House Prices

Contents

0.1 Overview

The aim of the project is to develop a predictive model that predicts final price of homes? This project is a submission for Kaggle House prices compeition to demostrate my knowledge of R having completed the HarvardX - Data science course.

Ask a home buyer to describe their dream house, and they probably won't begin with the height of the basement ceiling or the proximity to an east-west railroad. But this playground competition's dataset proves that much more influences price negotiations than the number of bedrooms or a white-picket fence. With 79 explanatory variables describing (almost) every aspect of residential homes in Ames, Iowa, the aim of this project is to predict the final price of each home.

The Ames Housing dataset was compiled by Dean De Cock for use in data science education. It's an incredible alternative for data scientists looking for a modernized and expanded version of the often cited Boston Housing dataset.

0.2 Goal and Metrics of the project

The objective of this project is to predict the sales price for each house. For each Id in the test set, the value of the SalePrice variable must be predicted. The predicted value is evaluated using Root-Mean-Squared-Error (RMSE) between the logarithm of the predicted value and the logarithm of the observed sales price.

0.3 Introduction to Dataset

The dataset focuses on the quality and quantity of many physical attributes of a property that home owners are interested in. The dataset is divided into two, the train set and test set. The trainset is used for exploration while the test set is used for performance eveluation of the selected moodel. We will start by loading the data and relevant library.

```
#load the libraries required for the project, if not exist then download
#load the required packages, if not available then download
if(!require(tidyverse)) install.packages("tidyverse", repos = "http://cran.us.r-project.org")
```

```
## Loading required package: tidyverse
```

```
## -- Attaching packages ---------------------------------------------------------------------------
```

```
## v ggplot2 3.2.1     v purrr   0.3.3
## v tibble  2.1.3     v dplyr   0.8.3
## v tidyr   1.0.0     v stringr 1.4.0
## v readr   1.3.1     v forcats 0.4.0

## -- Conflicts ------------------------------------------------------------------------------------ tidy
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

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

```
## Loading required package: caret

## Loading required package: lattice

##
## Attaching package: 'caret'

## The following object is masked from 'package:purrr':
##
##     lift
```

```
if(!require(dplyr)) install.packages("tidyverse", repos = "http://cran.us.r-project.org")
if(!require(ggplot2)) install.packages("caret", repos = "http://cran.us.r-project.org")
if(!require(data.table)) install.packages("data.table", repos = "http://cran.us.r-project.org")
```

```
## Loading required package: data.table

##
## Attaching package: 'data.table'

## The following objects are masked from 'package:dplyr':
##
##     between, first, last

## The following object is masked from 'package:purrr':
##
##     transpose
```

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

```
## Loading required package: randomForest

## randomForest 4.6-14

## Type rfNews() to see new features/changes/bug fixes.

##
## Attaching package: 'randomForest'
```

```
## The following object is masked from 'package:dplyr':
##
##     combine

## The following object is masked from 'package:ggplot2':
##
##     margin
```

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

```
## Loading required package: Boruta

## Loading required package: ranger

##
## Attaching package: 'ranger'

## The following object is masked from 'package:randomForest':
##
##     importance
```

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

```
## Loading required package: rpart
```

```
#read the data
train_set <- read.csv("https://projects.smartsolutaris.com/train.csv", stringsAsFactors=FALSE)
test_set <- read.csv("https://projects.smartsolutaris.com/test.csv", stringsAsFactors=FALSE)
```

Then, evaluating the dimensions of the train set and test set

```
dim(train_set)
```

```
## [1] 1460   81
```

```
dim(test_set)
```

```
## [1] 1459   80
```

```
#train_set is a data frame
class(train_set)
```

```
## [1] "data.frame"
```

We need to understand the structure of the dataset, this will allow us gain insight into the data. To gain insight the keyword str is used to define the structure and type of each column. The column definition and data dictionary can be found here => https://projects.smartsolutaris.com/data_description.txt

```r
#we need to understand the structure of the dataset. To gain insight the keyword str is used.
str(train_set)
```

```
## '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 ...
```

```
#description of factors can be found here   https://projects.smartsolutaris.com/data_description.txt
```

From the result above, we can notice that the dataset has missing value. In some cases, the NA represents that the feature doesnt exist. Thus the next step is cleaning the data.

0.4 Exploratory Data Analysis and Feature Analysis

Here we check if there are missing values, skewed fields. Boruta Algorithm is used for Feature andlysis and handling of missing data

0.4.1 Feature Analysis - Boruta: Wrapper Algorithm for All Relevant Feature Selection.

Boruta is a feature selection algorithm. Precisely, it works as a wrapper algorithm around Random Forest. feature selection is a crucial step in predictive modeling. This technique achieves supreme importance when a data set comprised of several variables is given for model building. This section determines what features may be relevant to predicting house sale price.

