Lab1

Zijie Feng (zijfe244)

2018-11-20

Assignment 1

1.1

```
## 1.1
data <- readxl::read_xlsx("spambase.xlsx")
n=dim(data)[1]
set.seed(12345)
id <- sample(1:n, floor(n*0.5))
train <- data[id,]
test <- data[-id,]
rm(n,id)
```

1.2

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```
## Confusion matrix of training set:
```

```
##       prediction
## Labels   0   1
##      0 803 142
##      1  81 344
```

```
## error rate: 0.163
```

```
## precision: 0.708
```

```
## recall: 0.809
```

```
## Confusion matrix of test set:
```

```
##       prediction
## Labels   0   1
##      0 791 146
##      1  97 336
```

```
## error rate: 0.177
```

```
## precision: 0.697
```

```
## recall: 0.776
```

The error rate of training set is around 16% and of test set is around 18%. Since both of two error rates are low (<0.2) and close to each other, and the approximately values of precision and recall are high (>=0.7), the model is quickly good.

hint: "IS SPAM" is "TRUE POSITIVE" here

1.3

```
## Confusion matrix of training set:
##       prediction
## Labels   0   1
##      0 944   1
##      1 419   6
## error rate: 0.307
## precision: 0.857
## recall: 0.014
## Confusion matrix of test set:
##       prediction
## Labels   0   1
##      0 936   1
##      1 427   6
## error rate: 0.312
## precision: 0.857
## recall: 0.014
```

The error rate of training set is around 31% and of test set is around 31%. All the error rates are higher than the results in step 2, because the new rule makes the results more difficult to get 1 (spam), which results in the low values of recalls. Therefore, both the training and the test sets are hard to get spam email when predicting, and the model is not good enough.

1.4

```
## Confusion matrix of training set:
##       prediction
## Labels   0   1
##      0 806 139
##      1  98 327
## error rate: 0.173
## precision: 0.702
## recall: 0.769
## Confusion matrix of test set:
##       prediction
## Labels   0   1
##      0 672 265
##      1 184 249
## error rate: 0.328
## precision: 0.484
## recall: 0.575
```

The error rate of training set is around 17% and of test set is around 33%. Although the train error rate of 30-Nearest Neighbor model is close to the one of logistic regression model, the test error rate of 30-Nearest Neighbor model is higher than the others. In this case, such model is worse than the model we get from step 2.

1.5

```
## Confusion matrix of training set:

##       prediction
## Labels   0   1
##      0 945   0
##      1   0 425

## error rate: 0

## precision: 1

## recall: 1

## Confusion matrix of test set:

##       prediction
## Labels   0   1
##      0 648 289
##      1 182 251

## error rate: 0.344

## precision: 0.465

## recall: 0.58
```

When K=1, the result from the training set is extremely great. The error rate is 0 and both precision and recall are 1. However, the result from the test set is not as good as the one from training set. For test set, all the values from model with K=1 are similar with the values from model with K=30. There is no modification when K decreases to 1, but a degeneration occurred (error rate increases). Only considering 1 neighbor is not eough, and training set would have a wrong impact to the new input set. Thus, we can conclude the 1-Nearest Neighbor model is overfitting and it is a bad model.

Assignment 3

3.1

```
## 3.1
k_cross_validation <- function(X_e, Y, Nfolds){
  set.seed(12345)
  n <- nrow(X_e)
  id <- floor(n/Nfolds)
  ids <- list()
  ids_remain <- 1:n
  # set N groups
  for(i in 1:Nfolds){
    if(i!=Nfolds){
      id0 <- sample(1:length(ids_remain), size = id)
    }else{
      id0 <- 1:length(ids_remain) # selected positions in the remain index
    }
    ids[[i]] <- ids_remain[id0]
    ids_remain <- ids_remain[-id0]
  }
  # calculate the total MSE
  SSE <- 0
  for (i in 1:Nfolds) {
    id <- ids[[i]]
    Xtrain <- X_e[-id,]
    Xtest <- X_e[id,]
    Ytrain <- Y[-id]
    Ytest <- Y[id]
    X_ext <- cbind(1,Xtrain)
    mdl <- solve(t(X_ext)%*%X_ext)%*%t(X_ext)%*%Ytrain
    X_ext <- cbind(1,Xtest)
    Ypred <- X_ext%*%mdl
    SSE <- SSE + t((Ypred-Ytest))%*%(Ypred-Ytest)
  }
  SSE/Nfolds
}

