NBA6921 Project Team8

Yuanhong Cao, Annabelle Gu, Yihui Guo

11/13/2021

```r
rm(list=ls())
options(digits = 3, scipen = 999)
library(tidyverse)
library(ISLR)
library(jtools)
library(caret)
library(ROCR)
library(glmnet)
library(ggcorrplot)
library(cowplot)
library(lmtest)
library(corrr)
library(dplyr)
library(ggplot2)
library(caret)
library(e1071)
library(MASS)
library(leaps)
library(randomForest)
library(rpart)
library(ranger)
library(gbm)
set.seed(2)
train <- read.csv("train.csv")
```

```r
Score <- matrix(, nrow = 4, ncol = 1)
rownames(Score) <- c("Linear Regression", "Random Forest", "Elastic Net", "Boosting")
colnames(Score) <- c("RMSE")

Score
```

```
                  RMSE
Linear Regression   NA
Random Forest       NA
Elastic Net         NA
Boosting            NA
```

```r
train <- train[,!colnames(train) %in% c('Id','MSZoning','Street',
                                        'Alley', 'LotShape', 'LandContour',
                                        'Utilities', 'LotConfig', 'LandSlope',
                                        'Neighborhood', 'Condition1',
```

```
                                                'Condition2', 'BldgType', 'HouseStyle',
                                                'RoofStyle', 'RoofMatl', 'Exterior1st',
                                                'Exterior2nd', 'MasVnrType',
                                                'ExterQual','ExterCond', 'Foundation',
                                                'BsmtQual', 'BsmtCond', 'BsmtExposure',
                                                'BsmtFinType1', 'BsmtFinType2',
                                                'Heating', 'HeatingQC', 'CentralAir',
                                                'Electrical', 'KitchenQual',
                                                'Functional', 'FireplaceQu',
                                                'GarageType', 'GarageFinish',
                                                'GarageQual', 'GarageCond',
                                                'PavedDrive', 'PoolQC' ,'Fence',
                                                'MiscFeature', 'SaleType',
                                                'SaleCondition', 'MasVnrArea')]
```

```
colSums(is.na(train))
```

```
   MSSubClass   LotFrontage       LotArea   OverallQual   OverallCond
            0           259             0             0             0
    YearBuilt  YearRemodAdd    BsmtFinSF1    BsmtFinSF2     BsmtUnfSF
            0             0             0             0             0
  TotalBsmtSF     X1stFlrSF     X2ndFlrSF  LowQualFinSF     GrLivArea
            0             0             0             0             0
 BsmtFullBath  BsmtHalfBath      FullBath      HalfBath  BedroomAbvGr
            0             0             0             0             0
 KitchenAbvGr  TotRmsAbvGrd    Fireplaces   GarageYrBlt    GarageCars
            0             0             0            81             0
   GarageArea    WoodDeckSF   OpenPorchSF EnclosedPorch    X3SsnPorch
            0             0             0             0             0
  ScreenPorch      PoolArea       MiscVal        MoSold        YrSold
            0             0             0             0             0
    SalePrice
            0
```

```
train$LotFrontage[is.na(train$LotFrontage)] <- median(train$LotFrontage,
                                                      na.rm=TRUE)
train$GarageYrBlt[is.na(train$GarageYrBlt)] <- median(train$GarageYrBlt,
                                                      na.rm=TRUE)
```

```
colSums(is.na(train))
```

```
   MSSubClass   LotFrontage       LotArea   OverallQual   OverallCond
            0             0             0             0             0
    YearBuilt  YearRemodAdd    BsmtFinSF1    BsmtFinSF2     BsmtUnfSF
            0             0             0             0             0
  TotalBsmtSF     X1stFlrSF     X2ndFlrSF  LowQualFinSF     GrLivArea
            0             0             0             0             0
 BsmtFullBath  BsmtHalfBath      FullBath      HalfBath  BedroomAbvGr
            0             0             0             0             0
 KitchenAbvGr  TotRmsAbvGrd    Fireplaces   GarageYrBlt    GarageCars
            0             0             0             0             0
   GarageArea    WoodDeckSF   OpenPorchSF EnclosedPorch    X3SsnPorch
```

```
          0            0            0            0            0
  ScreenPorch     PoolArea      MiscVal       MoSold       YrSold
            0            0            0            0            0
    SalePrice
            0
```

```
train_ind <- sample(1:nrow(train),4/5*nrow(train))
house_train <- train[train_ind,]
house_test <- train[-train_ind,]
```

Variation

```
range(house_train$SalePrice, na.rm = TRUE)
```

```
[1]  35311 625000
```

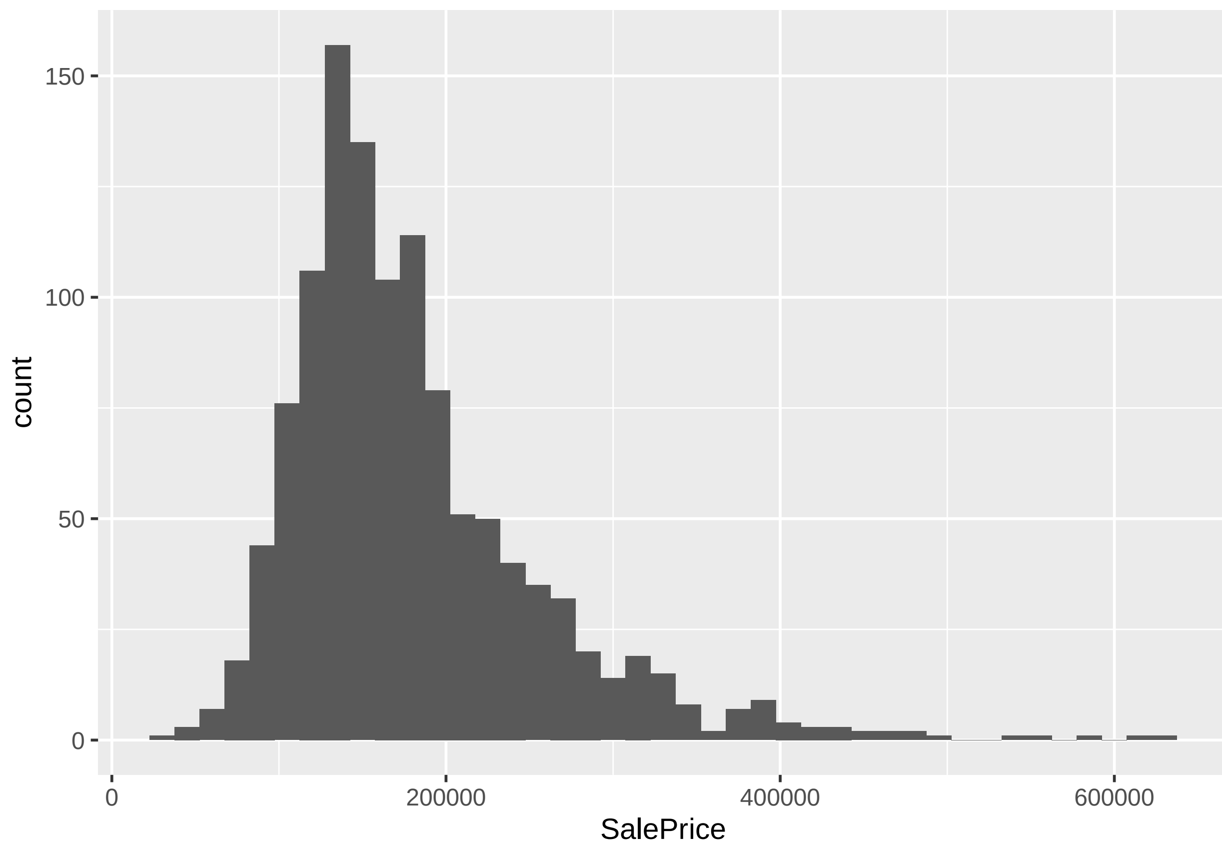
```
quantile(house_train$SalePrice, na.rm = TRUE)
```

```
    0%    25%    50%    75%   100%
 35311 130875 162700 213500 625000
```

```
quantile(house_train$SalePrice,
        probs = seq(from = 0, to = 1, by = .1),
        na.rm = TRUE)
```

```
    0%    10%    20%    30%    40%    50%    60%    70%    80%    90%   100%
 35311 109000 125000 137000 147000 162700 179000 196950 229074 277150 625000
```

```
ggplot(data = house_train) +
  geom_histogram(mapping = aes(x = SalePrice), binwidth = 15000)
```




```
minprice <- min(house_test$SalePrice)
maxprice <- max(house_test$SalePrice)



house_train$SalePrice = log(house_train$SalePrice)
house_test$SalePrice = log(house_test$SalePrice)
train$SalePrice = log(train$SalePrice)
```

```
str(house_train)
```

```
'data.frame':   1168 obs. of  36 variables:
 $ MSSubClass   : int  70 20 20 20 60 60 60 20 50 20 ...
 $ LotFrontage  : int  60 69 70 73 71 92 75 72 60 70 ...
 $ LotArea      : int  11414 7162 10150 8899 12209 11764 9750 8872 10410 8400 ...
 $ OverallQual  : int  7 5 5 7 6 8 7 5 3 6 ...
 $ OverallCond  : int  8 7 5 5 5 7 6 8 4 3 ...
 $ YearBuilt    : int  1910 1966 1958 2007 2001 1999 1998 1965 1915 1957 ...
 $ YearRemodAdd : int  1993 1966 1958 2007 2002 2007 1998 2008 1950 1957 ...
 $ BsmtFinSF1   : int  0 0 456 24 690 524 975 595 0 189 ...
 $ BsmtFinSF2   : int  0 0 0 0 0 0 0 0 0 661 ...
 $ BsmtUnfSF    : int  728 876 456 1316 114 628 133 317 672 628 ...
 $ TotalBsmtSF  : int  728 876 912 1340 804 1152 1108 912 672 1478 ...
 $ X1stFlrSF    : int  1136 904 912 1340 804 1164 1108 912 694 1478 ...
 $ X2ndFlrSF    : int  883 0 0 0 1157 1106 989 0 520 0 ...
```

