The honor code

(a) Please state the names of people who you worked with for this homework. You can also provide your comments about the homework here.

> Cinidy Liu

(b) Please type/write the following sentences yourself and sign at the end. We want to make it *extra* clear that nobody cheats even unintentionally.

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

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

1.

(a) F LASSO is a biased estimator, so it cannot prevent bias

(b) F $\hat{\theta}_{Lasso}$ could have negative coordinates.

(c) T For instance, when X is not full rank, it's possible to have many Lasso solutions.

(d) F it is recommended because different features may have different magnitudes. Otherwise, the regularization is unfair

(e) F when $n>d$ we invert $\underset{d \times d}{(X^TX+\lambda I_d)^{-1}}$, which is the original ridge regression

(f) T Gram matrix is a PSD

(g) T $K(x,z)=(x^Tz+1)^p$, p can be any large as it does not affect computational complexity

(h) F $K(x,z)=(x^Tz+1)^1$

(i) T in documentation, it says it often faster to fit a whole path than compute a single fit

1. $\min_{\theta \in R^d} \frac{1}{n}\|X\theta - y\|_2^2 + w\|\theta\|_1$

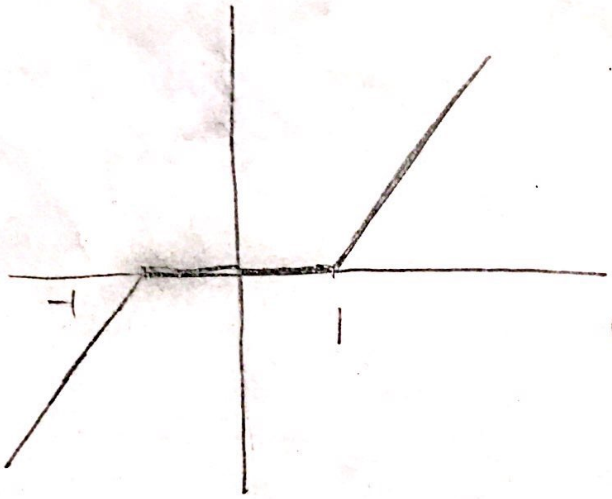
$= \min_{\theta \in R^d} \frac{1}{n}(\|X\theta - y\|_2^2 + nw\|\theta\|_1)$

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

$= \min_{\theta \in R^d} \frac{1}{n}(\|X\theta - y\|_2^2 + \lambda\|\theta\|_1)$

$= \min_{\theta \in R^d} \|X\theta - y\|_2^2 + \lambda\|\theta\|_1$

2.



non-decreasing

3.

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

is the loss function when fixing all θ except θ_j

let this function be g and $\alpha = \theta_j$ we get

$$g(\alpha) = \sum_{i=1}^{n}\Big(a x_{ij} + \sum_{k=1; k\neq j}^{d} \theta_k x_{ik} - y_i\Big)^2 + \lambda|\alpha| + \lambda \sum_{k=1; k\neq j}^{d} |\theta_k|$$

4. $$\frac{\partial g}{\partial \alpha} = 2\sum_{i=1}^{n} x_{ij}\Big(a x_{ij} + \sum_{k=1; k\neq j}^{d} \theta_k x_{ik} - y_i\Big) + \lambda \quad \text{if } a > 0$$

$$\frac{\partial g}{\partial \alpha} = 2\sum_{i=1}^{n} x_{ij}\Big(a x_{ij} + \sum_{k=1; k\neq j}^{d} \theta_k x_{ik} - y_i\Big) - \lambda \quad \text{if } a < 0$$

5. if $a^* > 0$

$$\frac{\partial g}{\partial a} = 2\sum_{i=1}^{n} a x_{ij}^2 + 2\sum_{i=1}^{n} x_{ij}\underbrace{\left(\sum_{k=1, k\neq j}^{d} \theta_k x_{ik} - y_i\right)}_{= -c_j} + \lambda = 0$$

$$a^* \cdot a_j - c_j + \lambda = 0$$

$$a^* = \frac{1}{a_j}(c_j - \lambda)$$

6. if $a^* < 0$

$$\frac{\partial g}{\partial a} = 2\sum_{i=1}^{n} a x_{ij}^2 + 2\sum_{i=1}^{n} x_{ij}\underbrace{\left(\sum_{k=1, k\neq j}^{d} \theta_k x_{ik} - y_i\right)}_{= -c_j} - \lambda = 0$$

$$a^* a_j - c_j - \lambda = 0$$

$$a^* = \frac{1}{a_j}(c_j + \lambda)$$

if $a^* = 0$

$$-\lambda\alpha - c_j = 0 \qquad \beta \in [-1, 1] \text{ slope}$$

$$\beta = \frac{-c_j}{\lambda} \in [-1, 1] \Rightarrow c_j \in [-\lambda, \lambda]$$

