Use Case Report

IE 7275 Data Mining in Engineering

House Price Prediction

Group 19

Akshay Anandbabu

Srinath Selvadurai

617-991-0856

857-318-9109

anandbabu.a@husky.neu.edu

selvadurai.s@husky.neu.edu

Signature of Student 1: <u>AKSHAY ANANDBABU</u>

Signature of Student 2: <u>SRINATH SELVADURAI</u>

Submission Date: 12/08/2017

TABLE OF CONTENTS

Selection Rationale:

In recent years, there is a surge in online services that act as an interface between buyers and sellers. This change has also reflected in the housing industry with websites like Zillow and Homesnap. Due to the increasing user base, more people have started to use these platforms as their first step in finding a house. These circumstances have led to the collection of troves of information relating to houses and their attributes. Further, these structured databases provide an opportunity to explore and interpret the nature of this economic sector.

Problem Statement:

• Explore a dataset and identify the important predictor variables that affect housing prices.

• Visualize the data to find any existing trends or patterns

• Create a predictive model that can estimate the price of houses by applying data mining and pattern recognition algorithms

Data Collection:

The dataset was obtained from the publicly available datasets on www.kaggle.com. It is adapted from the Ames Housing Dataset originally compiled by Dean De Cock. The data set contains 2930 observations and a large number of explanatory variables (23 nominal, 23 ordinal, 14 discrete, and 19 continuous) involved in assessing home values and one continuous response variable i.e. price. The data set was already partitioned into training, test and validation datasets.

Solution Design:

- As there are explanatory variables and response variables, some form supervised learning can be used to create a predictive model for the housing prices.
- As there are many categorical variables, they must be converted to numerical variables using methods like one hot encoding.
- The newly created variables would lead to an explosion in the dimensionality, so some form of data reduction would also be necessary.
- Data visualization techniques need to be performed to process the data from its raw form. It is necessary to check if there is any need for normalization or scaling and if there are any missing values that need to be imputed.
- Data visualization would also be needed to identify any meaningful pattern in the data as well as help in the selection of the data mining technique.

Metadata:

MSSubClass: Identifies the type of dwelling involved in the sale.

20	1-STORY 1946 & NEWER ALL STYLES
30	1-STORY 1945 & OLDER
40	1-STORY W/FINISHED ATTIC ALL AGES
45	1-1/2 STORY - UNFINISHED ALL AGES
50	1-1/2 STORY FINISHED ALL AGES
60	2-STORY 1946 & NEWER
70	2-STORY 1945 & OLDER
75	2-1/2 STORY ALL AGES
80	SPLIT OR MULTI-LEVEL
85	SPLIT FOYER
90	DUPLEX - ALL STYLES AND AGES
120	1-STORY PUD (Planned Unit Development) - 1946 & NEWER
150	1-1/2 STORY PUD - ALL AGES
160	2-STORY PUD - 1946 & NEWER
180	PUD - MULTILEVEL - INCL SPLIT LEV/FOYER
190	2 FAMILY CONVERSION - ALL STYLES AND AGES

MSZoning: Identifies the general zoning classification of the sale.

A	Agriculture
C	Commercial
FV	Floating Village Residential
I	Industrial
RH	Residential High Density
RL	Residential Low Density

RP Residential Low Density Park

RM Residential Medium Density

LotFrontage: Linear feet of street connected to property

LotArea: Lot size in square feet

Street: Type of road access to property

Grvl Gravel

Pave Paved

Alley: Type of alley access to property

Grvl Gravel

Pave Paved

NA No alley access

LotShape: General shape of property

Reg Regular

IR1 Slightly irregular

IR2 Moderately Irregular

IR3 Irregular

LandContour: Flatness of the property

Lvl Near Flat/Level

Bnk Banked - Quick and significant rise from street grade to building

HLS Hillside - Significant slope from side to side

Low Depression

Utilities: Type of utilities available

AllPub All public Utilities (E,G,W,& S)

NoSewr Electricity, Gas, and Water (Septic Tank)

NoSeWa Electricity and Gas Only

ELO Electricity only

LotConfig: Lot configuration

Inside Inside lot

Corner Corner lot

CulDSac Cul-de-sac

FR2 Frontage on 2 sides of property

FR3 Frontage on 3 sides of property

LandSlope: Slope of property

Gtl Gentle slope

Mod Moderate Slope

Sev Severe Slope

Neighborhood: Physical locations within Ames city limits

Blmngtn Bloomington Heights

Blueste Bluestem

BrDale Briardale

BrkSide Brookside

ClearCr Clear Creek

CollgCr College Creek

Crawfor Crawford

Edwards Edwards

Gilbert Gilbert

IDOTRR Iowa DOT and Rail Road

MeadowV Meadow Village

Mitchel Mitchell

Names North Ames

NoRidge Northridge

NPkVill Northpark Villa

NridgHt Northridge Heights

NWAmes Northwest Ames

OldTown Old Town

SWISU South & West of Iowa State University

Sawyer Sawyer

SawyerW Sawyer West

Somerst Somerset

StoneBr Stone Brook

Timber Timberland

Veenker Veenker

Condition1: Proximity to various conditions

Artery Adjacent to arterial street

Feedr Adjacent to feeder street

Norm Normal

RRNn Within 200' of North-South Railroad

RRAn Adjacent to North-South Railroad

PosN Near positive off-site feature--park, greenbelt, etc.

PosA Adjacent to postive off-site feature

RRNe	Within 200' of East-West Railroad

RRAe	Adjacent to East-West Railroad

Condition2: Proximity to various conditions (if more than one is present)

Artery	Adjacent to arterial street

Feedr	Adjacent to feeder street

Norm	Normal

RRNn	Within 200' of North-South Railroad

RRAn	Adjacent to North-South Railroad

PosN	Near positive off-site feature--park, greenbelt, etc.

PosA	Adjacent to postive off-site feature

RRNe	Within 200' of East-West Railroad

RRAe	Adjacent to East-West Railroad

BldgType: Type of dwelling

1Fam	Single-family Detached

2FmCon	Two-family Conversion; originally built as one-family dwelling

Duplx	Duplex

TwnhsE	Townhouse End Unit

TwnhsI	Townhouse Inside Unit

HouseStyle: Style of dwelling

1Story	One story

1.5Fin	One and one-half story: 2nd level finished

1.5Unf	One and one-half story: 2nd level unfinished

2Story	Two story

2.5Fin	Two and one-half story: 2nd level finished

2.5Unf	Two and one-half story: 2nd level unfinished
SFoyer	Split Foyer
SLvl	Split Level

OverallQual: Rates the overall material and finish of the house

10	Very Excellent
9	Excellent
8	Very Good
7	Good
6	Above Average
5	Average
4	Below Average
3	Fair
2	Poor
1	Very Poor

OverallCond: Rates the overall condition of the house

10	Very Excellent
9	Excellent
8	Very Good
7	Good
6	Above Average
5	Average
4	Below Average
3	Fair
2	Poor
1	Very Poor

YearBuilt: Original construction date

YearRemodAdd: Remodel date (same as construction date if no remodeling or additions)

RoofStyle: Type of roof

Flat	Flat

Gable	Gable

Gambrel	Gabrel (Barn)

Hip	Hip

Mansard	Mansard

Shed	Shed

RoofMatl: Roof material

ClyTile	Clay or Tile

CompShg	Standard (Composite) Shingle

Membran	Membrane

Metal	Metal

Roll	Roll

Tar&Grv	Gravel & Tar

WdShake	Wood Shakes

WdShngl	Wood Shingles

Exterior1st: Exterior covering on house

AsbShng	Asbestos Shingles

AsphShn	Asphalt Shingles

BrkComm	Brick Common

BrkFace	Brick Face

CBlock Cinder Block

CemntBd Cement Board

HdBoard Hard Board

ImStucc Imitation Stucco

MetalSd Metal Siding

Other   Other

Plywood Plywood

PreCast PreCast

Stone   Stone

Stucco Stucco

VinylSd Vinyl Siding

Wd Sdng Wood Siding

WdShing Wood Shingles

Exterior2nd: Exterior covering on house (if more than one material)

AsbShng Asbestos Shingles

AsphShn Asphalt Shingles

BrkComm Brick Common

BrkFace Brick Face

CBlock Cinder Block

CemntBd Cement Board

HdBoard Hard Board

ImStucc Imitation Stucco

MetalSd Metal Siding

Other   Other

Plywood Plywood

PreCast PreCast

Stone   Stone

Stucco Stucco

VinylSd Vinyl Siding

Wd Sdng Wood Siding

WdShing Wood Shingles

MasVnrType: Masonry veneer type

BrkCmn Brick Common

BrkFace Brick Face

CBlock Cinder Block

None None

Stone Stone

MasVnrArea: Masonry veneer area in square feet

ExterQual: Evaluates the quality of the material on the exterior

Ex Excellent

Gd Good

TA Average/Typical

Fa Fair

Po Poor

ExterCond: Evaluates the present condition of the material on the exterior

Ex Excellent

Gd Good

TA Average/Typical

Fa Fair

Po Poor

Foundation: Type of foundation

BrkTil Brick & Tile

CBlock Cinder Block

PConc Poured Contrete

Slab Slab

Stone Stone

Wood Wood

BsmtQual: Evaluates the height of the basement

Ex Excellent (100+ inches)

Gd Good (90-99 inches)

TA Typical (80-89 inches)

Fa Fair (70-79 inches)

Po Poor (<70 inches

NA No Basement

BsmtCond: Evaluates the general condition of the basement

Ex Excellent

Gd Good

TA Typical - slight dampness allowed

Fa Fair - dampness or some cracking or settling

Po Poor - Severe cracking, settling, or wetness

NA No Basement

BsmtExposure: Refers to walkout or garden level walls

Gd Good Exposure

Av	Average Exposure (split levels or foyers typically score average or above)

Mn	Mimimum Exposure

No	No Exposure

NA	No Basement

BsmtFinType1: Rating of basement finished area

GLQ	Good Living Quarters

ALQ	Average Living Quarters

BLQ	Below Average Living Quarters

Rec	Average Rec Room

LwQ	Low Quality

Unf	Unfinshed

NA	No Basement

BsmtFinSF1: Type 1 finished square feet

BsmtFinType2: Rating of basement finished area (if multiple types)

GLQ	Good Living Quarters

ALQ	Average Living Quarters

BLQ	Below Average Living Quarters

Rec	Average Rec Room

LwQ	Low Quality

Unf	Unfinshed

NA	No Basement

BsmtFinSF2: Type 2 finished square feet

BsmtUnfSF: Unfinished square feet of basement area

TotalBsmtSF: Total square feet of basement area

Heating: Type of heating

Floor	Floor Furnace
GasA	Gas forced warm air furnace
GasW	Gas hot water or steam heat
Grav	Gravity furnace
OthW	Hot water or steam heat other than gas
Wall	Wall furnace

HeatingQC: Heating quality and condition

Ex	Excellent
Gd	Good
TA	Average/Typical
Fa	Fair
Po	Poor

CentralAir: Central air conditioning

N	No
Y	Yes

Electrical: Electrical system

SBrkr	Standard Circuit Breakers & Romex
FuseA	Fuse Box over 60 AMP and all Romex wiring (Average)
FuseF	60 AMP Fuse Box and mostly Romex wiring (Fair)

FuseP	60 AMP Fuse Box and mostly knob & tube wiring (poor)
Mix	Mixed

1stFlrSF: First Floor square feet

2ndFlrSF: Second floor square feet

LowQualFinSF: Low quality finished square feet (all floors)

GrLivArea: Above grade (ground) living area square feet

BsmtFullBath: Basement full bathrooms

BsmtHalfBath: Basement half bathrooms

FullBath: Full bathrooms above grade

HalfBath: Half baths above grade

Bedroom: Bedrooms above grade (does NOT include basement bedrooms)

Kitchen: Kitchens above grade

KitchenQual: Kitchen quality

Ex	Excellent
Gd	Good
TA	Typical/Average
Fa	Fair
Po	Poor

TotRmsAbvGrd: Total rooms above grade (does not include bathrooms)

Functional: Home functionality (Assume typical unless deductions are warranted)

Typ Typical Functionality

Min1 Minor Deductions 1

Min2 Minor Deductions 2

Mod Moderate Deductions

Maj1 Major Deductions 1

Maj2 Major Deductions 2

Sev Severely Damaged

Sal Salvage only

Fireplaces: Number of fireplaces

FireplaceQu: Fireplace quality

Ex Excellent - Exceptional Masonry Fireplace

Gd Good - Masonry Fireplace in main level

TA Average - Prefabricated Fireplace in main living area or Masonry Fireplace in basement