best_subset_selection <- function(X, Y, Nfolds){
  # create all the possible combinations of features
  m <- ncol(X)
  idx <- 1:(2^m-1)
  t <- vector()
  mat <- sapply(idx, function(id){
    t <- rbind(t, as.integer(intToBits(id)))
  })
  mat <- mat[1:m,]
  # calculate all the costs for each combinations
  cost <- idx
  for(i in idx){
    X_e <- as.matrix(X[,(1:m)[as.logical(mat[,i])]])
    cost[i] <- k_cross_validation(X_e, Y, Nfolds)
  }
```

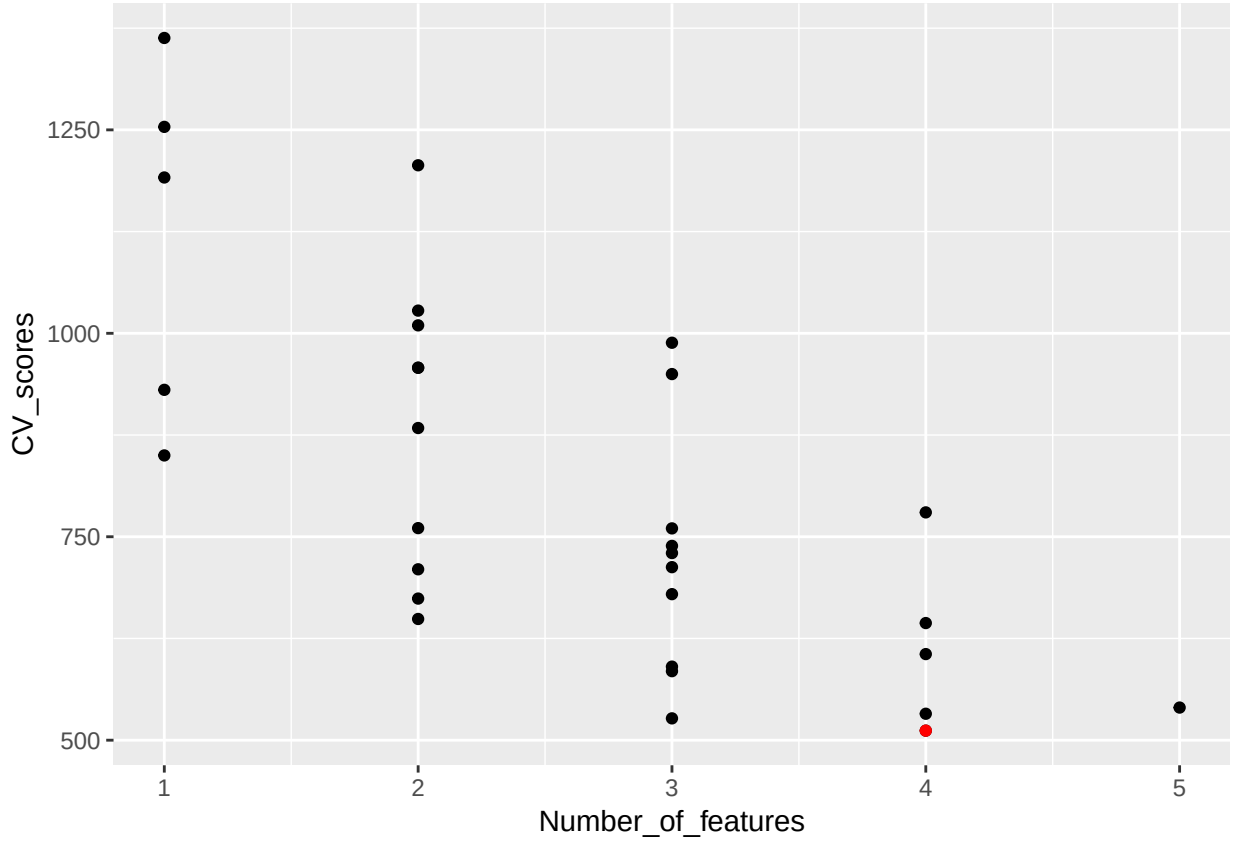
```r
  # select the combination with the smallest cost
  best <- which(cost==min(cost))

  # print and plot
  cat("Key features: ",colnames(X)[(1:m)[as.logical(mat[,best])]] )
  cat("\nMin cost: ",min(cost))
  dt <- data.frame(combination=idx, CV_scores=cost, Number_of_features=colSums(mat))
  ggplot(data=dt, aes(y=CV_scores,x=Number_of_features))+
    geom_point()+
    geom_point(data=dt[best,],aes(y=CV_scores,x=Number_of_features),color="red")
}
```

3.2

Using the data set **swiss**, fertility is the labels Y and the other variables are features X. We calculate the 5fold-cross-validation scores for the selection of best model. The optimal subset of features contains Agriculture, Education, Catholic and Infant.Mortality. The corresponding CV score is 511.7158, which is shown as the red point in the plot.