7. Based on (5) (6)

for a^* is positive, we need $c_j - \lambda > 0$

$$c_j > \lambda$$

for a^* is negative, we need $c_j + \lambda < 0$

$$c_j < -\lambda$$

8. $D^{+}(g)(\alpha) = \lim\limits_{\varepsilon \to 0, \varepsilon > 0} = \dfrac{\sum_{i=1}^{n} (A + \varepsilon X_{ij})^2 + \lambda|\alpha+\varepsilon| - \lambda|\alpha| - \sum_{i=1}^{n} (A)^2}{\varepsilon}$

let $A = (\alpha X_{ij} + \sum_{k=1, k\neq j}^{d} \theta_k X_{ik} - y_i)^2$

$$= \lim_{\varepsilon \to 0, \varepsilon > 0} \frac{2\sum_{i=1}^{n} \varepsilon X_{ij}(A) + \sum_{i=1}^{n} \varepsilon^2 X_{ij}^2 + \lambda\alpha + \lambda\varepsilon - \lambda\alpha}{\varepsilon}$$

$$= \lim_{\varepsilon \to 0, \varepsilon > 0} 2\sum_{i=1}^{n} X_{ij} A + \varepsilon \sum_{i=1}^{n} X_{ij}^2 + \lambda$$

$$= -C_j + \lambda$$

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

$$= \lim_{\varepsilon>0, \varepsilon\to 0} \frac{\sum_{i=1}^{n}(A-\varepsilon X_{ij})^2 + \lambda|\alpha-\varepsilon| - \lambda|\alpha| - \sum_{i=1}^{n} A^2}{\varepsilon}$$

$$= \lim_{\varepsilon>0, \varepsilon\to 0} \frac{\sum_{i=1}^{n} -2A\varepsilon X_{ij} + \sum_{i=1}^{n} \varepsilon^2 X_{ij}^2 + \varepsilon\lambda}{\varepsilon}$$

$$= \lim_{\varepsilon>0, \varepsilon\to 0} -2\sum_{i=1}^{n} X_{ij} A + \varepsilon \sum_{i=1}^{n} X_{ij}^2 + \lambda$$

$$= -2\sum_{i=1}^{n} X_{ij} \left(\alpha X_{ij} + \sum_{k=1, k\neq j}^{d} \theta_k X_{ik} - y_i\right) + \lambda$$

$$= C_j + \lambda$$

9. $D^+(g)(\alpha^*) = \lambda - C_j \geqslant 0$ since $\lambda > 0$ and $C_j \in [-\lambda, \lambda]$

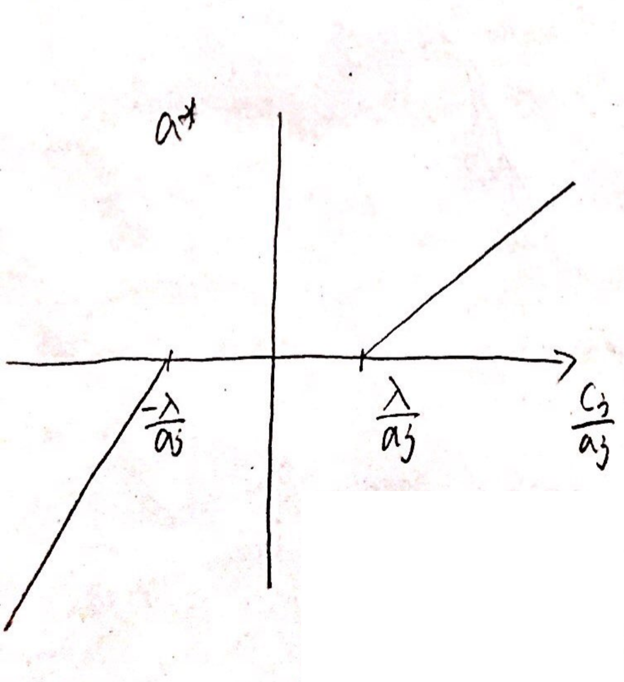
$D^-(g)(\alpha^*) = C_j + \lambda \geqslant 0$ since $C_j \in [-\lambda, \lambda]$

by result (4), $\alpha^* = 0$ satisfies the minimizer of g.

10. $\theta_j = \text{soft}\left(\frac{c_j}{a_j}, \frac{\lambda}{a_j}\right)$ in Algorithm 1

$a_j = 2\sum_{i=1}^{n} X_{ij}^2 \geq 0$

if not all $X_{ij} = 0$, we can assume $a_j > 0$



Based on Q5 and Q6 and Q8

$$a^* = \begin{cases} \frac{c_j}{a_j} - \frac{\lambda}{a_j} & c_j > \lambda \\ 0 & c_j \in [-\lambda, \lambda] \\ \frac{c_j}{a_j} + \frac{\lambda}{a_j} & c_j < -\lambda \end{cases}$$

then a^* can be written as

$$a^* = \begin{cases} \frac{c_j}{a_j} - \frac{\lambda}{a_j} & \text{if } \frac{c_j}{a_j} > \frac{\lambda}{a_j} \\ 0 & \text{if } \frac{c_j}{a_j} \in \left[-\frac{\lambda}{a_j}, \frac{\lambda}{a_j}\right] \\ \frac{c_j}{a_j} + \frac{\lambda}{a_j} & \text{if } \frac{c_j}{a_j} < -\frac{\lambda}{a_j} \end{cases}$$

which is just the definition of $\text{soft}\left(\frac{c_j}{a_j}, \frac{\lambda}{a_j}\right)$

HW5

caojilin

4/3/2019

#2.11 In the coordinate descent algorithm, we update θ_j while fixing all other coordinates. We showed that this reduces the problem to 1-dim lasso. And we can solve 1-dim lasso problem easily. After we updated θ_j, we update next coordinate until all coordinates converge.
We can get sparse solutions because when we are solving 1-dim lasso, the soft thresholding function makes the optimal solution 0 when $c_j \in [-\lambda, \lambda]$ according to question (10). On the other hand, ridge regression just makes parameters smaller not makes them to 0.