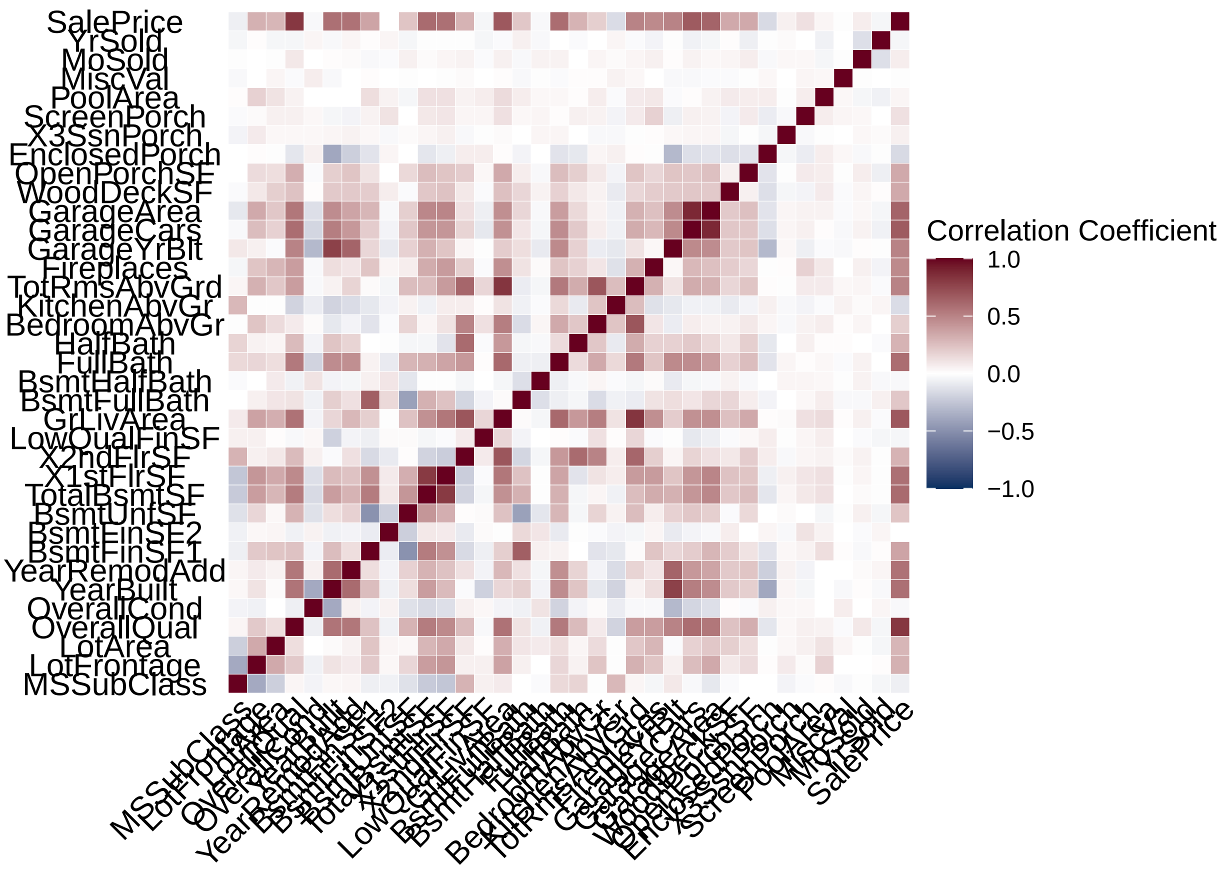
```
 $ LowQualFinSF : int  0 0 0 0 0 0 0 0 0 0 ...
 $ GrLivArea    : int  2019 904 912 1340 1961 2270 2097 912 1214 1478 ...
 $ BsmtFullBath : int  0 0 0 0 1 0 1 1 0 1 ...
 $ BsmtHalfBath : int  0 0 0 0 0 0 0 0 0 0 ...
 $ FullBath     : int  1 1 1 2 2 2 2 1 1 1 ...
 $ HalfBath     : int  0 0 0 0 1 1 1 0 0 1 ...
 $ BedroomAbvGr : int  3 3 2 3 3 4 3 2 3 3 ...
 $ KitchenAbvGr : int  1 1 1 1 1 1 1 1 1 1 ...
 $ TotRmsAbvGrd : int  8 6 5 6 7 9 8 5 6 6 ...
 $ Fireplaces   : int  0 0 0 0 1 1 1 0 0 2 ...
 $ GarageYrBlt  : int  1997 1966 1958 2007 2001 1999 1998 1992 1998 1957 ...
 $ GarageCars   : int  2 1 1 2 2 3 2 2 3 2 ...
 $ GarageArea   : int  532 408 275 396 560 671 583 576 936 442 ...
 $ WoodDeckSF   : int  509 0 0 100 125 132 253 0 216 114 ...
 $ OpenPorchSF  : int  135 0 0 30 192 57 170 240 0 0 ...
 $ EnclosedPorch: int  0 0 0 0 0 0 0 0 160 0 ...
 $ X3SsnPorch   : int  0 0 0 0 0 0 0 0 0 0 ...
 $ ScreenPorch  : int  0 0 0 0 0 0 0 0 0 216 ...
 $ PoolArea     : int  0 0 0 0 0 0 0 0 0 0 ...
 $ MiscVal      : int  0 0 0 0 0 0 0 0 0 0 ...
 $ MoSold       : int  10 12 7 8 6 4 6 12 1 6 ...
 $ YrSold       : int  2009 2008 2007 2007 2009 2010 2006 2008 2006 2009 ...
 $ SalePrice    : num  12 11.6 11.6 12.1 12.3 ...
```

```r
norm <- function(x) {
   (x - mean(x)) / sd(x)
}
denorm <- function(x,minval,maxval) {
    x*(maxval-minval) + minval
}
trainprice <- house_train$SalePrice
testprice <- house_test$SalePrice
totalprice <- train$SalePrice
train <- as.data.frame(lapply(train[,1:35], norm))
house_train <- as.data.frame(lapply(house_train[,1:35], norm))
house_test <- as.data.frame(lapply(house_test[,1:35], norm))
```

```r
house_train["SalePrice"] <- trainprice
house_test["SalePrice"] <- testprice
train["SalePrice"] <- totalprice
```

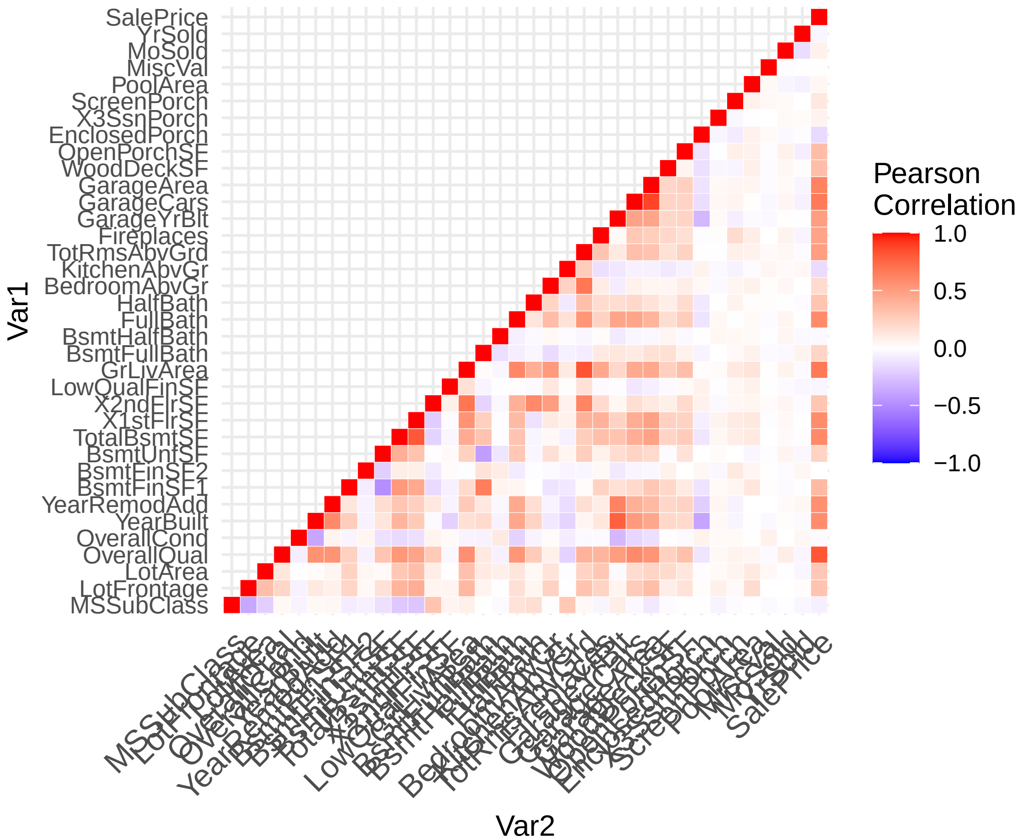
```r
num_cols =  unlist(lapply(house_train, is.numeric))
# Create the correlation matrix
corr = cor(house_train[,num_cols])
```

```r
ggcorrplot(corr,
     type = "full",lab = FALSE,
    legend.title = "Correlation Coefficient",
    colors = c("#053061", "white", "#67001f"),
    ggtheme = ggplot2::theme_void,
    outline.col = "white")
```



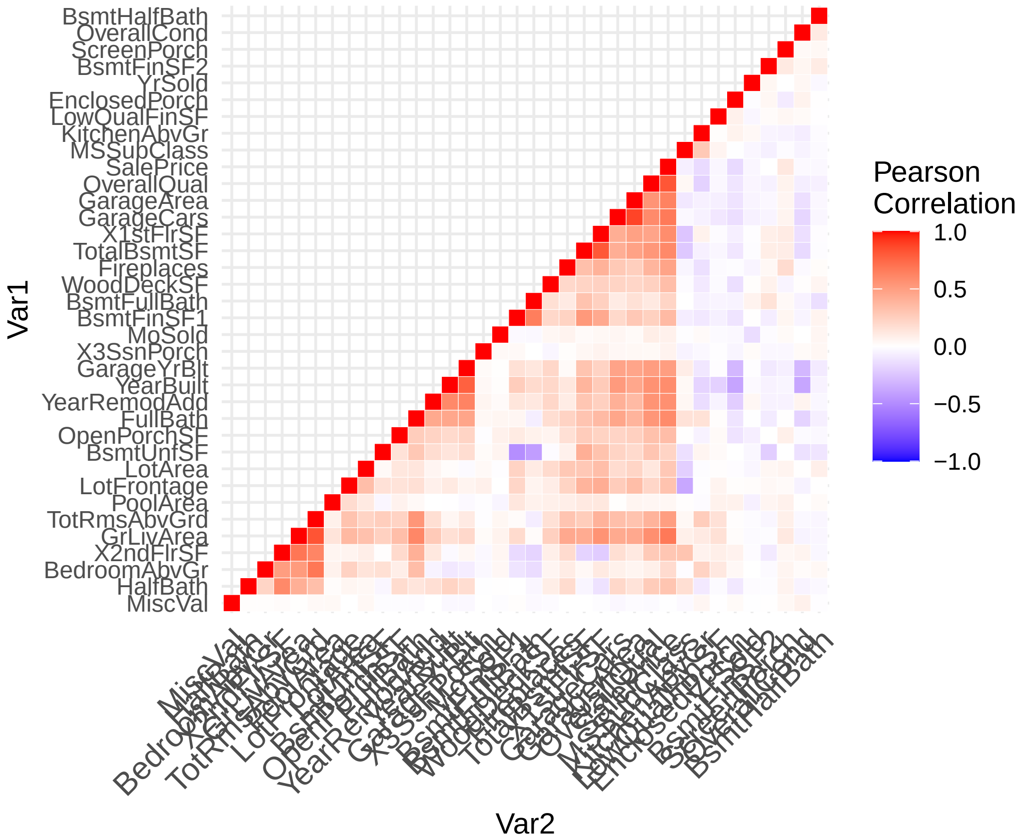

```r
# Get lower triangle of the correlation matrix
  get_lower_tri<-function(corr){
    corr[upper.tri(corr)] <- NA
    return(corr)
  }
  # Get upper triangle of the correlation matrix
  get_upper_tri <- function(corr){
    corr[lower.tri(corr)]<- NA
    return(corr)
  }


upper_tri <- get_upper_tri(corr)
# Melt the correlation matrix
library(reshape2)
melted_cormat <- melt(upper_tri, na.rm = TRUE)
# Heatmap
library(ggplot2)
ggplot(data = melted_cormat, aes(Var2, Var1, fill = value))+
 geom_tile(color = "white")+
 scale_fill_gradient2(low = "blue", high = "red", mid = "white",
   midpoint = 0, limit = c(-1,1), space = "Lab",
   name="Pearson\nCorrelation") +
  theme_minimal()+
 theme(axis.text.x = element_text(angle = 45, vjust = 1,
    size = 12, hjust = 1))+
```

```
coord_fixed()
```

