Homework 4

Question 1

For each statement write if it is true or false and provide a one sentence explanation. You will get points only if your explanation is correct.

1. Subset selection hierarchy:

   a. The predictors in the k-variable model identified by *forward stepwise* are a subset of the predictors in the $(k+1)$-variable model identified by *forward stepwise* selection.

   True - this is how forward stepwise works. It adds one at a time, so the $(k+1)$-variable model will be the exact same as the k-variable model, with one extra predictor.

   b. The predictors in the k-variable model identified by *backward stepwise* are a subset of the predictors in the $(k+1)$-variable model identified by *backward stepwise* selection.

   True - backward stepwise removes one predictor at a time. So to get from $(k+1)$-down to k, we remove one of the variables, thus we have a subset.

   c. The predictors in the k-variable model identified by *best subset* are a subset of the predictors in the $(k+1)$-variable model identified by *best subset* selection.

   False - for an example, look at the diabetes dataset.

   d. The predictors in the k-variable model identified by *best subset* are a subset of the predictors in the $(k+1)$-variable model identified by *forward stepwise* selection.

   False - although this might happen in some examples, it is not guaranteed to happen. For example, forward stepwise can be greedy enough that its first variable means it completely avoids the true best two variable solution in creating its own three variable solution.

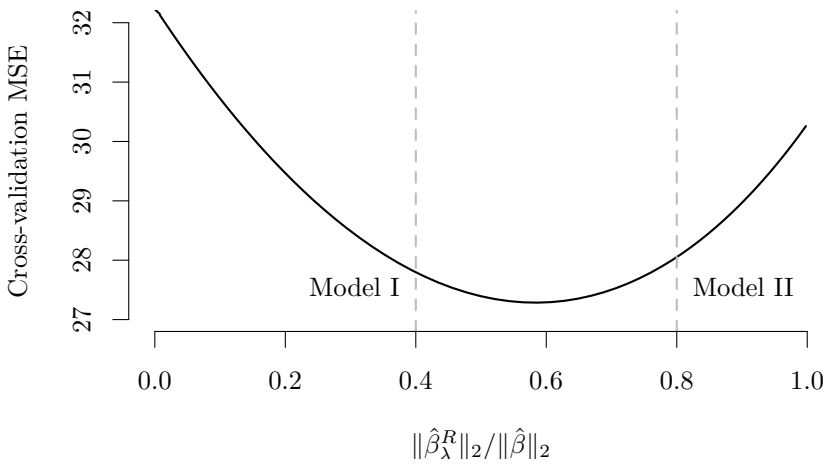
   e. The 1-variable model identified by *best subset* is the same as identified by *forward stepwise* selection.

   True - forward stepwise does the exact same search as best does for $k = 1$.

   f. The 1-variable model identified by *backward stepwise* is the same as identified by *forward stepwise* selection.

   False - Backward stepwise greedily throws away variables because their addition is not useful at some given point (because you judge it relative to other variables) - that does not mean they will not be useful later.

2. The following plot shows the cross validation scores for a ridge regression model with different values of the tuning parameter λ. Note that here what is plotted as the x-axis is the ratio of the ℓ_2 norms of the ridge regression estimate $\hat{\beta}^R_\lambda$ and the least squares estimate $\hat{\beta}$. The two dashed lines correspond to two values of λ, and hence two models.



a. Model I uses a larger λ than Model II.

True - first, when $\lambda = 0$ we get the full model, which is shown on the right of the plot. When λ is huge, we get no predictors, which is shown on the left of the plot. Or more directly, we can see that Model I has a lower value of $\sum_j \beta_j^2$ than than model II. This means that model I was created with a larger penalty on the size of the coefficients.

b. Model I has a higher bias than Model II.

True - Since a larger value of λ is used to train Model I compared to Model II, coefficients in Model I are more restricted, resulting in a higher bias.

c. Model I has a larger in-sample RSS than Model II.

True - The tuning parameter λ balances the in-sample loss with the size of the coefficients. When λ grows, we accept a worse (higher) RSS for a better (lower) penalty value. Your coefficients would not grow for no reason—they grow to lower the RSS.

d. Model I may have a larger test error than Model II.

True - For a given test set, there is some best value of λ, or equivalently some best model, but we do not know it.

Question 2

You can download files for this assignment from
https://www.kaggle.com/c/busn41204-w2021-hw5-housingprice

Filename	Description
`housing_train.csv`	contains data that you will use to build your model
`housing_test.csv`	contains data for which you will make predictions using your model
`sampleSubmission.csv`	example submission in the correct format
`DataDocumentation.txt`	a brief description of the variables

The data set contains prices of residential homes in Ames, Iowa. The data were cleaned and the categorical variables were converted to indicators.

Load the data using:

```
housing.train = read.csv('housing_train.csv')
housing.test = read.csv('housing_test.csv')
```

Load libraries

```
library(ggplot2)
library(dplyr)
library(bestglm)
library(glmnet)
library(Boruta)
library(xgboost)
```

1. Create
   - a histogram of `SalePrice`;
   - a histogram of `log(Sale_Price)`;
   - a scatterplot between `SalePrice` and `Gr_Liv_Area`, the overall quality scores of the homes;
   - a scatterplot between `log(Sale_Price)` and `Gr_Liv_Area`.

   Comment on what you observe. Which one do you think would be more appropriate as the response of a linear regression model, `SalePrice` or `log(Sale_Price)` ?