```
## Key features:  Agriculture Education Catholic Infant.Mortality
## Min cost:  511.7158
```

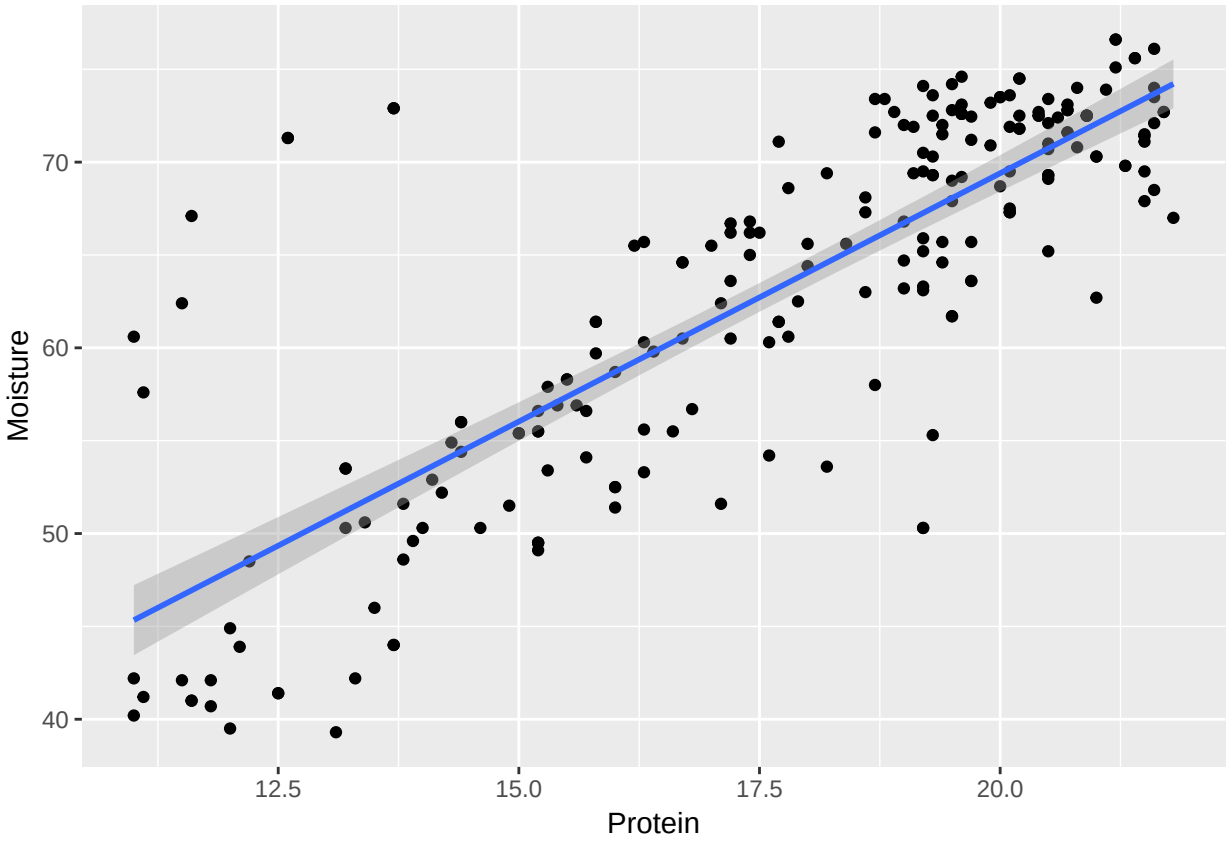


Higher education would negatively affect fertility is reasonable, since people will pay more attentions into eugenics. On the other hands, high infant mortality has a positive effect since high fertility can supplement infant mortality. Catholic also affect fertility based on its religious role and direction, for example, antiabortion. Agriculture could impact fertility since the amount of food is connected to the population as well.

It is hard to say that which feature have the largest impact on the target since the optimal subset always changes with different number of folds and seeds we assume. Perhaps using forward selection algorithm instead of best subset selection can help us know which feature are the most important.

Assignment 4

4.1



Although they are several outliers in the plot, the data seems still have a linear relation between protein and moisture.

4.2

Consider M_i in which moisture is normally distributed, and it is a polynomial function of protein, we can rewrite the model as the following probabilistic model

$$y \sim w_0 + w_1 x + w_2 x^2 + \cdots + w_i x^i + e, \quad e \sim N(0, \sigma^2),$$

or

$$y \sim N(WX^T, \sigma^2),$$

where $W = [w_0, w_1, ..., w_i]$ and $X = [x_1, x_2, ..., w_i]$.

Since the moisture is distributed normally, it is reasonable to deduce that

$$
\begin{aligned}
L(y|w,\sigma^2) =& \frac{1}{\sqrt{2\pi}\sigma}\exp[-\frac{(y-\sum_{k=0}^{i} w_k x^k)^2}{2\sigma^2}],\\
L(D|w,\sigma^2) =& \frac{1}{(\sqrt{2\pi}\sigma)^2}\exp[-\frac{\sum_{l=1}^{n}(y_l-\sum_{k=0}^{i} w_{lk}\cdot x_l^k)^2}{2\sigma^2}],\\
-\log L(D|w,\sigma^2) =& C+\frac{1}{2\sigma^2}\sum_{l=1}^{n}(y_l-\sum_{k=0}^{i} w_{lk}\cdot x_l^k)^2.
\end{aligned}
$$

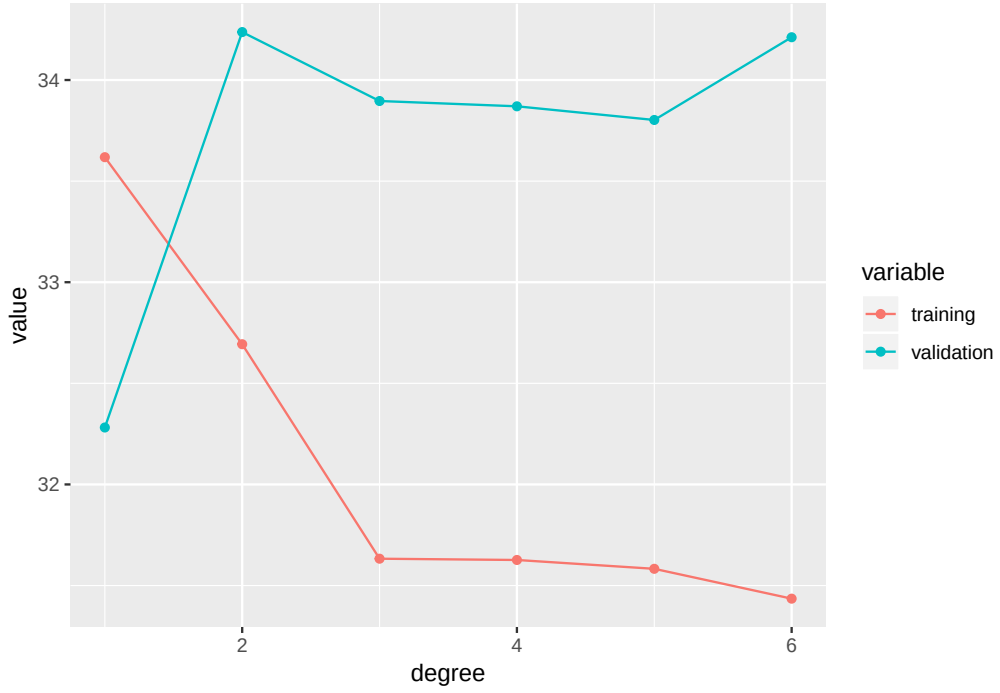
Therefore,