```r
#Feature ANalysis and Data Cleanup
Identifier <- "Id"
Target <- "SalePrice"
# deterimine data types
Features <- setdiff(names(train_set),c(Identifier,Target))
Features_datatypes <- sapply(Features,function(x){class(train_set[[x]])})
table(Features_datatypes)
```

```
## Features_datatypes
## character   integer
##        43        36
```

```r
# categorize data types in the data set?
unique_classes <- unique(Features_datatypes)

attr_datatypes <- lapply(unique_classes,function(x){names(Features_datatypes[Features_datatypes==x])})
names(attr_datatypes) <- unique_classes

# pull out the response variable
response <- train_set$SalePrice

# missing values are handled as follows: * missing numeric data is set to -1 * missing character data i
for (x in attr_datatypes$integer){
  train_set[[x]][is.na(train_set[[x]])] <- -1
}

for (x in attr_datatypes$character){
  train_set[[x]][is.na(train_set[[x]])] <- 0
}

for (x in attr_datatypes$integer){
  test_set[[x]][is.na(test_set[[x]])] <- -1
}

for (x in attr_datatypes$character){
 test_set[[x]][is.na(test_set[[x]])] <- 0
}

# remove identifier and response variables
train_df <- train_set[Features]
 #use Boruta to compute important attributes for SalePrice prediction
set.seed(13)
bor.results <- Boruta(train_df,response,
                      maxRuns=101,
                      doTrace=0)
bor.results
```

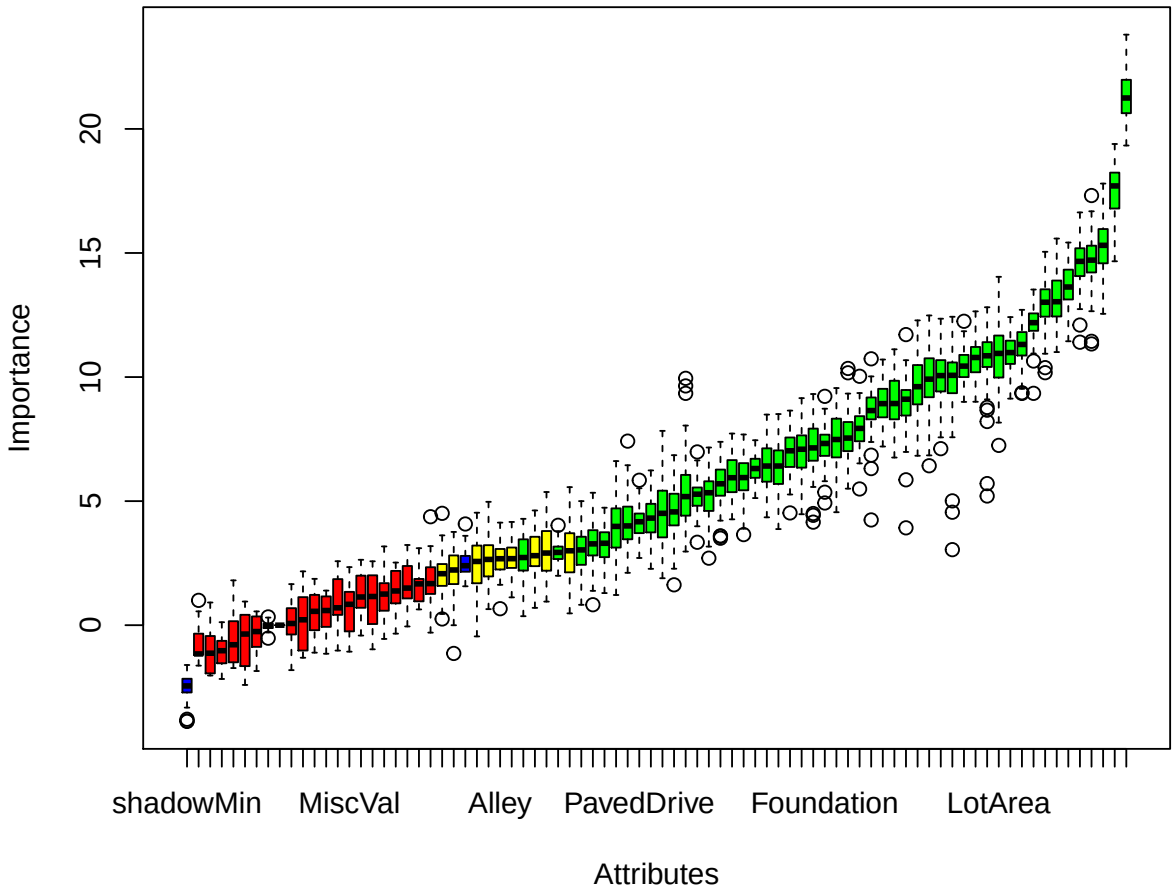
```
## Boruta performed 100 iterations in 6.445083 mins.
##  50 attributes confirmed important: BedroomAbvGr, BldgType,
## BsmtCond, BsmtFinSF1, BsmtFinType1 and 45 more;
##  20 attributes confirmed unimportant: BsmtFinSF2, BsmtHalfBath,
## Condition2, ExterCond, Heating and 15 more;
##  9 tentative attributes left: Alley, BsmtExposure, Condition1,
```

