House Price Prediction Project

Sami Halawa

08/01/2021

Contents

Section 1 - Introduction

As part of the Professional Certificate in Data Science course by Harvard, verified students are required to apply the knowledge learnt throughout the course to a dataset of their choice.

After exploring the Kaggle website and UCI Machine Learning Repository, I decided to select the following dataset from Kaggle:

- House Prices Advanced Regression Techniques see here

The dataset contains 79 variables that describe nearly all the features of residential properties in a city in Iowa.

We will attempt to explore which features allow us to predict the final sale price of each home.

Aim

We must build a model that predicts the sale price (SalePrice) for each house (Id).

Performance is evaluated on RMSE between the logarithm of the predicted value and the logarithm of the actual sale price.

From the leadership board at the time of writing this report,

- Median RMSLE = 0.14222
- Top 90th percentile = 0.120214
- Top 95th percentile = 0.1179585

I will attempt to produce a model that generates an RMSLE <= median RMSLE generated from the competition leadership board.

AIM: Produce a model RMSLE <= 0.14222

$$RMSLE = \sqrt{\frac{1}{N}\sum_{u,i}(log(\hat{y}_{u,i}) - log(y_{u,i}))^2}$$

```
# Creating a function to calculate the RMSLE

RMSLE <- function(a){
  RMSE(log(test_set$SalePrice),log(a))
}
```

Executive Summary

In this project, I start off with building a predictive model based on linear regression techniques.

I then apply more advanced machine learning algorithms to explore whether we can further improve the accuracy of our predictions.

The model that produces the most accurate result is the *Ensemble* model, having produced an **RMSLE of 0.10447** which represents a *70.1%* improvement in RMSLE relative to our first model *(Mean Model)*.

Section 2 - Method / Analysis

Step 1 - Installing required packages

The following code installs all of the necessary packages used in this project.

```r
# Installing packages, if required. Note: this process may take a few minutes.

if(!require(tidyverse)) install.packages("tidyverse", repos = "http://cran.us.r-project.org",
                                         dependencies = TRUE)

if(!require(caret)) install.packages("caret", repos = "http://cran.us.r-project.org",
                                     dependencies = TRUE)

if(!require(data.table)) install.packages("data.table", repos = "http://cran.us.r-project.org",
                                          dependencies = TRUE)

if(!require(lubridate)) install.packages("lubridate", repos = "http://cran.us.r-project.org",
                                         dependencies = TRUE)

if(!require(dplyr)) install.packages("dplyr", repos = "http://cran.us.r-project.org",
                                     dependencies = TRUE)

if(!require(caret)) install.packages("caret", repos = "http://cran.us.r-project.org",
                                     dependencies = TRUE)

if(!require(readxl)) install.packages("readxl", repos = "http://cran.us.r-project.org",
                                      dependencies = TRUE)

if(!require(downloader)) install.packages("downloader", repos = "http://cran.us.r-project.org",
                                          dependencies = TRUE)

if(!require(RCurl)) install.packages("RCurl", repos = "http://cran.us.r-project.org",
                                     dependencies = TRUE)

if(!require(kableExtra)) install.packages("knitExtra", repos = "http://cran.us.r-project.org",
                                          dependencies = TRUE)

if(!require(knitr)) install.packages("knitr", repos = "http://cran.us.r-project.org",
                                     dependencies = TRUE)

if(!require(reshape2)) install.packages("reshape2", repos = "http://cran.us.r-project.org",
                                        dependencies = TRUE)

if(!require(rpart.plot)) install.packages("rpart.plot", repos = "http://cran.us.r-project.org",
                                          dependencies = TRUE)

if(!require(arm)) install.packages("arm", repos = "http://cran.us.r-project.org",
                                   dependencies = TRUE)

if(!require(xgboost)) install.packages("xgboost", repos = "http://cran.us.r-project.org",
                                       dependencies = TRUE)
```

```r
if(!require(gbm)) install.packages("gbm", repos = "http://cran.us.r-project.org",
                                   dependencies = TRUE)

if(!require(randomForest)) install.packages("randomForest", repos = "http://cran.us.r-project.org",
                                            dependencies = TRUE)


# Loading packages

library(tidyverse)
library(caret)
library(data.table)
library(lubridate)
library(dplyr)
library(readr)
library(downloader)
library(RCurl)
library(readxl)
library(kableExtra)
library(knitr)
library(randomForest)
library(reshape2)
library(rpart.plot)
library(arm)
library(xgboost)
library(gbm)
library(randomForest)
```

Step 2 - Preparing the Data

Importing Data

```r
# Reading in the data from Github

#### TEST data

test_url <- "https://raw.githubusercontent.com/Sami-Halawa/edx-Capstone-CYO-Project/main/test.csv"

test <- tempfile()
download.file(test_url, test)

test <- read_csv(test)
```

```
##
## -- Column specification --------------------------------------------------------------
## cols(
##   .default = col_character(),
##   Id = col_double(),
##   MSSubClass = col_double(),
##   LotFrontage = col_double(),
##   LotArea = col_double(),
```

```
##   OverallQual = col_double(),
##   OverallCond = col_double(),
##   YearBuilt = col_double(),
##   YearRemodAdd = col_double(),
##   MasVnrArea = col_double(),
##   BsmtFinSF1 = col_double(),
##   BsmtFinSF2 = col_double(),
##   BsmtUnfSF = col_double(),
##   TotalBsmtSF = col_double(),
##   `1stFlrSF` = col_double(),
##   `2ndFlrSF` = col_double(),
##   LowQualFinSF = col_double(),
##   GrLivArea = col_double(),
##   BsmtFullBath = col_double(),
##   BsmtHalfBath = col_double(),
##   FullBath = col_double()
##   # ... with 17 more columns
## )
## i Use `spec()` for the full column specifications.
```

```r
# Converting to a dataframe

test <- as.data.frame(test)

### TRAIN data

train_url <- "https://raw.githubusercontent.com/Sami-Halawa/edx-Capstone-CYO-Project/main/train.csv"

train <- tempfile()
download.file(train_url, train)

train <- read_csv(train)
```

```
##
## -- Column specification --------------------------------------------------------
## cols(
##   .default = col_character(),
##   Id = col_double(),
##   MSSubClass = col_double(),
##   LotFrontage = col_double(),
##   LotArea = col_double(),
##   OverallQual = col_double(),
##   OverallCond = col_double(),
##   YearBuilt = col_double(),
##   YearRemodAdd = col_double(),
##   MasVnrArea = col_double(),
##   BsmtFinSF1 = col_double(),
##   BsmtFinSF2 = col_double(),
##   BsmtUnfSF = col_double(),
##   TotalBsmtSF = col_double(),
##   `1stFlrSF` = col_double(),
##   `2ndFlrSF` = col_double(),
##   LowQualFinSF = col_double(),
##   GrLivArea = col_double(),
```

```
##   BsmtFullBath = col_double(),
##   BsmtHalfBath = col_double(),
##   FullBath = col_double()
##   # ... with 18 more columns
## )
## i Use `spec()` for the full column specifications.
```

```r
# Converting to a dataframe

train <- as.data.frame(train)
```

Step 3 - Preprocessing

Prior to conducting any data analysis, it is important to familiarise ourselves with the dataset. We will use the head and str functions to obtain a snapshot of the edx dataset.

We will also use the summary function to obtain a summary for each variable and check whether there are any missing values (NA's).