#3.1-3.2

```r
MSE = function(y, x, beta){
  sum((y-(x %*% beta))^2)/length(y)
}
R2 = function(y, x, beta){
  cor(y, x %*% beta)^2
}
```

#3.3

```
test.mse= rep(0, 11)
modelQuality = matrix(rep(0,22),nrow=11)
modelQualityRImp =matrix(rep(0,22),nrow=11)
for (i in 1:length(Xnames)) {
  formula = as.formula(paste("log(SalePrice + 1) ~ ", paste(Xnames[1:i], collapse= "+")))
  lmod = lm(formula, data = AmesTinyTrain)
  training.label = log(AmesTinyTrain$SalePrice+1)
  mse = MSE(training.label, model.matrix(lmod),lmod$coefficients)
  r2 = R2(training.label,model.matrix(lmod),lmod$coefficients)
  modelQuality[i,1] = mse
  modelQuality[i,2] = r2
  r.out = summary(lmod)
  modelQualityRImp[i,1] = sum(r.out$residuals^2)/nrow(model.matrix(lmod))
  modelQualityRImp[i,2] = r.out$r.squared

  test.label = log(AmesTinyTest$SalePrice+1)
  test.mse[i] = sum((test.label-predict(lmod, AmesTinyTest))^2)/nrow(AmesTinyTest)
}

modelQuality = as.data.frame(modelQuality)
modelQualityRImp = as.data.frame(modelQualityRImp)
colnames(modelQuality) <- c("MSE", "R2")
colnames(modelQualityRImp) <- c("MSE", "R2")
modelQuality
```

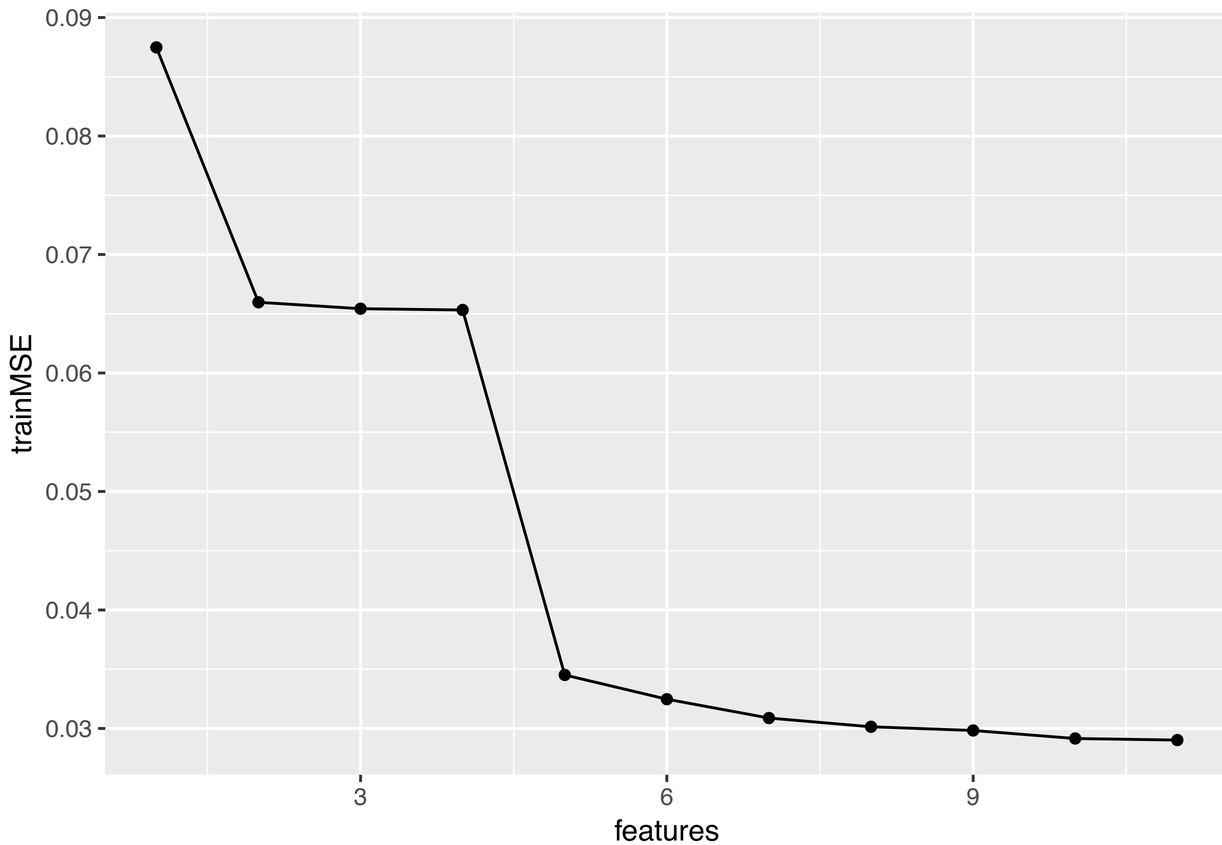
```
##             MSE        R2
## 1  0.08748009 0.4745942
## 2  0.06596899 0.6037900
## 3  0.06542675 0.6070466
## 4  0.06531829 0.6076980
## 5  0.03451592 0.7926972
## 6  0.03247054 0.8049818
## 7  0.03087920 0.8145394
## 8  0.03014235 0.8189649
## 9  0.02982983 0.8208419
## 10 0.02915284 0.8249079
## 11 0.02902088 0.8257005
```

```
modelQualityRImp
```

```
##             MSE        R2
## 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
```