```
## Electrical, EnclosedPorch and 4 more;
```

```
#get the important attributes
getSelectedAttributes(bor.results)
```

```
##  [1] "MSSubClass"    "MSZoning"      "LotArea"       "LandContour"
##  [5] "Neighborhood"  "BldgType"      "HouseStyle"    "OverallQual"
##  [9] "OverallCond"   "YearBuilt"     "YearRemodAdd"  "Exterior1st"
## [13] "Exterior2nd"   "MasVnrArea"    "ExterQual"     "Foundation"
## [17] "BsmtQual"      "BsmtCond"      "BsmtFinType1"  "BsmtFinSF1"
## [21] "BsmtFinType2"  "BsmtUnfSF"     "TotalBsmtSF"   "HeatingQC"
## [25] "CentralAir"    "X1stFlrSF"     "X2ndFlrSF"     "GrLivArea"
## [29] "BsmtFullBath"  "FullBath"      "HalfBath"      "BedroomAbvGr"
## [33] "KitchenAbvGr"  "KitchenQual"   "TotRmsAbvGrd"  "Functional"
## [37] "Fireplaces"    "FireplaceQu"   "GarageType"    "GarageYrBlt"
## [41] "GarageFinish"  "GarageCars"    "GarageArea"    "GarageQual"
## [45] "GarageCond"    "PavedDrive"    "WoodDeckSF"    "OpenPorchSF"
## [49] "Fence"         "SaleCondition"
```

