The honor code.

(a). Zheng Wu

this hw is quite time consuming. Dealing with dummy variables takes a large amount of time.

(b). I hereby state that all of my solutions were entirely in my words and were written by me. I have not looked at other student's solutions and I have fairly credited all external sources in this write up.

Stat154 hw5.

1. True or False.

(a) False.

L1 Regularization will not prevent bias but provide a bias-variance trade-off.

(b) False.

Lasso estimator gives sparse solutions. For zeros and non-zeros.

(c) True. Lasso solutions can be many. but $\hat{X}\beta_{lasso}$ will always be the same.

(d) False. scaling and centering are necessary

(e) False. $n > d$, we invert a smaller matrix.

(f) True K is symmetric, $v^T K v \geq 0$.

(h) True. ϕ as long as ϕ do this.

(i) True. $\phi^T \phi$ will only need to be calculated for once.

2. Coordinate descent for Lasso

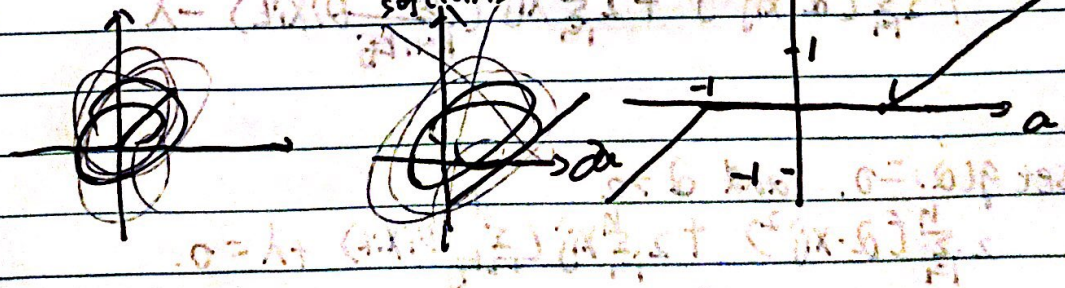
(1) Definition of coordinate descent: fix one parameter, all other optimize over one parameter.

$w = \frac{\lambda}{n}$

that the Lasso formulation with average square loss will give the same output as the total square loss.

(2) $soft(a, \delta) = sign(a)(|a| - \delta)_+$.

if $a > 0$. as the function domain.



It is indeed an non-decreasing function when $a > 0$.

(3). $\min \sum_{i=1}^{n} (x_i^T\theta - y_i)^2 + \lambda\|\theta\|_1$.

θ includes 2 parts, θ_j and θ_k where $k \neq j$.

thus the objective becomes:

$$\min \sum_{i=1}^{n} (x_i^T(\theta_j + \sum_{k=1,k\neq j}^{d} \theta_k - y_i)^2 + \lambda\|\theta_j + \sum_{k=1,k\neq j}^{d} \theta_k\|$$

$$= \min \sum_{i=1}^{n} (x_i^T\theta_j + \sum_{k=1,k\neq j}^{d} \theta_k \cdot x_i^T - y_i)^2 + \lambda|\theta_j| + \lambda\sum_{k=1,k\neq j}^{d}|\theta_k|$$

Substitute all θ_j as a, we obtain the following objective.

$$\sum_{i=1}^{n} (ax_{ij} + \sum_{k=1,k\neq j}^{d} \theta_k x_{ik} - y_i)^2 + \lambda|a| + \lambda\sum_{k=1,k\neq j}^{d}|\theta_k|.$$

thus proved

(4) $g(a) = \sum_{i=1}^{n} [ax_{ij} + \sum_{k=1,k\neq j}^{d} \theta_k x_{ik} - y_i]^2 + \lambda|a| + \lambda\sum_{k=1,k\neq j}^{d}|\theta_k|$.

$g(a)$ is differentiable when $a \neq 0$.

$\Rightarrow$ $g(a)$

~~$\theta_j = \text{soft}(c_j, \frac{\lambda}{a_j})$~~

~~$= \text{sign}(\frac{\theta_j}{a_j}) \left|\frac{c_j}{a_j} - \frac{\lambda}{a_j}\right| +$~~

~~$= \text{sign}(\frac{\theta_j}{a_j}) \left|\frac{c_j - \lambda}{a_j}\right|$~~

$$g(a) = 2\sum_{i=1}^{n} [ax_{ij} + \sum_{k=1,k\neq j}^{d} \theta_k x_{ik} - y_i) \cdot x_{ij} + |\lambda|.$$

$$g'(a) = \begin{cases} 2\sum_{i=1}^{n}[a \cdot x_{ij}^2] + 2\sum_{i=1}^{n} x_{ij}(\sum_{k=1,k\neq j}^{d} \theta_i x_{ik}) + \lambda & \text{for } a > 0 \\ 2\sum_{i=1}^{n}[a \cdot x_{ij}^2] + 2\sum_{i=1}^{n} x_{ij}(\sum_{k=1,k\neq j}^{d} \theta_i x_{ik}) - \lambda & \text{for } a < 0. \end{cases}$$

(5). set $g(a) = 0$. and $a > 0$.

$$2\sum_{i=1}^{n}[a \cdot x_{ij}^2] + 2\sum_{i=1}^{n} x_{ij}(\sum_{k=1,k\neq j}^{d} \theta_i x_{ik}) + \lambda = 0.$$

where $a_j = 2\sum_{i=1}^{n} x_{ij}^2$, $c_j = 2\sum_{i=1}^{n} x_{ij}(y_i - \sum_{k=1,k\neq j}^{d} \theta_k x_{ik})$.

thus $a \cdot a_j - c_j + \lambda = 0$.

$$a = \frac{c_j - \lambda}{a_j}.$$

(6). similarly. $g(\alpha) = 0$ and $\alpha < 0$

$$a_j \cdot \alpha - c_j - \lambda = 0$$

$$\alpha = \frac{c_j + \lambda}{a_j}$$

(7). for α^* to be positive, $c_j > \lambda$

for α^* to be negative, $c_j < -\lambda$.

(8) $D^+ g(\alpha) = \lim_{\theta \to \alpha, \varepsilon \to 0} \frac{g(\alpha+\varepsilon) - g(\alpha)}{\varepsilon} \geq 0$

$D^- g(\alpha) = \lim_{\varepsilon < 0, \varepsilon \to 0} \frac{g(\alpha-\varepsilon) - g(\alpha)}{\varepsilon} \leq 0$

$$D^+ g(\alpha) = \lim_{\varepsilon > 0, \varepsilon \to 0} \frac{\sum_{i=1}^n [(\alpha+\varepsilon)x_{ij} + \sum_{k=1, k\neq j}^d \theta_k x_{ik} - y_i]^2 + \lambda|\alpha+\varepsilon| - \sum_{i=1}^n [\alpha x_{ij} + \sum_{k=1, k\neq j}^d \theta_k x_{ik} - y_i]^2 + \lambda|\alpha|}{\varepsilon}$$

$$= \lim_{\varepsilon > 0, \varepsilon \to 0} \frac{\sum_{i=1}^n [(\alpha+\varepsilon)x_{ij} + \Sigma}{\varepsilon}$$

when $\alpha > 0$, $a_j \alpha - c_j + \lambda = 0$

$$D^+ g(\alpha) = \lim_{\varepsilon > 0, \varepsilon \to 0} \frac{\sum_{i=1}^n [(\alpha+\varepsilon)x_{ij} + \sum_{k=1, k\neq j}^d \theta_k x_{ik} - y_i]^2 + \lambda|\alpha+\varepsilon| - \left[\sum_{i=1}^n [\alpha x_{ij} + \sum_{k=1, k\neq j}^d \theta_k x_{ik} - y_i]^2 + \lambda|\alpha|\right]}{\varepsilon}$$

$$= \lim_{\varepsilon > 0, \varepsilon \to 0} \frac{\sum_{i=1}^n [(\alpha+\varepsilon)^2 x_{ij}^2] + \sum_{i=1}^n (\sum_k \theta_k x_{ik} - y_i)^2 + \sum_{i=1}^n \cdot 2[\alpha+\varepsilon](x_{ij} \sum_{k=1, k\neq j}^d \theta_k x_{ik} - y_i) + \lambda|\alpha+\varepsilon|}{\varepsilon}$$

continous.....

