Housing Prices

Lupita Sahu

12 October 2019

Introduction

The aim of this project is to predict Sale Prices of houses based on their different features. This project was taken from a Kaggle competition. The link to the competition is here: https://www.kaggle.com/c/house-prices-advanced-regression-techniques/overview

We have a train dataset to train the model and apply the same model on test dataset and submit the predicted Sale price for evaluation. Kaggle will take the predicted price and return the RMSE. For the simplicity and brevity of the report most of the code is not shown in this report. The R script contains all the codes.

You will have to download the data "house-prices-advanced-regression-techniques.zip" from Github link https://github.com/SLupita/HousingPricePrediction_Kaggle and paste in your working directory, if you want to run the R script.

I have explored a variety of machine learning models including the simple linear regression. Random Forest and GBM gave good accuracy, but GBM performed best after hypertuning. I have finally considered a hypertuned GBM model for our final submission.

The data

We will first load the data. Since the zip file already downloaded, we are just going to unzip and look at the files. The files where we have our data are: train.csv and test.csv.

Exploratory analysis

Let's look at the train and test datasets. There are 81 variables in total in train dataset and 80 in test dataset. Some of them have NA values. We will deal with them later in this project.

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

```
##        Id           MSSubClass       MSZoning     LotFrontage
##  Min.   :   1.0   Min.   : 20.0   C (all):  10   Min.   : 21.00
##  1st Qu.: 365.8   1st Qu.: 20.0   FV     :  65   1st Qu.: 59.00
##  Median : 730.5   Median : 50.0   RH     :  16   Median : 69.00
##  Mean   : 730.5   Mean   : 56.9   RL     :1151   Mean   : 70.05
##  3rd Qu.:1095.2   3rd Qu.: 70.0   RM     : 218   3rd Qu.: 80.00
##  Max.   :1460.0   Max.   :190.0                  Max.   :313.00
##                                                  NA's   :259
##     LotArea        Street      Alley      LotShape  LandContour
##  Min.   :  1300   Grvl:   6   Grvl:  50   IR1:484   Bnk:  63
##  1st Qu.:  7554   Pave:1454   Pave:  41   IR2: 41   HLS:  50
##  Median :  9478               NA's:1369   IR3: 10   Low:  36
##  Mean   : 10517                           Reg:925   Lvl:1311
##  3rd Qu.: 11602
##  Max.   :215245
##
##   Utilities      LotConfig    LandSlope   Neighborhood   Condition1
##  AllPub:1459   Corner : 263   Gtl:1382   NAmes  :225   Norm   :1260
##  NoSeWa:   1   CulDSac:  94   Mod:  65   CollgCr:150   Feedr  :  81
```

```
##               FR2    :  47   Sev:  13   OldTown:113   Artery :  48
##               FR3    :   4              Edwards:100   RRAn   :  26
##               Inside :1052              Somerst: 86   PosN   :  19
##                                         Gilbert: 79   RRAe   :  11
##                                         (Other):707   (Other):  15
##    Condition2     BldgType      HouseStyle    OverallQual
##  Norm   :1445   1Fam  :1220   1Story :726   Min.   : 1.000
##  Feedr  :   6   2fmCon:  31   2Story :445   1st Qu.: 5.000
##  Artery :   2   Duplex:  52   1.5Fin :154   Median : 6.000
##  PosN   :   2   Twnhs :  43   SLvl   : 65   Mean   : 6.099
##  RRNn   :   2   TwnhsE: 114   SFoyer : 37   3rd Qu.: 7.000
##  PosA   :   1                 1.5Unf : 14   Max.   :10.000
##  (Other):   2                 (Other): 19
##   OverallCond      YearBuilt     YearRemodAdd    RoofStyle
##  Min.   :1.000   Min.   :1872   Min.   :1950   Flat   :  13
##  1st Qu.:5.000   1st Qu.:1954   1st Qu.:1967   Gable  :1141
##  Median :5.000   Median :1973   Median :1994   Gambrel:  11
##  Mean   :5.575   Mean   :1971   Mean   :1985   Hip    : 286
##  3rd Qu.:6.000   3rd Qu.:2000   3rd Qu.:2004   Mansard:   7
##  Max.   :9.000   Max.   :2010   Max.   :2010   Shed   :   2
##
##     RoofMatl     Exterior1st   Exterior2nd    MasVnrType    MasVnrArea
##  CompShg:1434   VinylSd:515   VinylSd:504   BrkCmn : 15   Min.   :   0.0
##  Tar&Grv:  11   HdBoard:222   MetalSd:214   BrkFace:445   1st Qu.:   0.0
##  WdShngl:   6   MetalSd:220   HdBoard:207   None   :864   Median :   0.0
##  WdShake:   5   Wd Sdng:206   Wd Sdng:197   Stone  :128   Mean   : 103.7
##  ClyTile:   1   Plywood:108   Plywood:142   NA's   :  8   3rd Qu.: 166.0
##  Membran:   1   CemntBd: 61   CmentBd: 60                 Max.   :1600.0
##  (Other):   2   (Other):128   (Other):136                 NA's   :8
##  ExterQual ExterCond  Foundation  BsmtQual   BsmtCond    BsmtExposure
##  Ex: 52    Ex:   3   BrkTil:146   Ex  :121   Fa  :  45   Av  :221
##  Fa: 14    Fa:  28   CBlock:634   Fa  : 35   Gd  :  65   Gd  :134
##  Gd:488    Gd: 146   PConc :647   Gd  :618   Po  :   2   Mn  :114
##  TA:906    Po:   1   Slab  : 24   TA  :649   TA  :1311   No  :953
##            TA:1282   Stone :  6   NA's: 37   NA's:  37   NA's: 38
##                      Wood  :  3
##
##  BsmtFinType1   BsmtFinSF1     BsmtFinType2   BsmtFinSF2
##  ALQ :220     Min.   :   0.0   ALQ :  19    Min.   :   0.00
##  BLQ :148     1st Qu.:   0.0   BLQ :  33    1st Qu.:   0.00
##  GLQ :418     Median : 383.5   GLQ :  14    Median :   0.00
##  LwQ : 74     Mean   : 443.6   LwQ :  46    Mean   :  46.55
##  Rec :133     3rd Qu.: 712.2   Rec :  54    3rd Qu.:   0.00
##  Unf :430     Max.   :5644.0   Unf :1256    Max.   :1474.00
##  NA's: 37                      NA's:  38
##    BsmtUnfSF       TotalBsmtSF      Heating     HeatingQC CentralAir
##  Min.   :   0.0   Min.   :   0.0   Floor:   1   Ex:741    N:  95
##  1st Qu.: 223.0   1st Qu.: 795.8   GasA :1428   Fa: 49    Y:1365
##  Median : 477.5   Median : 991.5   GasW :  18   Gd:241
##  Mean   : 567.2   Mean   :1057.4   Grav :   7   Po:  1
##  3rd Qu.: 808.0   3rd Qu.:1298.2   OthW :   2   TA:428
##  Max.   :2336.0   Max.   :6110.0   Wall :   4
##
##  Electrical     X1stFlrSF      X2ndFlrSF     LowQualFinSF
```

```
##  FuseA:  94   Min.   : 334   Min.   :   0   Min.   :  0.000
##  FuseF:  27   1st Qu.: 882   1st Qu.:   0   1st Qu.:  0.000
##  FuseP:   3   Median :1087   Median :   0   Median :  0.000
##  Mix  :   1   Mean   :1163   Mean   : 347   Mean   :  5.845
##  SBrkr:1334   3rd Qu.:1391   3rd Qu.: 728   3rd Qu.:  0.000
##  NA's :   1   Max.   :4692   Max.   :2065   Max.   :572.000
##
##    GrLivArea     BsmtFullBath     BsmtHalfBath       FullBath
##  Min.   : 334   Min.   :0.0000   Min.   :0.00000   Min.   :0.000
##  1st Qu.:1130   1st Qu.:0.0000   1st Qu.:0.00000   1st Qu.:1.000
##  Median :1464   Median :0.0000   Median :0.00000   Median :2.000
##  Mean   :1515   Mean   :0.4253   Mean   :0.05753   Mean   :1.565
##  3rd Qu.:1777   3rd Qu.:1.0000   3rd Qu.:0.00000   3rd Qu.:2.000
##  Max.   :5642   Max.   :3.0000   Max.   :2.00000   Max.   :3.000
##
##     HalfBath       BedroomAbvGr    KitchenAbvGr   KitchenQual
##  Min.   :0.0000   Min.   :0.000   Min.   :0.000   Ex:100
##  1st Qu.:0.0000   1st Qu.:2.000   1st Qu.:1.000   Fa: 39
##  Median :0.0000   Median :3.000   Median :1.000   Gd:586
##  Mean   :0.3829   Mean   :2.866   Mean   :1.047   TA:735
##  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   GarageType
##  Min.   : 2.000   Maj1:  14   Min.   :0.000   Ex  : 24    2Types :  6
##  1st Qu.: 5.000   Maj2:   5   1st Qu.:0.000   Fa  : 33    Attchd :870
##  Median : 6.000   Min1:  31   Median :1.000   Gd  :380    Basment: 19
##  Mean   : 6.518   Min2:  34   Mean   :0.613   Po  : 20    BuiltIn: 88
##  3rd Qu.: 7.000   Mod :  15   3rd Qu.:1.000   TA  :313    CarPort:  9
##  Max.   :14.000   Sev :   1   Max.   :3.000   NA's:690    Detchd :387
##                   Typ :1360                               NA's   : 81
##   GarageYrBlt   GarageFinish   GarageCars      GarageArea     GarageQual
##  Min.   :1900   Fin :352     Min.   :0.000   Min.   :   0.0   Ex  :   3
##  1st Qu.:1961   RFn :422     1st Qu.:1.000   1st Qu.: 334.5   Fa  :  48
##  Median :1980   Unf :605     Median :2.000   Median : 480.0   Gd  :  14
##  Mean   :1979   NA's: 81     Mean   :1.767   Mean   : 473.0   Po  :   3
##  3rd Qu.:2002                3rd Qu.:2.000   3rd Qu.: 576.0   TA  :1311
##  Max.   :2010                Max.   :4.000   Max.   :1418.0   NA's:  81
##  NA's   :81
##  GarageCond  PavedDrive   WoodDeckSF      OpenPorchSF     EnclosedPorch
##  Ex  :   2   N:  90     Min.   :  0.00   Min.   :  0.00   Min.   :  0.00
##  Fa  :  35   P:  30     1st Qu.:  0.00   1st Qu.:  0.00   1st Qu.:  0.00
##  Gd  :   9   Y:1340     Median :  0.00   Median : 25.00   Median :  0.00
##  Po  :   7              Mean   : 94.24   Mean   : 46.66   Mean   : 21.95
##  TA  :1326              3rd Qu.:168.00   3rd Qu.: 68.00   3rd Qu.:  0.00
##  NA's:  81              Max.   :857.00   Max.   :547.00   Max.   :552.00
##
##    X3SsnPorch      ScreenPorch        PoolArea        PoolQC
##  Min.   :  0.00   Min.   :  0.00   Min.   :  0.000   Ex  :   2
##  1st Qu.:  0.00   1st Qu.:  0.00   1st Qu.:  0.000   Fa  :   2
##  Median :  0.00   Median :  0.00   Median :  0.000   Gd  :   3
##  Mean   :  3.41   Mean   : 15.06   Mean   :  2.759   NA's:1453
##  3rd Qu.:  0.00   3rd Qu.:  0.00   3rd Qu.:  0.000
##  Max.   :508.00   Max.   :480.00   Max.   :738.000
```

```
##
##    Fence      MiscFeature    MiscVal           MoSold
##  GdPrv:  59   Gar2:   2   Min.   :    0.00   Min.   : 1.000
##  GdWo :  54   Othr:   2   1st Qu.:    0.00   1st Qu.: 5.000
##  MnPrv: 157   Shed:  49   Median :    0.00   Median : 6.000
##  MnWw :  11   TenC:   1   Mean   :   43.49   Mean   : 6.322
##  NA's :1179   NA's:1406   3rd Qu.:    0.00   3rd Qu.: 8.000
##                           Max.   :15500.00   Max.   :12.000
##
##      YrSold        SaleType    SaleCondition    SalePrice
##  Min.   :2006   WD     :1267   Abnorml: 101   Min.   : 34900
##  1st Qu.:2007   New    : 122   AdjLand:   4   1st Qu.:129975
##  Median :2008   COD    :  43   Alloca :  12   Median :163000
##  Mean   :2008   ConLD  :   9   Family :  20   Mean   :180921
##  3rd Qu.:2009   ConLI  :   5   Normal :1198   3rd Qu.:214000
##  Max.   :2010   ConLw  :   5   Partial: 125   Max.   :755000
##                 (Other):   9
```

