Machine Learning Lab1 Group Report

Group 10, combined by Jooyoung Lee

Weng Hang Wong, Jooyoung Lee, Fengjuan Chen

2019 11 22

Overall, good work!

Assignment 1 - *Weng Hang Wong*

1. From the spambase.xlsx file, we classfied the regular email and spam email by implementing the prediction with logistic regression. When the probabiliy of the predicted Y is larger than 0.5 , we classified it as spam email, otherwise, it is a regular email.

From the confusion matrices of 50% classfication, we can see the comparison between the prediction result and the actual result. The misclassification rates of the training and test result is 16.28% and 17.73% respectively, which we can see that the acurracy of these two training/test predictions have no big difference and both the predictions seems quite close to the real world result.

```
##                Actual
## Predicted       regular spam
##   train regular     804   93
##   train spam        127  346
```

```
##               Actual
## Predicted        0   1
##   test regular 808  92
##   test spam    143 327
```

Misclassification rates seem to be different in text.

```
## [1] "misclassification of training data: 0.160583941605839"
```

```
## [1] "misclassification of test data: 0.171532846715328"
```

2. We classify the email as spam by changing the probability of the predicted Y is larger than 0.8. Compare the above prediction(50%) and the below prediction(80%), we can easily see that the the about prediction(50%), of which misclassfication on the regular email and spam email are more even, and the missclassfication rate of prediction(50%) is relatively lower that the prediction(80%), which is 25.74% and 27.59% of the training data and test data respectively.

However, the prediction(80%) of the regular emails have lower accurracy compare to the prediction(50%). Since we increased the probability of prediction in $Y > 0.8$, so meanwhile, the prediction has higher tendency on the regular emails than spam email.

```
##                Actual
## Predicted       regular spam
##   train regular     921  333
##   train spam         10  106
```

```
##               Actual
## Predicted      regular spam
##   test regular     931  314
##   test spam         20  105
```

```
## [1] "misclassification of training data: 0.25036496350365"
```

```
## [1] "misclassification of test data: 0.243795620437956"
```

3. Now we use the standard classifier kknn, which use kknn() to find out the k nearest neighbour. When K = 30, we can see from the confusion matrix and also the missclassfication rate indicated that the accuracy of this prediction lower than the prediction of logistic regression using 50%. The misclassfication of this case is 13.87% and 20.07% of training data and test data respectively, which is lower than 16.28% and higher than 17.73% of the logistic regression prediction. In conclusion, the prediction of kknn(k=30) is has a better prediction on training data.

```
##                Actual
## Predicted       regular spam
##   train regular     851   86
##   train spam         80  353
```

```
##               Actual
## Predicted      regular spam
##   test regular     822  127
##   test spam        129  292
```

```
## [1] "misclassification rates of training data: 0.121167883211679"
```

```
## [1] "misclassification rates of test data: 0.186861313868613"
```

4. When K=1, compare to the prediction of K=30, we can see that the misclassfication rate is increased when the K is decrease. The misclassification rate of K=1 is 42.92% and 25.98% on training data and test data respectively. Since the neighbours is equal to 1, in this case, it is very easy to be misclassfied when the volumn of the neighbours is too small.

Numbers in text do not match the ones printed.

```
##                Actual
## Predicted       regular spam
##   train regular     633  298
##   train spam        318  121
```

```
##               Actual
## Predicted      regular spam
##   test regular     756  157
##   test spam        195  262
```

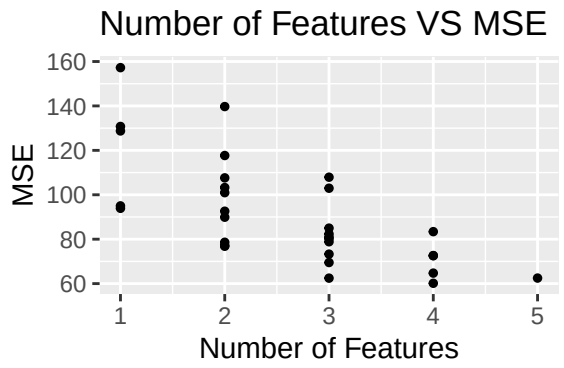
This is definitely wrong!

```
## [1] "misclassfication rate of training data: 0.44963503649635"
```

```
## [1] "misclassfication rate of test data: 0.256934306569343"
```

Check your code, looks like you have used the wrong data for computing the confusion matrix.