$$- \sum_{i=1}^n \alpha^2 x_{ij}^2 - \sum_{i=1}^n (\sum_k^d \theta_k x_{ik} - y_i)^2 - \sum_{i=1}^n \cdot 2 \cdot \alpha x_{ij} [\sum_{k=1, k\neq j}^d \theta_k x_{ik} - y_i] - \lambda|\alpha|$$

$$= \lim_{\varepsilon > 0, \varepsilon \to 0} \frac{\sum_{i=1}^n [(\alpha+\varepsilon)^2 x_{ij}^2 - \alpha^2 x_{ij}^2] + 2 \sum_{i=1}^n \cdot \varepsilon x_{ij} [\sum_{k=1, k\neq j}^d \theta_k x_{ik} - y_i] + \lambda|\varepsilon|}{\varepsilon}$$

$$= \lim_{\varepsilon > 0, \varepsilon \to 0} \frac{\sum_{i=1}^n [2\alpha \cdot \varepsilon + \varepsilon^2) x_{ij}^2] + 2 \sum_{i=1}^n \varepsilon \cdot x_{ij} [\sum_{k=1, k\neq j}^d \theta_k x_{ik} - y_i] + \lambda|\varepsilon|}{\varepsilon}$$

since $a_j = 2 \sum_{i=1}^n x_{ij}^2 \qquad c_j = 2 \sum_{i=1}^n x_{ij} (y_i - \sum_{k=1, k\neq j}^d \theta_k x_{ik})$

$$\Rightarrow \lim_{\varepsilon > 0, \varepsilon \to 0} \frac{2 \cdot \varepsilon \cdot a_j + \frac{1}{2} \varepsilon^2 \cdot a_j + \cdot \varepsilon \cdot c_j + \lambda|\varepsilon|}{\varepsilon} = \frac{\frac{1}{2} \varepsilon^2 \cdot a_j - \varepsilon c_j + \lambda|\varepsilon|}{\varepsilon}$$

Similarly

$$D^-(g)(\hat{a}) = \frac{\frac{1}{2}\varepsilon^2 \cdot a_j + \varepsilon \cdot (c_j \mp \lambda)\varepsilon}{\varepsilon}$$

(9). if $c_j \in [-\lambda, \lambda]$

then acordy to (7) c_j can

a^* is not positive. a^* is not either negative.

thus $a^* = 0$.

and $D^+(g)(\hat{a}) \geq 0$ $\quad D^-(g)(\hat{a}) \leq 0$

$\Rightarrow D^+(g)(\hat{a}) = 0$

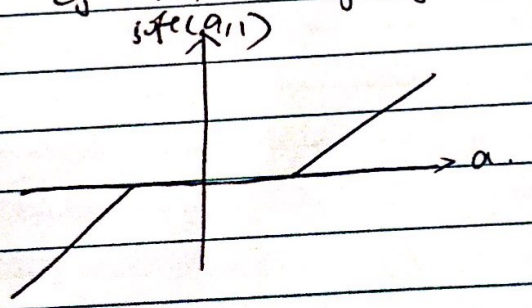
thus $a^* = 0$ is the minimizer of g when $c_j \in [-\lambda, \lambda]$.

(10). acordy to (5)(6)(7)(8)

when $c_j \in [-\lambda, \lambda]$ $\quad a^* = 0$ is the minimizer.

$c_j > \lambda$, $\quad a_j^{-1}(c_j - \lambda)$ is the minimizer

$c_j < -\lambda$, $\quad a_j^{-1}(c_j - \lambda)$ is the minimizer.



they are exactly equavalent functions.

(11) the algorithm is looking for the minimize of the function. when $a^* \neq 0$. simply differentiating the function shall get us the minimizer. when $a^* = 0$. we use sandwich theroom for differentiation and we will get a sparse solution.

stat154_hw4

3.Regression analysis on Ames data

```
Ames <- read.delim("AmesHousing.txt", header = TRUE, sep = "\t", dec = ".")
```

```
continuousVar <- colnames(Ames)[grep("Frontage|SF|Area|Porch",
colnames(Ames))]

AmesTiny <- Ames[, c(continuousVar,
c("Overall.Qual",
"Overall.Cond","Neighborhood",
"SalePrice"))]
# check NA
colSums(is.na(AmesTiny))
```

```
##    Lot.Frontage        Lot.Area    Mas.Vnr.Area    BsmtFin.SF.1
##             490               0              23               1
##    BsmtFin.SF.2     Bsmt.Unf.SF   Total.Bsmt.SF     X1st.Flr.SF
##               1               1               1               0
##     X2nd.Flr.SF Low.Qual.Fin.SF     Gr.Liv.Area     Garage.Area
##               0               0               0               1
##    Wood.Deck.SF   Open.Porch.SF  Enclosed.Porch     X3Ssn.Porch
##               0               0               0               0
##    Screen.Porch       Pool.Area    Overall.Qual    Overall.Cond
##               0               0               0               0
##    Neighborhood       SalePrice
##               0               0
```

```
AmesTiny$Garage.Area[is.na(AmesTiny$Garage.Area)] = 0
# change factor variable to actual factor in the data frame
AmesTiny$Overall.Qual <- factor(AmesTiny$Overall.Qual)
AmesTiny$Overall.Cond <- factor(AmesTiny$Overall.Cond)
# fill the continuous variable with column mean
for(i in 1:ncol(AmesTiny)){
AmesTiny[is.na(AmesTiny[,i]), i] <- mean(AmesTiny[,i], na.rm = TRUE)
}
```

```
## Warning in mean.default(AmesTiny[, i], na.rm = TRUE): argument is not
## numeric or logical: returning NA

## Warning in mean.default(AmesTiny[, i], na.rm = TRUE): argument is not
## numeric or logical: returning NA

## Warning in mean.default(AmesTiny[, i], na.rm = TRUE): argument is not
## numeric or logical: returning NA
```

```
set.seed(12345678)
# divide the data into training and test datasets
testSize <- floor(nrow(AmesTiny)*0.1)
testIndex <- sample(seq_len(nrow(AmesTiny)), size = testSize)
AmesTinyTrain <- AmesTiny[-testIndex, ]
AmesTinyTest <- AmesTiny[testIndex, ]
```

3.1

```
#1.MSE function
mse=function(beta,x,y){
  tmp=t(y-x%*%beta)%*%(y-x%*%beta)/dim(x)[1]
  return(tmp)
}
```

```
#2. R^2 function
r2=function(beta,x,y){
  tmp=(cor(y,x%*%beta))^2
  return(tmp)
}
```

```
#3.

y=I(log(AmesTinyTrain$SalePrice+1))
x1=cbind(rep(1,2637),AmesTinyTrain$Gr.Liv.Area)
lm1=lm(log(SalePrice+1) ~ Gr.Liv.Area,data=AmesTinyTrain)
beta1=lm1$coefficients

dumQ=dummy(AmesTinyTrain$Overall.Qual,drop=F)
dumC=dummy(AmesTinyTrain$Overall.Cond,drop=F)
dim(dumQ)
```

```
## [1] 2637   10
```

```
dim(dumC)
```

```
## [1] 2637    9
```

```
dumQ=dumQ[,-1]
dumC=dumC[,-1]

x2=cbind(x1,AmesTinyTrain$Garage.Area)
lm2=lm(log(SalePrice+1) ~ Gr.Liv.Area+Garage.Area,data=AmesTinyTrain)
beta2=lm2$coefficients

x3=cbind(x2,AmesTinyTrain$Open.Porch.SF)
lm3=lm(log(SalePrice+1) ~ Gr.Liv.Area+Garage.Area+Open.Porch.SF,data=AmesTinyTrain)
beta3=lm3$coefficients

x4=cbind(x3,AmesTinyTrain$Lot.Area)
lm4=lm(log(SalePrice+1) ~ Gr.Liv.Area+Garage.Area+Open.Porch.SF+Lot.Area,data=AmesTinyTrain)
beta4=lm4$coefficients

x5=cbind(x4,dumQ)
lm5=lm(log(SalePrice+1) ~ Gr.Liv.Area+Garage.Area+Open.Porch.SF+Lot.Area+dumQ,data=AmesTinyTrain)
beta5=lm5$coefficients

x6=cbind(x5,dumC)
lm6=lm(log(SalePrice+1) ~ Gr.Liv.Area+Garage.Area+Open.Porch.SF+Lot.Area+dumQ+dumC,data=AmesTinyTrain)
```

```
beta6=lm6$coefficients

x7=cbind(x6,log(AmesTinyTrain$Gr.Liv.Area+1))
lm7=lm(log(SalePrice+1) ~ Gr.Liv.Area+Garage.Area+Open.Porch.SF+Lot.Area+dumQ+dumC+I(log(Gr.Liv.Area+1))
beta7=lm7$coefficients

x8=cbind(x7,log(AmesTinyTrain$Gr.Liv.Area+1)^2)
lm8=lm(log(SalePrice+1) ~ Gr.Liv.Area+Garage.Area+Open.Porch.SF+Lot.Area+dumQ+dumC+I(log(Gr.Liv.Area+1))
beta8=lm8$coefficients

x9=cbind(x8,log(AmesTinyTrain$Gr.Liv.Area+1)^3)
lm9=lm(log(SalePrice+1) ~ Gr.Liv.Area+Garage.Area+Open.Porch.SF+Lot.Area+dumQ+dumC+I(log(Gr.Liv.Area+1))
beta9=lm9$coefficients

x10=cbind(x9,log(AmesTinyTrain$Gr.Liv.Area+1)^4)
lm10=lm(log(SalePrice+1) ~ Gr.Liv.Area+Garage.Area+Open.Porch.SF+Lot.Area+dumQ+dumC+I(log(Gr.Liv.Area+1
beta10=lm10$coefficients

x11=cbind(x10,log(AmesTinyTrain$Gr.Liv.Area+1)^5)
lm11=lm(log(SalePrice+1) ~ Gr.Liv.Area+Garage.Area+Open.Porch.SF+Lot.Area+dumQ+dumC+I(log(Gr.Liv.Area+1
beta11=lm11$coefficients


x=list(x1,x2,x3,x4,x5,x6,x7,x8,x9,x10,x11)
beta=list(beta1,beta2,beta3,beta4,beta5,beta6,beta7,beta8,beta9,beta10,beta11)
modelQuality=matrix(0,nrow=11,ncol=2)
for(i in 1:11){
  modelQuality[i,1]=mse(beta[[i]],x[[i]],y)
  modelQuality[i,2]=r2(beta[[i]],x[[i]],y)
}
as.data.frame(modelQuality) #modelQuality contains all training mses
```

```
##            V1        V2
## 1  0.08748009 0.4745942
## 2  0.06596899 0.6037900
## 3  0.06542675 0.6070466
## 4  0.06531829 0.6076980
## 5  0.03451592 0.7926972
## 6  0.03247054 0.8049818
## 7  0.03087920 0.8145394
## 8  0.03014235 0.8189649
## 9  0.02982983 0.8208419
## 10 0.02915284 0.8249079
## 11 0.02902088 0.8257005
```