Fa Fair - Prefabricated Fireplace in basement

Po Poor - Ben Franklin Stove

NA No Fireplace

GarageType: Garage location

2Types More than one type of garage

Attchd Attached to home

Basment Basement Garage

BuiltIn Built-In (Garage part of house - typically has room above garage)

CarPort Car Port

Detchd Detached from home

NA No Garage

GarageYrBlt: Year garage was built

GarageFinish: Interior finish of the garage

Fin Finished

RFn Rough Finished

Unf Unfinished

NA No Garage

GarageCars: Size of garage in car capacity

GarageArea: Size of garage in square feet

GarageQual: Garage quality

Ex Excellent

Gd Good

TA Typical/Average

Fa Fair

Po Poor

NA No Garage

GarageCond: Garage condition

Ex	Excellent
Gd	Good
TA	Typical/Average
Fa	Fair
Po	Poor
NA	No Garage

PavedDrive: Paved driveway

Y	Paved
P	Partial Pavement
N	Dirt/Gravel

WoodDeckSF: Wood deck area in square feet

OpenPorchSF: Open porch area in square feet

EnclosedPorch: Enclosed porch area in square feet

3SsnPorch: Three season porch area in square feet

ScreenPorch: Screen porch area in square feet

PoolArea: Pool area in square feet

PoolQC: Pool quality

Ex	Excellent
Gd	Good
TA	Average/Typical

Fa Fair

NA No Pool

Fence: Fence quality

GdPrv Good Privacy

MnPrv Minimum Privacy

GdWo Good Wood

MnWw Minimum Wood/Wire

NA No Fence

MiscFeature: Miscellaneous feature not covered in other categories

Elev Elevator

Gar2 2nd Garage (if not described in garage section)

Othr Other

Shed Shed (over 100 SF)

TenC Tennis Court

NA None

MiscVal: $Value of miscellaneous feature

MoSold: Month Sold (MM)

YrSold: Year Sold (YYYY)

SaleType: Type of sale

WD Warranty Deed - Conventional

CWD Warranty Deed - Cash

VWD Warranty Deed - VA Loan

New Home just constructed and sold

COD Court Officer Deed/Estate

Con Contract 15% Down payment regular terms

ConLw Contract Low Down payment and low interest

ConLI Contract Low Interest

ConLD Contract Low Down

Oth Other

SaleCondition: Condition of sale

Normal Normal Sale

Abnorml Abnormal Sale - trade, foreclosure, short sale

AdjLand Adjoining Land Purchase

Alloca Allocation - two linked properties with separate deeds, typically condo with a garage unit

Family Sale between family members

Partial Home was not completed when last assessed (associated with New Homes)

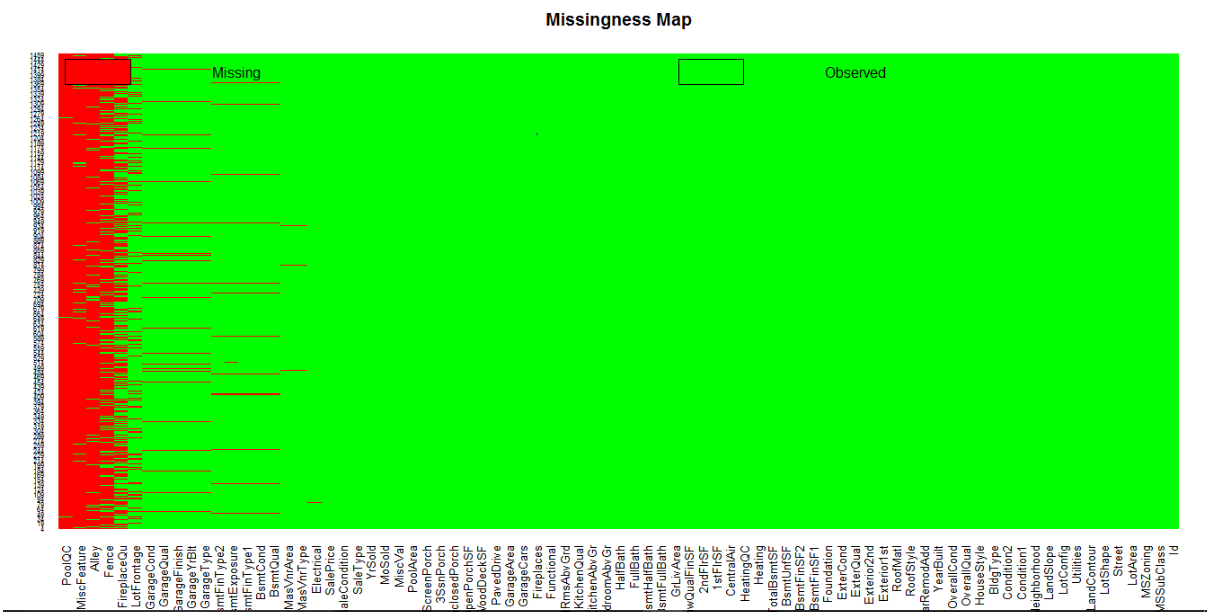
DATA VISUALIZATION AND PROCESSING

We create a missing value map to identify the features with the highest count of missing values. Combining this with our knowledge of the metadata, we can safely omit these variables

train<-read_csv("ProjTrain.csv")

train<-setDT(train)

missmap(train[-1], col=c('red', 'green'), y.cex=0.5, x.cex=0.8)



Also, finding the total percentage of the missing values in the dataset

```
sum(is.na(train)) / (nrow(train) *ncol(train))
[1] 0.0355991
```

We removed these features based on our inferences

train<-train[,-c("PoolQC","MiscFeature")]

We split the dataset into categorical and numerical:

cat_var <- names(train)[which(sapply(train, is.character))]

cat_car <- c(cat_var, 'BedroomAbvGr', 'HalfBath', ' KitchenAbvGr','BsmtFullBath', 'BsmtHalfBath', 'MSSubClass')

numeric_var <- names(train)[which(sapply(train, is.numeric))]

We convert all the categorical variables into factors to facilitate visualization

train[,(cat_var) := lapply(.SD, as.factor), .SDcols = cat_var]

train_cat <- train[,.SD, .SDcols = cat_var]

train_cont <- train[,.SD,.SDcols = numeric_var]

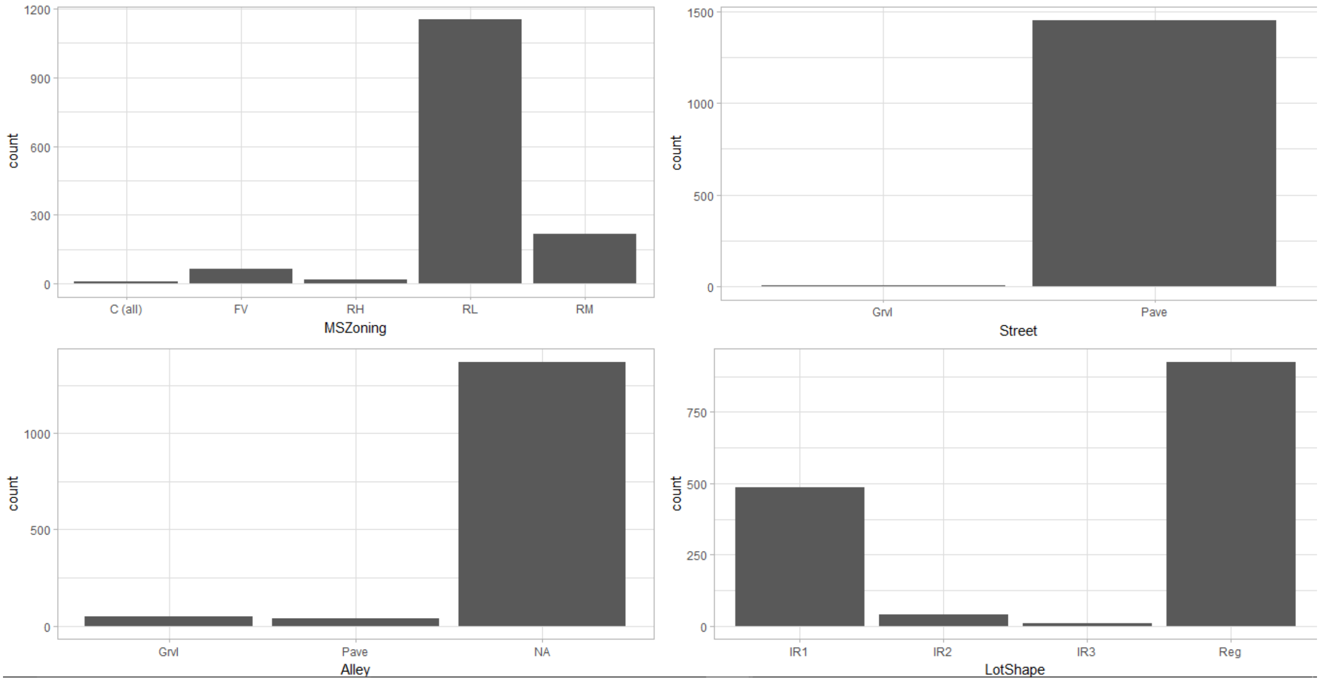
Function definitions for creating multiple grid plots. We defined two functions to create a bar plot and density plot respectively and a third function to arrange to call the other functions and display them in a grid.

```
bar_plot <- function(dataset, i)
  {
  data <- data.frame(x=dataset[[i]])
  g <- ggplot(data=data, aes(x=factor(x))) +
    stat_count() +
    xlab(colnames(dataset)[i]) +
    theme_light()
  return (g)
}
```

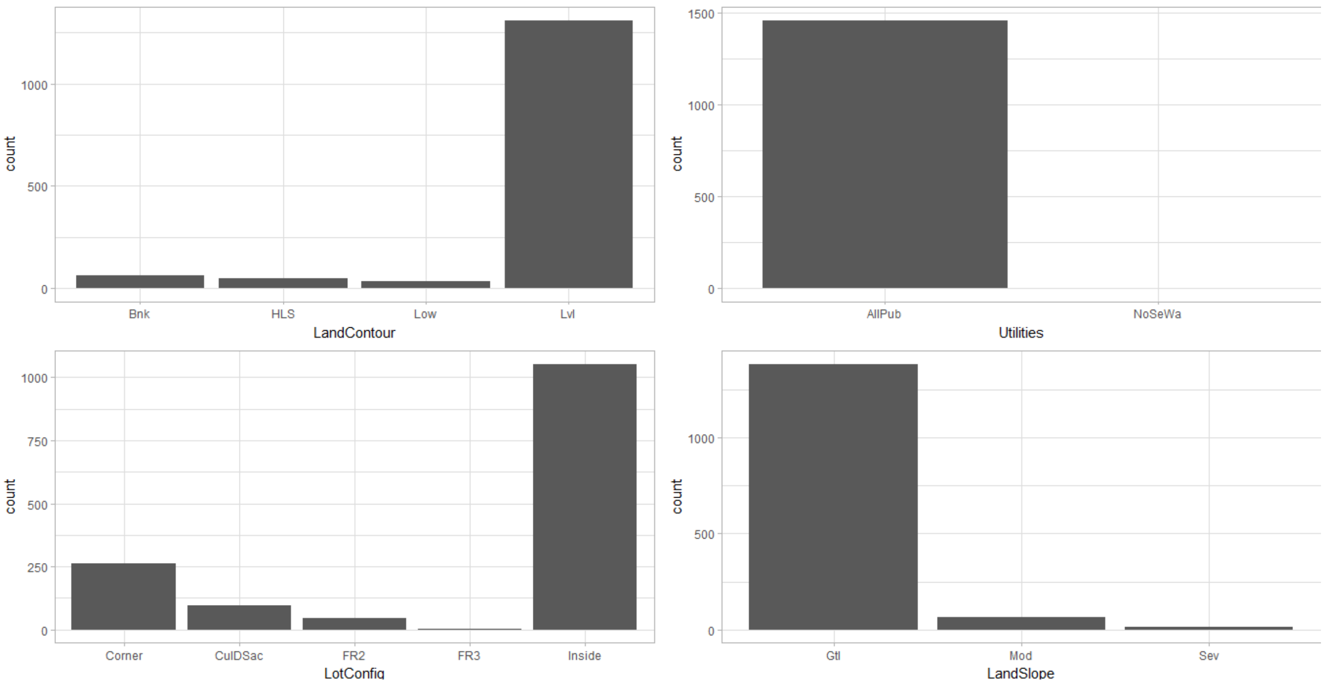
```
plot_arrange <- function(dataset, fun, ii, ncol=3) {
  gg<- list()
  for (i in ii) {
    t <- fun(dataset=dataset, i=i)
    gg <- c(gg, list(t))
  }
  do.call("grid.arrange", c(gg, ncol=ncol))
}
density_plot <- function(dataset, i)
  {
  data <- data.frame(x=dataset[[i]], SalePrice = dataset$SalePrice)
  g <- ggplot(data) +
    geom_line(aes(x = x), stat = 'density', size = 1,alpha = 1.0) +
    xlab(paste0((colnames(dataset)[i])))
  return(g)
}
```