```
plot(bor.results)
```




```
#From the result above, notice that 50 attributes are confirmed important.
attStats(bor.results)
```

	meanImp	medianImp	minImp	maxImp	normHits	decision
MSSubClass	10.0150630	10.0585017	7.1138241	12.3515162	1.00	Confirmed
MSZoning	9.6196905	9.6111321	6.8279895	12.2843150	1.00	Confirmed
LotFrontage	1.7530458	1.6754663	-0.2987363	4.3716214	0.04	Rejected
LotArea	10.8461116	10.9525669	7.2464649	14.0390530	1.00	Confirmed
Street	0.4111002	0.5914736	-1.1504427	1.3942054	0.00	Rejected
Alley	2.6750594	2.6785115	0.6641863	4.1338571	0.64	Tentative
LotShape	2.9784016	2.9993645	0.4766516	5.5639638	0.63	Tentative
LandContour	3.2376645	3.3015819	1.2947024	4.7349664	0.83	Confirmed
Utilities	0.0000000	0.0000000	0.0000000	0.0000000	0.00	Rejected
LotConfig	0.2224083	0.2176043	-1.3116917	2.1693442	0.00	Rejected
LandSlope	2.4702249	2.5694847	-0.4514954	4.5328581	0.48	Tentative
Neighborhood	8.6826415	8.6491193	4.2413846	10.7351491	1.00	Confirmed
Condition1	2.1668323	2.2250758	-1.1385116	3.7584138	0.43	Tentative
Condition2	-0.7109773	-1.1582337	-1.6288056	1.0010015	0.00	Rejected

	meanImp	medianImp	minImp	maxImp	normHits	decision
BldgType	7.4633655	7.4843162	4.5532316	9.5569741	1.00	Confirmed
HouseStyle	7.0131080	7.0904829	4.4716972	9.1508474	1.00	Confirmed
OverallQual	17.5117756	17.7028791	14.6660665	19.3925063	1.00	Confirmed
OverallCond	5.4317954	5.1817066	2.9678336	9.9489961	0.99	Confirmed
YearBuilt	12.9618514	13.0159398	10.1754553	15.0486351	1.00	Confirmed
YearRemodAdd	11.2929459	11.3143737	9.3415434	12.7087489	1.00	Confirmed
RoofStyle	2.5769620	2.6511643	0.6469199	4.9718325	0.52	Tentative
RoofMatl	1.0111311	1.1558984	-0.9712564	2.5753463	0.01	Rejected
Exterior1st	4.3053551	4.3152415	2.2720035	6.2378285	0.96	Confirmed
Exterior2nd	4.1309712	4.0009763	2.1108359	7.4179183	0.95	Confirmed
MasVnrType	2.9374326	2.8011309	0.6987453	4.6239673	0.67	Tentative
MasVnrArea	5.9647926	5.9432537	4.2668867	7.7237733	1.00	Confirmed
ExterQual	12.1954121	12.1917678	9.3441517	13.5275093	1.00	Confirmed
ExterCond	1.2452012	1.1368282	-0.4170957	2.6357803	0.02	Rejected
Foundation	7.2821148	7.3233878	4.9249700	9.2234019	1.00	Confirmed
BsmtQual	8.9788318	9.1076687	3.9235470	11.7115549	1.00	Confirmed
BsmtCond	4.6390891	4.5659843	1.6269682	6.8512259	1.00	Confirmed
BsmtExposure	2.9734849	2.9069897	0.9516483	5.3672038	0.63	Tentative
BsmtFinType1	5.9276912	5.9482507	3.6504047	7.6858485	1.00	Confirmed
BsmtFinSF1	9.8849390	9.9157671	6.4255480	12.4826686	1.00	Confirmed
BsmtFinType2	2.9869466	3.0337659	0.8214114	4.9997953	0.80	Confirmed
BsmtFinSF2	1.3964738	1.3791362	-0.3403870	2.5243592	0.01	Rejected
BsmtUnfSF	6.4363625	6.4155910	3.8735718	8.5093142	1.00	Confirmed
TotalBsmtSF	14.7352240	14.7116936	11.3345352	17.3156288	1.00	Confirmed
Heating	0.8272464	0.7004750	-1.0220808	2.5776912	0.00	Rejected
HeatingQC	6.3027706	6.3132299	5.1226729	7.4544502	1.00	Confirmed
CentralAir	6.4339356	6.4141244	4.3471557	8.4875055	1.00	Confirmed
Electrical	2.7151059	2.6806553	1.1217611	4.1609296	0.65	Tentative
X1stFlrSF	14.6062854	14.6622853	11.4048711	16.6351398	1.00	Confirmed
X2ndFlrSF	15.2398072	15.3089757	12.5481723	17.7981689	1.00	Confirmed
LowQualFinSF	-0.3572059	-0.2560683	-1.8466569	0.5547814	0.00	Rejected
GrLivArea	21.3300243	21.2465543	19.3317862	23.7990390	1.00	Confirmed
BsmtFullBath	4.5901162	4.5075853	1.8920350	7.8308909	0.98	Confirmed
BsmtHalfBath	1.5900925	1.6644952	0.6364851	3.1030147	0.01	Rejected
FullBath	10.7449283	10.7884473	9.0051766	12.6471064	1.00	Confirmed
HalfBath	7.9463234	7.9306304	5.4897160	10.0317441	1.00	Confirmed
BedroomAbvGr	7.2415045	7.1442611	4.1525355	9.3166720	1.00	Confirmed
KitchenAbvGr	5.2395258	5.2797233	3.3421116	6.9826927	1.00	Confirmed
KitchenQual	10.4606795	10.4408999	9.0000337	12.2478160	1.00	Confirmed
TotRmsAbvGrd	9.0342897	8.9328634	6.7566501	11.1179940	1.00	Confirmed
Functional	3.3195454	3.2777709	0.8249590	5.3377562	0.82	Confirmed
Fireplaces	9.8137735	10.0693439	3.0445114	12.4299860	1.00	Confirmed
FireplaceQu	10.7448694	10.8583386	5.2061841	12.8133448	1.00	Confirmed
GarageType	8.9599127	8.9287772	7.2015018	10.7064060	1.00	Confirmed
GarageYrBlt	10.9693031	10.9864988	9.1308071	12.4168158	1.00	Confirmed
GarageFinish	7.6038744	7.5410643	5.5031663	10.3470867	1.00	Confirmed
GarageCars	13.1558990	13.0365477	11.0069651	15.5824200	1.00	Confirmed
GarageArea	13.6745035	13.6292180	11.4429160	15.4214319	1.00	Confirmed
GarageQual	5.2098347	5.3372045	2.7009687	7.1606593	1.00	Confirmed
GarageCond	5.6880819	5.6984124	3.5053177	7.3900937	1.00	Confirmed
PavedDrive	4.1157837	4.1691935	2.7083331	5.8425406	0.99	Confirmed
WoodDeckSF	3.9472605	3.9810188	1.2165799	6.6138923	0.88	Confirmed

	meanImp	medianImp	minImp	maxImp	normHits	decision
OpenPorchSF	6.9854104	7.0288576	4.5241445	8.6486414	1.00	Confirmed
EnclosedPorch	2.0595946	2.0747056	0.2505073	4.5080630	0.32	Tentative
X3SsnPorch	-0.4931520	-0.7876477	-1.7276357	1.8065193	0.00	Rejected
ScreenPorch	1.6422217	1.4967227	-0.0505559	3.2320466	0.07	Rejected
PoolArea	-1.0425395	-1.0287129	-2.1675185	0.1200381	0.00	Rejected
PoolQC	-1.0536091	-1.1289794	-2.0286997	0.9144868	0.00	Rejected
Fence	2.9185473	2.9322437	1.9907863	4.0259910	0.79	Confirmed
MiscFeature	0.4693449	0.5579630	-1.1064375	1.8672623	0.00	Rejected
MiscVal	0.5664266	0.8420788	-1.0665473	2.3410332	0.01	Rejected
MoSold	-0.5875716	-0.3570484	-2.4077277	0.9536509	0.00	Rejected
YrSold	0.0795957	0.0692829	-1.8107680	1.6551763	0.00	Rejected
SaleType	1.2672563	1.2542439	-0.5536398	3.1740607	0.02	Rejected
SaleCondition	2.7261019	2.7269203	0.3630806	4.2863960	0.69	Confirmed

```
comp_boruta <- arrange(cbind(attr=rownames(attStats(bor.results)), attStats(bor.results)),desc(medianImp
confirmed_columns <- comp_boruta %>% filter(decision == "Confirmed") %>% select(attr)
removed_columns <- comp_boruta %>% filter(decision != "Confirmed") %>% select(attr)
```

Based on the result generated from Boruta Analysis, the 50 confirmed attributes as important will be used for model building