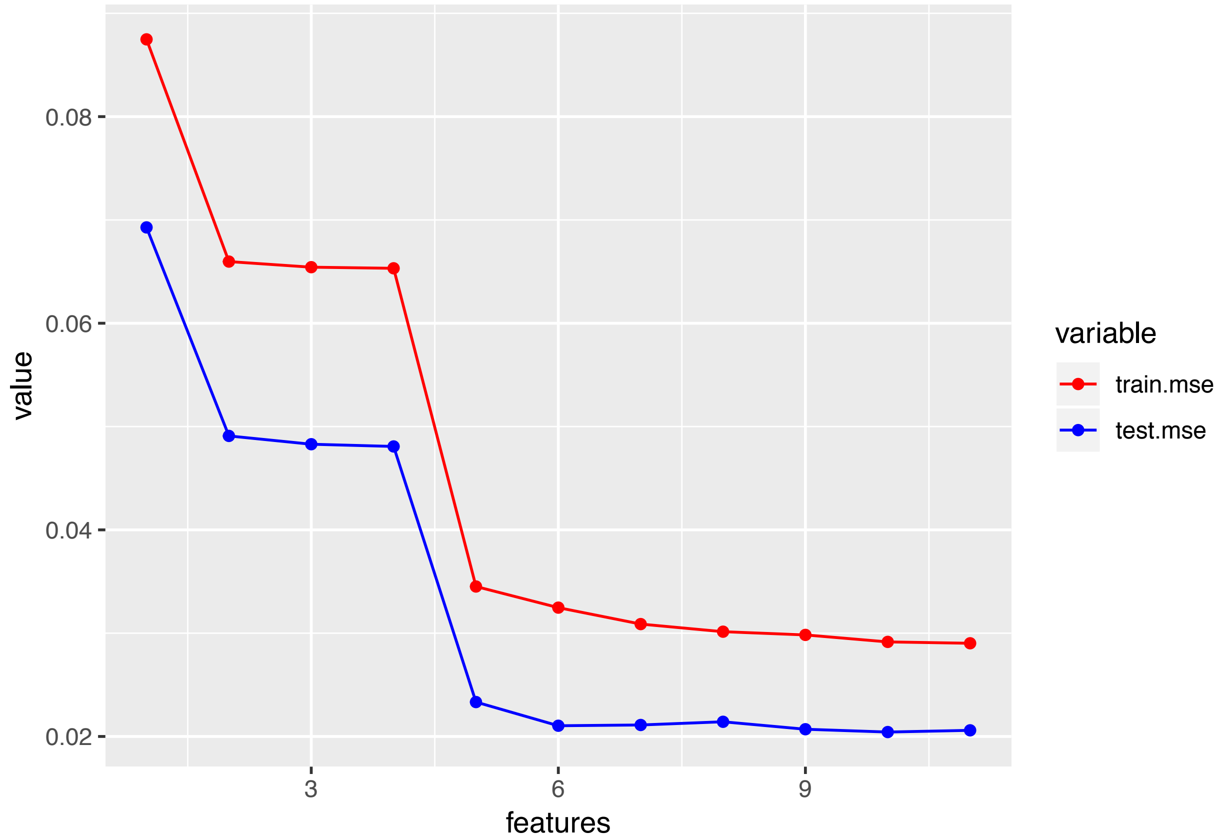
#3.4

the training MSE is decreasing as the number of predictors increase



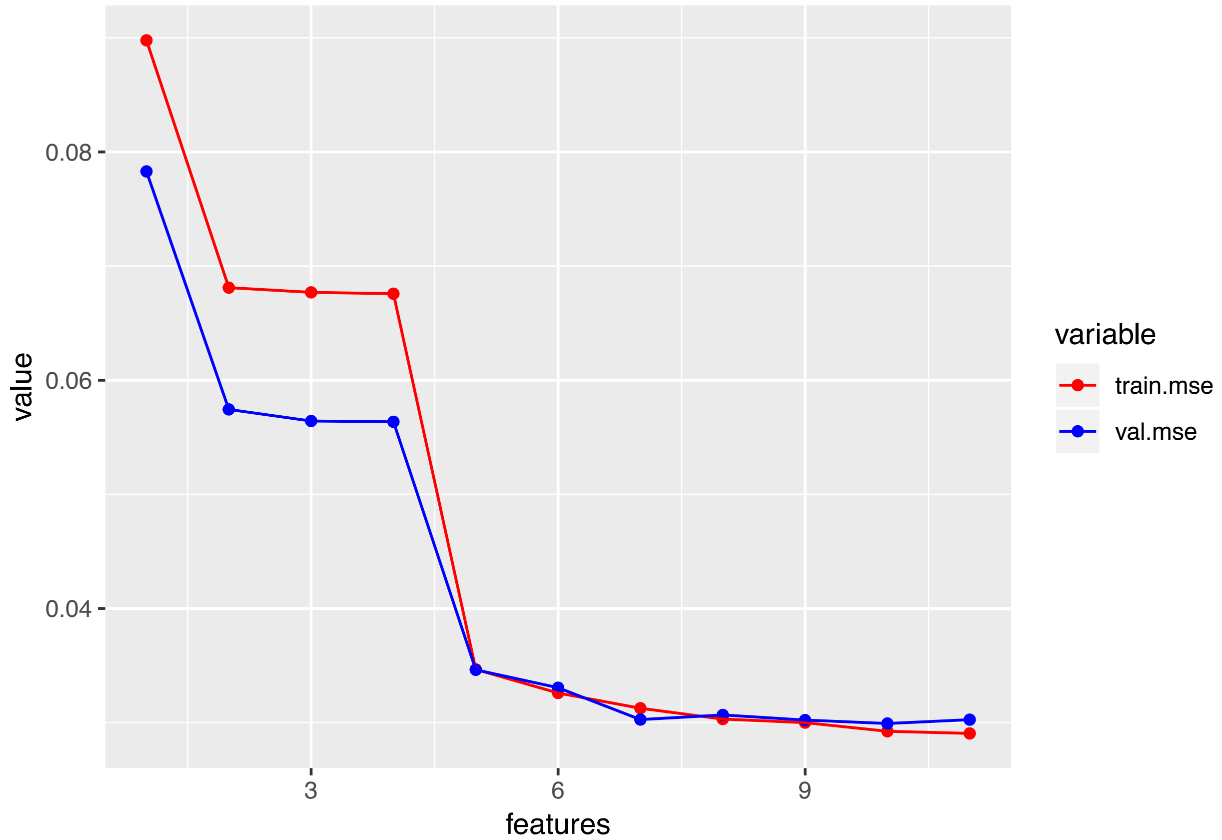
#3.5

test MSE is decreseing in [1:10] and starts increasing at 11 and the 10th model gives the lowest test MSE



#3.2.2

```
modelQualitySingleVal = matrix(rep(0,22),nrow=11)
for (i in 1:length(Xnames)) {
  formula = as.formula(paste("log(SalePrice + 1) ~ ", paste(Xnames[1:i], collapse= "+")))
  lmod = lm(formula, data = AmesTinyActTrain)
  training.label = log(AmesTinyActTrain$SalePrice+1)
  mse = MSE(training.label, model.matrix(lmod),lmod$coefficients)
  modelQualitySingleVal[i,1] = mse

