y = "Percentage of NA", title = "Bar chart to vizualize missing values in full da
```



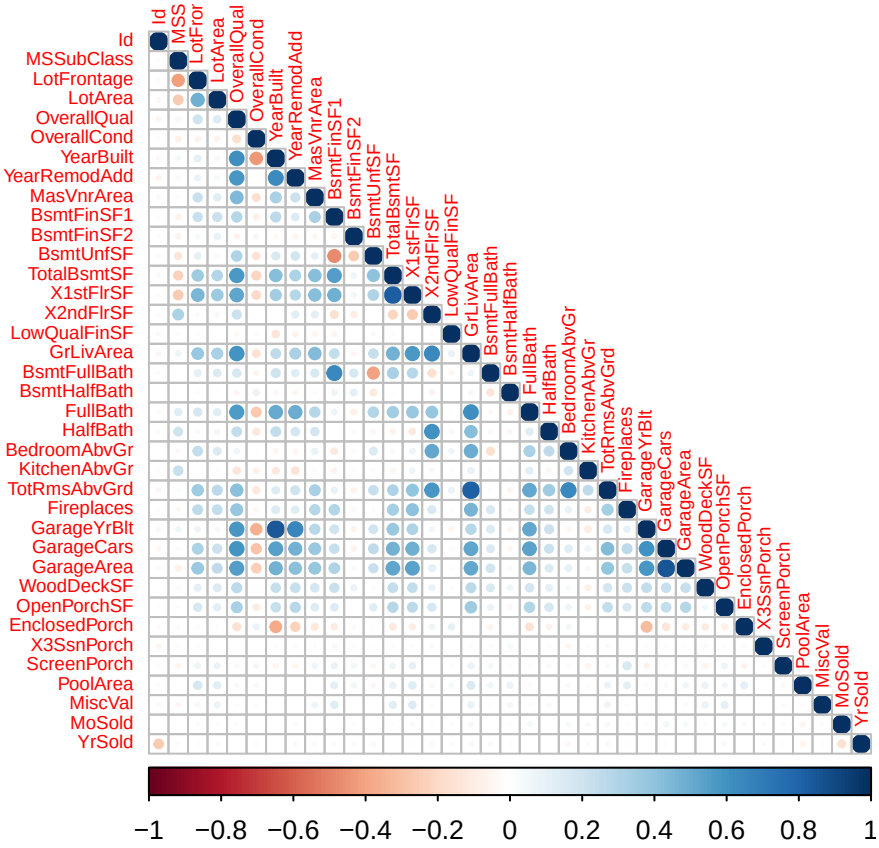
Insights :

We can observe from the above graph that some features like PoolQc , MiscFeature and Alley have more than 90% missing values . Such column will not be useful in predicting the model. We will remove features which have over 40% missing values.

```
# Removing columns with more than 40% NA's
full_df <- full_df[, !(colnames(full_df) %in% c("PoolQC", "MiscFeature", "Alley", "Fence", "FireplaceQu
```

```
# Separating the numeric and categorical variables to check their distribution
num_vars <- full_df[, sapply(full_df, is.numeric)]
catg_vars <- full_df[, !sapply(full_df, is.numeric)]
```