```
#creating a confirmed dataset from Boruta Analysis to create the trainset
#creating a confirmed dataset from Boruta Analysis to create the trainset
final_trainset <- train_set %>% select(Id, GrLivArea,OverallQual,X2ndFlrSF,TotalBsmtSF,X1stFlrSF,Garage
                                       GarageCars,YearBuilt,ExterQual,YearRemodAdd,GarageYrBlt,LotArea,
                                       FireplaceQu,FullBath,Fireplaces,MSSubClass,BsmtFinSF1,
                                       BsmtQual,TotRmsAbvGrd,GarageType,Neighborhood,HalfBath,GarageFin
                                       Foundation,BedroomAbvGr,HouseStyle,OpenPorchSF,BsmtUnfSF,Central
                                       BsmtFinType1,MasVnrArea,GarageCond,GarageQual,KitchenAbvGr,Overa
                                       BsmtCond,BsmtFullBath,PavedDrive,WoodDeckSF,
                                       LandContour,BsmtFinType2,Fence,SaleCondition,SalePrice)

final_testset <- test_set %>% select(Id, GrLivArea,OverallQual,X2ndFlrSF,TotalBsmtSF,X1stFlrSF,GarageAr
                                     GarageCars,YearBuilt,ExterQual,YearRemodAdd,GarageYrBlt,LotArea,
                                     FireplaceQu,FullBath,Fireplaces,MSSubClass,BsmtFinSF1,
                                     BsmtQual,TotRmsAbvGrd,GarageType,Neighborhood,HalfBath,GarageFinis
                                     Foundation,BedroomAbvGr,HouseStyle,OpenPorchSF,BsmtUnfSF,CentralAi
                                     BsmtFinType1,MasVnrArea,GarageCond,GarageQual,KitchenAbvGr,Overall
                                     BsmtCond,BsmtFullBath,PavedDrive,WoodDeckSF,
                                     LandContour,BsmtFinType2,Fence,SaleCondition)
```

0.4.2 Data Modelling

for cross validation purpose, there is need to divide the feature extracted train_set into training and validation.

```
#for cross validation purpose, there is need to divide the train_set into training and validation.
#creating index for division
index <- createDataPartition(y = final_trainset$SalePrice, times = 1, p = 0.1, list = FALSE)
#using the index to split into training and validation
```

```
training <- final_trainset[-index,]
validation <- final_trainset[index,]

#Convert all character variable into factor in one line:
training <- training %>% mutate_if(is.character, as.factor)
validation <- validation %>% mutate_if(is.character, as.factor)
```

0.4.2.1 Evaluation of the Models

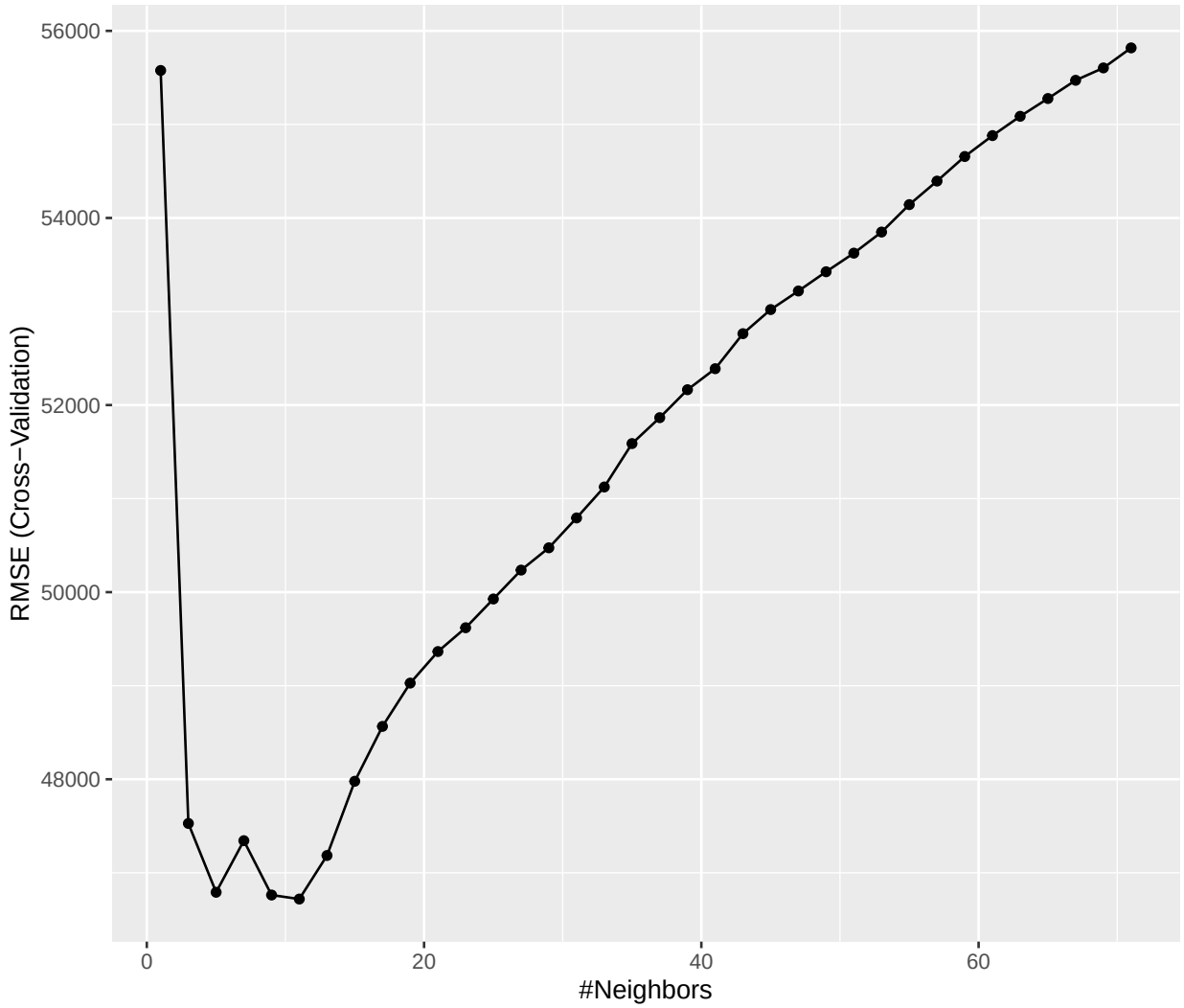
The predicted value is evaluated using Root-Mean-Squared-Error (RMSE) between the logarithm of the predicted value and the logarithm of the observed sales price.

