Group A5

Group A5

2018-11-25

Assignment 1

from *Jiawei Wu*

1.1

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

1.2

```
## [1] "Confusion matrix for testing data is:"
```

```
##       predicted
## actual    0    1  Sum
##    0    791  146  937
##    1     97  336  433
##    Sum  888  482 1370
```

```
## [1] "Misclassification rate for testing data is:"
```

```
## [1] 0.1773723
```

```
## [1] "Confusion matrix for training data is:"
```

```
##       predicted
## actual    0    1  Sum
##    0    803  142  945
##    1     81  344  425
##    Sum  884  486 1370
```

```
## [1] "Misclassification rate for training data is:"
```

```
## [1] 0.1627737
```

As the result shows, when $P(Y = 1|X) > 0.5$ the misclassification rates of the training data and testing data are 0.1628 and 0.1774, the probability of misclassification between training and testing data is similar and it is around 0.17. And as we can see from the matrixes, this model will have larger probability to predict regular emails as the spam.

1.3

```
## [1] "Confusion matrix for testing data is:"
```

```
##       predicted
## actual    0    1  Sum
##    0    936    1  937
##    1    427    6  433
##    Sum 1363    7 1370
```

```
## [1] "Misclassification rate for testing data is:"
```

```
## [1] 0.3124088
```

```
## [1] "Confusion matrix for training data is:"
```

```
##       predicted
## actual    0    1  Sum
##    0    944    1  945
##    1    419    6  425
##    Sum 1363    7 1370
```

```
## [1] "Misclassification rate for training data is:"
```

```
## [1] 0.3065693
```

As the result shows, when decision level $P(Y = 1|X)$ is larger than 0.9 the misclassification rates of the training data and testing data are 0.3066 and 0.3124, the probability of misclassification between training and testing data is similar, and means the probability of misclassification of a email is around 0.31. The probability is so high that we can say it is a bad model. And from the Confusion matrix we can notice that nearly all the spam emails are predicted as the regular emails. This may cause by the high decision level which is so flexible that let most non-spam email misclassified.

1.4

```
## [1] "Confusion matrix for testing data is:"
```

```
##       predicted
## actual    0    1  Sum
##    0    672  265  937
##    1    187  246  433
##    Sum  859  511 1370
```

```
## [1] "Misclassification rate for testing data is:"
```

```
## [1] 0.329927
```

```
## [1] "Confusion matrix for training data is:"
```

```
##       predicted
## actual    0    1  Sum
##    0    807  138  945
##    1     98  327  425
##    Sum  905  465 1370
```

```
## [1] "Misclassification rate for training data is:"
```

```
## [1] 0.1722628
```

As the result shows, misclassification rates for training data and testing data are 0.1723 and 0.3299. Compared with the step 2 we can notice that the misclassification rates using k-nearest neighbor method is quite different between using training data and testing data. And the rates of the training data is lower than training data, which means the model will fit training data better than the testing data. But in step 2, the difference is not obverious, which means the model can fit the training data and testing data in same level.

1.5

```
## [1] "Confusion matrix for testing data is:"
```

```
##       predicted
## actual    0    1  Sum
##    0    640  297  937
##    1    177  256  433
##    Sum  817  553 1370
```

```
## [1] "Misclassification rate for testing data is:"
```

```
## [1] 0.3459854
```

```
## [1] "Confusion matrix for training data is:"
```

```
##       predicted
## actual    0    1  Sum
##    0    945    0  945
##    1      0  425  425
##    Sum  945  425 1370
```

```
## [1] "Misclassification rate for training data is:"
```

```
## [1] 0
```

Compared with results in step 4, when k=1 the misclassification rate of testing data is larger than that when k=30 and the misclassification rate of training data is smaller than that when k=30.

We can notice that decrease of k lead to the increase of the misclassification rate of testing data and lead to the decrease of misclassification rate of training data. And when k is 1, the model will fit the training data perfectly and misclassification rate is 0. It is because when the k is decreasing we consider more errors and it will lead to a high variance and low bias of the model, which means the model may overfitted the training data, so the misclassification of testing data will increase.