Assignment 3 - *Jooyoung Lee*




```
## $CV
## [1] 60.15763
##
## $Features
## [1] 1 3 4 5
```

R-code used for analyzing data *swiss* is based on help file on LISAM, and youtube video [https://www.youtube.com/watch?v=AFg2MvhFeho].

To be specific, it is Cross-Validation MSE

The resulting plot shows how mean square error(MSE) is distributed as number of features changes. The average of MSE shows trend of decreasing as number of features increases; as a number of explanatory variables increases, the average squared difference between predicted value and the actual value decreases.

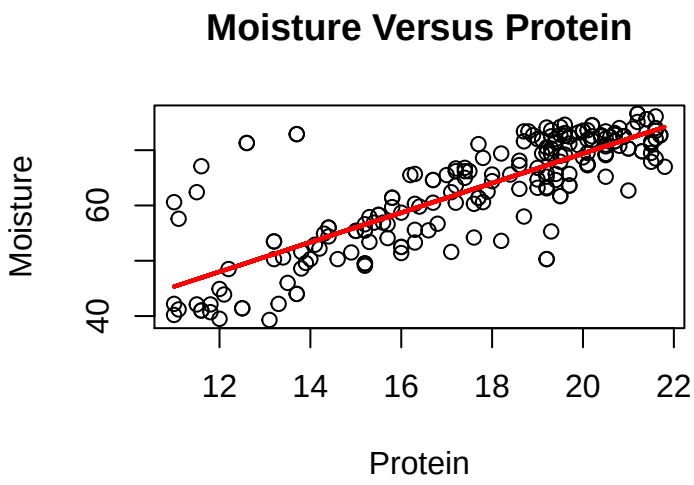
The lowest point of MSE is obtained when the number of features is equal to 4. In the r-code function, code for detecting the positions of features is implemented. According to the result, feature 1, 3, 4 and 5, which are Agriculture, Education, Catholic, Infant.Mortality, are giving the lowest MSE, 60.1576291, when they are used as explanatory variables. To conclude, when Agriculture, Education, Catholic, Infant.Mortality are used as predictors, most accurate prediction on Fertility is carried out.

Standard linear regression function `summary(lm(Fertility~., data=swiss))` can be run for analyzing data *swiss*. Small p-values represent that such feature is statistically significant in explaning the response variable, which is in this case, Fertility. Education has the lowest p-value, which means it has the largest impact on Fertility. Agriculture, Education, Catholic, Infant.Mortality all have low p-values less than 2%, and this result matches to the result turned out after executing 5-fold cross validation.

Good analysis, but would be nice if you actually showed the model summary to support your statements.

Assignment 4 - *Fengjuan Chen*

1. Running linear regression function shows following result.



It seems the linear model well describes the data. Among 215 observations, only six points are seemed to be significant outliers that are deviated from this linear pattern.

2. The probabilistic model M_i can be represented as below:

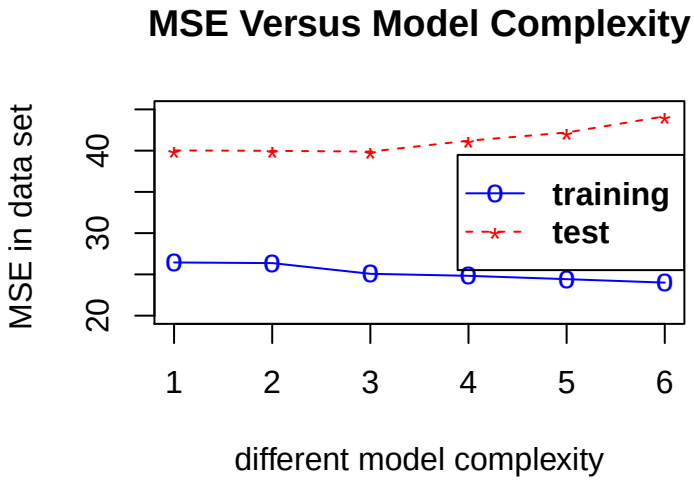
The expression that you have mentioned is not a probabilistic model.

$$M_i = \beta_0 * x^0 + \beta_1 * x^1 + \beta_2 * x^2 + ... + \beta_i * x^i$$

Since this model used for analyzing training data is a linear regression, it is possible to get unbiased estimators for parameters, β_is. This is a linear model, because exponents of parameters are 1. MSE is an appropriate measure to present the test errors, as its expected value is equal to the variance of the response variable.