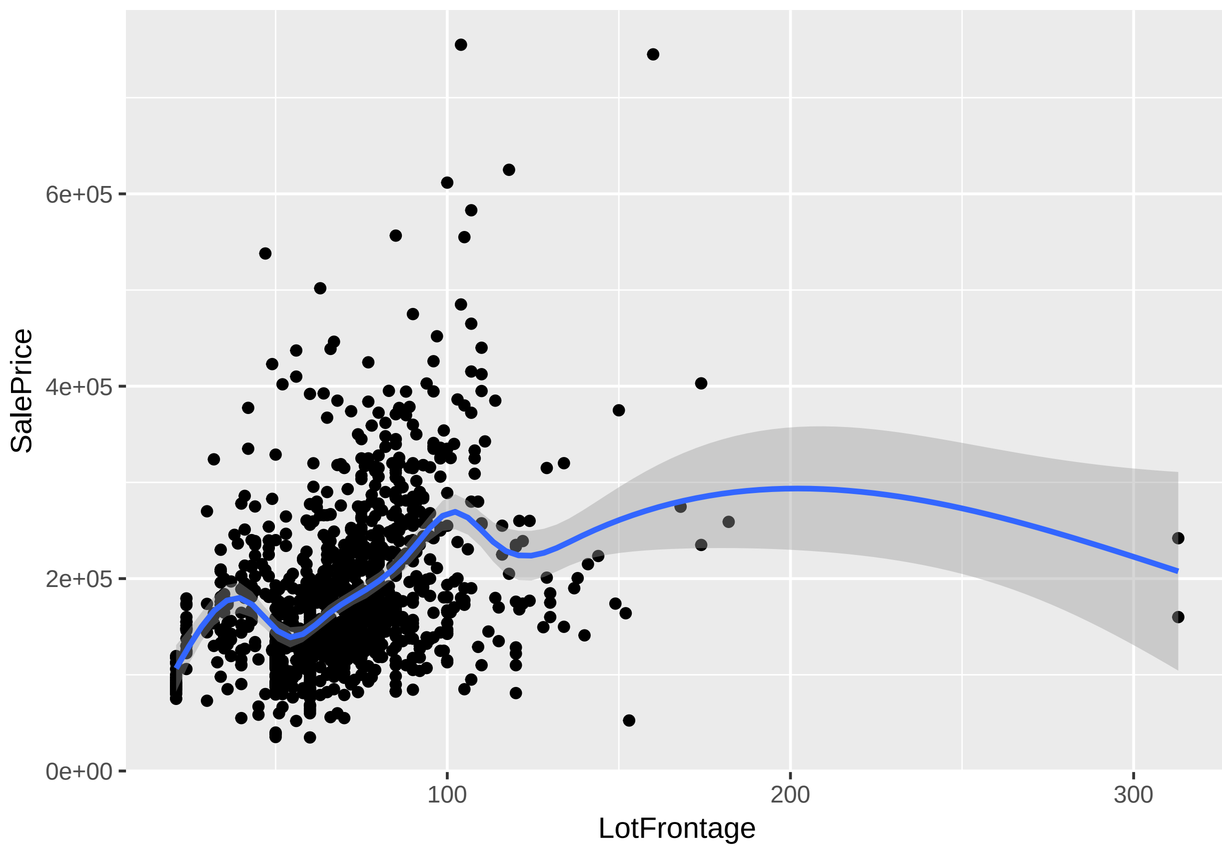
```
## [1] 1459   80
```

```
##  [1] "Id"            "MSSubClass"    "MSZoning"      "LotFrontage"
##  [5] "LotArea"       "Street"        "Alley"         "LotShape"
##  [9] "LandContour"   "Utilities"     "LotConfig"     "LandSlope"
## [13] "Neighborhood"  "Condition1"    "Condition2"    "BldgType"
## [17] "HouseStyle"    "OverallQual"   "OverallCond"   "YearBuilt"
## [21] "YearRemodAdd"  "RoofStyle"     "RoofMatl"      "Exterior1st"
## [25] "Exterior2nd"   "MasVnrType"    "MasVnrArea"    "ExterQual"
## [29] "ExterCond"     "Foundation"    "BsmtQual"      "BsmtCond"
## [33] "BsmtExposure"  "BsmtFinType1"  "BsmtFinSF1"    "BsmtFinType2"
## [37] "BsmtFinSF2"    "BsmtUnfSF"     "TotalBsmtSF"   "Heating"
## [41] "HeatingQC"     "CentralAir"    "Electrical"    "X1stFlrSF"
## [45] "X2ndFlrSF"     "LowQualFinSF"  "GrLivArea"     "BsmtFullBath"
## [49] "BsmtHalfBath"  "FullBath"      "HalfBath"      "BedroomAbvGr"
## [53] "KitchenAbvGr"  "KitchenQual"   "TotRmsAbvGrd"  "Functional"
## [57] "Fireplaces"    "FireplaceQu"   "GarageType"    "GarageYrBlt"
## [61] "GarageFinish"  "GarageCars"    "GarageArea"    "GarageQual"
## [65] "GarageCond"    "PavedDrive"    "WoodDeckSF"    "OpenPorchSF"
## [69] "EnclosedPorch" "X3SsnPorch"    "ScreenPorch"   "PoolArea"
## [73] "PoolQC"        "Fence"         "MiscFeature"   "MiscVal"
## [77] "MoSold"        "YrSold"        "SaleType"      "SaleCondition"
```