Assignment 3

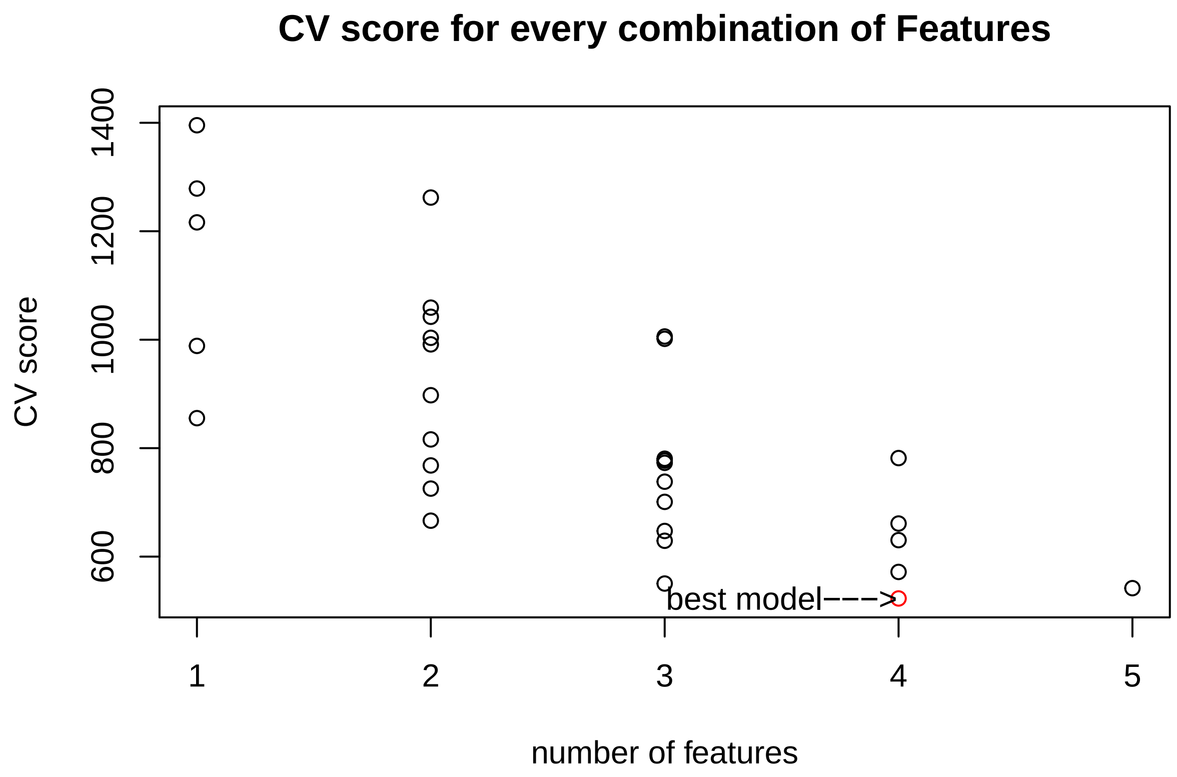
from *Andreas Christopoulos Charitos*

3.1

```
feature_selection<-function(X,Y,N){

  n<-ncol(X)
  idx<-1:2^n-1
  t<-vector()
  mat<-sapply(idx, function(id){
    t<-cbind(t,as.integer(intToBits(id)))
    t})
  m<-mat[1:n,2:ncol(mat)]
  ######################################
```