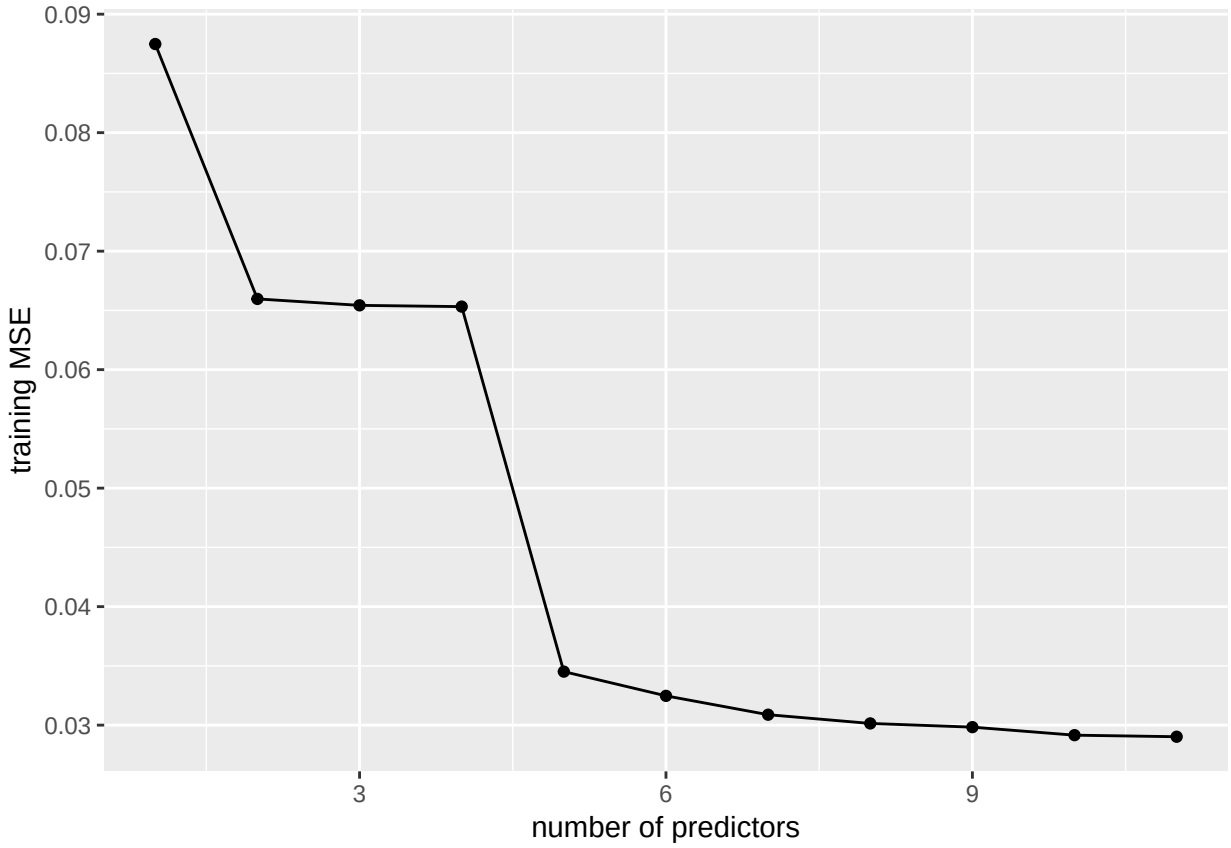
```
modelQuality
```

```
##             [,1]      [,2]
##  [1,] 0.08748009 0.4745942
##  [2,] 0.06596899 0.6037900
##  [3,] 0.06542675 0.6070466
##  [4,] 0.06531829 0.6076980
##  [5,] 0.03451592 0.7926972
##  [6,] 0.03247054 0.8049818
```

```
##  [7,] 0.03087920 0.8145394
##  [8,] 0.03014235 0.8189649
##  [9,] 0.02982983 0.8208419
## [10,] 0.02915284 0.8249079
## [11,] 0.02902088 0.8257005
```

```
#which model has smallest training MSE
#model 11 has the smallest MSE
```

```
n=1:11
ggplot() +
  geom_point(aes(x =  n, y = modelQuality[,1])) +
  geom_line(aes(x = n, y = modelQuality[,1])) +
  xlab("number of predictors") +
  ylab("training MSE")
```




```
#trend shows that as number of predictor increases, training model MSE decreases.
```

```
dumQ=dummy(AmesTinyTest$Overall.Qual,drop=F)
dumC=dummy(AmesTinyTest$Overall.Cond,drop=F)
dim(dumQ)
```

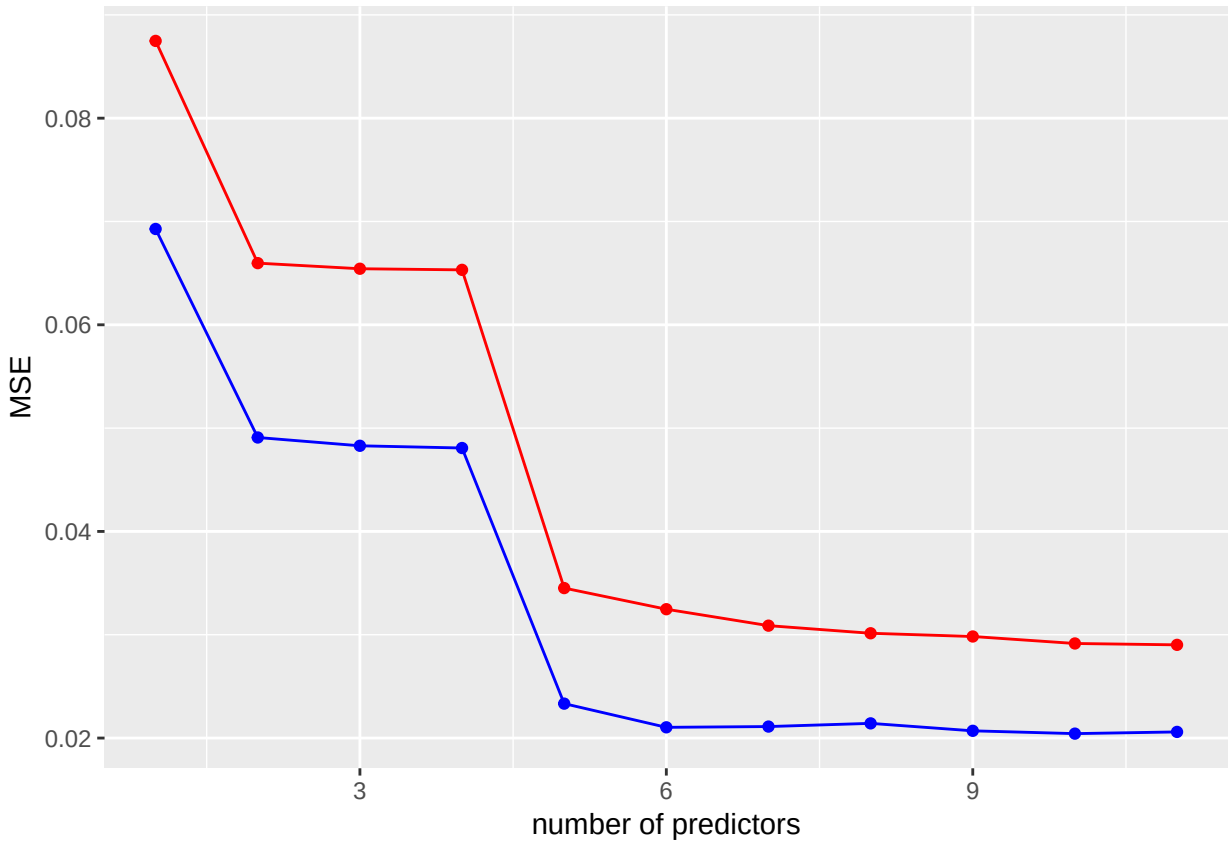
```
## [1] 293  10
```

```
dim(dumC)
```

```
## [1] 293   9
```

```
dumQ=dumQ[,-1]
dumC=dumC[,-1]
```

```r
y_hat1=predict.lm(lm1,AmesTinyTest,type='response')
y_hat2=predict.lm(lm2,AmesTinyTest,type='response')
y_hat3=predict.lm(lm3,AmesTinyTest,type='response')
y_hat4=predict.lm(lm4,AmesTinyTest,type='response')
y_hat5=predict.lm(lm5,AmesTinyTest,type='response')
y_hat6=predict.lm(lm6,AmesTinyTest,type='response')
y_hat7=predict.lm(lm7,AmesTinyTest,type='response')
y_hat8=predict.lm(lm8,AmesTinyTest,type='response')
y_hat9=predict.lm(lm9,AmesTinyTest,type='response')
y_hat10=predict.lm(lm10,AmesTinyTest,type='response')
y_hat11=predict.lm(lm11,AmesTinyTest,type='response')
y=log(AmesTinyTest$SalePrice+1)
mse1=t(y-y_hat1)%*%(y-y_hat1)/293
mse2=t(y-y_hat2)%*%(y-y_hat2)/293
mse3=t(y-y_hat3)%*%(y-y_hat3)/293
mse4=t(y-y_hat4)%*%(y-y_hat4)/293
mse5=t(y-y_hat5)%*%(y-y_hat5)/293
mse6=t(y-y_hat6)%*%(y-y_hat6)/293
mse7=t(y-y_hat7)%*%(y-y_hat7)/293
mse8=t(y-y_hat8)%*%(y-y_hat8)/293
mse9=t(y-y_hat9)%*%(y-y_hat9)/293
mse10=t(y-y_hat10)%*%(y-y_hat10)/293
mse11=t(y-y_hat11)%*%(y-y_hat11)/293
mse_test=c(mse1,mse2,mse3,mse4,mse5,mse6,mse7,mse8,mse9,mse10,mse11)

#plot training mse and test mse in the same plot
ggplot() +
  geom_point(aes(x =  n, y = modelQuality[,1]), color = "red") +
  geom_line(aes(x =  n, y = modelQuality[,1]), color = "red") +
  geom_point(aes(x = n, y = mse_test), color = "blue") +
  geom_line(aes(x = n, y = mse_test), color = "blue") +
  xlab("number of predictors") +
  ylab("MSE")
```




```
#MSE does not always decrease in test data, model 10 has the lowest test MSE.
```