```
# Correlation plot for numeric vars
corrplot(cor(na.omit(num_vars)), method = "circle", tl.cex = 0.60 , type = "lower")
```



Insights

We can observe from the correlation plot above that there is multicollinearity . We remove features that have high correlated values such as YearRemodAdd, GarageYrBlt, GarageArea, GarageCond, TotalBsmtSF, TotalRmsAbvGrd and BsmtFinSF1

Feature Engineering

We need to build new features based on the exisiting information to reduce the sparsity in the data and feeding only the relevant data to the model.

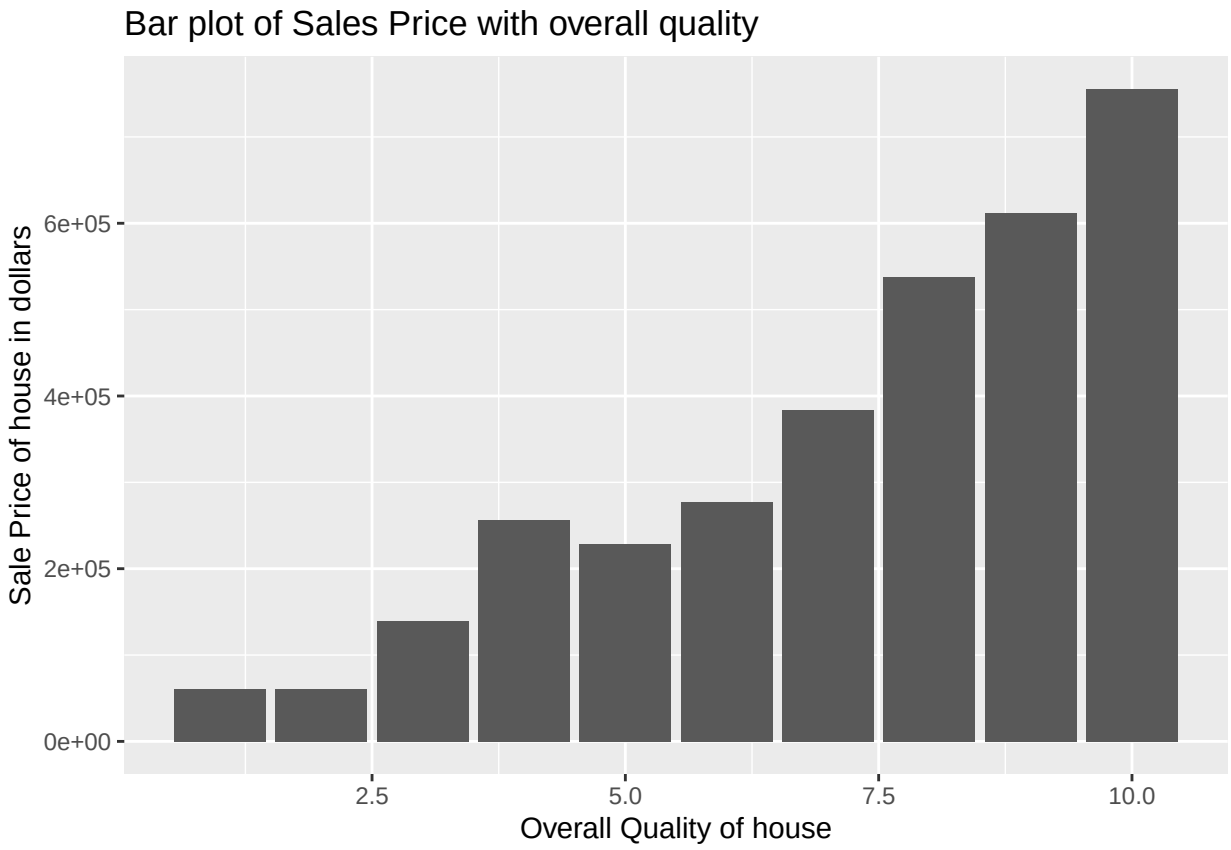
```
# Making binary feature of house Remodel
full_df$isRemodeledRecent = ifelse(full_df$YearRemodAdd == full_df$YrSold, 1, 0)

# Checking for new houses
full_df$new_house_status = ifelse(full_df$YrSold == full_df$YearBuilt, 1, 0)

# Feature for overall quality of houses
full_df$overallQualGood = ifelse(as.integer(full_df$OverallQual) - 5 < 0, 0, as.integer(full_df$Overall
full_df$overallQualGood = ifelse(5 - as.integer(full_df$OverallQual) < 0, 0, 5 - as.integer(full_df$Ove


# Finding Average room size of house
full_df$averageRoomSizeAbvGrd = full_df$GrLivArea / full_df$TotRmsAbvGrd

# Finding bathroom above grounds
full_df$bathRoomToRoomAbvGrd = (full_df$FullBath + full_df$HalfBath) / full_df$TotRmsAbvGrd
```

```
# Looking for Landscape interactions
full_df$landscapeInteraction = as.integer(full_df$LotShape) * as.integer(full_df$LandContour)

# Checking for number of garage in the house
full_df$garageInteraction = full_df$GarageCars * as.integer(full_df$GarageQual)