  val.label = log(AmesTinyActVal$SalePrice+1)
  modelQualitySingleVal[i, 2] = sum((val.label-predict(lmod, AmesTinyActVal))^2)/nrow(AmesTinyActVal)
}

dat <- data.frame(features = 1:length(Xnames), train.mse = modelQualitySingleVal[,1], val.mse = modelQua
dat.m <- melt(dat, id.vars = "features")
ggplot(dat.m, aes(features, value, colour = variable)) +
  geom_point() + geom_line(aes(features, value, colour = variable))+
  scale_colour_manual(values = c("red", "blue"))
```




```
which(modelQualitySingleVal[,2] == min(modelQualitySingleVal[,2]))
```

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

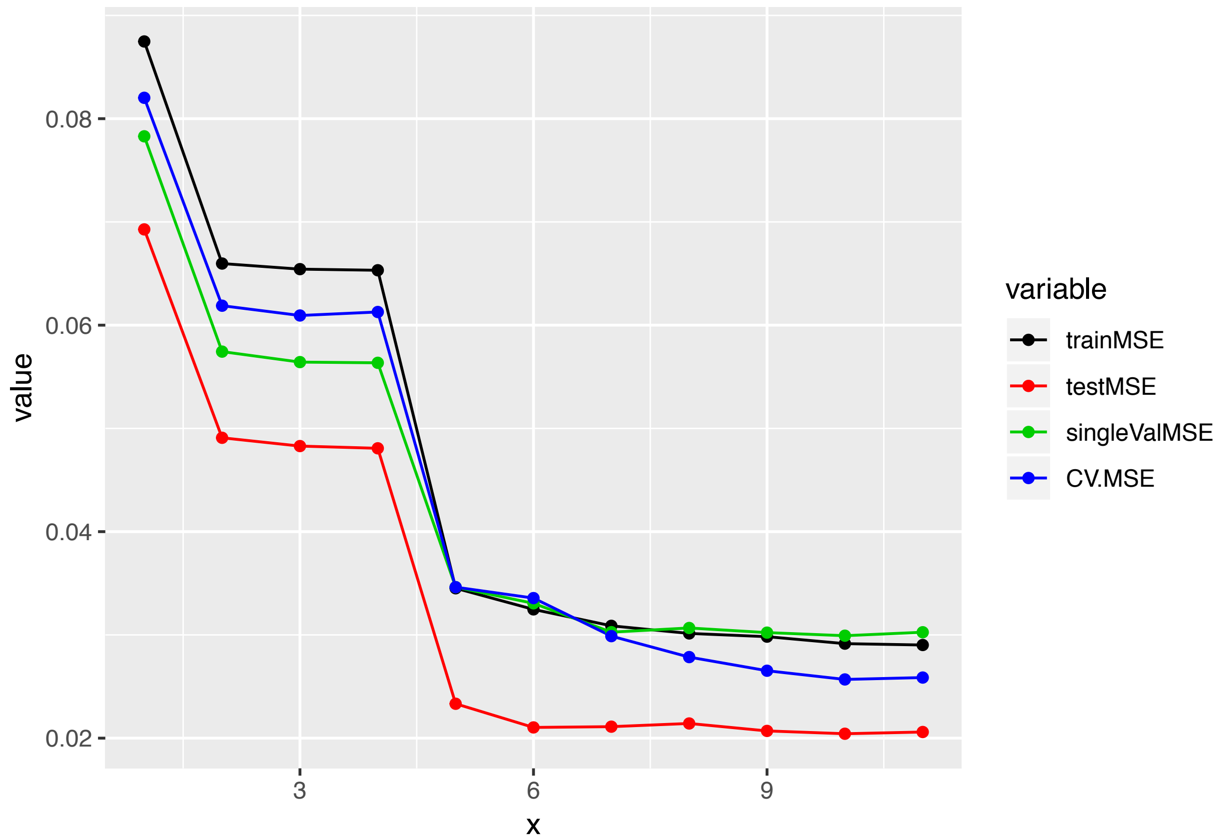
the 10th model gives the lowest validation MSE