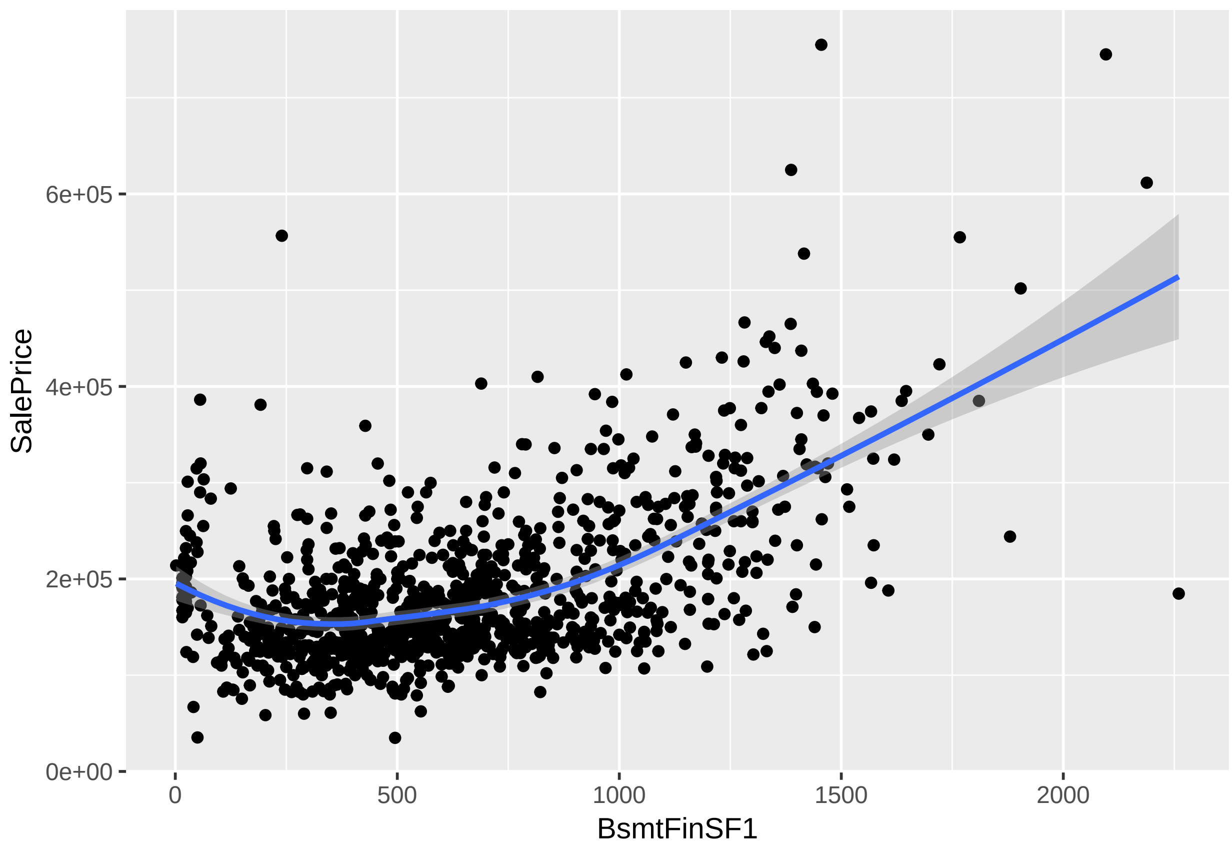
Test dataset does not contain the response variable i.e. SalePrice, which is what we need to predict and submit for evaluation.

Let's look at the data distribution for various variables. I'm including few graphs here as including graphs for all 80 predictor variables will dramatically increase the length of the report.


```
## `geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'
```

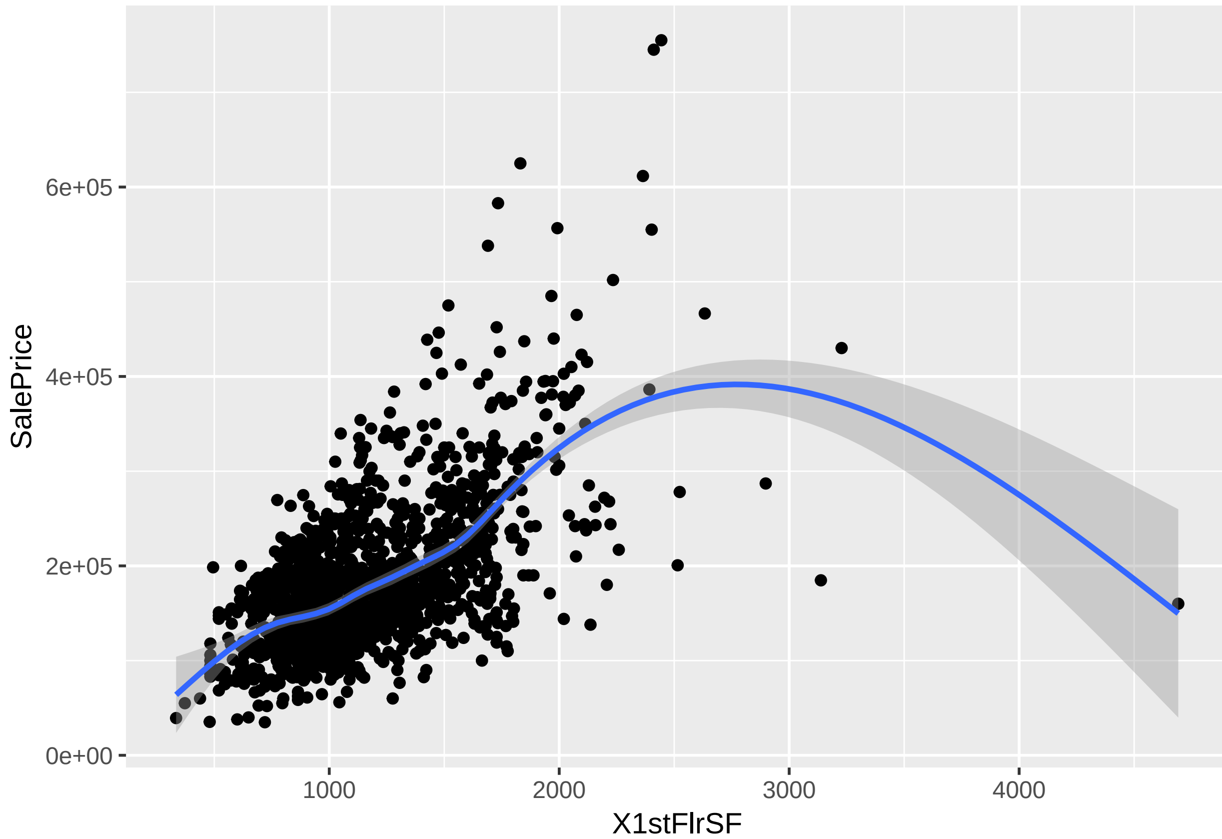



```
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```

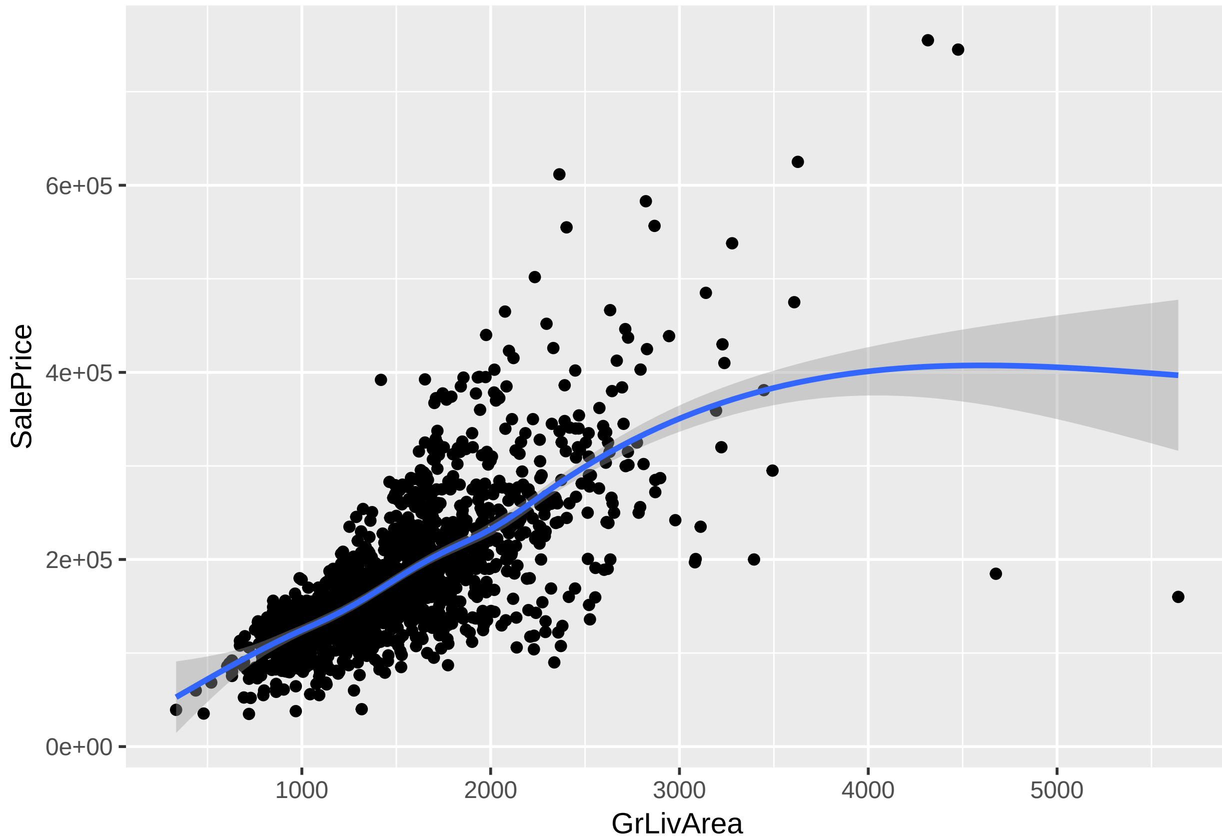


From the following 2 graphs it looks like there is an outlier.