```
## Evaluation of predicted sale price

RMSE <- function(predicted_val, actual_value)
{
  log(predicted_val) - log(actual_value)
}
```

0.4.2.2 Training the models

0.4.2.2.1 KNN - K-Nearest Neighbour

```
control <- trainControl(method = "cv", number = 10, p = .9)
train_knn <- train(SalePrice ~ ., method="knn", data = training, tuneGrid = data.frame(k = seq(1, 71, 2)
                   trControl = control)
ggplot(train_knn)
```




```
train_knn$bestTune
```

	k
6	11

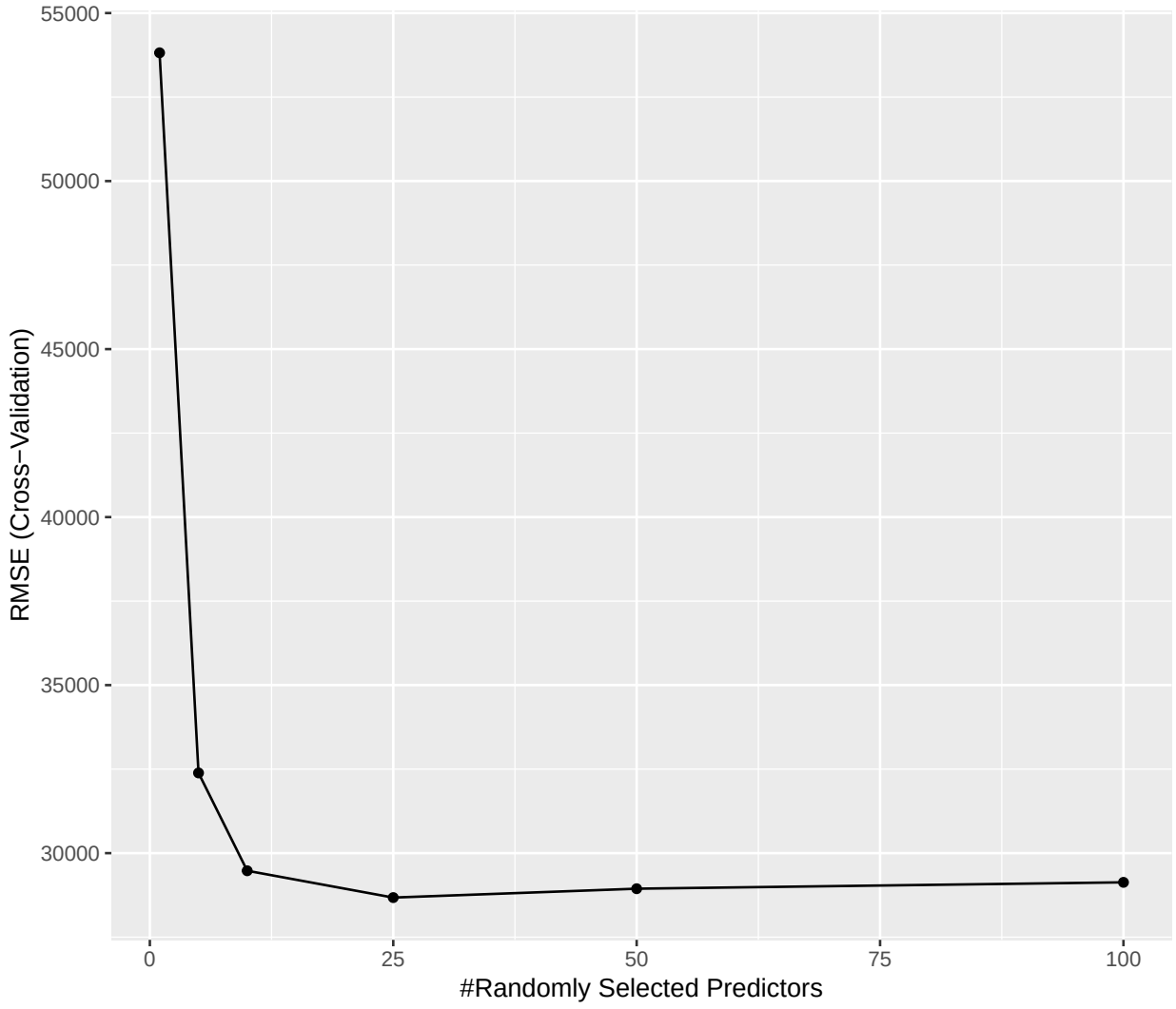
```
knn_y_hat <- predict(train_knn, validation)

rmse_knn <- RMSE(knn_y_hat, validation$SalePrice)
mean(rmse_knn)
```

```
## [1] 0.02129967
```

```
#creating a table to aggregate RMSE Models
rmse_results <- tibble(method = "K-Nearest Neighbour", RMSE = mean(rmse_knn))
```

0.4.2.2.2 Random Forest