Plot

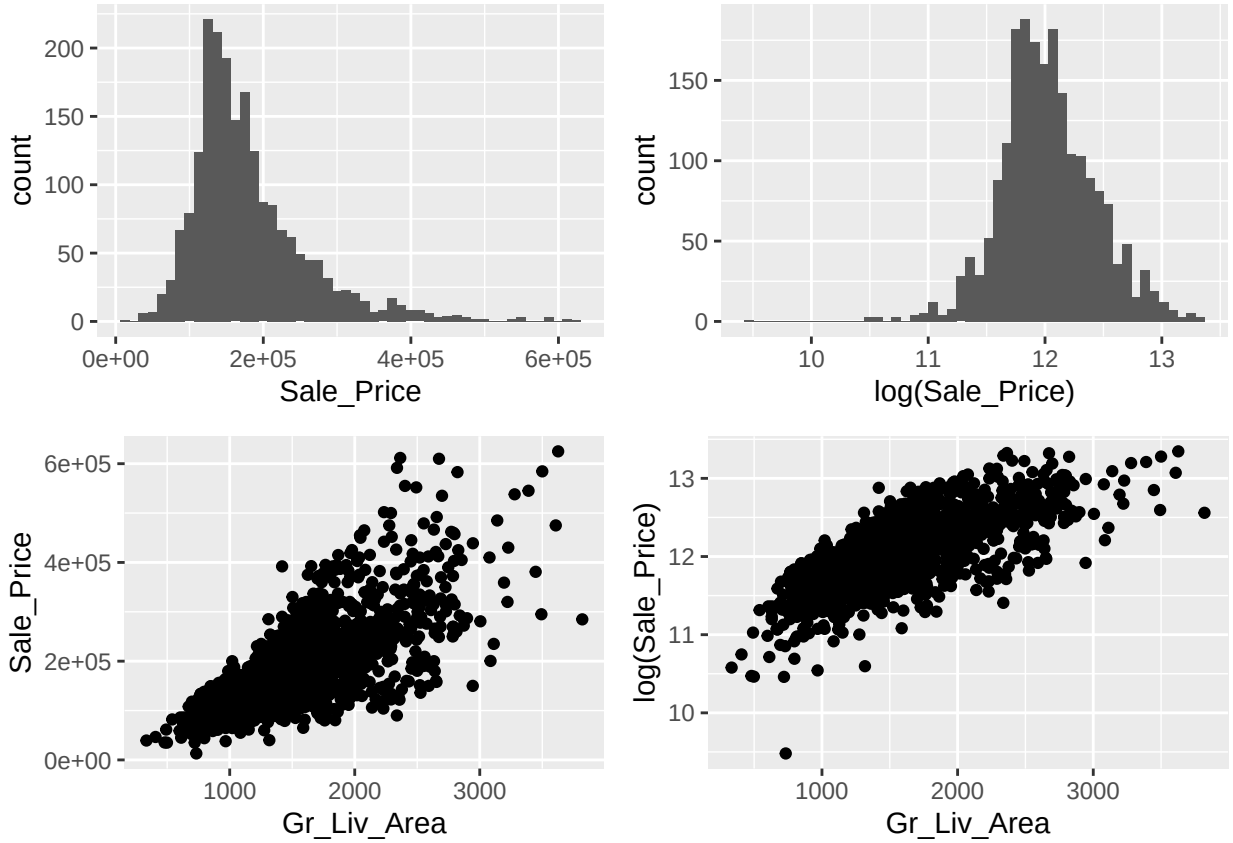
```
p1 <- housing.train %>%
   ggplot( aes(x=Sale_Price) ) +
    geom_histogram(bins=50)

p2 <- housing.train %>%
  ggplot( aes(x=log(Sale_Price)) ) +
    geom_histogram(bins=50)

p3 <- housing.train %>%
   ggplot( aes(y=Sale_Price, x=Gr_Liv_Area) ) +
     geom_point()

p4 <- housing.train %>%
   ggplot( aes(y=log(Sale_Price), x=Gr_Liv_Area) ) +
     geom_point()
```

```
gridExtra::grid.arrange(p1, p2, p3, p4, ncol = 2)
```



Based on these plots, `log(Sale_Price)` is a less skewed distribution (looks close to normal), while `Sale_Price` has a long right tail/right skew. We get heteroskedasticity with the raw version and some evidence of non-linearity, while the log version looks good against `Gr_Liv_Area`. I would choose the log version as the more appropriate response in a linear model.

2. Fit a linear regression on `log(Sale_Price)` using forward stepwise selection.

   - Create a plot displaying the cross validation score as a function of the number of predictors.
   - Create a plot displaying BIC scores as a function of the number of predictors.
   - How many variables get selected in the chosen model :
     - by cross validation;
     - by 1-sd rule;
     - by BIC.