3. Following plot shows how MSE differs as the complextiy of model increases.

This doesn't make much sense to me. What about the bias?



The plot shows $M3$ is the best model with lowest MSE in test data.The MSE in training data decreses when the model becomes more complex. However, the MSE in test data firstly decreses and then increses when the model becomes more complex. It represents increasing complexity of the model can be lead to overfitting, that it does not describe the test data reasonably.

This is becuase the bias-variance tradeoff. When the complexity of models increses, the variance becomes high, while bias decreases; this overfitting model does not describe the general pattern of the whole data. The opposite situation happens when the model complexity is decreased.

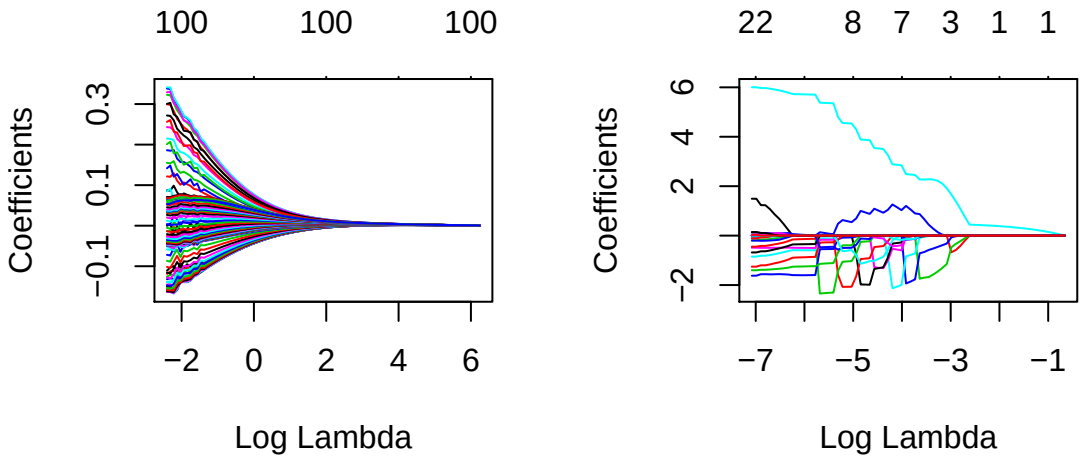
Correct!

4. By using stepAIC, 63 variables are selected to fit the model. The resulting number of coefficients are 64, but one has to be subtracted because on of the coefficients is the intercept.

5. 6. Following plots show how coefficients changes as log lambda value increases.

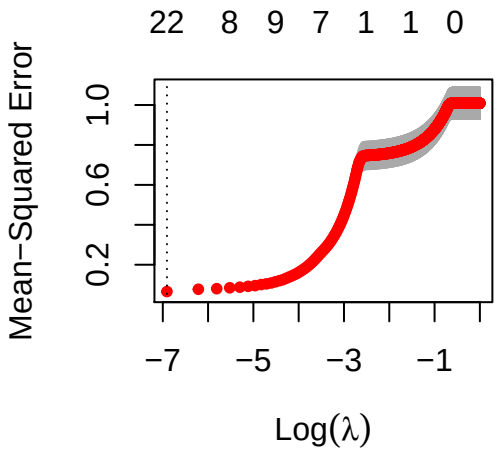
```
## Loading required package: Matrix
```

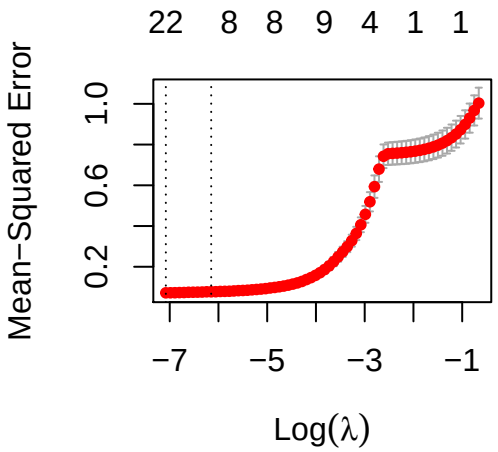
```
## Loaded glmnet 3.0-1
```



As lambda increasing, the coefficients in both ridge and Lasso converge to zero. But the coefficients in ridge regression are going to zero smoothly, while in Lasso they are suddenly down to zero. Therefore, if it is to give heavy penalty on too complex, in other words, overfitting model, Lasso is more effective.