$$
\begin{aligned}
\arg\max_w[L(D|w)] &= \arg\min_w[-\log L(D|w)]\\
&= \arg\min_w[\frac{1}{2\sigma^2}\sum_{l=1}^{n}(y_l-\sum_{k=0}^{i} w_{lk}\cdot x_l^k)^2]\\
&= \arg\min_w[\sum_{l=1}^{n}(y_l-\hat{y}_l)^2]\\
&= \arg\min_w[\frac{1}{n}\sum_{l=1}^{n}(y_l-\hat{y}_l)^2].
\end{aligned}
$$

So the maximum likehood of the parameters w in condition to the data is proportional to minimum of the MSE, which is the reason why MSE criterion can be used for fitting model to a training set.

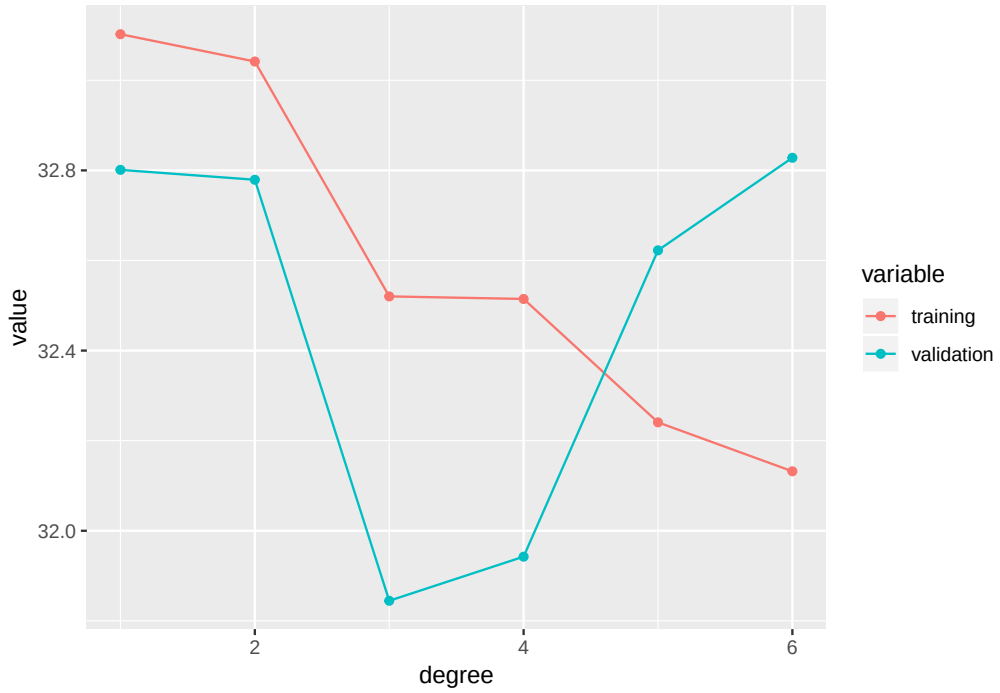
4.3

The following figure shows the MSEs of both training data and validation data with polynomial regressions with different degrees, based on seed(12345).



Frankly speaking, it is hard to consider which model is the best by seed(12345), since all the MSEs of validation set are high besides the model with degree $i = 1$, but the model with degree $i = 1$ has a large training MSE as well. The irregular trend when i is between 1 and 2 may cause by the partition of the data. The outliers mentioned in task 1 are mostly divide into training data, and it may lead to an increase of error. But in testing data, there are only 3 of the outliers, which will only lead to a small gap between data and the fitted line.

The following figure is created based on seed(123), which is better to interpret in accordance with bias-variance tradeoff. The training MSE will decrease gradually when the degree of model grows, but the validation MSE will experience an increase after decrease to some extents. The best model must have low and similar values of training and validation MSEs. According to the seed(123), the model with degree 4 or 5 might be the best model.



4.4

There are 64 variables (63 channels, 1 intercept) selected by `stepAIC` with both directions.

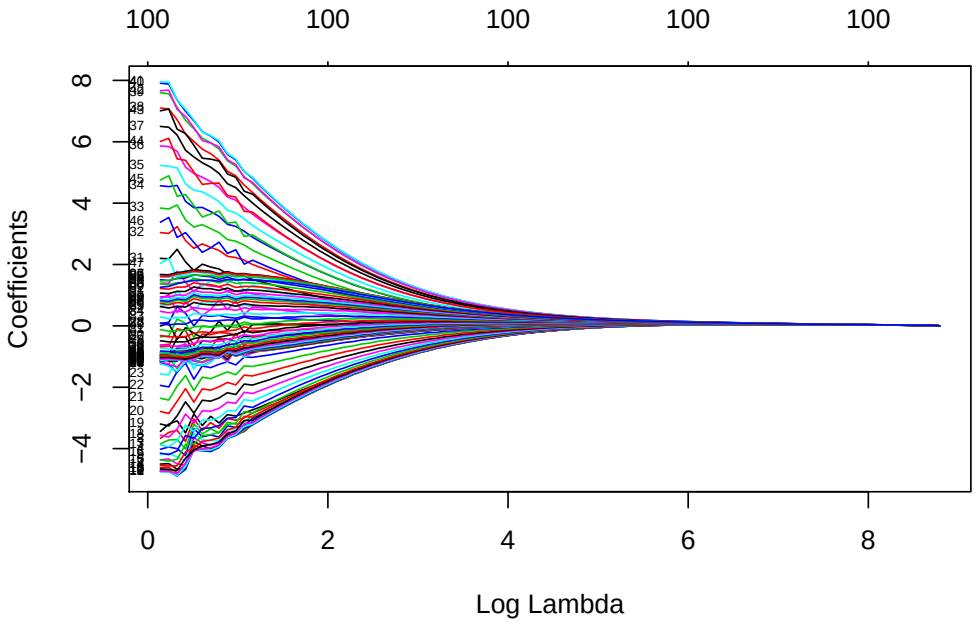
```
## Number of remaining variables: 64
```