```
## `geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'
```




```
## `geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'
```

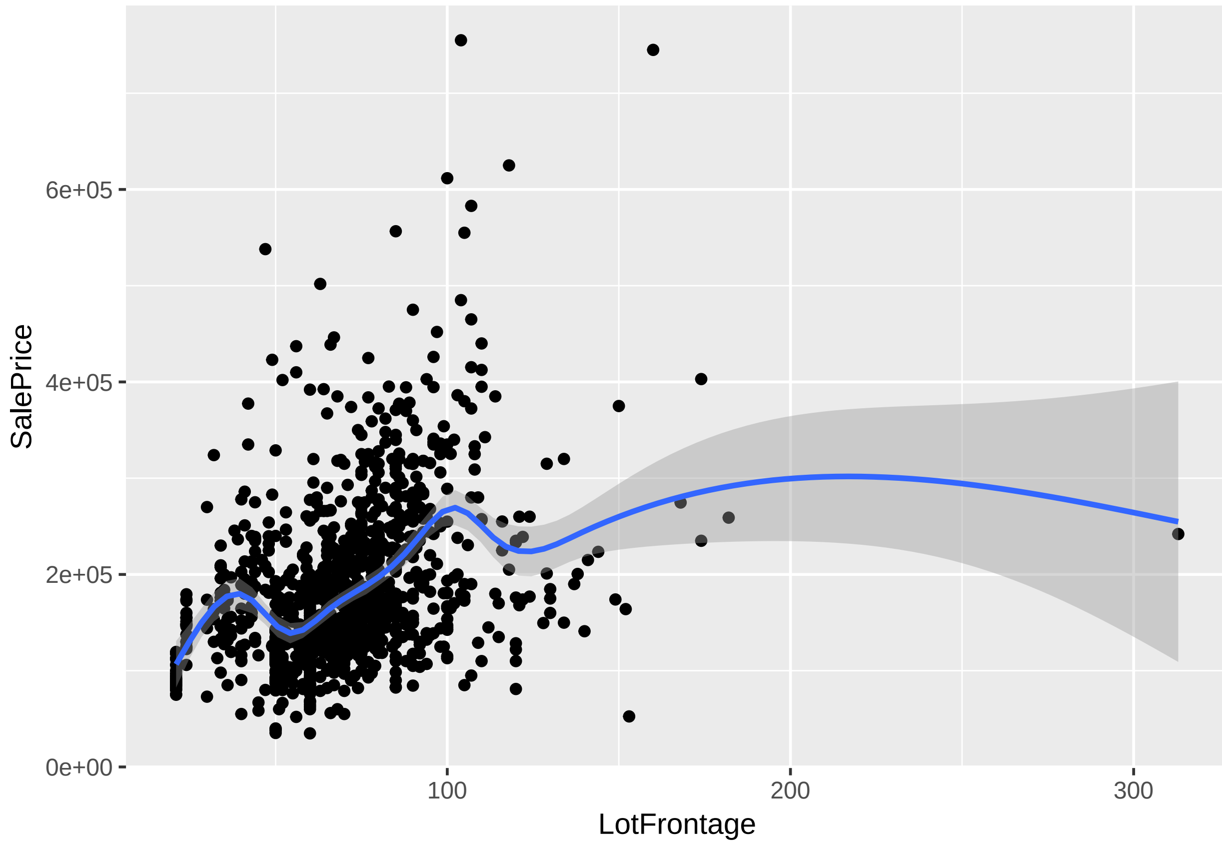


Data wrangling

We will remove the data where GrLivArea is less than 5000 to get rid of the outlier as the presence of an outlier can drastically affect the performance of our machine learning model.

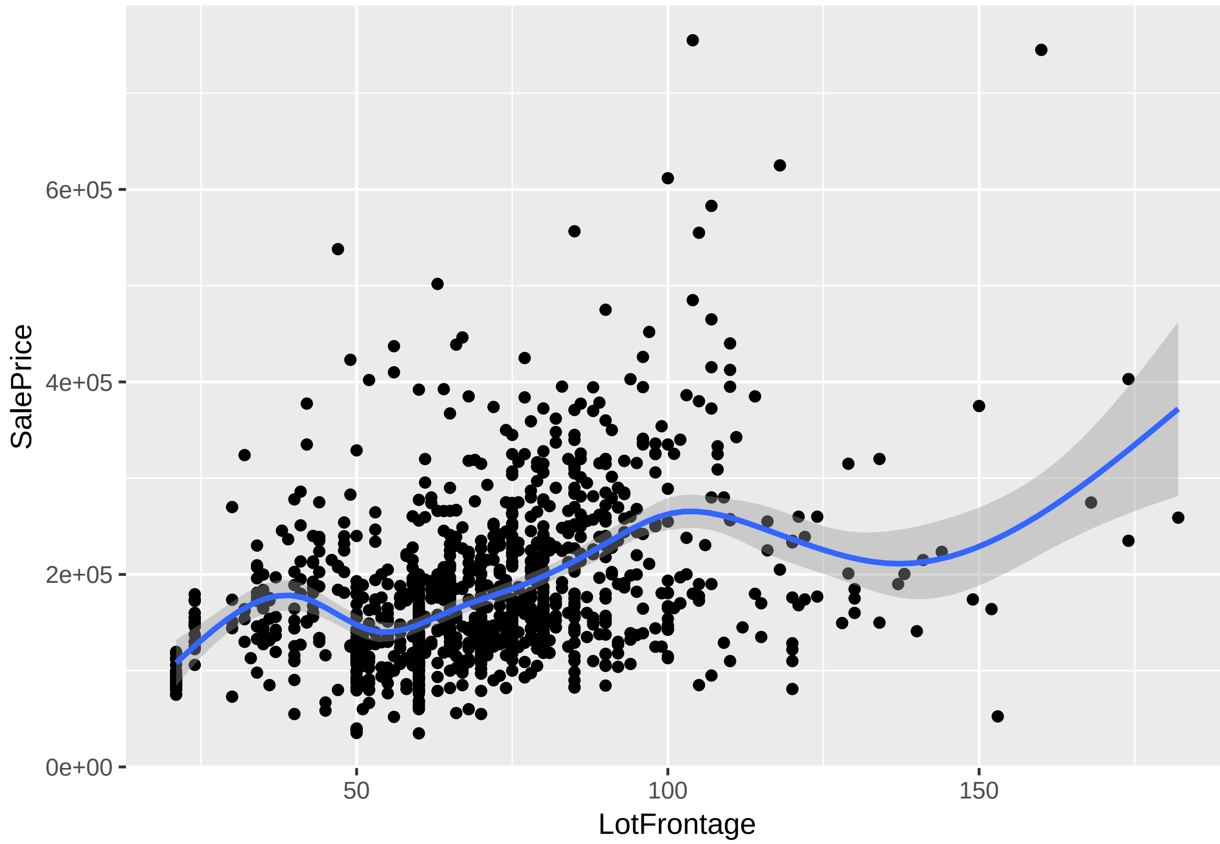
Similarly removing data where LotFrontage is less than 300 to remove the outlier.

```
## `geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'
```



The graphs after removing the outlier:

```
## `geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'
```



It looks a little better now.

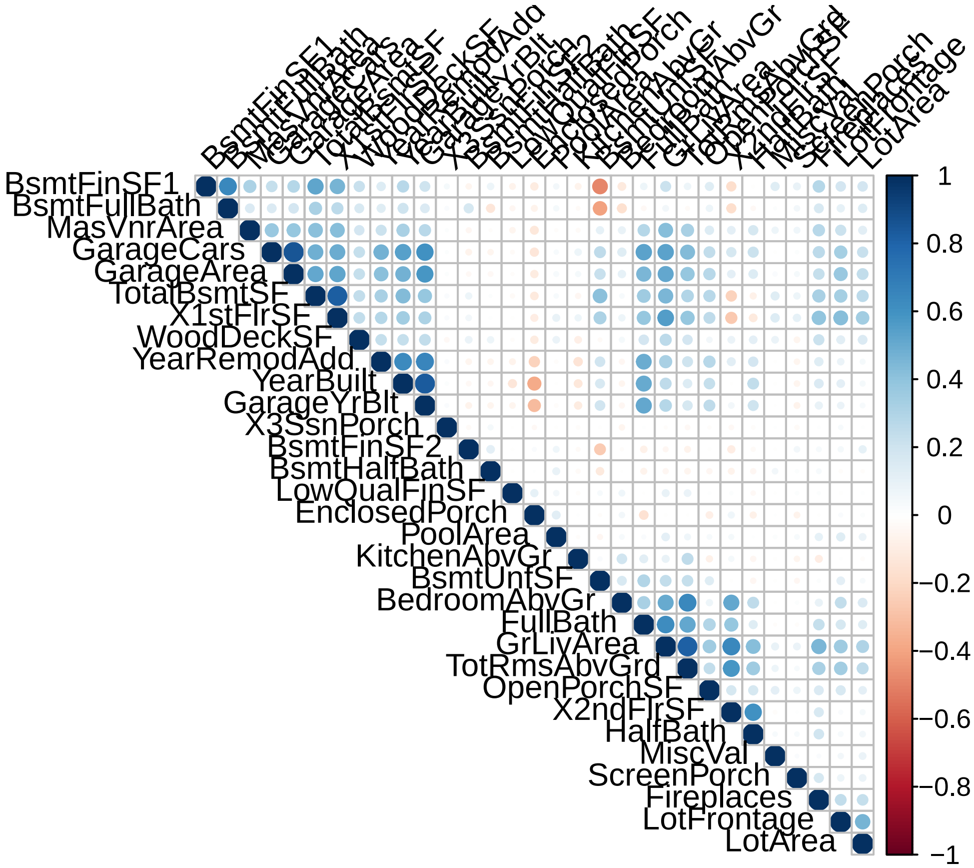
We will merge train and test datasets for easier manipulations auch as: imputing NA values, removing columns with least variance etc.

Separating factor and continuous variables

MSSubClass, OverallQual, OverallCond, MoSold and YrSold are considered as continuous variables while they should be categorical. wSo we Will convert them first before seggregating.

We will separate the continuous and categorical variables.

We need to first take a look at the correlations using correlation plot and heatmap



We can see from above that, the following varibles pairs are highly corralated with each other with correlationcoefficient more than 0.8: GarageArea and GarageCars GrLivArea and TotRmsAbvGrd TotalBsmtSF and X1stFlrSF YearBuilt and GarageYrBlt

Eliminating continuous variables having high correlation with each other

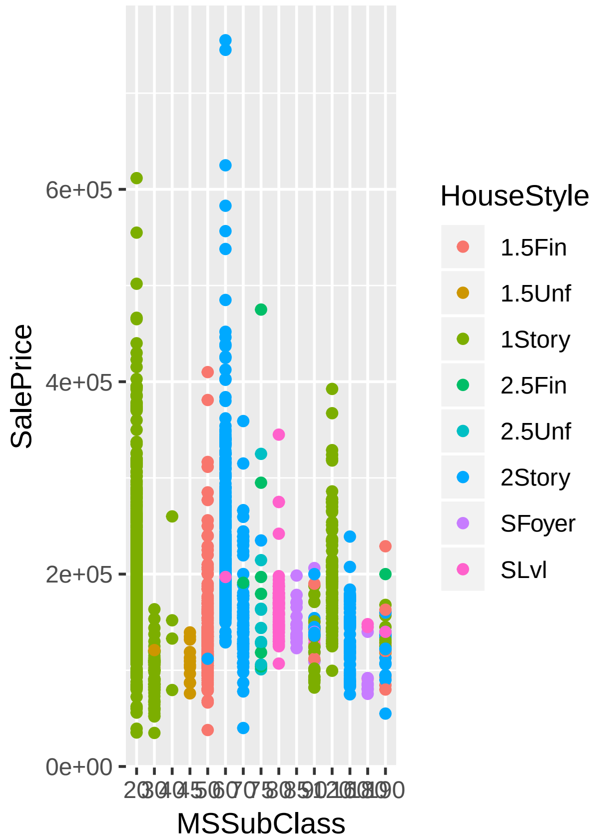
Now we will calculate correlation of each variable with our response variable i.e SalePrice in the Train dataset.

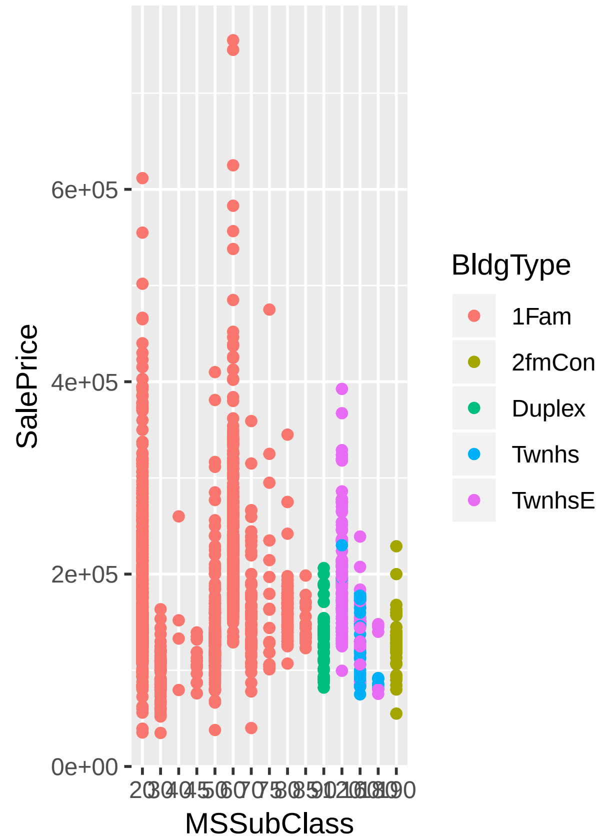
```
##  LotFrontage      LotArea    YearBuilt YearRemodAdd   MasVnrArea
##    0.3762264    0.3072591    0.5264318    0.5215603    0.4931032
##   BsmtFinSF1    BsmtUnfSF  TotalBsmtSF    X1stFlrSF    X2ndFlrSF
##    0.4171590    0.2137868    0.6585724    0.6348055    0.3081626
##    GrLivArea BsmtFullBath     FullBath     HalfBath TotRmsAbvGrd
##    0.7275233    0.2381120    0.5668086    0.2696883    0.5506544
##   Fireplaces  GarageYrBlt   GarageCars   GarageArea   WoodDeckSF
##    0.4657239    0.5060381    0.6471528    0.6272387    0.3379112
##  OpenPorchSF    SalePrice
##    0.3472712    1.0000000
```

Ee will remove the ones having high correlation amonst themselves. We will retain GarageCars, GrLivArea, TotalBsmtSF, YearBuilt because of their higher correlation value with SalePrice and will remove: GarageArea, TotRmsAbvGrd, X1stFlrSF and GarageYrBlt.

Finding redundant factor variables in data

Data in MSSubClass looks like all the information are already present in BldgType and HouseStyle. We will confirm the same after looking at the plots.





From the first and second plots it's clear that each MSSubClass is mostly holding details for HouseStyle and BldgType respectively.

We can remove MSSubClass from the list of predictors.

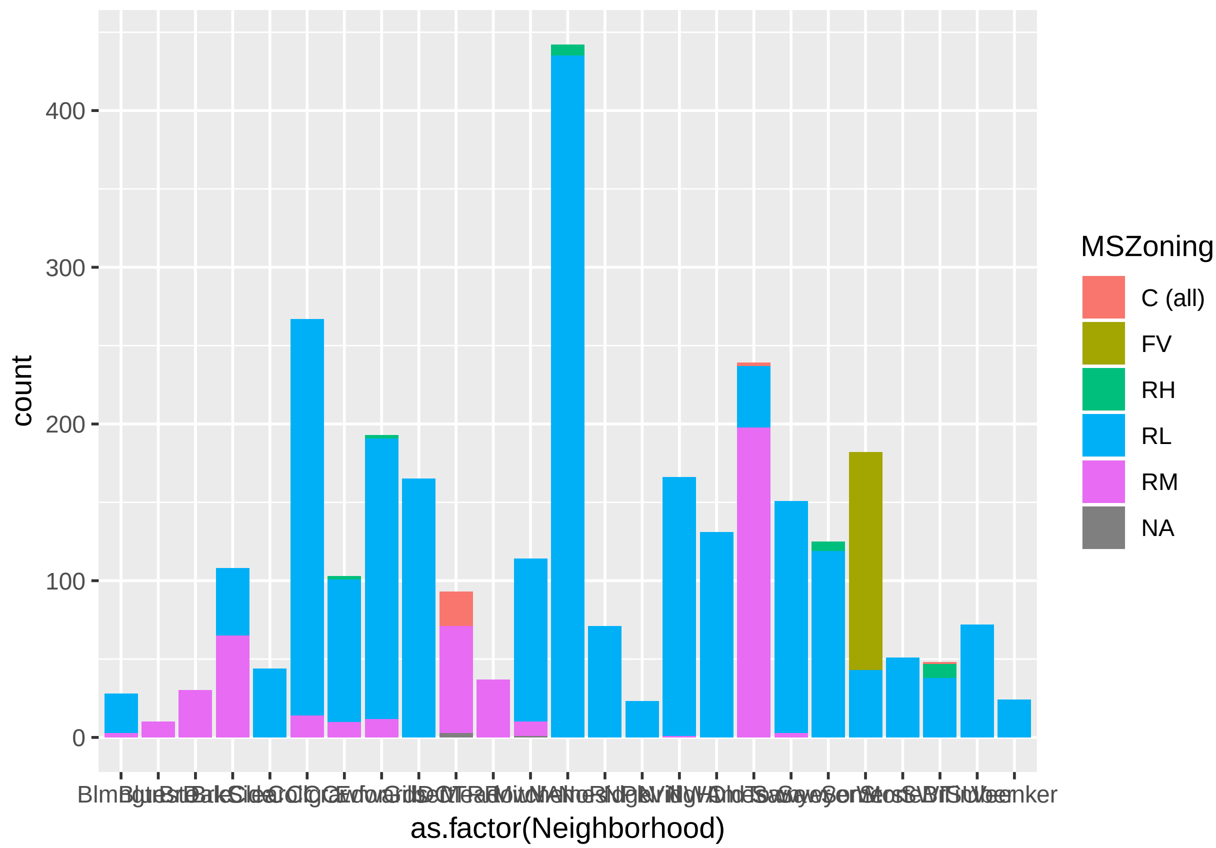
Handling of NA values

Let's look at the column names containing NA values.

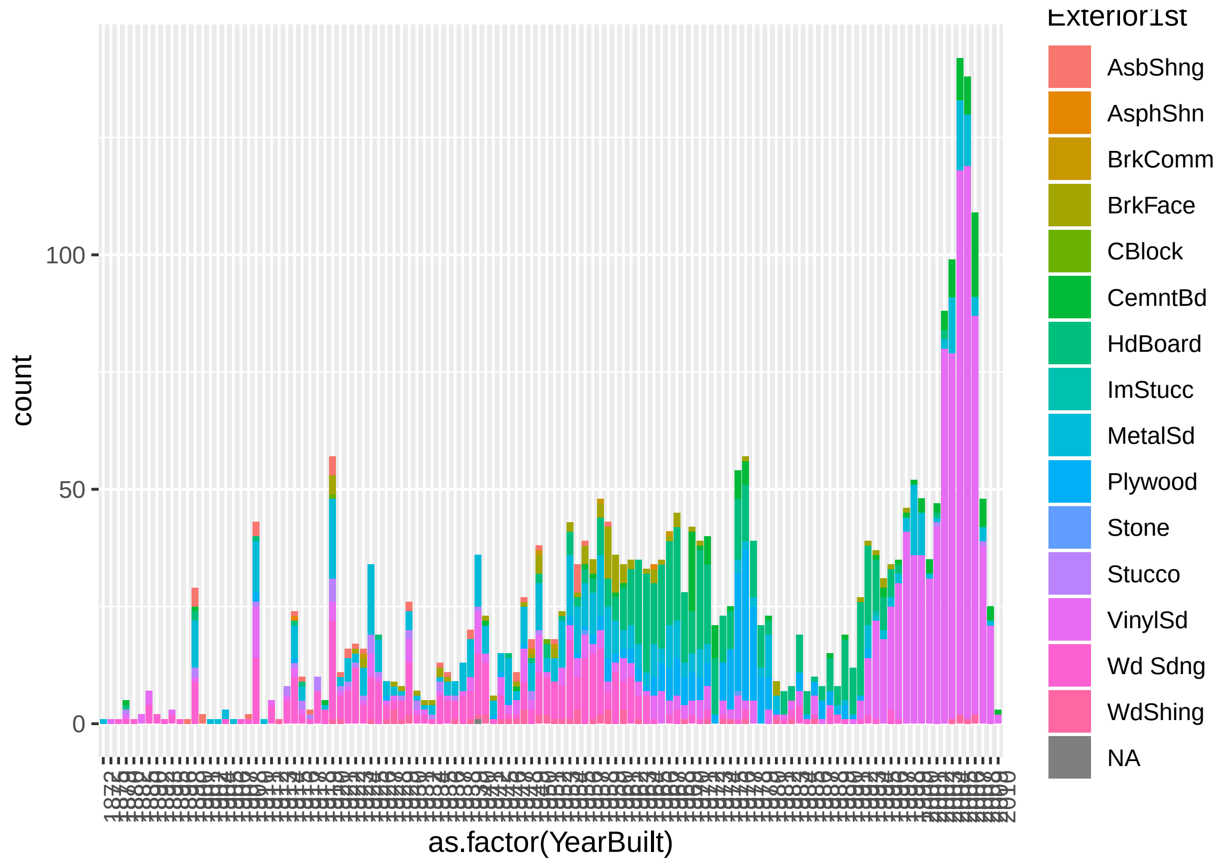
```
##  [1] "MSZoning"     "LotFrontage"  "Alley"        "Utilities"
##  [5] "Exterior1st"  "Exterior2nd"  "MasVnrType"   "MasVnrArea"
##  [9] "BsmtQual"     "BsmtCond"     "BsmtExposure" "BsmtFinType1"
## [13] "BsmtFinSF1"   "BsmtFinType2" "BsmtFinSF2"   "BsmtUnfSF"
## [17] "TotalBsmtSF"  "Electrical"   "BsmtFullBath" "BsmtHalfBath"
## [21] "KitchenQual"  "Functional"   "FireplaceQu"  "GarageType"
## [25] "GarageFinish" "GarageCars"   "GarageQual"   "GarageCond"
## [29] "PoolQC"       "Fence"        "MiscFeature"  "SaleType"
```

From this list, the following variables can not be set to 0/None as it won't make any sense: MSZoning Electrical Exterior1st Exterior2nd GarageCars & GarageFinish (Since GarageType is not "NA") SaleType

So we will have to set a meaningful value for them. For the following variables, the most occurring values have been used to fill the NA values.