# Binning  New Dwelling in the house
full_df$isNewerDwelling = ifelse(full_df$MSSubClass %in% c(20,60,120),1,0)


# Looking for nearby transits and services of the house
full_df$isConditionNearRail = ifelse(full_df$Condition1 %in%
                                       c("RRAe","RRAn","RRNe","RRNn") | full_df$Condition2 %in%
                                       c("RRAe","RRAn","RRNe","RRNn"),1,0)
full_df$isConditionNearArtery  = ifelse(full_df$Condition1 == "Artery" | full_df$Condition2 == "Artery"
full_df$isConditionNearFeedr  = ifelse(full_df$Condition1 == "Feedr" | full_df$Condition2 == "Feedr",1,
full_df$isConditionNearPosFeature  = ifelse(full_df$Condition1 %in%
                                              c("PosA"," PosN") | full_df$Condition2 %in%
                                              c("PosA"," PosN"),1,0)

# Finding the house sell season of the house
full_df$soldHighSeason = ifelse(full_df$MoSold %in% c(5,6,7),1,0)

# Calculating score based on exterior quality of the house
full_df$scoreExterior = as.integer(full_df$ExterQual) * as.integer(full_df$ExterCond)

# Making a column with total number of bathrooms
full_df$no_of_baths <- full_df$FullBath + (full_df$HalfBath*0.5) + full_df$BsmtFullBath + (full_df$Bsmt

# Making a column with all variable related to poarch
full_df$total_poarch_area <- full_df$OpenPorchSF + full_df$EnclosedPorch + full_df$X3SsnPorch + full_df

# Calculating total sqaure feets
full_df$total_sq_feet <- full_df$GrLivArea + full_df$TotalBsmtSF

# Transforming Numeric columns to binary columns
full_df$MSSubClass <- ifelse(full_df$MSSubClass > 100, 1,0)
full_df$MSSubClass <- as.factor(full_df$MSSubClass)

full_df$OverallCond <- ifelse(full_df$OverallCond > 5, 1,0)
full_df$OverallCond <- as.factor(full_df$OverallCond)

full_df$MoSold <- ifelse(full_df$MoSold > 6, 1,0)
full_df$YrSold <- as.factor(full_df$YrSold)

full_df$YearBuilt <- ifelse(full_df$YearBuilt > 1940, 1,0)
full_df$MoSold <- as.factor(full_df$MoSold)

# Binning Neighborhood
full_df$neighborhood_prive_level[full_df$Neighborhood %in% c('StoneBr', 'NridgHt', 'NoRidge')] <- 2
full_df$neighborhood_prive_level[full_df$Neighborhood %in% c('MeadowV', 'IDOTRR', 'BrDale')] <- 0
full_df$neighborhood_prive_level[!full_df$Neighborhood %in% c('MeadowV', 'IDOTRR', 'BrDale', 'StoneBr',
```

```
# Removing columns which were used in feature engineering and high correlation
drop_cols <- c("Id","FullBath", "HalfBath", "BsmtFullBath", "BsmtHalfBath", "Neighborhood", "OpenPorchS
               "EnclosedPorch", "X3SsnPorch", "ScreenPorch", "GrLivArea", "TotalBsmtSF", "GarageArea",
               "TotRmsAbvGrd" , "GarageCond", "GarageYrBlt" , "YearRemodAdd", "BsmtFinSF1")
full_df <- full_df[, !(colnames(full_df) %in% drop_cols)]
```

PLOTS FOR BETTER VISUALIZING