```
##   (Intercept)      Channel1      Channel2      Channel4      Channel5
##      7.093133  10559.893784 -12636.966607   8489.323117 -10408.966948
##      Channel7      Channel8     Channel11     Channel12     Channel13
##  -5376.017738   7215.595409  -9505.520235  37240.918374 -41564.546571
##     Channel14     Channel15     Channel17     Channel19     Channel20
##  34938.179314 -23761.450875   4296.572462  14279.808102 -23855.616123
##     Channel22     Channel24     Channel25     Channel26     Channel28
##  18444.905722 -20138.426065  18137.431996  -7670.318234  20079.898191
##     Channel29     Channel30     Channel32     Channel34     Channel36
## -36351.013717  18071.275531   3838.013358  -9242.884498   8070.938452
##     Channel37     Channel39     Channel40     Channel41     Channel42
##  -9045.587624  18664.454171 -20069.708579  22257.776227 -21760.853228
##     Channel45     Channel46     Channel47     Channel48     Channel50
```

```
##  18145.803786  -8225.696060  -4986.549169   2876.074542 -13009.409717
##     Channel51     Channel52     Channel54     Channel55     Channel56
##  29251.160946 -26833.976402  30954.861519 -35183.287363  14912.986496
##     Channel59     Channel60     Channel61     Channel63     Channel64
##  -8030.277501  13071.415506  -7850.189324  15059.274961 -19909.466348
##     Channel65     Channel67     Channel68     Channel69     Channel71
##   4190.183533  13850.508143 -25873.365427  18362.384676  -9223.909939
##     Channel73     Channel74     Channel78     Channel79     Channel80
##  12456.497755  -5624.411385  -7927.104791  15473.187794 -22391.894812
##     Channel81     Channel84     Channel85     Channel87     Channel88
##  13852.452651 -11442.629734  20228.671387 -15938.315283   5647.072201
##     Channel92     Channel94     Channel98     Channel99
##   6595.995241  -5497.846381  -8728.596111   8554.587048
```

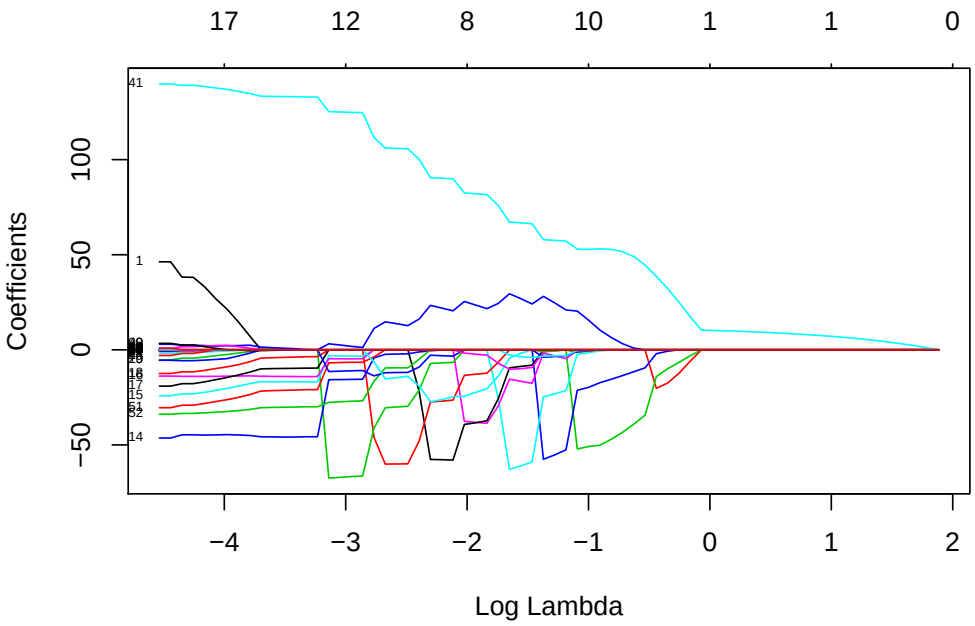
4.5

When lambda is larger, all the coefficients of ridge regression would tend to 0. The number of coefficients will not decrease.



4.6

Compared with the paths in step 5, all the coefficients will also go to 0 finally, but some of the coefficients can jump away from 0 sometimes and the number of coefficients will decrease gradually. Most of coefficients of LASSO model (without defining bounds) are much larger than coefficients of ridge model. It seems that Lasso method would chose the best combination of features depending on different lambdas. Additionally in the LASSO model, the larger the coefficient is, the later it becomes 0.

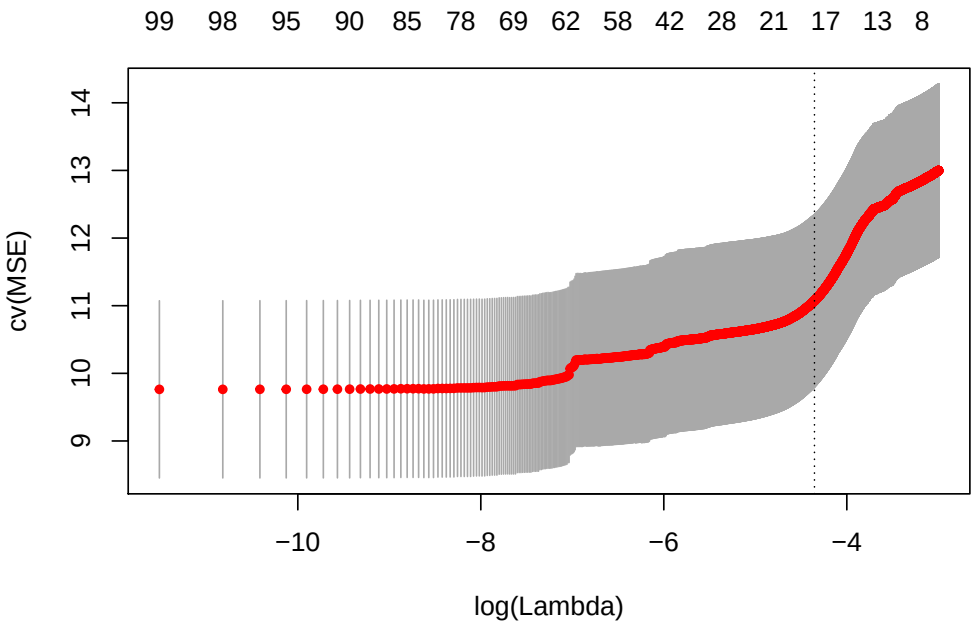


4.7