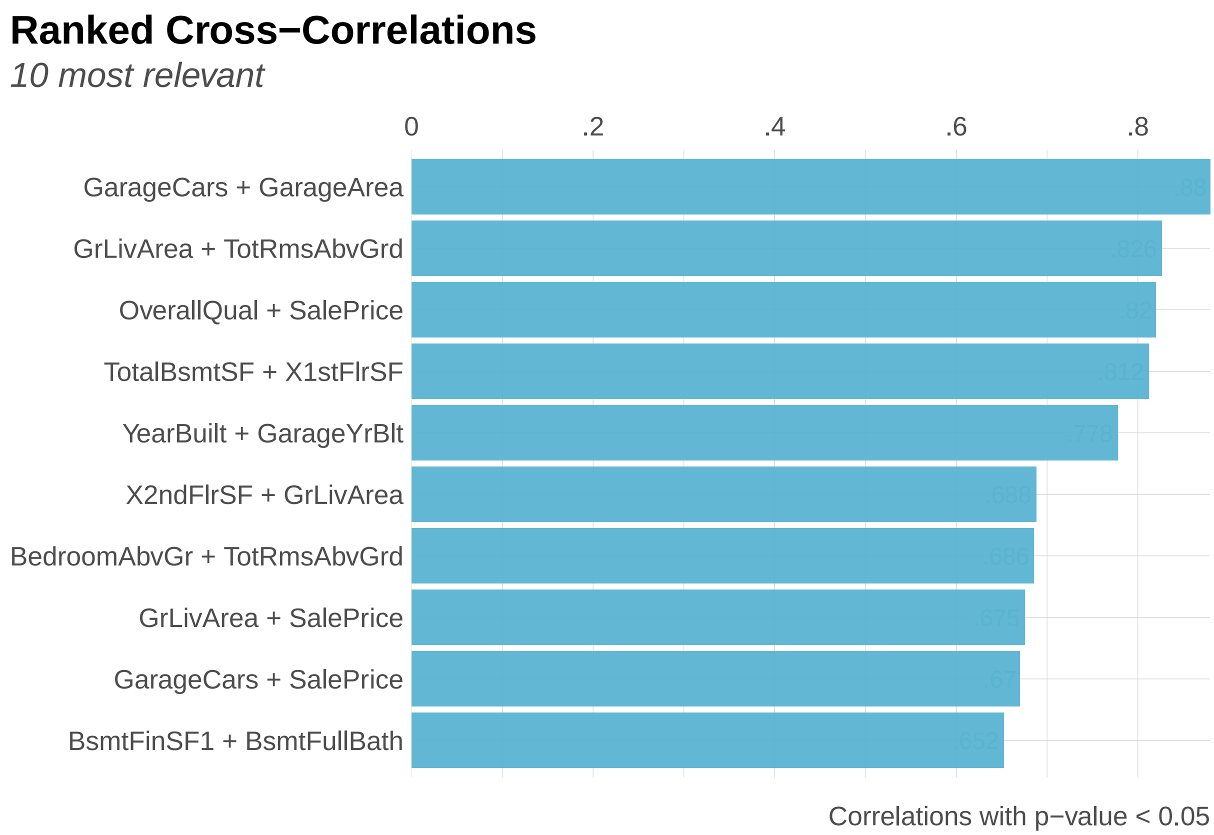



```
reorder_cormat <- function(corr){
# Use correlation between variables as distance
dd <- as.dist((1-corr)/2)
hc <- hclust(dd)
corr <-corr[hc$order, hc$order]
}


# Reorder the correlation matrix
corr <- reorder_cormat(corr)
upper_tri <- get_upper_tri(corr)
# Melt the correlation matrix
melted_cormat <- melt(upper_tri, na.rm = TRUE)
# Create a ggheatmap
ggheatmap <- ggplot(melted_cormat, aes(Var2, Var1, fill = value))+
 geom_tile(color = "white")+
 scale_fill_gradient2(low = "blue", high = "red", mid = "white",
   midpoint = 0, limit = c(-1,1), space = "Lab",
    name="Pearson\nCorrelation") +
  theme_minimal()+ # minimal theme
 theme(axis.text.x = element_text(angle = 45, vjust = 1,
    size = 12, hjust = 1))+
 coord_fixed()
```

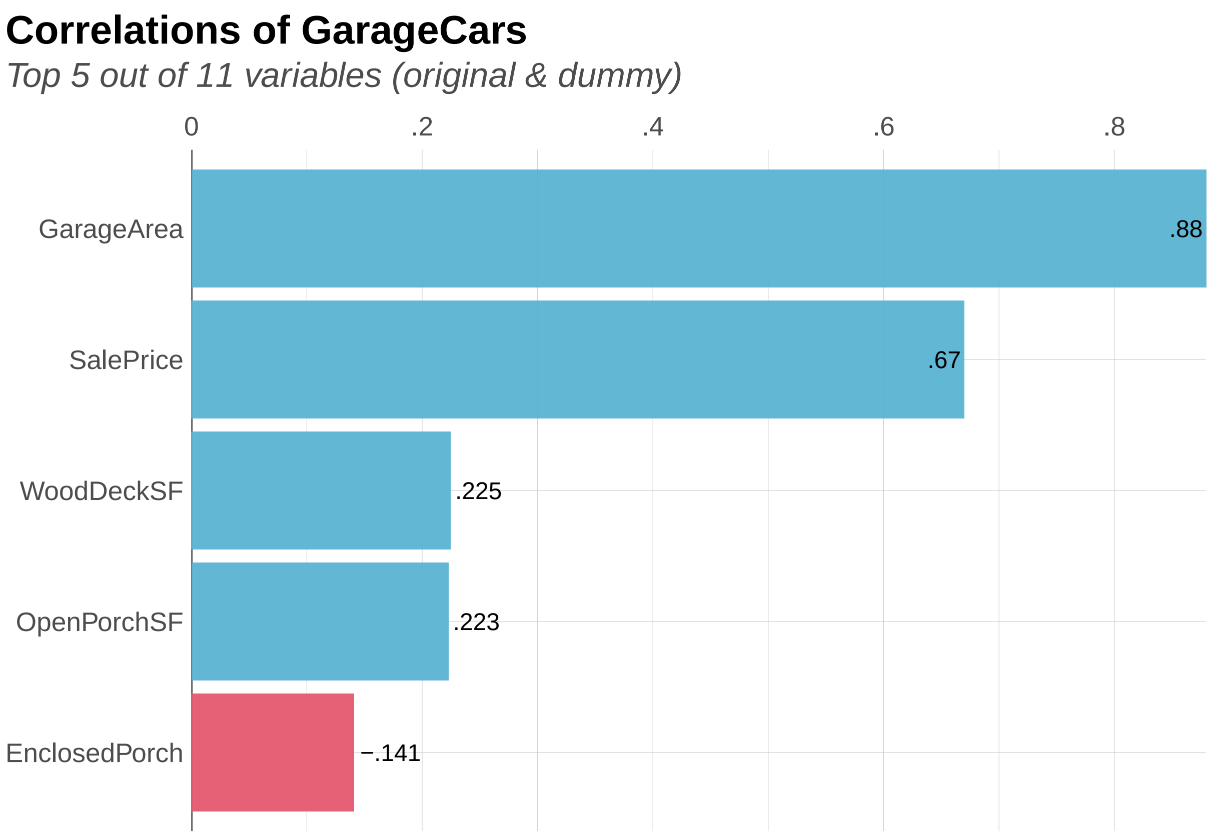
```r
# Print the heatmap
print(ggheatmap)
```