Quick Snapshot of the data

```r
# Snapshot of the train set

str(train)
```

```
## 'data.frame':    1460 obs. of  81 variables:
##  $ Id           : num  1 2 3 4 5 6 7 8 9 10 ...
##  $ MSSubClass   : num  60 20 60 70 60 50 20 60 50 190 ...
##  $ MSZoning     : chr  "RL" "RL" "RL" "RL" ...
##  $ LotFrontage  : num  65 80 68 60 84 85 75 NA 51 50 ...
##  $ LotArea      : num  8450 9600 11250 9550 14260 ...
##  $ 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" ...
##  $ OverallQual  : num  7 6 7 7 8 5 8 7 7 5 ...
##  $ OverallCond  : num  5 8 5 5 5 5 5 6 5 6 ...
##  $ YearBuilt    : num  2003 1976 2001 1915 2000 ...
##  $ YearRemodAdd : num  2003 1976 2002 1970 2000 ...
##  $ RoofStyle    : chr  "Gable" "Gable" "Gable" "Gable" ...
##  $ RoofMatl     : chr  "CompShg" "CompShg" "CompShg" "CompShg" ...
##  $ Exterior1st  : chr  "VinylSd" "MetalSd" "VinylSd" "Wd Sdng" ...
##  $ Exterior2nd  : chr  "VinylSd" "MetalSd" "VinylSd" "Wd Shng" ...
```

```
##  $ MasVnrType   : chr  "BrkFace" "None" "BrkFace" "None" ...
##  $ MasVnrArea   : num  196 0 162 0 350 0 186 240 0 0 ...
##  $ ExterQual    : chr  "Gd" "TA" "Gd" "TA" ...
##  $ ExterCond    : chr  "TA" "TA" "TA" "TA" ...
##  $ Foundation   : chr  "PConc" "CBlock" "PConc" "BrkTil" ...
##  $ BsmtQual     : chr  "Gd" "Gd" "Gd" "TA" ...
##  $ BsmtCond     : chr  "TA" "TA" "TA" "Gd" ...
##  $ BsmtExposure : chr  "No" "Gd" "Mn" "No" ...
##  $ BsmtFinType1 : chr  "GLQ" "ALQ" "GLQ" "ALQ" ...
##  $ BsmtFinSF1   : num  706 978 486 216 655 ...
##  $ BsmtFinType2 : chr  "Unf" "Unf" "Unf" "Unf" ...
##  $ BsmtFinSF2   : num  0 0 0 0 0 0 0 32 0 0 ...
##  $ BsmtUnfSF    : num  150 284 434 540 490 64 317 216 952 140 ...
##  $ TotalBsmtSF  : num  856 1262 920 756 1145 ...
##  $ Heating      : chr  "GasA" "GasA" "GasA" "GasA" ...
##  $ HeatingQC    : chr  "Ex" "Ex" "Ex" "Gd" ...
##  $ CentralAir   : chr  "Y" "Y" "Y" "Y" ...
##  $ Electrical   : chr  "SBrkr" "SBrkr" "SBrkr" "SBrkr" ...
##  $ 1stFlrSF     : num  856 1262 920 961 1145 ...
##  $ 2ndFlrSF     : num  854 0 866 756 1053 ...
##  $ LowQualFinSF : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ GrLivArea    : num  1710 1262 1786 1717 2198 ...
##  $ BsmtFullBath : num  1 0 1 1 1 1 1 1 0 1 ...
##  $ BsmtHalfBath : num  0 1 0 0 0 0 0 0 0 0 ...
##  $ FullBath     : num  2 2 2 1 2 1 2 2 2 1 ...
##  $ HalfBath     : num  1 0 1 0 1 1 0 1 0 0 ...
##  $ BedroomAbvGr : num  3 3 3 3 4 1 3 3 2 2 ...
##  $ KitchenAbvGr : num  1 1 1 1 1 1 1 1 2 2 ...
##  $ KitchenQual  : chr  "Gd" "TA" "Gd" "Gd" ...
##  $ TotRmsAbvGrd : num  8 6 6 7 9 5 7 7 8 5 ...
##  $ Functional   : chr  "Typ" "Typ" "Typ" "Typ" ...
##  $ Fireplaces   : num  0 1 1 1 1 0 1 2 2 2 ...
##  $ FireplaceQu  : chr  NA "TA" "TA" "Gd" ...
##  $ GarageType   : chr  "Attchd" "Attchd" "Attchd" "Detchd" ...
##  $ GarageYrBlt  : num  2003 1976 2001 1998 2000 ...
##  $ GarageFinish : chr  "RFn" "RFn" "RFn" "Unf" ...
##  $ GarageCars   : num  2 2 2 3 3 2 2 2 2 1 ...
##  $ GarageArea   : num  548 460 608 642 836 480 636 484 468 205 ...
##  $ GarageQual   : chr  "TA" "TA" "TA" "TA" ...
##  $ GarageCond   : chr  "TA" "TA" "TA" "TA" ...
##  $ PavedDrive   : chr  "Y" "Y" "Y" "Y" ...
##  $ WoodDeckSF   : num  0 298 0 0 192 40 255 235 90 0 ...
##  $ OpenPorchSF  : num  61 0 42 35 84 30 57 204 0 4 ...
##  $ EnclosedPorch: num  0 0 0 272 0 0 0 228 205 0 ...
##  $ 3SsnPorch    : num  0 0 0 0 0 320 0 0 0 0 ...
##  $ ScreenPorch  : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ PoolArea     : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ PoolQC       : chr  NA NA NA NA ...
##  $ Fence        : chr  NA NA NA NA ...
##  $ MiscFeature  : chr  NA NA NA NA ...
##  $ MiscVal      : num  0 0 0 0 0 700 0 350 0 0 ...
##  $ MoSold       : num  2 5 9 2 12 10 8 11 4 1 ...
##  $ YrSold       : num  2008 2007 2008 2006 2008 ...
##  $ SaleType     : chr  "WD" "WD" "WD" "WD" ...
```

```
##  $ SaleCondition: chr  "Normal" "Normal" "Normal" "Abnorml" ...
##  $ SalePrice    : num  208500 181500 223500 140000 250000 ...
##  - attr(*, "spec")=
##   .. cols(
##   ..   Id = col_double(),
##   ..   MSSubClass = col_double(),
##   ..   MSZoning = col_character(),
##   ..   LotFrontage = col_double(),
##   ..   LotArea = col_double(),
##   ..   Street = col_character(),
##   ..   Alley = col_character(),
##   ..   LotShape = col_character(),
##   ..   LandContour = col_character(),
##   ..   Utilities = col_character(),
##   ..   LotConfig = col_character(),
##   ..   LandSlope = col_character(),
##   ..   Neighborhood = col_character(),
##   ..   Condition1 = col_character(),
##   ..   Condition2 = col_character(),
##   ..   BldgType = col_character(),
##   ..   HouseStyle = col_character(),
##   ..   OverallQual = col_double(),
##   ..   OverallCond = col_double(),
##   ..   YearBuilt = col_double(),
##   ..   YearRemodAdd = col_double(),
##   ..   RoofStyle = col_character(),
##   ..   RoofMatl = col_character(),
##   ..   Exterior1st = col_character(),
##   ..   Exterior2nd = col_character(),
##   ..   MasVnrType = col_character(),
##   ..   MasVnrArea = col_double(),
##   ..   ExterQual = col_character(),
##   ..   ExterCond = col_character(),
##   ..   Foundation = col_character(),
##   ..   BsmtQual = col_character(),
##   ..   BsmtCond = col_character(),
##   ..   BsmtExposure = col_character(),
##   ..   BsmtFinType1 = col_character(),
##   ..   BsmtFinSF1 = col_double(),
##   ..   BsmtFinType2 = col_character(),
##   ..   BsmtFinSF2 = col_double(),
##   ..   BsmtUnfSF = col_double(),
##   ..   TotalBsmtSF = col_double(),
##   ..   Heating = col_character(),
##   ..   HeatingQC = col_character(),
##   ..   CentralAir = col_character(),
##   ..   Electrical = col_character(),
##   ..   `1stFlrSF` = col_double(),
##   ..   `2ndFlrSF` = col_double(),
##   ..   LowQualFinSF = col_double(),
##   ..   GrLivArea = col_double(),
##   ..   BsmtFullBath = col_double(),
##   ..   BsmtHalfBath = col_double(),
##   ..   FullBath = col_double(),
```

```
##   ..   HalfBath = col_double(),
##   ..   BedroomAbvGr = col_double(),
##   ..   KitchenAbvGr = col_double(),
##   ..   KitchenQual = col_character(),
##   ..   TotRmsAbvGrd = col_double(),
##   ..   Functional = col_character(),
##   ..   Fireplaces = col_double(),
##   ..   FireplaceQu = col_character(),
##   ..   GarageType = col_character(),
##   ..   GarageYrBlt = col_double(),
##   ..   GarageFinish = col_character(),
##   ..   GarageCars = col_double(),
##   ..   GarageArea = col_double(),
##   ..   GarageQual = col_character(),
##   ..   GarageCond = col_character(),
##   ..   PavedDrive = col_character(),
##   ..   WoodDeckSF = col_double(),
##   ..   OpenPorchSF = col_double(),
##   ..   EnclosedPorch = col_double(),
##   ..   `3SsnPorch` = col_double(),
##   ..   ScreenPorch = col_double(),
##   ..   PoolArea = col_double(),
##   ..   PoolQC = col_character(),
##   ..   Fence = col_character(),
##   ..   MiscFeature = col_character(),
##   ..   MiscVal = col_double(),
##   ..   MoSold = col_double(),
##   ..   YrSold = col_double(),
##   ..   SaleType = col_character(),
##   ..   SaleCondition = col_character(),
##   ..   SalePrice = col_double()
##   .. )
```

```r
summary(train)
```

```
##        Id           MSSubClass      MSZoning          LotFrontage
##  Min.   :   1.0   Min.   : 20.0   Length:1460        Min.   : 21.00
##  1st Qu.: 365.8   1st Qu.: 20.0   Class :character   1st Qu.: 59.00
##  Median : 730.5   Median : 50.0   Mode  :character   Median : 69.00
##  Mean   : 730.5   Mean   : 56.9                      Mean   : 70.05
##  3rd Qu.:1095.2   3rd Qu.: 70.0                      3rd Qu.: 80.00
##  Max.   :1460.0   Max.   :190.0                      Max.   :313.00
##                                                      NA's   :259
##     LotArea          Street             Alley             LotShape
##  Min.   :  1300   Length:1460        Length:1460        Length:1460
##  1st Qu.:  7554   Class :character   Class :character   Class :character
##  Median :  9478   Mode  :character   Mode  :character   Mode  :character
##  Mean   : 10517
##  3rd Qu.: 11602
##  Max.   :215245
##
##  LandContour         Utilities          LotConfig          LandSlope
##  Length:1460        Length:1460        Length:1460        Length:1460
##  Class :character   Class :character   Class :character   Class :character
```

```
##  Mode  :character   Mode  :character   Mode  :character   Mode  :character
##
##
##
##
##  Neighborhood        Condition1         Condition2          BldgType
##  Length:1460        Length:1460        Length:1460        Length:1460
##  Class :character   Class :character   Class :character   Class :character
##  Mode  :character   Mode  :character   Mode  :character   Mode  :character
##
##
##
##
##   HouseStyle         OverallQual      OverallCond      YearBuilt     YearRemodAdd
##  Length:1460        Min.   : 1.000   Min.   :1.000   Min.   :1872   Min.   :1950
##  Class :character   1st Qu.: 5.000   1st Qu.:5.000   1st Qu.:1954   1st Qu.:1967
##  Mode  :character   Median : 6.000   Median :5.000   Median :1973   Median :1994
##                     Mean   : 6.099   Mean   :5.575   Mean   :1971   Mean   :1985
##                     3rd Qu.: 7.000   3rd Qu.:6.000   3rd Qu.:2000   3rd Qu.:2004
##                     Max.   :10.000   Max.   :9.000   Max.   :2010   Max.   :2010
##
##   RoofStyle           RoofMatl         Exterior1st        Exterior2nd
##  Length:1460        Length:1460        Length:1460        Length:1460
##  Class :character   Class :character   Class :character   Class :character
##  Mode  :character   Mode  :character   Mode  :character   Mode  :character
##
##
##
##
##   MasVnrType          MasVnrArea      ExterQual          ExterCond
##  Length:1460        Min.   :   0.0   Length:1460        Length:1460
##  Class :character   1st Qu.:   0.0   Class :character   Class :character
##  Mode  :character   Median :   0.0   Mode  :character   Mode  :character
##                     Mean   : 103.7
##                     3rd Qu.: 166.0
##                     Max.   :1600.0
##                     NA's   :8
##   Foundation          BsmtQual           BsmtCond         BsmtExposure
##  Length:1460        Length:1460        Length:1460        Length:1460
##  Class :character   Class :character   Class :character   Class :character
##  Mode  :character   Mode  :character   Mode  :character   Mode  :character
##
##
##
##
##  BsmtFinType1         BsmtFinSF1     BsmtFinType2         BsmtFinSF2
##  Length:1460        Min.   :   0.0   Length:1460        Min.   :   0.00
##  Class :character   1st Qu.:   0.0   Class :character   1st Qu.:   0.00
##  Mode  :character   Median : 383.5   Mode  :character   Median :   0.00
##                     Mean   : 443.6                      Mean   :  46.55
##                     3rd Qu.: 712.2                      3rd Qu.:   0.00
##                     Max.   :5644.0                      Max.   :1474.00
##
##    BsmtUnfSF       TotalBsmtSF       Heating           HeatingQC
```

```
##  Min.   :   0.0   Min.   :   0.0   Length:1460        Length:1460
##  1st Qu.: 223.0   1st Qu.: 795.8   Class :character   Class :character
##  Median : 477.5   Median : 991.5   Mode  :character   Mode  :character
##  Mean   : 567.2   Mean   :1057.4
##  3rd Qu.: 808.0   3rd Qu.:1298.2
##  Max.   :2336.0   Max.   :6110.0
##
##   CentralAir         Electrical          1stFlrSF      2ndFlrSF
##  Length:1460        Length:1460        Min.   : 334   Min.   :   0
##  Class :character   Class :character   1st Qu.: 882   1st Qu.:   0
##  Mode  :character   Mode  :character   Median :1087   Median :   0
##                                        Mean   :1163   Mean   : 347
##                                        3rd Qu.:1391   3rd Qu.: 728
##                                        Max.   :4692   Max.   :2065
##
##   LowQualFinSF       GrLivArea     BsmtFullBath     BsmtHalfBath        FullBath
##  Min.   :  0.000   Min.   : 334   Min.   :0.0000   Min.   :0.00000   Min.   :0.000
##  1st Qu.:  0.000   1st Qu.:1130   1st Qu.:0.0000   1st Qu.:0.00000   1st Qu.:1.000
##  Median :  0.000   Median :1464   Median :0.0000   Median :0.00000   Median :2.000
##  Mean   :  5.845   Mean   :1515   Mean   :0.4253   Mean   :0.05753   Mean   :1.565
##  3rd Qu.:  0.000   3rd Qu.:1777   3rd Qu.:1.0000   3rd Qu.:0.00000   3rd Qu.:2.000
##  Max.   :572.000   Max.   :5642   Max.   :3.0000   Max.   :2.00000   Max.   :3.000
##
##     HalfBath       BedroomAbvGr    KitchenAbvGr   KitchenQual
##  Min.   :0.0000   Min.   :0.000   Min.   :0.000   Length:1460
##  1st Qu.:0.0000   1st Qu.:2.000   1st Qu.:1.000   Class :character
##  Median :0.0000   Median :3.000   Median :1.000   Mode  :character
##  Mean   :0.3829   Mean   :2.866   Mean   :1.047
##  3rd Qu.:1.0000   3rd Qu.:3.000   3rd Qu.:1.000
##  Max.   :2.0000   Max.   :8.000   Max.   :3.000
##
##   TotRmsAbvGrd     Functional          Fireplaces    FireplaceQu
##  Min.   : 2.000   Length:1460        Min.   :0.000   Length:1460
##  1st Qu.: 5.000   Class :character   1st Qu.:0.000   Class :character
##  Median : 6.000   Mode  :character   Median :1.000   Mode  :character
##  Mean   : 6.518                      Mean   :0.613
##  3rd Qu.: 7.000                      3rd Qu.:1.000
##  Max.   :14.000                      Max.   :3.000
##
##   GarageType         GarageYrBlt   GarageFinish        GarageCars
##  Length:1460        Min.   :1900   Length:1460        Min.   :0.000
##  Class :character   1st Qu.:1961   Class :character   1st Qu.:1.000
##  Mode  :character   Median :1980   Mode  :character   Median :2.000
##                     Mean   :1979                      Mean   :1.767
##                     3rd Qu.:2002                      3rd Qu.:2.000
##                     Max.   :2010                      Max.   :4.000
##                     NA's   :81
##    GarageArea      GarageQual         GarageCond         PavedDrive
##  Min.   :   0.0   Length:1460        Length:1460        Length:1460
##  1st Qu.: 334.5   Class :character   Class :character   Class :character
##  Median : 480.0   Mode  :character   Mode  :character   Mode  :character
##  Mean   : 473.0
##  3rd Qu.: 576.0
##  Max.   :1418.0
```

```
##
##    WoodDeckSF       OpenPorchSF      EnclosedPorch      3SsnPorch
##  Min.    :  0.00   Min.    :  0.00   Min.    :  0.00   Min.    :  0.00
##  1st Qu.:  0.00   1st Qu.:  0.00   1st Qu.:  0.00   1st Qu.:  0.00
##  Median :  0.00   Median : 25.00   Median :  0.00   Median :  0.00
##  Mean    : 94.24   Mean    : 46.66   Mean    : 21.95   Mean    :  3.41
##  3rd Qu.:168.00   3rd Qu.: 68.00   3rd Qu.:  0.00   3rd Qu.:  0.00
##  Max.    :857.00   Max.    :547.00   Max.    :552.00   Max.    :508.00
##
##   ScreenPorch        PoolArea          PoolQC             Fence
##  Min.    :  0.00   Min.    :  0.000   Length:1460        Length:1460
##  1st Qu.:  0.00   1st Qu.:  0.000   Class :character   Class :character
##  Median :  0.00   Median :  0.000   Mode  :character   Mode  :character
##  Mean    : 15.06   Mean    :  2.759
##  3rd Qu.:  0.00   3rd Qu.:  0.000
##  Max.    :480.00   Max.    :738.000
##
##  MiscFeature          MiscVal             MoSold          YrSold
##  Length:1460        Min.    :    0.00   Min.    : 1.000   Min.    :2006
##  Class :character   1st Qu.:    0.00   1st Qu.: 5.000   1st Qu.:2007
##  Mode  :character   Median :    0.00   Median : 6.000   Median :2008
##                     Mean    :   43.49   Mean    : 6.322   Mean    :2008
##                     3rd Qu.:    0.00   3rd Qu.: 8.000   3rd Qu.:2009
##                     Max.    :15500.00   Max.    :12.000   Max.    :2010
##
##    SaleType          SaleCondition        SalePrice
##  Length:1460        Length:1460        Min.    : 34900
##  Class :character   Class :character   1st Qu.:129975
##  Mode  :character   Mode  :character   Median :163000
##                                         Mean    :180921
##                                         3rd Qu.:214000
##                                         Max.    :755000
##
```