We create a data frame with `y` as `log(Sale_Price)`

```
X = housing.train[, 1:(ncol(housing.train)-1)]
y = log(housing.train$Sale_Price)
Xy = cbind(X, y)
```

Run forward regression

```
fwd.cv = bestglm(Xy,
         family = gaussian,
         IC     = "CV",
```

```
         CVArgs = list(Method="HTF", K=5, REP=5),
         method = "forward")
```

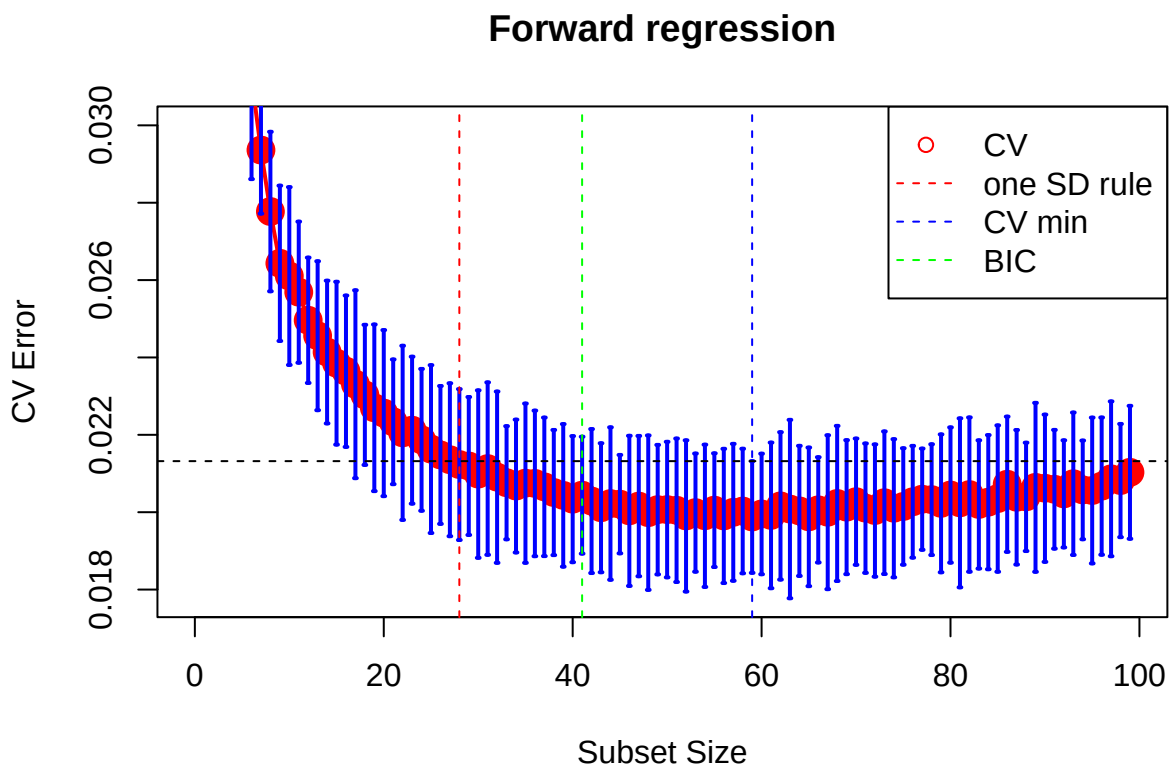
```
fwd.bic = bestglm(Xy,
                  family = gaussian,
                  IC     = "BIC",
                  method = "forward")
```

Create the plot. This is almost identical as in the diabetes example.

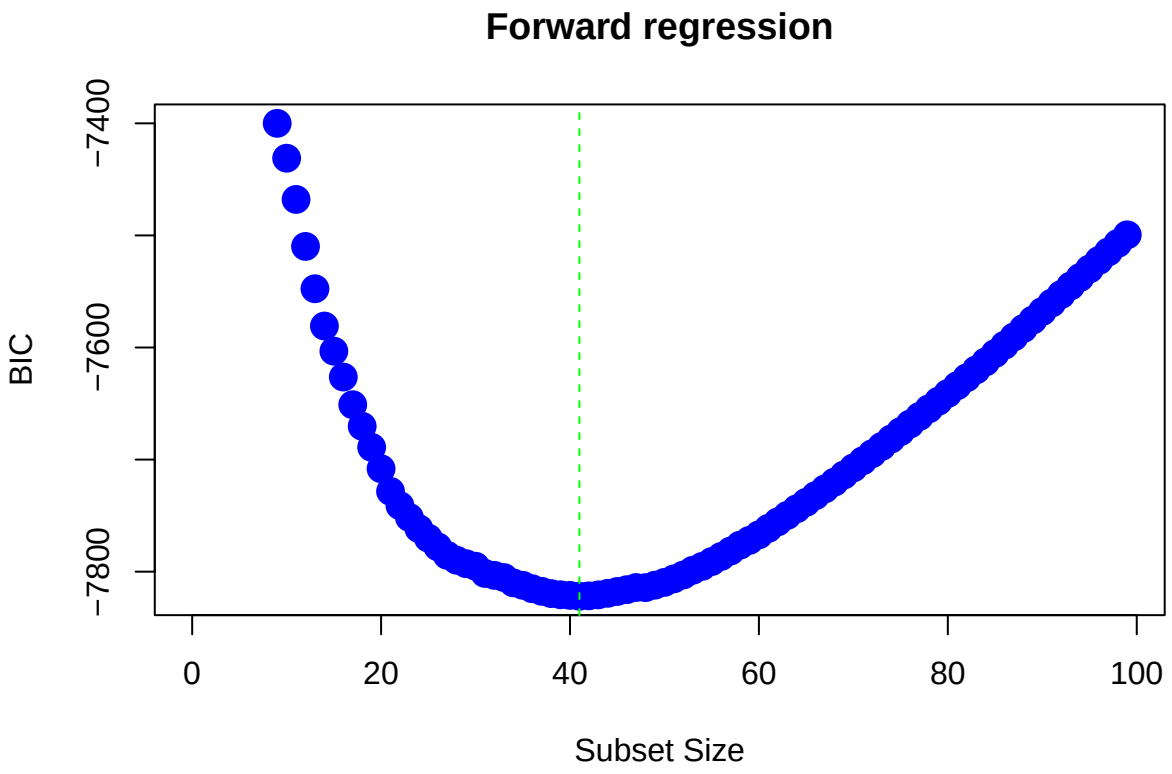
```
cverrs = fwd.cv$Subsets$CV
sdCV = fwd.cv$Subsets$sdCV
CVLo = cverrs - sdCV
CVHi = cverrs + sdCV
k = 0:(length(cverrs)-1)
plot(k, cverrs, xlab="Subset Size", ylab="CV Error", main='Forward regression',
     ylim=c(min(CVLo), 0.03),     # we can better see the relevant part of the CV curve
     type="n")
points(k, cverrs, cex=2, col="red", pch=16)
lines(k, cverrs, col="red", lwd=2)
# plot error bars
segments(k, CVLo, k, CVHi, col="blue", lwd=2)
eps = 0.15
segments(k-eps, CVLo, k+eps, CVLo, col="blue", lwd=2)
segments(k-eps, CVHi, k+eps, CVHi, col="blue", lwd=2)

indBest = oneSDRule(fwd.cv$Subsets[,c("CV", "sdCV")])
abline(v=indBest-1, lty=2, col='red')
indMin = which.min(cverrs)
fmin = sdCV[indMin]
cutOff = fmin + cverrs[indMin]
abline(h=cutOff, lty=2)
abline(v=indMin-1, lty=2, col='blue')

indBic = which.min(fwd.bic$Subsets$BIC)
abline(v=indBic-1, lty=2, col='green')
legend("topright",
       legend=c("CV", "one SD rule", "CV min", "BIC"),
       col=c("red", "red", "blue", "green"),
       pch=c(1,NA,NA,NA), lty=c(NA,2,2,2))
```




```
bic.plot = fwd.bic$Subsets$BIC
plot(k, bic.plot, xlab="Subset Size", ylab="BIC", main='Forward regression',
     ylim=c(min(bic.plot), -7400),    # we can better see the relevant part of the BIC curve
     type="n")
points(k, bic.plot, cex=2, col="blue", pch=16)
abline(v=indBic-1, lty=2, col='green')
```



