Harvard Data Science Capstone: House Price Prediction

Karlo Kostanjevec

Content

1. Introduction

The purpose of this project is to create House Price Prediction System based on the House Prices - Advanced Regression Techniques dataset from Kaggle. Our goals is to generate predictions of the final prices of houses. Regarding my motivation for picking this data-set for final project it is worth noting that in most cases buying a real-estate is the largest investments for the majority of people and, moreover, I am also currently in a process of buying a real estate. Therefore, the topic of house prices is important from both general and personal perspective. The link to the data-set and ongoing Kaggle competition is here: https://www.kaggle.com/c/house-prices-advanced-regression-techniques .

In order to build House Price Prediction System we will use machine learning (ML) approach. In general, machine learning is a variation of artificial intelligence where algorithms are used to improve a system automatically via experience and by the use of data. Therefore, machine learning can be seen as a cross-disciplinary field between data science and artificial intelligence.

More specifically, we will use advanced regression machine learning techniques and measure the performance of these techniques with residual mean squared error (RMSE) metrics which allows us to see typical error loss. Moreover, Kaggle web-site states that this data-set is a part of a contest, in which the submission with the lowest RMSE wins. The RMSE must be calculated between the logarithm of the predicted value and the logarithm of the observed sale price. Therefore, in order to measure the performance of ML predictions, described log RMSE metrics will be used.

As it can be seen on Kaggle web-page, our respective data-set is offered in two separated files, one for training (train.csv) and another one for testing (test.csv). The data-set has 79 explanatory variables describing (almost) every aspect of residential homes in Ames, Iowa, United States of America. Furthermore, through data exploration of the respective data-sets from Kaggle (test.csv and train csv) it will be visible that the test.csv has no SalePrice variable (it has 1 variable less than train.csv). Therefore, we will use train.csv for our ML models and the data-set will be divided in three parts: 60% for training, 20% for testing and 20% for final validation of trained models.

First, we will import the data (automatically, from my GitHub repo) and afterwards we will clean imported datasets. Afterwards, we will do some data engineering and remodeling, including the creation of three new variables which we will use for later purposes: data visualization and machine learning. Therefore, after data engineering we will visualize some aspects of the data that are crucial for future ML models, including correlations. Finally, in the fifth chapter we will train four ML models and test them on test set and perform final validation on validation set. In the last part of this project, we will summarize all results of our ML models, highlight the best performing model (random forest with train RMSE 0.144 and validation RMSE 0.134) and highlight some limitations and proposition for future improvement.

2. Data Import and Overview

Before proceeding, install and load the following packages:

```
knitr::opts_chunk$set(warning = FALSE, message = FALSE, echo = T)
if(!require(plyr)) install.packages("plyr", repos = "http://cran.us.r-project.org")
if(!require(tidyverse)) install.packages("tidyverse", repos = "http://cran.us.r-project.org")
if(!require(e1071)) install.packages("e1071", repos = "http://cran.us.r-project.org")
if(!require(caret)) install.packages("caret", repos = "http://cran.us.r-project.org")
if(!require(glmnet)) install.packages("glmnet", repos = "http://cran.us.r-project.org")
if(!require(Matrix)) install.packages("Matrix", repos = "http://cran.us.r-project.org")
if(!require(lattice)) install.packages("lattice", repos = "http://cran.us.r-project.org")
if(!require(dummies)) install.packages("dummies", repos = "http://cran.us.r-project.org")
if(!require(lares)) install.packages("lattice", repos = "http://cran.us.r-project.org")
load("my_work_space.RData")
```

The respective data can be downloaded from the GitHub repository with the code below:

```
training_data = read.csv(file = file.path("https://raw.githubusercontent.com/kkostanjevec/HarvardCap_Hou

test_data = read.csv(file = file.path("https://raw.githubusercontent.com/kkostanjevec/HarvardCap_House_
```

There is a third file called data_description, which gives details on every variable, and can also be found in the same GitHub repository: https://github.com/kkostanjevec/HarvardCap_House_price_pred/blob/main/data_description.txt

Data exploration shows that the data we need to analyze in order to create a ML models has two separate files: test.csv (test_data) has 1459 observations and 80 variables, while train.csv (training_data) has 1460 observations and 81 variable. There is 1 variable more in training_data than in test_data: test data has no SalePrice variable. Therefore, we will use training_data (train.csv) for ML training, testing and validating process.

```
str(training_data)
```

```
## 'data.frame':    1460 obs. of  81 variables:
##  $ Id           : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ MSSubClass   : int  60 20 60 70 60 50 20 60 50 190 ...
##  $ MSZoning     : chr  "RL" "RL" "RL" "RL" ...
##  $ LotFrontage  : int  65 80 68 60 84 85 75 NA 51 50 ...
##  $ LotArea      : int  8450 9600 11250 9550 14260 14115 10084 10382 6120 7420 ...
##  $ Street       : chr  "Pave" "Pave" "Pave" "Pave" ...
##  $ Alley        : chr  NA NA NA NA ...
##  $ LotShape     : chr  "Reg" "Reg" "IR1" "IR1" ...
##  $ LandContour  : chr  "Lvl" "Lvl" "Lvl" "Lvl" ...
##  $ Utilities    : chr  "AllPub" "AllPub" "AllPub" "AllPub" ...
```

```
##  $ LotConfig    : chr  "Inside" "FR2" "Inside" "Corner" ...
##  $ LandSlope    : chr  "Gtl" "Gtl" "Gtl" "Gtl" ...
##  $ Neighborhood : chr  "CollgCr" "Veenker" "CollgCr" "Crawfor" ...
##  $ Condition1   : chr  "Norm" "Feedr" "Norm" "Norm" ...
##  $ Condition2   : chr  "Norm" "Norm" "Norm" "Norm" ...
##  $ BldgType     : chr  "1Fam" "1Fam" "1Fam" "1Fam" ...
##  $ HouseStyle   : chr  "2Story" "1Story" "2Story" "2Story" ...
##  $ OverallQual  : int  7 6 7 7 8 5 8 7 7 5 ...
##  $ OverallCond  : int  5 8 5 5 5 5 5 6 5 6 ...
##  $ YearBuilt    : int  2003 1976 2001 1915 2000 1993 2004 1973 1931 1939 ...
##  $ YearRemodAdd : int  2003 1976 2002 1970 2000 1995 2005 1973 1950 1950 ...
##  $ RoofStyle    : chr  "Gable" "Gable" "Gable" "Gable" ...
##  $ RoofMatl     : chr  "CompShg" "CompShg" "CompShg" "CompShg" ...
##  $ Exterior1st  : chr  "VinylSd" "MetalSd" "VinylSd" "Wd Sdng" ...
##  $ Exterior2nd  : chr  "VinylSd" "MetalSd" "VinylSd" "Wd Shng" ...
##  $ MasVnrType   : chr  "BrkFace" "None" "BrkFace" "None" ...
##  $ MasVnrArea   : int  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   : int  706 978 486 216 655 732 1369 859 0 851 ...
##  $ BsmtFinType2 : chr  "Unf" "Unf" "Unf" "Unf" ...
##  $ BsmtFinSF2   : int  0 0 0 0 0 0 0 32 0 0 ...
##  $ BsmtUnfSF    : int  150 284 434 540 490 64 317 216 952 140 ...
##  $ TotalBsmtSF  : int  856 1262 920 756 1145 796 1686 1107 952 991 ...
##  $ Heating      : chr  "GasA" "GasA" "GasA" "GasA" ...
##  $ HeatingQC    : chr  "Ex" "Ex" "Ex" "Gd" ...
##  $ CentralAir   : chr  "Y" "Y" "Y" "Y" ...
##  $ Electrical   : chr  "SBrkr" "SBrkr" "SBrkr" "SBrkr" ...
##  $ X1stFlrSF    : int  856 1262 920 961 1145 796 1694 1107 1022 1077 ...
##  $ X2ndFlrSF    : int  854 0 866 756 1053 566 0 983 752 0 ...
##  $ LowQualFinSF : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ GrLivArea    : int  1710 1262 1786 1717 2198 1362 1694 2090 1774 1077 ...
##  $ BsmtFullBath : int  1 0 1 1 1 1 1 1 0 1 ...
##  $ BsmtHalfBath : int  0 1 0 0 0 0 0 0 0 0 ...
##  $ FullBath     : int  2 2 2 1 2 1 2 2 2 1 ...
##  $ HalfBath     : int  1 0 1 0 1 1 0 1 0 0 ...
##  $ BedroomAbvGr : int  3 3 3 3 4 1 3 3 2 2 ...
##  $ KitchenAbvGr : int  1 1 1 1 1 1 1 1 2 2 ...
##  $ KitchenQual  : chr  "Gd" "TA" "Gd" "Gd" ...
##  $ TotRmsAbvGrd : int  8 6 6 7 9 5 7 7 8 5 ...
##  $ Functional   : chr  "Typ" "Typ" "Typ" "Typ" ...
##  $ Fireplaces   : int  0 1 1 1 1 0 1 2 2 2 ...
##  $ FireplaceQu  : chr  NA "TA" "TA" "Gd" ...
##  $ GarageType   : chr  "Attchd" "Attchd" "Attchd" "Detchd" ...
##  $ GarageYrBlt  : int  2003 1976 2001 1998 2000 1993 2004 1973 1931 1939 ...
##  $ GarageFinish : chr  "RFn" "RFn" "RFn" "Unf" ...
##  $ GarageCars   : int  2 2 2 3 3 2 2 2 2 1 ...
##  $ GarageArea   : int  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   : int  0 298 0 0 192 40 255 235 90 0 ...
##  $ OpenPorchSF  : int  61 0 42 35 84 30 57 204 0 4 ...
##  $ EnclosedPorch: int  0 0 0 272 0 0 0 228 205 0 ...
##  $ X3SsnPorch   : int  0 0 0 0 0 320 0 0 0 0 ...
##  $ ScreenPorch  : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ PoolArea     : int  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      : int  0 0 0 0 0 700 0 350 0 0 ...
##  $ MoSold       : int  2 5 9 2 12 10 8 11 4 1 ...
##  $ YrSold       : int  2008 2007 2008 2006 2008 2009 2007 2009 2008 2008 ...
##  $ SaleType     : chr  "WD" "WD" "WD" "WD" ...
##  $ SaleCondition: chr  "Normal" "Normal" "Normal" "Abnorml" ...
##  $ SalePrice    : int  208500 181500 223500 140000 250000 143000 307000 200000 129900 118000 ...
```

```r
str(test_data)
```

```
## 'data.frame':    1459 obs. of  80 variables:
##  $ Id           : int  1461 1462 1463 1464 1465 1466 1467 1468 1469 1470 ...
##  $ MSSubClass   : int  20 20 60 60 120 60 20 60 20 20 ...
##  $ MSZoning     : chr  "RH" "RL" "RL" "RL" ...
##  $ LotFrontage  : int  80 81 74 78 43 75 NA 63 85 70 ...
##  $ LotArea      : int  11622 14267 13830 9978 5005 10000 7980 8402 10176 8400 ...
##  $ Street       : chr  "Pave" "Pave" "Pave" "Pave" ...
##  $ Alley        : chr  NA NA NA NA ...
##  $ LotShape     : chr  "Reg" "IR1" "IR1" "IR1" ...
##  $ LandContour  : chr  "Lvl" "Lvl" "Lvl" "Lvl" ...
##  $ Utilities    : chr  "AllPub" "AllPub" "AllPub" "AllPub" ...
##  $ LotConfig    : chr  "Inside" "Corner" "Inside" "Inside" ...
##  $ LandSlope    : chr  "Gtl" "Gtl" "Gtl" "Gtl" ...
##  $ Neighborhood : chr  "NAmes" "NAmes" "Gilbert" "Gilbert" ...
##  $ Condition1   : chr  "Feedr" "Norm" "Norm" "Norm" ...
##  $ Condition2   : chr  "Norm" "Norm" "Norm" "Norm" ...
##  $ BldgType     : chr  "1Fam" "1Fam" "1Fam" "1Fam" ...
##  $ HouseStyle   : chr  "1Story" "1Story" "2Story" "2Story" ...
##  $ OverallQual  : int  5 6 5 6 8 6 6 6 7 4 ...
##  $ OverallCond  : int  6 6 5 6 5 5 7 5 5 5 ...
##  $ YearBuilt    : int  1961 1958 1997 1998 1992 1993 1992 1998 1990 1970 ...
##  $ YearRemodAdd : int  1961 1958 1998 1998 1992 1994 2007 1998 1990 1970 ...
##  $ RoofStyle    : chr  "Gable" "Hip" "Gable" "Gable" ...
##  $ RoofMatl     : chr  "CompShg" "CompShg" "CompShg" "CompShg" ...
##  $ Exterior1st  : chr  "VinylSd" "Wd Sdng" "VinylSd" "VinylSd" ...
##  $ Exterior2nd  : chr  "VinylSd" "Wd Sdng" "VinylSd" "VinylSd" ...
##  $ MasVnrType   : chr  "None" "BrkFace" "None" "BrkFace" ...
##  $ MasVnrArea   : int  0 108 0 20 0 0 0 0 0 0 ...
##  $ ExterQual    : chr  "TA" "TA" "TA" "TA" ...
##  $ ExterCond    : chr  "TA" "TA" "TA" "TA" ...
##  $ Foundation   : chr  "CBlock" "CBlock" "PConc" "PConc" ...
##  $ BsmtQual     : chr  "TA" "TA" "Gd" "TA" ...
##  $ BsmtCond     : chr  "TA" "TA" "TA" "TA" ...
##  $ BsmtExposure : chr  "No" "No" "No" "No" ...
```

```
##  $ BsmtFinType1 : chr  "Rec" "ALQ" "GLQ" "GLQ" ...
##  $ BsmtFinSF1   : int  468 923 791 602 263 0 935 0 637 804 ...
##  $ BsmtFinType2 : chr  "LwQ" "Unf" "Unf" "Unf" ...
##  $ BsmtFinSF2   : int  144 0 0 0 0 0 0 0 0 78 ...
##  $ BsmtUnfSF    : int  270 406 137 324 1017 763 233 789 663 0 ...
##  $ TotalBsmtSF  : int  882 1329 928 926 1280 763 1168 789 1300 882 ...
##  $ Heating      : chr  "GasA" "GasA" "GasA" "GasA" ...
##  $ HeatingQC    : chr  "TA" "TA" "Gd" "Ex" ...
##  $ CentralAir   : chr  "Y" "Y" "Y" "Y" ...
##  $ Electrical   : chr  "SBrkr" "SBrkr" "SBrkr" "SBrkr" ...
##  $ X1stFlrSF    : int  896 1329 928 926 1280 763 1187 789 1341 882 ...
##  $ X2ndFlrSF    : int  0 0 701 678 0 892 0 676 0 0 ...
##  $ LowQualFinSF : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ GrLivArea    : int  896 1329 1629 1604 1280 1655 1187 1465 1341 882 ...
##  $ BsmtFullBath : int  0 0 0 0 0 0 1 0 1 1 ...
##  $ BsmtHalfBath : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ FullBath     : int  1 1 2 2 2 2 2 2 1 1 ...
##  $ HalfBath     : int  0 1 1 1 0 1 0 1 1 0 ...
##  $ BedroomAbvGr : int  2 3 3 3 2 3 3 3 2 2 ...
##  $ KitchenAbvGr : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ KitchenQual  : chr  "TA" "Gd" "TA" "Gd" ...
##  $ TotRmsAbvGrd : int  5 6 6 7 5 7 6 7 5 4 ...
##  $ Functional   : chr  "Typ" "Typ" "Typ" "Typ" ...
##  $ Fireplaces   : int  0 0 1 1 0 1 0 1 1 0 ...
##  $ FireplaceQu  : chr  NA NA "TA" "Gd" ...
##  $ GarageType   : chr  "Attchd" "Attchd" "Attchd" "Attchd" ...
##  $ GarageYrBlt  : int  1961 1958 1997 1998 1992 1993 1992 1998 1990 1970 ...
##  $ GarageFinish : chr  "Unf" "Unf" "Fin" "Fin" ...
##  $ GarageCars   : int  1 1 2 2 2 2 2 2 2 2 ...
##  $ GarageArea   : int  730 312 482 470 506 440 420 393 506 525 ...
##  $ GarageQual   : chr  "TA" "TA" "TA" "TA" ...
##  $ GarageCond   : chr  "TA" "TA" "TA" "TA" ...
##  $ PavedDrive   : chr  "Y" "Y" "Y" "Y" ...
##  $ WoodDeckSF   : int  140 393 212 360 0 157 483 0 192 240 ...
##  $ OpenPorchSF  : int  0 36 34 36 82 84 21 75 0 0 ...
##  $ EnclosedPorch: int  0 0 0 0 0 0 0 0 0 0 ...
##  $ X3SsnPorch   : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ ScreenPorch  : int  120 0 0 0 144 0 0 0 0 0 ...
##  $ PoolArea     : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ PoolQC       : chr  NA NA NA NA ...
##  $ Fence        : chr  "MnPrv" NA "MnPrv" NA ...
##  $ MiscFeature  : chr  NA "Gar2" NA NA ...
##  $ MiscVal      : int  0 12500 0 0 0 0 500 0 0 0 ...
##  $ MoSold       : int  6 6 3 6 1 4 3 5 2 4 ...
##  $ YrSold       : int  2010 2010 2010 2010 2010 2010 2010 2010 2010 2010 ...
##  $ SaleType     : chr  "WD" "WD" "WD" "WD" ...
##  $ SaleCondition: chr  "Normal" "Normal" "Normal" "Normal" ...
```