7. With using following plot, optimal model can be found.





Cross-validation method gives the optimal Lasso model with the optimal lambda 0, which is linear regression with all the variables. If lambda is set to be not equal to zero, optimal model will be the one with 23 variables, with lambda equals 0.0008421867.

Any reason why you would do this? And how do you say 23 variables? The plot seems to only show 22.

8. Comparison: The stepAIC method uses 63 variables to fit the data, while ridge and Lasso are functional by using panelty parameter lambda. Consequently, ridge and Lasso will give different results when different lambda is chosen. Operating cross-validation helps to choose optimal lambda in ridge and Lasso regression. Without specifying value of lambda, the same parameter lambda is returned as a result. But if the interval of lambda is given, the values of lambda are dependent to the interval given.

Comparison of steps 4 and 7 can be more detailed.

Appendix

```
#1
library(kknn)
data <- readxl::read_xlsx("C:/Users/Young/Documents/ML/Lab1/spambase.xlsx")
# 1.1

n=dim(data)[1]
set.seed(12345)
id=sample(1:n, floor(n*0.5))
train=data[id,]
test=data[-id,]

# 1.2

# Logistic regression model
D <- glm(Spam~.,data = train, family = "binomial")

# Classcification of train $ test data
train$pred <- predict(D, newdata=train ,type='response')
train$pred_50 <- ifelse(train$pred > 0.50, 1, 0)
```

```r
test$pred <- predict(D, newdata=test, type='response')
test$pred_50 <- ifelse(test$pred > 0.50, 1, 0)

# Misclassification error ~ train data
confusion_train <- table(Predicted = train$pred_50, Actual = train$Spam)
missrate_train <- 1 - sum(diag(confusion_train))/sum(confusion_train)
row.names(confusion_train) <- c("train regular", "train spam ")
colnames(confusion_train) <- c("regular", "spam")

# Misclassification error ~ train data
confusion_test <- table(Predicted = test$pred_50, Actual = test$Spam)
row.names(confusion_test) <- c("test regular","test spam")
colnames(confusion_train) <- c("regular", "spam")
missrate_test <- 1 - sum(diag(confusion_test)/ sum(confusion_test))

print(confusion_train)
print(confusion_test)
print(paste("misclassification of training data:", missrate_train))
print(paste("misclassification of test data:", missrate_test))

# Classcification of train $ test data
train$pred_80 <- ifelse(train$pred > 0.80, 1, 0)
test$pred_80 <- ifelse(test$pred > 0.80, 1, 0)

# Misclassification error ~ train data
confusion_train80 <- table(Predicted = train$pred_80, Actual = train$Spam)
row.names(confusion_train80) <- c("train regular", "train spam ")
colnames(confusion_train80) <- c("regular", "spam")
missrate_train80 <- 1 - sum(diag(confusion_train80))/sum(confusion_train80)

# Misclassification error ~ train data
confusion_test80 <- table(Predicted = test$pred_80, Actual = test$Spam)
missrate_test80 <- 1 - sum(diag(confusion_test80)/ sum(confusion_test80))
row.names(confusion_test80) <- c("test regular", "test spam")
colnames(confusion_test80) <- c("regular", "spam")

print(confusion_train80)
print(confusion_test80)

print(paste("misclassification of training data:", missrate_train80))
print(paste("misclassification of test data:", missrate_test80))

# k nearest test k=30
k_30 <- kknn(formula = Spam~. , train, test, k =30)
k_pred_30 <- fitted(k_30)
k_pred_05 <- ifelse(k_pred_30 > 0.5, 1, 0)
confusion_k_30 <- table(Predicted = k_pred_05 ,Actual = test$Spam)
row.names(confusion_k_30) <- c("test regular", "test spam")
colnames(confusion_k_30) <- c("regular", "spam")