```
## lambda.min: 0
## lambda.1se: 0.01286
```

```
## Number of remaining variables: 18
```

```
## (Intercept)    Channel1   Channel16   Channel17   Channel18   Channel19
##  23.4489368  33.8013282 -47.6926269 -37.0737596 -19.3589784 -10.5970043
##   Channel20   Channel21   Channel22   Channel23   Channel40   Channel41
##  -5.4264846  -1.5018192  -0.4841268  -0.1243477  80.4779822  66.3823414
##   Channel49   Channel50   Channel51   Channel98   Channel99  Channel100
##  -1.3072953 -17.3683868 -48.1188356   2.4265824   1.9311971   0.2034079
```



By searching in `seq(0,0.01,0.000001)`, the optimal lambda (lambda.1se) is around 0.01286. Its log value is shown as vertical line on right part of the following figure. The number of remaining variables is 18 (with intercept). Additionally, the CV scores will increase when lambda grows.

4.8

```
##     AIC_MSE LASSO_MSE
## 1 0.8598985  9.827187
```

With appropriate lambda, the LASSO model removes variables more strictly than the model with AIC regularization. This is the result why LASSO regression could take feature selection and penalyze large coefficients by turning them to 0. However, some of the remaining variables are same in both regularized models, and AIC regularization can provide a more accurate model than LASSO regularization, because AIC model has smaller MSE.

Apdendix

```
knitr::opts_chunk$set(echo = TRUE)
library(kknn)
library(ggplot2)
library(reshape2)
library(glmnet)
library(MASS)
library(doParallel)
## 1.1
data <- readxl::read_xlsx("spambase.xlsx")
n=dim(data)[1]
set.seed(12345)
id <- sample(1:n, floor(n*0.5))
train <- data[id,]
test <- data[-id,]
rm(n,id)
## 1.2
glm <- glm(formula = Spam~., family = binomial(link = "logit"), data = train)
ptrain <- predict(glm, train, type = "response") # the results are probabilities
ptest <- predict(glm, test, type = "response")
train$predict <- as.numeric(ptrain>0.5)
test$predict <- as.numeric(ptest>0.5)

mat_train <- table(train$Spam,train$predict,dnn = c("Labels","prediction"))
mat_test <- table(test$Spam,test$predict,dnn = c("Labels","prediction"))