In order to clean the data we will merge the data-sets and clean them both at once.

```r
# join datasets for data cleaning
test_data$SalePrice <- 0
dataset <- rbind(training_data, test_data)
```

3. Methods and Analysis I: Data cleaning

The datasets from Kaggle have 34 columns with missing values which need to be cleaned.

```
# data set is filled with missing values - this needs to be addressed
na.cols <- which(colSums(is.na(dataset)) > 0)
sort(colSums(sapply(dataset[na.cols], is.na)), decreasing = TRUE)
```

```
##       PoolQC  MiscFeature        Alley        Fence  FireplaceQu  LotFrontage
##         2909         2814         2721         2348         1420          486
##  GarageYrBlt GarageFinish   GarageQual   GarageCond   GarageType     BsmtCond
##          159          159          159          159          157           82
## BsmtExposure     BsmtQual BsmtFinType2 BsmtFinType1   MasVnrType   MasVnrArea
##           82           81           80           79           24           23
##     MSZoning    Utilities BsmtFullBath BsmtHalfBath   Functional  Exterior1st
##            4            2            2            2            2            1
##  Exterior2nd   BsmtFinSF1   BsmtFinSF2    BsmtUnfSF  TotalBsmtSF   Electrical
##            1            1            1            1            1            1
##  KitchenQual   GarageCars   GarageArea     SaleType
##            1            1            1            1
```

```
paste('There are', length(na.cols), 'columns with missing values')
```

```
## [1] "There are 34 columns with missing values"
```

First, we will deal with missing values in numerical variables.

```
# dealing with numerical variable - assume that 'NAs' in these variables means 0.
# e.g. LotFrontage : NA most likely means no lot frontage
dataset$LotFrontage[is.na(dataset$LotFrontage)] <- 0
dataset$MasVnrArea[is.na(dataset$MasVnrArea)] <- 0
dataset$BsmtFinSF1[is.na(dataset$BsmtFinSF1)] <- 0
dataset$BsmtFinSF2[is.na(dataset$BsmtFinSF2)] <- 0
dataset$BsmtUnfSF[is.na(dataset$BsmtUnfSF)] <- 0
dataset$TotalBsmtSF[is.na(dataset$TotalBsmtSF)] <- 0
dataset$BsmtFullBath[is.na(dataset$BsmtFullBath)] <- 0
dataset$BsmtHalfBath[is.na(dataset$BsmtHalfBath)] <- 0
dataset$GarageCars[is.na(dataset$GarageCars)] <- 0
dataset$GarageArea[is.na(dataset$GarageArea)] <- 0
```

There is also a mistake in the data-set.

```
# for the variable "GarageYrBlt". We can assume that the year
# the garage was built is the same when the house itself was built.
dataset$GarageYrBlt[is.na(dataset$GarageYrBlt)] <- dataset$YearBuilt[is.na(dataset$GarageYrBlt)]
summary(dataset$GarageYrBlt)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    1872    1958    1978    1976    2001    2207
```

```r
# correcting the error in the dataset
dataset$GarageYrBlt[dataset$GarageYrBlt==2207] <- 2007
```

Next, we will deal with missing values in categorical variables.

```r
# dealing with ‘NAs‘ in categorical values.
# we find "real" NAs, then impute them with the most common value for this feature.
dataset$KitchenQual[is.na(dataset$KitchenQual)] <- names(sort(-table(dataset$KitchenQual)))[1]
dataset$MSZoning[is.na(dataset$MSZoning)] <- names(sort(-table(dataset$MSZoning)))[1]
dataset$SaleType[is.na(dataset$SaleType)] <- names(sort(-table(dataset$SaleType)))[1]
dataset$Exterior1st[is.na(dataset$Exterior1st)] <- names(sort(-table(dataset$Exterior1st)))[1]
dataset$Exterior2nd[is.na(dataset$Exterior2nd)] <- names(sort(-table(dataset$Exterior2nd)))[1]
dataset$Functional[is.na(dataset$Functional)] <- names(sort(-table(dataset$Functional)))[1]
```

```r
# for empty values, we just change the ‘NA‘ value to a new value - ’No’,
# for the rest we change NAs to their actual meaning,
# e.g. NA for basement features is "no basement", etc.
dataset$Alley = factor(dataset$Alley, levels=c(levels(dataset$Alley), "No"))
dataset$Alley[is.na(dataset$Alley)] = "No"
dataset$BsmtQual = factor(dataset$BsmtQual, levels=c(levels(dataset$BsmtQual), "No"))
dataset$BsmtQual[is.na(dataset$BsmtQual)] = "No"
dataset$BsmtCond = factor(dataset$BsmtCond, levels=c(levels(dataset$BsmtCond), "No"))
dataset$BsmtCond[is.na(dataset$BsmtCond)] = "No"
dataset$BsmtExposure[is.na(dataset$BsmtExposure)] = "No"
dataset$BsmtFinType1 = factor(dataset$BsmtFinType1, levels=c(levels(dataset$BsmtFinType1), "No"))
dataset$BsmtFinType1[is.na(dataset$BsmtFinType1)] = "No"
dataset$BsmtFinType2 = factor(dataset$BsmtFinType2, levels=c(levels(dataset$BsmtFinType2), "No"))
dataset$BsmtFinType2[is.na(dataset$BsmtFinType2)] = "No"
dataset$Fence = factor(dataset$Fence, levels=c(levels(dataset$Fence), "No"))
dataset$Fence[is.na(dataset$Fence)] = "No"
dataset$FireplaceQu = factor(dataset$FireplaceQu, levels=c(levels(dataset$FireplaceQu), "No"))
dataset$FireplaceQu[is.na(dataset$FireplaceQu)] = "No"
dataset$GarageType = factor(dataset$GarageType, levels=c(levels(dataset$GarageType), "No"))
dataset$GarageType[is.na(dataset$GarageType)] = "No"
dataset$GarageFinish = factor(dataset$GarageFinish, levels=c(levels(dataset$GarageFinish), "No"))
dataset$GarageFinish[is.na(dataset$GarageFinish)] = "No"
dataset$GarageQual = factor(dataset$GarageQual, levels=c(levels(dataset$GarageQual), "No"))
dataset$GarageQual[is.na(dataset$GarageQual)] = "No"
dataset$GarageCond = factor(dataset$GarageCond, levels=c(levels(dataset$GarageCond), "No"))
dataset$GarageCond[is.na(dataset$GarageCond)] = "No"
dataset$MasVnrType = factor(dataset$MasVnrType, levels=c(levels(dataset$MasVnrType), "No"))
dataset$MasVnrType[is.na(dataset$MasVnrType)] = "No"
dataset$MiscFeature = factor(dataset$MiscFeature, levels=c(levels(dataset$MiscFeature), "No"))
dataset$MiscFeature[is.na(dataset$MiscFeature)] = "No"
dataset$PoolQC = factor(dataset$PoolQC, levels=c(levels(dataset$PoolQC), "No"))
dataset$PoolQC[is.na(dataset$PoolQC)] = "No"
dataset$Electrical = factor(dataset$Electrical, levels=c(levels(dataset$Electrical), "UNK"))
dataset$Electrical[is.na(dataset$Electrical)] = "UNK"
```

Some features can be removed because they don’t have informative purpose.

```r
# remove some unnecessary features
dataset$Utilities <- NULL
dataset$Id <- NULL
```

Final check of the data-set regarding NA/missing values:

```r
# now check again if we have null values.
na.cols <- which(colSums(is.na(dataset)) > 0)
paste('There are now', length(na.cols), 'columns with missing values')
```

```
## [1] "There are now 0 columns with missing values"
```

As it can be seen, the data-set is now clean and has no missing values.