Missing Values

As can be seen from the output below, there are a number of attributes / fields that contain missing values. The top 3 attributes by total number of missing values are:

- PoolQC (1456)
- MiscFeature (1408)
- Alley (1352)

These will need to be amended / filled in prior to conducting analysis and building our models.

```r
# Checking for missing values on a column by column basis

test_na <- data.frame(NAs=colSums(is.na(test))) %>% filter(NAs>0) %>% arrange(desc(NAs))

test_na
```

```
##                 NAs
```

```
## PoolQC       1456
## MiscFeature  1408
## Alley        1352
## Fence        1169
## FireplaceQu   730
## LotFrontage   227
## GarageYrBlt    78
## GarageFinish   78
## GarageQual     78
## GarageCond     78
## GarageType     76
## BsmtCond       45
## BsmtQual       44
## BsmtExposure   44
## BsmtFinType1   42
## BsmtFinType2   42
## MasVnrType     16
## MasVnrArea     15
## MSZoning        4
## Utilities       2
## BsmtFullBath    2
## BsmtHalfBath    2
## Functional      2
## Exterior1st     1
## Exterior2nd     1
## BsmtFinSF1      1
## BsmtFinSF2      1
## BsmtUnfSF       1
## TotalBsmtSF     1
## KitchenQual     1
## GarageCars      1
## GarageArea      1
## SaleType        1
```

```r
train_na <- data.frame(NAs=colSums(is.na(train))) %>% filter(NAs>0) %>% arrange(desc(NAs))

train_na
```

```
##               NAs
## PoolQC       1453
## MiscFeature  1406
## Alley        1369
## Fence        1179
## FireplaceQu   690
## LotFrontage   259
## GarageType     81
## GarageYrBlt    81
## GarageFinish   81
## GarageQual     81
## GarageCond     81
## BsmtExposure   38
## BsmtFinType2   38
## BsmtQual       37
## BsmtCond       37
```

```
## BsmtFinType1   37
## MasVnrType      8
## MasVnrArea      8
## Electrical      1
```

Updating Values

My approach for filling in missing values is as follows:

1. Where attributes represent features that can be quantified, I will replace missing values with zeros (0).

- LotFrontage represents "Linear feet of street connected to property". I will replace NAs with 0.

2. Where they represent qualitative features, I will replace missing values with "None" if applicable or the most frequent value where "None" does not exist.

- PoolQC represents the quality of the swimming pool, if a property has one. I will replace the NA values with "None"

```r
# UPDATING MISSING VALUES in TRAIN SET

train <- train %>%
  mutate(PoolQC = ifelse(is.na(PoolQC), "None", PoolQC),
         MiscFeature = ifelse(is.na(MiscFeature), "None", MiscFeature),
         Alley = ifelse(is.na(Alley), "None", Alley),Fence = ifelse(is.na(Fence), "None", Fence),
         FireplaceQu = ifelse(is.na(FireplaceQu), "None", FireplaceQu),
         GarageFinish = ifelse(is.na(GarageFinish), "None", GarageFinish),
         GarageQual = ifelse(is.na(GarageQual), "None", GarageQual),
         GarageCond = ifelse(is.na(GarageCond), "None", GarageCond),
         GarageType = ifelse(is.na(GarageType), "None", GarageType),
         BsmtCond = ifelse(is.na(BsmtCond), "None", BsmtCond),
         BsmtQual = ifelse(is.na(BsmtQual), "None", BsmtQual),
         BsmtExposure = ifelse(is.na(BsmtExposure), "None", BsmtExposure),
         BsmtFinType1 = ifelse(is.na(BsmtFinType1), "None", BsmtFinType1),
         BsmtFinType2 = ifelse(is.na(BsmtFinType2), "None", BsmtFinType2),
         MasVnrType = ifelse(is.na(MasVnrType), "None", MasVnrType),
         MSZoning = ifelse(is.na(MSZoning), "RL", MSZoning),
         Utilities = ifelse(is.na(Utilities), "Allpub", Utilities),
         Functional = ifelse(is.na(Functional), "Typ", Functional),
         Exterior1st = ifelse(is.na(Exterior1st), "VinylSd", Exterior1st),
         Exterior2nd = ifelse(is.na(Exterior2nd), "VinylSd", Exterior2nd),
         KitchenQual = ifelse(is.na(KitchenQual), "None", KitchenQual),
         SaleType = ifelse(is.na(SaleType), "WD", SaleType),
         LotFrontage = ifelse(is.na(LotFrontage), 0, LotFrontage),
         GarageYrBlt = ifelse(is.na(GarageYrBlt), 0, GarageYrBlt),
         MasVnrArea = ifelse(is.na(MasVnrArea), 0, MasVnrArea),
         BsmtFullBath = ifelse(is.na(BsmtFullBath), 0, BsmtFullBath),
         BsmtHalfBath = ifelse(is.na(BsmtHalfBath), 0, BsmtHalfBath),
         BsmtFinSF1 = ifelse(is.na(BsmtFinSF1), 0, BsmtFinSF1),
         BsmtFinSF2 = ifelse(is.na(BsmtFinSF2), 0, BsmtFinSF2),
         BsmtUnfSF = ifelse(is.na(BsmtUnfSF), 0, BsmtUnfSF),
         TotalBsmtSF = ifelse(is.na(TotalBsmtSF), 0, TotalBsmtSF),
```

```r
         GarageCars = ifelse(is.na(GarageCars), 0, GarageCars),
         GarageArea = ifelse(is.na(GarageArea), 0, GarageArea),
         Electrical = ifelse(is.na(Electrical), "SBrkr", Electrical))

# New total number of NAs

sum(is.na(train))
```

```
## [1] 0
```

```r
# UPDATING MISSING VALUES in TEST SET

test <- test %>%
  mutate(PoolQC = ifelse(is.na(PoolQC), "None", PoolQC),
         MiscFeature = ifelse(is.na(MiscFeature), "None", MiscFeature),
         Alley = ifelse(is.na(Alley), "None", Alley),Fence = ifelse(is.na(Fence), "None", Fence),
         FireplaceQu = ifelse(is.na(FireplaceQu), "None", FireplaceQu),
         GarageFinish = ifelse(is.na(GarageFinish), "None", GarageFinish),
         GarageQual = ifelse(is.na(GarageQual), "None", GarageQual),
         GarageCond = ifelse(is.na(GarageCond), "None", GarageCond),
         GarageType = ifelse(is.na(GarageType), "None", GarageType),
         BsmtCond = ifelse(is.na(BsmtCond), "None", BsmtCond),
         BsmtQual = ifelse(is.na(BsmtQual), "None", BsmtQual),
         BsmtExposure = ifelse(is.na(BsmtExposure), "None", BsmtExposure),
         BsmtFinType1 = ifelse(is.na(BsmtFinType1), "None", BsmtFinType1),
         BsmtFinType2 = ifelse(is.na(BsmtFinType2), "None", BsmtFinType2),
         MasVnrType = ifelse(is.na(MasVnrType), "None", MasVnrType),
         MSZoning = ifelse(is.na(MSZoning), "RL", MSZoning),
         Utilities = ifelse(is.na(Utilities), "Allpub", Utilities),
         Functional = ifelse(is.na(Functional), "Typ", Functional),
         Exterior1st = ifelse(is.na(Exterior1st), "VinylSd", Exterior1st),
         Exterior2nd = ifelse(is.na(Exterior2nd), "VinylSd", Exterior2nd),
         KitchenQual = ifelse(is.na(KitchenQual), "None", KitchenQual),
         SaleType = ifelse(is.na(SaleType), "WD", SaleType),
         LotFrontage = ifelse(is.na(LotFrontage), 0, LotFrontage),
         GarageYrBlt = ifelse(is.na(GarageYrBlt), 0, GarageYrBlt),
         MasVnrArea = ifelse(is.na(MasVnrArea), 0, MasVnrArea),
         BsmtFullBath = ifelse(is.na(BsmtFullBath), 0, BsmtFullBath),
         BsmtHalfBath = ifelse(is.na(BsmtHalfBath), 0, BsmtHalfBath),
         BsmtFinSF1 = ifelse(is.na(BsmtFinSF1), 0, BsmtFinSF1),
         BsmtFinSF2 = ifelse(is.na(BsmtFinSF2), 0, BsmtFinSF2),
         BsmtUnfSF = ifelse(is.na(BsmtUnfSF), 0, BsmtUnfSF),
         TotalBsmtSF = ifelse(is.na(TotalBsmtSF), 0, TotalBsmtSF),
         GarageCars = ifelse(is.na(GarageCars), 0, GarageCars),
         GarageArea = ifelse(is.na(GarageArea), 0, GarageArea),
         Electrical = ifelse(is.na(Electrical), "SBrkr", Electrical))

# New total number of NAs

sum(is.na(test))
```

```
## [1] 0
```

We have now successfully removed NAs from our dataset.