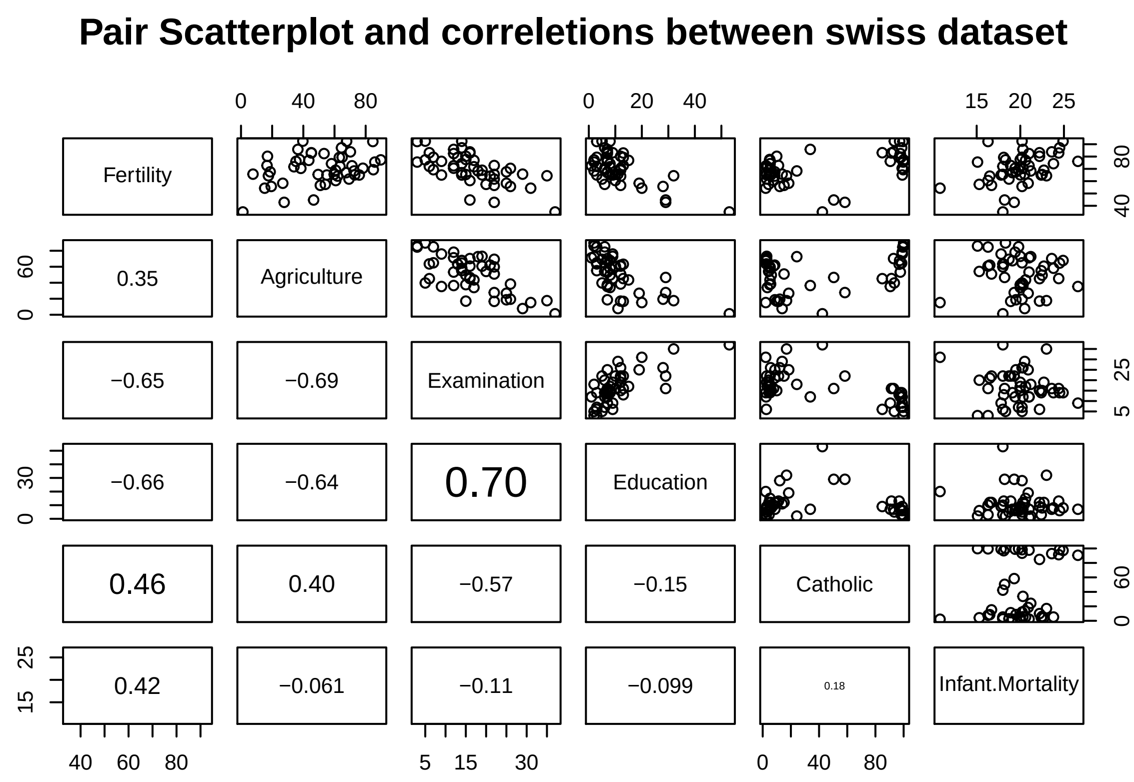
```r
  set.seed(12345)
  #X<-X[sample(nrow(X)),]
  #Y<-Y[sample(length(Y))]
  id<-sample(nrow(X))
  X<-X[id,]
  Y<-Y[id]
  #Create N equally size folds
  folds <- cut(seq(1,nrow(X)),breaks=N,labels=FALSE)
  d<-matrix(0,nrow=N,ncol=dim(m)[2])
  n_features<-rep(0,ncol(d))
  for (i in 1:ncol(m)){
    x<-X[which(m[,i]==1)]
    n_features[i]<-ncol(x)
    for(j in 1:N){

      testIndexes <- which(folds==j,arr.ind=TRUE)
      testX <- as.matrix(x[testIndexes, ])
      trainX <- as.matrix(x[-testIndexes, ])
      testy<-Y[testIndexes]
      trainy<-Y[-testIndexes]
      trainX<-cbind(1,trainX)
      testX<-cbind(1,testX)

      w<-round(as.vector(solve(t(trainX)%*%trainX)%*%t(trainX)%*%trainy),3)
      y_pred<-round(as.matrix(testX%*%w),3)
      sse<-sum((testy-y_pred)^2)
      d[j,i]<-sse
    }
  }
 d<-d
 s<-apply(d, MARGIN = 2, function(x) mean(x, na.rm=TRUE))
  bindex<-which(s==min(s))
  best_comb<-X[which(m[,bindex]==1)]

  plot(x=n_features,y=s,type="p",xlab="number of features",ylab="CV score",col=ifelse(s==s[bindex],"red
  text(x=3.5,y=522.8431,labels=c("best model--->"))
  return(list("best combination"=colnames(best_comb),"best cv score"=s[bindex]))
}
```

3.2



```
## $`best combination`
## [1] "Agriculture"      "Education"        "Catholic"
## [4] "Infant.Mortality"
##
## $`best cv score`
## [1] 522.8431
```

We tested the linear model for every combination of the 5 independent features ("Agriculture","Examination","Education", "Catholic","Infant.Mortality") which is 31 diffrent models and evaluating each one with 5-fold Cross Validation we observe that the best combination of features predicting "Fertility" is ("Agriculture","Education", "Catholic","Infant.Mortality") and located as the red circle in plot.