missrate_k_30 <- 1-sum(diag(confusion_k_30)) / sum(confusion_k_30)

# k nearest training k=30
k <- kknn(Spam~., train, train, k=30)
```

```r
k_pred <- fitted(k)
k_05 <- ifelse(k_pred >0.5, 1,0)
confusion_k <- table (Predicted= k_05, Actual = train$Spam)
row.names(confusion_k) <- c("train regular", "train spam")
colnames(confusion_k) <- c("regular", "spam")
missrate_k <- 1-sum(diag(confusion_k)) / sum(confusion_k)
# knn_30 <- train.kknn(Spam~., data=train, kmax=30)
# test$knn_pred <- predict(knn_30, test)

print(confusion_k)
print(confusion_k_30)

print(paste("misclassification rates of training data:", missrate_k))
print(paste("misclassification rates of test data:", missrate_k_30))

# k =1 training
k_train <- kknn(formula = Spam~. , train, train, k =1)
k_pred_train <- fitted(k_train)
k_train_spam <- ifelse(k_pred_train > 0.5, 1, 0)
confusion_k_train <- table(Predicted = k_train_spam ,Actual = test$Spam)
row.names(confusion_k_train) <- c("train regular", "train spam")
colnames(confusion_k_train) <- c("regular", "spam")
missrate_k_train <- 1-sum(diag(confusion_k_train)) / sum(confusion_k_train)

# K= 1 test
k_1 <- kknn(formula = Spam~. , train, test, k =1)
k_pred_1 <- fitted(k_1)
k_pred <- ifelse(k_pred_1 > 0.5, 1, 0)
confusion_k_1 <- table(Predicted = k_pred ,Actual = test$Spam)
row.names(confusion_k_1) <- c("test regular", "test spam")
colnames(confusion_k_1) <- c("regular", "spam")
missrate_k_1 <- 1-sum(diag(confusion_k_1)) / sum(confusion_k_1)


print(confusion_k_train)
print(confusion_k_1)

print(paste("misclassfication rate of training data:",missrate_k_train))
print(paste("misclassfication rate of test data:",missrate_k_1))



# 3

#linear regression
mylin=function(X,Y, Xpred){
  X1 <- cbind(1, X)
  beta <- solve(t(X1)%*%X1) %*% t(X1) %*% Y
  Xpred1=cbind(1,Xpred)
  Res=Xpred1%*%beta
  return(Res)
}
```

```
myCV=function(X,Y,Nfolds){
  n=length(Y)
  p=ncol(X)
  set.seed(12345)
  ind=sample(n,n)
  X1=X[ind,]
  Y1=Y[ind]
  sF=floor(n/Nfolds)
  MSE=numeric(2^p-1)
  Nfeat=numeric(2^p-1)
  Features=list()
  curr=0

  #we assume 5 features.

  for (f1 in 0:1)
    for (f2 in 0:1)
      for(f3 in 0:1)
        for(f4 in 0:1)
          for(f5 in 0:1){
            model= c(f1,f2,f3,f4,f5)

            feat <- which(model==1)

            if (sum(model)==0) next()

            SSE=0
            for (k in 1:Nfolds){

              start <- ((k-1)*sF)+1
              end <- (k*sF)
              if (k==Nfolds) { end <- n}
              testID <- (start:end)
              trainID <- -(start:end)

              Ypred <- mylin(X1[trainID,feat], Y1[trainID], X1[testID,feat])
              Yp <- Y1[testID]
              SSE=SSE+sum((Ypred-Yp)^2)
            }
            curr=curr+1
            MSE[curr]=SSE/n
            Nfeat[curr]=sum(model)
            Features[[curr]]=model

          }

  library(ggplot2)

  plott <- ggplot(data.frame(Nfeat, MSE), aes(x=Nfeat, y=MSE)) +
    geom_point(size=1, colour="black") + ggtitle("Number of Features VS MSE") +
    xlab("Number of Features") + ylab("MSE")

  print(plott)
```

```
  i=which.min(MSE)
  return(list(CV=MSE[i], Features=which(Features[[i]]==1)))
}

myCV(as.matrix(swiss[,2:6]), swiss[[1]], 5)


# 4.
library("readxl")
data <- read_xlsx('C:/Users/Young/Documents/ML/Lab1/tecator.xlsx')

a <- plot(data$Protein, data$Moisture, main="Moisture Versus Protein", xlab="Protein",
     ylab="Moisture")
a <- a+lines(data$Protein, predict(lm(data$Moisture~data$Protein)), col="red", lwd=2)

my_data_2 <- read_xlsx('C:/Users/Young/Documents/ML/Lab1/tecator.xlsx')