Converting characters to Factors

In order to apply advanced regression techniques e.g. Rpart and Random Forest, we need to convert character columns to factors.

```
# Generating a list of character columns

## Test Set

col_names <- colnames(test[,sapply(test,class)=="character"])

# Converting these columns to factors

test[col_names] <- lapply(test[col_names] , factor)

## Train Set

# Generating a list of character columns

col_names_2 <- colnames(train[,sapply(train,class)=="character"])

# Converting these columns to factors

train[col_names_2] <- lapply(train[col_names_2] , factor)
```

Step 4 - Creating the Train, Test and Validation data sets

As we did with the MovieLens project, I will split the Train data into a test, train and validation set.

I have used 10% of the data to generate the validation set.

I will them split the train dataset into a new train and test datasets to be used for this project. Again, I will use a 90-10 split.

```
## Creating the validation set

set.seed(1, sample.kind="Rounding") # if using R 3.5 or earlier, use `set.seed(1)`

test_index <- createDataPartition(y=train$SalePrice, times=1, p=0.1, list=FALSE)

sales_data <-train[-test_index,]

temp <- train[test_index,]


# Making sure that the key variables in the validation set are also in the sales_data set

validation <- temp %>%
```

```r
  semi_join(sales_data, by = "OverallQual") %>%
  semi_join(sales_data, by = "GrLivArea") %>%
  semi_join(sales_data, by = "GarageCars")


# Adding rows removed from validation set back into the sales_data set

removed <- anti_join(temp, validation)

sales_data <- rbind(sales_data, removed)
```

```r
###### Creating the new Train and Test datasets

# Partitioning the data

set.seed(1, sample.kind="Rounding") # if using R 3.5 or earlier, use `set.seed(1)`

test_index <- createDataPartition(y=sales_data$SalePrice, times=1, p=0.1, list=FALSE)

# Creating the train and test sets

train_set <-sales_data[-test_index,]

test_set <- sales_data[test_index,]

# Ensuring column names are valid

colnames(test_set) <- make.names(colnames(test_set))
colnames(train_set) <- make.names(colnames(train_set))

# Removing OverallCond, GrLivArea, GarageCars from the test set that do not appear in the training set

test_set <- test_set %>%
  semi_join(train_set, by="OverallQual") %>%
  semi_join(train_set, by="GrLivArea") %>%
  semi_join(train_set, by="GarageCars")
```

Step 5 - Analysis of Data

We will analyse the data using the train dataset.

```r
# Number of distinct property Ids in the train set

a <- train_set %>%
  group_by(Id)

nrow(a)
```

```
## [1] 1230
```

There are 1230 properties included in our train set.

Distribution of Sales price

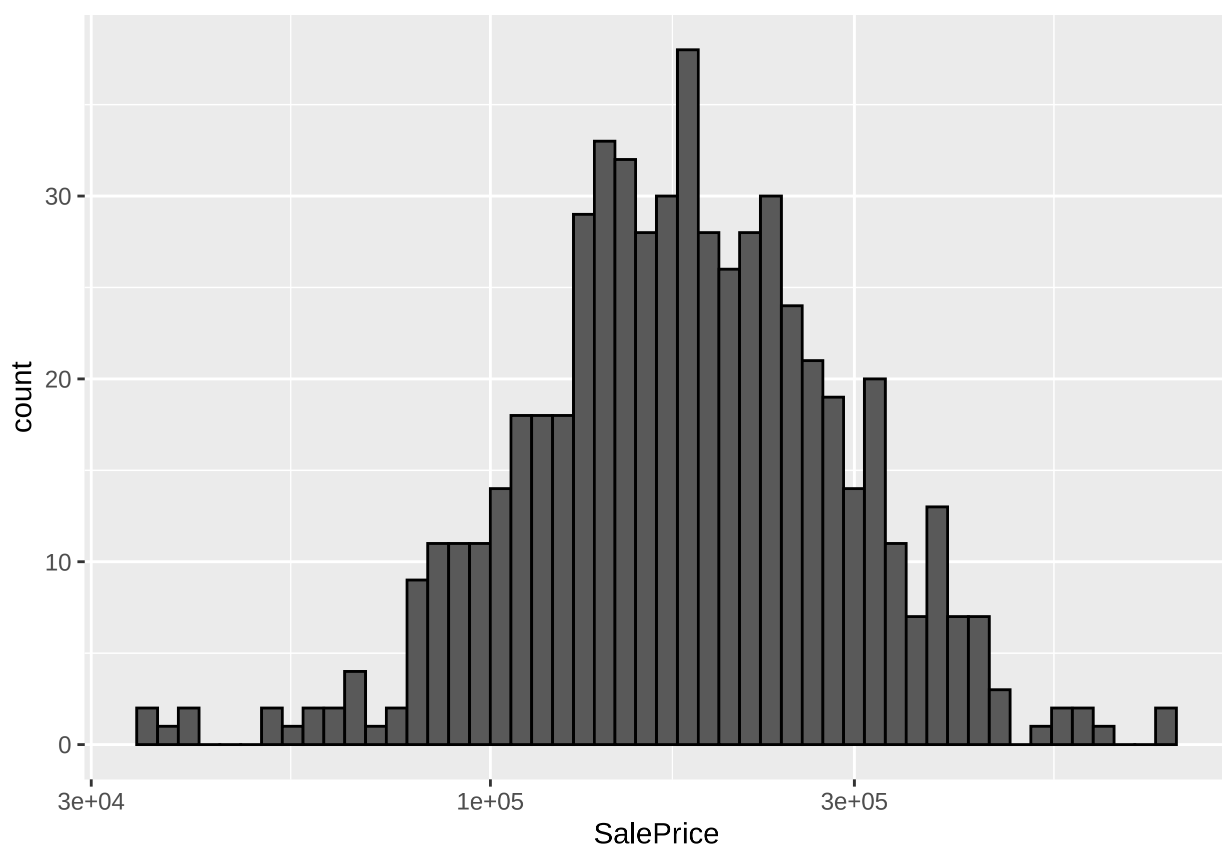
From the chart and summary table below, we can see that there is significant variation in the sales price (SalePrice), which is the the value we wish to predict.

- Minimum bin = 34,900
- Mean bin = 200,545
- Maximum bin = 755,000

```
# Grouping sale prices

sp_grouped <- train_set %>%
  group_by(SalePrice) %>%
  summarize(n=n())
```

```
## 'summarise()' ungrouping output (override with '.groups' argument)
```

```
# Producing a Histogram of Sale prices

sp_grouped %>%
  ggplot(aes(SalePrice)) +
  geom_histogram(bins = 50, color = "black") +
  scale_x_log10()
```



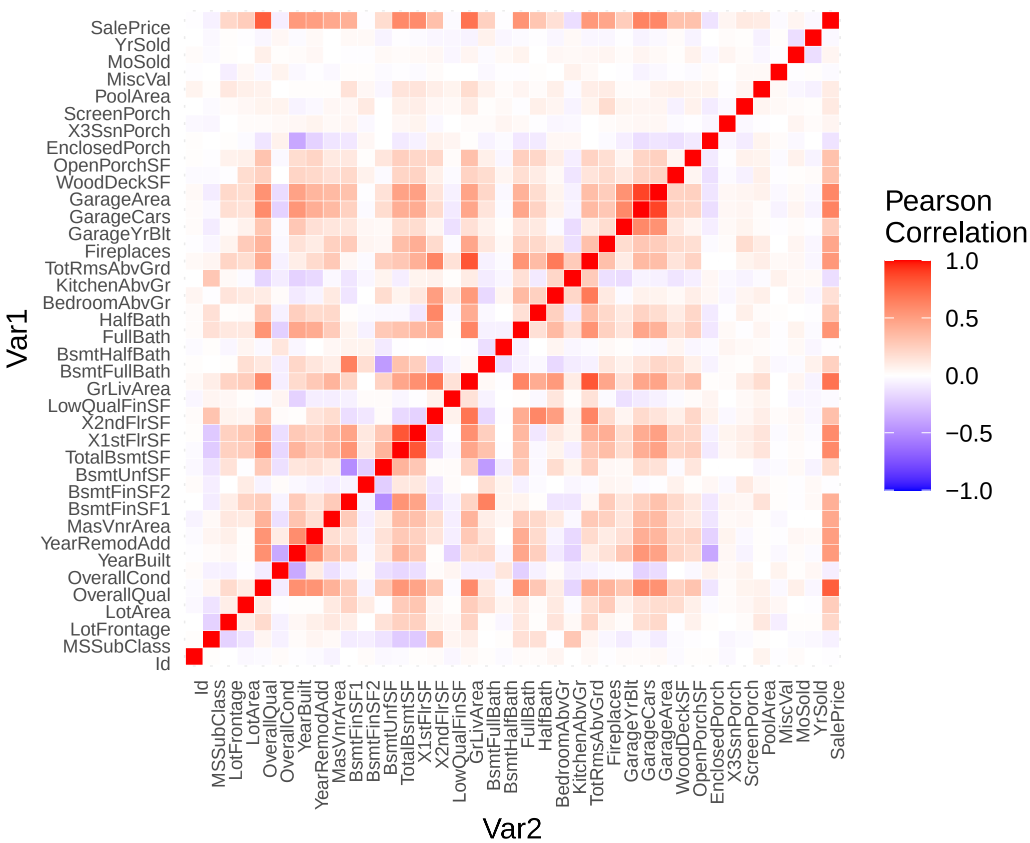
```r
# Summary of sale prices

summary(sp_grouped)
```

```
##    SalePrice            n
##  Min.   : 34900   Min.   : 1.000
##  1st Qu.:134000   1st Qu.: 1.000
##  Median :179400   Median : 1.000
##  Mean   :200545   Mean   : 2.103
##  3rd Qu.:245350   3rd Qu.: 2.000
##  Max.   :755000   Max.   :19.000
```