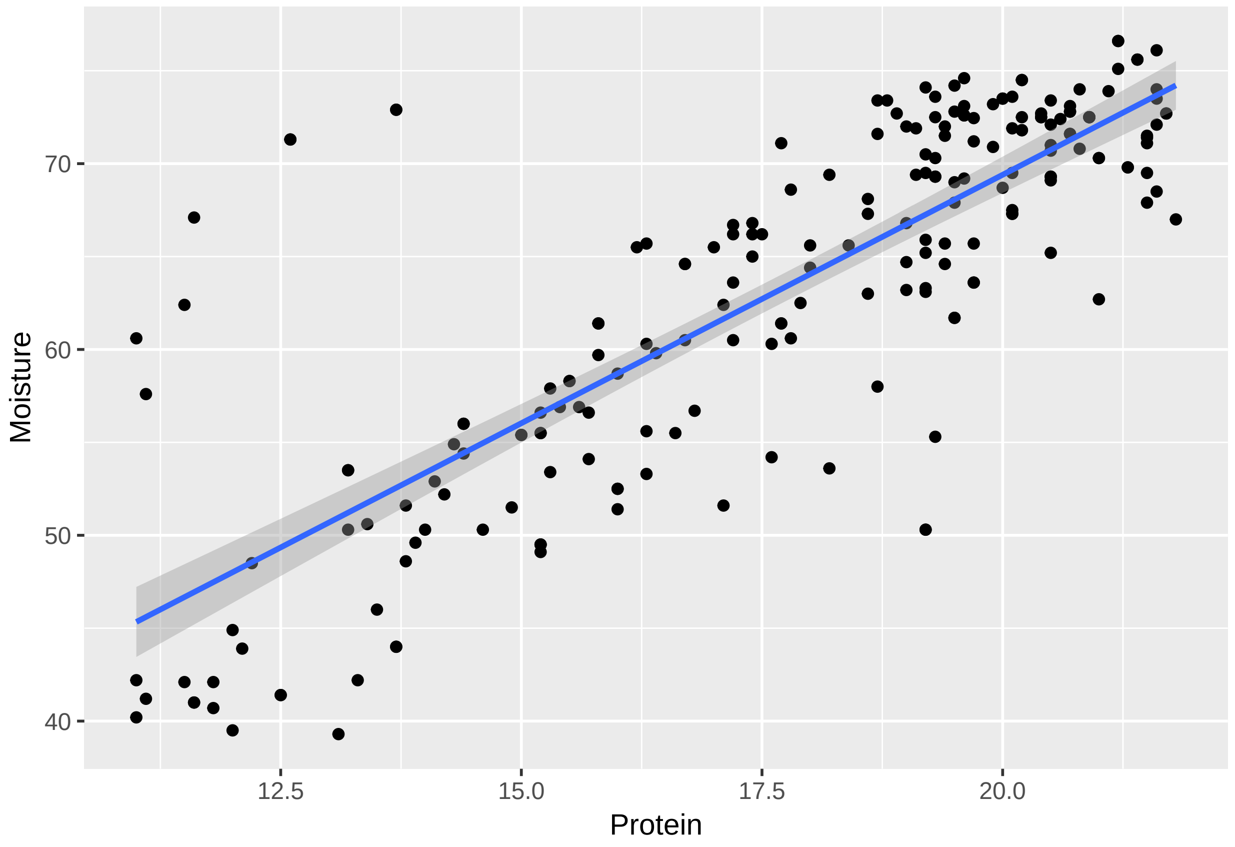
Using the above plot of pair scatterplota we can obatain usefull information aboout the conection between Fertility and the features chosen by the best model.Starting from the connection of Fertility and Agriculture we can see that higher percentage of males involved in agriculture occupation tend to have higher fertility.The connection between Fertility and Education seems negative meaning that higher percentage of education is connected to lower fertility.Moving to the connection of Fertility and Catholic there seems to be 2 clusters that they might be reprecent the diffrence between catholic and protestant fertility.Finally,the connection between Fertility and Infant.Mortality is positive.

In conclusion,the feature chosen as we can see from the scatterplots and the correlations between Fetility have a large impact on explaing Fertility.Moreover the fact that Examination was not chosen by the model might be because there is high correlation between Examination and Education and the effect both on the model is very small.