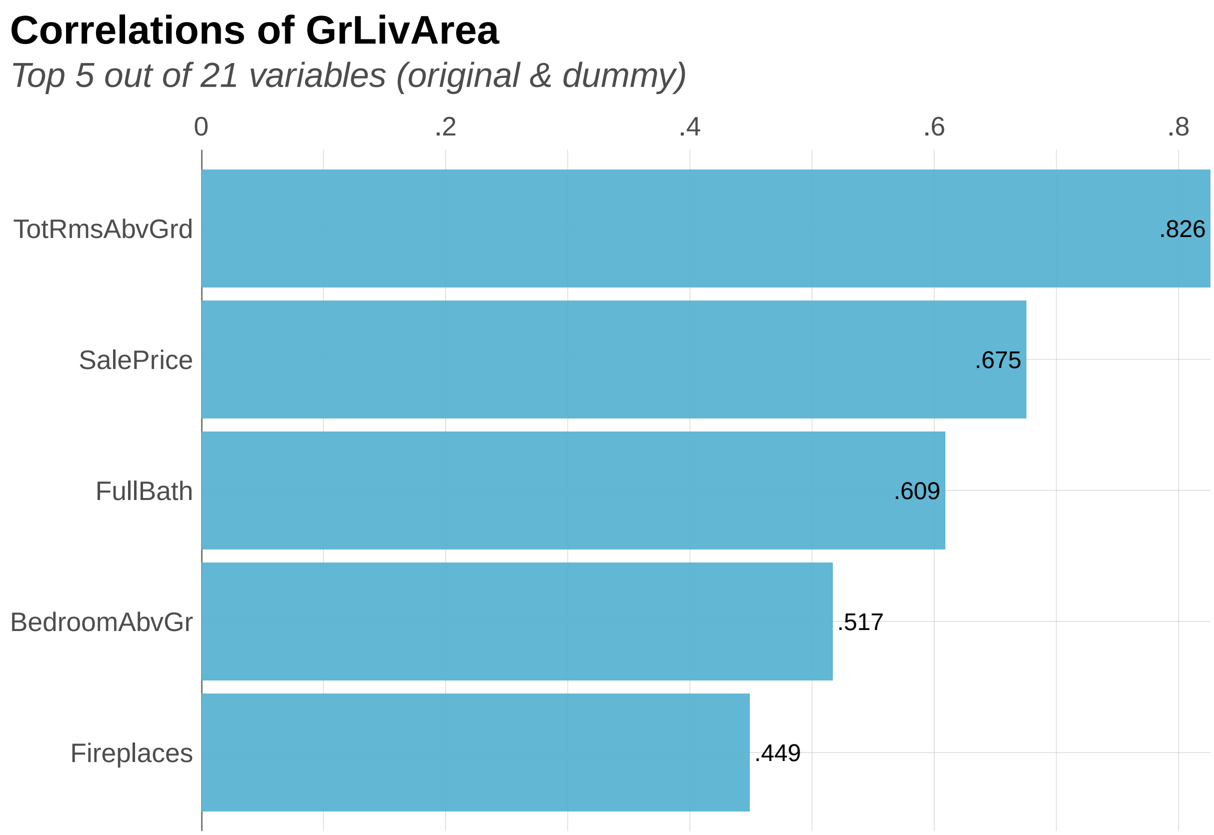

```r
#rank order for the cross-corr
library(lares)

corr_cross(house_train[,num_cols], # name of dataset
  max_pvalue = 0.05, # display only significant correlations (at 5% level)
  top = 10 # display top 10 couples of variables (by correlation coefficient)
)
```




```
#focus on one variable vs the rest of all
#looking at GarageCars since it appears to be the most correlated one
#GarageCars is high correlated with GarageAreas therefore, we would adjust
#those variables in our model
corr_var(house_train[,num_cols], # name of dataset
  GarageCars, # name of variable to focus on
  top = 5 # display top 5 correlations
)
```




```
corr_var(house_train[,num_cols], # name of dataset
  GrLivArea, # name of variable to focus on
  top = 5 # display top 5 correlations
)
```




```
# Convert correlation matrix to data frame
corr_df =   as_cordf(corr) %>%
# Focus on the Salary variable
  focus(SalePrice) %>%
# Get the absolute value of the correlation
# coefficient
  mutate(SalePrice = abs(SalePrice)) %>%
# Sort variables by absolute value of correlation
# coefficient
  arrange(SalePrice) %>%
# Clean up headers
  rename(`correlation with SalePrice` = term ) %>%
  rename(corr_coef = SalePrice)
corr_df
```

```
# A tibble: 35 x 2
   `correlation with SalePrice` corr_coef
   <chr>                            <dbl>
 1 BsmtFinSF2                    0.000198
 2 MiscVal                       0.00540
 3 OverallCond                   0.0264
 4 BsmtHalfBath                  0.0285
 5 LowQualFinSF                  0.0380
 6 YrSold                        0.0385
 7 PoolArea                      0.0418
 8 X3SsnPorch                    0.0622
```

```
 9 MSSubClass                      0.0679
10 MoSold                          0.0729
# ... with 25 more rows
```

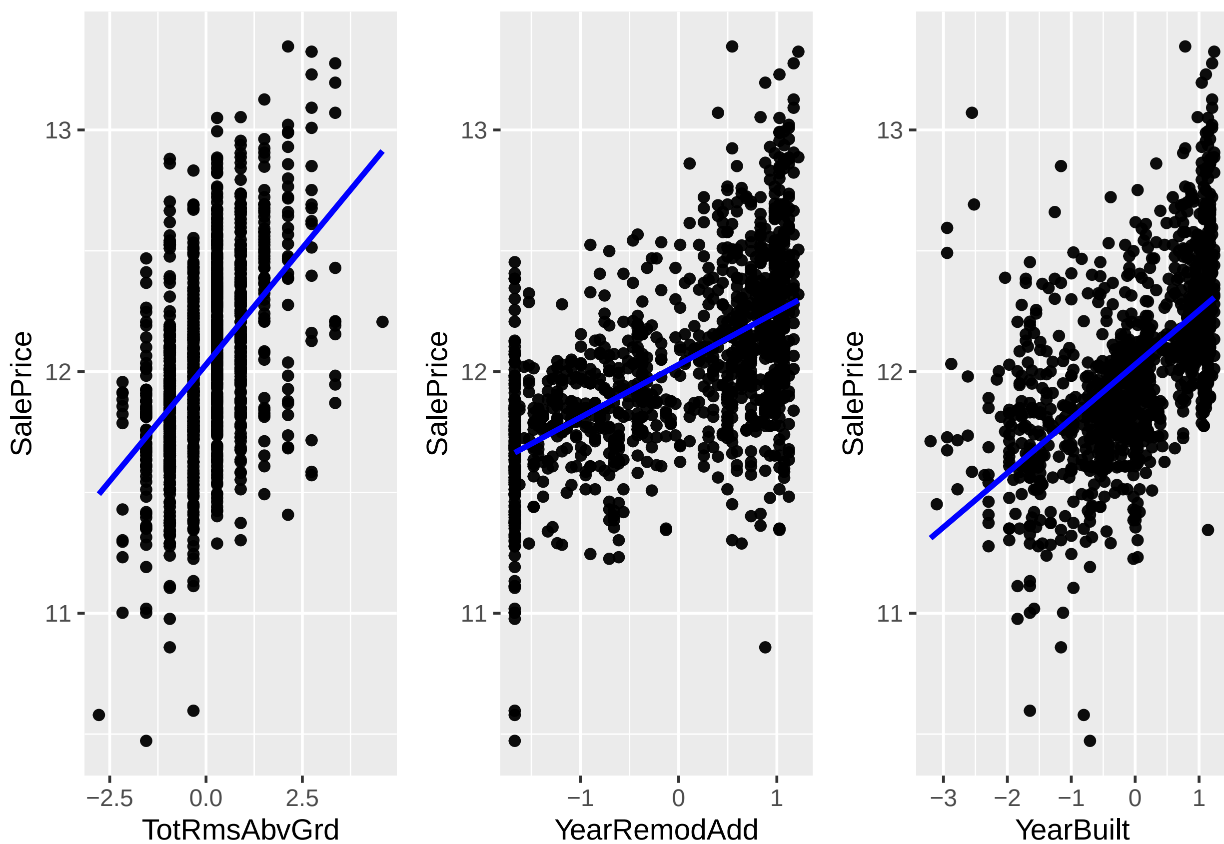
```r
# x = which(corr_df$corr_coef >= 0.5)
x = corr_df[which(corr_df$corr_coef >= 0.5),]
new_var = x['correlation with SalePrice']
# new_var
# house_train %>%
house_train <- house_train[,colnames(house_train) %in%
                             c(new_var$`correlation with SalePrice`,
                               'SalePrice')]
house_test <- house_test[,colnames(house_test) %in%
                           c(new_var$`correlation with SalePrice`,
                             'SalePrice')]
# house_train = house_train %>% select(new_var)
# house_test = house_test %>% select(new_var)
```

```r
corr_df[which(corr_df$corr_coef >= 0.5),]
```

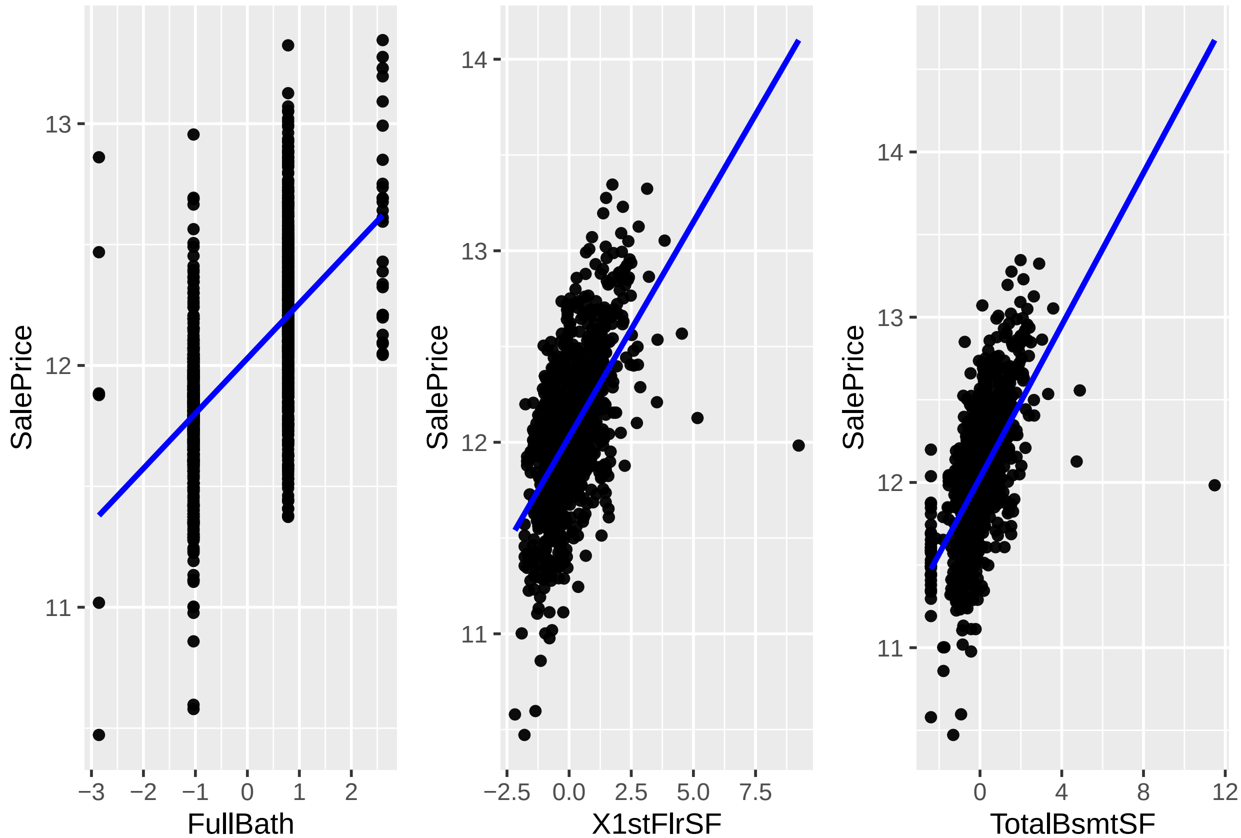
```
# A tibble: 11 x 2
   `correlation with SalePrice` corr_coef
   <chr>                            <dbl>
 1 GarageYrBlt                      0.501
 2 TotRmsAbvGrd                     0.502
 3 YearRemodAdd                     0.568
 4 X1stFlrSF                        0.583
 5 YearBuilt                        0.584
 6 FullBath                         0.592
 7 TotalBsmtSF                      0.600
 8 GarageArea                       0.634
 9 GarageCars                       0.670
10 GrLivArea                        0.675
11 OverallQual                      0.820
```

```r
p1 <- ggplot(house_train,mapping = aes(x = TotRmsAbvGrd,y=SalePrice)) +
      geom_point() +
      geom_smooth(method = "lm", formula = y~x,
                  se=FALSE,colour = "blue")
p2 <- ggplot(house_train,mapping = aes(x =YearRemodAdd,y=SalePrice)) +
      geom_point() +
      geom_smooth(method = "lm", formula = y~x,
                  se=FALSE,colour = "blue")
p3 <- ggplot(house_train,mapping = aes(x =YearBuilt,y=SalePrice)) +
      geom_point() +
      geom_smooth(method = "lm", formula = y~x,
                  se=FALSE,colour = "blue")
p4 <- ggplot(house_train,mapping = aes(x = FullBath,y=SalePrice)) +
      geom_point() +
      geom_smooth(method = "lm", formula = y~x,
                  se=FALSE,colour = "blue")
p5 <- ggplot(house_train,mapping = aes(x =X1stFlrSF,y=SalePrice)) +
      geom_point() +
```