Correlation Matrix

To gain an insight into the relationship between variables, I will generate the correlation matrix and produce a list of the variables that are correlated with SalePrice.

```r
# Generating the correlation matrix

cor_train <- round(cor(train_set[, !sapply(train_set, is.factor)]),2)

# Producing a heat map of correlations

melted <- melt(cor_train, na.rm = TRUE)

# Heatmap

ggplot(data = melted, aes(Var2, Var1, fill = value))+
 geom_tile(color = "white")+
 scale_fill_gradient2(low = "blue", high = "red", mid = "white",
   midpoint = 0, limit = c(-1,1), space = "Lab",
   name="Pearson\nCorrelation") +
  theme_minimal()+
 theme(axis.text.x = element_text(angle = 90, vjust = 1,
    size = 7, hjust = 1), axis.text.y = element_text(vjust = 1,
    size = 7, hjust = 1))+
 coord_fixed()
```




```r
# Producing a table of variables correlated with Sale Price

price_cor <- melted %>% filter(Var1=="SalePrice") %>% arrange(desc(value))
price_cor %>% knitr:: kable()
```

Var1	Var2	value
SalePrice	SalePrice	1.00
SalePrice	OverallQual	0.79
SalePrice	GrLivArea	0.70
SalePrice	GarageCars	0.63
SalePrice	GarageArea	0.61
SalePrice	TotalBsmtSF	0.60
SalePrice	X1stFlrSF	0.59
SalePrice	FullBath	0.55
SalePrice	TotRmsAbvGrd	0.53
SalePrice	YearBuilt	0.52
SalePrice	YearRemodAdd	0.50
SalePrice	Fireplaces	0.46
SalePrice	MasVnrArea	0.45
SalePrice	BsmtFinSF1	0.41
SalePrice	X2ndFlrSF	0.33
SalePrice	WoodDeckSF	0.32
SalePrice	OpenPorchSF	0.32
SalePrice	HalfBath	0.30
SalePrice	LotArea	0.26
SalePrice	GarageYrBlt	0.26
SalePrice	BsmtFullBath	0.24
SalePrice	LotFrontage	0.21
SalePrice	BsmtUnfSF	0.18
SalePrice	BedroomAbvGr	0.16
SalePrice	ScreenPorch	0.11
SalePrice	PoolArea	0.10
SalePrice	X3SsnPorch	0.05
SalePrice	MoSold	0.05
SalePrice	Id	-0.01
SalePrice	BsmtFinSF2	-0.01
SalePrice	BsmtHalfBath	-0.01
SalePrice	LowQualFinSF	-0.02
SalePrice	MiscVal	-0.02
SalePrice	YrSold	-0.03
SalePrice	MSSubClass	-0.06
SalePrice	OverallCond	-0.07
SalePrice	EnclosedPorch	-0.12
SalePrice	KitchenAbvGr	-0.14

As can be seen from above only 38 out of 79 property features actually correlate with the sale price.

The top 10 correlated features are:

```r
# Top 10 features correlated with SalePrice sorted by absolute correlation

price_cor <- price_cor %>% filter(Var2 !="SalePrice") %>% arrange(desc(abs(value)))

head(price_cor,10,value)
```

```
##         Var1         Var2 value
## 1  SalePrice  OverallQual  0.79
## 2  SalePrice    GrLivArea  0.70
```

```
## 3  SalePrice   GarageCars  0.63
## 4  SalePrice   GarageArea  0.61
## 5  SalePrice  TotalBsmtSF  0.60
## 6  SalePrice    X1stFlrSF  0.59
## 7  SalePrice     FullBath  0.55
## 8  SalePrice TotRmsAbvGrd  0.53
## 9  SalePrice    YearBuilt  0.52
## 10 SalePrice YearRemodAdd  0.50
```

We will explore using some of these top features to build our initial regression models.

PREDICTIVE MODELS - Basic Regression:

1) Mean model

For our first model, we will simply assume that the Sales Price is equal to the average of all sale prices in our train set.
The equation for this model is:

$$Y_{u,i} = \mu + e_i$$

```r
# Calculating the mean rating

mu <- mean(train_set$SalePrice)

mu
```

```
## [1] 181049.2
```

The mean sale price = 181,049.2

```r
# Calculating the RMSLE using our predefined function

RMSLE_mu <- RMSLE(mu)

# Creating a tibble to store RMSEs

results <- tibble(Model="Mean Rating", RMSLE=format(round(RMSLE_mu,5),nsmall=5))
results %>% knitr::kable()
```

RMSLE

Model	RMSLE
Mean Rating	0.34941

The basic model produces an **RMSLE = 0.34941**.

2) Adding in Overall Quality effect

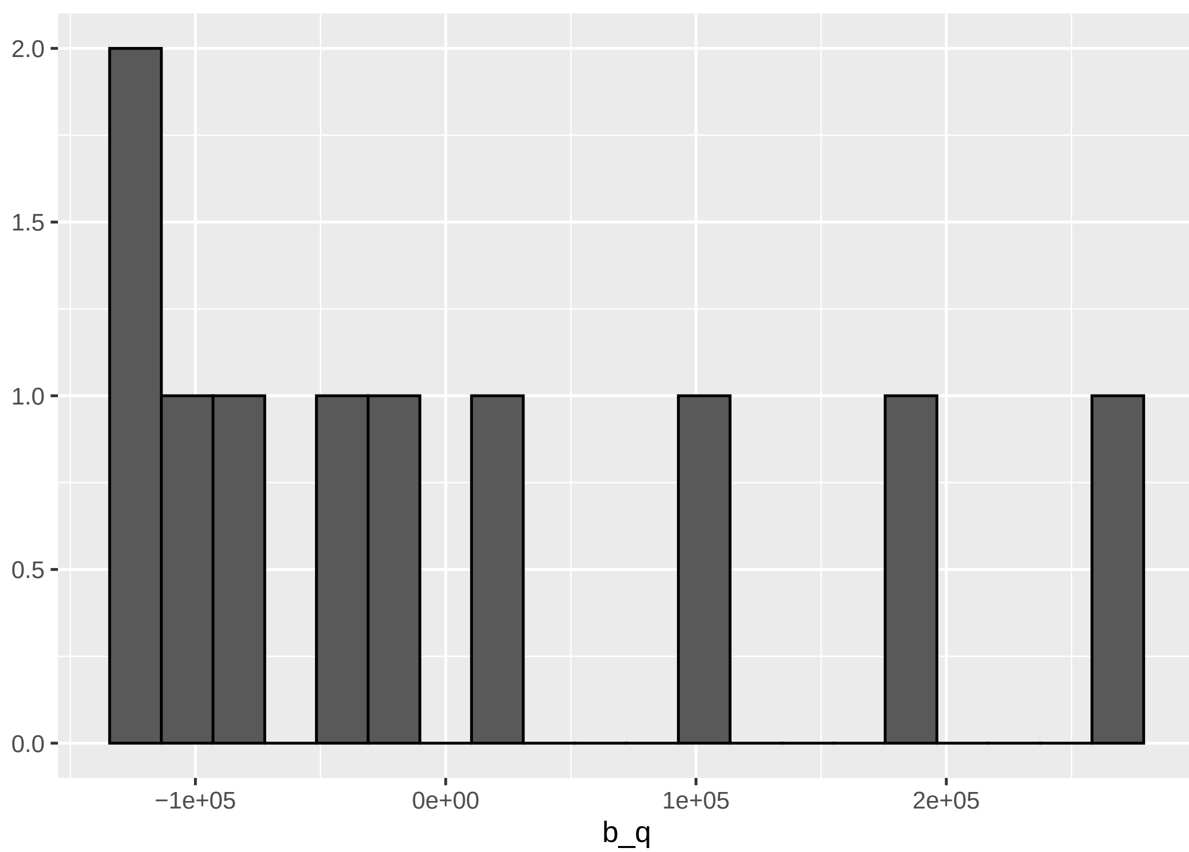
We will now add in the OverallQual variable to our regression equation.

Given the high correlation with SalePrice shown in the correlation matrix, I would expect adding this variable will reduce the RMSLE.

Our equation now becomes:

$$Y_i = \mu + b_q + e_i$$

```
### Overall Quality effect model

# Generating the estimates of b_q

b_q <- train_set %>%
  group_by(OverallQual) %>%
  summarise(n=n(), b_q = mean(SalePrice-mu))


# Creating a plot of b_q

qplot(b_q, data=b_q, bins=20, color=I("black"))
```



```r
# Predicted sale price

pred_qual <- mu + test_set %>%
  left_join(b_q, by='OverallQual') %>%
  pull(b_q)
```

As can be seen from the plot above, there is significant variability in b_q values. Hence we would expect it to be a good predictor of sale price.

```r
# Calculating the RMSLE using our predefined function

RMSLE_qual <- RMSLE(pred_qual)


# Adding results to our dataframe

results <- rbind(results, c(Model="Overall Quality effect",
                             RMSLE=format(round(RMSLE_qual,5),nsmall=5)))


results %>% knitr::kable()
```

RMSLE

Model	RMSLE
Mean Rating	0.34941
Overall Quality effect	0.21905

This model generated an **RMSLE = 0.21905**, which is a 37.3% improvement against the basic mean model.

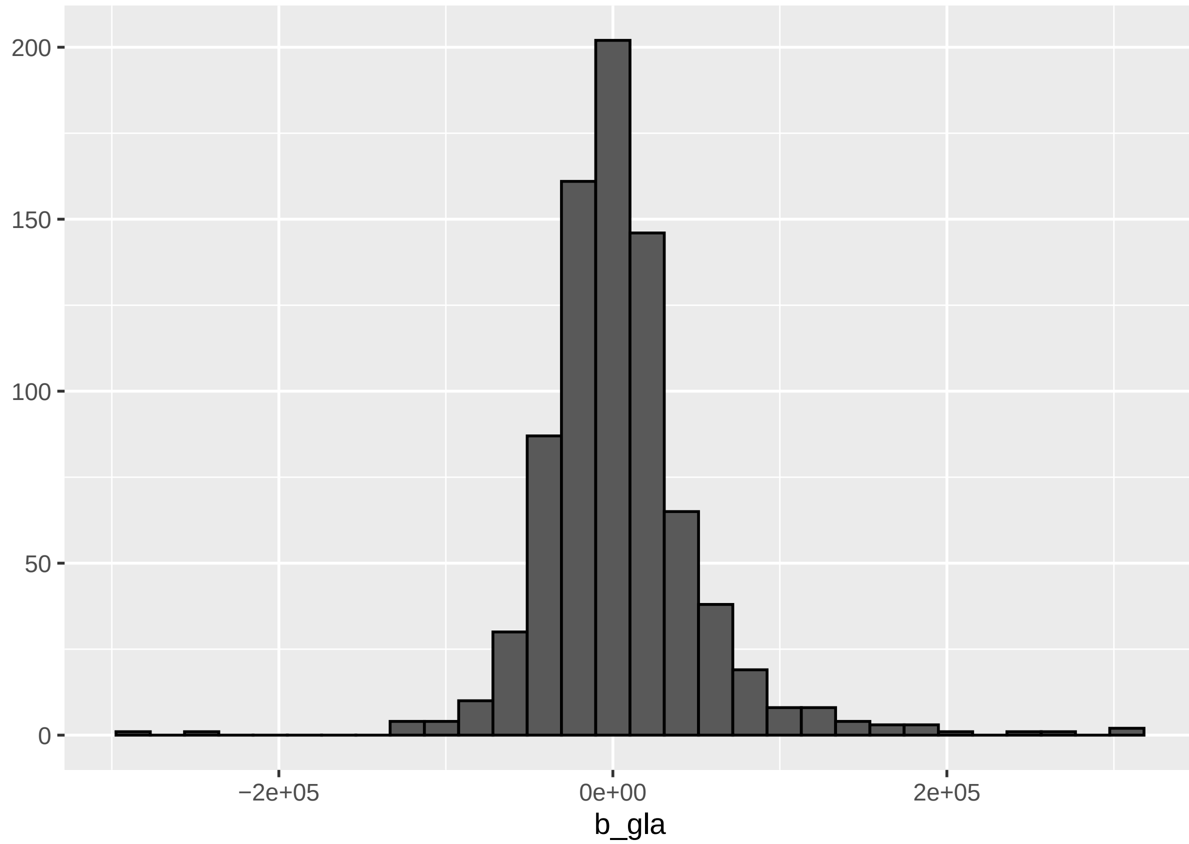
Hence, the *OverallQual* variable is a good predictor of sale price, as expected.