```
control <- trainControl(method="cv", number = 5)
grid <- data.frame(mtry = c(1, 5, 10, 25, 50, 100))
train_rf <- train(SalePrice ~ .,
                  method = "rf",
                  data = training,
                  ntree = 150,
                  trControl = control,
                  tuneGrid = grid,
                  nSamp = 5000)
ggplot(train_rf)
```




```
train_rf$bestTune
```

	mtry
4	25

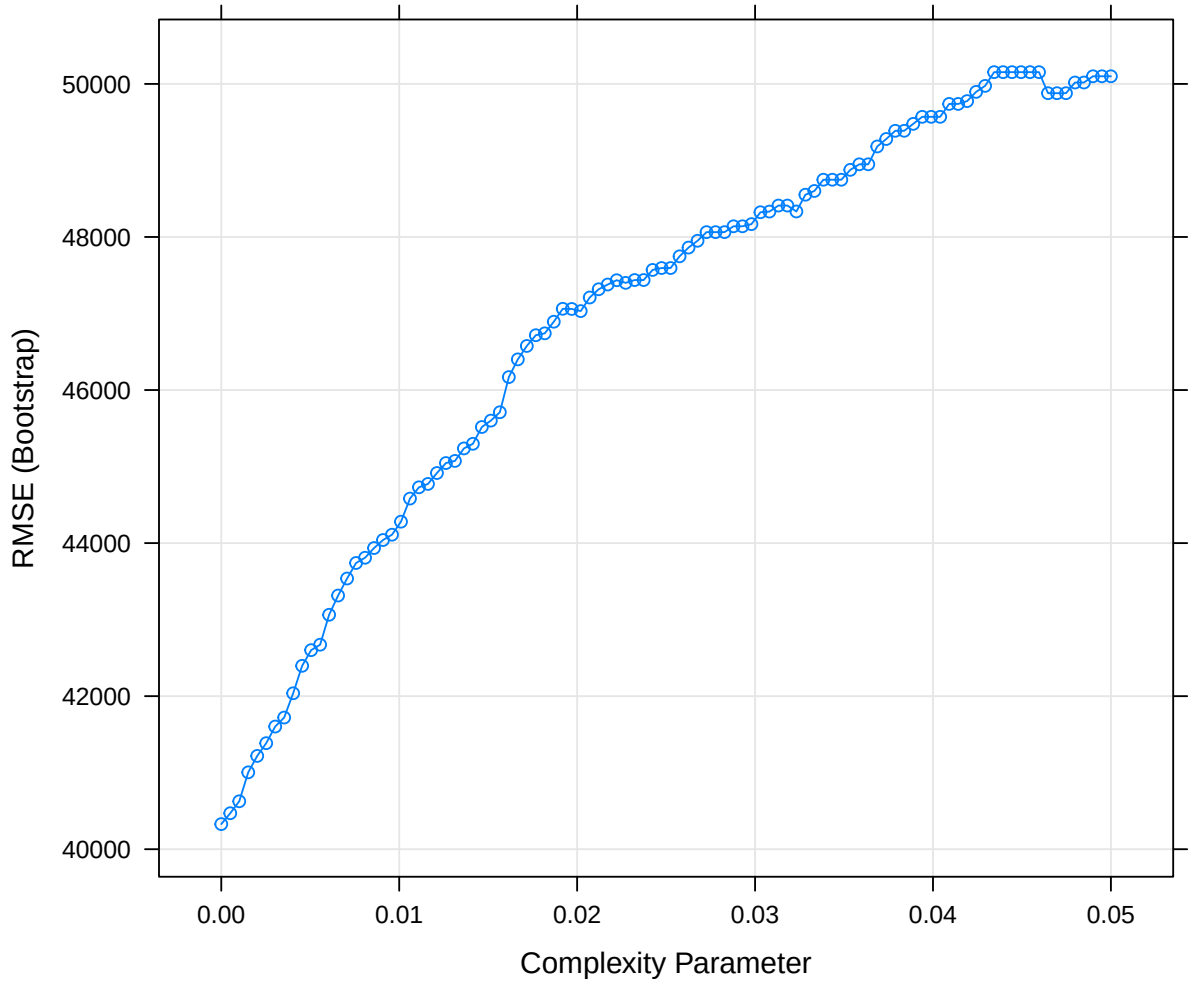
```
rf_yhat <- predict(train_rf, validation, type = "raw")
rmse_rf <- RMSE(rf_yhat, validation$SalePrice)
mean(rmse_rf)
```

```
## [1] 0.02386768
```

```
rmse_results<- bind_rows(rmse_results, tibble(method = "Random Forest", RMSE = mean(rmse_rf)))
```

0.4.2.2.3 Regression Tree

```
train_rpart <- train(SalePrice ~., method="rpart", tuneGrid = data.frame(cp = seq(0, 0.05, len = 100)),
                     data = training)
plot(train_rpart)
```