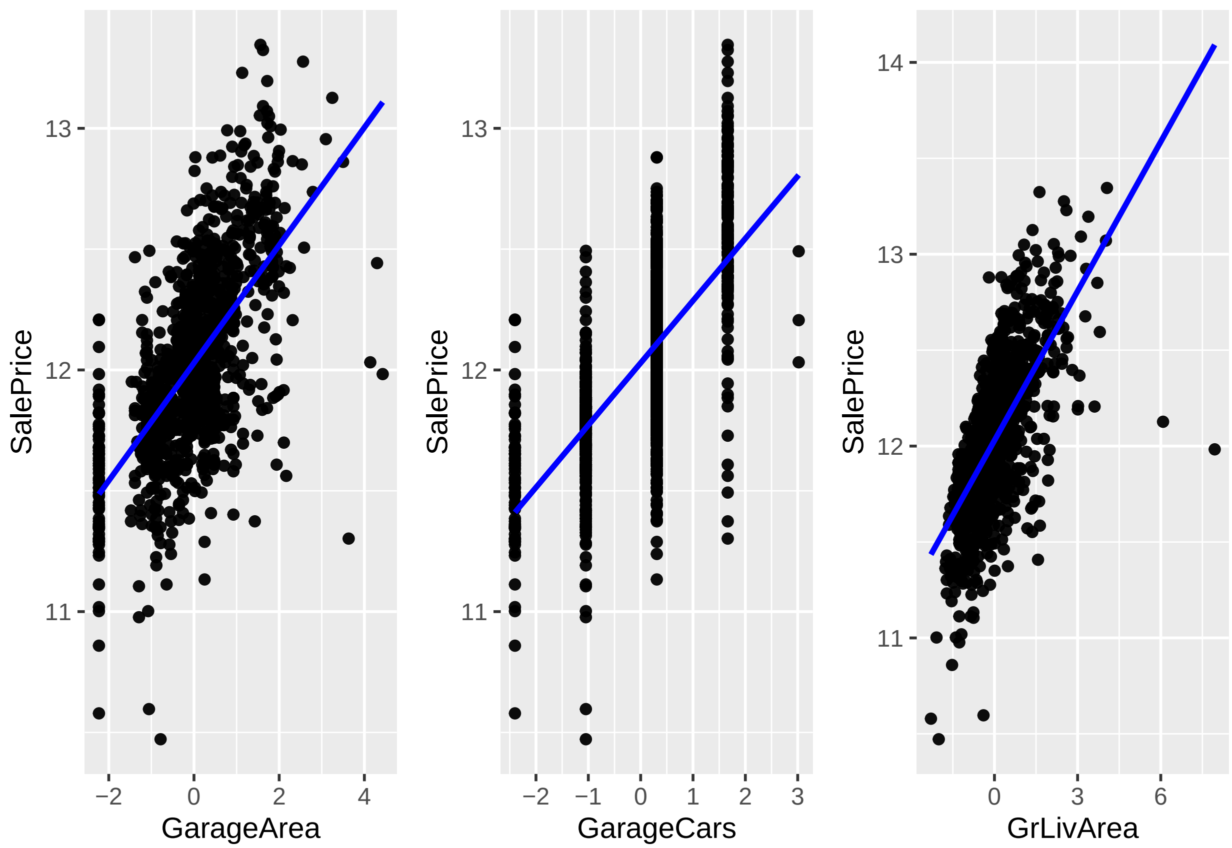
```
      geom_smooth(method = "lm", formula = y~x,
                  se=FALSE,colour = "blue")
p6 <- ggplot(house_train,mapping = aes(x =TotalBsmtSF,y=SalePrice)) +
      geom_point() +
      geom_smooth(method = "lm", formula = y~x,
                  se=FALSE,colour = "blue")
p7 <- ggplot(house_train,mapping = aes(x = GarageArea,y=SalePrice)) +
      geom_point() +
      geom_smooth(method = "lm", formula = y~x,
                  se=FALSE,colour = "blue")
p8 <- ggplot(house_train,mapping = aes(x =GarageCars,y=SalePrice)) +
      geom_point() +
      geom_smooth(method = "lm", formula = y~x,
                  se=FALSE,colour = "blue")
p9 <- ggplot(house_train,mapping = aes(x =GrLivArea,y=SalePrice)) +
      geom_point() +
      geom_smooth(method = "lm", formula = y~x,
                  se=FALSE,colour = "blue")
p10 <- ggplot(house_train,mapping = aes(x =OverallQual,y=SalePrice)) +
      geom_point() +
      geom_smooth(method = "lm", formula = y~x,
                  se=FALSE,colour = "blue")
plot_grid(p1,p2,p3, ncol = 3)
```



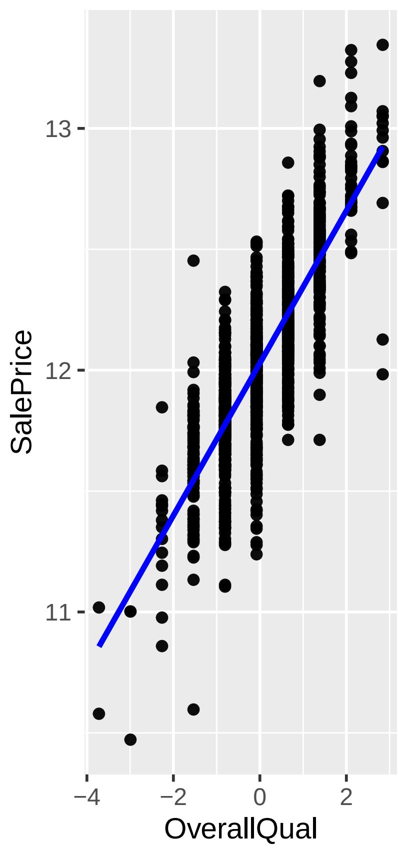
```r
plot_grid(p4,p5,p6, ncol = 3)
```




```r
plot_grid(p7,p8,p9, ncol = 3)
```




```
plot_grid(p10, ncol = 3)
```