Assignment 4

from *Zijie Feng*

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

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

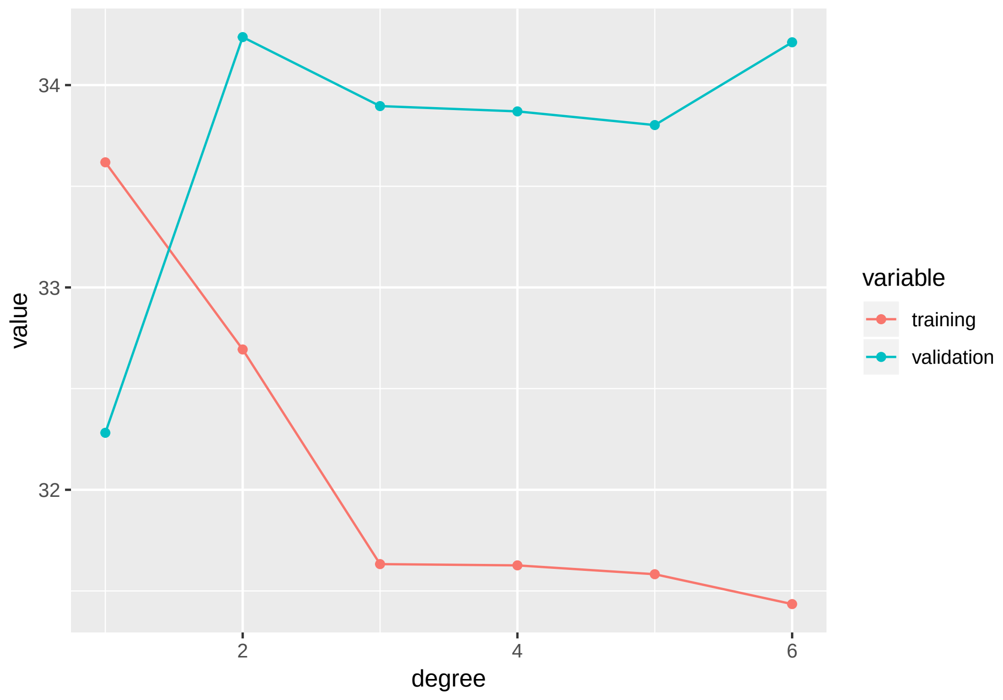
Therefore,

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

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.