4. Methods and Analysis II: Data engineering

In this section we will do some data engineering and remodeling in order to enhance variables we have in the data-set from Kaggle for our machine learning purposes. As stated in the data-description: respective data-set from Kaggle has 79 explanatory variables describing (almost) every aspect of residential homes in Ames, Iowa, USA. However, in order to enhance our data we will first recode some descriptive variables in order to get ordinal data on numerical score-lists; secondly, we will create some new features/variables from existing variables; and thirdly, we will deal with skewness of the target value - SalePrice - by applying log transformation.

- First, we will recode some descriptive variables in order to get ordinal data on numerical score-lists which will be more useful for later purposes. For example, we will recode variables ExterQual, HeatingQC and KitchenQual (quality of exterior material, heating quality and quality of kitchen in a house) from "None", "Poor", "Fair", "TA=Typical", "Good" and "Excellent" into 0 for none, 1 for poor, 2 for fair, 3 for typical 4 for good and 6 for excellent quality.

```r
# Recoding descriptive variables into ordinal
dataset$ExterQual<- recode(dataset$ExterQual,"None"=0,"Po"=1,"Fa"=2,"TA"=3,"Gd"=4,"Ex"=6)
dataset$ExterCond<- recode(dataset$ExterCond,"None"=0,"Po"=1,"Fa"=2,"TA"=3,"Gd"=4,"Ex"=6)
dataset$BsmtQual<- recode(dataset$BsmtQual,"No"=0,"Po"=1,"Fa"=2,"TA"=3,"Gd"=4,"Ex"=6)
dataset$BsmtCond<- recode(dataset$BsmtCond,"No"=0,"Po"=1,"Fa"=2,"TA"=3,"Gd"=4,"Ex"=6)
dataset$BsmtExposure<- recode(dataset$BsmtExposure,"No"=0,"No"=1,"Mn"=2,"Av"=3,"Gd"=6)
dataset$BsmtFinType1<- recode(dataset$BsmtFinType1,"No"=0,"Unf"=1,"LwQ"=2,"Rec"=3,"BLQ"=4,
                              "ALQ"=5,"GLQ"=6)
dataset$BsmtFinType2<- recode(dataset$BsmtFinType2,"No"=0,"Unf"=1,"LwQ"=2,"Rec"=3,"BLQ"=4,
                              "ALQ"=5,"GLQ"=6)
dataset$HeatingQC<- recode(dataset$HeatingQC,"None"=0,"Po"=1,"Fa"=2,"TA"=3,"Gd"=4,"Ex"=6)
dataset$KitchenQual<- recode(dataset$KitchenQual,"None"=0,"Po"=1,"Fa"=2,"TA"=3,"Gd"=4,"Ex"=6)
dataset$Functional<- recode(dataset$Functional,"None"=0,"Sev"=1,"Maj2"=2,"Maj1"=3,"Mod"=4,
                            "Min2"=5,"Min1"=6,"Typ"=7)
dataset$FireplaceQu<- recode(dataset$FireplaceQu,"No"=0,"Po"=1,"Fa"=2,"TA"=3,"Gd"=4,"Ex"=6)
dataset$GarageFinish<- recode(dataset$GarageFinish,"No"=0,"Unf"=1,"RFn"=2,"Fin"=3)
dataset$GarageQual<- recode(dataset$GarageQual,"No"=0,"Po"=1,"Fa"=2,"TA"=3,"Gd"=4,"Ex"=6)
dataset$GarageCond<- recode(dataset$GarageCond,"No"=0,"Po"=1,"Fa"=2,"TA"=3,"Gd"=4,"Ex"=6)
dataset$PoolQC<- recode(dataset$PoolQC,"No"=0,"Po"=1,"Fa"=2,"TA"=3,"Gd"=4,"Ex"=6)
dataset$Fence<- recode(dataset$Fence,"No"=0,"MnWw"=1,"GdWo"=2,"MnPrv"=3,"GdPrv"=6)
```

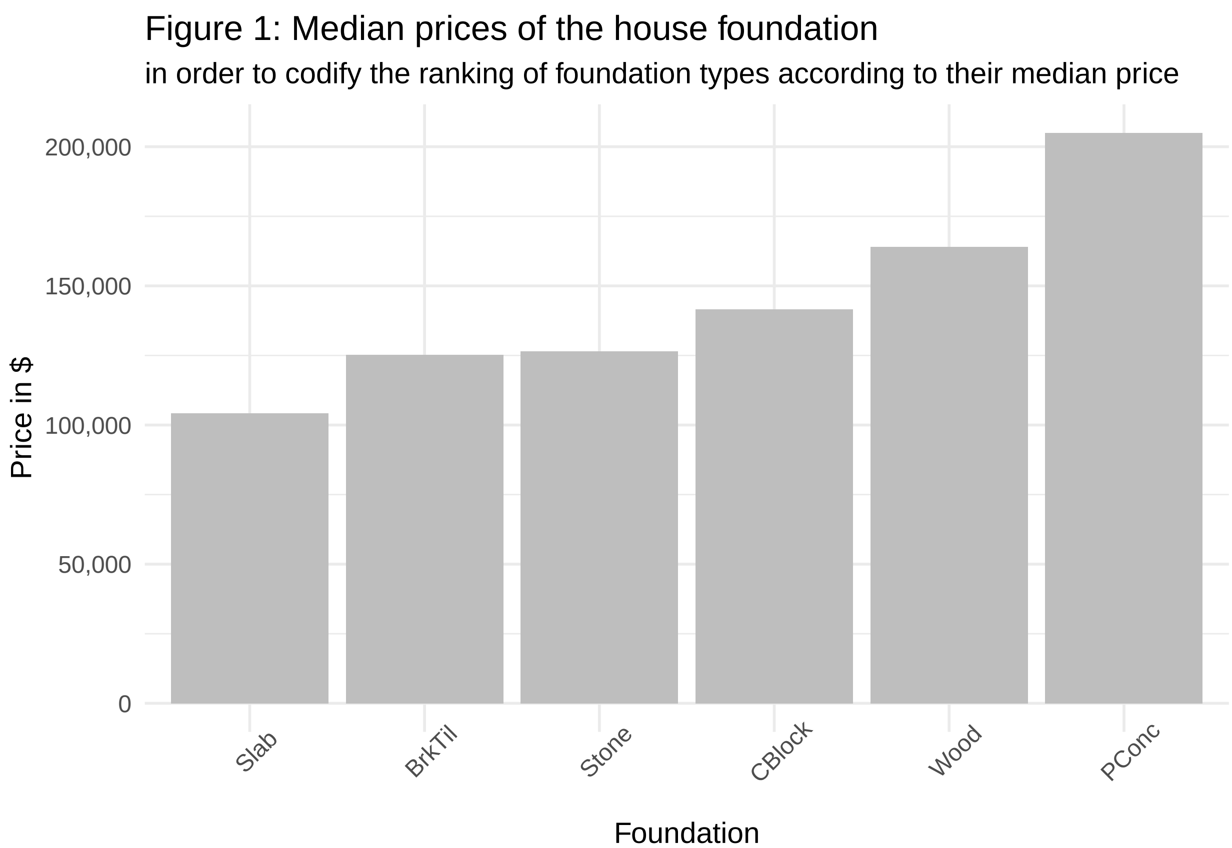
- Next, we will add three new features or variables into the existing data-set.

First newly created variable is "TotalInsideSF" which will show the total inside surface of the house by combining the total surface of the 1. and 2. floor square feet.

```
# i) new feature - area/size
# Total surface of the house, combining the total inside surface (1 and 2 floor square feet)
dataset['TotalInsideSF'] <- as.numeric(dataset$X1stFlrSF + dataset$X2ndFlrSF)
```

Second newly created variable is "FoundationScore" which will measure the quality of the foundation material of a house according to their median house value on an scale from 1 (worst foundation material) to 6 (best foundation material).

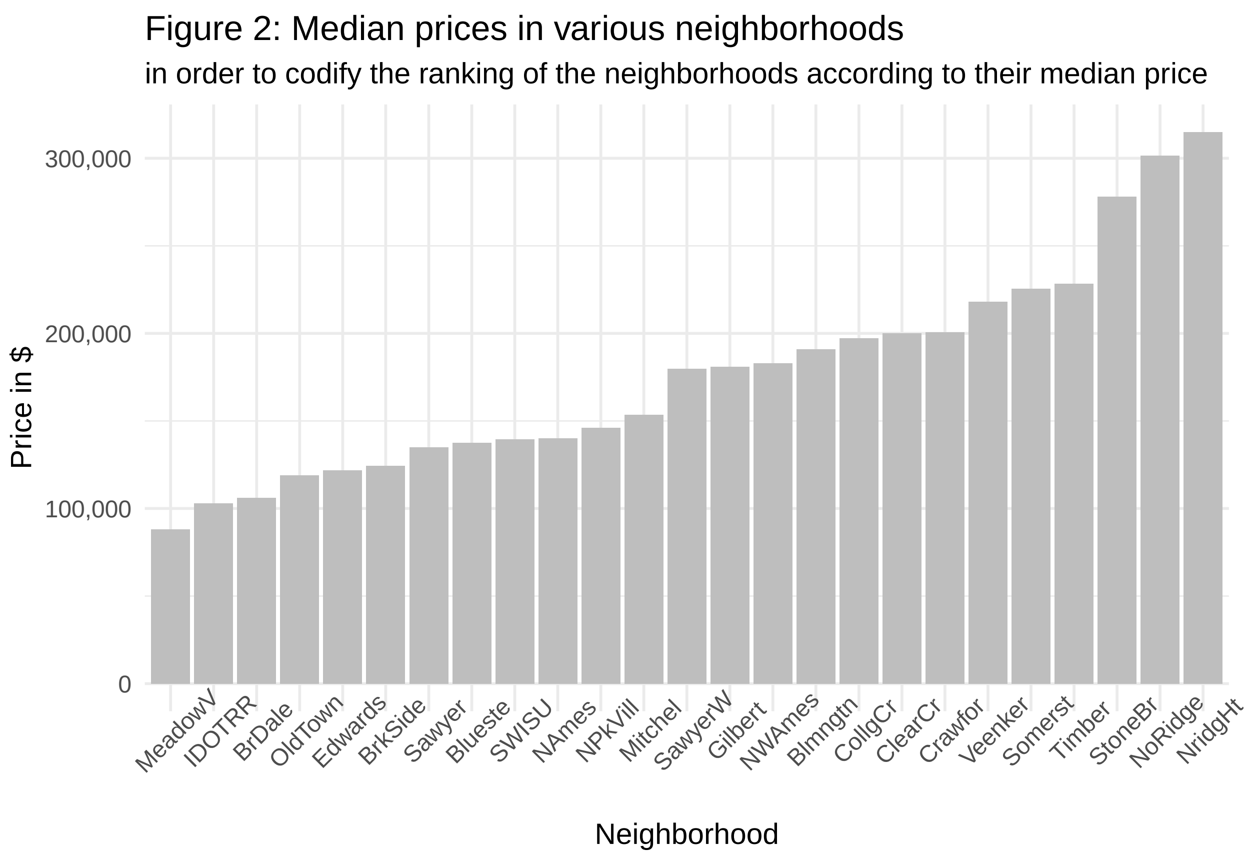
```
# ii) new feature - quality of the foundation material of a house
#     codify the ranking of the foundation material according to the median house value.
training_data[,c('Foundation','SalePrice')] %>%
  group_by(Foundation) %>%
  summarise(avg = median(SalePrice, na.rm = TRUE)) %>%
  arrange(avg) %>%
  mutate(sorted = factor(Foundation, levels=Foundation)) %>%
  ggplot(aes(x=sorted, y=avg)) +
  geom_bar(stat = "identity", fill="grey") +
  scale_y_continuous(labels = scales::comma)+
  labs(x='Foundation', y='Price in $') +
  theme_minimal()+
  theme(axis.text.x = element_text(angle=45))+
  labs(title ="Figure 1: Median prices of the house foundation",
       subtitle="in order to codify the ranking of foundation types according to their median price")
```




```
dataset$FoundationScore <- recode(dataset$Foundation, 'Slab' = 1, 'BrkTil' = 2,
                                  'Stone' = 2, 'CBlock' = 3, 'Wood' = 4, 'PConc' = 6)
```