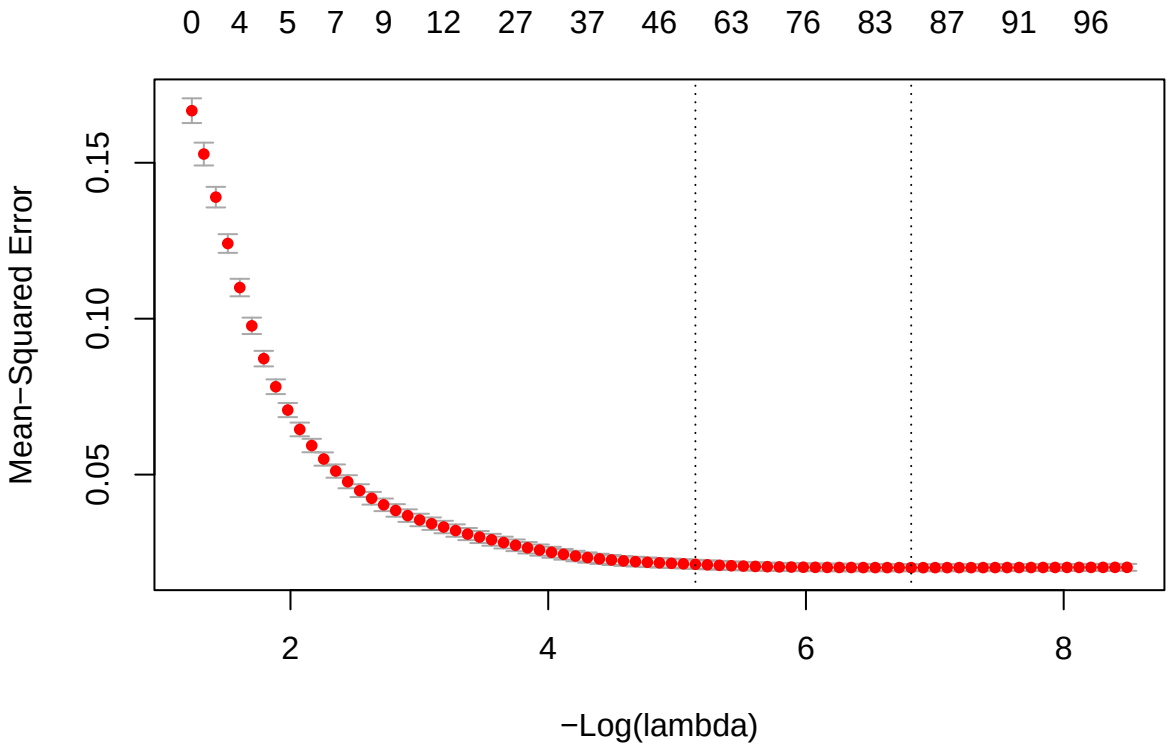
The number of variables for different selection schemes is as follows:

```
data.frame(oneSDRule=indBest-1, CV=indMin-1, BIC=indBic-1)
```