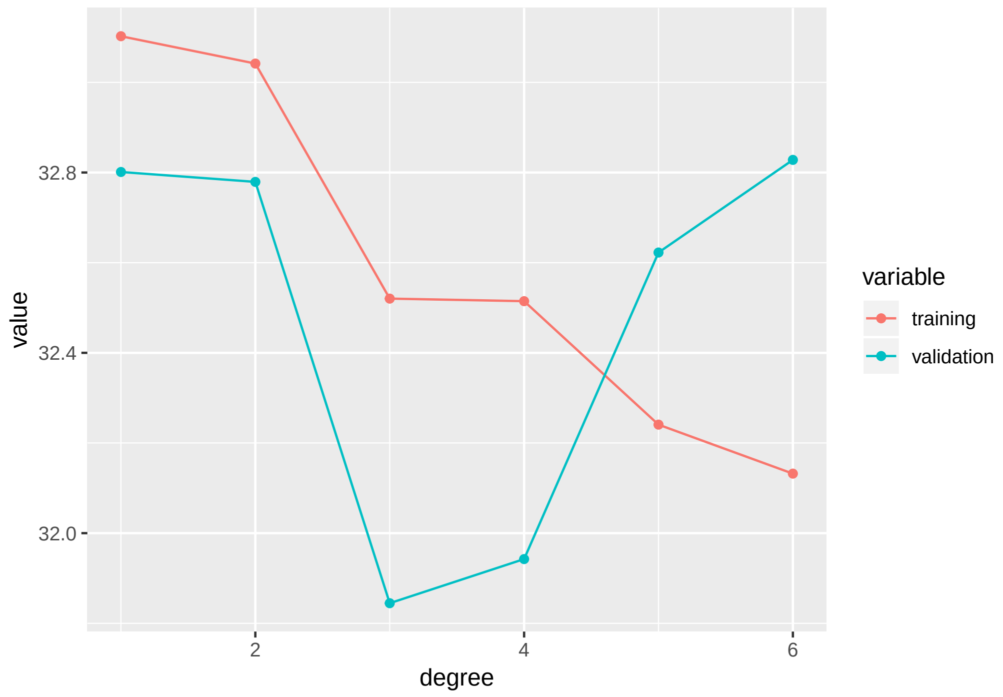
4.3

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

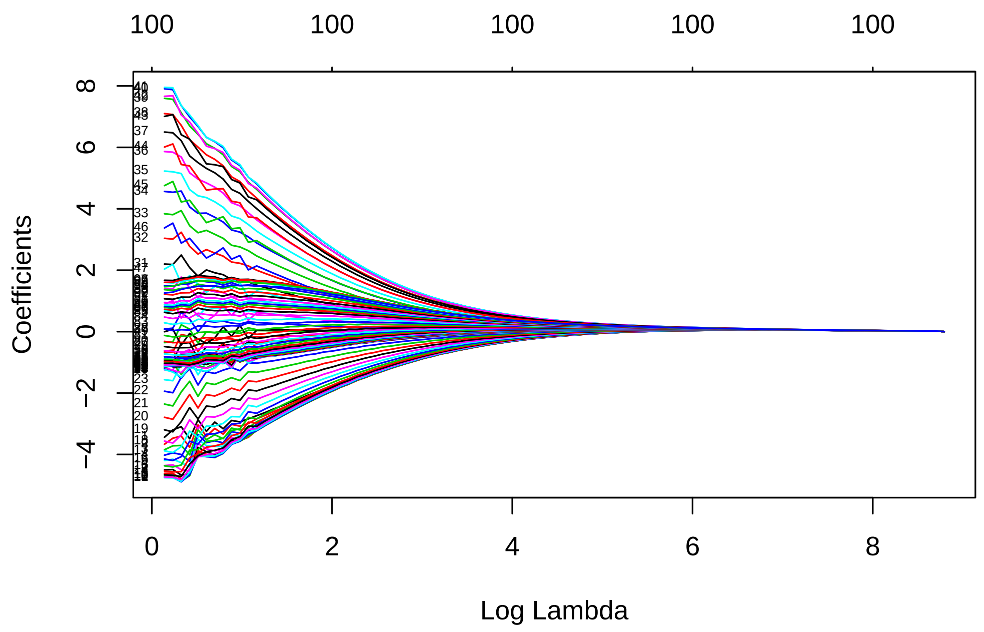
There are 64 variables (63 channels, 1 intercept) selected by `stepAIC` with both directions.

```
## 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
```

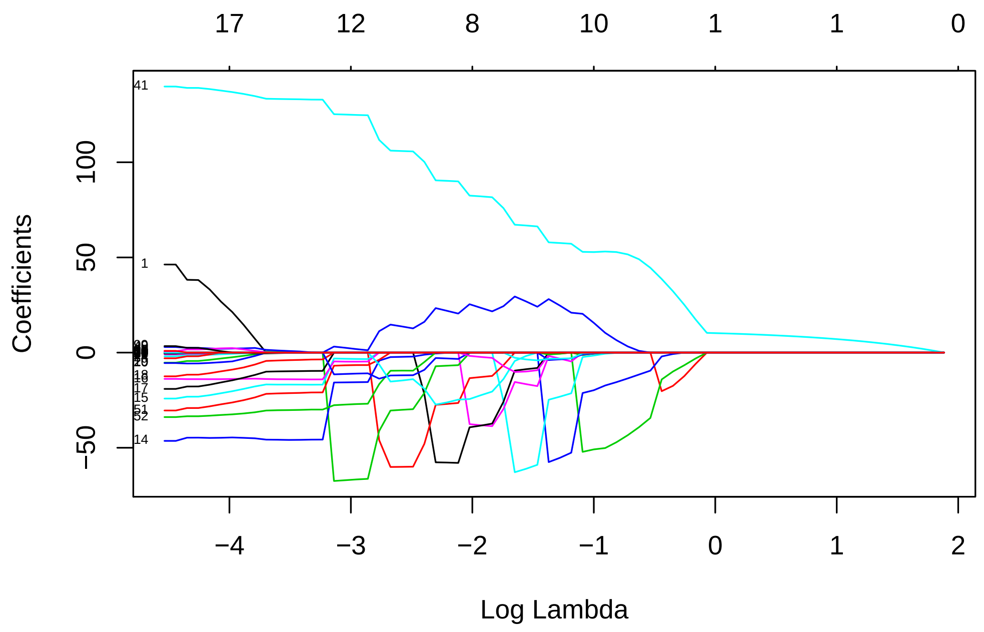
4.5

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.