Third new feature deals with the location/neighborhood of a house.It is common wisdom that the location of a house is one of the most important predictor of its price. Therefore, the new variable "NeighborhoodScored" measures the "quality" of location of a house according to their median price value on scale from 1(worst location) - 7 (best location).

```
training_data[,c('Neighborhood','SalePrice')] %>%
  group_by(Neighborhood) %>%
  summarise(avg = median(SalePrice, na.rm = TRUE)) %>%
  arrange(avg) %>%
  mutate(sorted = factor(Neighborhood, levels=Neighborhood)) %>%
  ggplot(aes(x=sorted, y=avg)) +
  geom_bar(stat = "identity", fill="grey") +
  scale_y_continuous(labels = scales::comma)+
  labs(x='Neighborhood', y='Price in $') +
  theme_minimal()+
  theme(axis.text.x = element_text(angle=45))+
  labs(title ="Figure 2: Median prices in various neighborhoods",
       subtitle="in order to codify the ranking of the neighborhoods according to their median price")
```

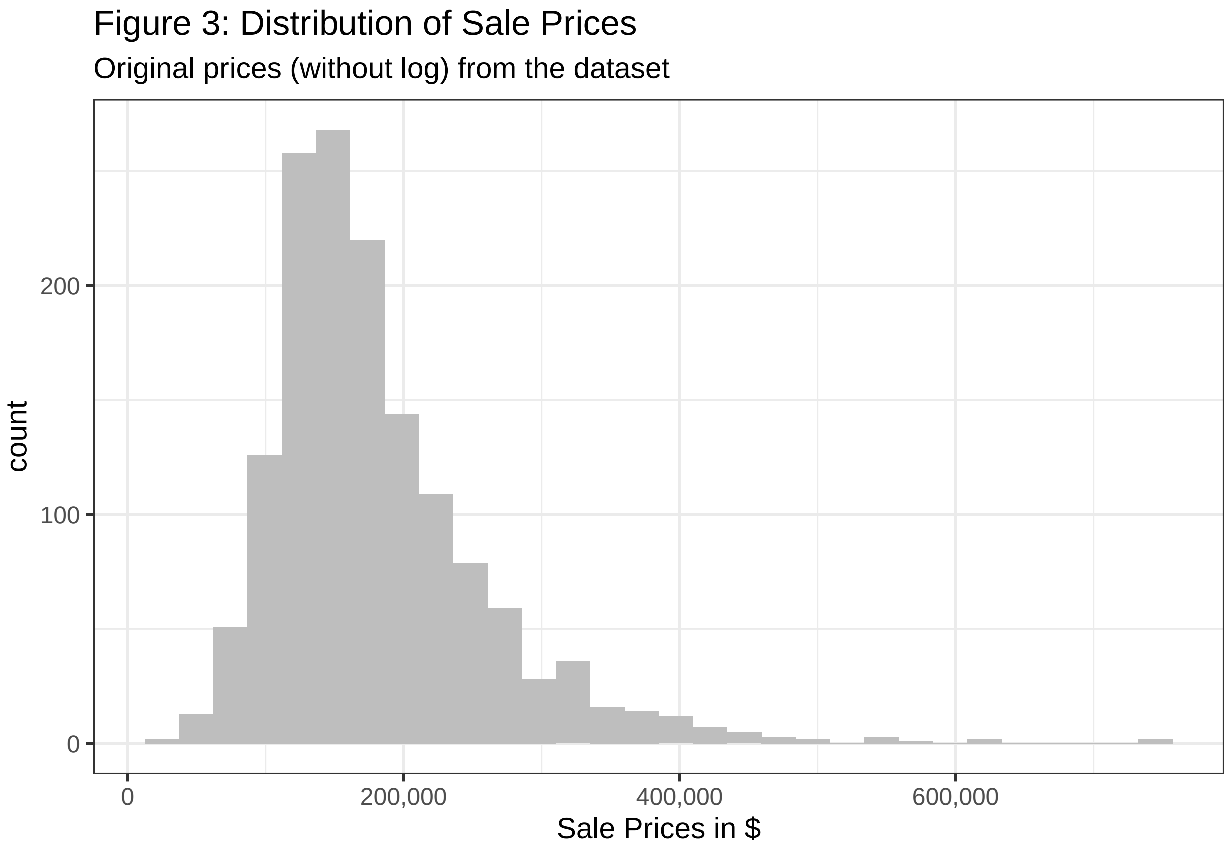



```
dataset$NeighborhoodScored <- recode(dataset$Neighborhood, 'MeadowV' = 1, 'IDOTRR' = 2,
                                     'BrDale' = 2, 'OldTown' = 3, 'Edwards' = 3,
                                     'BrkSide' = 3,'Sawyer' = 4, 'Blueste' = 4,
                                     'SWISU' = 4, 'NAmes' = 4, 'NPkVill' = 4,
                                     'Mitchel' = 4,'SawyerW' = 5, 'Gilbert' = 5,
                                     'NWAmes' =5, 'Blmngtn' = 5, 'CollgCr' = 5,
                                     'ClearCr' = 5, 'Crawfor' = 5, 'Veenker' = 6,
                                     'Somerst' = 6, 'Timber' = 6, 'StoneBr' = 7,
                                     'NoRidge' = 7, 'NridgHt' =7)
```

- Next, we want to analyse our target variable that we want to predict, namely, SalePrice (selling price of a house).

On the Figure 3 below we can see the distribution of this variable.

```
ggplot(training_data, aes(SalePrice)) +
  geom_histogram(fill="grey") +
  scale_x_continuous(labels = scales::comma)+
  labs(title="Figure 3: Distribution of Sale Prices",
       subtitle = "Original prices (without log) from the dataset",
       x="Sale Prices in $")+
  theme_bw()
```



The distribution is right or positive skewed. Skewness of the data refers to its imbalance and asymmetry from the mean of a data distribution. Positive skew means that the extreme data results are larger which brings the mean (average) up. This also means that the mean will be larger than the median. The distribution shows that most of the houses are sold under 200,000. This is confirmed if we statistically summarize Sale Price variable:

```
summary(training_data$SalePrice) # original $ prices
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   34900  129975  163000  180921  214000  755000
```

Mean price is around 180.000 dollars and median is 163.000 dollars. Therefore, in order to deals with this skewness we will log transform the target value (SalePrice).

```
# Dealing with Skewness - Transform the target value - SalePrice - applying log
dataset$SalePrice <- log(dataset$SalePrice)
```

At the end, we will factorize some features

```
# Factorize features
dataset$MSSubClass <- as.factor(dataset$MSSubClass)
dataset$MoSold <- as.factor(dataset$MoSold)
dataset$YrSold <- as.factor(dataset$YrSold)
```

5. Methods and Analysis III: Data visualisations

As stated at the beginning: if we check the test_data from Kaggle we can see it has no SalePrice variable that we are trying to predict. Therefore, we will need to focus on the training part of the data-set (i.e. now cleaned dataset1[1:1460,]) for training, testing and validating our ML models. But first, we will check the statistical summaries of cleaned data-set which will be the basis for further work.

```r
train <- dataset[1:1460,]
summary(train) # check the statistical summaries of the cleaned data-set
```

```
##    MSSubClass    MSZoning          LotFrontage       LotArea
##  20     :536   Length:1460        Min.   :  0.00   Min.   :  1300
##  60     :299   Class :character   1st Qu.: 42.00   1st Qu.:  7554
##  50     :144   Mode  :character   Median : 63.00   Median :  9478
##  120    : 87                      Mean   : 57.62   Mean   : 10517
##  30     : 69                      3rd Qu.: 79.00   3rd Qu.: 11602
##  160    : 63                      Max.   :313.00   Max.   :215245
##  (Other):262
##     Street             Alley        LotShape          LandContour
##  Length:1460        No:1460   Length:1460        Length:1460
##  Class :character             Class :character   Class :character
##  Mode  :character             Mode  :character   Mode  :character
##
##
##
##
##   LotConfig          LandSlope         Neighborhood        Condition1
##  Length:1460        Length:1460        Length:1460        Length:1460
##  Class :character   Class :character   Class :character   Class :character
##  Mode  :character   Mode  :character   Mode  :character   Mode  :character
##
##
##
##
##   Condition2          BldgType          HouseStyle          OverallQual
##  Length:1460        Length:1460        Length:1460        Min.   : 1.000
##  Class :character   Class :character   Class :character   1st Qu.: 5.000
##  Mode  :character   Mode  :character   Mode  :character   Median : 6.000
##                                                           Mean   : 6.099
##                                                           3rd Qu.: 7.000
##                                                           Max.   :10.000
##
##   OverallCond      YearBuilt     YearRemodAdd    RoofStyle
##  Min.   :1.000   Min.   :1872   Min.   :1950   Length:1460
##  1st Qu.:5.000   1st Qu.:1954   1st Qu.:1967   Class :character
##  Median :5.000   Median :1973   Median :1994   Mode  :character
##  Mean   :5.575   Mean   :1971   Mean   :1985
##  3rd Qu.:6.000   3rd Qu.:2000   3rd Qu.:2004
##  Max.   :9.000   Max.   :2010   Max.   :2010
##
##    RoofMatl          Exterior1st         Exterior2nd         MasVnrType
##  Length:1460        Length:1460        Length:1460        No:1460
##  Class :character   Class :character   Class :character
##  Mode  :character   Mode  :character   Mode  :character
```

```
##
##
##
##
##    MasVnrArea       ExterQual       ExterCond      Foundation
##  Min.   :   0.0   Min.   :2.000   Min.   :1.000   Length:1460
##  1st Qu.:   0.0   1st Qu.:3.000   1st Qu.:3.000   Class :character
##  Median :   0.0   Median :3.000   Median :3.000   Mode  :character
##  Mean   : 103.1   Mean   :3.432   Mean   :3.086
##  3rd Qu.: 164.2   3rd Qu.:4.000   3rd Qu.:3.000
##  Max.   :1600.0   Max.   :6.000   Max.   :6.000
##
##     BsmtQual    BsmtCond  BsmtExposure    BsmtFinType1   BsmtFinSF1
##  Min.   :0   Min.   :0   Min.   :0.000   Min.   :0     Min.   :   0.0
##  1st Qu.:0   1st Qu.:0   1st Qu.:0.000   1st Qu.:0     1st Qu.:   0.0
##  Median :0   Median :0   Median :0.000   Median :0     Median : 383.5
##  Mean   :0   Mean   :0   Mean   :1.161   Mean   :0     Mean   : 443.6
##  3rd Qu.:0   3rd Qu.:0   3rd Qu.:2.000   3rd Qu.:0     3rd Qu.: 712.2
##  Max.   :0   Max.   :0   Max.   :6.000   Max.   :0     Max.   :5644.0
##
##   BsmtFinType2   BsmtFinSF2        BsmtUnfSF       TotalBsmtSF
##  Min.   :0     Min.   :   0.00   Min.   :   0.0   Min.   :   0.0
##  1st Qu.:0     1st Qu.:   0.00   1st Qu.: 223.0   1st Qu.: 795.8
##  Median :0     Median :   0.00   Median : 477.5   Median : 991.5
##  Mean   :0     Mean   :  46.55   Mean   : 567.2   Mean   :1057.4
##  3rd Qu.:0     3rd Qu.:   0.00   3rd Qu.: 808.0   3rd Qu.:1298.2
##  Max.   :0     Max.   :1474.00   Max.   :2336.0   Max.   :6110.0
##
##    Heating            HeatingQC      CentralAir        Electrical
##  Length:1460        Min.   :1.000   Length:1460        UNK:1460
##  Class :character   1st Qu.:3.000   Class :character
##  Mode  :character   Median :6.000   Mode  :character
##                     Mean   :4.653
##                     3rd Qu.:6.000
##                     Max.   :6.000
##
##    X1stFlrSF      X2ndFlrSF     LowQualFinSF       GrLivArea
##  Min.   : 334   Min.   :   0   Min.   :  0.000   Min.   : 334
##  1st Qu.: 882   1st Qu.:   0   1st Qu.:  0.000   1st Qu.:1130
##  Median :1087   Median :   0   Median :  0.000   Median :1464
##  Mean   :1163   Mean   : 347   Mean   :  5.845   Mean   :1515
##  3rd Qu.:1391   3rd Qu.: 728   3rd Qu.:  0.000   3rd Qu.:1777
##  Max.   :4692   Max.   :2065   Max.   :572.000   Max.   :5642
##
##   BsmtFullBath     BsmtHalfBath       FullBath        HalfBath
##  Min.   :0.0000   Min.   :0.00000   Min.   :0.000   Min.   :0.0000
##  1st Qu.:0.0000   1st Qu.:0.00000   1st Qu.:1.000   1st Qu.:0.0000
##  Median :0.0000   Median :0.00000   Median :2.000   Median :0.0000
##  Mean   :0.4253   Mean   :0.05753   Mean   :1.565   Mean   :0.3829
##  3rd Qu.:1.0000   3rd Qu.:0.00000   3rd Qu.:2.000   3rd Qu.:1.0000
##  Max.   :3.0000   Max.   :2.00000   Max.   :3.000   Max.   :2.0000
##
##   BedroomAbvGr    KitchenAbvGr    KitchenQual    TotRmsAbvGrd
##  Min.   :0.000   Min.   :0.000   Min.   :2.00   Min.   : 2.000
```