3.2

```
set.seed(123456)
valSize <- floor(nrow(AmesTinyTrain)*0.2)
valIndex <- sample(seq_len(nrow(AmesTinyTrain)), size = valSize)
# actual training data
AmesTinyActTrain <- AmesTinyTrain[-valIndex, ]
AmesTinyActVal <- AmesTinyTrain[valIndex, ]
```

3.2.2

```
#To obtain validation mses
mse_validation=matrix(0,nrow=11,ncol=1)
models=c(lm1,lm2,lm3,lm4,lm5,lm6,lm7,lm8,lm9,lm10,lm11)


#specify actual y value(after log transform)
y=I(log(AmesTinyActVal$SalePrice+1))

#dummify variables
```

```
dumQ=dummy(AmesTinyActVal$Overall.Qual,drop=F)
dumC=dummy(AmesTinyActVal$Overall.Cond,drop=F)
dim(dumQ)
```

```
## [1] 527  10
```

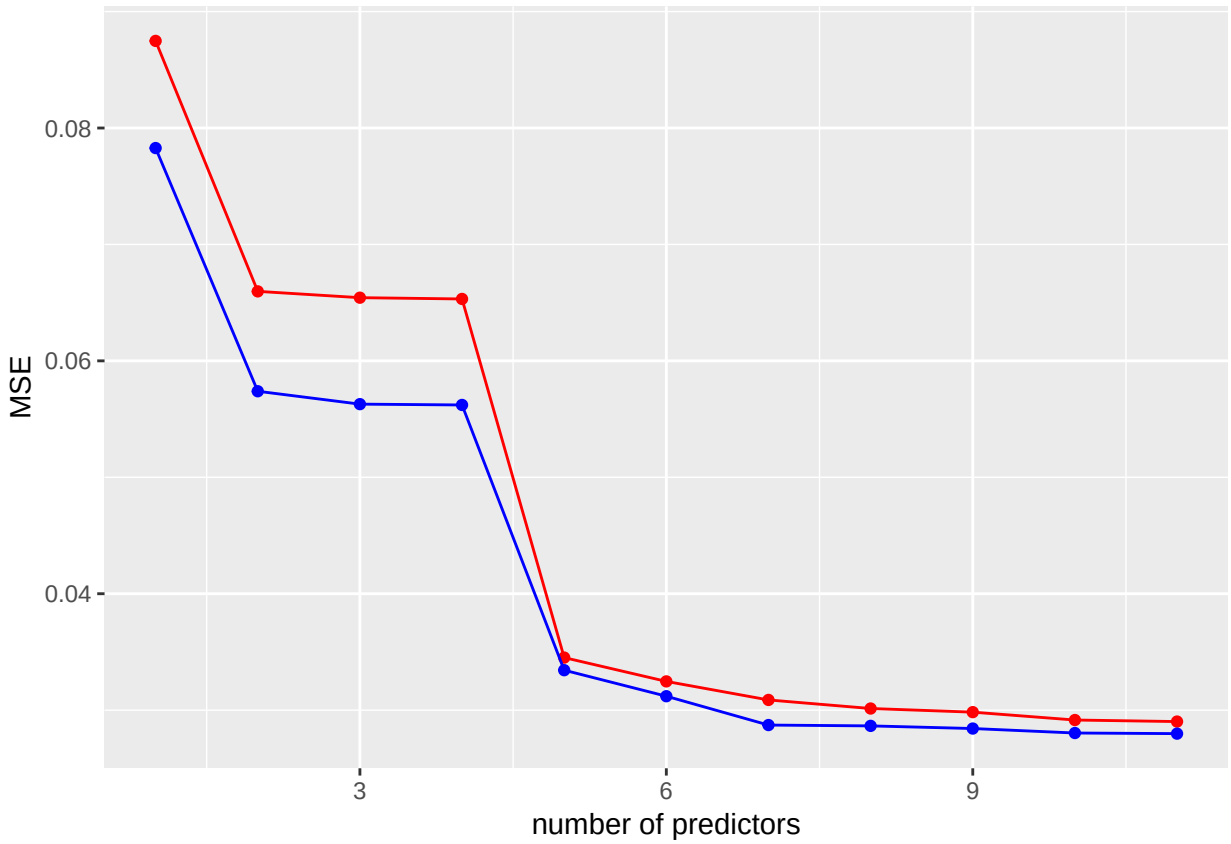
```
dim(dumC)
```

```
## [1] 527   9
```

```
dumQ=dumQ[,-1]
dumC=dumC[,-1]

mse_validation[1]=mean((predict.lm(lm1,AmesTinyActVal)-y)^2)
mse_validation[2]=mean((predict.lm(lm2,AmesTinyActVal)-y)^2)
mse_validation[3]=mean((predict.lm(lm3,AmesTinyActVal)-y)^2)
mse_validation[4]=mean((predict.lm(lm4,AmesTinyActVal)-y)^2)
mse_validation[5]=mean((predict.lm(lm5,AmesTinyActVal)-y)^2)
mse_validation[6]=mean((predict.lm(lm6,AmesTinyActVal)-y)^2)
mse_validation[7]=mean((predict.lm(lm7,AmesTinyActVal)-y)^2)
mse_validation[8]=mean((predict.lm(lm8,AmesTinyActVal)-y)^2)
mse_validation[9]=mean((predict.lm(lm9,AmesTinyActVal)-y)^2)
mse_validation[10]=mean((predict.lm(lm10,AmesTinyActVal)-y)^2)
mse_validation[11]=mean((predict.lm(lm11,AmesTinyActVal)-y)^2)

mse_validation
```

```
##              [,1]
##  [1,] 0.07827796
##  [2,] 0.05739079
##  [3,] 0.05628324
##  [4,] 0.05621300
##  [5,] 0.03343410
##  [6,] 0.03119784
##  [7,] 0.02872096
##  [8,] 0.02864968
##  [9,] 0.02842257
## [10,] 0.02803786
## [11,] 0.02798503
```

```
ggplot() +
  geom_point(aes(x =  n, y = modelQuality[,1]), color = "red") +
  geom_line(aes(x =  n, y = modelQuality[,1]), color = "red") +
  geom_point(aes(x = n, y = mse_validation), color = "blue") +
  geom_line(aes(x = n, y = mse_validation), color = "blue") +
  xlab("number of predictors") +
  ylab("MSE")
```




```
# Model 11 has the lowest validation MSE
```