```
# Bar plot of Sales Price with overall quality
ggplot(data = train_df, aes(x = OverallQual, y = SalePrice)) +
  geom_bar(stat = "identity", position = "dodge") +
  labs(x = "Overall Quality of house", y = "Sale Price of house in dollars", title = "Bar plot of Sales
```

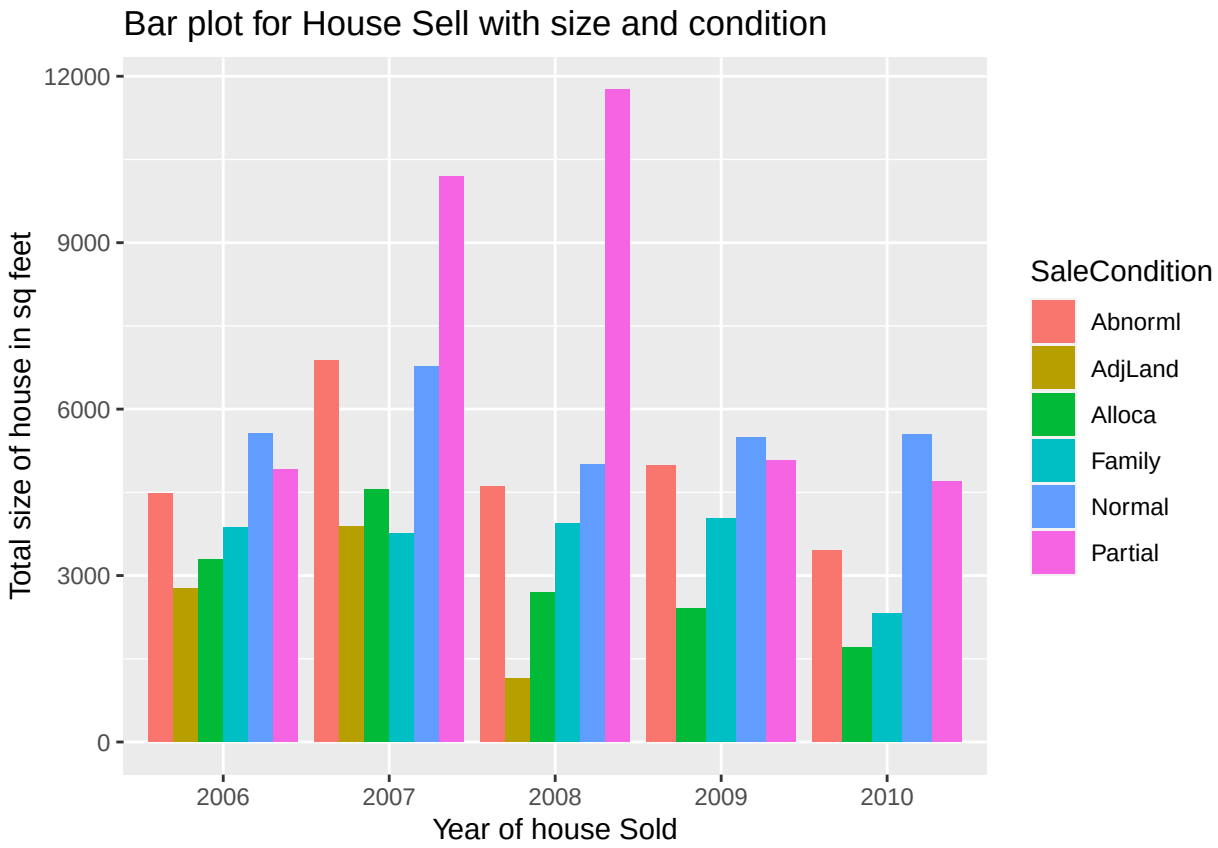


Insights

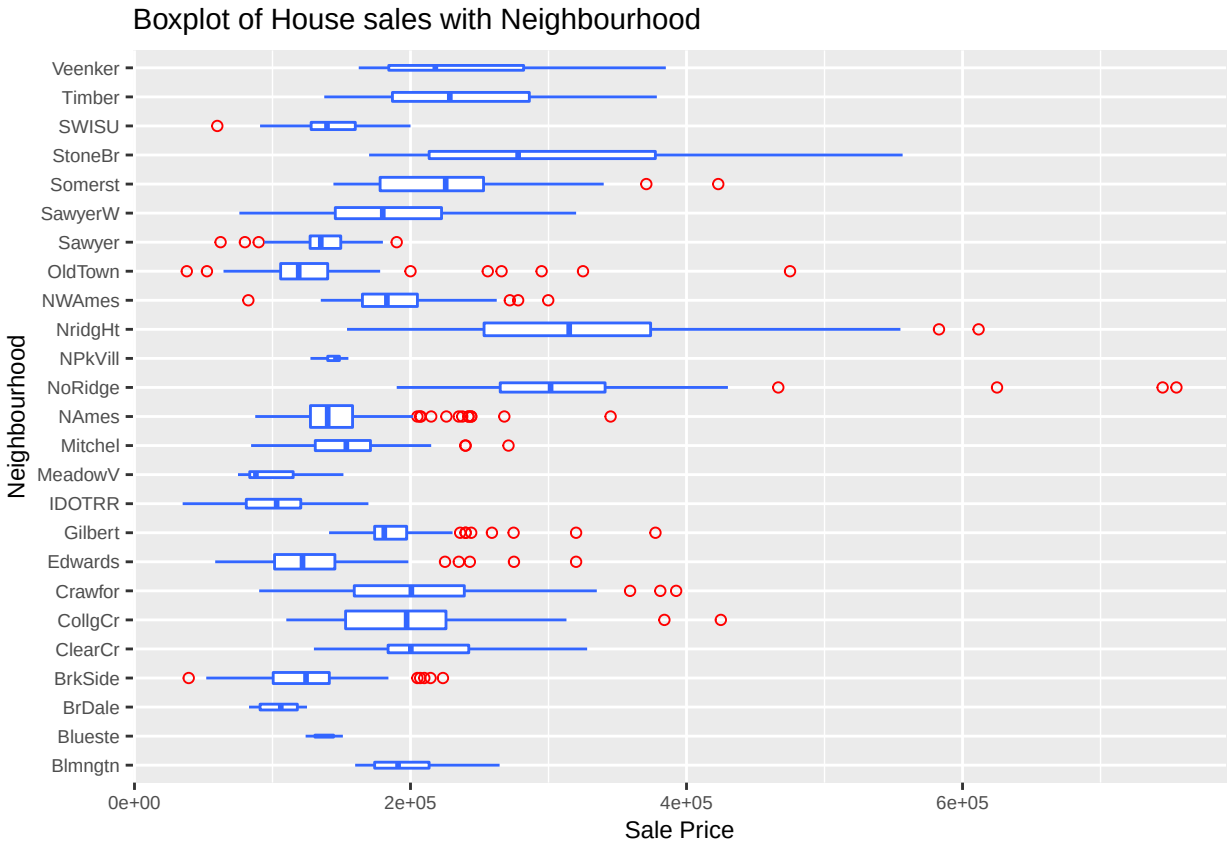
We can see that the with increase in SP of the house the quality of house also increases .