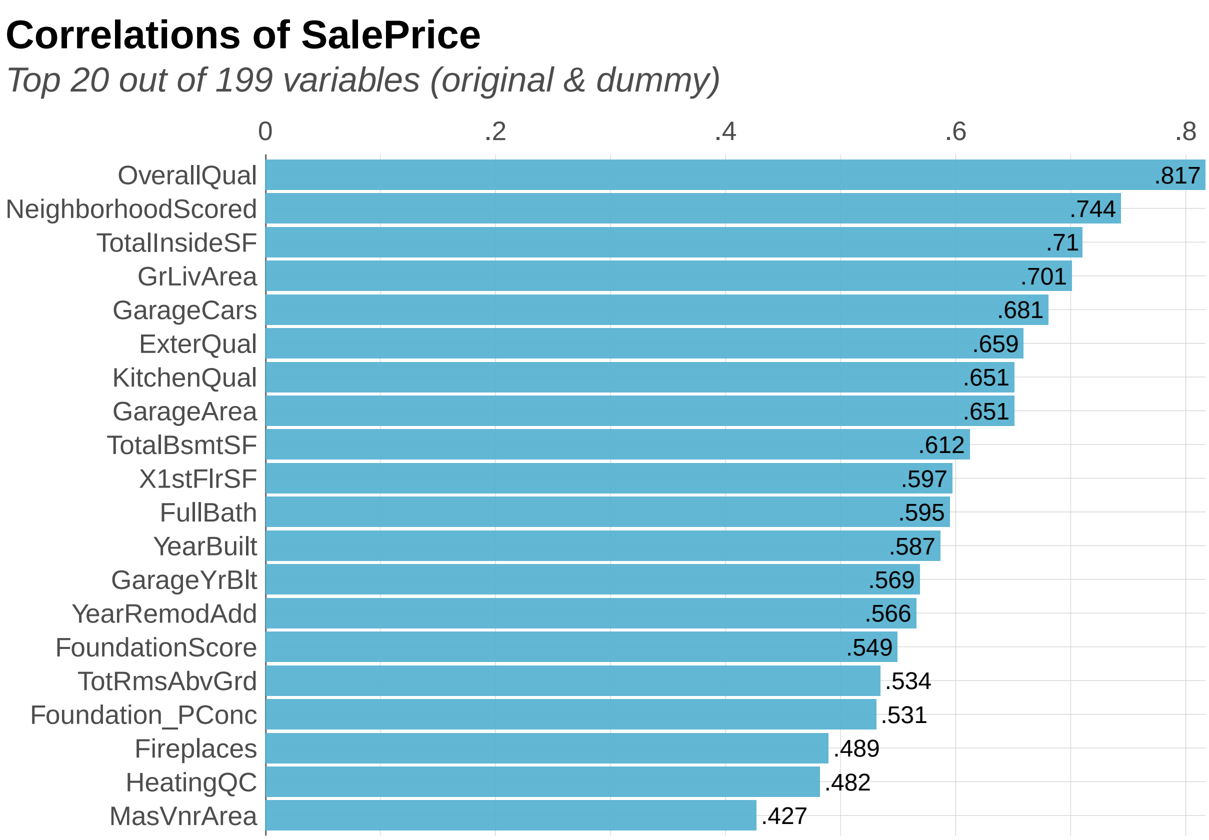
```
##  1st Qu.:2.000   1st Qu.:1.000   1st Qu.:3.00   1st Qu.: 5.000
##  Median :3.000   Median :1.000   Median :3.00   Median : 6.000
##  Mean   :2.866   Mean   :1.047   Mean   :3.58   Mean   : 6.518
##  3rd Qu.:3.000   3rd Qu.:1.000   3rd Qu.:4.00   3rd Qu.: 7.000
##  Max.   :8.000   Max.   :3.000   Max.   :6.00   Max.   :14.000
##
##    Functional      Fireplaces     FireplaceQu GarageType  GarageYrBlt
##  Min.   :1.000   Min.   :0.000   Min.   :0    No:1460    Min.   :1872
##  1st Qu.:7.000   1st Qu.:0.000   1st Qu.:0               1st Qu.:1959
##  Median :7.000   Median :1.000   Median :0               Median :1978
##  Mean   :6.842   Mean   :0.613   Mean   :0               Mean   :1977
##  3rd Qu.:7.000   3rd Qu.:1.000   3rd Qu.:0               3rd Qu.:2001
##  Max.   :7.000   Max.   :3.000   Max.   :0               Max.   :2010
##
##   GarageFinish   GarageCars      GarageArea       GarageQual   GarageCond
##  Min.   :0     Min.   :0.000   Min.   :   0.0   Min.   :0    Min.   :0
##  1st Qu.:0     1st Qu.:1.000   1st Qu.: 334.5   1st Qu.:0    1st Qu.:0
##  Median :0     Median :2.000   Median : 480.0   Median :0    Median :0
##  Mean   :0     Mean   :1.767   Mean   : 473.0   Mean   :0    Mean   :0
##  3rd Qu.:0     3rd Qu.:2.000   3rd Qu.: 576.0   3rd Qu.:0    3rd Qu.:0
##  Max.   :0     Max.   :4.000   Max.   :1418.0   Max.   :0    Max.   :0
##
##   PavedDrive          WoodDeckSF      OpenPorchSF     EnclosedPorch
##  Length:1460        Min.   :  0.00   Min.   :  0.00   Min.   :  0.00
##  Class :character   1st Qu.:  0.00   1st Qu.:  0.00   1st Qu.:  0.00
##  Mode  :character   Median :  0.00   Median : 25.00   Median :  0.00
##                     Mean   : 94.24   Mean   : 46.66   Mean   : 21.95
##                     3rd Qu.:168.00   3rd Qu.: 68.00   3rd Qu.:  0.00
##                     Max.   :857.00   Max.   :547.00   Max.   :552.00
##
##    X3SsnPorch      ScreenPorch        PoolArea          PoolQC      Fence
##  Min.   :  0.00   Min.   :  0.00   Min.   :  0.000   Min.   :0   Min.   :0
##  1st Qu.:  0.00   1st Qu.:  0.00   1st Qu.:  0.000   1st Qu.:0   1st Qu.:0
##  Median :  0.00   Median :  0.00   Median :  0.000   Median :0   Median :0
##  Mean   :  3.41   Mean   : 15.06   Mean   :  2.759   Mean   :0   Mean   :0
##  3rd Qu.:  0.00   3rd Qu.:  0.00   3rd Qu.:  0.000   3rd Qu.:0   3rd Qu.:0
##  Max.   :508.00   Max.   :480.00   Max.   :738.000   Max.   :0   Max.   :0
##
##  MiscFeature    MiscVal             MoSold     YrSold      SaleType
##  No:1460     Min.   :    0.00   6      :253   2006:314   Length:1460
##              1st Qu.:    0.00   7      :234   2007:329   Class :character
##              Median :    0.00   5      :204   2008:304   Mode  :character
##              Mean   :   43.49   4      :141   2009:338
##              3rd Qu.:    0.00   8      :122   2010:175
##              Max.   :15500.00   3      :106
##                                 (Other):400
##  SaleCondition        SalePrice     TotalInsideSF  FoundationScore
##  Length:1460        Min.   :10.46   Min.   : 334   Min.   :1.000
##  Class :character   1st Qu.:11.78   1st Qu.:1124   1st Qu.:3.000
##  Mode  :character   Median :12.00   Median :1458   Median :3.000
##                     Mean   :12.02   Mean   :1510   Mean   :4.195
##                     3rd Qu.:12.27   3rd Qu.:1775   3rd Qu.:6.000
##                     Max.   :13.53   Max.   :5642   Max.   :6.000
##
```

```
##   NeighborhoodScored
##   Min.   :1.000
##   1st Qu.:4.000
##   Median :5.000
##   Mean   :4.499
##   3rd Qu.:5.000
##   Max.   :7.000
##
```

Before we dive into ML models, it is prudent to check the correlations of all variables with our target variable - Sale Price - in order to see which variables could be the most important/correlated for predicting the sale price of houses. In the figure below we can see 20 most correlated (Pearson correlation method) variables with SalePrice:

Figure 4: Correlation Report: 20 most correlated variables with SalePrice variable

```r
corr_var(train,                 # name of the dataset (which is now cleaned and remodeled)
         SalePrice,             # name of the variable to focus on
         method = "pearson",    # name of the correlation approach
         top = 20)              # display top 20 correlations
```



The chart above helps us to see the variables that could be important in predicting the SalePrice variable of houses. We could summarize 20 most correlated variables in the correlation chart above in 4 dimensions:

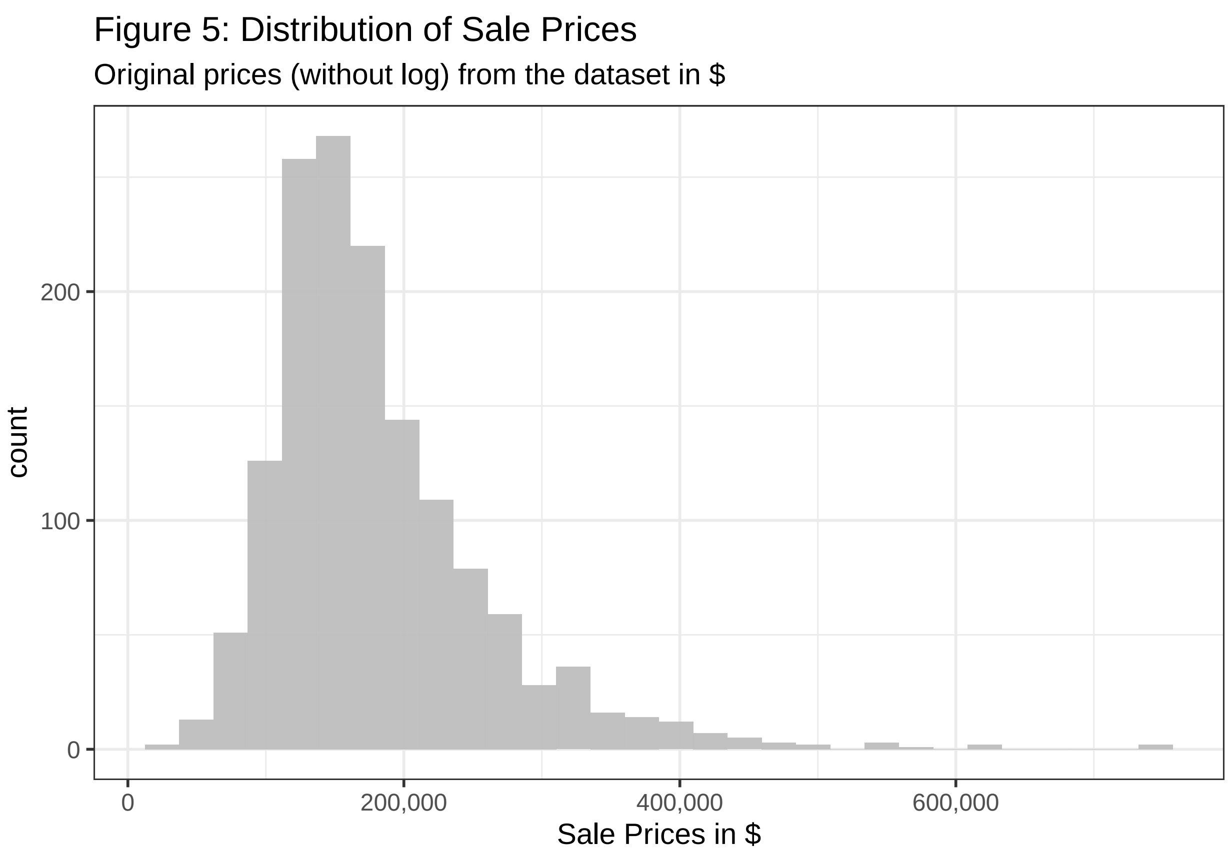
- 1. Dimension - Quality - refers to the quality of a house and includes the following variables from the dataset: OverallQual (Rates the overall material and finish of the house - most correlated variable); KitchenQual (Kitchen quality and condition); FullBath (Full bathrooms above ground);

FoundationScore (Type of foundation - score on the 1-6 score-list); Fireplaces (Number of fireplaces); HeatingQC (Heating quality and condition); ExterQual (Evaluates the quality of the material on the exterior).

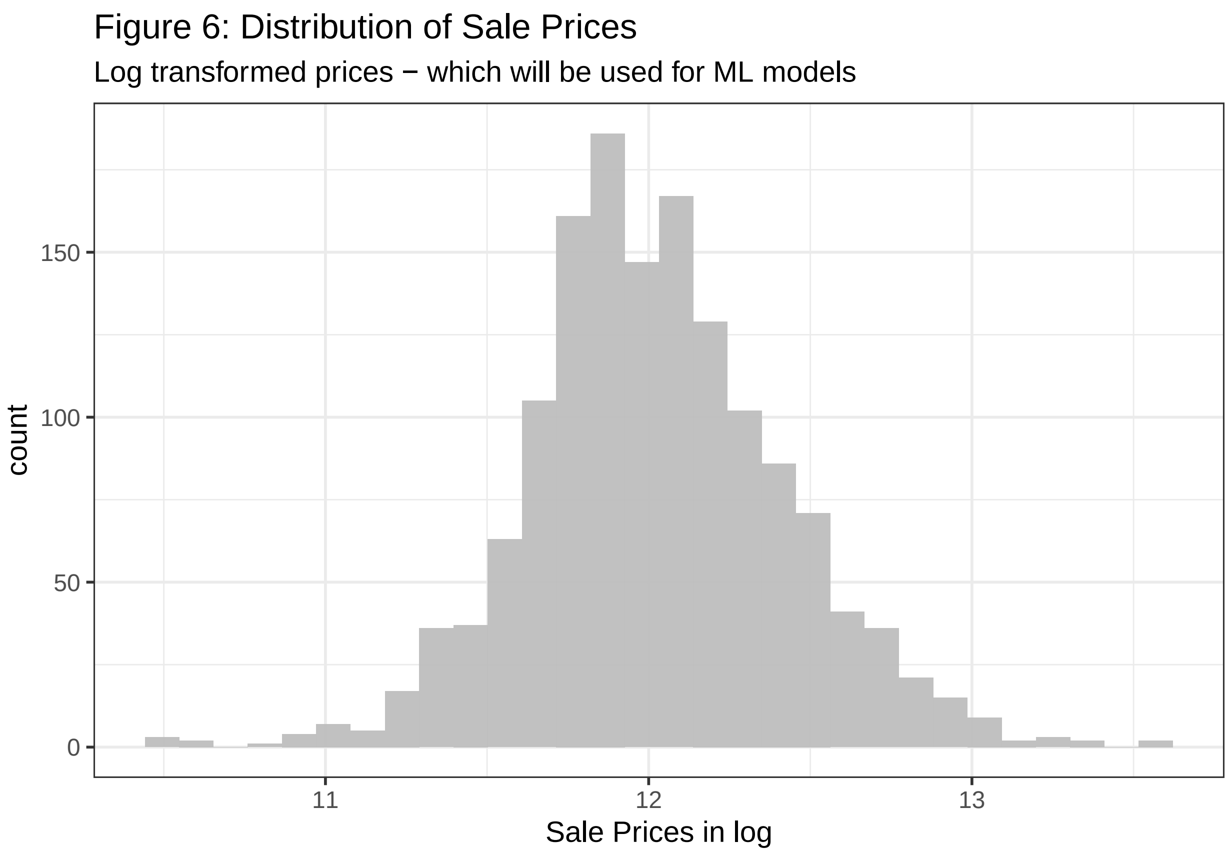
- 2. Dimension - Location - refers to the location of a house and includes the following variable: NeighborhoodScored (Location of a house on the 1-7 score-list).
- 3. Dimension - Size - refers to the size of a house and includes the following variables: TotalInsideSF (Total inside surface of a house, combining 1st and 2nd floor); GarageCars (Size of garage in car capacity); GrLivArea (Above ground living area square feet); GarageArea (Size of garage in square feet); TotalBsmtSF (Total square feet of basement area); TotRmsAbvGrd (Total rooms above grade).
- 4. Dimension - Age - refers to the age of a house and includes the following variables: YearBuilt (Original construction date of a house); YearRemodAdd (Remodel date of a house).

Before visually exploring these correlations in more detail, it would be wise to once again check the target variable - SalePrice - before and after log transformation.

```r
# SalePrice variable - original prices in $
ggplot(training_data, aes(SalePrice)) +
  geom_histogram(fill="grey") +
  scale_x_continuous(labels = scales::comma)+
  labs(title="Figure 5: Distribution of Sale Prices",
       subtitle = "Original prices (without log) from the dataset in $",
       x="Sale Prices in $")+
  theme_bw()
```



```r
# SalePrice variable - log transformed prices
ggplot(train, aes(SalePrice)) +
  geom_histogram(fill="grey") +
  labs(title="Figure 6: Distribution of Sale Prices",
       subtitle="Log transformed prices - which will be used for ML models",
       x="Sale Prices in log")+
  theme_bw()
```



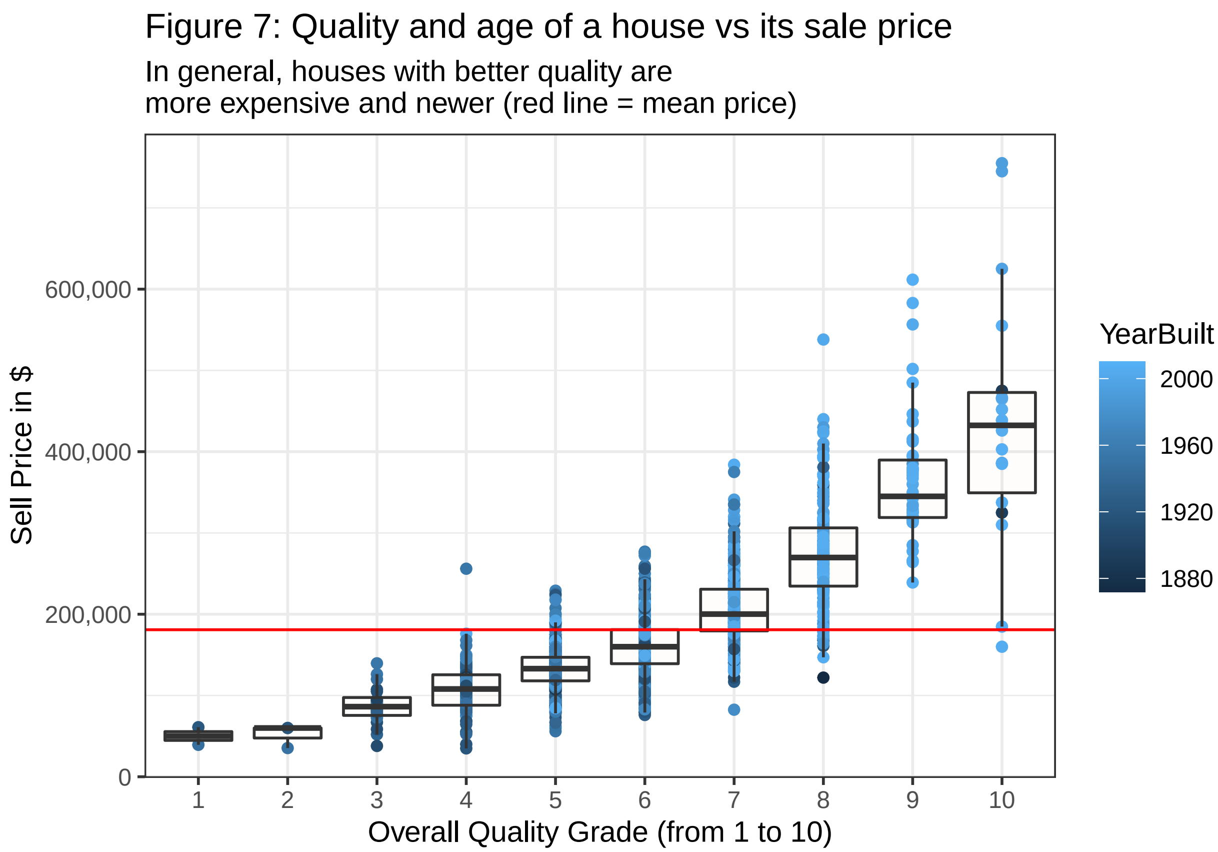
From the graphs above it is visible that the log transformation of our target variable (SalePrice) corrected positive skewness of the original data. This log-transformed SalePrice variable will be used later in our ML models.

We can now graphically visualize the most important aspects of previously defined 4 dimensions (quality, location, size and age) and put them in a relation with SalePrice variable (in both original and log-transformed form). One note: for some visualizations we will use the original Sale Price values in dollars, because it is easier to interpret original values than log transformed ones; although, later for ML models we will use log transformed Sale Price variable because it is closer to the normal distribution.