3) Overall Quality and Gross Living Area model

We will now explore adding in the second strongest correlated variable, GrLivArea, which represents the Gross Living Area.

$$Y_i = \mu + b_q + b_{gla} + e_i$$

```r
### Overall Quality & Gross Living Area effect model

# Generating the estimates of b_gla

gla_avg <- train_set %>%
  left_join(b_q, by='OverallQual') %>%
  group_by(GrLivArea) %>%
  summarise(b_gla=mean(SalePrice - mu - b_q))


# Creating a plot of b_gla

qplot(b_gla, data=gla_avg, bins=30, color=I("black"))
```




```r
# Generating Predictions

pred_ql <- test_set %>%
  left_join(b_q, by='OverallQual') %>%
  left_join(gla_avg, by='GrLivArea') %>%
  mutate(prediction = mu + b_q + b_gla) %>%
  pull(prediction)
```

From the plot of b_gla above, I noticed that whilst there is evidence of variability the majority of values are centred around 0. This would lead me to predict a low value add from inserting this variable to our model.

```r
# Calculating the RMSLE using our predefined function

RMSLE_ql <- RMSLE(pred_ql)

# Adding results to our dataframe

results <- rbind(results, c(Model="Overall Quality & Gross Living Area effect",
                            RMSLE=format(round(RMSLE_ql ,5),nsmall=5)))

results %>% knitr::kable()
```

RMSLE

Model	RMSLE
Mean Rating	0.34941
Overall Quality effect	0.21905
Overall Quality & Gross Living Area effect	0.26697

This model generated an **RMSLE = 0.26697**, which is a 23.6% improvement against the basic mean model.

However, our RMSLE actually increased by 21.9% relative to the RMSLE of the Overall Quality effect model.

Hence, I will drop this variable and replace it with the next highly correlated variable GarageCars.

4) Overall Quality and Garage Cars effect

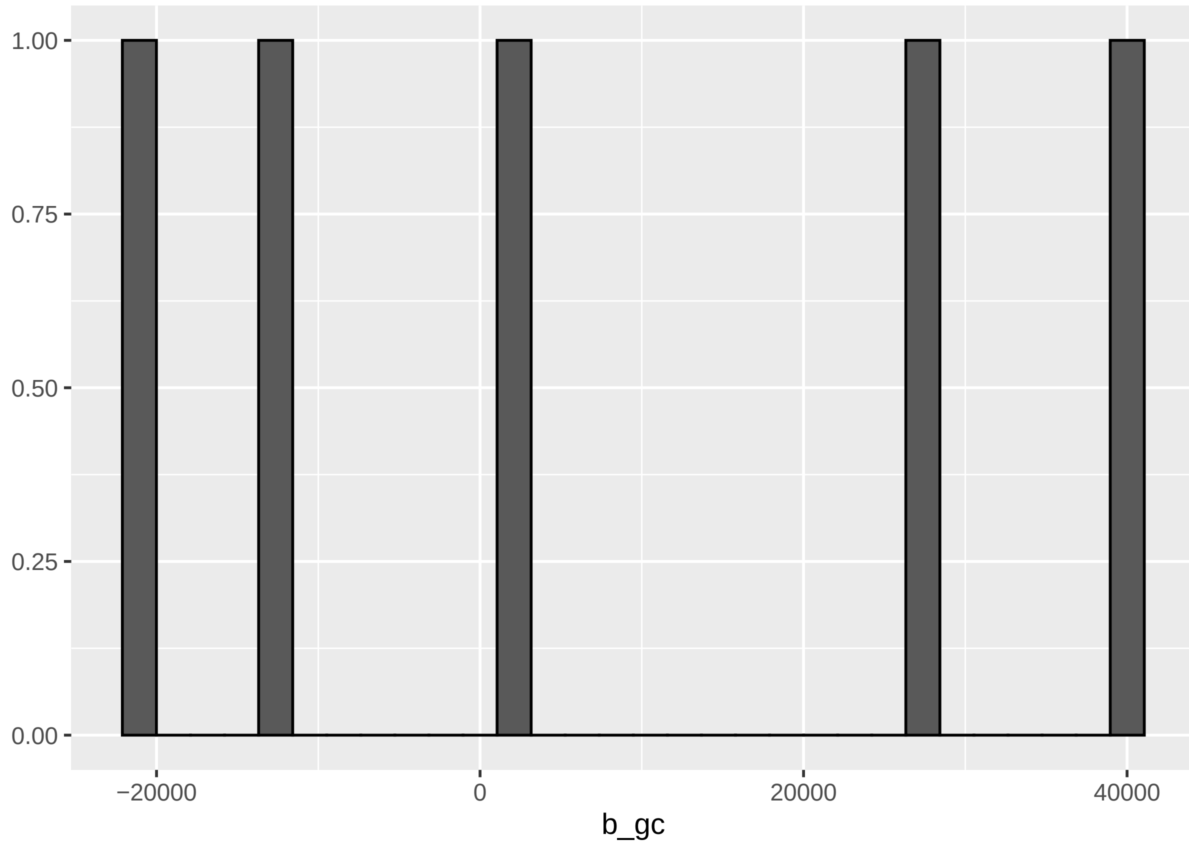
We now explore adding in a Garage Cars effect, given it was the second highest correlated feature with sale price.

$$Y_i = \mu + b_q + b_{gc} + e_i$$

```
### Overall Quality and Garage Cars effect model

# Generating the estimates of b_gc

gc_avg <- train_set %>%
  left_join(b_q, by='OverallQual') %>%
  group_by(GarageCars) %>%
  summarise(b_gc=mean(SalePrice - mu - b_q))
```

```
## 'summarise()' ungrouping output (override with '.groups' argument)
```

```
# Creating a plot of b_gc

qplot(b_gc, data=gc_avg, bins=30, color=I("black"))
```




```r
# Generating predictions

pred_qg <- test_set %>%
  left_join(b_q, by='OverallQual') %>%
  left_join(gc_avg, by='GarageCars') %>%
  mutate(prediction = mu + b_q + b_gc) %>%
  pull(prediction)
```

Based on the plot of b_gc above, it is clear that there is large variability in b_gc values. Hence, I would expect that this variable will help improve our prediction of sale prices.

```r
# Calculating the RMSLE using our predefined function

RMSLE_qg <- RMSLE(pred_qg)

# Adding results to our dataframe

results <- rbind(results, c(Model="Overall Quality & Garage Cars effect",
                            RMSLE=format(round(RMSLE_qg ,5),nsmall=5)))

results %>% knitr::kable()
```

RMSLE

Model	RMSLE
Mean Rating	0.34941
Overall Quality effect	0.21905
Overall Quality & Gross Living Area effect	0.26697
Overall Quality & Garage Cars effect	0.20642

This model generated an **RMSLE = 0.20642** which is a 40.9% improvement against the basic model and a 5.8% improvement against the Overall Quality effect model.

PREDICTIVE MODELS - Advanced:

We will now explore using more advanced machine learning techniques to attempt to generate further improvements in our model's predictive abilities.

We explore the performance of a series of advanced machine learning algorithms:

- Regression Tree (rpart)
- Random Forest (rf)
- K-Nearest Neighbours (knn)
- Stochastic Gradient Boost (gbm)

We will utilise the *Caret* package to train our respective models and generate their predicted values. The steps we will take will be as follows:

1. Train our model
2. Predict the Sale Price using the test set
3. Calculate the RMSLE
4. Evaluate the models performance

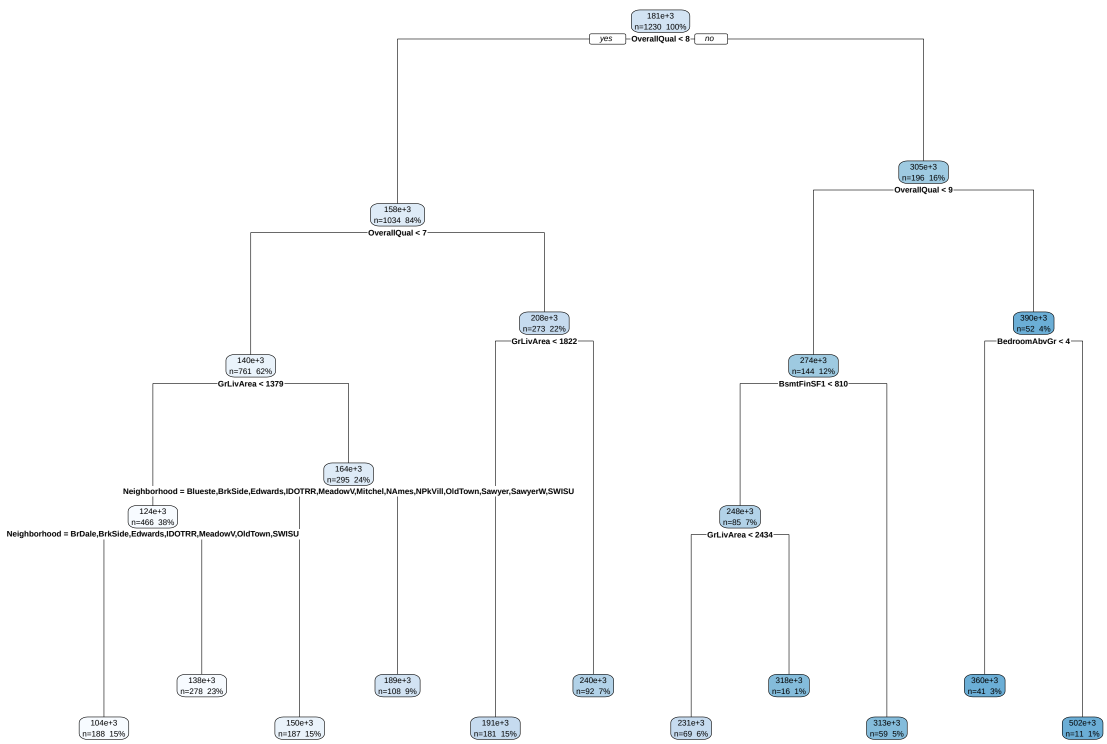
In order to produce **reproducible results** we will utilise the *Train Control* parameter and *Set.Seed*, where necessary, in our code.

Regression Tree

The regression tree approach utilises binary recursive partitioning to split the data into partitions / branches and then continue splitting each partition into smaller groups.