```
# Bar plot for House Sell with size and condition
ggplot(data = full_df, aes(x = YrSold, y = total_sq_feet, fill = SaleCondition )) +
  geom_bar(stat = "identity", position = "dodge") +
  labs(x = "Year of house Sold", y = "Total size of house in sq feet", title = "Bar plot for House Sell
```

```
## Warning: Removed 1 rows containing missing values (geom_bar).
```




```
# Boxplot of House sales with Neighbourhood
ggplot(data = train_df, aes(x = Neighborhood, y = SalePrice)) +
  geom_boxplot(varwidth = TRUE, fill = "white", colour = "#3366FF", outlier.colour = "red", outlier.sha
  theme(text = element_text(size=9)) +
  labs(x = "Neighbourhood", y = "Sale Price", title = "Boxplot of House sales with Neighbourhood") +
  coord_flip()
```



Insights

1.Neighborhoods like Veenkar,Timber,StoneBr,SwayerW etc have no outliers.

1.Neighborhoods like No Ridge have extreme outlier with SP greater than 600000.

```
# Scatter plot for Price vs House area above ground
ggplot(data = train_df, aes(x = GrLivArea, y = SalePrice)) +
  geom_point(shape=1) +
  geom_smooth(method=lm, se=FALSE) +
  labs(x = "Above ground living area", y = "Sale price",
       title = "Scatterplot for Price vs House area above ground") +
  theme(text = element_text(size=9))
```

```
## `geom_smooth()` using formula 'y ~ x'
```

Insights

1.SP increases with increase in area above ground

2.Most of the houses have less than 3000Sqft area above the ground