In the first two figures (Figure 7 and 8) we show relations between quality, location and age of a house vs. its sale price:

```r
training_data %>%
  ggplot(aes(factor(OverallQual), SalePrice))+
  geom_point(alpha = 1, aes(color = YearBuilt))+
  geom_boxplot(alpha =0.01, aes(group=OverallQual))+
  geom_hline (aes(yintercept = mean(SalePrice)), color="red")+
  theme_bw()+
```

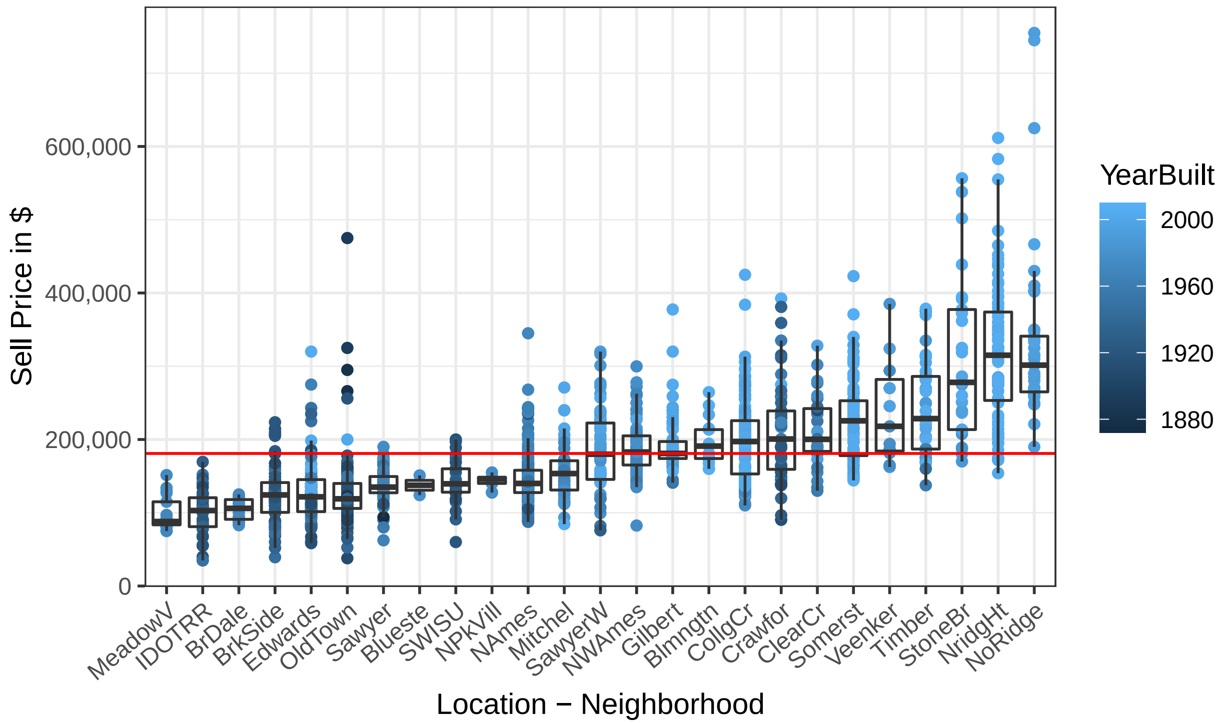
```r
  scale_y_continuous(labels = scales::comma)+
  labs(title = "Figure 7: Quality and age of a house vs its sale price",
       subtitle = "In general, houses with better quality are\nmore expensive and newer (red line = mean
       x= "Overall Quality Grade (from 1 to 10)",
       y= "Sell Price in $")
```




```r
training_data %>%
  ggplot(aes(reorder(Neighborhood, SalePrice), SalePrice)) +
  geom_point(alpha = 1, aes(color = YearBuilt))+
  geom_boxplot(alpha =0.01) +
  geom_hline(aes(yintercept = mean(SalePrice)),color="red") +
  theme_bw()+
  scale_y_continuous(labels = scales::comma)+
  theme(axis.text.x = element_text(angle = 40, hjust = 1)) +
  labs(title = "Figure 8:Location and age of a house vs its sale price",
       subtitle = "In general, newer houses are located in more\nexpansive neighborhoods (red line = mea
       x= "Location - Neighborhood",
       y= "Sell Price in $")
```

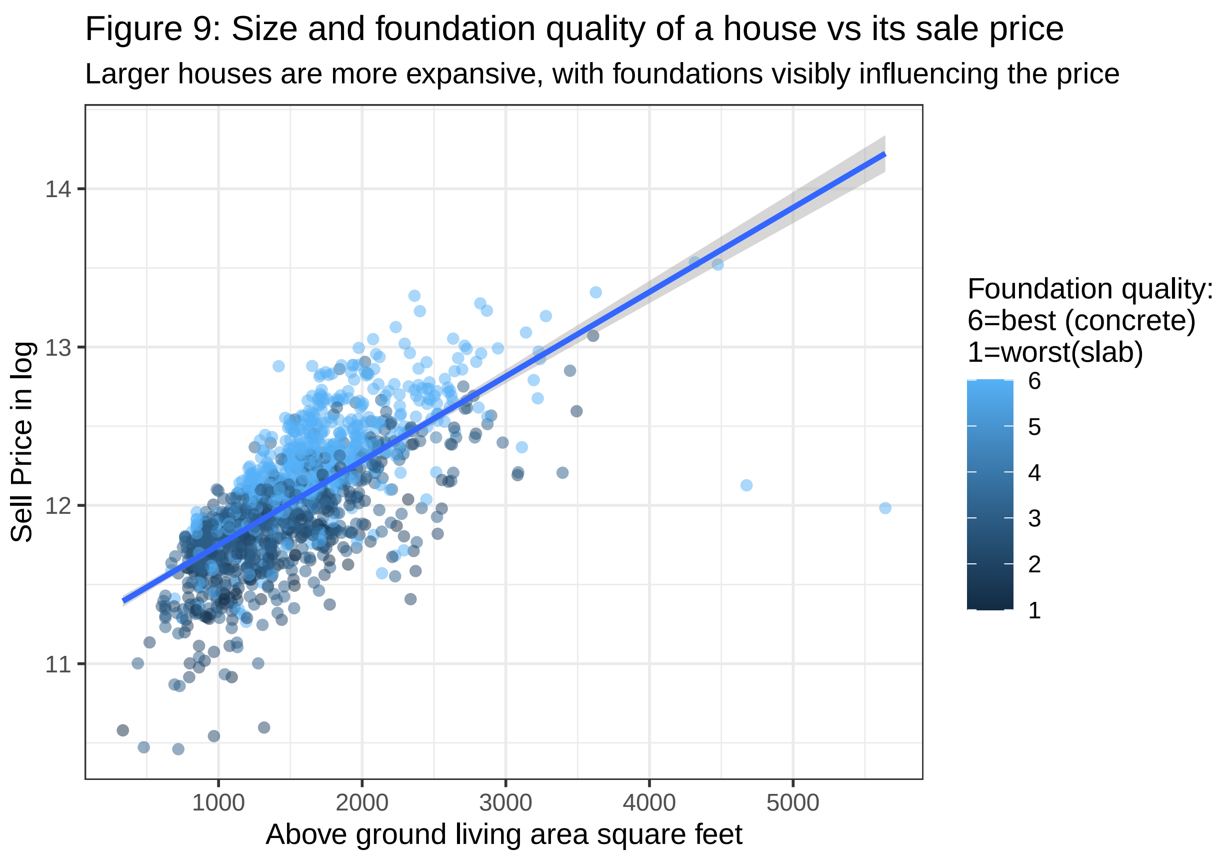
Figure 8:Location and age of a house vs its sale price

In general, newer houses are located in more
expansive neighborhoods (red line = mean price)

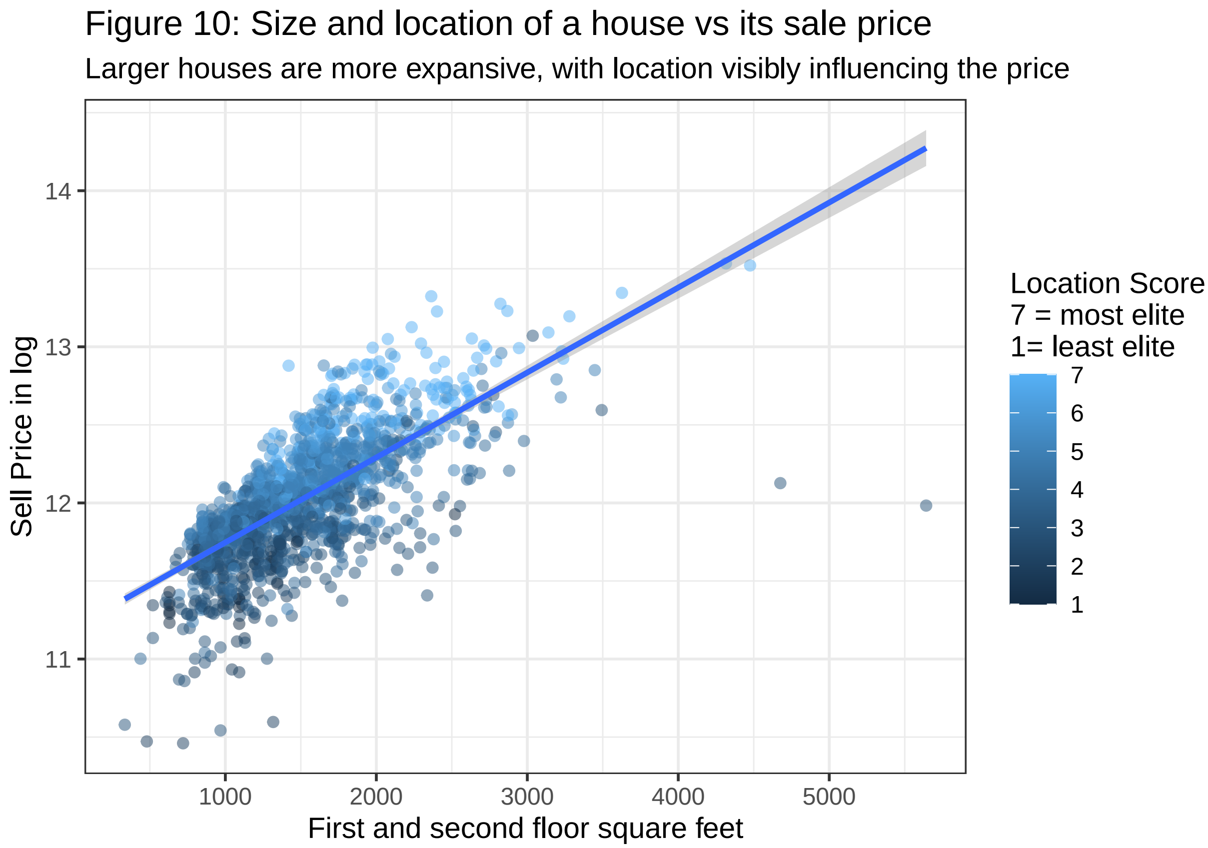


In the next two figures we explore the relations between size, foundation quality and location of a house vs. its sell price. Size of a house is showed with two variables: i) GrLivArea variable (Above ground living area square feet); and ii) with our custom made TotalInsideSF variable (1.+ 2. floor square feet). Foundation quality is showed with our custom made Foundation Score variable (1-6 scale), while location is showed with our custom made Neighborhood Scored variable (1-7 scale). Both figures use log transformed sale prices which will be used for ML models.

```
train %>%
  ggplot(aes(GrLivArea, SalePrice))+
  geom_point(alpha = 0.5, aes(color = FoundationScore))+
  geom_smooth(method = "lm")+
  theme_bw()+
  labs(title = "Figure 9: Size and foundation quality of a house vs its sale price",
       subtitle = "Larger houses are more expansive, with foundations visibly influencing the price",
       x= "Above ground living area square feet",
       y= "Sell Price in log",
       color="Foundation quality:\n6=best (concrete) \n1=worst(slab)")
```




```
train %>%
  ggplot(aes(TotalInsideSF, SalePrice))+
  geom_point(alpha = 0.5, aes(color = NeighborhoodScored))+
  geom_smooth(method = "lm")+
  theme_bw()+
  labs(title = "Figure 10: Size and location of a house vs its sale price",
       subtitle = "Larger houses are more expansive, with location visibly influencing the price",
       x= "First and second floor square feet",
       y= "Sell Price in log",
       color="Location Score:\n7 = most elite \n1= least elite")
```



The figures above visually present what we already saw in the correlation report (Figure 4).

First, from the graphs above it is visible that houses with higher rated quality are more expansive. Moreover, higher rated houses are also usually newer. Best rated houses (grades around 9 and 10) have around 200.000$ higher prices than the average.

Second, we could also see that there are luxurious locations such as Northridge, Northridge Heights and Stone Brook that have around 100.000$ higher prices than the average. On the other side, locations such as Meadow Village, Iowa DOT and Briardale have significantly lower prices than the average price. Furthermore, elite location usually have newer houses, while older houses are usually located in least elite neighborhoods. Exception could be Crawfor neighborhood with slightly older but above average priced houses.