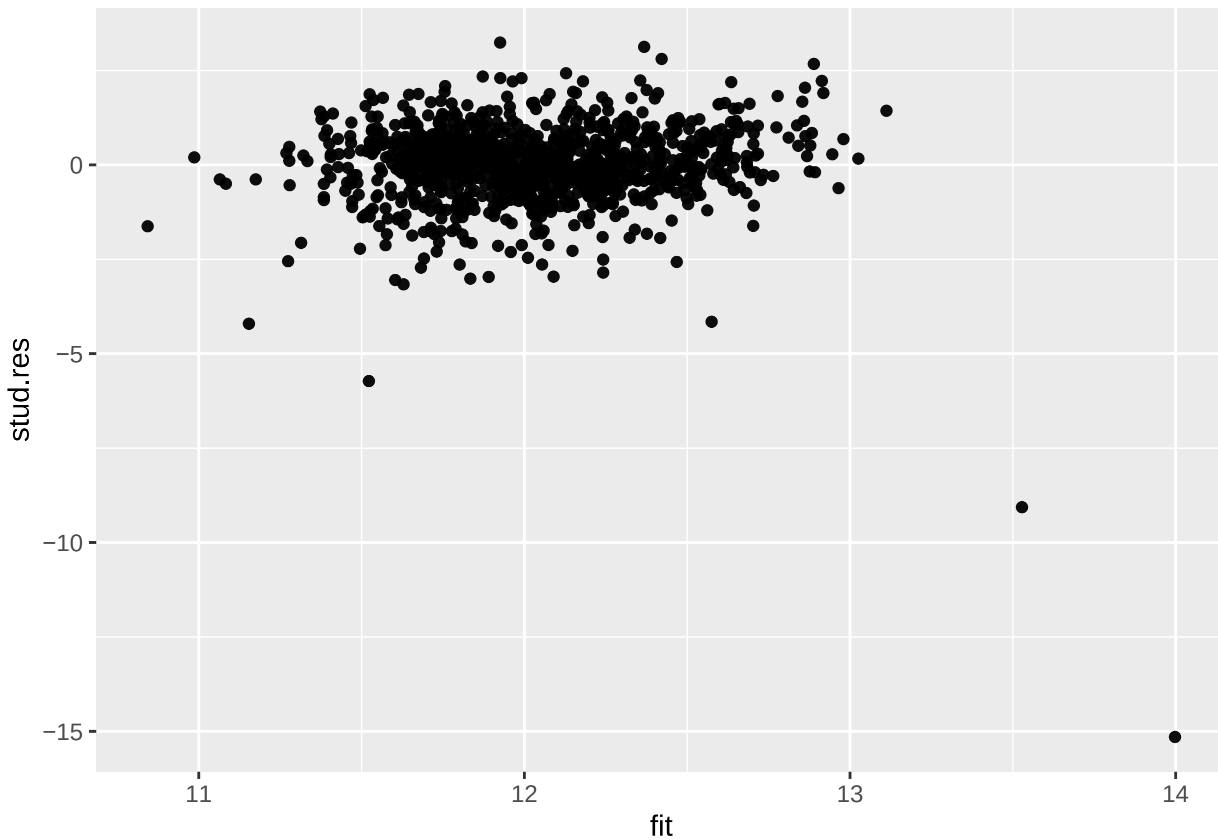
Linear Regression

```
#build base model
lm1 <- lm(SalePrice~., data = house_train)
summary(lm1)
```

```
Call:
lm(formula = SalePrice ~ ., data = house_train)

Residuals:
    Min      1Q  Median      3Q     Max 
-2.0151 -0.0750  0.0085  0.0906  0.5275 

Coefficients:
              Estimate Std. Error t value             Pr(>|t|)    
(Intercept)  12.029148   0.004817 2497.19 < 0.0000000000000002 ***
OverallQual   0.135660   0.007914   17.14 < 0.0000000000000002 ***
YearBuilt     0.079051   0.008824    8.96 < 0.0000000000000002 ***
YearRemodAdd  0.048187   0.006719    7.17       0.0000000000013 ***
TotalBsmtSF   0.032870   0.008998    3.65               0.00027 ***
X1stFlrSF     0.021118   0.009098    2.32               0.02044 *  
GrLivArea     0.098481   0.010755    9.16 < 0.0000000000000002 ***
FullBath      0.000301   0.007174    0.04               0.96653    
```

```
TotRmsAbvGrd  0.011784   0.008925    1.32             0.18698
GarageYrBlt  -0.034113   0.008367   -4.08      0.0000487672144 ***
GarageCars    0.049146   0.011088    4.43      0.0000101959037 ***
GarageArea    0.014874   0.010916    1.36             0.17328
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.165 on 1156 degrees of freedom
Multiple R-squared:  0.818, Adjusted R-squared:  0.817
F-statistic:  473 on 11 and 1156 DF,  p-value: <0.0000000000000002
```

```r
#check if there are any outliers
fit <- fitted(lm1)
stud.res <- studres(lm1)
stud.fit <- data.frame("fit"=fit,"stud.res"=stud.res)
ggplot(stud.fit, mapping = aes(x=fit,y=stud.res))+
geom_point()
```




```r
#index1 <- which(stud.res > 5)
index2 <- which(stud.res < -5)
index <- index2
index
```

```
142 706 837
142 706 837
```

```r
summary(lm1)$sigma
```

```
[1] 0.165
```

```r
#summary(lm1)$r.squared

#remove outliers
adformula <- formula(SalePrice~.)
lm_no_outlier = lm(adformula, data = house_train[-index,])
summary(lm_no_outlier)$sigma
```

```
[1] 0.14
```

```r
#summary(lm_no_outlier)$r.squared
```

```r
#since FullBath & GarageArea & TotRmsAbvGrd 's pval is greater than 0.05,
#we dont think it is statistically significant,we run again with a smaller model

lm2 <- lm(SalePrice~.-FullBath-GarageArea-TotRmsAbvGrd, data = house_train)
summary(lm2)
```

```
Call:
lm(formula = SalePrice ~ . - FullBath - GarageArea - TotRmsAbvGrd,
    data = house_train)

Residuals:
    Min      1Q  Median      3Q     Max
-2.0103 -0.0761  0.0076  0.0912  0.5206

Coefficients:
             Estimate Std. Error t value             Pr(>|t|)
(Intercept)  12.02915    0.00482 2496.80 < 0.0000000000000002 ***
OverallQual   0.13442    0.00788   17.06 < 0.0000000000000002 ***
YearBuilt     0.07711    0.00861    8.96 < 0.0000000000000002 ***
YearRemodAdd  0.04739    0.00668    7.09      0.0000000000023 ***
TotalBsmtSF   0.03298    0.00889    3.71              0.00022 ***
X1stFlrSF     0.02243    0.00905    2.48              0.01333 *
GrLivArea     0.10948    0.00680   16.10 < 0.0000000000000002 ***
GarageYrBlt  -0.03164    0.00813   -3.89              0.00011 ***
GarageCars    0.06172    0.00649    9.50 < 0.0000000000000002 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.165 on 1159 degrees of freedom
Multiple R-squared:  0.818, Adjusted R-squared:  0.817
F-statistic:  650 on 8 and 1159 DF,  p-value: <0.0000000000000002
```

```r
summary(lm2)$sigma
```

```
[1] 0.165
```

```
summary(lm2)$r.squared
```

```
[1] 0.818
```

```
#remove outliers
adformula <- formula(SalePrice~.-FullBath-GarageArea-TotRmsAbvGrd)
lm2_no_outlier = lm(adformula, data = house_train[-index,])
summary(lm2_no_outlier)$sigma
```

```
[1] 0.141
```

```
summary(lm2_no_outlier)$r.squared
```

```
[1] 0.865
```

```
#r^2 decreased to 0.761
```

```
#compare two linear regression model
anova(lm2_no_outlier,lm_no_outlier)
```

```
Analysis of Variance Table

Model 1: SalePrice ~ (OverallQual + YearBuilt + YearRemodAdd + TotalBsmtSF +
    X1stFlrSF + GrLivArea + FullBath + TotRmsAbvGrd + GarageYrBlt +
    GarageCars + GarageArea) - FullBath - GarageArea - TotRmsAbvGrd
Model 2: SalePrice ~ OverallQual + YearBuilt + YearRemodAdd + TotalBsmtSF +
    X1stFlrSF + GrLivArea + FullBath + TotRmsAbvGrd + GarageYrBlt +
    GarageCars + GarageArea
  Res.Df  RSS Df Sum of Sq    F   Pr(>F)
1   1156 22.9
2   1153 22.4  3     0.499 8.54 0.000013 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
#Pval is smaller than 0.05, we chose lm2_no_outlier to perform our test
```

```
#outlier
house_train1 = house_train[-index,]

#best subset selection
# Draw validation set
house_validation_data = house_train1 %>% sample_frac(size = 0.3)
# Create the remaining training set
house_training_data = setdiff(house_train1, house_validation_data)

nvars = 7

regfit.best=regsubsets(SalePrice~.-FullBath-GarageArea-TotRmsAbvGrd,
                       data=house_training_data,nvmax=nvars)
best.sum <- summary(regfit.best)
best.model <- which.max(best.sum$adjr2)
best.model
```

```
[1] 7
```

```r
coef(regfit.best,id=best.model)
```

```
 (Intercept)  OverallQual    YearBuilt YearRemodAdd  TotalBsmtSF    GrLivArea
     12.0337       0.1180       0.0795       0.0442       0.0812       0.1337
 GarageYrBlt   GarageCars
     -0.0323       0.0593
```

```r
validation.mat=model.matrix(SalePrice~.-FullBath-GarageArea-TotRmsAbvGrd,
                            data=house_validation_data)
val.errors = numeric(nvars)

for(each in 1:nvars){
  coefi = coef(regfit.best,id=each)
  pred = validation.mat[,names(coefi)]%*%coefi
  val.errors[each]=mean((house_validation_data$SalePrice - pred)^2)
  sprintf("the val error is",val.errors[each])
}
best.subset.model = which.min(val.errors)
best.subset.model
```

```
[1] 7
```

```r
#train on our test data in order to determine the accuracy
best.fit=regsubsets(SalePrice~.-FullBath-GarageArea-TotRmsAbvGrd,
                    data=house_train1,nvmax =7)
coefi_final1<- coef(best.fit,best.subset.model)
coefi_final1
```

```
 (Intercept)  OverallQual    YearBuilt YearRemodAdd  TotalBsmtSF    GrLivArea
     12.0336       0.1153       0.0802       0.0457       0.0835       0.1431
 GarageYrBlt   GarageCars
     -0.0316       0.0479
```

```r
#test data
test.mat1=model.matrix(SalePrice~.-FullBath-GarageArea-TotRmsAbvGrd,
                       data=house_test)

pred_test_lm = test.mat1[,names(coefi_final1)]%*%coefi_final1
```

```r
head(house_test$SalePrice)
```

```
[1] 12.1 12.4 11.4 11.1 11.3 12.0
```

```r
head(pred_test_lm)
```

```
  [,1]
1 12.0
2 12.5
```

```
3 11.6
4 11.3
5 11.3
6 12.0
```

```r
pred_test_lm_org = exp(pred_test_lm)
```

```r
library(Metrics)
#final lm model - accuracy
rmse(pred_test_lm,house_test$SalePrice)
```

```
[1] 0.184
```

```r
#save as score
Score["Linear Regression","RMSE"] = rmse(pred_test_lm,house_test$SalePrice)

Score
```

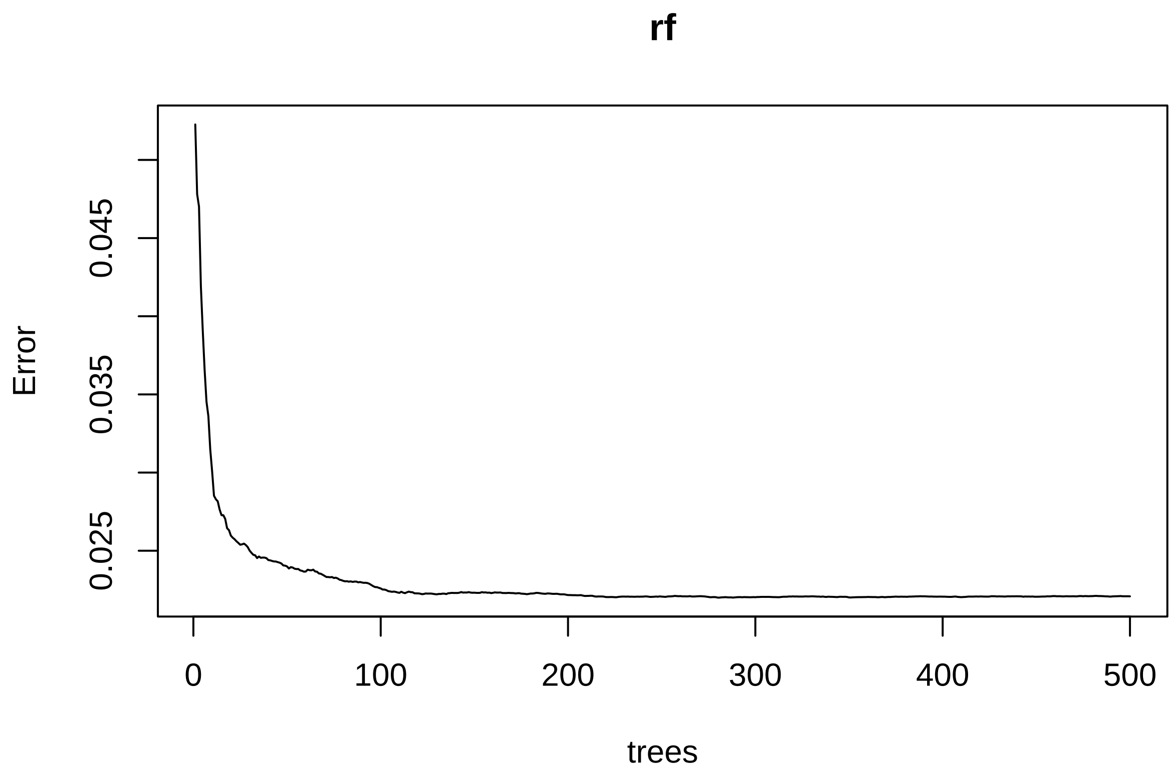
```
                   RMSE
Linear Regression 0.184
Random Forest        NA
Elastic Net          NA
Boosting             NA
```