BARPLOTS FOR MAJOR CATEGORICAL VARIABLES

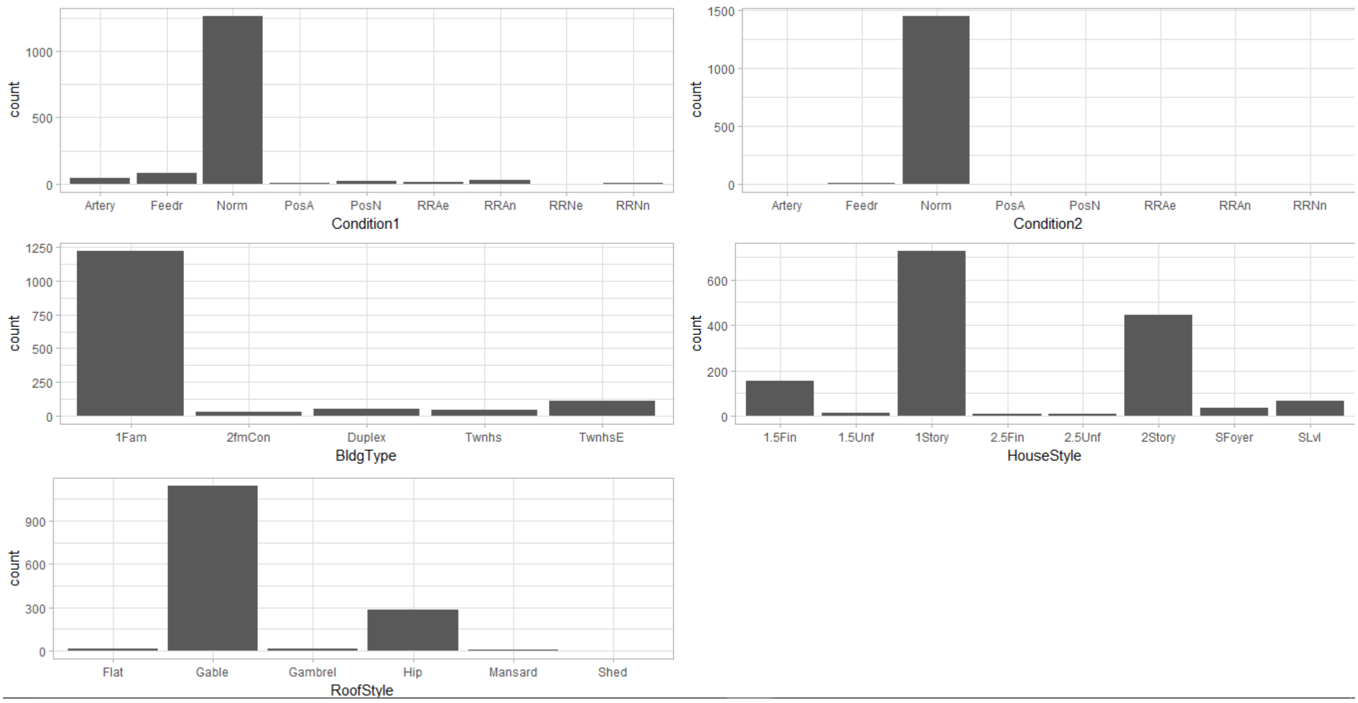
plot_arrange(train_cat, fun = bar_plot, ii = 1:4, ncol = 2)



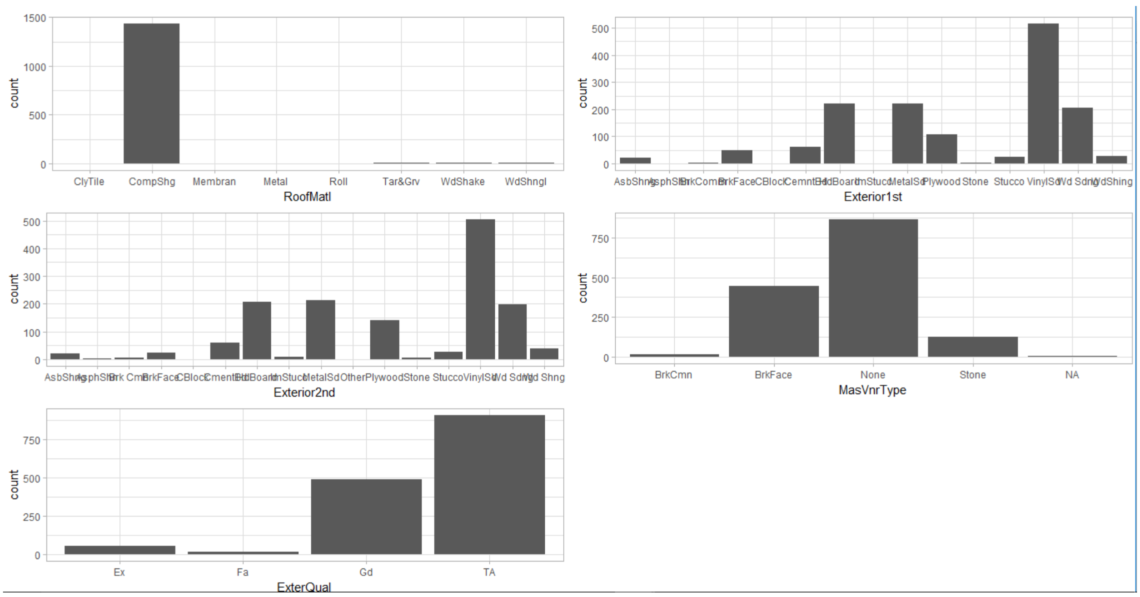
plot_arrange(train_cat, fun = bar_plot, ii = 5:8, ncol = 2)



plot_arrange(train_cat, fun = bar_plot, ii = 15:19, ncol = 2)



plot_arrange(train_cat, fun = bar_plot, ii = 1:4, ncol = 2)



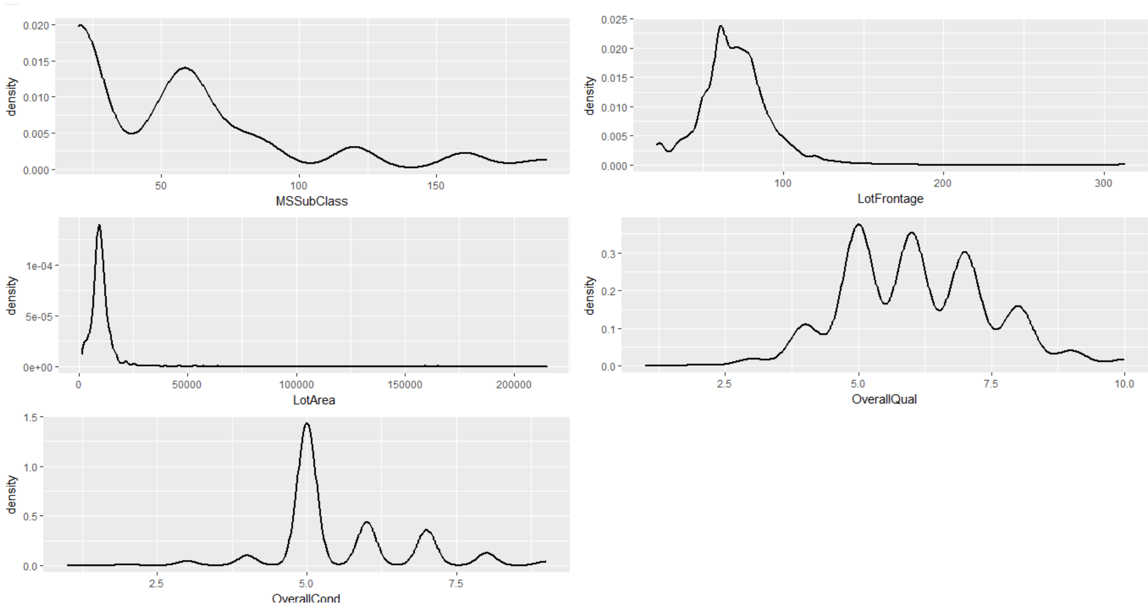
It is evident that for most of the categorical variables, there is a clear majority for one type of category implying that there is a possibility to group categories together into a single attribute

For example:

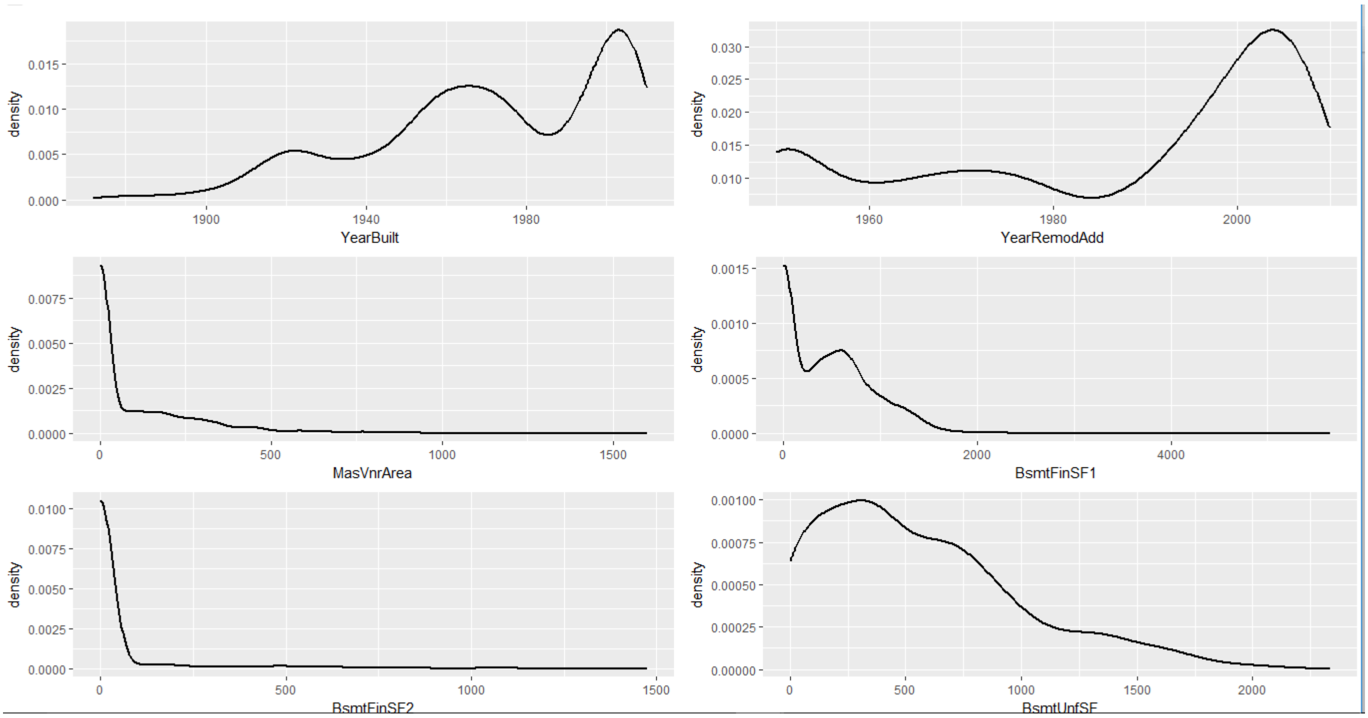
- most roof materials are composite shingles
- most building types are single family detached (1fam)