error_train <- round(1-(sum(diag(mat_train))/dim(train)[1]),3)
error_test <- round(1-(sum(diag(mat_test))/dim(test)[1]),3)
cat("Confusion matrix of training set:\n")
mat_train
cat(paste0("error rate: ", error_train,"\n"))
cat(paste0("precision: ", round(mat_train[2,2]/sum(mat_train[,2]),3),"\n"))
cat(paste0("recall: ", round(mat_train[2,2]/sum(mat_train[2,]),3),"\n"))
cat("\n")
cat("Confusion matrix of test set:\n")
mat_test
cat(paste0("error rate: ", error_test,"\n"))
cat(paste0("precision: ", round(mat_test[2,2]/sum(mat_test[,2]),3),"\n"))
cat(paste0("recall: ", round(mat_test[2,2]/sum(mat_test[2,]),3),"\n"))
## 1.3
train$predict <- as.numeric(ptrain>0.9)
test$predict <- as.numeric(ptest>0.9)

mat_train <- table(train$Spam,train$predict,dnn = c("Labels","prediction"))
mat_test <- table(test$Spam,test$predict,dnn = c("Labels","prediction"))

error_train <- round(1-(sum(diag(mat_train))/dim(train)[1]),3)
error_test <- round(1-(sum(diag(mat_test))/dim(test)[1]),3)
cat("Confusion matrix of training set:\n")
mat_train
cat(paste0("error rate: ", error_train,"\n"))
cat(paste0("precision: ", round(mat_train[2,2]/sum(mat_train[,2]),3),"\n"))
```

```r
cat(paste0("recall: ", round(mat_train[2,2]/sum(mat_train[2,]),3),"\n"))
cat("\n")
cat("Confusion matrix of test set:\n")
mat_test
cat(paste0("error rate: ", error_test,"\n"))
cat(paste0("precision: ", round(mat_test[2,2]/sum(mat_test[,2]),3),"\n"))
cat(paste0("recall: ", round(mat_test[2,2]/sum(mat_test[2,]),3),"\n"))
## 1.4
train$Spam <- as.factor(train$Spam)   # the prediction should be factor variable, not numeric
test$Spam <- as.factor(test$Spam)

mdl <- kknn(Spam~.,train = train, test = train, k = 30)
ptrain <- mdl$fitted.values
mdl <- kknn(Spam~.,train = train, test = test, k = 30)
ptest <- mdl$fitted.values
train$predict <- ptrain
test$predict <- ptest

mat_train <- table(train$Spam,train$predict,dnn = c("Labels","prediction"))
mat_test <- table(test$Spam,test$predict,dnn = c("Labels","prediction"))

error_train <- round(1-(sum(diag(mat_train))/dim(train)[1]),3)
error_test <- round(1-(sum(diag(mat_test))/dim(test)[1]),3)
cat("Confusion matrix of training set:\n")
mat_train
cat(paste0("error rate: ", error_train,"\n"))
cat(paste0("precision: ", round(mat_train[2,2]/sum(mat_train[,2]),3),"\n"))
cat(paste0("recall: ", round(mat_train[2,2]/sum(mat_train[2,]),3),"\n"))
cat("\n")
cat("Confusion matrix of test set:\n")
mat_test
cat(paste0("error rate: ", error_test,"\n"))
cat(paste0("precision: ", round(mat_test[2,2]/sum(mat_test[,2]),3),"\n"))
cat(paste0("recall: ", round(mat_test[2,2]/sum(mat_test[2,]),3),"\n"))
## 1.5
mdl <- kknn(Spam~.,train = train, test = train, k = 1)
ptrain <- mdl$fitted.values
mdl <- kknn(Spam~.,train = train, test = test, k = 1)
ptest <- mdl$fitted.values
train$predict <- ptrain
test$predict <- ptest

mat_train <- table(train$Spam,train$predict,dnn = c("Labels","prediction"))
mat_test <- table(test$Spam,test$predict,dnn = c("Labels","prediction"))

error_train <- round(1-(sum(diag(mat_train))/dim(train)[1]),3)
error_test <- round(1-(sum(diag(mat_test))/dim(test)[1]),3)
cat("Confusion matrix of training set:\n")
mat_train
cat(paste0("error rate: ", error_train,"\n"))
cat(paste0("precision: ", round(mat_train[2,2]/sum(mat_train[,2]),3),"\n"))
cat(paste0("recall: ", round(mat_train[2,2]/sum(mat_train[2,]),3),"\n"))
cat("\n")
```

```r
cat("Confusion matrix of test set:\n")
mat_test
cat(paste0("error rate: ", error_test,"\n"))
cat(paste0("precision: ", round(mat_test[2,2]/sum(mat_test[,2]),3),"\n"))
cat(paste0("recall: ", round(mat_test[2,2]/sum(mat_test[2,]),3),"\n"))
## 3.1
k_cross_validation <- function(X_e, Y, Nfolds){
  set.seed(12345)
  n <- nrow(X_e)
  id <- floor(n/Nfolds)
  ids <- list()
  ids_remain <- 1:n
  # set N groups
  for(i in 1:Nfolds){
    if(i!=Nfolds){
      id0 <- sample(1:length(ids_remain), size = id)
    }else{
      id0 <- 1:length(ids_remain) # selected positions in the remain index
    }
    ids[[i]] <- ids_remain[id0]
    ids_remain <- ids_remain[-id0]
  }
  # calculate the total MSE
  SSE <- 0
  for (i in 1:Nfolds) {
    id <- ids[[i]]
    Xtrain <- X_e[-id,]
    Xtest <- X_e[id,]
    Ytrain <- Y[-id]
    Ytest <- Y[id]
    X_ext <- cbind(1,Xtrain)
    mdl <- solve(t(X_ext)%*%X_ext)%*%t(X_ext)%*%Ytrain
    X_ext <- cbind(1,Xtest)
    Ypred <- X_ext%*%mdl
    SSE <- SSE + t((Ypred-Ytest))%*%(Ypred-Ytest)
  }
  SSE/Nfolds
}

best_subset_selection <- function(X, Y, Nfolds){
  # create all the possible combinations of features
  m <- ncol(X)
  idx <- 1:(2^m-1)
  t <- vector()
  mat <- sapply(idx, function(id){
    t <- rbind(t, as.integer(intToBits(id)))
  })
  mat <- mat[1:m,]
  # calculate all the costs for each combinations
  cost <- idx
  for(i in idx){
    X_e <- as.matrix(X[,(1:m)[as.logical(mat[,i])]])
    cost[i] <- k_cross_validation(X_e, Y, Nfolds)
```