3.3.1

```
#reload data and re-process data
Ames <- read.delim("AmesHousing.txt", header = TRUE, sep = "\t", dec = ".")
continuousVar <- colnames(Ames)[grep("Frontage|SF|Area|Porch",
colnames(Ames))]

AmesTiny <- Ames[, c(continuousVar,
c("Overall.Qual",
"Overall.Cond","Neighborhood",
"SalePrice"))]
# check NA
colSums(is.na(AmesTiny))
```

```
##    Lot.Frontage        Lot.Area    Mas.Vnr.Area    BsmtFin.SF.1
##             490               0              23               1
##    BsmtFin.SF.2    Bsmt.Unf.SF  Total.Bsmt.SF    X1st.Flr.SF
##               1               1               1               0
##    X2nd.Flr.SF Low.Qual.Fin.SF    Gr.Liv.Area    Garage.Area
##               0               0               0               1
##    Wood.Deck.SF  Open.Porch.SF Enclosed.Porch    X3Ssn.Porch
##               0               0               0               0
##    Screen.Porch       Pool.Area    Overall.Qual    Overall.Cond
```

```
##               0               0               0               0
##    Neighborhood       SalePrice
##               0               0
```

```
AmesTiny$Garage.Area[is.na(AmesTiny$Garage.Area)] = 0
# change factor variable to actual factor in the data frame
AmesTiny$Overall.Qual <- factor(AmesTiny$Overall.Qual)
AmesTiny$Overall.Cond <- factor(AmesTiny$Overall.Cond)
# fill the continuous variable with column mean
for(i in 1:ncol(AmesTiny)){
AmesTiny[is.na(AmesTiny[,i]), i] <- mean(AmesTiny[,i], na.rm = TRUE)
}
```

```
## Warning in mean.default(AmesTiny[, i], na.rm = TRUE): argument is not
## numeric or logical: returning NA

## Warning in mean.default(AmesTiny[, i], na.rm = TRUE): argument is not
## numeric or logical: returning NA

## Warning in mean.default(AmesTiny[, i], na.rm = TRUE): argument is not
## numeric or logical: returning NA
```

```
set.seed(12345678)
# divide the data into training and test datasets
testSize <- floor(nrow(AmesTiny)*0.1)
testIndex <- sample(seq_len(nrow(AmesTiny)), size = testSize)
AmesTinyTrain <- AmesTiny[-testIndex, ]
AmesTinyTest <- AmesTiny[testIndex, ]
```

```
set.seed(12345678)
folds <- createFolds(log(AmesTinyTrain$SalePrice+1), k = 5)

#create dummy variables for each folds
dumQ_1=dummy(AmesTinyTrain[folds[[1]],]$Overall.Qual,drop=F)
dumC_1=dummy(AmesTinyTrain[folds[[1]],]$Overall.Cond,drop=F)

dumQ_2=dummy(AmesTinyTrain[folds[[2]],]$Overall.Qual,drop=F)
dumC_2=dummy(AmesTinyTrain[folds[[2]],]$Overall.Cond,drop=F)

dumQ_3=dummy(AmesTinyTrain[folds[[3]],]$Overall.Qual,drop=F)
dumC_3=dummy(AmesTinyTrain[folds[[3]],]$Overall.Cond,drop=F)

dumQ_4=dummy(AmesTinyTrain[folds[[4]],]$Overall.Qual,drop=F)
dumC_4=dummy(AmesTinyTrain[folds[[4]],]$Overall.Cond,drop=F)

dumQ_5=dummy(AmesTinyTrain[folds[[5]],]$Overall.Qual,drop=F)
dumC_5=dummy(AmesTinyTrain[folds[[5]],]$Overall.Cond,drop=F)


#specify actual y value(after log transform)
y=I(log(AmesTinyTrain$SalePrice+1))

#create empty matrix to hold mse-cv
mse_cv=matrix(0,nrow=11,ncol=5)
```

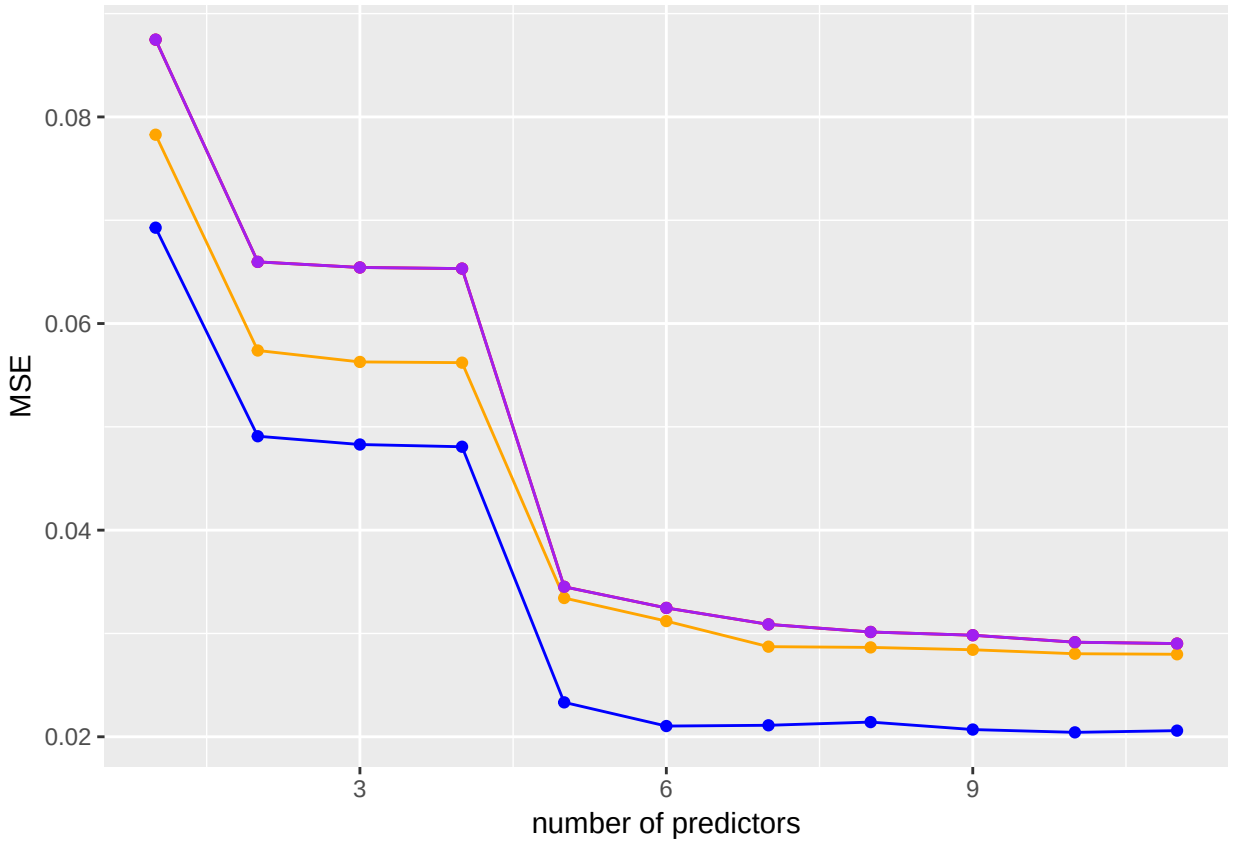
```
#select dumQ and dumC for each fold and adjust size, perform mse calculations
for(i in 1:5){
  if(i==1){
    dumQ=dumQ_1
    dumC=dumC_1
      dumQ=dumQ[,-1]
dumC=dumC[,-1]
  }
  if(i==2){
    dumQ=dumQ_2
    dumC=dumC_2
      dumQ=dumQ[,-1]
dumC=dumC[,-1]
  }
  if(i==3){
    dumQ=dumQ_3
    dumC=dumC_3
      dumQ=dumQ[,-1]
dumC=dumC[,-1]
  }
  if(i==4){
    dumQ=dumQ_4
    dumC=dumC_4
      dumQ=dumQ[,-1]
dumC=dumC[,-1]
  }
  if(i==5){
    dumQ=dumQ_5
    dumC=dumC_5
      dumQ=dumQ[,-1]
dumC=dumC[,-1]
  }
mse_cv[1,i]=mean((predict.lm(lm1,AmesTinyTrain[folds[[i]],])-y[folds[[i]]])^2)
mse_cv[2,i]=mean((predict.lm(lm2,AmesTinyTrain[folds[[i]],])-y[folds[[i]]])^2)
mse_cv[3,i]=mean((predict.lm(lm3,AmesTinyTrain[folds[[i]],])-y[folds[[i]]])^2)
mse_cv[4,i]=mean((predict.lm(lm4,AmesTinyTrain[folds[[i]],])-y[folds[[i]]])^2)
mse_cv[5,i]=mean((predict.lm(lm5,AmesTinyTrain[folds[[i]],])-y[folds[[i]]])^2)
mse_cv[6,i]=mean((predict.lm(lm6,AmesTinyTrain[folds[[i]],])-y[folds[[i]]])^2)
mse_cv[7,i]=mean((predict.lm(lm7,AmesTinyTrain[folds[[i]],])-y[folds[[i]]])^2)
mse_cv[8,i]=mean((predict.lm(lm8,AmesTinyTrain[folds[[i]],])-y[folds[[i]]])^2)
mse_cv[9,i]=mean((predict.lm(lm9,AmesTinyTrain[folds[[i]],])-y[folds[[i]]])^2)
mse_cv[10,i]=mean((predict.lm(lm10,AmesTinyTrain[folds[[i]],])-y[folds[[i]]])^2)
mse_cv[11,i]=mean((predict.lm(lm11,AmesTinyTrain[folds[[i]],])-y[folds[[i]]])^2)
}

mse_cv
```

```
##              [,1]       [,2]       [,3]       [,4]       [,5]
##  [1,] 0.09494526 0.08058022 0.09781204 0.07378344 0.09028723
##  [2,] 0.07449943 0.06566400 0.07197599 0.05355306 0.06415647
##  [3,] 0.07333463 0.06534016 0.07085147 0.05312622 0.06448325
##  [4,] 0.07335489 0.06485402 0.07116316 0.05277365 0.06444827
##  [5,] 0.03495869 0.03469815 0.04162908 0.02626318 0.03502917
##  [6,] 0.03379326 0.03079172 0.03801485 0.02667094 0.03308395
```

```
##  [7,] 0.03247273 0.03097612 0.03415206 0.02731315 0.02948440
##  [8,] 0.03132264 0.03097218 0.03197828 0.02796449 0.02847572
##  [9,] 0.03125205 0.03088720 0.03100707 0.02772437 0.02827939
## [10,] 0.02978318 0.03064180 0.02987451 0.02762585 0.02783852
## [11,] 0.03014250 0.03033954 0.02970411 0.02729315 0.02762524
```