```
# Checking skewness and fixing using logarithms
num_df <- select_if(full_df, is.numeric)
skewedVariables <- sapply(num_df, function(x) ifelse(skew(x) > 0.75, x <- log1p(x), x))
skewedVariables <- skewedVariables[skewedVariables > 0.50]
skewed_df <- as.data.frame(subset(full_df, select = names(skewedVariables)))
skewed_df <- log1p(skewed_df)
rest_df <- full_df %>% select(-colnames(skewed_df))
full_df <- cbind(skewed_df, rest_df)
```

Data Pre Processing ***Imputing missing values***

```
# Using random forest imputation for missing values
randomImputation <- function (df_orignal) {
  df_fact <- df_orignal %>% select_if(is.factor)
  for(a in colnames(df_fact)){
    missing <- is.na(df_fact[a])
    n.missing <- sum(missing)
    a.obs <- df_fact[a][!missing]
    imputed <- df_fact[a]
```

```
    imputed[missing] <- sample(a.obs, n.missing, replace=TRUE)
    df_fact[a]<- imputed
  }
  df_num <- df_orignal %>% select_if(is.numeric)
  for(b in colnames(df_num)){
    missing <- is.na(df_num[b])
    n.missing <- nrow(df_num) - length(df_num[b][!missing])
    b.obs <- df_num[b][!missing]
    imputed <- df_num[b]
    min(b.obs)
    max(b.obs)
    c(min(b.obs):max(b.obs))
    imputed[missing] <- sample(x = min(b.obs):max(b.obs), size = n.missing,replace = TRUE)
    df_num[b]<- imputed
  }
  df <- cbind(df_fact,df_num)
  return (df)
}

full_df <- randomImputation(full_df)
```

Data standardization

```
# Separating the numeric and categorical variables before Standardization with sd = 1, mean = 0
scaled_variables <- c( "WoodDeckSF", "PoolArea", "MiscVal")
scaled_df <- full_df %>% select(all_of(scaled_variables))
scaled_df <- scale(scaled_df)
not_scaled <- full_df %>% select(-all_of(scaled_variables))
full_df <- cbind(scaled_df, not_scaled)

# Creating dummy variables for categorical data
dummy_df <- fastDummies::dummy_cols(.data = full_df, select_columns = colnames(select_if(full_df, is.fa
                                    remove_first_dummy = TRUE, ignore_na = TRUE, remove_selected_columns
```

Modeling Pre processing

```
# Train and Test split
x_train <- dummy_df[1:1460,]
x_test <- dummy_df[-(1:1460),]
y_train <- y_train_df

# Parameters for glmnet models
controlling_params <- trainControl(method = "cv", number = 7, savePredictions = "final",
                                   index = createResample(x_train$OverallQual, 7), allowParallel =TRUE)

# Using Ridge, Lasso and Elastic net regression
glmnet_lasso_grid <- expand.grid(.alpha = 1, .lambda = seq(0.001,0.1,by = 0.001))
glmnet_ridge_grid <- expand.grid(.alpha = 0, .lambda = seq(0.001,0.1,by = 0.001))
glmnet_elastic_grid <- expand.grid(.alpha = 0.3, .lambda = 0.009)

# Parameter Tuning for glm models
```

```
tunned_params <- list(glmnet=caretModelSpec(method="glmnet", tuneGrid = glmnet_lasso_grid),
                      glmnet=caretModelSpec(method="glmnet", tuneGrid = glmnet_ridge_grid),
                      glmnet = caretModelSpec(method="glmnet", tuneGrid = glmnet_elastic_grid))

# Running all the glmnet models together
models_list <- caretList(x = x_train, y = log(y_train), trControl = controlling_params, metric="RMSE", t

# Ensemble learning model with cross-validation
Ensemble_model <- caretEnsemble(models_list, metric="RMSE", trControl = trainControl(method = "cv", numb

# prediction on test set
finalPredictions <- predict(Ensemble_model, newdata=x_test)

# Taking exponential of prediction
finalPredictions <- expm1(finalPredictions)

# Writing CSV for submission
df <- data.frame("Id" = test_df$Id, "SalePrice" = finalPredictions)
names(df) <- c("Id", "SalePrice")
write.csv(df, file = "y_predicted.csv", row.names = F)
```