#3.3.1

```
set.seed(123)
folds <- createFolds(AmesTinyTrain$SalePrice, k = 5)
cvMSE = matrix(rep(0,55),nrow=11)

for (i in 1:length(Xnames)) {
  formula = as.formula(paste("log(SalePrice + 1) ~ ", paste(Xnames[1:i], collapse= "+")))
  for (f in 1:5) {
    train = AmesTinyTrain[-folds[[f]], ]
    lmod = lm(formula, data = train)
    test = AmesTinyTrain[folds[[1]], ]
    pred<- predict(lmod, test)
    true_y<- log(test$SalePrice + 1)
    mse1 = 1/length(folds[[1]]) * sum((pred-true_y)^2)
    cvMSE[i,f] = mse1
  }
}
```

#3.3.2

```r
dat <- data.frame(x = 1:length(Xnames), trainMSE = modelQuality$MSE, testMSE = test.mse, singleValMSE =
dat.m <- melt(dat, id.vars = "x")
ggplot(dat.m, aes(x, value, colour = variable)) +
  geom_line() + geom_point(aes(x, value, colour = variable))+
  scale_colour_manual(values = 1:5)
```




```r
#10th model
# which(dat$CV.MSE == min(dat$CV.MSE))
```

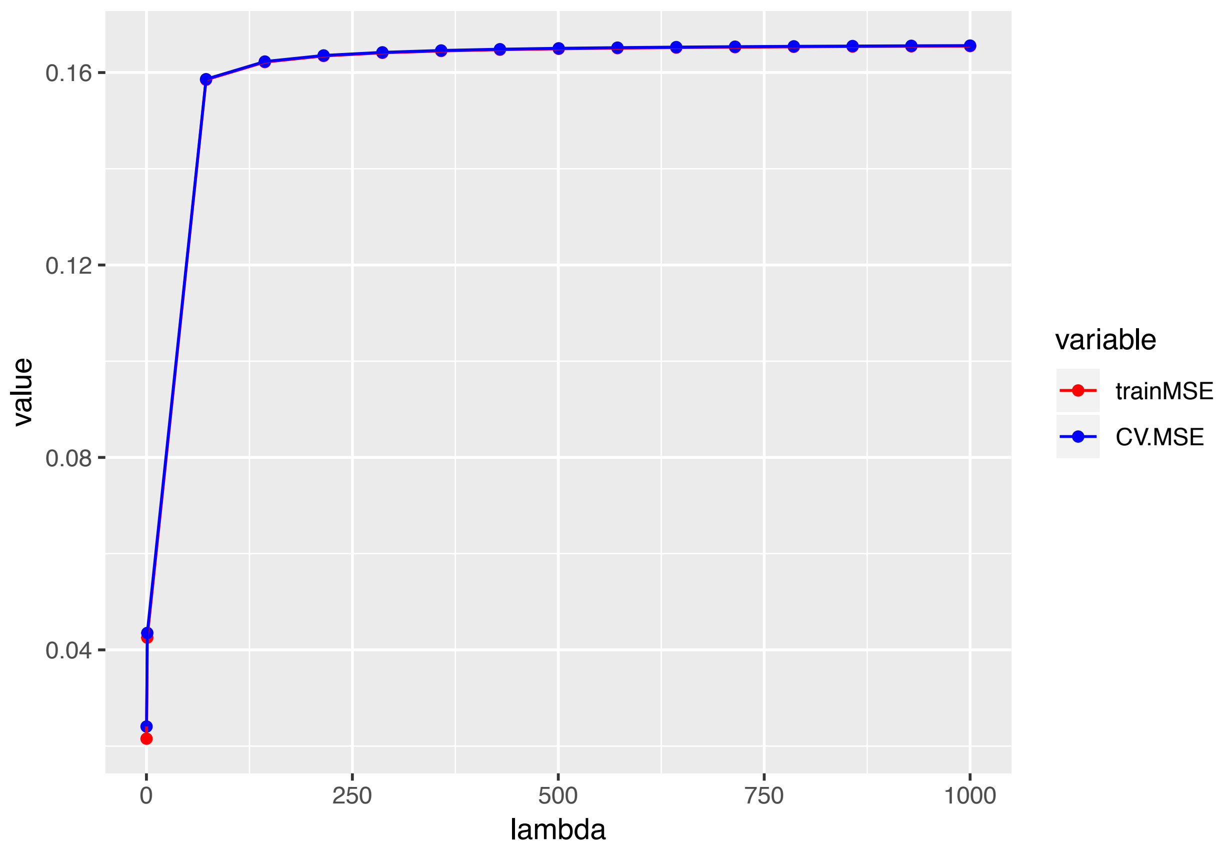
the 10th model gives the lowest CV-MSE

#3.4.1

```