This approach is generally more easy to interpret than other more advanced models, but given the simplicity it often is not the most accurate.

```r
# Generating the tree

tree <- rpart(SalePrice ~., data=train_set)

# Plotting the tree

rpart.plot(tree, extra = 101)
```



From the tree above, I noticed that the OverallQual is the key decision variable and the threshold that generates a lower sale price, at the first step, is 8 which is quite low. This low threshold makes me cast doubt on the predictive power of this approach.

```r
# Training our model

train_rpart <- train(SalePrice ~., method="rpart", data=train_set)

# Predictions

pred_rpart <- predict(train_rpart, test_set, type="raw")

# Calculating the RMSLE

RMSLE_rpart <- RMSLE(pred_rpart)

results <- rbind(results, c(Model="R Part",
```

```
                    RMSLE=format(round(RMSLE_rpart, 5),nsmall=5)))
results %>% knitr::kable()
```

Model	RMSLE
Mean Rating	0.34941
Overall Quality effect	0.21905
Overall Quality & Gross Living Area effect	0.26697
Overall Quality & Garage Cars effect	0.20642
R Part	0.25823

Variable importance

```
# Generating Variable Importance
varImp(train_rpart)
```

```
## rpart variable importance
##
##   only 20 most important variables shown (out of 260)
##
##                      Overall
## OverallQual           100.00
## GrLivArea              81.33
## GarageCars             75.47
## ExterQualTA            43.05
## YearBuilt              39.15
## FullBath               37.46
## ExterQualGd            33.86
## PoolQCGd                0.00
## BsmtFullBath            0.00
## ScreenPorch             0.00
## GarageQualTA            0.00
## TotalBsmtSF             0.00
## BldgTypeTwnhsE          0.00
## HouseStyle2.5Unf        0.00
## RoofMatlRoll            0.00
## NeighborhoodNridgHt     0.00
## HouseStyleSLvl          0.00
## Condition1RRNe          0.00
## FenceMnWw               0.00
## Exterior1stBrkFace      0.00
```

From the table above, we note that the top 3 variables considered important by this model are:

- OverallQual
- GrLivArea
- GarageCars

```r
# Generating a list of predictor names

ind <- !(train_rpart$finalModel$frame$var == "<leaf>")

tree_terms <-
   train_rpart$finalModel$frame$var[ind] %>%
   unique() %>%
   as.character()

tree_terms
```

```
## [1] "OverallQual"
```

As can be seen from the table above, the rpart approach only took "OverallQual" as a predictor.

We would therefore expect our RMSLE for this model to be similar to our linear Overall Quality effect model, as it includes the same predictor.

The **RMSLE = 0.25823**, which is a 26.1% improvement in RMSLE relative to the basic model. However, it actually is an increase of 25% relative to the *Overall Quality & Garage Cars effect model.*

This increase was expected for reasons described earlier.

Random Forest

The Random forest is a classification algorithm which consists of many decision trees.Random forests are popular for numerous reasons including:

- They run efficiently on large datasets
- It can handle thousands of variables as inputs

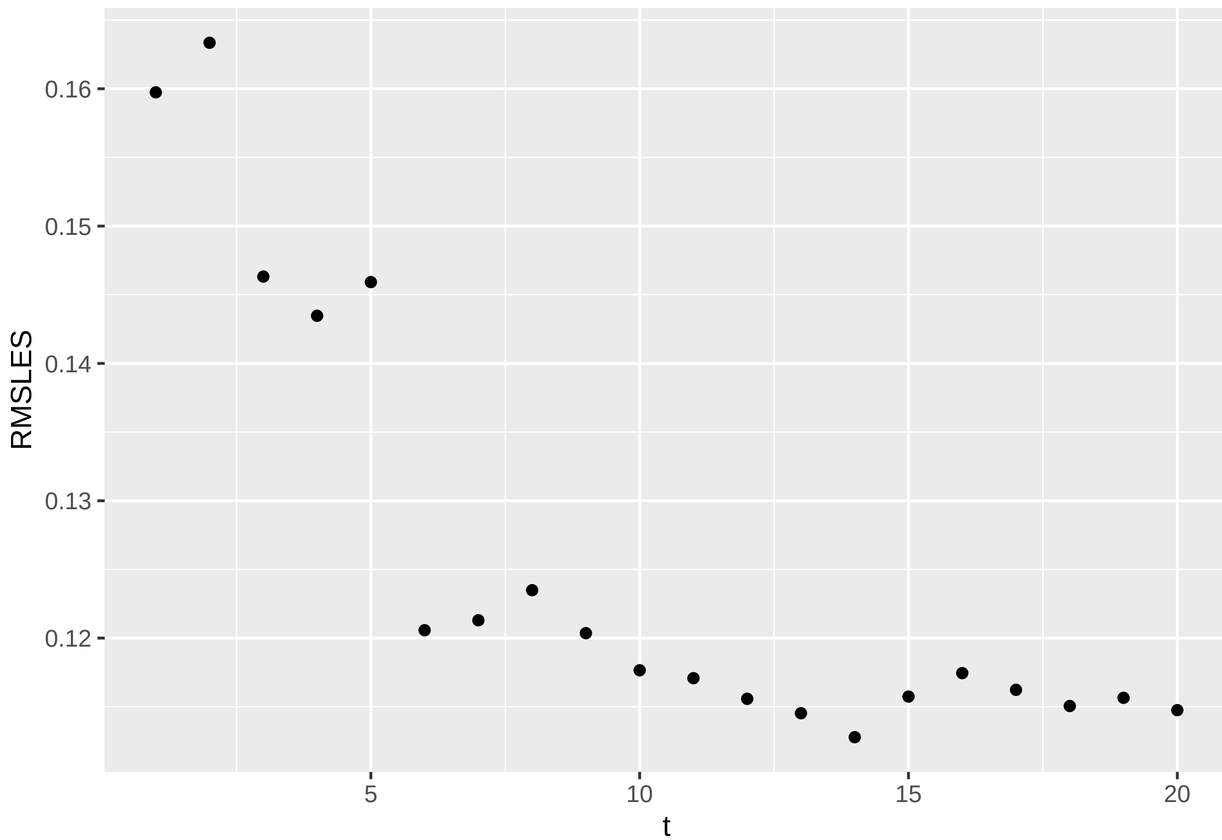
The optimal model has approx 500 trees and produces an RMSLE which is not significantly smaller than that produced using 14 trees, which I found to be the optimal value (see below).

Hence for reproducibility and simplicity I have used a sequence of 1 to 20 trees to determine the optimal number of trees for our train data.

```r
# Determining optimal number of trees

t <- seq(1,20,1)

set.seed(1,sample.kind = "Rounding")

RMSLES <- sapply(t, function(t){

  set.seed(1,sample.kind = "Rounding")
  train_rf <- train(SalePrice ~., method="rf", data=train_set, ntree=t)

  pred_rf <- predict(train_rf, test_set, type="raw")
```

```r
  return(RMSLE(pred_rf))
})
```

```r
# Plotting the results

qplot(t, RMSLES)
```



From the plot of RMSLE against the number of trees, we notice that there is a significant drop in RMSLE when using up to 5 trees. The fall in RMSLE continues, albeit at a smaller rate, until we arrive at 14 trees after which we observe an up tick in RMSLE.

```r
# Determining the number of trees that minimises the RMSLE

t_opt <- t[which.min(RMSLES)]

t[which.min(RMSLES)]
```

```
## [1] 14
```

```r
RMSLE_rf <- min(RMSLES)


# Adding RMSLE to the table
```

```r
results <- rbind(results, c(Model="Random Forest",
                            RMSLE=format(round(RMSLE_rf, 5),nsmall=5)))

results %>% knitr::kable()
```

Model	RMSLE
Mean Rating	0.34941
Overall Quality effect	0.21905
Overall Quality & Gross Living Area effect	0.26697
Overall Quality & Garage Cars effect	0.20642
R Part	0.25823
Random Forest	0.11278

The **RMSLE from this model = 0.11278** which is a 67.7% decrease relative to the RMSLE of the basic model and a *56.3% decrease* against the Rpart model and *45.4% decrease* against the Overal Quality and Garage Cars effect model.

Hence, this is our winning candidate model thus far.

```r
# Generating the predictions for Random Forest for use later

 train_rf <- train(SalePrice ~., method="rf", data=train_set, ntree=t_opt)

  pred_rf <- predict(train_rf, test_set, type="raw")

  pred_rf <- unname(pred_rf)
```

Variable importance

The output below shows the variables considered important by this model:

```r
# Generating Variable Importance

varImp(train_rf)
```

```
## rf variable importance
##
##   only 20 most important variables shown (out of 260)
##
##                 Overall
## OverallQual     100.000
## GrLivArea        32.116
## ExterQualTA      23.703
## TotalBsmtSF      18.726
## X1stFlrSF        13.415
## GarageCars       10.734
## GarageArea        7.687
## BsmtFinSF1        7.284
## X2ndFlrSF         7.205
## LotArea           5.257
## FullBath          3.947
## FoundationPConc   3.345
```

```
## YearRemodAdd      2.994
## YearBuilt         2.911
## MasVnrArea        2.717
## GarageYrBlt       2.358
## BsmtUnfSF         2.317
## LotFrontage       1.372
## TotRmsAbvGrd      1.353
## BedroomAbvGr      1.325
```

The top 3 most important variables are:

- OverallQual
- GarageCars
- ExterQualTA

K-Nearest Neighbours

The K-Nearest Neighbours model is a supervised machine learning algorithm that assumes that similar things exist in close proximity to each other.

One of the main advantages in the KNN algorithm over others is that:

- It is capable of performing multi-class classification
- It is an efficient algorithm and often produces results quickly

```r
# Generating the model and estimates

set.seed(1, sample.kind="Rounding")

train_knn <- train(SalePrice ~., method="knn", data=train_set)

pred_knn <- predict(train_knn, test_set, type="raw")

# Calculating the RMSLE

RMSLE_knn <- RMSLE(pred_knn)


# Adding RMSLE to the table

results <- rbind(results, c(Model="KNN",
                            RMSLE=format(round(RMSLE_knn, 5),nsmall=5)))

results %>% knitr::kable()
```

Model	RMSLE
Mean Rating	0.34941
Overall Quality effect	0.21905
Overall Quality & Gross Living Area effect	0.26697
Overall Quality & Garage Cars effect	0.20642
R Part	0.25823
Random Forest	0.11278
KNN	0.19773

The **RMSLE from this model = 0.19773** which is a 43.4% decrease against the basic model. However, it actually represents an increase of 75.3% against the random forest model.

Variable importance

The output below shows the variables considered important by this model:

```r
# Generating Variable Importance

varImp(train_knn)
```

```
## loess r-squared variable importance
##
##   only 20 most important variables shown (out of 80)
##
##              Overall
## OverallQual   100.00
## GrLivArea      81.41
## TotalBsmtSF    70.80
## GarageArea     67.13
## ExterQual      65.22
## GarageCars     63.62
## X1stFlrSF      63.37
## KitchenQual    57.98
## BsmtQual       54.68
## FullBath       48.74
## BsmtFinSF1     45.78
## TotRmsAbvGrd   44.76
## YearBuilt      43.92
## X2ndFlrSF      42.71
## YearRemodAdd   40.97
## MasVnrArea     34.36
## Fireplaces     33.60
## GarageFinish   29.52
## GarageType     29.18
## HeatingQC      25.70
```

The top 3 most important variables are:

- OverallQual
- GrLivArea
- TotalBsmtSF

As you will note, the TotalBsmtSF variable appears in the top 3 for this model but was towards the bottom of the list in the random forest model.