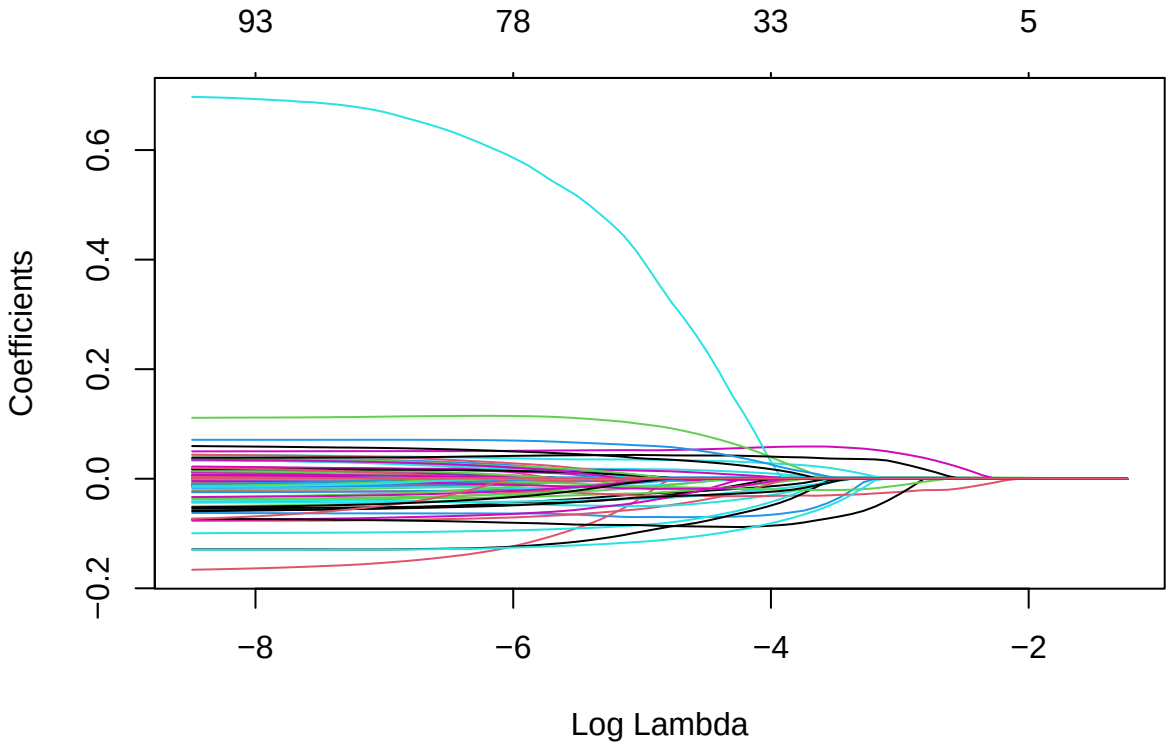
```
##   oneSDRule CV BIC
## 1        28 59  41
```

3. Fit a linear regression on `log(Sale_Price)` using the lasso.
    - Create a plot displaying the cross validation score as a function of λ.
    - Create a plot displaying the coefficient values as a function of λ.
    - How many variables get selected in the chosen model:
        - by cross validation;
        - by 1-sd rule.

```
lasso.cv = cv.glmnet(as.matrix(X), y, family="gaussian")
plot(lasso.cv, sign.lambda=-1)
```



```
plot(lasso.cv$glmnet.fit, xvar='lambda')
```



The number of variables for two different λ values isvariable selection schemes is as follows:

```
coef.1se = coef(lasso.cv, s=lasso.cv$lambda.1se)
coef.min = coef(lasso.cv, s=lasso.cv$lambda.min)

# count number of non-zeros
nnz.1se = length( rownames(coef.1se)[coef.1se[,1] != 0] ) - 1
nnz.min = length( rownames(coef.min)[coef.min[,1] != 0] ) - 1

data.frame(oneSDRule=nnz.1se, CV=nnz.min)
```

```
##   oneSDRule CV
## 1        57 85
```

4

. Comment on the 5 models found in parts 2 and 3. Which one seems most promising for out-of-sample prediction?

Let us take a look at different error estimates.

```
indMin.lasso = which(lasso.cv$lambda == lasso.cv$lambda.min)
indBest.lasso = which(lasso.cv$lambda == lasso.cv$lambda.1se)

tb = data.frame(CV_err = c(fwd.cv$Subsets[indMin,'CV'],
                           fwd.cv$Subsets[indBic,'CV'],
                           fwd.cv$Subsets[indBest,'CV'],
                           lasso.cv$cvm[indBest.lasso],
                           lasso.cv$cvm[indMin.lasso]),
           CV_std = c(fwd.cv$Subsets[indMin,'sdCV'],
                      fwd.cv$Subsets[indBic,'sdCV'],
                      fwd.cv$Subsets[indBest,'sdCV'],
                      lasso.cv$cvsd[indBest.lasso],
                      lasso.cv$cvsd[indMin.lasso]),
           NumNonZero = c(indMin-1,
                          indBic-1,
                          indBest-1,
                          lasso.cv$glmnet.fit$df[indMin.lasso],
                          lasso.cv$glmnet.fit$df[indBest.lasso])
           )
rownames(tb) = c('min CV', 'BIC', 'one SD rule', 'min CV Lasso', 'one SD rule Lasso')

tb
```

```
##                   CV_err  CV_std NumNonZero
## min CV            0.0199 0.00144         59
## BIC               0.0204 0.00151         41
## one SD rule       0.0212 0.00195         28
## min CV Lasso      0.0212 0.00146         85
## one SD rule Lasso 0.0202 0.00115         57
```

We have seen that the CV curve is rather flat. I would go with the simplest model, that is, the one that has the smallest number of variables. Here, this would be one obtained by forward subset regression and one standard deviation rule.