train = AmesTiny[setdiff(names(AmesTiny), c("SalePrice"))]

x_train <- model.matrix( ~ .-1, train)
train.label = log(AmesTiny$SalePrice + 1)
mod.ridge = glmnet(x_train, log(AmesTiny$SalePrice + 1), alpha = 0, lambda = 1)
```

#3.4.2

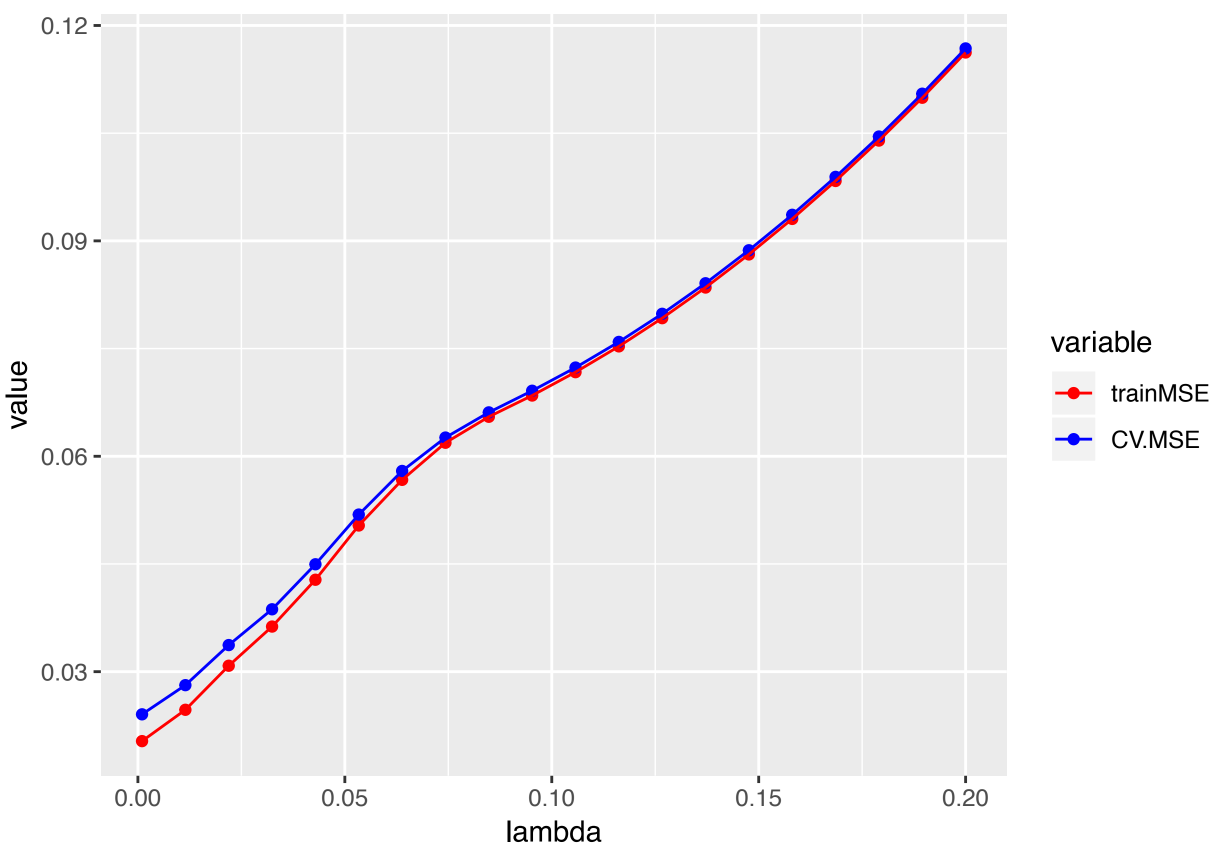
```
lambdas = c(0.1, seq(1,1000,length.out = 15))

training.error = rep(0,length(lambdas))
test.error = rep(0,length(lambdas))

for (i in 1:length(lambdas)) {
  fit.ridge= glmnet(x_train, train.label, alpha=0, lambda = lambdas[i])
  training.error[i] = mean((train.label - predict(fit.ridge, x_train))^2)
}

cvmod = cv.glmnet(x_train, train.label, alpha=0, lambda = lambdas, type.measure = 'mse',nfolds = 5)

dat <- data.frame(lambda = lambdas, trainMSE = training.error, CV.MSE = rev(cvmod$cvm))
dat.m <- melt(dat, id.vars = "lambda")
ggplot(dat.m, aes(lambda, value, colour = variable)) +
  geom_point() + geom_line(aes(lambda, value, colour = variable))+
  scale_colour_manual(values = c("red", "blue"))
```