```
## # A tibble: 6 x 2
##   Electrical     n
##   <fct>      <int>
## 1 SBrkr       2669
## 2 FuseA        188
## 3 FuseF         50
## 4 FuseP          8
## 5 Mix            1
## 6 <NA>           1
```




```
## # A tibble: 10 x 2
##    SaleType     n
##    <fct>    <int>
##  1 WD        2524
##  2 New        238
##  3 COD         87
##  4 ConLD       26
##  5 CWD         12
##  6 ConLI        9
##  7 ConLw        8
##  8 Oth          7
##  9 Con          5
## 10 <NA>         1
```

For 2 records, the values GarageCars and GarageFinish are NA even though GarageType is not NA, which means Garage is available, but values for these variables are not available.

```
##       BedroomAbvGr KitchenAbvGr
## 11171            3            1
```

```
## # A tibble: 4 x 2
##   GarageFinish     n
##   <fct>        <int>
## 1 Fin             24
## 2 RFn             34
## 3 Unf            719
## 4 <NA>             2
```

As we see for most cases where GarageType="Detchd", the value for GarageFinish="Fin" Hence replacing NA values accordingly.

GarageCars has good correlation with OverallQual, GrLivArea and YearBuilt. Using linear regression with OverallQual, GrLivArea and YearBuilt as parameters to predict the value of missing values for GarageCars.

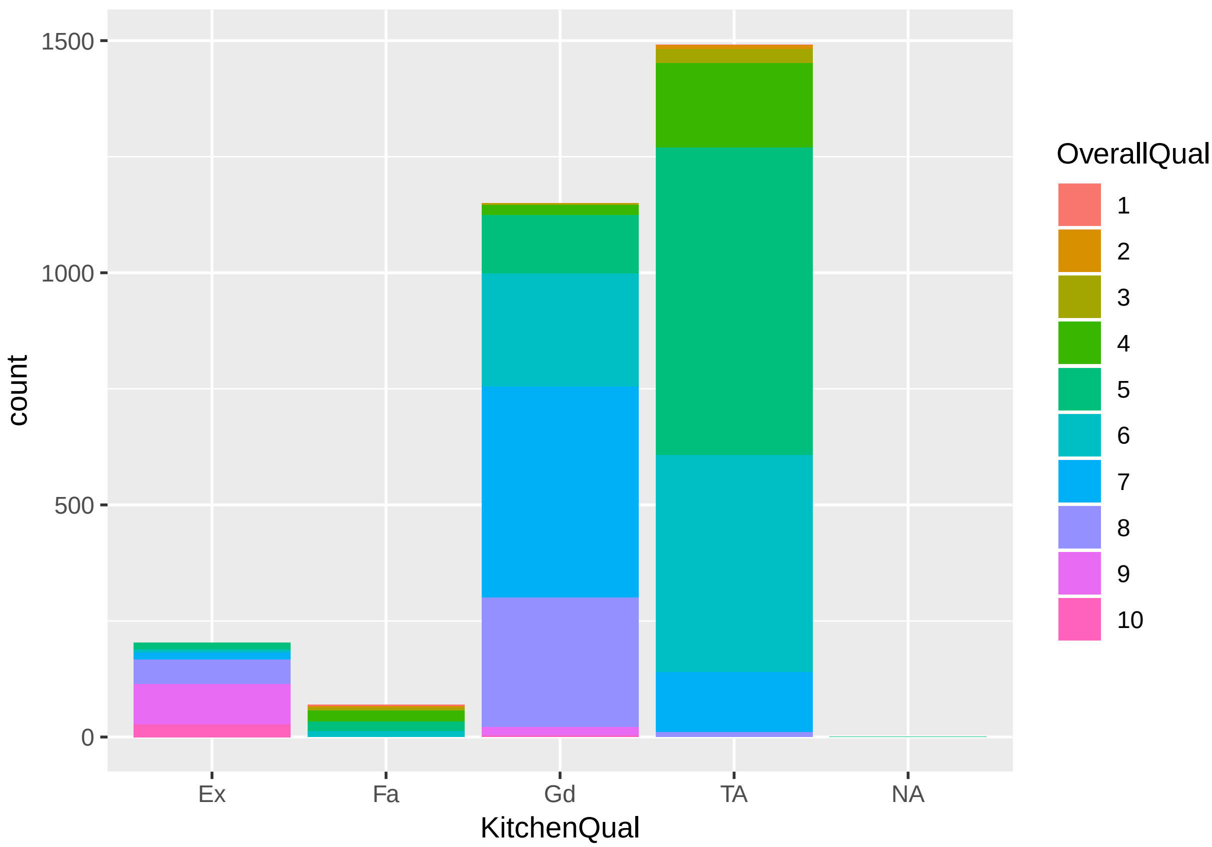
Filling NA values for variables related to basement where there are no basements.

```
##      BsmtQual BsmtFinSF1 BsmtUnfSF
## 6611     <NA>         NA        NA
```

Setting missing values to "None" for categorical variables.

Filling NA values for KitchenQu.