Random Forest

```r
#build base model
rf <-randomForest(SalePrice ~., data=house_train)
rf
```

```
Call:
 randomForest(formula = SalePrice ~ ., data = house_train)
               Type of random forest: regression
                     Number of trees: 500
No. of variables tried at each split: 3

          Mean of squared residuals: 0.0221
                    % Var explained: 85
```

```r
plot(rf)
```




```r
# number of trees with lowest MSE
which.min(rf$mse)
```

```
[1] 280
```

```r
# RMSE of this optimal random forest
sqrt(rf$mse[which.min(rf$mse)])
```

```
[1] 0.148
```

```r
#tuning parameter
house_tree_tune <- rpart(SalePrice ~ .,data = house_train,method="anova",
                         maxdepth=7)
house_tree_tune
```

```
n= 1168

node), split, n, deviance, yval
      * denotes terminal node

 1) root 1168 172.00 12.0
   2) OverallQual< 0.288 728  53.70 11.8
     4) GrLivArea< -0.276 439  26.70 11.7
       8) YearBuilt< -0.659 139   9.60 11.5
```

```
        16) OverallQual< -1.9 16   1.45 11.1 *
        17) OverallQual>=-1.9 123   5.22 11.6 *
       9) YearBuilt>=-0.659 300   9.34 11.8
        18) TotalBsmtSF< -0.112 163   4.45 11.7 *
        19) TotalBsmtSF>=-0.112 137   2.57 11.9 *
     5) GrLivArea>=-0.276 289  15.30 12.0
      10) OverallQual< -0.441 120   5.59 11.9 *
      11) OverallQual>=-0.441 169   6.38 12.1 *
   3) OverallQual>=0.288 440  38.80 12.4
     6) OverallQual< 1.02 259  10.50 12.2
      12) GrLivArea< 1.1 221   6.76 12.2 *
      13) GrLivArea>=1.1 38   1.07 12.5 *
     7) OverallQual>=1.02 181  14.20 12.6
      14) OverallQual< 1.75 133   7.20 12.5 *
      15) OverallQual>=1.75 48   3.24 12.8 *
```

```r
# hyperparameter grid search
hyper_grid <- expand.grid(
  mtry = seq(5, 10, by = 1),
  node_size = seq(4, 16, by = 2),
  sample_size = c(.5, .6, .70, .80),
  OOB_RMSE = 0
)
# total number of combinations
nrow(hyper_grid)
```

```
[1] 168
```

```r
for(i in 1:nrow(hyper_grid)) {
  # train model
  model <- ranger(
  formula = SalePrice ~ .,
  data = house_train,
  num.trees = 348,
  mtry = hyper_grid$mtry[i],
  min.node.size = hyper_grid$node_size[i],
  sample.fraction = hyper_grid$sample_size[i] )
  # add OOB error to grid
  hyper_grid$OOB_RMSE[i] <- sqrt(model$prediction.error)
}
```

```r
hyper_grid %>%
arrange(OOB_RMSE) %>% head(10)
```

```
   mtry node_size sample_size OOB_RMSE
1     6         6         0.7    0.147
2     7         6         0.8    0.147
3     6         8         0.8    0.147
4     5        12         0.8    0.147
5     6         4         0.6    0.147
6     8        14         0.8    0.147
7     6        12         0.7    0.147
```

```
8     6          4          0.8     0.147
9     5          6          0.7     0.147
10    5          6          0.8     0.147
```

```
best.rf <- hyper_grid %>%
  arrange(OOB_RMSE) %>%
  head(1)
best.rf
```

```
  mtry node_size sample_size OOB_RMSE
1    6         6          0.7    0.147
```

```
optimal_rf <- ranger(
formula = SalePrice ~ .,
data = house_train,
num.trees = 348,
mtry = best.rf$mtry,
min.node.size = best.rf$node_size,
sample.fraction = best.rf$sample_size,
importance = 'impurity')
```

```
#make predictions on
predict_rf <- predict(optimal_rf, house_test)$predictions
```

```
#store them in Score
Score["Random Forest","RMSE"] = RMSE(predict_rf, house_test$SalePrice)

Score
```

```
                   RMSE
Linear Regression 0.184
Random Forest     0.192
Elastic Net          NA
Boosting             NA
```

Elastic net

```
# Predictor variables
x <- model.matrix(SalePrice~., house_train)[,-1]
# Outcome variable
y <- house_train$SalePrice
```

```
# Build the model using the training set
set.seed(123)
model <- train(
  SalePrice ~., data = house_train, method = "glmnet",
  trControl = trainControl("cv", number = 10),
  tuneLength = 10
)
# Best tuning parameter
model$bestTune
```

```
  alpha  lambda
5   0.1 0.00957
```

```r
coef(model$finalModel, model$bestTune$lambda)
```

```
12 x 1 sparse Matrix of class "dgCMatrix"
                  s1
(Intercept)  12.0291
OverallQual   0.1327
YearBuilt     0.0707
YearRemodAdd  0.0462
TotalBsmtSF   0.0337
X1stFlrSF     0.0222
GrLivArea     0.0924
FullBath      0.0024
TotRmsAbvGrd  0.0144
GarageYrBlt  -0.0240
GarageCars    0.0485
GarageArea    0.0157
```

```r
# Make predictions on the test data
x.test <- model.matrix(SalePrice ~., house_test)[,-1]
predictions <- model %>% predict(x.test)
```

```r
Score["Elastic Net","RMSE"] = RMSE(predictions, house_test$SalePrice)

Score
```

```
                   RMSE
Linear Regression 0.184
Random Forest     0.192
Elastic Net       0.191
Boosting             NA
```

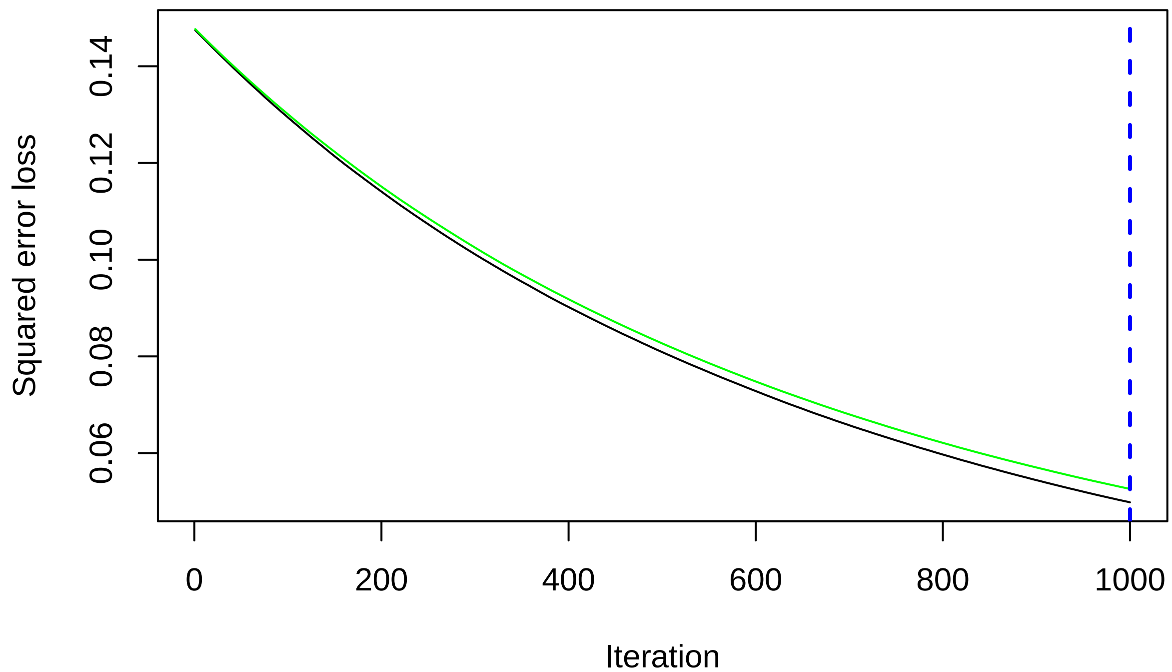
Boosting

- Basic GBM model

```r
hit_gbm <- gbm(
  formula = SalePrice ~ .,
  data = house_train,
  distribution = "gaussian",# SSE loss function
  n.trees = 1000,
  shrinkage = 0.001, #learning rate
  cv.folds = 10,
  interaction.depth = 5 #depth of each tree
)
# find index for number trees with minimum CV error
best <- which.min(hit_gbm$cv.error)
# get MSE and compute RMSE
sqrt(hit_gbm$cv.error[best])
```

```
[1] 0.229
```

```r
gbm.perf(hit_gbm, method = "cv")
```

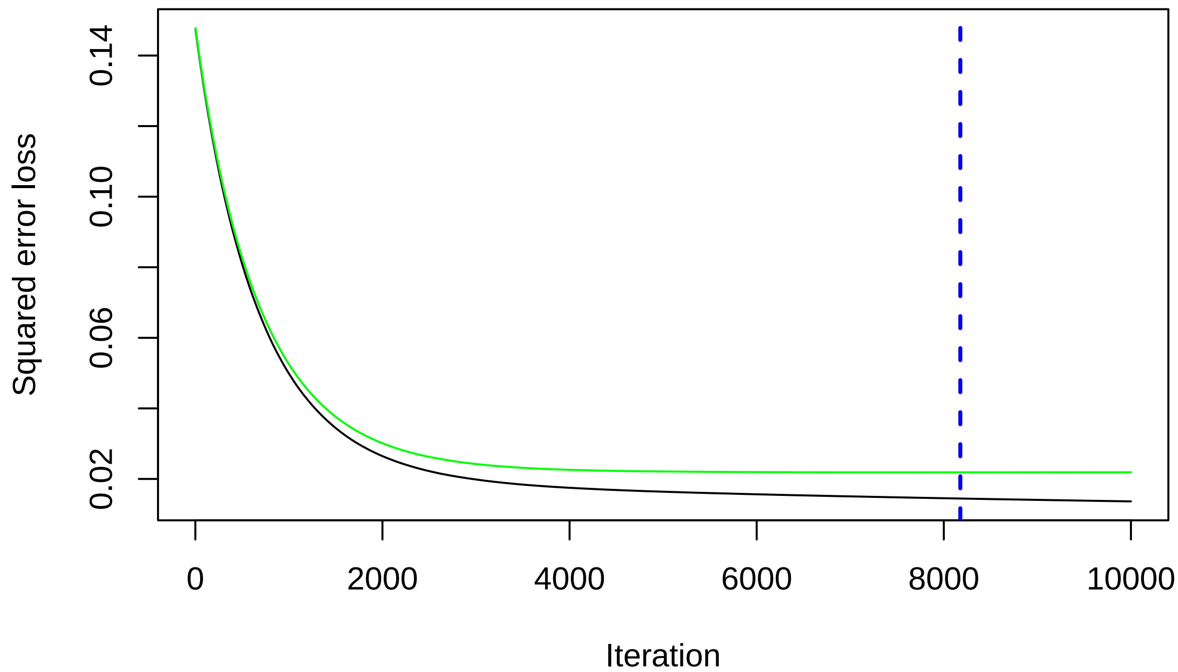



```
[1] 1000
```

```r
hit_gbm <- gbm(
  formula = SalePrice ~ .,
  data = house_train,
  distribution = "gaussian",# SSE loss function
  n.trees = 10000,
  shrinkage = 0.001, #learning rate
  cv.folds = 10,
  interaction.depth = 5 #depth of each tree
)
# find index for number trees with minimum CV error
best <- which.min(hit_gbm$cv.error)
# get MSE and compute RMSE
sqrt(hit_gbm$cv.error[best])
```

```
[1] 0.148
```

```r
gbm.perf(hit_gbm, method = "cv")
```




```
[1] 8176
```

```r
pred.gbm.final <- predict.gbm(hit_gbm, n.trees=4000, newdata = house_test)
rmse.gbm.final.rmse <- sqrt(mean((house_test$SalePrice -
                          pred.gbm.final)^2))
rmse.gbm.final.rmse
```

```
[1] 0.188
```

```r
pred.gbm.final
```

```
  [1] 12.0 12.6 11.7 11.2 11.5 11.9 12.1 11.5 12.3 11.5 11.8 12.2 12.1 12.6 12.1
 [16] 12.2 12.0 12.3 11.7 12.5 12.1 12.0 11.6 11.8 11.9 12.1 11.7 12.6 12.1 11.7
 [31] 12.2 12.2 12.0 11.9 11.7 11.8 12.4 11.9 12.2 12.2 12.2 11.7 11.8 12.3 12.8
 [46] 11.8 12.4 12.4 12.4 12.1 12.0 11.6 12.5 12.1 11.8 12.5 11.5 11.6 11.7 12.3
 [61] 12.5 12.8 12.6 11.7 11.4 12.0 12.3 12.2 11.6 11.7 12.0 12.4 11.6 11.7 12.8
 [76] 12.3 11.6 11.4 12.2 12.0 11.9 12.5 11.9 11.9 11.8 12.2 12.2 12.7 12.2 11.6
 [91] 12.9 12.2 11.4 11.7 11.7 11.9 11.6 11.9 11.5 11.3 12.7 12.4 11.9 11.8 11.5
[106] 11.8 12.2 11.6 12.7 12.0 12.5 11.9 12.5 12.4 11.8 12.0 11.8 12.2 11.7 11.8
[121] 12.2 12.4 12.5 12.8 11.9 12.7 11.9 12.4 12.2 12.4 12.3 11.4 12.2 11.6 12.2
[136] 12.0 12.0 12.2 12.8 11.8 12.4 12.9 11.6 11.2 12.6 11.9 11.8 11.7 12.4 12.4
```

```
[151] 11.8 12.1 12.3 11.8 12.7 12.1 11.8 11.7 12.2 11.6 11.9 11.9 11.6 12.4 11.9
[166] 11.7 12.4 12.0 12.3 11.9 12.2 11.7 11.8 11.9 12.3 11.7 12.1 11.9 11.8 12.1
[181] 11.8 12.2 12.0 12.1 12.1 12.6 12.5 12.3 12.1 11.9 11.7 11.7 11.8 11.7 12.7
[196] 12.1 11.0 11.5 12.3 12.1 12.7 12.0 11.4 12.4 11.8 11.7 11.9 11.7 12.6 11.5
[211] 12.1 11.9 12.1 11.8 11.8 11.5 12.5 11.8 12.1 11.6 12.2 11.8 11.8 11.7 12.1
[226] 12.3 12.3 11.8 11.7 12.5 11.8 12.1 11.6 12.0 12.8 11.5 11.7 11.7 12.2 12.2
[241] 12.5 11.9 12.1 12.9 12.0 11.7 12.2 11.8 12.5 11.8 11.9 12.1 11.8 12.0 12.1
[256] 12.7 11.9 12.0 12.2 12.9 11.9 11.9 11.9 11.9 11.9 11.8 12.4 12.2 12.2 11.4
[271] 11.3 11.8 11.5 11.9 11.3 11.7 12.0 12.6 12.6 12.3 12.0 11.5 12.2 11.8 12.3
[286] 12.6 11.9 11.5 12.7 12.5 12.3 11.9
```

```r
CV_RSq <- (cor(pred.gbm.final, house_test$SalePrice))^2
CV_RSq
```

```
[1] 0.843
```

```r
# create hyperparameter grid
hyper_grid <- expand.grid(
  shrinkage = c(.001, .1),
  interaction.depth = c(1, 5),
  n.minobsinnode = c(5, 10),
  bag.fraction = c(.7, .8),
  optimal_trees = 0,
  min_RMSE = 0
)

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