DENSITY PLOTS FOR CONTINUOUS VARIABLES

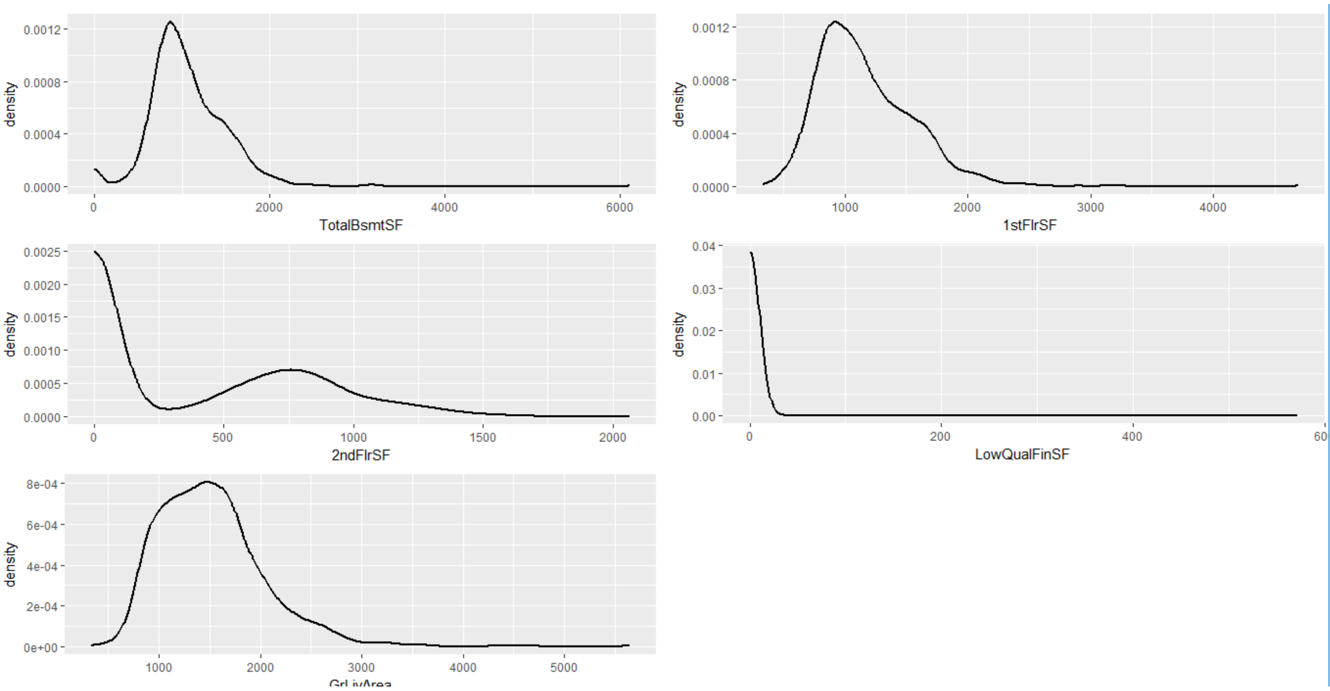
plot_arrange(train_cont, fun = density_plot, ii = 2:6, ncol = 2)



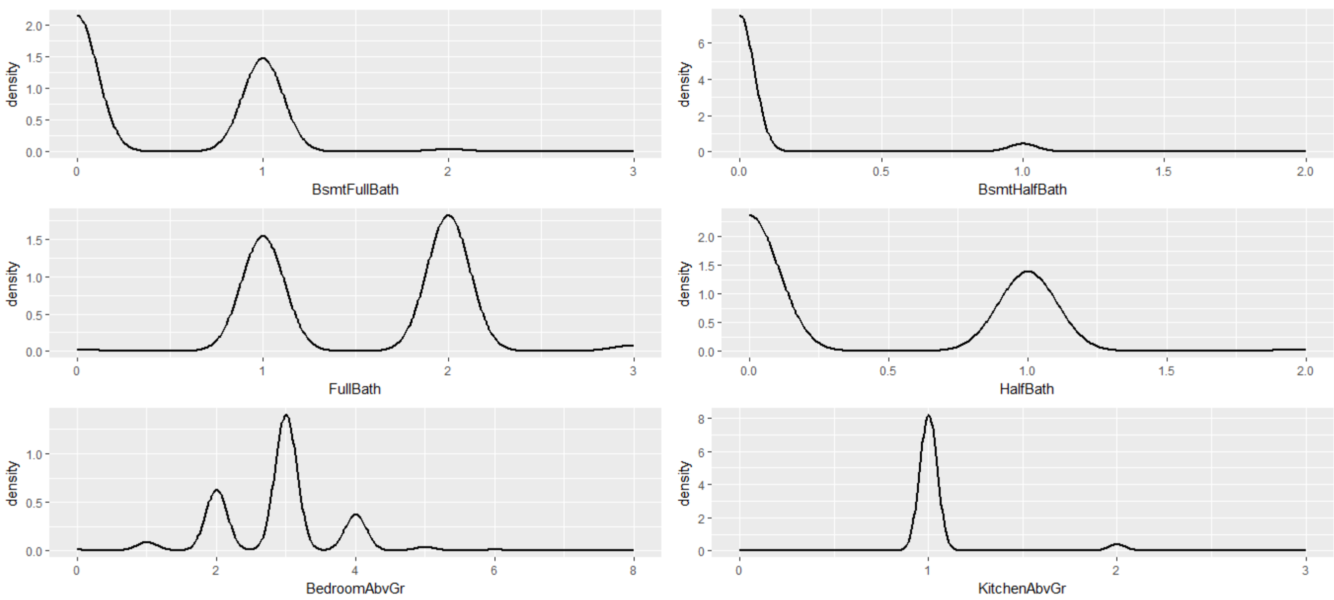
plot_arrange(train_cont, fun = density_plot, ii = 7:12, ncol = 2)



plot_arrange(train_cont, fun = density_plot, ii = 13:17, ncol = 2)



plot_arrange(train_cont, fun = density_plot, ii = 18:23, ncol = 2)



The density plots are skewed, suggesting transformation using either scaling or taking log functions.

CORRELATION PLOT:

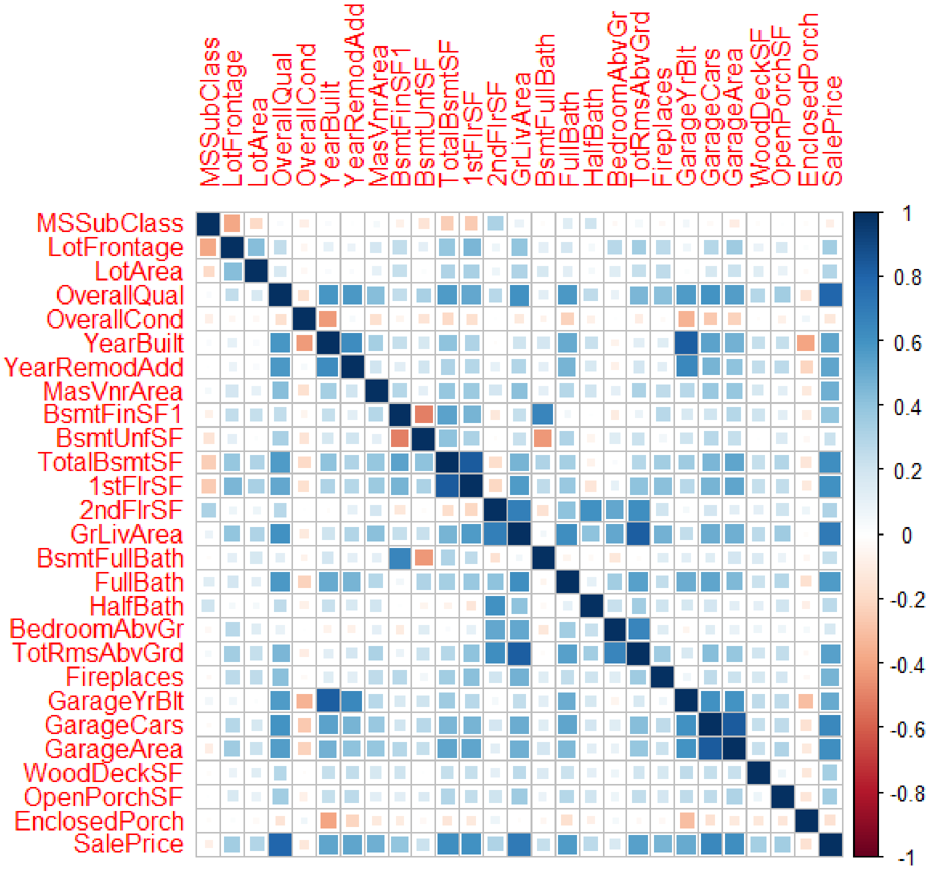
corr <- cor(na.omit(train_cont[,-1, with = FALSE]))

corr<- cor(na.omit(train_cont[,-1, with = FALSE]))

row_indic <- apply (corr, 1, function(x) sum (x > 0.3 | x < -0.3) > 1)

corr<- corr[row_indic ,row_indic]

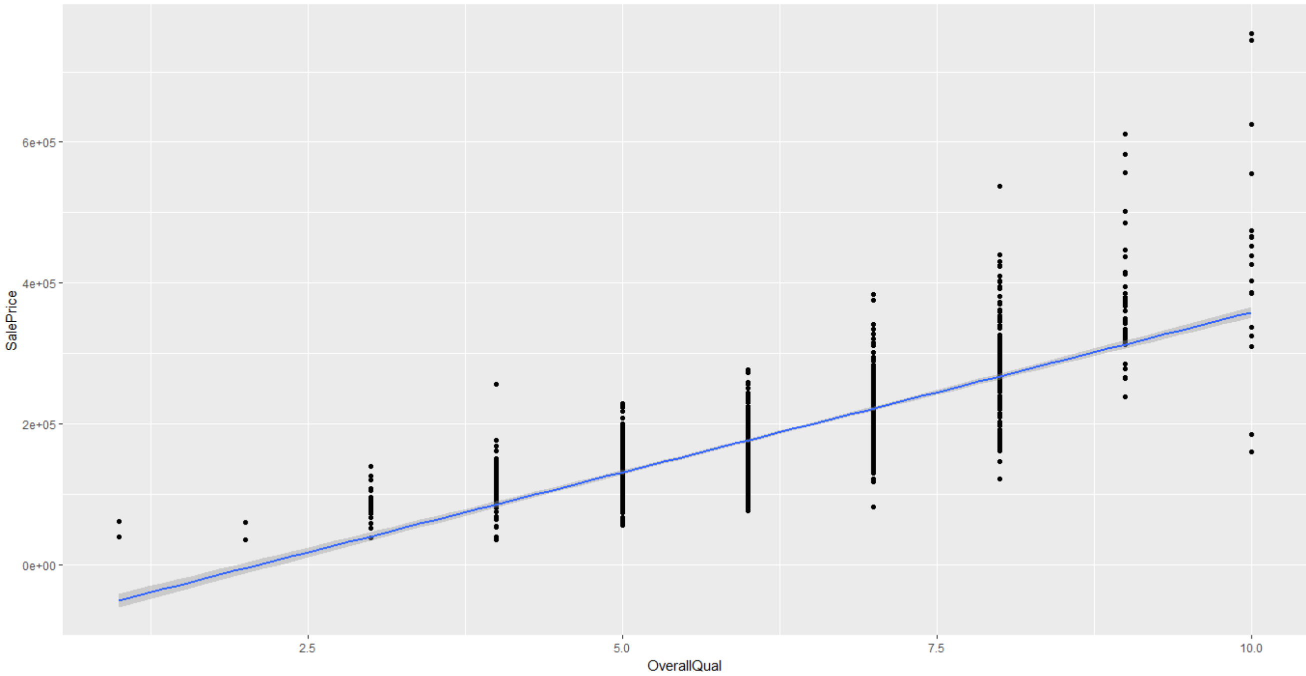
corrplot::corrplot(corr, method="square")



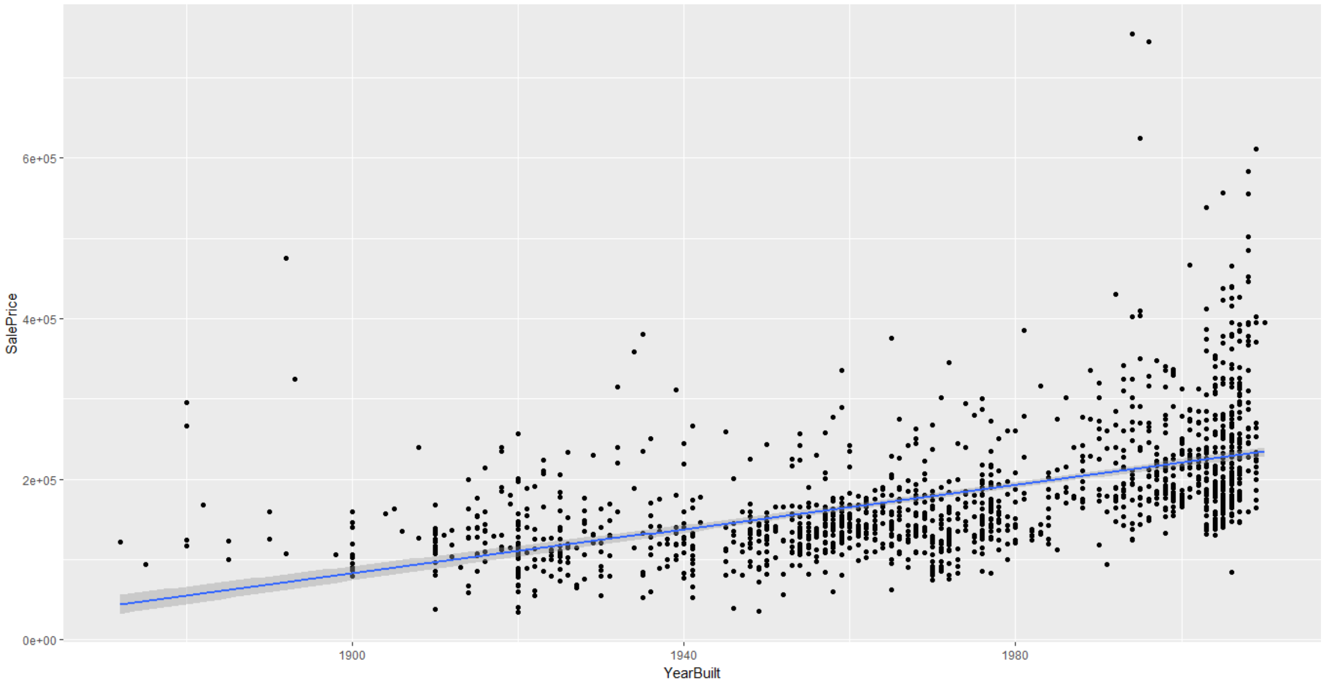
The correlation plots depict the interaction between the variables