3.3.2

```
#calcaulate average cv-mses over folds
mse_cv_mean=rowMeans(mse_cv,dims=1)

ggplot() +
  geom_point(aes(x =  n, y = modelQuality[,1]), color = "red") +
  geom_line(aes(x =  n, y = modelQuality[,1]), color = "red") +
  geom_point(aes(x = n, y = mse_test), color = "blue") +
  geom_line(aes(x = n, y = mse_test), color = "blue") +
  geom_point(aes(x =  n, y = mse_validation), color = "orange") +
  geom_line(aes(x =  n, y = mse_validation), color = "orange") +
  geom_point(aes(x =  n, y = mse_cv_mean), color = "purple") +
  geom_line(aes(x =  n, y = mse_cv_mean), color = "purple") +
  xlab("number of predictors") +
  ylab("MSE")
```




```
#the 11th model has the smallest average cv_mse
```

3.4.1

```
Data = dummy.data.frame(data = AmesTiny, names = "Neighborhood", drop = FALSE)

ridge_model <- glmnet(x = as.matrix(Data[,-49]), y = log(Data$SalePrice + 1), alpha = 0, lambda = 1, sta
```

3.4.2

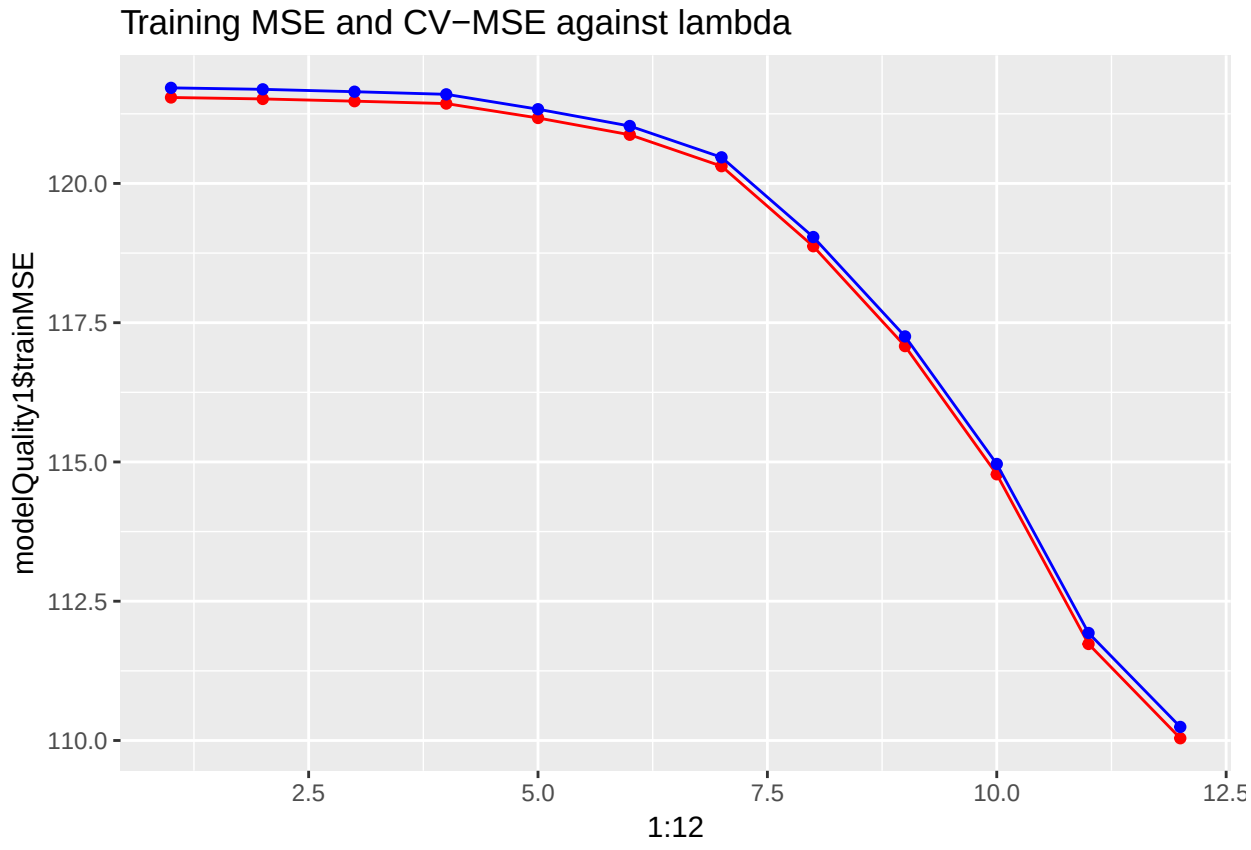
```
ridge_models <- glmnet(x = as.matrix(Data[, -49]), y = log(Data$SalePrice + 1), alpha = 0, lambda = c(0

modelQuality1 = data.frame("trainMSE" = rep(0,12), "CVMSE" = rep(0,12))
response = log(Data$SalePrice + 1)


#training mse
 m=as.matrix(Data[,-49])
 m=mapply(m,FUN=as.numeric)
 m=matrix(data=m,ncol=48,nrow=2930)

for (i in 1:12) {
  error = response - m %*% ridge_models$beta[,i]
  modelQuality1[i,"trainMSE"] = mean(error^2)
}



#cv-mses
set.seed(12345678)
folds <- createFolds(AmesTiny$SalePrice, k = 5)

modelQuality2 <- data.frame("fold1" = rep(0,12), "fold2" = rep(0,12), "fold3" = rep(0,12), "fold4" = rep

for (i in 1:5) {
  DataTrain = Data[-folds[i][[1]],]
  DataValid = Data[folds[i][[1]],]
  response = log(DataValid$SalePrice + 1)

  model <- glmnet(x = as.matrix(DataTrain[,-49]), y = log(DataTrain$SalePrice + 1), alpha = 0, lambda =

   m2=as.matrix(DataValid[,-49])
 m2=mapply(m,FUN=as.numeric)
 m2=matrix(data=m,ncol=48,nrow=2930)
  for (j in 1:12) {

    error = response - m2 %*% model$beta[,j]
    modelQuality2[j,i] = mean(error^2)
  }

}
```

```
## Warning in response - m2 %*% model$beta[, j]: longer object length is not a
```

```
## multiple of shorter object length

## Warning in response - m2 %*% model$beta[, j]: longer object length is not a
## multiple of shorter object length

## Warning in response - m2 %*% model$beta[, j]: longer object length is not a
## multiple of shorter object length

## Warning in response - m2 %*% model$beta[, j]: longer object length is not a
## multiple of shorter object length

## Warning in response - m2 %*% model$beta[, j]: longer object length is not a
## multiple of shorter object length

## Warning in response - m2 %*% model$beta[, j]: longer object length is not a
## multiple of shorter object length

## Warning in response - m2 %*% model$beta[, j]: longer object length is not a
## multiple of shorter object length

## Warning in response - m2 %*% model$beta[, j]: longer object length is not a
## multiple of shorter object length

## Warning in response - m2 %*% model$beta[, j]: longer object length is not a
## multiple of shorter object length

## Warning in response - m2 %*% model$beta[, j]: longer object length is not a
## multiple of shorter object length

## Warning in response - m2 %*% model$beta[, j]: longer object length is not a
## multiple of shorter object length

## Warning in response - m2 %*% model$beta[, j]: longer object length is not a
## multiple of shorter object length

## Warning in response - m2 %*% model$beta[, j]: longer object length is not a
## multiple of shorter object length

## Warning in response - m2 %*% model$beta[, j]: longer object length is not a
## multiple of shorter object length

## Warning in response - m2 %*% model$beta[, j]: longer object length is not a
## multiple of shorter object length

## Warning in response - m2 %*% model$beta[, j]: longer object length is not a
## multiple of shorter object length

## Warning in response - m2 %*% model$beta[, j]: longer object length is not a
## multiple of shorter object length

## Warning in response - m2 %*% model$beta[, j]: longer object length is not a
## multiple of shorter object length

## Warning in response - m2 %*% model$beta[, j]: longer object length is not a
```

```
## multiple of shorter object length

## Warning in response - m2 %*% model$beta[, j]: longer object length is not a
## multiple of shorter object length

## Warning in response - m2 %*% model$beta[, j]: longer object length is not a
## multiple of shorter object length

## Warning in response - m2 %*% model$beta[, j]: longer object length is not a
## multiple of shorter object length

## Warning in response - m2 %*% model$beta[, j]: longer object length is not a
## multiple of shorter object length

## Warning in response - m2 %*% model$beta[, j]: longer object length is not a
## multiple of shorter object length
```

```r
for (i in 1:12) {
  modelQuality1[i,"CVMSE"] = sum(modelQuality2[i,1:5])/5
}

ggplot() +
  geom_point(aes(x = 1:12, y = modelQuality1$trainMSE), color = "red") +
  geom_line(aes(x = 1:12, y = modelQuality1$trainMSE), color = "red") +
  geom_point(aes(x = 1:12, y = modelQuality1$CVMSE), color = "blue") +
  geom_line(aes(x = 1:12, y = modelQuality1$CVMSE), color = "blue") +
  ggtitle("Training MSE and CV-MSE against lambda")
```