Third, scatter-plots above put in a relation size of a house (via: i) above ground living area and ii) first and second floor square feet) and its Sale Prices. As expected, as the size of the house increases so does its price. It is also visible that the foundation quality and the location have a visible influence on the selling price of a house: e.g. for the houses of the same size, if foundation quality or location is better/more elite, this increases its price.

6. Machine Learning Models and Results

For data cleaning and feature engineering, we merged train and test data-sets from Kaggle. We find out that Kaggle test data-set has no SalePrice variable, so it cannot be useful for using it for ML testing or ML validation. In order to do ML predictions we will need to focus on the cleaned "train" part the data (i.e. dataset1[1:1460,]).

We could enlarge the train part of the data-set and this would give us slightly better RMSE results, but this would leave too small proportion of the data for test and validation. That is why the ratio 60%/20%/20%

seems appropriate. Our ML models will focus on previously emphasized variables that are most correlated with the sale price of houses from 4 dimensions, namely: size, location, age and quality of houses.

Therefore, we will pick the most correlated variables with the sale price - correlation above 0.45 - for the subset of the data in order to do ML predictions of the house prices.

```
# Separation of the top correlated variables with Sale Price (with correlation above 0.45)
basedf<- subset(train, select = c(OverallQual , NeighborhoodScored, TotalInsideSF, GarageCars,
                                  GrLivArea, KitchenQual, GarageArea, TotalBsmtSF , FullBath,
                                  YearBuilt, YearRemodAdd, FoundationScore, TotRmsAbvGrd,
                                  Fireplaces, HeatingQC, ExterQual, SalePrice))
```

After creating the smaller data-set containing only the most correlated variables with SalePrice variable, we will divide it in three parts in the following ratio: 60% for train_set, 20% for test_set, and 20% for validation.

```
## Tripartite Data partition:
set.seed(99, sample.kind = "Rounding")

# Set the fractions of the df for training, validation, and test.
fractionTraining <- 0.6
fractionValidation <- 0.2
fractionTest <- 0.2
# Compute sample sizes.
sampleSizeTraining <- floor(fractionTraining * nrow(basedf))
sampleSizeValidation <- floor(fractionValidation * nrow(basedf))
sampleSizeTest <- floor(fractionTest * nrow(basedf))
# Creating the randomly-sampled indices for the dataframe. Use setdiff() to
# avoid overlapping subsets of indices.
indicesTraining <- sort(sample(seq_len(nrow(basedf)), size=sampleSizeTraining))
indicesNotTraining <- setdiff(seq_len(nrow(basedf)), indicesTraining)
indicesValidation <- sort(sample(indicesNotTraining, size=sampleSizeValidation))
indicesTest <- setdiff(indicesNotTraining, indicesValidation)
# Finally, output the three df for training, test and final validation.
train_set <- basedf[indicesTraining, ]
validation <- basedf[indicesValidation, ]
test_set <- basedf[indicesTest, ]
```

After defining our train, test and validation data-sets we can create our ML models. We will use four different models in order to do prediction: linear regression model, lasso regression model, ridge regression model and random forest model.

We will measure the performance of these models with residual mean squared error (RMSE) metrics. The RMSE tells us the average distance between the predicted values from the model and the actual values in the dataset and this allow us to see typical error loss. The result of RMSE is in the same units as the outcome variable (in our case - log of sale price). In other words, we can interpret the RMSE similarly to a standard deviation: it is the typical error we make when predicting a selling price of a house. Therefore, ML model is better in predicting house prices when its RMSE score is lower or in other words: the lower the RMSE, the better a given model is able to "fit" a dataset.

Moreover, Kaggle web-site states that the RMSE must be calculated between the logarithm of the predicted value and the logarithm of the observed sale price. Therefore, in order to measure the performance of our ML predictions, the described log RMSE metrics will be used.

Before training our ML models we will define cross-validation plan. Cross-validation is also known as a resampling method because it involves fitting the same statistical method multiple times using different subsets of the data. Here we do ten-fold cross-validation to train our models with the caret package.

```r
cv_plan <- trainControl(method = "cv", number = 10)
```

- 1. ML Model: Linear Model - We will add previously defined cross-validation to our lm model, do preprocess in order to perform a Principal Component Analysis, and also center and scale predictors and identify predictors with near zero variance.

```r
set.seed(99, sample.kind = "Rounding")

model_lm <- train(SalePrice ~ .,
                  method = "lm",
                  trControl = cv_plan,
                  preProcess = c("nzv", "center", "scale", "pca"),
                  data = train_set)

pred_lm <- predict(model_lm, test_set)
rmse_lm <- RMSE((test_set$SalePrice), pred_lm)
rmse_lm
```

```
## [1] 0.1463283
```

RMSE score of our first ML linear model is 0.1463

- 2. ML Model: Ridge regression - For the next to models we will use GLMnet package. GLMnet package offers amended regression approach similar to linear regression, but it also provides a way how to, on one side, penalizes number of non-zero-coefficients - called "lasso regression" - and on the other side provides a way how to penalizes absolute magnitude of coefficients - called "ridge regression". This helps in dealing with collinearity and small datasets. Function tuneGrid offers a way how to choose between pure "ridge regression" (setting the alpha = 0) and pure "lasso regression" (setting the alpha = 1). Other tuning settings are similar to linear regression (except not having PCA). We will try both "Ridge regression" and "Lasso regression" approaches.

```r
set.seed(99, sample.kind = "Rounding")

model_ridge <- train(SalePrice ~ .,
                     data=train_set,
                     tuneGrid = expand.grid(alpha = 0,
                                            lambda = seq(0.0001, 1, length = 20)),
                     method = "glmnet",
                     trControl = cv_plan,
                     preProcess = c("nzv", "center", "scale"))

pred_ridge <- predict(model_ridge, test_set)
rmse_ridge <- RMSE((test_set$SalePrice), pred_ridge)
rmse_ridge
```

```
## [1] 0.1478478
```

- 3. ML Model: Lasso regression

```r
set.seed(99, sample.kind = "Rounding")

model_lasso <- train(SalePrice ~ .,
                     data=train_set,
                     tuneGrid = expand.grid(alpha = 1,
                                            lambda = seq(0.0001, 1, length = 20)),
                     method = "glmnet",
                     trControl = cv_plan,
                     preProcess = c("nzv", "center", "scale"))

pred_lasso <- predict(model_lasso, test_set)
rmse_lasso <- RMSE((test_set$SalePrice), pred_lasso)
rmse_lasso
```

```
## [1] 0.1471446
```

RMSE score of Ridge regression model is 0.1478 and of Lasso regression model slightly better, namely 0.1471. Both models have slightly worse result than Linear regression model.

- 4. ML Model: Random Forest - As the last ML model we will use random forest ML approach. For this we will use "ranger" method from the caret package. Ranger of caret package is a fast implementation of random forest, particularly suited for high dimensional data and for our case of not very high computational power. Here, the most important tuning parameter is the number of randomly selected variables at each split for which we use tuneLength control in the code. The default of tuneLength is 3 (it means that it tries 3 different models), but we will set it to 13.

```r
set.seed(99, sample.kind = "Rounding")

model_rf <- train(SalePrice ~ .,
                  tuneLength = 13,
                  data = train_set,
                  method = "ranger",
                  trControl = cv_plan)

pred_rf <- predict(model_rf, test_set)
rmse_rf <- RMSE((test_set$SalePrice), pred_rf)
rmse_rf
```

```
## [1] 0.1439269
```

We can see that random forest ML approach generated the best RMSE score = 0.1439. At the end of this section we present the RMSE results of all 4 trained models:

```r
# RMSE score compare
data.frame(Models = c("Linear Reggresion","Ridge Regression", "Lasso Regression", "Random Forest"),
           Train_RMSE = round(c(rmse_lm,rmse_ridge,rmse_lasso,rmse_rf), 6))
```

```
##              Models Train_RMSE
## 1 Linear Reggresion   0.146328
## 2  Ridge Regression   0.147848
## 3  Lasso Regression   0.147145
## 4     Random Forest   0.143927
```

In order to check the RMSE results obtained in the previous section and in order to avoid overtraining we will do the final validation of trained models on the validation data-set. At the begging of this section we kept 20% of the data for the final validation of our trained ML models. We will do validation for all trained results in order to compare train/test and train/validation results and to see if random forest ML model is still the one with the lowest RMSE.

```
# Predictions of the Linear model - final validation
pred_val_lm <- predict(model_lm, validation)
rmse_val_lm <- RMSE((validation$SalePrice), pred_val_lm)
```

```
# Predictions of Ridge model - final validation
pred_val_ridge <- predict(model_ridge, validation)
rmse_val_ridge <- RMSE((validation$SalePrice), pred_val_ridge)
```

```
# Predictions of Lasso model - final validation
pred_val_lasso <- predict(model_lasso, validation)
rmse_val_lasso <- RMSE((validation$SalePrice), pred_val_lasso)
```

```
# Predicting of Random Forest model - final validation
pred_val_rf <- predict(model_rf, validation)
rmse_val_rf <- RMSE((validation$SalePrice), pred_val_rf)
```

```
# Comparison of RMSEs between train/test and validation sets
data.frame(Model_type = c("Linear Reggresion","Ridge Regression", "Lasso Regression", "Random Forest"),
           RMSE_original_train = c(rmse_lm,rmse_ridge,rmse_lasso,rmse_rf),
           RMSE_validation = c(rmse_val_lm, rmse_val_ridge, rmse_val_lasso, rmse_val_rf))
```

```
##          Model_type RMSE_original_train RMSE_validation
## 1 Linear Reggresion           0.1463283       0.1437728
## 2  Ridge Regression           0.1478478       0.1443947
## 3  Lasso Regression           0.1471446       0.1440679
## 4     Random Forest           0.1439269       0.1346283
```

The final validation-results of our ML models are similar with the results we acquired during training. Differences between the RMSE values of original train and validation phase are not large.

The best result during training (0.144) and final validation (0.135) was the one acquired with the Random Forest ML approach .

7. Summary and Conclusion

In this project the goal was to create the house price prediction system with the help of the Kaggle data-set.

The first thing we did was to clean the data, after which we did some data remodeling and engineering in order to enhance variables we have for our machine learning purposes. Namely, first we recoded some descriptive variables in order to get ordinal data; secondly, we created some new features/variables from existing variables - foundation score and location score; and thirdly, we dealt with skewness of the target value - SalePrice - by applying the log transformation.

After that, we visualized variables that are most correlated with the sale price of houses. The results showed that four dimension most correlated with the sale price of a house are its: size, location, age and quality. We focused on the 16 most correlated variables (with correlation above 0.45) from those 4 dimension in order to do ML predictions of the house prices. Further, we dived this final, cleaned and focused data-set in 3 parts: 60% of it for training the data, 20% for testing and 20% for the final validation of our ML models.

Finally, we used four machine learning models to build our house prediction system, namely: linear regression model, lasso regression model, ridge regression model and random forest model. The RMSE metric was used as a measure of the performance of these models. Obtained RMSE scores during training/testing and final validation phase are summarized in the table below:

Machine Learning Model	RMSE - training
Linear Reggresion	0.146328
Ridge Regression	0.147847
Lasso Regression	0.147144
Random Forest	0.143926

Machine Learning Model	RMSE - validation
Linear Reggresion	0.143772
Ridge Regression	0.144394
Lasso Regression	0.144067
Random Forest	0.134628

The best result during training (0.144) and final validation (0.135) was the one acquired with the Random Forest ML approach.

Naturally, the presented work can be used as a basis for future improvement. The best Kaggle results have much better RMSE scores. Some of the limitations and possible future updates of this work would include:

- Machine learning models: we could have used some other, more advanced ML models in order to acquire better RMSE scores
- Data cleaning and engineering: data cleaning was important prerequisite for building ML models in this case. This work could have been done in some other way then it was presented here. Also, some other custom-made features/variables could have been used in order use them during machine learning.
- Outliers: when the data was visualized some outliers were visible, however I decided not to delete them in order to use all the original data.
- The data-set used for training of our machine learning models is not large: 876 observations and 16 most correlated variables with our outcome variable (sale price). Some other approach of data training, data partition and cross-validations could have been used in order to train ML models with lower RMSE values.