5

. Build a model to predict the final sale prices in `housing_test.csv`.

This does not need to be a linear model. Explore some variable selection procedures for nonparametric methods.

First, let us split data into two parts. The first to rank variables on. The second one to train models with different values of tuning parameters.

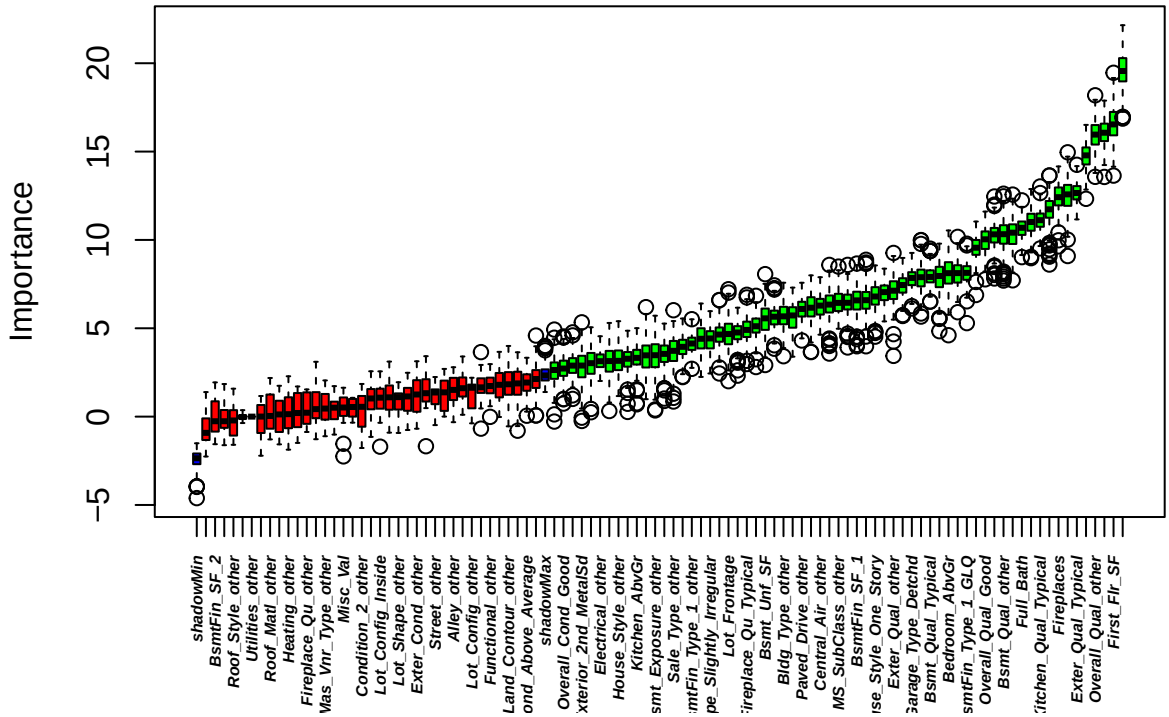
```
set.seed(1)
inVariableRanking = sample(nrow(Xy), 700)
```

Use `Boruta` on roughly 30% of observations. Note that any other procedure for ranking of variables could be used as well. I'm setting `maxRuns=1000` as I really want to resolve `Tentative` variables.

```
ranking.boruta <- Boruta(y~., data=Xy[inVariableRanking,], maxRuns=1000, doTrace=0)
ranking.boruta
```

```
## Boruta performed 291 iterations in 2.09 mins.
##  63 attributes confirmed important: Bedroom_AbvGr, Bldg_Type_other, Bsmt_Exposure_No,
## Bsmt_Exposure_other, Bsmt_Full_Bath and 58 more;
##  36 attributes confirmed unimportant: Alley_other, Bsmt_Cond_other, Bsmt_Half_Bath, BsmtFin_SF_2,
## Condition_1_other and 31 more;
```

```
plot(ranking.boruta, xlab="", xaxt="n")
lz<-lapply(1:ncol(ranking.boruta$ImpHistory), function(i)
              ranking.boruta$ImpHistory[is.finite(ranking.boruta$ImpHistory[, i]), i])
names(lz)<-colnames(ranking.boruta$ImpHistory)
lb<-sort(sapply(lz, median))
axis(side=1, las=2, labels=names(lb),
     at=1:ncol(ranking.boruta$ImpHistory), cex.axis=0.5, font = 4)
```