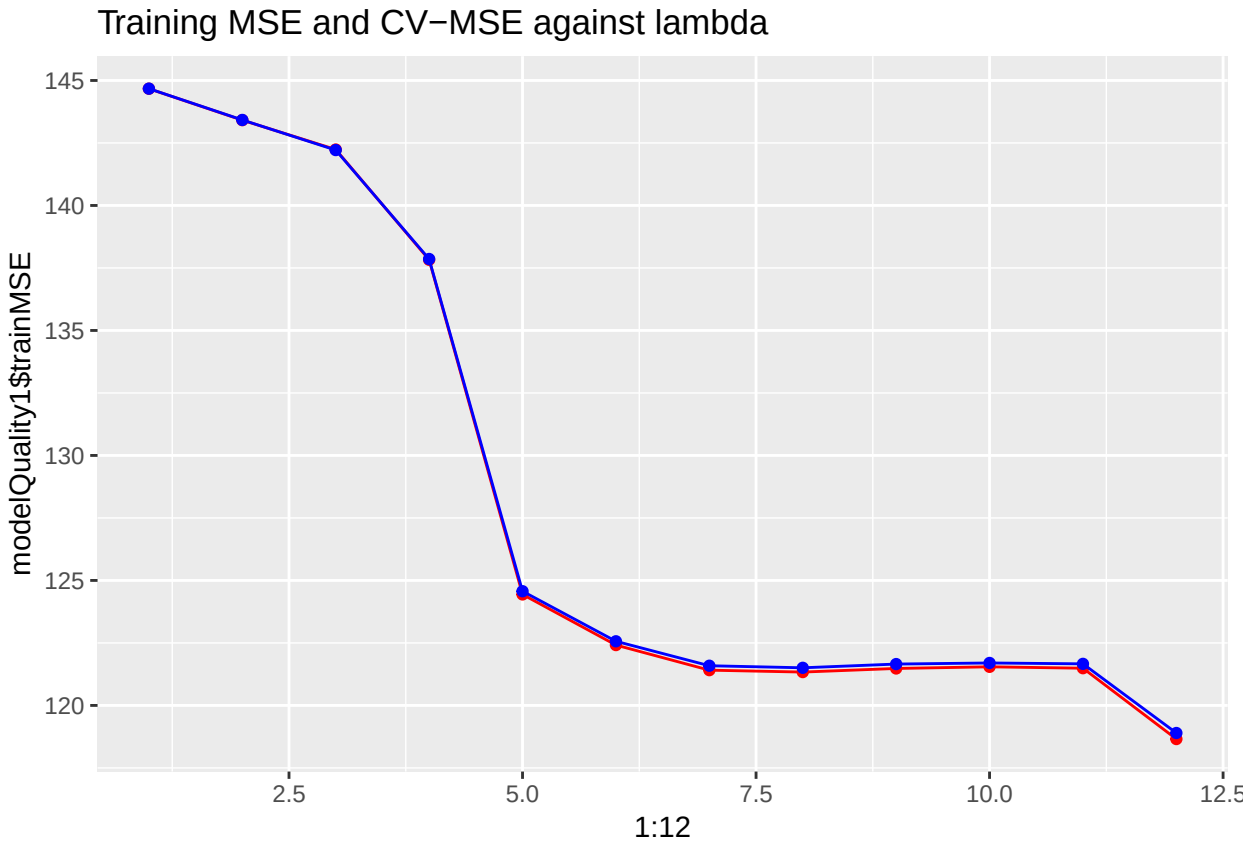
3.5

```
lasso_model <- glmnet(x = as.matrix(Data[, -49]), y = log(Data$SalePrice + 1), alpha = 1, lambda = c(0.


modelQuality1 = data.frame("trainMSE" = rep(0,12), "CVMSE" = rep(0,12))
response = log(Data$SalePrice + 1)


#training mse
for (i in 1:12) {
  error = response - m %*% lasso_model$beta[,i]
  modelQuality1[i,"trainMSE"] = mean(error^2)
}

#cv-mses
set.seed(12345678)
folds <- createFolds(AmesTiny$SalePrice, k = 5)

modelQuality2 <- data.frame("fold1" = rep(0,12), "fold2" = rep(0,12), "fold3" = rep(0,12), "fold4" = rep

for (i in 1:5) {
  DataTrain = Data[-folds[i][[1]],]
  DataValid = Data[folds[i][[1]],]
  response = log(DataValid$SalePrice + 1)

  model <- glmnet(x = as.matrix(DataTrain[,-49]), y = log(DataTrain$SalePrice + 1), alpha = 1, lambda =

  for (j in 1:12) {
    error = response - m2 %*% model$beta[,j]
    modelQuality2[j,i] = mean(error^2)
  }

}
```

```
## Warning in response - m2 %*% model$beta[, j]: longer object length is not a
## multiple of shorter object length

## Warning in response - m2 %*% model$beta[, j]: longer object length is not a
## multiple of shorter object length

## Warning in response - m2 %*% model$beta[, j]: longer object length is not a
## multiple of shorter object length

## Warning in response - m2 %*% model$beta[, j]: longer object length is not a
## multiple of shorter object length

## Warning in response - m2 %*% model$beta[, j]: longer object length is not a
## multiple of shorter object length

## Warning in response - m2 %*% model$beta[, j]: longer object length is not a
## multiple of shorter object length
```