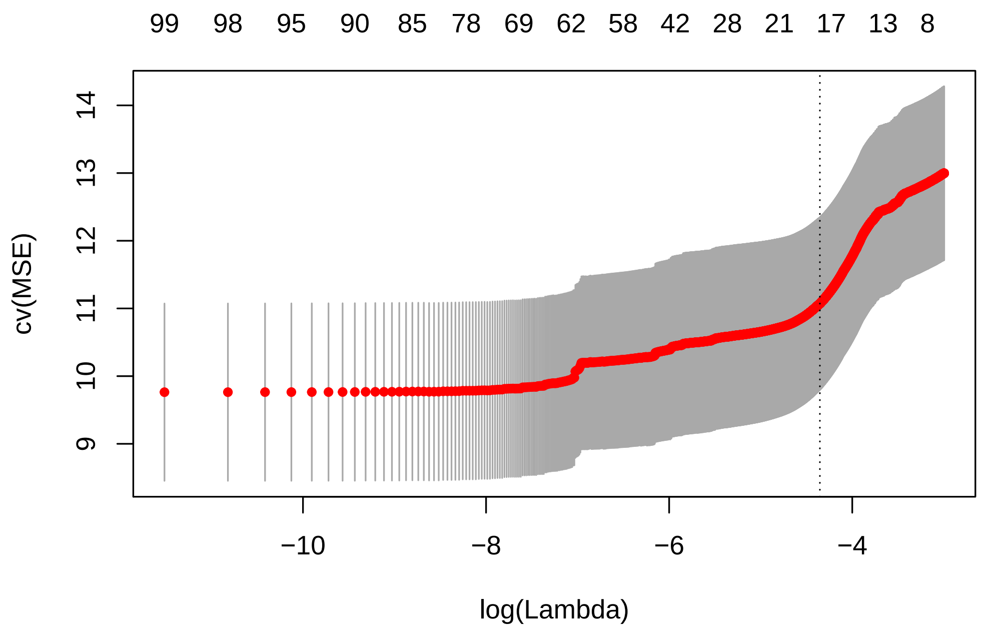


4.7

```
## 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$set(echo = TRUE)
library("readxl")
library(kknn)
library(ggplot2)
library(reshape2)
library(glmnet)
library(MASS)
library(doParallel)
#####Task 1
spambase <- read_xlsx("spambase.xlsx")
n=dim(spambase)[1]
set.seed(12345)
id=sample(1:n, floor(n*0.5))
train=spambase[id,]
test=spambase[-id,]
#####Task 2
####model
trainlm <- glm(Spam ~.,family=binomial(link="logit"), data=train)
####testingdata
testpre <- predict(trainlm, test, type="response")
p1 <- 0.5
testF <- cut(testpre, breaks=c(-Inf, p1, Inf), labels=c(0, 1))
cTabt    <- table(test$Spam, testF, dnn=c("actual", "predicted"))
missratetest1<-1-sum(diag(cTabt))/sum(cTabt)
###output
paste("Confusion matrix for testing data is:")
addmargins(cTabt)
paste("Misclassification rate for testing data is:")
missratetest1
###trainingdata
trainpre <- predict(trainlm, train, type="response")
p1 <- 0.5
trainF <- cut(trainpre, breaks=c(-Inf, p1, Inf), labels=c(0, 1))
cTabn    <- table(train$Spam, trainF, dnn=c("actual", "predicted"))
missratetrain1<-1-sum(diag(cTabn))/sum(cTabn)
####output
paste("Confusion matrix for training data is:")
addmargins(cTabn)
paste("Misclassification rate for training data is:")
missratetrain1
#####Task 3
####Testing
testpre <- predict(trainlm, test, type="response")
p2 <- 0.9
testF2 <- cut(testpre, breaks=c(-Inf, p2, Inf), labels=c(0, 1))
cTabt2 <- table(test$Spam, testF2, dnn=c("actual", "predicted"))
missratetest2 <- 1-sum(diag(cTabt2))/sum(cTabt2)
paste("Confusion matrix for testing data is:")
addmargins(cTabt2)
paste("Misclassification rate for testing data is:")
missratetest2
```

```
#####Training
trainpre <- predict(trainlm, train, type="response")
p2 <- 0.9
trainF2 <- cut(trainpre, breaks=c(-Inf, p2, Inf), labels=c(0, 1))
cTabn2 <- table(train$Spam, trainF2, dnn=c("actual", "predicted"))
missratetrain2 <- 1-sum(diag(cTabn2))/sum(cTabn2)
paste("Confusion matrix for training data is:")
addmargins(cTabn2)
paste("Misclassification rate for training data is:")
missratetrain2
#####Task 4
###Test
k1<-kknn(Spam ~., train, test,k=30)
p1 <- 0.5
testF3 <- cut(k1$fitted.values, breaks=c(-Inf, p1, Inf), labels=c(0, 1))
cTab3   <- table(test$Spam, testF3, dnn=c("actual", "predicted"))
missrate3<-1-sum(diag(cTab3))/sum(cTab3)
paste("Confusion matrix for testing data is:")
addmargins(cTab3)
paste("Misclassification rate for testing data is:")
missrate3

####Train
k1t<-kknn(Spam ~., train, train,k=30)
testF3 <- cut(k1t$fitted.values, breaks=c(-Inf, p1, Inf), labels=c(0, 1))
cTab3   <- table(train$Spam, testF3, dnn=c("actual", "predicted"))
missrate3<-1-sum(diag(cTab3))/sum(cTab3)
paste("Confusion matrix for training data is:")
addmargins(cTab3)
paste("Misclassification rate for training data is:")
missrate3