```
[1] 16
```

```r
# grid search
for(i in 1:nrow(hyper_grid)) {
   print(i)
  # train model
  gbm.tune <- gbm(
    formula = SalePrice ~ .,
    distribution = "gaussian",
    data = house_train,
    n.trees = 4000,
    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],
    cv.folds = 10)

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

```
[1] 1
```

```
[1] 2
[1] 3
[1] 4
[1] 5
[1] 6
[1] 7
[1] 8
[1] 9
[1] 10
[1] 11
[1] 12
[1] 13
[1] 14
[1] 15
[1] 16
```

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

```
   shrinkage interaction.depth n.minobsinnode bag.fraction optimal_trees
1      0.100                 5              5          0.8            96
2      0.100                 5              5          0.7           111
3      0.100                 5             10          0.8            98
4      0.100                 5             10          0.7           150
5      0.001                 5             10          0.7          4000
6      0.001                 5              5          0.7          4000
7      0.001                 5              5          0.8          4000
8      0.001                 5             10          0.8          4000
9      0.100                 1             10          0.7           293
10     0.100                 1             10          0.8           468
   min_RMSE
1     0.147
2     0.147
3     0.148
4     0.150
5     0.150
6     0.150
7     0.150
8     0.151
9     0.151
10    0.152
```

```r
best.model <- hyper_grid %>%
  arrange(min_RMSE) %>%
  head(1)
best.model
```

```
  shrinkage interaction.depth n.minobsinnode bag.fraction optimal_trees
1       0.1                 5              5          0.8            96
  min_RMSE
1    0.147
```

- Let's re-run the GBM model with optimal hyper parameters

```
hit_gbm.final <- gbm(
  formula = SalePrice ~ .,
  data = house_train,
  distribution = "gaussian",
  n.trees = 4000,
  interaction.depth = best.model$interaction.depth,
  shrinkage = best.model$shrinkage,
  n.minobsinnode = best.model$n.minobsinnode,
  bag.fraction = best.model$bag.fraction,
  cv.folds = 10)
# find index for number trees with minimum CV error
best <- which.min(hit_gbm.final$cv.error)
# get MSE and compute RMSE
sqrt(hit_gbm.final$cv.error[best])
```

```
[1] 0.149
```

- Make predictions on the test data

```
pred.gbm.final <- predict.gbm(hit_gbm.final, n.trees=4000, newdata = house_test)
rmse.gbm.final.rmse <- sqrt(mean((house_test$SalePrice -
                          pred.gbm.final)^2))
rmse.gbm.final.rmse
```

```
[1] 0.205
```

```
pred.gbm.final
```

```
  [1] 12.1 12.6 11.7 11.2 11.4 12.0 12.1 11.6 12.3 11.7 11.7 12.2 12.1 12.6 12.1
 [16] 12.2 12.0 12.3 11.7 12.4 12.1 12.1 11.6 11.9 12.0 12.1 11.8 12.6 12.0 11.7
 [31] 12.2 12.3 11.9 11.9 11.6 11.9 12.3 12.0 12.1 12.3 12.1 11.7 11.8 12.3 12.8
 [46] 11.9 12.4 12.3 12.3 12.2 12.0 11.7 12.4 12.1 11.9 12.5 11.4 11.6 11.7 12.2
 [61] 12.5 12.8 12.6 11.6 11.0 12.1 12.3 12.2 11.8 11.7 12.1 12.4 11.8 11.7 12.9
 [76] 12.2 11.5 11.5 12.2 12.0 12.0 12.6 11.9 11.9 11.9 12.2 12.2 12.7 12.3 11.8
 [91] 13.1 12.2 11.4 11.7 11.7 11.8 11.6 11.9 11.5 11.4 12.4 12.4 11.9 11.8 11.5
[106] 11.8 12.2 11.5 12.7 11.9 12.4 12.0 12.6 12.5 11.9 11.9 11.7 12.3 11.6 11.7
[121] 12.2 12.4 12.5 12.7 11.9 12.6 11.9 12.4 12.2 12.4 12.3 11.5 12.5 11.7 12.4
[136] 12.0 12.1 12.4 12.8 11.8 12.3 12.9 11.6 11.0 12.6 11.9 11.8 11.6 12.5 12.4
[151] 11.8 12.0 12.4 11.7 12.7 12.1 11.8 11.7 12.1 11.4 12.0 11.9 11.7 12.3 11.9
[166] 11.7 12.4 11.8 12.3 11.5 12.2 11.7 11.9 12.0 12.2 11.6 12.1 11.7 11.8 12.1
[181] 11.8 12.2 12.0 12.1 12.0 12.7 12.6 12.3 12.1 11.9 11.7 11.7 11.7 11.6 12.7
[196] 12.1 10.8 11.5 12.3 12.1 12.8 12.0 11.6 12.5 11.9 11.8 11.9 11.8 12.7 11.5
[211] 12.1 11.9 12.1 11.7 12.0 11.4 12.6 11.8 12.1 11.6 12.2 11.9 11.9 11.5 12.1
[226] 12.3 12.4 11.8 11.6 12.5 11.8 12.2 11.6 12.0 12.8 11.4 11.7 11.6 12.2 12.3
[241] 12.5 12.0 12.0 13.1 12.0 11.8 12.2 11.9 12.4 11.8 11.9 12.1 11.7 11.9 12.1
[256] 12.8 11.9 12.1 12.2 13.0 11.9 12.0 12.0 11.8 11.9 11.9 12.5 12.1 12.3 11.4
[271] 11.1 11.8 11.4 11.8 11.5 11.7 12.0 12.6 12.6 12.3 11.9 11.6 12.3 11.7 12.4
[286] 12.7 12.0 11.6 12.8 12.5 12.1 11.8
```

```r
#store them in Score
Score["Boosting","RMSE"] = RMSE(pred.gbm.final, house_test$SalePrice)
Score
```

```
                   RMSE
Linear Regression 0.184
Random Forest     0.192
Elastic Net       0.191
Boosting          0.205
```

```r
test <- read.csv("test.csv")
test <- test[,!colnames(test) %in% c('Id','MSZoning','Street','Alley',
                                     'LotShape', 'LandContour', 'Utilities',
                                     'LotConfig', 'LandSlope', 'Neighborhood',
                                     'Condition1', 'Condition2', 'BldgType',
                                     'HouseStyle', 'RoofStyle', 'RoofMatl',
                                     'Exterior1st', 'Exterior2nd', 'MasVnrType',
                                     'ExterQual','ExterCond', 'Foundation',
                                     'BsmtQual', 'BsmtCond', 'BsmtExposure',
                                     'BsmtFinType1', 'BsmtFinType2', 'Heating',
                                     'HeatingQC', 'CentralAir', 'Electrical',
                                     'KitchenQual', 'Functional', 'FireplaceQu',
                                     'GarageType', 'GarageFinish', 'GarageQual',
                                     'GarageCond', 'PavedDrive', 'PoolQC' ,
                                     'Fence', 'MiscFeature', 'SaleType',
                                     'SaleCondition', 'MasVnrArea')]
```

```r
final.fit=regsubsets(SalePrice~.-FullBath-GarageArea-TotRmsAbvGrd,
                     data=train[,-index],nvmax =7)
```

Reordering variables and trying again:

```r
final_model <- coef(final.fit,best.subset.model)
final_model
```

```
 (Intercept)   MSSubClass  OverallQual  OverallCond    YearBuilt BsmtHalfBath
    12.02405     -0.04467      0.26503      0.04150      0.08609      0.00795
    HalfBath  OpenPorchSF
     0.03783      0.02388
```

```r
#test data
final_test=model.matrix(SalePrice~.-FullBath-GarageArea-TotRmsAbvGrd,
                        data=train[,-index])

final_test_lm = final_test[,names(final_model)]%*%final_model
```