These are all confirmed and tentative variables identified by Boruta.

```
potential_variables = names(ranking.boruta$finalDecision)[
   ranking.boruta$finalDecision %in% c("Confirmed", "Tentative")
   ]
potential_variables
```

```
##  [1] "Lot_Frontage"                          "Lot_Area"
##  [3] "Year_Built"                            "Year_Remod_Add"
##  [5] "Mas_Vnr_Area"                          "BsmtFin_SF_1"
##  [7] "Bsmt_Unf_SF"                           "First_Flr_SF"
##  [9] "Bsmt_Full_Bath"                        "Full_Bath"
## [11] "Half_Bath"                             "Bedroom_AbvGr"
## [13] "Kitchen_AbvGr"                         "TotRms_AbvGrd"
## [15] "Fireplaces"                            "Garage_Area"
## [17] "Wood_Deck_SF"                          "Open_Porch_SF"
## [19] "Screen_Porch"                          "Longitude"
## [21] "Latitude"                              "MS_SubClass_other"
## [23] "MS_Zoning_Residential_Medium_Density" "MS_Zoning_other"
## [25] "Lot_Shape_Slightly_Irregular"          "Neighborhood_other"
## [27] "Bldg_Type_other"                       "House_Style_One_Story"
## [29] "House_Style_other"                     "Overall_Qual_Above_Average"
## [31] "Overall_Qual_Good"                     "Overall_Qual_Very_Good"
## [33] "Overall_Qual_other"                    "Overall_Cond_Good"
## [35] "Overall_Cond_other"                    "Roof_Style_Hip"
## [37] "Exterior_2nd_MetalSd"                  "Exterior_2nd_VinylSd"
## [39] "Exterior_2nd_Wd.Sdng"                  "Exter_Qual_Typical"
## [41] "Exter_Qual_other"                      "Foundation_CBlock"
## [43] "Bsmt_Qual_Typical"                     "Bsmt_Qual_other"
## [45] "Bsmt_Exposure_No"                      "Bsmt_Exposure_other"
## [47] "BsmtFin_Type_1_GLQ"                    "BsmtFin_Type_1_other"
## [49] "Heating_QC_Typical"                    "Central_Air_other"
## [51] "Electrical_other"                      "Kitchen_Qual_Typical"
## [53] "Kitchen_Qual_other"                    "Fireplace_Qu_Typical"
## [55] "Garage_Type_Detchd"                    "Garage_Type_other"
## [57] "Garage_Finish_RFn"                     "Garage_Finish_Unf"
## [59] "Garage_Finish_other"                   "Paved_Drive_other"
## [61] "Sale_Type_other"                       "Sale_Condition_other"
## [63] "Gr_Liv_Area"
```

We rank them according to `median(ImpHistory)`.

```
ranked_variables = sort( sapply(lz, median), decreasing = T )
ranked_variables = ranked_variables[names(ranked_variables) %in% potential_variables]
ranked_variables
```

```
##                        Gr_Liv_Area                         First_Flr_SF
##                              19.56                                16.54
##                        Garage_Area                   Overall_Qual_other
##                              16.07                                15.96
##                         Year_Built                   Exter_Qual_Typical
##                              14.79                                12.66
##                           Lot_Area                           Fireplaces
##                              12.58                                12.42
##                     Year_Remod_Add                  Kitchen_Qual_Typical
##                              11.73                                11.12
```

```
##                       TotRms_AbvGrd                                 Full_Bath
##                               11.01                                     10.69
##                        Mas_Vnr_Area                           Bsmt_Qual_other
##                               10.41                                     10.32
##                           Longitude                          Overall_Qual_Good
##                               10.32                                     10.03
##                 Overall_Qual_Very_Good                        BsmtFin_Type_1_GLQ
##                                9.56                                      8.14
##                        Open_Porch_SF                             Bedroom_AbvGr
##                                8.12                                      8.12
##                    Kitchen_Qual_other                         Bsmt_Qual_Typical
##                                7.96                                      7.91
##                            Latitude                        Garage_Type_Detchd
##                                7.89                                      7.85
##            Overall_Qual_Above_Average                         Exter_Qual_other
##                                7.45                                      7.13
##                     Garage_Finish_Unf                  House_Style_One_Story
##                                7.05                                      6.80
##                             Half_Bath                           BsmtFin_SF_1
##                                6.59                                      6.58
##                        Roof_Style_Hip                        MS_SubClass_other
##                                6.44                                      6.44
## MS_Zoning_Residential_Medium_Density                        Central_Air_other
##                                6.34                                      6.28
##                          Wood_Deck_SF                        Paved_Drive_other
##                                6.16                                      6.09
##                       Bsmt_Full_Bath                          Bldg_Type_other
##                                5.73                                      5.67
##                    Garage_Finish_other                             Bsmt_Unf_SF
##                                5.66                                      5.56
##                     Foundation_CBlock                    Fireplace_Qu_Typical
##                                5.12                                      4.91
##                     Heating_QC_Typical                          Lot_Frontage
##                                4.76                                      4.67
##                   Exterior_2nd_VinylSd       Lot_Shape_Slightly_Irregular
##                                4.65                                      4.42
##                      Bsmt_Exposure_No               BsmtFin_Type_1_other
##                                4.41                                      4.14
##                    Neighborhood_other                       Sale_Type_other
##                                3.95                                      3.72
##                    Garage_Finish_RFn                   Bsmt_Exposure_other
##                                3.55                                      3.48
##                  Sale_Condition_other                          Kitchen_AbvGr
##                                3.47                                      3.33
##                       MS_Zoning_other                      House_Style_other
##                                3.27                                      3.16
##                     Garage_Type_other                       Electrical_other
##                                3.14                                      3.14
##                           Screen_Porch                   Exterior_2nd_MetalSd
##                                3.03                                      2.89
##                  Exterior_2nd_Wd.Sdng                      Overall_Cond_Good
##                                2.89                                      2.72
##                     Overall_Cond_other
##                                2.64
```