#####Task 5
####Test
k2<-kknn(Spam ~., train, test,k=1)
p1 <- 0.5
testF4 <- cut(k2$fitted.values, breaks=c(-Inf, p1, Inf), labels=c(0, 1))
cTab4   <- table(test$Spam, testF4, dnn=c("actual", "predicted"))
missrate4<-1-sum(diag(cTab4))/sum(cTab4)
paste("Confusion matrix for testing data is:")
addmargins(cTab4)
paste("Misclassification rate for testing data is:")
missrate4

####Train
k2t<-kknn(Spam ~., train, train,k=1)

testF4 <- cut(k2t$fitted.values, breaks=c(-Inf, p1, Inf), labels=c(0, 1))
cTab4   <- table(train$Spam, testF4, dnn=c("actual", "predicted"))
missrate4<-1-sum(diag(cTab4))/sum(cTab4)
paste("Confusion matrix for training data is:")
addmargins(cTab4)
paste("Misclassification rate for training data is:")
missrate4
```

```r
feature_selection<-function(X,Y,N){

  n<-ncol(X)
  idx<-1:2^n-1
  t<-vector()
  mat<-sapply(idx, function(id){
    t<-cbind(t,as.integer(intToBits(id)))
    t})
  m<-mat[1:n,2:ncol(mat)]
  ######################################
  set.seed(12345)
  #X<-X[sample(nrow(X)),]
  #Y<-Y[sample(length(Y))]
  id<-sample(nrow(X))
  X<-X[id,]
  Y<-Y[id]
  #Create N equally size folds
  folds <- cut(seq(1,nrow(X)),breaks=N,labels=FALSE)
  d<-matrix(0,nrow=N,ncol=dim(m)[2])
  n_features<-rep(0,ncol(d))
  for (i in 1:ncol(m)){
    x<-X[which(m[,i]==1)]
    n_features[i]<-ncol(x)
    for(j in 1:N){

      testIndexes <- which(folds==j,arr.ind=TRUE)
      testX <- as.matrix(x[testIndexes, ])
      trainX <- as.matrix(x[-testIndexes, ])
      testy<-Y[testIndexes]
      trainy<-Y[-testIndexes]
      trainX<-cbind(1,trainX)
      testX<-cbind(1,testX)

      w<-round(as.vector(solve(t(trainX)%*%trainX)%*%t(trainX)%*%trainy),3)
      y_pred<-round(as.matrix(testX%*%w),3)
      sse<-sum((testy-y_pred)^2)
      d[j,i]<-sse
    }
  }
 d<-d
 s<-apply(d, MARGIN = 2, function(x) mean(x, na.rm=TRUE))
  bindex<-which(s==min(s))
  best_comb<-X[which(m[,bindex]==1)]

  plot(x=n_features,y=s,type="p",xlab="number of features",ylab="CV score",col=ifelse(s==s[bindex],"red
  text(x=3.5,y=522.8431,labels=c("best model--->"))
  return(list("best combination"=colnames(best_comb),"best cv score"=s[bindex]))
}
Y<-swiss[,"Fertility"]
X<-swiss[!names(swiss)%in%c("Fertility")]
D<-feature_selection(X,Y,5)
D
panel.cor <- function(x, y, digits = 2, prefix = "", cex.cor, ...)
```

```r
{
    usr <- par("usr"); on.exit(par(usr))
    par(usr = c(0, 1, 0, 1))
    r <-cor(x, y)
    txt <- format(c(r, 0.123456789), digits = digits)[1]
    txt <- paste0(prefix, txt)
    if(missing(cex.cor)) cex.cor <- 0.8/strwidth(txt)
    text(0.5, 0.5, txt, cex = cex.cor * r)
}

pairs(swiss,lower.panel = panel.cor,main="Pair Scatterplot and correletions between swiss dataset")
## 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()
```

```r
##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
```