```
rpart_yhat <- predict(train_rpart, validation)
rmse_rpart <- RMSE(rpart_yhat, validation$SalePrice)
mean(rmse_rpart)
```

```
## [1] -0.0004491115
```

```
rmse_results <- bind_rows(rmse_results, tibble(method = "Regression Tree", RMSE = mean(rmse_rpart)))
```

0.4.2.3 Ensemble Model

one can usually greatly improve the final results by combining the results of different algorithms.we compute new class probabilities by taking the average of random forest, regression tree and kNN

```
p <- (knn_y_hat +rpart_yhat + rf_yhat)/3
rmse_em <- RMSE(p, validation$SalePrice)
rmse_results<- bind_rows(rmse_results, tibble(method = "Ensemble", RMSE = mean(rmse_em)))
mean(rmse_em)
```

```
## [1] 0.02019637
```

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

method	RMSE
K-Nearest Neighbour	0.0212997
Random Forest	0.0238677
Regression Tree	-0.0004491
Ensemble	0.0201964

0.5 Performance Analysis and Result

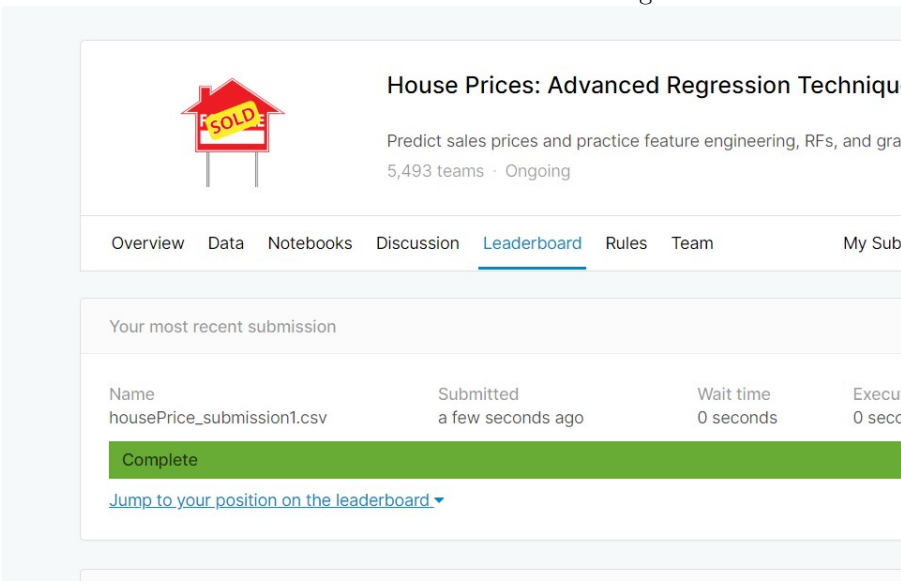
From the result above, the KNN method and Regression Tree will be used to train the full train dataset and the result used to predict SalePrice for test dataset.

```
#train the full test dataset using KNN

control <- trainControl(method = "cv", number = 10, p = .9)
train_final<- train(SalePrice ~ ., method="knn", data = final_trainset, tuneGrid = data.frame(k = seq(1
                    trControl = control)


#Prepare the prediction data
prediction <- predict(train_final, final_testset)
#Bind the result to the test dataset
output <- cbind(final_testset, prediction)
#prepare the final prediction result for Submission File Format
final_prep <- data.frame(Id= output$Id, SalePrice = output$prediction)

#Write the fianl result to a text file
write.csv(final_prep, file = "housePrice_submission1.csv", row.names = F)

#train the full test dataset using regression tree
final_rpart <- train(SalePrice ~., method="rpart", tuneGrid = data.frame(cp = seq(0, 0.05, len = 100)),
                     data = final_trainset)
```

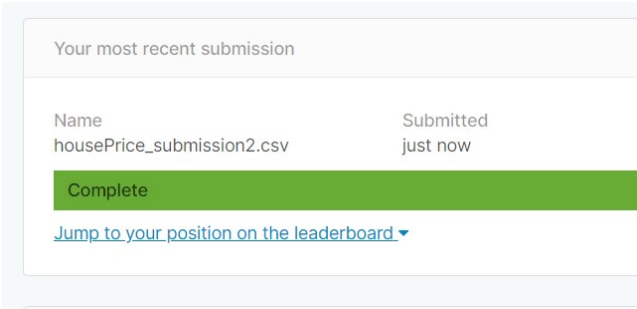
```r
#Prepare the prediction data
prediction_rpart <- predict(final_rpart, final_testset)
#Bind the result to the test dataset
output <- cbind(final_testset, prediction_rpart)
#prepare the final prediction result for Submission File Format
final_prep <- data.frame(Id= output$Id, SalePrice = output$prediction_rpart)

#Write the fianl result to a text file
write.csv(final_prep, file = "housePrice_submission2.csv", row.names = F)
```

0.6 Conclusion

Upon calculation the RMSE of the predicted value for KNN is 0.26514 while RPart is 0.19305. the image below shows the Kaggle computed result for KNN



the image below shows the Kaggle computed result for Regression Tree



The result implies that Regression Tree has a better prediction with low RMSE. Thus the prediction model can be improved since the RMSE target is zero.