when $\lambda = 0.1$, we have both minimum training MSE and CV-MSE

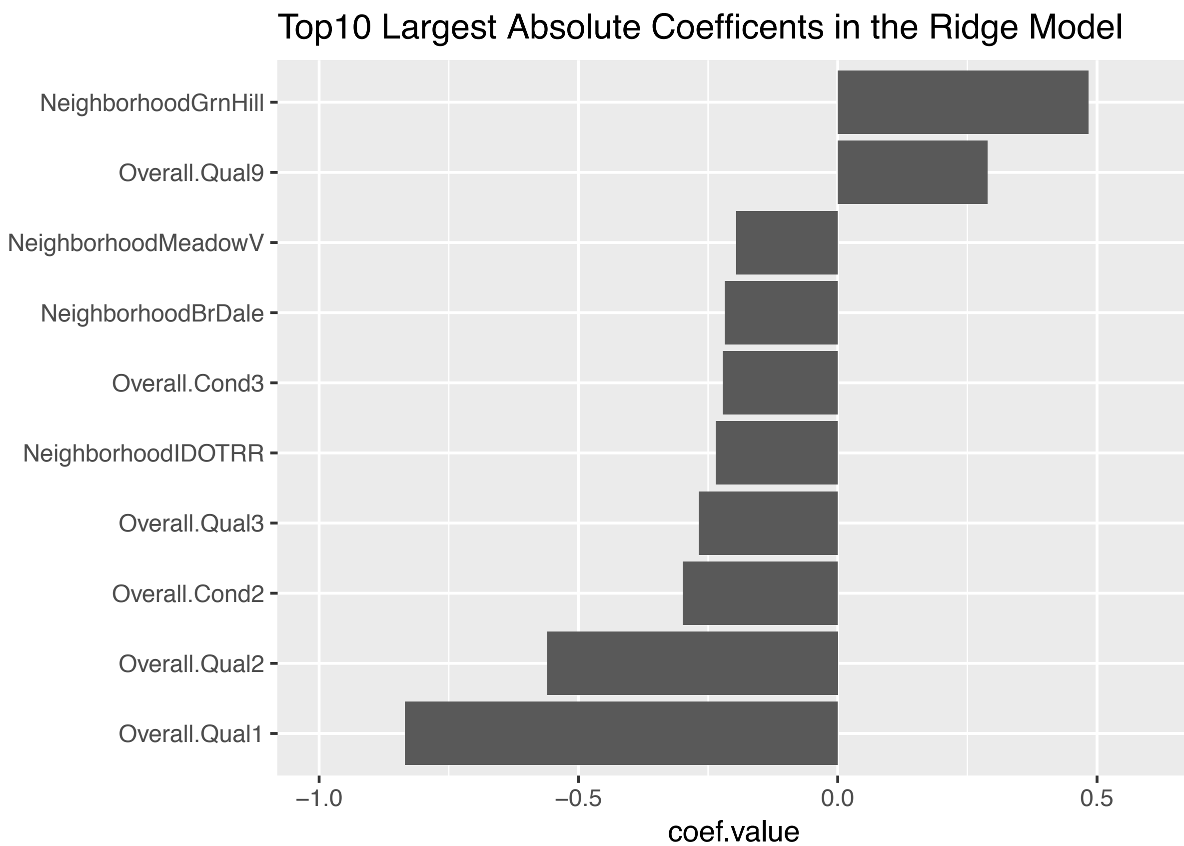
#3.5.1

```r
# cv.mod = cv.glmnet(x_train, train.label, type.measure = 'mse', alpha=1)
# lambdas = cv.mod$lambda
lambdas = seq(0.001,0.2,length.out = 20)
training.error = rep(0,length(lambdas))
test.error = rep(0,length(lambdas))

for (i in 1:length(lambdas)) {
  fit.lasso= glmnet(x_train, train.label, alpha=1, lambda = lambdas[i])
  training.error[i] = mean((train.label - predict(fit.lasso, x_train))^2)
}

cvmod = cv.glmnet(x_train, train.label, alpha=1, lambda = lambdas, type.measure = 'mse',nfolds = 5)

dat <- data.frame(lambda = lambdas, trainMSE = training.error, CV.MSE = rev(cvmod$cvm))
dat.m <- melt(dat, id.vars = "lambda")
ggplot(dat.m, aes(lambda, value, colour = variable)) +
  geom_point() + geom_line(aes(lambda, value, colour = variable))+
  scale_colour_manual(values = c("red", "blue"))
```



#3.6.1

```
## Warning: package 'bindrcpp' was built under R version 3.4.4
```



#3.6.2

```
## Lasso picked 59 variables and eliminated the other 4 variables
```