SCATTERPLOTS FOR VARIABLES WITH HIGH CORRELATION

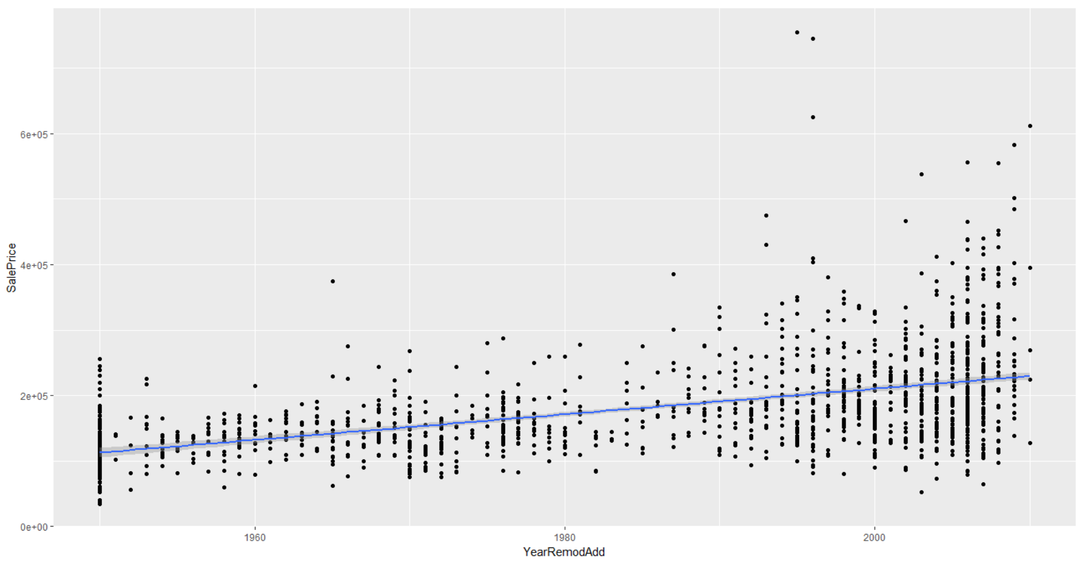
```
train %>%
ggplot(aes(OverallQual,SalePrice))+
  geom_point()+
  geom_smooth(method = lm)
```



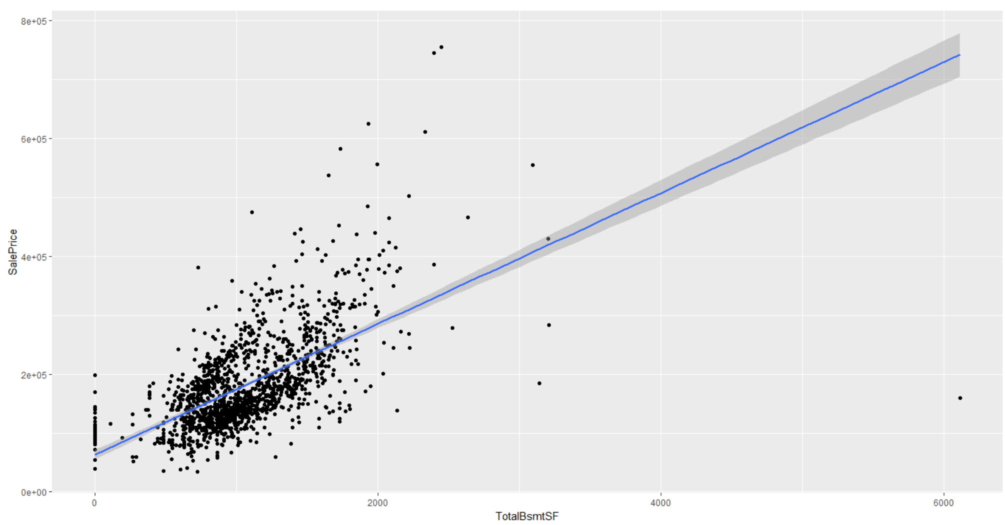
```
train %>%
  ggplot(aes(YearBuilt,SalePrice))+
  geom_point()+
  geom_smooth(method = lm)
```



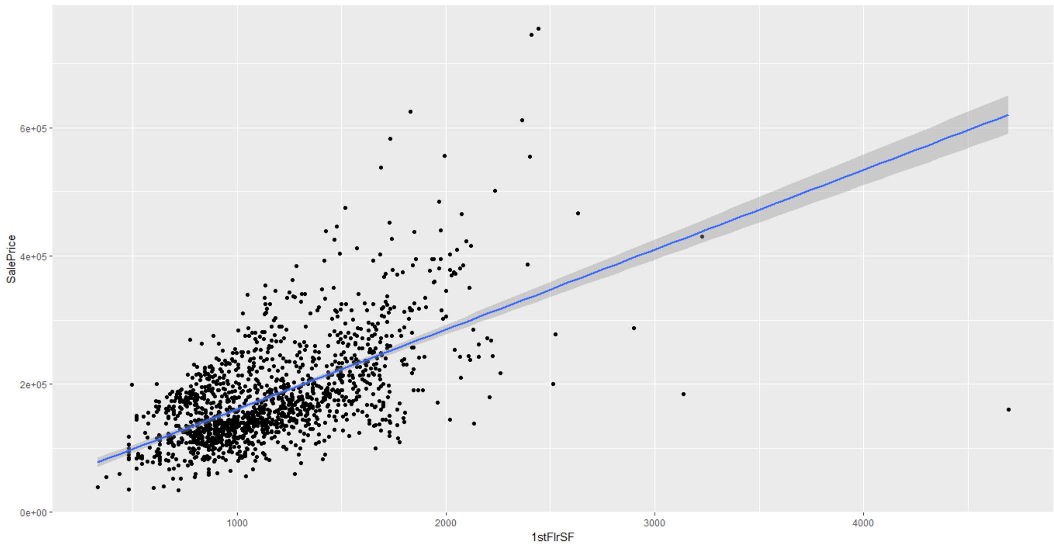
```
train %>%
  ggplot(aes(YearRemodAdd,SalePrice))+
  geom_point()+
  geom_smooth(method = lm)
```



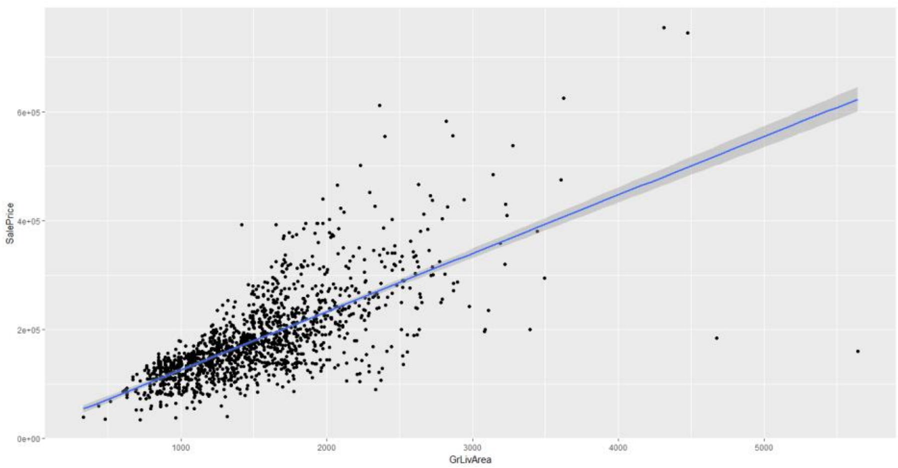

```
train %>%
  ggplot(aes(TotalBsmtSF,SalePrice))+
  geom_point()+
  geom_smooth(method = lm)
```



```r
train %>%
  ggplot(aes(`1stFlrSF`,SalePrice))+
  geom_point()+
  geom_smooth(method = lm)
```




```r
train %>%
  ggplot(aes(GrLivArea,SalePrice))+
  geom_point()+
  geom_smooth(method = lm)
```



- From the scatter plots we can observe that newer (either built or remodelled recently) and larger houses demand a higher sales price in the market.
- These correlation lines suggest that there exists a linear relationship between these predictor variables and response variables which might be adequately captured in a linear regression model

Data Mining Techniques:

1.Feature Selection and Extraction :

We are combining the training and test sets,removing the SalePrice which is what we are predicting. We remove Id as it has nothing to do with house pricing

df.all <- rbind(within(train, rm('Id','SalePrice','PoolQC','MiscFeature','Alley')), within(test, rm('Id','SalePrice','PoolQC','MiscFeature','Alley')))

dim(df.all)

```
[1] 2920   76
```

We see that poolQC, MiscFeatures, Alley have maximum missing values so we remove them from our model. Also we personally believe that the quality of the pool and alley does not drive major price changes (from missingness map)

We now segregate the numeric variables and categorical variables to perform transformation on these variables

num_feat<- names(which(sapply(df.all, is.numeric)))

cat_feat <- names(which(sapply(df.all, is.character)))

df.numeric <- df.all[num_feat]

2. Principal Component Analysis

screeplot<-fa.parallel(df.numeric, fa = "pc",show.legend = F, n.iter = 100, main = "Scree Plot")+

abline(h=1)

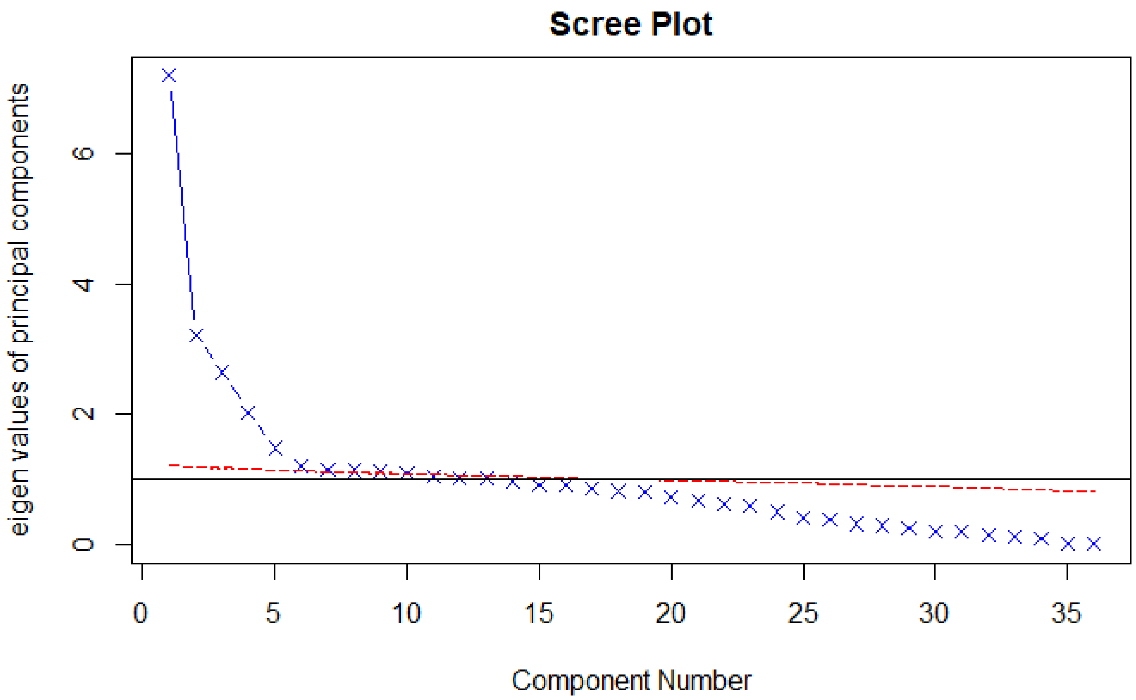
principal(df.numeric, nfactors = 12, rotate = "varimax")