```
  }
  # select the combination with the smallest cost
  best <- which(cost==min(cost))

  # print and plot
  cat("Key features: ",colnames(X)[(1:m)[as.logical(mat[,best])]] )
  cat("\nMin cost: ",min(cost))
  dt <- data.frame(combination=idx, CV_scores=cost, Number_of_features=colSums(mat))
  ggplot(data=dt, aes(y=CV_scores,x=Number_of_features))+
    geom_point()+
    geom_point(data=dt[best,],aes(y=CV_scores,x=Number_of_features),color="red")
}
## 3.2
X <- as.matrix(scale(swiss[,2:6]))
Y <- as.numeric(swiss[,1])
Nfolds <- 5
best_subset_selection(X,Y,Nfolds)
## 4.1
data <- readxl::read_xlsx("tecator.xlsx")
ggplot(data=data, aes(x=Protein,y=Moisture))+
  geom_point()+
  geom_smooth(method="lm")
## 4.3
set.seed(12345);
id <- sample(1:nrow(data),nrow(data))
train <- as.data.frame(data[id[1:107],c("Protein","Moisture")])
val <- as.data.frame(data[id[108:215],c("Protein","Moisture")])
MSE_train <- vector()
MSE_val <- vector()
for(i in 1:6){
  md <- lm(Moisture ~ poly(Protein, degree=i), data = train)
  pred1 <- predict(md,train)
  MSE_train[i] <- mean((train[,2]-pred1)^2)
  pred2 <- predict(md,val)
  MSE_val[i] <- mean((val[,2]-pred2)^2)
}
dt <- data.frame(degree=1:6,training=MSE_train, validation=MSE_val)
dt1 <- melt(dt, id="degree")
ggplot(data = dt1, aes(x=degree, y=value, color=variable))+
  geom_line()+
  geom_point()
set.seed(123)
id <- sample(1:nrow(data),nrow(data))
train <- as.data.frame(data[id[1:107],c("Protein","Moisture")])
val <- as.data.frame(data[id[108:215],c("Protein","Moisture")])
MSE_train <- vector()
MSE_val <- vector()
for(i in 1:6){
  md <- lm(Moisture ~ poly(Protein, degree=i), data = train)
  pred1 <- predict(md,train)
  MSE_train[i] <- mean((train[,2]-pred1)^2)
  pred2 <- predict(md,val)
  MSE_val[i] <- mean((val[,2]-pred2)^2)
```

```
}
dt <- data.frame(degree=1:6,training=MSE_train, validation=MSE_val)
dt1 <- melt(dt, id="degree")
ggplot(data = dt1, aes(x=degree, y=value, color=variable))+
  geom_line()+
  geom_point()
##4.4
data_fat <- as.data.frame(data[,-c(1,103,104)])
md_AIC <- lm(Fat ~ ., data = data_fat)
mdl_AIC <- stepAIC(md_AIC, direction = 'both',trace = FALSE )
cat("Number of remaining variables:",length(mdl_AIC$coefficients),"\n")
mdl_AIC$coefficients
##4.5
X <- as.matrix(data_fat[,1:100])
Y <- as.matrix(data_fat[,101])
md_RR <- glmnet(X, Y,
                alpha = 0, family = "gaussian")
plot(md_RR, xvar="lambda", label=TRUE)
##4.6
md_LASSO <- glmnet(X, Y,
                alpha = 1, family = "gaussian")
plot(md_LASSO, xvar="lambda", label=TRUE)
##4.7
clnum<-parallel::detectCores()
cl <- parallel::makeCluster(getOption("cl.cores", clnum))
registerDoParallel(cl)
set.seed(12345)
cvfit=cv.glmnet(X, Y,family = "gaussian",
                alpha = 1,
                type.measure = "mse",
                lambda = seq(0,0.05,0.00001),
                parallel = TRUE
                )
stopCluster(cl)
cat("lambda.min:",cvfit$lambda.min,"\nlambda.1se:",cvfit$lambda.1se,"\n")
co <- coef(cvfit,s=cvfit$lambda.1se)
cat("Number of remaining variables:",length(co@x),"\n")
print(co[co[,1]!=0,])
plot(cvfit, ylab="cv(MSE)")
##4.8
pAIC <- predict(mdl_AIC, data_fat[,1:100])
eAIC <- mean((pAIC-data_fat$Fat)^2 )
p <- predict(cvfit, newx=as.matrix(data_fat[,1:100]), s="lambda.1se")
eLASSO <- mean((p-data_fat$Fat)^2 )

mse <- data.frame(AIC_MSE=eAIC, LASSO_MSE=eLASSO)
mse
```