Perhaps this explains explain the increase in RMSLE observed. We will revisit this assertion when determining the most important variables for the subsequent models.

Gradient boosting

Gradient boosting is a ML technique which produces a prediction model in the form of an ensemble typically of decision trees. The model is built in a stage by stage approach.

From my research online, I noted that Gradient Boost models are often the winning models used for a number of Kaggle competition, hence I will attempt to explore one of these models with the housing dataset.

Stochastic Gradient Boosting

```
### Please note that this part of the code may take several minutes to run!

# Train control to ensure reproducibility

set.seed(321, sample.kind = "Rounding")

seeds <- vector(mode = "list", length = 51)

for(i in 1:50) seeds[[i]] <- sample.int(1000, 20)

seeds[[51]] <- sample.int(1000, 1)


my_cont <- trainControl(number= 5, seeds=seeds)



# Applying a Stochastic gradient boost

set.seed(1, sample.kind = "Rounding")

train_gb <- train(SalePrice ~., method="gbm", data=train_set,trControl=my_cont)

pred_gb <- predict(train_gb, test_set, type="raw")
```

```
# RMSLE

RMSLE_gb <- RMSLE(pred_gb)


# Adding RMSLE to the table

results <- rbind(results, c(Model="Stochastic Gradient Boosting",
                            RMSLE=format(round(RMSLE_gb, 5),nsmall=5)))

results %>% knitr::kable()
```

Model	RMSLE
Mean Rating	0.34941
Overall Quality effect	0.21905
Overall Quality & Gross Living Area effect	0.26697
Overall Quality & Garage Cars effect	0.20642
R Part	0.25823
Random Forest	0.11278
KNN	0.19773
Stochastic Gradient Boosting	0.11627

The **RMSLE from this model = 0.11627** which is a 66.7% decrease against the basic model. However, the RMSLE increased 3.1% relative to the random forest model.

Hence, the Random Forest model still wins.

Variable importance

The output below shows the variables considered important by this model:

```
# Generating Variable Importance

varImp(train_gb)
```

```
## gbm variable importance
##
##   only 20 most important variables shown (out of 260)
##
##                  Overall
## OverallQual      100.000
## GrLivArea         32.727
## BsmtFinSF1        17.777
## TotalBsmtSF       14.127
## GarageCars        10.885
## X1stFlrSF         10.179
## YearBuilt          9.231
## X2ndFlrSF          8.372
## LotArea            7.246
## TotRmsAbvGrd       3.648
## OpenPorchSF        3.582
## YearRemodAdd       2.910
## FullBath           2.818
## GarageTypeAttchd   2.424
## Fireplaces         2.290
## OverallCond        2.040
## GarageArea         1.861
## LotFrontage        1.346
## BedroomAbvGr       1.180
## MasVnrArea         1.117
```

The top 3 most important variables are:

- OverallQual

- GrLivArea
- BsmtFinSF1

Interestingly, the TotalBsmtSF variable features at number 4 and yet this model produces an RMSLE of 0.11627.

However, I noticed that this model place less importance on this variable (17.8) and the remaining variables. Thus whilst not a great predictor, TotalBsmtSF does have some importance in predicting sale price.

Ensemble - Averaging

We will explore whether building an ensemble allows us to improve the accuracy of our prediction of sale prices.

We will take the approach of creating our ensemble using the logarithmic average of the top 2 performing models.

- Random Forest
- Stochastic Gradient Boost (gbm)

```r
# Computing our estimate of Sale Prices

pred_ensemble <- exp((log(pred_gb) + log(pred_rf))/ 2)

# Calculating the RMSLE

RMSLE_ensemble <- RMSLE(pred_ensemble)

# Adding RMSLE to the table

results <- rbind(results, c(Model="Ensemble",
                            RMSLE=format(round(RMSLE_ensemble, 5),nsmall=5)))

results %>% knitr::kable()
```

Model	RMSLE
Mean Rating	0.34941
Overall Quality effect	0.21905
Overall Quality & Gross Living Area effect	0.26697
Overall Quality & Garage Cars effect	0.20642
R Part	0.25823
Random Forest	0.11278
KNN	0.19773
Stochastic Gradient Boosting	0.11627
Ensemble	0.10447

The **RMSLE from this model = 0.10447** which is a 70.1% decrease against the basic model.

The ensemble model now produces the lowest RMSLE and therefore becomes our **final model**.

Section 3 - Results

The below table summarises the RMSEs obtained by applying the respective models.

```r
# Generating the Results table to compare the performance of the various models

results %>% arrange(desc(RMSLE)) %>% knitr::kable()
```

Model	RMSLE
Mean Rating	0.34941
Overall Quality & Gross Living Area effect	0.26697
R Part	0.25823
Overall Quality effect	0.21905
Overall Quality & Garage Cars effect	0.20642
KNN	0.19773
Stochastic Gradient Boosting	0.11627
Random Forest	0.11278
Ensemble	0.10447

From the table above, the following in clear:

1. The OverallQual variable is a very significant predictor of the sale price.
2. Regression models are successful in generating predictions for sale prices, with the best regression model tested having generated a 41% reduction in RMSLE.
3. However, we generated very significant improvements in predictive power when we applied advanced machine learning algorithms, except for Rpart.
4. The regression tree model (Rpart) actually produced slightly worse estimates. This was due to it primarily basing estimates on OverallQual
5. The Random Forest model generated very strong estimates of sale prices, with an *RMSLE of 0.11278* which is 67.7% lower than our basic starting model.
6. The Stochastic Gradient Boosting model produced similar results to the Random Forest model, having produced an RMSLE = 0.11627. This is a *66.7% overall improvement* in RMSLE, however a *3.1% increase* relative to our Random Forest model.
7. The Ensemble model produced the best estimate of sale price. It generated an *RMSLE of 0.10447*, which is a 70.1% improvement against our basic mean model.
8. The RMSLE produced by our best model is within the top 95th percentile of RMSLEs from the competition leadership board. Hence I would consider this project very successful.

Performance against the Validation set

Our final model chosen using the train and test set was the Ensemble of Random Forest and Stochastic Gradient Boosting.

We will now test this against our final hold-out test set (validation). To do so we will carry out the following steps:

1. Generate predictions for the ensemble models, using the **sales_data** dataset (our new train set).
2. Combine the estimates of these models to produce the ensemble estimates.
3. Compare these estimates against the actual sales prices in the **validation** (test) data set.

Generating Stochastic Gradient Boosting estimates

```
# Generating the predictions by Stochastic Gradient Boosting


### Please note that this part of the code may take several minutes to run!

# Train control to ensure reproducibility

set.seed(321, sample.kind = "Rounding")

seeds <- vector(mode = "list", length = 51)

for(i in 1:50) seeds[[i]] <- sample.int(1000, 20)

seeds[[51]] <- sample.int(1000, 1)


my_cont <- trainControl(number= 5, seeds=seeds)



# Applying a Stochastic gradient boost

set.seed(1, sample.kind = "Rounding")

train_gb_v <- train(SalePrice ~., method="gbm", data=sales_data,trControl=my_cont)

pred_gb_v <- predict(train_gb_v, validation, type="raw")
```

Generating Random Forest estimates

We will generate the Random Forest estimates using the optimal number of trees (t_opt) determined earlier when using the test / train sets.

```r
### Generating our Random Forest estimates


# Training our final model on the Sales Data

  train_rf_v <- train(SalePrice ~., method="rf", data=sales_data, ntree=t_opt)

# Generating Predictions

  pred_rf_v <- predict(train_rf_v, validation, type="raw")

  pred_rf_v <- unname(pred_rf_v)
```

```r
### Combining the estimates to produce the Ensemble estimates

# Computing the ensemble estimates of Sale Prices

pred_ensemble_v <- exp((log(pred_gb_v) + log(pred_rf_v))/ 2)

# Calculating the RMSLE

RMSLE_ensemble_v <- RMSE(log(validation$SalePrice), log(pred_ensemble_v))


# Adding RMSLE to the table

results <- rbind(results, c(Model="Ensemble on Validation set",
                            RMSLE=format(round(RMSLE_ensemble_v, 5),nsmall=5)))

results %>% knitr::kable()
```

Model	RMSLE
Mean Rating	0.34941
Overall Quality effect	0.21905
Overall Quality & Gross Living Area effect	0.26697
Overall Quality & Garage Cars effect	0.20642
R Part	0.25823
Random Forest	0.11278
KNN	0.19773
Stochastic Gradient Boosting	0.11627
Ensemble	0.10447
Ensemble on Validation set	0.09941

The **RMSE of the Ensemble on the Validation set = 0.09941**.

I am very happy with this result, as it is within the top 95% of RMSLEs on the Kaggle leadership board.

Section 4 - Conclusion

In this project, we started off by using regression models to attempt to predict sale prices based on features of respective properties.

During our analysis we identified that the key variables that determine the sale price are:

```
# Top 10 features correlated with SalePrice sorted by absolute correlation

price_cor <- price_cor %>% arrange(desc(abs(value)))

head(price_cor,10,value) %>% knitr::kable()
```

Var1	Var2	value
SalePrice	OverallQual	0.79
SalePrice	GrLivArea	0.70
SalePrice	GarageCars	0.63
SalePrice	GarageArea	0.61
SalePrice	TotalBsmtSF	0.60
SalePrice	X1stFlrSF	0.59
SalePrice	FullBath	0.55
SalePrice	TotRmsAbvGrd	0.53
SalePrice	YearBuilt	0.52
SalePrice	YearRemodAdd	0.50

The final model chosen is the Ensemble of Random Forest and Stochastic Gradient Boosting models.

When testing this model against the Validation dataset, we obtained an **RMSLE = 0.09941**

The performance on the validation set is similar to that on our train_set, hence I would consider the ensemble model to be successful in generating a good estimate of sale prices.

Limitations

New features

Over time, we should expect new property features to be added. This would require us to retrain our model.

Regional Variation

This model was built based on data for a town in Iowa. Whilst there may be an overlap in features that apply to other towns and / or regions, we should expect that the importance of these features may differ.

For example in towns / regions with a large proportion of commuters, we may notice a significant importance in features that capture ease of access to transport links.

Hence we may observe a lack of direct applicability of our final model to datasets for other regions.

Changing preferences

Over time, client preferences may change making it difficult to predict sales prices.

For example, back in the early 20th century wallpaper was extremely popular. However, in the 21st century it is not.

Such change in taste can occur for a number of features / variables in this dataset and hence it is important that we retrain our final model periodically.

Future work

From my research, I noted that xgboost models typically produced the lowest RMSLE and often featured in competition winning models.

I attempted to apply this approach using "xgbDART", however I noticed that this method requires computing power beyond my PC's capabilities and will take a significant amount of time to run.

Hence, I excluded this model from my project in the interest of reproducibility of my results.

References

- https://www.kaggle.com/c/house-prices-advanced-regression-techniques
- https://rafalab.github.io/dsbook/caret.html
- https://topepo.github.io/caret/available-models.html

Link to GitHub repository

Please see below for a hyperlink to the GitHub repository for this project.

GitHub repository for this project

THANK YOU for taking the time to read my report, I hope you enjoyed it!