We see that we require 12 principal components to capture maximum variance and we might use this information to obtain scores while calculating model performance later

```
Matrix was not positive definite, smoothing was donePrincipal Components
Analysis
Call: principal(r = df.numeric, nfactors = 12, rotate = "varimax")
Standardized loadings (pattern matrix) based upon correlation matrix
                RC1   RC2   RC3   RC4   RC5  RC12   RC9   RC8   RC7   RC6
RC11  RC10   h2    u2 com
MSSubClass     0.01  0.27 -0.70  0.16  0.31 -0.04 -0.02  0.07  0.05  0.02
-0.05  0.04 0.70 0.300 1.9
LotFrontage    0.21  0.15  0.77  0.05  0.02  0.00  0.01  0.14 -0.05 -0.01
-0.03 -0.08 0.70 0.300 1.4
LotArea        0.06  0.13  0.61  0.19  0.06 -0.02 -0.02 -0.01  0.09  0.03
0.04  0.15 0.46 0.539 1.5
OverallQual    0.82  0.22  0.02  0.01 -0.11  0.08  0.12  0.03  0.00  0.04
0.02 -0.02 0.76 0.238 1.3
OverallCond   -0.16  0.03 -0.06  0.01 -0.24  0.36  0.03  0.00  0.33 -0.08
0.64 -0.02 0.74 0.259 2.8
```

```
YearBuilt       0.73 -0.01 -0.05  0.09 -0.12 -0.49 -0.11 -0.08 -0.08  0.00
-0.15  0.00 0.85 0.153 2.1
YearRemodAdd    0.72  0.05 -0.15 -0.01 -0.19 -0.15 -0.17  0.05  0.04 -0.06
0.31  0.00 0.73 0.271 1.9
MasVnrArea      0.45  0.19  0.06  0.14  0.06  0.09  0.23 -0.19  0.15 -0.07
-0.29 -0.18 0.51 0.493 4.5
BsmtFinSF1      0.30 -0.11  0.18  0.84  0.04  0.05  0.11  0.05  0.09  0.00
-0.05 -0.11 0.87 0.127 1.5
BsmtFinSF2     -0.10 -0.07  0.18  0.08 -0.01 -0.16  0.10  0.10  0.13 -0.06
0.09  0.69 0.60 0.398 1.6
BsmtUnfSF       0.41 -0.11  0.13 -0.76  0.12  0.15  0.08  0.02 -0.11  0.02
-0.06 -0.08 0.85 0.149 2.0
TotalBsmtSF     0.69 -0.25  0.39  0.13  0.16  0.14  0.24  0.11  0.03  0.00
-0.08  0.05 0.83 0.171 2.7
X1stFlrSF       0.63 -0.20  0.47  0.11  0.31  0.19  0.23  0.10  0.08  0.01
-0.05  0.05 0.88 0.120 3.5
X2ndFlrSF       0.10  0.93 -0.11 -0.07  0.02  0.07 -0.02  0.06 -0.02  0.03
0.05 -0.03 0.91 0.093 1.1
LowQualFinSF   -0.07  0.04 -0.06 -0.10  0.05  0.06  0.03  0.71  0.01 -0.05
0.02  0.03 0.53 0.470 1.1
GrLivArea       0.54  0.63  0.24  0.02  0.25  0.20  0.16  0.19  0.04  0.03
0.01  0.02 0.92 0.082 3.3
BsmtFullBath    0.17 -0.15  0.08  0.80  0.02  0.00  0.00  0.04 -0.18 -0.05
0.04  0.13 0.75 0.247 1.4
BsmtHalfBath   -0.05 -0.01  0.01 -0.03 -0.02 -0.09  0.04  0.02  0.85  0.08
-0.06 -0.01 0.74 0.260 1.1
FullBath        0.64  0.34  0.02 -0.15  0.32 -0.08 -0.08  0.09 -0.01  0.04
0.04  0.01 0.68 0.323 2.4
HalfBath        0.13  0.73 -0.08  0.07 -0.24 -0.17  0.09 -0.07 -0.06 -0.02
-0.08 -0.01 0.68 0.317 1.6
BedroomAbvGr    0.01  0.60  0.29 -0.22  0.36  0.05 -0.01  0.12  0.08  0.02
0.04  0.03 0.65 0.354 2.7
KitchenAbvGr   -0.12  0.06 -0.10  0.00  0.84 -0.02 -0.06 -0.04 -0.05  0.00
0.03 -0.03 0.74 0.257 1.1
TotRmsAbvGrd    0.37  0.64  0.25 -0.12  0.40  0.14  0.07  0.13  0.02  0.01
0.05  0.02 0.83 0.167 3.2
Fireplaces      0.33  0.25  0.24  0.19 -0.07  0.23  0.39 -0.05  0.11  0.06
-0.05  0.09 0.51 0.492 5.5
GarageYrBlt     0.76  0.01 -0.14  0.00 -0.10 -0.43 -0.20  0.02 -0.13 -0.01
-0.08 -0.01 0.85 0.148 2.0
GarageCars      0.74  0.17  0.17  0.05 -0.02 -0.10  0.01 -0.14 -0.03  0.04
-0.13 -0.03 0.66 0.342 1.4
GarageArea      0.72  0.11  0.24  0.11 -0.03 -0.05  0.02 -0.09 -0.04  0.03
-0.12 -0.04 0.63 0.374 1.5
WoodDeckSF      0.35  0.11  0.11  0.19 -0.04  0.04 -0.28 -0.04  0.27 -0.03
0.04  0.37 0.48 0.517 4.9
OpenPorchSF     0.34  0.19  0.04  0.01 -0.12 -0.05  0.22  0.21 -0.18  0.22
0.17  0.04 0.38 0.624 5.9
EnclosedPorch -0.19  0.04  0.01 -0.03 -0.02  0.70 -0.15  0.08 -0.13  0.00
-0.06  0.01 0.57 0.426 1.4
X3SsnPorch      0.02 -0.04  0.16  0.04  0.00 -0.20 -0.05  0.09  0.20 -0.04
0.18 -0.62 0.54 0.463 1.9
ScreenPorch    -0.02  0.06 -0.01  0.00 -0.04 -0.14  0.83  0.04  0.02 -0.02
0.08  0.08 0.73 0.273 1.1
PoolArea        0.02  0.07  0.15  0.17 -0.05  0.01  0.00  0.60  0.02  0.05
-0.07 -0.02 0.43 0.570 1.4
MiscVal        -0.05  0.01  0.07  0.04  0.15 -0.13  0.07 -0.06 -0.16  0.03
0.56 -0.05 0.40 0.602 1.6
MoSold          0.04  0.00  0.02 -0.02  0.08 -0.01  0.02 -0.17  0.03  0.74
0.20  0.01 0.63 0.373 1.3
YrSold          0.01 -0.02  0.01  0.03  0.06 -0.02  0.02 -0.17 -0.05 -0.73
0.20  0.04 0.61 0.386 1.3

                       RC1  RC2  RC3  RC4  RC5 RC12  RC9  RC8  RC7  RC6
RC11 RC10
SS loadings           6.09 3.20 2.43 2.28 1.66 1.44 1.34 1.22 1.18 1.18
1.17 1.12
Proportion Var        0.17 0.09 0.07 0.06 0.05 0.04 0.04 0.03 0.03 0.03
0.03 0.03
```

```
Cumulative Var         0.17 0.26 0.33 0.39 0.44 0.48 0.51 0.55 0.58 0.61
0.64 0.68
Proportion Explained  0.25 0.13 0.10 0.09 0.07 0.06 0.06 0.05 0.05 0.05
0.05 0.05
Cumulative Proportion 0.25 0.38 0.48 0.58 0.64 0.70 0.76 0.81 0.86 0.91
0.95 1.00

Mean item complexity =  2.2
Test of the hypothesis that 12 components are sufficient.

The root mean square of the residuals (RMSR) is  0.05
 with the empirical chi square  10543.04  with prob <  0

Fit based upon off diagonal values = 0.94
```



Function that maps a categoric value to its corresponding numeric value and returns that column to the data frame

map.fcn <- function(cols, map.list, df){

for (col in cols){

df[col] <- as.numeric(map.list[df.all[,col]])

}

return(df)

}

finding all QUAL columns to convert catergorical into numeric vars

Qual.cols <- c('ExterQual', 'ExterCond', 'GarageQual', 'GarageCond', 'FireplaceQu', 'KitchenQual', 'HeatingQC', 'BsmtQual')

Qual.list <- c('None' = 0, 'Po' = 1, 'Fa' = 2, 'TA' = 3, 'Gd' = 4, 'Ex' = 5)

df.numeric <- map.fcn(Qual.cols, Qual.list, df.numeric)

Converting Basement cols

bsmt.list <- c('None' = 0, 'No' = 1, 'Mn' = 2, 'Av' = 3, 'Gd' = 4)

df.numeric = map.fcn(c('BsmtExposure'), bsmt.list, df.numeric)

BsmtFinType1 and BsmtFinType2

bsmt.fin.list <- c('None' = 0, 'Unf' = 1, 'LwQ' = 2,'Rec'= 3, 'BLQ' = 4, 'ALQ' = 5, 'GLQ' = 6)

df.numeric <- map.fcn(c('BsmtFinType1','BsmtFinType2'), bsmt.fin.list, df.numeric)

Home Functionality rating

functional.list <- c('None' = 0, 'Sal' = 1, 'Sev' = 2, 'Maj2' = 3, 'Maj1' = 4, 'Mod' = 5, 'Min2' = 6, 'Min1' = 7, 'Typ'= 8)

df.numeric['Functional'] <- as.numeric(functional.list[df.all$Functional])

Garage Finish

garage.fin.list <- c('None' = 0,'Unf' = 1, 'RFn' = 1, 'Fin' = 2)

df.numeric['GarageFinish'] <- as.numeric(garage.fin.list[df.all$GarageFinish])

Fence

fence.list <- c('None' = 0, 'MnWw' = 1, 'GdWo' = 1, 'MnPrv' = 2, 'GdPrv' = 4)

df.numeric['Fence'] <- as.numeric(fence.list[df.all$Fence])

MSdwell.list <- c('20' = 1, '30'= 0, '40' = 0, '45' = 0,'50' = 0, '60' = 1, '70' = 0, '75' = 0, '80' = 0, '85' = 0, '90' = 0, '120' = 1, '150' = 0, '160' = 0, '180' = 0, '190' = 0)

df.numeric['NewerDwelling'] <- as.numeric(MSdwell.list[as.character(df.all$MSSubClass)])

We now find correlations between the variables and sales price to see which features have the highest correlation with sales price

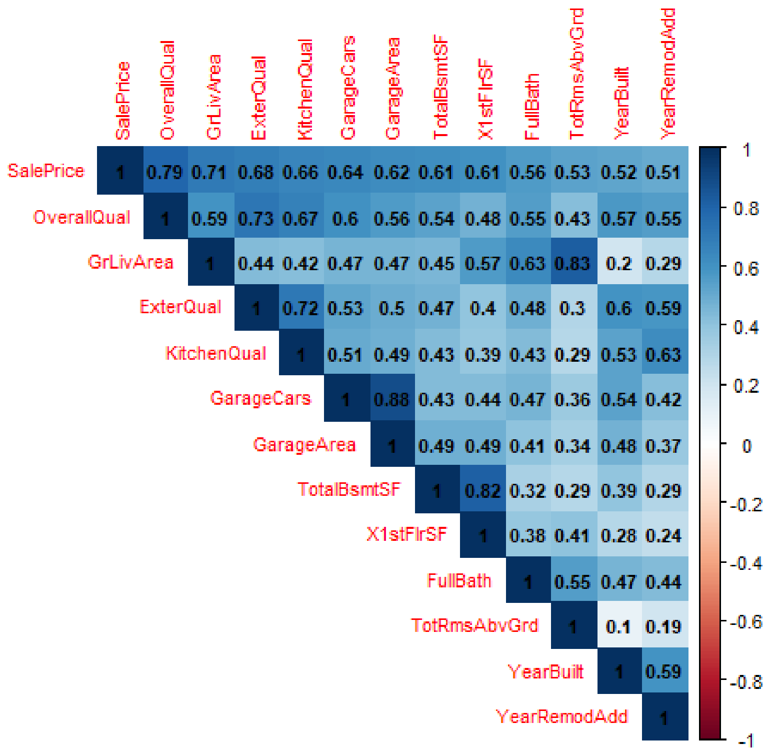
corr.df <- cbind(df.numeric[1:1460,], train['SalePrice']) #we check on the training dataset

correlation<-cor(corr.df)

corr.Max <- as.matrix(sort(correlation[,'SalePrice'], decreasing = TRUE))

corr.idx <- names(which(apply(corr.Max, 1, function(x) (x > 0.5 | x < -0.5)))) # correlation greater than 0.5 in either direction

corrplot(as.matrix(correlation[corr.idx,corr.idx]), type = 'upper', method='color', addCoef.col = 'black', tl.cex = .7,cl.cex = .7, number.cex=.7)