Now let us run xgboost with the number of variables as a tuning parameter.

```
# create hyperparameter grid
hyper_grid <- expand.grid(
  number_of_var = c(30, 40, 45, length(ranked_variables)),
  shrinkage = c(.01, 0.05, .1, .2),     ## controls the learning rate
  interaction.depth = c(3, 5, 7),       ## tree depth
  bag.fraction = c(.5, .65, .8),        ## percent of training data to sample for each tree
  optimal_trees = 0,               # a place to dump results
  min_RMSE = 0                     # a place to dump results
)
```

This will take a lot of time.

```
for(i in 1:nrow(hyper_grid)) {
  # create parameter list
  params <- list(
    eta = hyper_grid$shrinkage[i],
    max_depth = hyper_grid$interaction.depth[i],
    subsample = hyper_grid$bag.fraction[i]
  )

  vars_to_include = names(ranked_variables)[1:hyper_grid$number_of_var[i]]
  X.train = as.matrix( Xy[-inVariableRanking, vars_to_include] )
  Y.train = Xy[-inVariableRanking, 'y']

  # train model
  xgb.tune <- xgb.cv(
    params = params,
    data = X.train,
    label = Y.train,
    nrounds = 5000,
    nfold = 5,
    objective = "reg:squarederror",     # for regression models
    verbose = 0,                        # 0 - silent,
    verbosity = 0,
    early_stopping_rounds = 10          # stop if no improvement for 10 consecutive trees
  )

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

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

```
##    number_of_var shrinkage interaction.depth bag.fraction optimal_trees min_RMSE
## 1             63      0.01                 3         0.80          1842    0.136
## 2             63      0.01                 3         0.65          2117    0.136
## 3             63      0.05                 3         0.80           488    0.137
## 4             63      0.01                 7         0.65          1016    0.137
## 5             63      0.01                 3         0.50          1807    0.137
## 6             40      0.01                 5         0.50          1184    0.139
## 7             63      0.01                 5         0.50          1343    0.139
```

```
## 8             63      0.01                5         0.65          1418    0.139
## 9             63      0.05                3         0.65           418    0.139
## 10            63      0.05                5         0.50           272    0.139
```

Let us now refit on all data.

```
vars_to_include = names(ranked_variables)[1:oo[1,]$number_of_var]
vars_to_include
```

```
##  [1] "Gr_Liv_Area"                          "First_Flr_SF"
##  [3] "Garage_Area"                          "Overall_Qual_other"
##  [5] "Year_Built"                           "Exter_Qual_Typical"
##  [7] "Lot_Area"                             "Fireplaces"
##  [9] "Year_Remod_Add"                       "Kitchen_Qual_Typical"
## [11] "TotRms_AbvGrd"                        "Full_Bath"
## [13] "Mas_Vnr_Area"                         "Bsmt_Qual_other"
## [15] "Longitude"                            "Overall_Qual_Good"
## [17] "Overall_Qual_Very_Good"               "BsmtFin_Type_1_GLQ"
## [19] "Open_Porch_SF"                        "Bedroom_AbvGr"
## [21] "Kitchen_Qual_other"                   "Bsmt_Qual_Typical"
## [23] "Latitude"                             "Garage_Type_Detchd"
## [25] "Overall_Qual_Above_Average"           "Exter_Qual_other"
## [27] "Garage_Finish_Unf"                    "House_Style_One_Story"
## [29] "Half_Bath"                            "BsmtFin_SF_1"
## [31] "Roof_Style_Hip"                       "MS_SubClass_other"
## [33] "MS_Zoning_Residential_Medium_Density" "Central_Air_other"
## [35] "Wood_Deck_SF"                         "Paved_Drive_other"
## [37] "Bsmt_Full_Bath"                       "Bldg_Type_other"
## [39] "Garage_Finish_other"                  "Bsmt_Unf_SF"
## [41] "Foundation_CBlock"                    "Fireplace_Qu_Typical"
## [43] "Heating_QC_Typical"                   "Lot_Frontage"
## [45] "Exterior_2nd_VinylSd"                 "Lot_Shape_Slightly_Irregular"
## [47] "Bsmt_Exposure_No"                     "BsmtFin_Type_1_other"
## [49] "Neighborhood_other"                   "Sale_Type_other"
## [51] "Garage_Finish_RFn"                    "Bsmt_Exposure_other"
## [53] "Sale_Condition_other"                 "Kitchen_AbvGr"
## [55] "MS_Zoning_other"                      "House_Style_other"
## [57] "Garage_Type_other"                    "Electrical_other"
## [59] "Screen_Porch"                         "Exterior_2nd_MetalSd"
## [61] "Exterior_2nd_Wd.Sdng"                 "Overall_Cond_Good"
## [63] "Overall_Cond_other"
```

```
X.all = as.matrix( Xy[, vars_to_include] )
Y.all = Xy[, 'y']

# parameter list
params <- list(
  eta = oo[1,]$shrinkage,
  max_depth = oo[1,]$interaction.depth,
  subsample = oo[1,]$bag.fraction
)

# train final model
xgb.fit.final <- xgboost(
  params = params,
  data = X.all,
```

```
  label = Y.all,
  nrounds = oo[1,]$optimal_trees,
  objective = "reg:squarederror",
  verbose = 0
)
```

We make predictions on the test set.

```
yhat.xgb <- predict(xgb.fit.final, newdata=as.matrix(housing.test[,vars_to_include]))
write.csv(data.frame(Id=1:length(yhat.xgb), Sale_Price=exp(yhat.xgb)),
          file = "mkolar_submission.csv",
          row.names = FALSE,
          quote = FALSE)
```