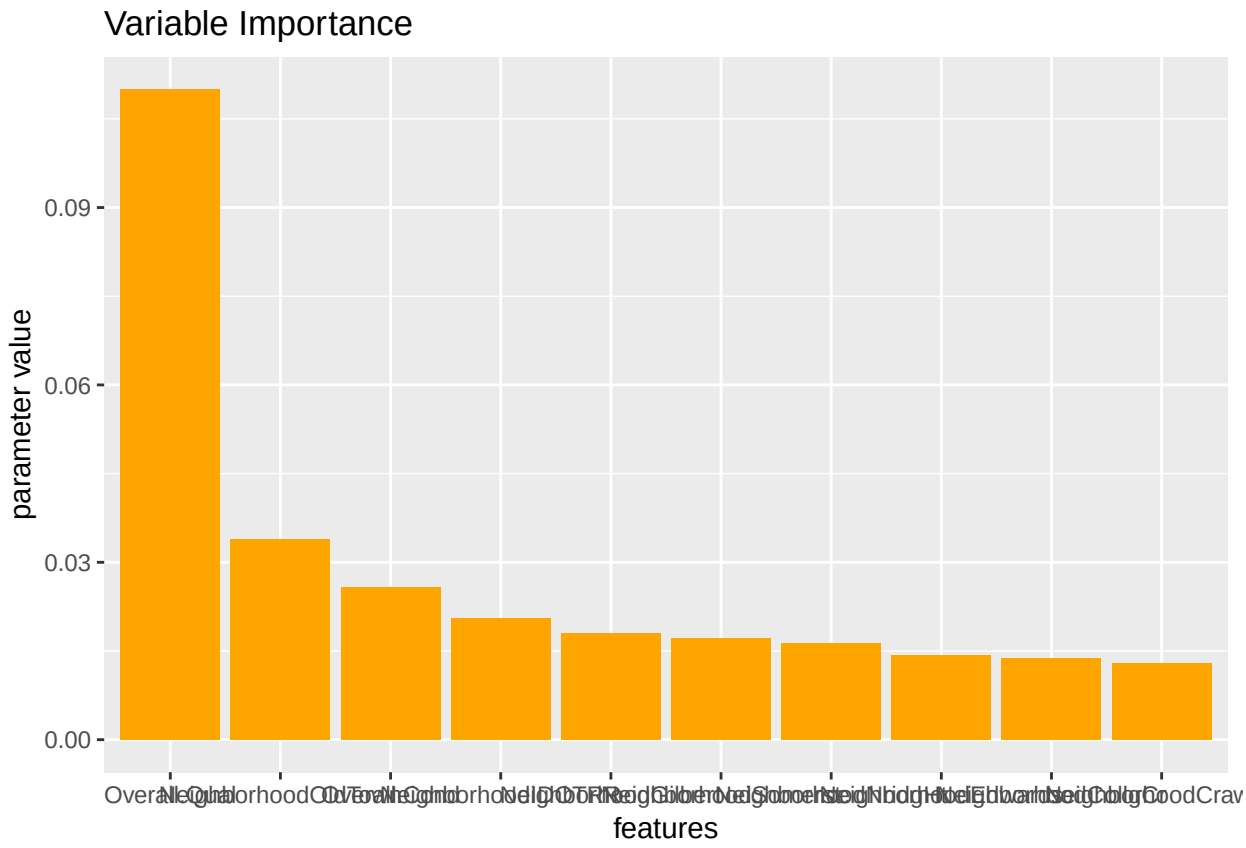
```
## Warning in response - m2 %*% model$beta[, j]: longer object length is not a
## multiple of shorter object length

## Warning in response - m2 %*% model$beta[, j]: longer object length is not a
## multiple of shorter object length

## Warning in response - m2 %*% model$beta[, j]: longer object length is not a
## multiple of shorter object length

## Warning in response - m2 %*% model$beta[, j]: longer object length is not a
## multiple of shorter object length

## Warning in response - m2 %*% model$beta[, j]: longer object length is not a
## multiple of shorter object length

## Warning in response - m2 %*% model$beta[, j]: longer object length is not a
## multiple of shorter object length

## Warning in response - m2 %*% model$beta[, j]: longer object length is not a
## multiple of shorter object length

## Warning in response - m2 %*% model$beta[, j]: longer object length is not a
## multiple of shorter object length

## Warning in response - m2 %*% model$beta[, j]: longer object length is not a
## multiple of shorter object length

## Warning in response - m2 %*% model$beta[, j]: longer object length is not a
## multiple of shorter object length

## Warning in response - m2 %*% model$beta[, j]: longer object length is not a
## multiple of shorter object length

## Warning in response - m2 %*% model$beta[, j]: longer object length is not a
## multiple of shorter object length

## Warning in response - m2 %*% model$beta[, j]: longer object length is not a
## multiple of shorter object length

## Warning in response - m2 %*% model$beta[, j]: longer object length is not a
## multiple of shorter object length

## Warning in response - m2 %*% model$beta[, j]: longer object length is not a
## multiple of shorter object length

## Warning in response - m2 %*% model$beta[, j]: longer object length is not a
## multiple of shorter object length

## Warning in response - m2 %*% model$beta[, j]: longer object length is not a
## multiple of shorter object length

## Warning in response - m2 %*% model$beta[, j]: longer object length is not a
## multiple of shorter object length
```

```r
for (i in 1:12) {
  modelQuality1[i,"CVMSE"] = sum(modelQuality2[i,1:5])/5
}


ggplot() +
  geom_point(aes(x = 1:12, y = modelQuality1$trainMSE), color = "red") +
  geom_line(aes(x = 1:12, y = modelQuality1$trainMSE), color = "red") +
  geom_point(aes(x = 1:12, y = modelQuality1$CVMSE), color = "blue") +
  geom_line(aes(x = 1:12, y = modelQuality1$CVMSE), color = "blue") +
  ggtitle("Training MSE and CV-MSE against lambda")
```



3.6.1

```r
best = 12

beta <- head(sort(abs(ridge_models$beta[,best]), decreasing = TRUE), n = 10)

ggplot(data.frame(beta, names(beta)), aes(x = reorder(names.beta., -beta), y = beta)) +
  geom_bar(stat="identity", fill = "Orange") +
  xlab("features") +
  ylab("parameter value") +
  ggtitle("Variable Importance")
```

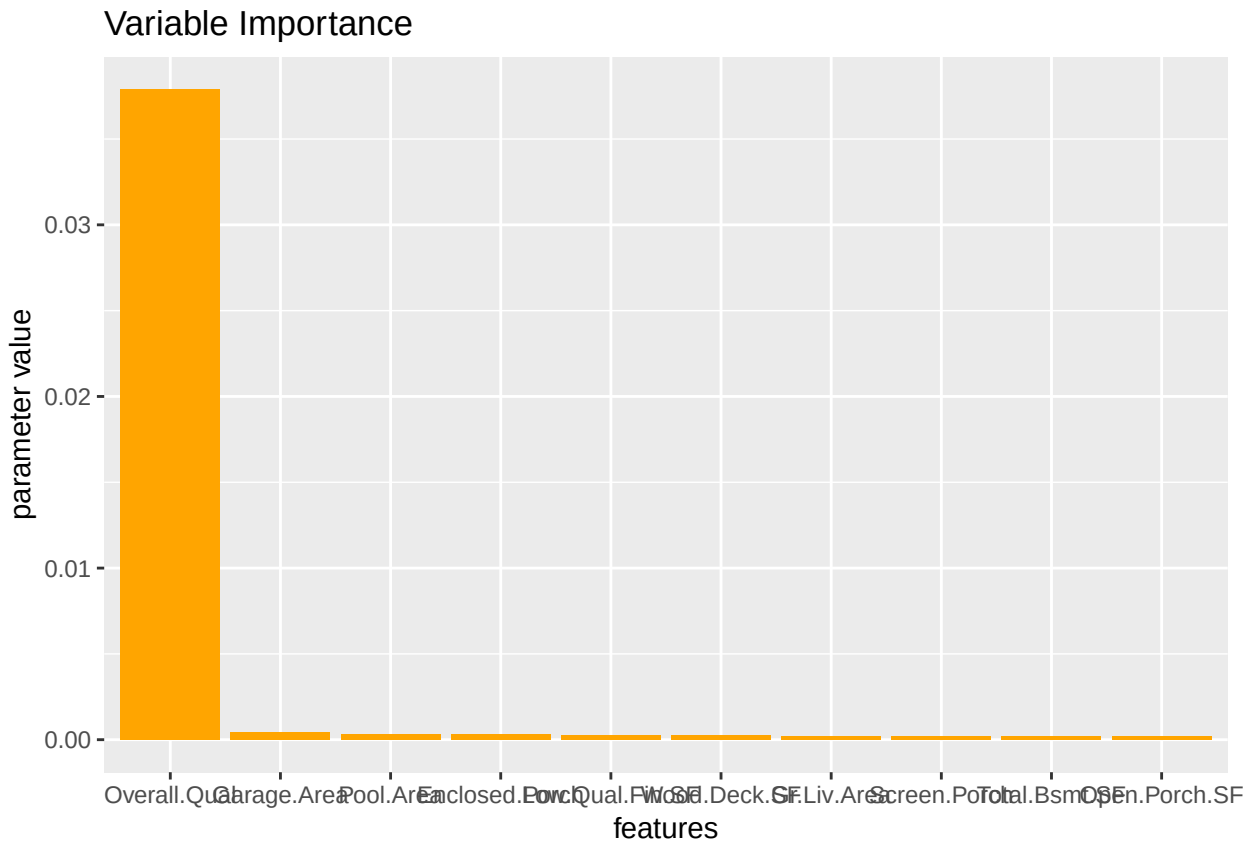


3.6.2

```
best = 12

beta<- head(sort(abs(lasso_model$beta[,best]), decreasing = TRUE), n = 10)

data.frame(beta, names(beta))
```

```
##                         beta     names.beta.
## Overall.Qual    0.0378959568    Overall.Qual
## Garage.Area     0.0004508860     Garage.Area
## Pool.Area       0.0003380491       Pool.Area
## Enclosed.Porch  0.0003198712  Enclosed.Porch
## Low.Qual.Fin.SF 0.0002696750 Low.Qual.Fin.SF
## Wood.Deck.SF    0.0002501075    Wood.Deck.SF
## Gr.Liv.Area     0.0002321549     Gr.Liv.Area
## Screen.Porch    0.0002206889    Screen.Porch
## Total.Bsmt.SF   0.0001942016   Total.Bsmt.SF
## Open.Porch.SF   0.0001772191   Open.Porch.SF
```

```
ggplot(data.frame(beta, names(beta)), aes(x = reorder(names.beta., -beta), y = beta)) +
  geom_bar(stat="identity", fill = "Orange") +
  xlab("features") +
  ylab("parameter value") +
  ggtitle("Variable Importance")
```


Variable Importance
parameter value
0.03
0.02
0.01
0.00
Overall.Qual
Garage.Area
Pool.Area
Enclosed.Porch
Low.Qual.Fin.SF
Wood.Deck.SF
Liv.Area
Screen.Porch
Total.Bsmt.SF
Open.Porch.SF
features