We see that the 12 variables impact the sales prices the most. They are OverallQual, GrLivArea, ExternalQual, KitchenQual, GarageCars,GarageArea,TotalBsmtSF, X1stFlrSF, FullBath, TotRmsAbvGrd, YearBuilt, YearRemodAdd

3. Normalizing numeric variables and creating dummies for categorical variables(OHE)

scaler <- preProcess(df.numeric)

```
df.numeric <- predict(scaler, df.numeric)

dummy <- dummyVars(" ~ .",data=df.all[,cat_feat])

df.categoric <- data.frame(predict(dummy,newdata=df.all[,cat_feat]))
```

Combining numeric and cat variables:

```
df.total<-cbind(df.numeric, df.categoric)

head(df.total)
```

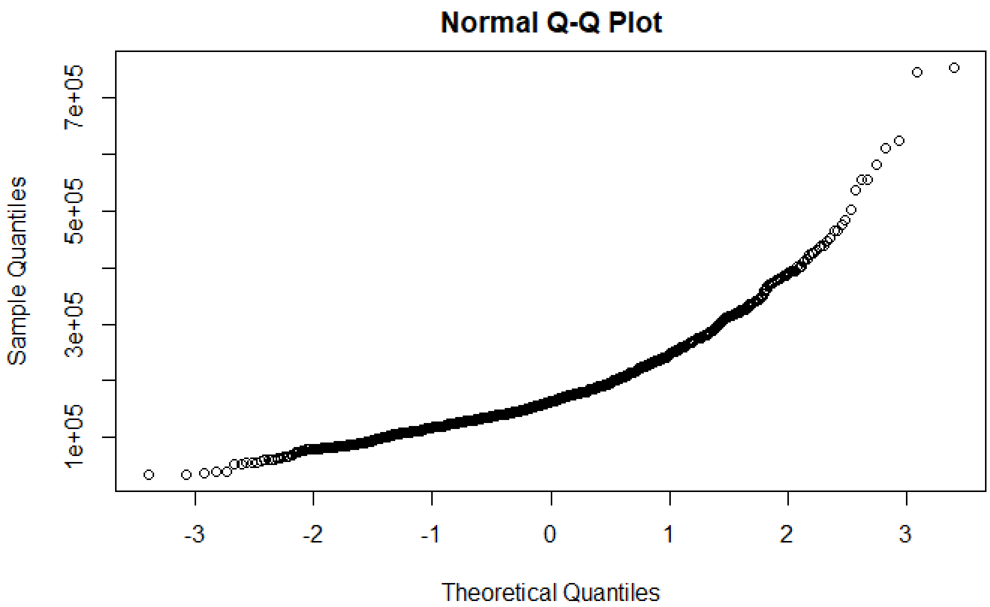
The following code shows the head of the combined table after normalizing and one hot encoding

	MSSubClass <dbl>	LotFrontage <dbl>	LotArea <dbl>	OverallQual <dbl>	OverallCond <dbl>	YearBuilt <dbl>
1	0.0733624	-0.20799102	-0.20710624	0.65136768	-0.5171112	1.0508138
2	-0.8724133	0.40980919	-0.09187064	-0.07182381	2.1792545	0.1567069
3	0.0733624	-0.08443098	0.07346740	0.65136768	-0.5171112	0.9845837
4	0.3098063	-0.41392442	-0.09688088	0.65136768	-0.5171112	-1.8633125
5	0.0733624	0.57455591	0.37508405	1.37455917	-0.5171112	0.9514686
6	-0.1630815	0.61574259	0.36055434	-0.79501530	-0.5171112	0.7196631

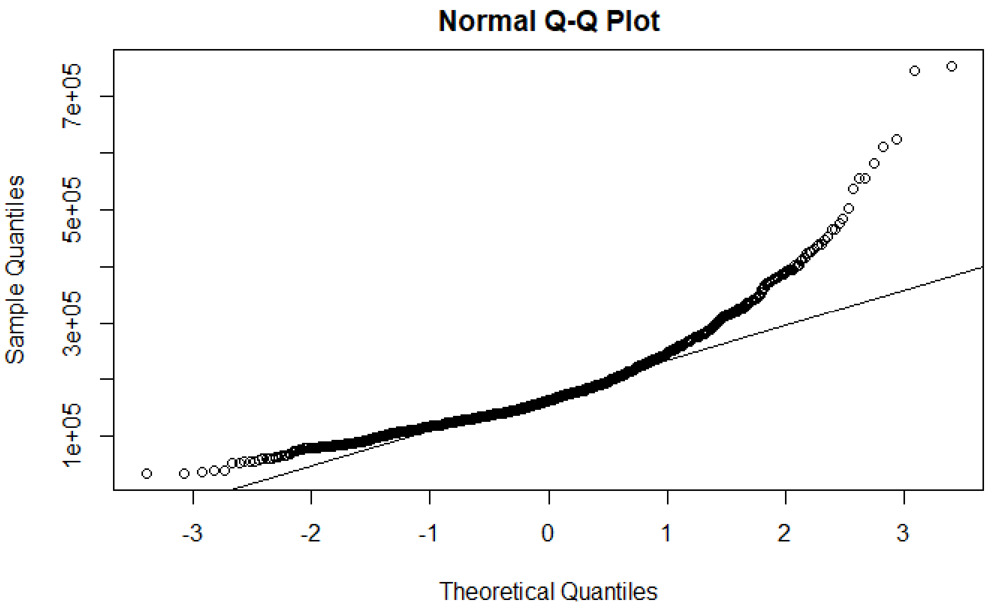
1-6 of 6 rows | 1-11 of 294 columns

Checking the distribution for skewness:

```
qqnorm(train$SalePrice)
```



qqline(train$SalePrice)

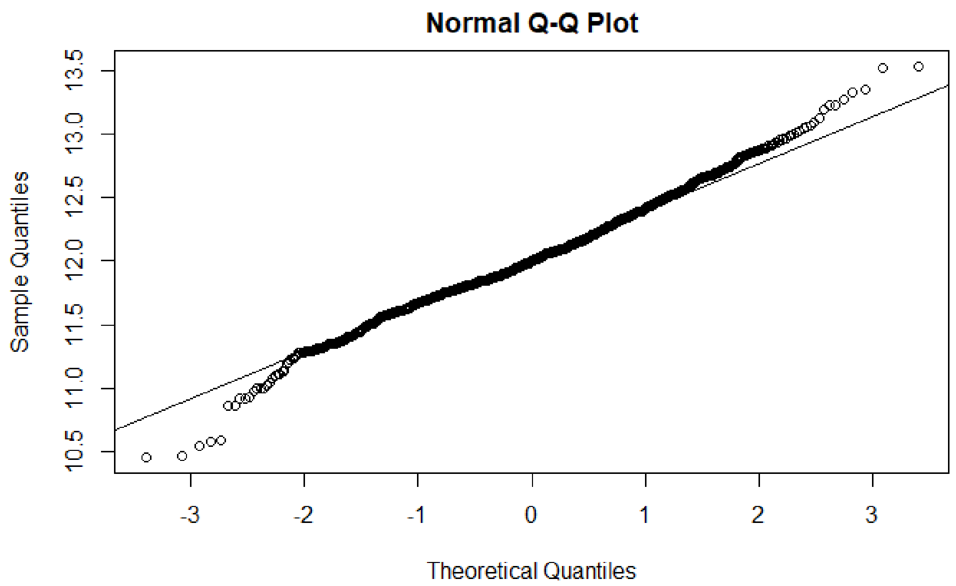


We see the numeric variables are slightly right skewed from the qq plot so perform a log transformation to sustain normality as required by linear regression

log_train<-log(train$SalePrice+1)

qqnorm(log_train)

qqline(log_train)



4. MODEL BUILDING AND IMPLEMENTATION OF ALGORITHM:

We train a decision tree model using XGB. We decided XGBoost would be best for our model as it consists of a large number of predictors so dividing them into weak learners and training each weak learner to perform better would be our best option. This also gave us a good practical experience in using boosted trees.

xgb_train <- df.total[1:1460,]

xgb_test <- df.total[1461:nrow(df.total),]

densetrain <- xgb.DMatrix(as.matrix(xgb_train), label=log_train) #dense matrix since most of the values are non-zeroes

densetest <- xgb.DMatrix(as.matrix(xgb_test))

The extreme gradient boosting is an ensemble technique which reduces the training data into subsamples, trains these weak samples and integrates to create one single decision tree.

set.seed(84)

xgb.paramaters <- list(booster = 'gbtree',

objective = 'reg:linear',

colsample_bytree=1,

eta=0.005,

max_depth=4,

min_child_weight=3,

alpha=0.3,   #learning rate

lambda=0.4,

gamma=0.01, # less overfit

subsample=0.6,

seed=5,

silent=TRUE)

#K - fold cross validation where the training set is divided into subsamples of k-1 training sets and 1 validation set

xgb.cv(xgb.paramaters, densetrain, nrounds = 10000, nfold = 4,

early_stopping_rounds = 500)