```
##      KitchenAbvGr KitchenQual
## 9610            1        <NA>
```




```
## # A tibble: 5 x 2
##   KitchenQual     n
##   <fct>       <int>
## 1 TA            662
## 2 Gd            126
## 3 Fa             21
## 4 Ex             15
## 5 <NA>            1
```

Maximum value for KitchenQual for the OverallQual as specified in the data where KitchenQual=NA is "TA". Hence we will reaplce the NA values with "TA".

Some variables related to basements have NA (no basement). But the NA values for BsmtQual, BsmtExposure and BsmtFinType1 are inconsistent for some entries. So we need to fill values for those variable where the other 2 variables are not NA.

```
##      BsmtExposure BsmtQual BsmtFinType1
## 949          <NA>       Gd          Unf
## 2810         <NA>       Gd          Unf
## 7581           No     <NA>          Unf
## 7591           No     <NA>          Unf
## 8891         <NA>       Gd          Unf
```

If we look at the maximum occurring value for BsmtExposure while BsmtFinType1=="Unf" and BsmtQual=="Gd", it's "No". We will use this value to fill up the NA values.

```
## # A tibble: 5 x 2
##   BsmtExposure     n
##   <fct>        <int>
## 1 Av              61
## 2 Gd              11
## 3 Mn              23
## 4 No             266
## 5 <NA>             3
```

If we look at the maximum occurring value for BsmtQual while BsmtFinType1=="Unf" and BsmtExposure=="No", it's "TA". We will use this value.

```
## # A tibble: 5 x 2
##   BsmtQual     n
##   <fct>    <int>
## 1 Ex          26
## 2 Fa          54
## 3 Gd         269
## 4 TA         338
## 5 <NA>         2
```

Working on missing values for the remaining basement specific variables.

LotFrontage is NA for many. We cannot set LofFrontage to 0 for properties which have valid vlaues for LotConfig. We will use random forest to impute these values.

```
##   missForest iteration 1 in progress...done!
##   missForest iteration 2 in progress...done!
##   missForest iteration 3 in progress...done!
##   missForest iteration 4 in progress...done!
##   missForest iteration 5 in progress...done!
##   missForest iteration 6 in progress...done!
##   missForest iteration 7 in progress...done!
##   missForest iteration 8 in progress...done!
##   missForest iteration 9 in progress...done!
##   missForest iteration 10 in progress...done!
```

Filling up missing values for MasVnrType and MasVnrArea while considering the inconsistencies (similar to the way basement specific values were filled).

```
## # A tibble: 4 x 2
##   MasVnrType     n
##   <chr>      <int>
## 1 BrkFace      228
## 2 Stone         69
```

```
## 3 BrkCmn         6
## 4 <NA>           1
```

In order to reduce the total number of parameters, we can remove the ones with near zero variance.

Applying machine learning

Now that our data is almost ready we can apply some machine learning algorithms and predict the SalePrice in test dataset. However there are few more steps we need to do to prepare the data for machine learning.

Making data ready before applying machine learning

Standardizing the continuous variables in order to avoid bias by certain variables having wider range of values.

```
total$LotFrontage <- standardize(total$LotFrontage)
total$LotArea <- standardize(total$LotArea)
total$MasVnrArea <- standardize(total$MasVnrArea)
total$BsmtFinSF1 <- standardize(total$BsmtFinSF1)
total$BsmtUnfSF <- standardize(total$BsmtUnfSF)
total$TotalBsmtSF <- standardize(total$TotalBsmtSF)
total$X2ndFlrSF <- standardize(total$X2ndFlrSF)
total$GrLivArea <- standardize(total$GrLivArea)
total$WoodDeckSF <- standardize(total$WoodDeckSF)
total$OpenPorchSF <- standardize(total$OpenPorchSF)
```

Label encoding few factor variables

Since many machine algorithms only understand numeric inputs, converting factor variables to numeric is essential. Variables with qualitative values ranging from “Excellent” to “Poor” can easily be converted to numeric values which will make them look like continuous variables.

```
# Converting the levels of all quality factor variables to 1-5
levels <- c("Po","Fa","TA","Gd","Ex")
total$ExterQual <- as.numeric(factor(total$ExterQual, levels = levels))
total$ExterCond <- as.numeric(factor(total$ExterCond, levels = levels))
total$BsmtQual <- as.numeric(factor(total$BsmtQual, levels = levels))
total$BsmtQual[is.na(total$BsmtQual)] <- 0
total$KitchenQual <- as.numeric(factor(total$KitchenQual, levels = levels))
total$FireplaceQu <- as.numeric(factor(total$FireplaceQu, levels = levels))
total$FireplaceQu[is.na(total$FireplaceQu)] <- 0
total$HeatingQC <- as.numeric(factor(total$HeatingQC, levels = levels))
total$OverallQual <- as.numeric(total$OverallQual)
total$OverallCond <- as.numeric(total$OverallCond)

# CentralAir has 2 levels, so making them binary
total$CentralAir <- as.numeric(factor(total$CentralAir, levels = c("N","Y")))-1

total$LotShape <- as.numeric(factor(total$LotShape, levels = c("IR3","IR2","IR1","Reg")))

total$BsmtExposure <- as.numeric(factor(total$BsmtExposure, levels = c("No","Mn","Av","Gd")))
total$BsmtExposure[is.na(total$BsmtExposure)] <- 0

levels <- c("Unf","LwQ","Rec","BLQ","ALQ","GLQ")
total$BsmtFinType1 <- as.numeric(factor(total$BsmtFinType1, levels = levels))
total$BsmtFinType1[is.na(total$BsmtFinType1)] <- 0
```

```
total$GarageFinish <- as.numeric(factor(total$GarageFinish, levels = c("Unf","RFn","Fin")))
total$GarageFinish[is.na(total$GarageFinish)] <- 0
```

One Hot encoding of factor variables

We will be converting the other kinds of categorical variables into numeric using one-hot encoding as it's easier for machine learning algorithms to interpret that way.

```
# Listing the categorical variables
nums <- sapply(total, is.numeric)
categorical <- total[,!nums]

# One hot encoding
total <- data.frame(total)
total <- dummy.data.frame(total, names=colnames(categorical),sep="_")
```

Now we will divide the dataset back into train and test datasets.

```
set.seed(1)
inTest <- which(total$Id >= 1461)
train <- total[-inTest,]
test <- total[inTest,]

# Adding SalePrice back in train data set
train <- cbind(train,SalePrice)
```

Data partitioning for train and cross validating

We will divide the train dataset into training and testing sets for testing various machine learning algorithms for accuracy before applying on the final test dataset.

```
#Removing Id before splitting
train <- train[,-which(names(train) %in% c("Id"))]

# Divide this train dataset into training and testing datasets
set.seed(1)
inTrain <- createDataPartition(train$SalePrice, p=0.7, list=FALSE)

training <- train[inTrain,]
testing <- train[-inTrain,]
```

Machine Learning Algorithms

We will consider a variety of machine learning algorithms and compare the RMSE for all of them and choose the one with the minimum RMSE.

We will start with naive RMSE i.e. using just the average SalePrice as the expected value.

```
naive_rmse <- RMSE(log(mean(training$SalePrice)),log(testing$SalePrice))
RMSE_table <- data_frame(method = "Just the average", RMSE = naive_rmse)

# Actual price difference (without taking log)
naive_price_diff <- RMSE(mean(training$SalePrice),testing$SalePrice)

# Print RMSE table
print(RMSE_table)
```

```
## # A tibble: 1 x 2
##   method            RMSE
##   <chr>            <dbl>
## 1 Just the average 0.391
```

The RMSE value of 0.391 means the difference in actual and predicted house price is around $‘r naive_price_diff’, which is obviously not very good.

Linear Regression

Let’s apply the simplest machine learning algorithm “Linear Regression” and look at the RMSE value.

```
# The model
fit1 <- lm(SalePrice~.,training)

# Predicting in testing dataset
yhat1 <- predict(fit1, newdata=testing)

# RMSE
rmse_lm <- RMSE(log(yhat1), log(testing$SalePrice))
RMSE_table <- rbind(RMSE_table, data.frame(method = "Linear Regression", RMSE = rmse_lm))

# Print RMSE table
print(RMSE_table)
```

```
## # A tibble: 2 x 2
##   method             RMSE
##   <chr>             <dbl>
## 1 Just the average  0.391
## 2 Linear Regression 0.171
```

There is definitely a huge improvement on the RMSE value. However surely we can do better.

Random forest

Random forest is one of the models known for giving good accurcay. Let’s look at the performance with default parameters.

```
set.seed(1)
rfModel <- train(SalePrice ~ ., method = "rf", data = training)
yhat_rf <- predict(rfModel, newdata=testing)
RMSE_rf <- RMSE(log(yhat_rf),log(testing$SalePrice))

#Storing results in RMSE table
RMSE_table <- rbind(RMSE_table, data.frame(method = "Random Forest(default values)", RMSE = RMSE_rf))

# Print RMSE table
print(RMSE_table)
```

```
## # A tibble: 3 x 2
##   method                         RMSE
##   <chr>                         <dbl>
## 1 Just the average              0.391
## 2 Linear Regression             0.171
## 3 Random Forest(default values) 0.128
```

Moch better than linear regression. But we can still do better.

GBM

Gradient Boosting Machines is another popular model for achieving great accuracy like Random Forest model.

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

set.seed(1)
gbm.caret <- train(SalePrice ~ .,
                   data=training,
                   distribution="gaussian",
                   method="gbm",
                   trControl=trainControl,
                   verbose=FALSE,
                   metric="RMSE",
                   bag.fraction=0.8
                   )

print(gbm.caret)
```

```
## Stochastic Gradient Boosting
##
## 1023 samples
##  188 predictor
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 921, 920, 922, 920, 922, 920, ...
## Resampling results across tuning parameters:
##
##   interaction.depth  n.trees  RMSE      Rsquared   MAE
##   1                   50      34219.20  0.8366881  22886.13
##   1                  100      30521.71  0.8593912  19792.99
##   1                  150      29731.06  0.8657554  18935.06
##   2                   50      30835.05  0.8585480  19929.23
##   2                  100      29444.18  0.8679568  18422.72
##   2                  150      29388.35  0.8676452  17803.00
##   3                   50      30025.00  0.8618657  18945.65
##   3                  100      29014.07  0.8700740  17706.75
##   3                  150      29172.47  0.8687165  17406.25
##
## Tuning parameter 'shrinkage' was held constant at a value of 0.1
##
## Tuning parameter 'n.minobsinnode' was held constant at a value of 10
## RMSE was used to select the optimal model using the smallest value.
## The final values used for the model were n.trees = 100,
##  interaction.depth = 3, shrinkage = 0.1 and n.minobsinnode = 10.
```

```
# Prediction
predict_gbm <- predict(gbm.caret, newdata=testing)

rmse_gbm <- RMSE(log(testing$SalePrice), log(predict_gbm))

#Storing results in RMSE table
RMSE_table <- rbind(RMSE_table, data.frame(method = "GBM (default values)", RMSE = rmse_gbm))

# print results
```

```r
print(RMSE_table)
```

```
## # A tibble: 4 x 2
##   method                         RMSE
##   <chr>                         <dbl>
## 1 Just the average              0.391
## 2 Linear Regression             0.171
## 3 Random Forest(default values) 0.128
## 4 GBM (default values)          0.126
```

If we look at all the RMSE values obtained from various models, we observe that GBM has given us the maximum accuracy i.e. least RMSE. We will tune this model and find the best parameters to minimize the RMSE.

```r
#Hyper tuning
# Setting parameters
hyper_grid <- expand.grid(
    shrinkage = c(0.01, 0.1, 0.3),
    interaction.depth = c(1, 3, 5),
    n.minobsinnode = c(5, 10, 15),
    bag.fraction = c(.65, .8, 1),
    optimal_trees = 0,               # a place to dump results
    min_RMSE = 0                     # a place to dump results
)

# total number of combinations
nrow(hyper_grid)
```

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

```r
# Grid search
for(i in 1:nrow(hyper_grid)) {

    # reproducibility
    set.seed(1)

    # train model
    gbm.tune <- gbm(
        formula = SalePrice ~ .,
        distribution = "gaussian",
        data = training,
        n.trees = 5000,
        interaction.depth = hyper_grid$interaction.depth[i],
        shrinkage = hyper_grid$shrinkage[i],
        n.minobsinnode = hyper_grid$n.minobsinnode[i],
        bag.fraction = hyper_grid$bag.fraction[i],
        train.fraction = .75,
        n.cores = NULL, # will use all cores by default
        verbose = FALSE
    )

    # add min training error and trees to grid
    hyper_grid$optimal_trees[i] <- which.min(gbm.tune$valid.error)
    hyper_grid$min_RMSE[i] <- sqrt(min(gbm.tune$valid.error))
}
```

```r
hyper_grid %>%
    dplyr::arrange(min_RMSE) %>%
    head(10)
```

```
##    shrinkage interaction.depth n.minobsinnode bag.fraction optimal_trees
## 1       0.10                 5             10         1.00           406
## 2       0.10                 5              5         0.80           267
## 3       0.01                 5              5         0.65          1831
## 4       0.01                 5             10         0.80          1872
## 5       0.10                 5             10         0.80           271
## 6       0.10                 5             15         1.00           163
## 7       0.10                 3              5         0.65           434
## 8       0.10                 1             10         1.00          1308
## 9       0.10                 3             15         0.65           145
## 10      0.01                 5             15         0.65          1130
##    min_RMSE
## 1  24935.06
## 2  24948.21
## 3  25070.29
## 4  25258.84
## 5  25358.85
## 6  25456.73
## 7  25466.37
## 8  25560.61
## 9  25570.24
## 10 25659.72
```

These results help us to zoom into areas where we can refine our search. Let's adjust our grid and zoom into closer regions of the values that appear to produce the best results in our previous grid search. Optimal trees aren't crossing 2000, so the model isn't requiring too many trees. This grid contains 81 combinations that we'll search across.

```r
# modify hyperparameter grid
hyper_grid2 <- expand.grid(
  shrinkage = c(.01, .05, .1),
  interaction.depth = c(3, 5, 7),
  n.minobsinnode = c(5, 10, 15),
  bag.fraction = c(.65, .8, 1),
  optimal_trees = 0,               # a place to dump results
  min_RMSE = 0                     # a place to dump results
)

# total number of combinations
nrow(hyper_grid2)
```

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

```r
set.seed(1)
# Grid search
for(i in 1:nrow(hyper_grid2)) {

    # reproducibility
    set.seed(1)

    # train model
```

```r
    gbm.tune2 <- gbm(
        formula = SalePrice ~ .,
        distribution = "gaussian",
        data = training,
        n.trees = 2000,
        interaction.depth = hyper_grid2$interaction.depth[i],
        shrinkage = hyper_grid2$shrinkage[i],
        n.minobsinnode = hyper_grid2$n.minobsinnode[i],
        bag.fraction = hyper_grid2$bag.fraction[i],
        train.fraction = .75,
        n.cores = NULL, # will use all cores by default
        verbose = FALSE
    )

    # add min training error and trees to grid
    hyper_grid2$optimal_trees[i] <- which.min(gbm.tune2$valid.error)
    hyper_grid2$min_RMSE[i] <- sqrt(min(gbm.tune2$valid.error))
}

hyper_grid2 %>%
    dplyr::arrange(min_RMSE) %>%
    head(10)
```

```
##    shrinkage interaction.depth n.minobsinnode bag.fraction optimal_trees
## 1       0.05                 7              5         0.65           198
## 2       0.05                 7             10         0.80           393
## 3       0.05                 7             10         0.65           232
## 4       0.01                 7             10         0.80          1859
## 5       0.10                 5             10         1.00           406
## 6       0.10                 5              5         0.80           267
## 7       0.01                 7              5         0.65          1881
## 8       0.05                 5              5         0.65           431
## 9       0.01                 5              5         0.65          1831
## 10      0.05                 7             10         1.00           379
##    min_RMSE
## 1  24417.05
## 2  24725.09
## 3  24826.71
## 4  24915.70
## 5  24935.06
## 6  24948.21
## 7  25036.69
## 8  25050.99
## 9  25070.29
## 10 25173.12
```

Once we have found our top model we can train a model with those specific parameters. And since the model converged at 198 trees we train a cross validated model with 198 trees.

```r
set.seed(1)

# train GBM model
gbm.fit.final <- gbm(
     formula = SalePrice ~ .,
     distribution = "gaussian",
```

```r
    data = training,
    n.trees = 198,
    interaction.depth = 7,
    shrinkage = 0.05,
    n.minobsinnode = 5,
    bag.fraction = 0.65,
    train.fraction = 1,
    n.cores = NULL, # will use all cores by default
    verbose = FALSE
)
```

Variable importance

Let's look at the variable importance as per final GBM model.

Now it's time to predict the values in the testing dataset and observe the modified RMSE after hypertuning. Look at the RMSE table below.

```r
# predict values for test data
yhat <- predict(gbm.fit.final, newdata = testing, n.trees = gbm.fit.final$n.trees)

# results
rmse_gbm_final <- RMSE(log(yhat),log(testing$SalePrice))

RMSE_table <- rbind(RMSE_table, data.frame(method = "GBM (after hypertuning)", RMSE = rmse_gbm_final))

# print results
print(RMSE_table)
```

```
## # A tibble: 5 x 2
##   method                          RMSE
##   <chr>                          <dbl>
## 1 Just the average               0.391
## 2 Linear Regression              0.172
## 3 Random Forest(default values)  0.128
## 4 GBM (default values)           0.123
## 5 GBM (after hypertuning)        0.112
```

Final prediction

Now that out model is ready, we will use it to train our train dataset and then finally predict on the test dataset and submit on Kaggle.

```r
# Final prediction model for submission
set.seed(1)
gbm.fit.final <- gbm(
    formula = SalePrice ~ .,
    distribution = "gaussian",
    data = train,
    n.trees = 198,
    interaction.depth = 7,
    shrinkage = 0.05,
    n.minobsinnode = 5,
    bag.fraction = .65,
    train.fraction = 1,
    n.cores = NULL, # will use all cores by default
```

```
    verbose = FALSE
)

# Prediction on test dataset
gbm_predict <- predict(gbm.fit.final, test, n.trees=gbm.fit.final$n.trees)

# Combining Id and SalePrice
final_test <- cbind(test$Id,gbm_predict)
colnames(final_test) <- c("Id","SalePrice")

#Writing to csv file
write.csv(final_test,"final.csv", row.names=FALSE)
```

Conclusion

After submitting the above final.csv file on Kaggle, the best score given by Kaggle was 0.13002, which gave a ranking of 1822. This final score from Kaggle may not be this exact score, but may vary slightly with each submission. The best entry on Kaggle was 0.09949. This model undoubtedly has a lot of scope for improvement such as: more feature engineering and trying out models such as XGBoost, which can further reduce the RMSE value. However I was getting better prediction with GBM, which is why I did not include XGBoost. I need to explore more on that area.

Thanks for reviewing the report.