the iteration stops early if there is no significant improvement in RMSE for further iterations

```
[6329]  train-rmse:0.048336+0.000426   test-rmse:0.120862+0.005482
[6330]  train-rmse:0.048335+0.000427   test-rmse:0.120862+0.005482
[6331]  train-rmse:0.048334+0.000426   test-rmse:0.120862+0.005481
[6332]  train-rmse:0.048334+0.000426   test-rmse:0.120863+0.005482
[6333]  train-rmse:0.048333+0.000427   test-rmse:0.120863+0.005482
[6334]  train-rmse:0.048332+0.000427   test-rmse:0.120863+0.005482
[6335]  train-rmse:0.048332+0.000427   test-rmse:0.120863+0.005482
[6336]  train-rmse:0.048330+0.000428   test-rmse:0.120863+0.005481
[6337]  train-rmse:0.048329+0.000429   test-rmse:0.120863+0.005481
```

```
[6338]	train-rmse:0.048327+0.000427	test-rmse:0.120864+0.005480
[6339]	train-rmse:0.048326+0.000427	test-rmse:0.120864+0.005480
[6340]	train-rmse:0.048323+0.000428	test-rmse:0.120864+0.005479
[6341]	train-rmse:0.048322+0.000427	test-rmse:0.120864+0.005480
[6342]	train-rmse:0.048319+0.000427	test-rmse:0.120864+0.005481
[6343]	train-rmse:0.048318+0.000426	test-rmse:0.120863+0.005481
[6344]	train-rmse:0.048317+0.000427	test-rmse:0.120865+0.005482
[6345]	train-rmse:0.048314+0.000426	test-rmse:0.120865+0.005484
[6346]	train-rmse:0.048313+0.000426	test-rmse:0.120864+0.005484
[6347]	train-rmse:0.048312+0.000425	test-rmse:0.120862+0.005487
[6348]	train-rmse:0.048312+0.000426	test-rmse:0.120863+0.005487
[6349]	train-rmse:0.048311+0.000427	test-rmse:0.120863+0.005487
[6350]	train-rmse:0.048309+0.000427	test-rmse:0.120862+0.005488
[6351]	train-rmse:0.048308+0.000427	test-rmse:0.120862+0.005488
[6352]	train-rmse:0.048307+0.000427	test-rmse:0.120863+0.005487
[6353]	train-rmse:0.048305+0.000427	test-rmse:0.120862+0.005488
[6354]	train-rmse:0.048305+0.000427	test-rmse:0.120863+0.005489
[6355]	train-rmse:0.048303+0.000426	test-rmse:0.120863+0.005488
[6356]	train-rmse:0.048301+0.000425	test-rmse:0.120864+0.005487
[6357]	train-rmse:0.048299+0.000424	test-rmse:0.120866+0.005487
[6358]	train-rmse:0.048296+0.000421	test-rmse:0.120865+0.005487
[6359]	train-rmse:0.048294+0.000421	test-rmse:0.120866+0.005489
[6360]	train-rmse:0.048290+0.000418	test-rmse:0.120868+0.005488
[6361]	train-rmse:0.048290+0.000419	test-rmse:0.120869+0.005489
[6362]	train-rmse:0.048289+0.000419	test-rmse:0.120868+0.005488
[6363]	train-rmse:0.048288+0.000419	test-rmse:0.120868+0.005488
[6364]	train-rmse:0.048286+0.000420	test-rmse:0.120867+0.005486
[6365]	train-rmse:0.048284+0.000418	test-rmse:0.120869+0.005485
[6366]	train-rmse:0.048282+0.000418	test-rmse:0.120869+0.005484
[6367]	train-rmse:0.048281+0.000416	test-rmse:0.120868+0.005485
[6368]	train-rmse:0.048279+0.000416	test-rmse:0.120868+0.005485
[6369]	train-rmse:0.048277+0.000416	test-rmse:0.120867+0.005482
[6370]	train-rmse:0.048275+0.000416	test-rmse:0.120867+0.005482
[6371]	train-rmse:0.048273+0.000416	test-rmse:0.120867+0.005483
[6372]	train-rmse:0.048272+0.000416	test-rmse:0.120867+0.005483
[6373]	train-rmse:0.048271+0.000417	test-rmse:0.120868+0.005483
[6374]	train-rmse:0.048269+0.000418	test-rmse:0.120868+0.005484
[6375]	train-rmse:0.048267+0.000418	test-rmse:0.120867+0.005483
[6376]	train-rmse:0.048266+0.000418	test-rmse:0.120867+0.005483
[6377]	train-rmse:0.048265+0.000419	test-rmse:0.120866+0.005483
[6378]	train-rmse:0.048261+0.000418	test-rmse:0.120865+0.005480
[6379]	train-rmse:0.048259+0.000419	test-rmse:0.120864+0.005479
[6380]	train-rmse:0.048259+0.000419	test-rmse:0.120864+0.005479
[6381]	train-rmse:0.048257+0.000421	test-rmse:0.120864+0.005478
[6382]	train-rmse:0.048255+0.000419	test-rmse:0.120864+0.005479
[6383]	train-rmse:0.048254+0.000421	test-rmse:0.120866+0.005480
[6384]	train-rmse:0.048253+0.000419	test-rmse:0.120867+0.005479
[6385]	train-rmse:0.048251+0.000419	test-rmse:0.120868+0.005478
[6386]	train-rmse:0.048248+0.000418	test-rmse:0.120869+0.005475
[6387]	train-rmse:0.048247+0.000417	test-rmse:0.120870+0.005473
[6388]	train-rmse:0.048247+0.000417	test-rmse:0.120871+0.005473
[6389]	train-rmse:0.048247+0.000417	test-rmse:0.120871+0.005473
[6390]	train-rmse:0.048244+0.000416	test-rmse:0.120872+0.005471
[6391]	train-rmse:0.048242+0.000417	test-rmse:0.120872+0.005472
[6392]	train-rmse:0.048242+0.000416	test-rmse:0.120871+0.005474
[6393]	train-rmse:0.048240+0.000415	test-rmse:0.120871+0.005474
[6394]	train-rmse:0.048239+0.000414	test-rmse:0.120873+0.005473
[6395]	train-rmse:0.048236+0.000413	test-rmse:0.120873+0.005471
[6396]	train-rmse:0.048234+0.000413	test-rmse:0.120874+0.005471
[6397]	train-rmse:0.048233+0.000413	test-rmse:0.120872+0.005471
[6398]	train-rmse:0.048231+0.000414	test-rmse:0.120873+0.005470
[6399]	train-rmse:0.048231+0.000414	test-rmse:0.120873+0.005471
```

```
[6400]  train-rmse:0.048230+0.000413   test-rmse:0.120872+0.005473
[6401]  train-rmse:0.048227+0.000413   test-rmse:0.120870+0.005475
[6402]  train-rmse:0.048226+0.000412   test-rmse:0.120871+0.005474
[6403]  train-rmse:0.048226+0.000412   test-rmse:0.120871+0.005474
[6404]  train-rmse:0.048225+0.000413   test-rmse:0.120871+0.005475
[6405]  train-rmse:0.048222+0.000413   test-rmse:0.120871+0.005476
[6406]  train-rmse:0.048221+0.000413   test-rmse:0.120873+0.005477
[6407]  train-rmse:0.048218+0.000411   test-rmse:0.120871+0.005480
[6408]  train-rmse:0.048217+0.000411   test-rmse:0.120872+0.005481
[6409]  train-rmse:0.048214+0.000410   test-rmse:0.120869+0.005482
[6410]  train-rmse:0.048213+0.000411   test-rmse:0.120870+0.005483
[6411]  train-rmse:0.048213+0.000411   test-rmse:0.120870+0.005483
[6412]  train-rmse:0.048213+0.000411   test-rmse:0.120870+0.005483
[6413]  train-rmse:0.048211+0.000411   test-rmse:0.120870+0.005482
[6414]  train-rmse:0.048210+0.000412   test-rmse:0.120870+0.005483
[6415]  train-rmse:0.048210+0.000411   test-rmse:0.120870+0.005483
[6416]  train-rmse:0.048206+0.000413   test-rmse:0.120871+0.005485
[6417]  train-rmse:0.048202+0.000414   test-rmse:0.120871+0.005486
[6418]  train-rmse:0.048201+0.000414   test-rmse:0.120870+0.005487
[6419]  train-rmse:0.048199+0.000414   test-rmse:0.120871+0.005486
[6420]  train-rmse:0.048199+0.000413   test-rmse:0.120871+0.005485
[6421]  train-rmse:0.048198+0.000412   test-rmse:0.120869+0.005486
[6422]  train-rmse:0.048197+0.000412   test-rmse:0.120868+0.005485
[6423]  train-rmse:0.048195+0.000414   test-rmse:0.120870+0.005486
[6424]  train-rmse:0.048194+0.000413   test-rmse:0.120869+0.005487
[6425]  train-rmse:0.048193+0.000413   test-rmse:0.120870+0.005487
[6426]  train-rmse:0.048192+0.000414   test-rmse:0.120869+0.005486
[6427]  train-rmse:0.048190+0.000414   test-rmse:0.120869+0.005487
[6428]  train-rmse:0.048190+0.000414   test-rmse:0.120869+0.005487
[6429]  train-rmse:0.048188+0.000415   test-rmse:0.120868+0.005488
[6430]  train-rmse:0.048185+0.000416   test-rmse:0.120867+0.005486
[6431]  train-rmse:0.048183+0.000415   test-rmse:0.120867+0.005488
[6432]  train-rmse:0.048182+0.000415   test-rmse:0.120866+0.005488
Stopping. Best iteration:
[5932]  train-rmse:0.049089+0.000486   test-rmse:0.120797+0.005542
```

Trained model using boosting:

bst<-xgb.train(densetrain,params = xgb.paramaters, nrounds = 10000,

early_stopping_rounds = 300,

watchlist = list(train=densetrain))

```
[9860]  train-rmse:0.045301
[9861]  train-rmse:0.045299
[9862]  train-rmse:0.045299
[9863]  train-rmse:0.045299
[9864]  train-rmse:0.045297
[9865]  train-rmse:0.045297
[9866]  train-rmse:0.045296
[9867]  train-rmse:0.045296
[9868]  train-rmse:0.045295
[9869]  train-rmse:0.045294
[9870]  train-rmse:0.045293
[9871]  train-rmse:0.045293
[9872]  train-rmse:0.045293
[9873]  train-rmse:0.045290
[9874]  train-rmse:0.045288
[9875]  train-rmse:0.045288
```

```
[9876]  train-rmse:0.045288
[9877]  train-rmse:0.045288
[9878]  train-rmse:0.045285
[9879]  train-rmse:0.045285
[9880]  train-rmse:0.045284
[9881]  train-rmse:0.045284
[9882]  train-rmse:0.045280
[9883]  train-rmse:0.045278
[9884]  train-rmse:0.045278
[9885]  train-rmse:0.045278
[9886]  train-rmse:0.045278
[9887]  train-rmse:0.045278
[9888]  train-rmse:0.045275
[9889]  train-rmse:0.045275
[9890]  train-rmse:0.045275
[9891]  train-rmse:0.045275
[9892]  train-rmse:0.045273
[9893]  train-rmse:0.045273
[9894]  train-rmse:0.045273
[9895]  train-rmse:0.045271
[9896]  train-rmse:0.045269
[9897]  train-rmse:0.045269
[9898]  train-rmse:0.045269
[9899]  train-rmse:0.045264
[9900]  train-rmse:0.045264
[9901]  train-rmse:0.045264
[9902]  train-rmse:0.045264
[9903]  train-rmse:0.045264
[9904]  train-rmse:0.045262
[9905]  train-rmse:0.045260
[9906]  train-rmse:0.045259
[9907]  train-rmse:0.045259
[9908]  train-rmse:0.045259
[9909]  train-rmse:0.045259
[9910]  train-rmse:0.045259
[9911]  train-rmse:0.045257
[9912]  train-rmse:0.045257
[9913]  train-rmse:0.045257
[9914]  train-rmse:0.045257
[9915]  train-rmse:0.045257
[9916]  train-rmse:0.045257
[9917]  train-rmse:0.045255
[9918]  train-rmse:0.045253
[9919]  train-rmse:0.045253
[9920]  train-rmse:0.045253
[9921]  train-rmse:0.045253
[9922]  train-rmse:0.045253
[9923]  train-rmse:0.045253
[9924]  train-rmse:0.045253
[9925]  train-rmse:0.045253
[9926]  train-rmse:0.045252
[9927]  train-rmse:0.045251
[9928]  train-rmse:0.045251
[9929]  train-rmse:0.045251
[9930]  train-rmse:0.045250
[9931]  train-rmse:0.045250
[9932]  train-rmse:0.045246
[9933]  train-rmse:0.045246
[9934]  train-rmse:0.045244
[9935]  train-rmse:0.045243
[9936]  train-rmse:0.045243
[9937]  train-rmse:0.045241
```

```
[9938]  train-rmse:0.045240
[9939]  train-rmse:0.045238
[9940]  train-rmse:0.045235
[9941]  train-rmse:0.045233
[9942]  train-rmse:0.045233
[9943]  train-rmse:0.045232
[9944]  train-rmse:0.045232
[9945]  train-rmse:0.045232
[9946]  train-rmse:0.045230
[9947]  train-rmse:0.045230
[9948]  train-rmse:0.045230
[9949]  train-rmse:0.045230
[9950]  train-rmse:0.045230
[9951]  train-rmse:0.045230
[9952]  train-rmse:0.045230
[9953]  train-rmse:0.045226
[9954]  train-rmse:0.045226
[9955]  train-rmse:0.045224
[9956]  train-rmse:0.045224
[9957]  train-rmse:0.045223
[9958]  train-rmse:0.045221
[9959]  train-rmse:0.045220
[9960]  train-rmse:0.045220
[9961]  train-rmse:0.045220
[9962]  train-rmse:0.045220
[9963]  train-rmse:0.045218
[9964]  train-rmse:0.045212
[9965]  train-rmse:0.045212
[9966]  train-rmse:0.045212
[9967]  train-rmse:0.045212
[9968]  train-rmse:0.045212
[9969]  train-rmse:0.045212
[9970]  train-rmse:0.045212
[9971]  train-rmse:0.045212
[9972]  train-rmse:0.045212
[9973]  train-rmse:0.045211
[9974]  train-rmse:0.045210
[9975]  train-rmse:0.045210
[9976]  train-rmse:0.045209
[9977]  train-rmse:0.045209
[9978]  train-rmse:0.045209
[9979]  train-rmse:0.045209
[9980]  train-rmse:0.045209
[9981]  train-rmse:0.045208
[9982]  train-rmse:0.045207
[9983]  train-rmse:0.045205
[9984]  train-rmse:0.045204
[9985]  train-rmse:0.045204
[9986]  train-rmse:0.045204
[9987]  train-rmse:0.045204
[9988]  train-rmse:0.045204
[9989]  train-rmse:0.045204
[9990]  train-rmse:0.045204
[9991]  train-rmse:0.045204
[9992]  train-rmse:0.045204
[9993]  train-rmse:0.045204
[9994]  train-rmse:0.045203
[9995]  train-rmse:0.045203
[9996]  train-rmse:0.045203
[9997]  train-rmse:0.045203
[9998]  train-rmse:0.045203
[9999]  train-rmse:0.045203
```

```
[10000] train-rmse:0.045201
```

Future Scope

The prediction model shows promising scope in its ability to predict housing prices. Its use of a wide range of predictor variables would allow it to scale to various other uses. The model can be repurposed and trained with new data sets of housing data in different geographical locations. Hyper parameter tuning can be automated with multiple packages in the future .