my_data_3 <- cbind(my_data_2[,104],my_data_2[,103])
temp <- my_data_2[,103]
my_data_3 <- cbind(my_data_3,temp^2,temp^3,temp^4,temp^5,temp^6)
colnames(my_data_3) <- c("Moisture","Protein1","Protein2","Protein3","Protein4","Protein5","Protein6")
n <- dim(my_data_3)[1]
set.seed(12345)
id <- sample(1:n,floor(n*0.5))
train_data<- my_data_3[id,]
vali_data <- my_data_3[-id,]
num_of_train <- dim(train_data)[1]
num_of_vali <- dim(vali_data)[1]
store_mse <- matrix(0,nrow = 6,ncol = 3)
colnames(store_mse) <- c("i","training data","validation data")
store_mse[,1] <- c(1:6)
lin_mod_1 <- lm(Moisture~Protein1,data = train_data)
store_mse[1,2] <- sum(lin_mod_1$residuals^2)/num_of_train
pred_val_1 <- predict(lin_mod_1, vali_data)
store_mse[1,3] <- sum((vali_data[,1]-pred_val_1)^2)/num_of_vali
lin_mod_2 <- lm(Moisture~Protein1+Protein2,data = train_data)
store_mse[2,2] <- sum(lin_mod_2$residuals^2)/num_of_train
pred_val_2 <- predict(lin_mod_2, vali_data)
store_mse[2,3]<- sum((vali_data[,1]-pred_val_2)^2)/num_of_vali
lin_mod_3 <- lm(Moisture~Protein1+Protein2+Protein3,data = train_data)
store_mse[3,2] <- sum(lin_mod_3$residuals^2)/num_of_train
pred_val_3 <- predict(lin_mod_3, vali_data)
store_mse[3,3]<- sum((vali_data[,1]-pred_val_3)^2)/num_of_vali
lin_mod_4 <- lm(Moisture~Protein1+Protein2+Protein3+Protein4,data = train_data)
store_mse[4,2] <- sum(lin_mod_4$residuals^2)/num_of_train
pred_val_4 <- predict(lin_mod_4, vali_data)
store_mse[4,3]<- sum((vali_data[,1]-pred_val_4)^2)/num_of_vali

lin_mod_5 <- lm(Moisture~Protein1+Protein2+Protein3+Protein4+Protein5,data = train_data)
store_mse[5,2] <- sum(lin_mod_5$residuals^2)/num_of_train
pred_val_5 <- predict(lin_mod_5, vali_data)
store_mse[5,3]<- sum((vali_data[,1]-pred_val_5)^2)/num_of_vali

lin_mod_6 <- lm(Moisture~Protein1+Protein2+Protein3+Protein4+Protein5+Protein6,data = train_data)
```

```r
store_mse[6,2] <- sum(lin_mod_6$residuals^2)/num_of_train
pred_val_6 <- predict(lin_mod_6, vali_data)
store_mse[6,3]<- sum((vali_data[,1]-pred_val_6)^2)/num_of_vali

plot(store_mse[,1],store_mse[,2],xlab = "different model complexity",
      ylab = "MSE in data set ",type = "o",
      pch="o",col="blue",ylim = c(20,45), main="MSE Versus Model Complexity")
points(store_mse[,1],store_mse[,3],col="red", pch="*")
lines(store_mse[,1],store_mse[,3],col="red",lty=2)
legend("right",legend = c("training","test"),col=c("blue","red"),
       lty=c(1,2), pch = c("o","*"),ncol = 1,text.font = 2
       ,box.lty = 1)

library(MASS)
 my_data_4 <- my_data_2[,c(-1,-103,-104)]
 linear_model <- lm(Fat~.,my_data_4)
 variable_selection <-stepAIC(linear_model,direction = "both")
# summary(variable_selection)
#variable_selection$anova
# number and names of chosen variables
names(variable_selection$coefficients)
length(variable_selection$coefficients)

library(glmnet)
my_data_4 <- my_data_2[,c(-1,-103,-104)]
covarites <- scale(my_data_4[,1:100])
covarites <- as.matrix(covarites)
response <- scale(my_data_4[,101])
ridge_model <- glmnet(covarites,response,alpha = 0,family = "gaussian")
plot(ridge_model,xvar="lambda",label=TRUE)
lasso_model <- glmnet(covarites,response,alpha = 1,family = "gaussian")
plot(lasso_model,xvar="lambda",label=TRUE)


cvlasso_model <- cv.glmnet(covarites,response,alpha=1,lambda=seq(0,1,0.001),family="gaussian")

#cvlasso_model$lambda.min
plot(cvlasso_model)
coe_matrix <- coef(cvlasso_model,s="lambda.min")
num_of_variable <- coe_matrix[which(coe_matrix!=0)]
#print(length(num_of_variable))

cvlasso_model <-cv.glmnet(covarites,response,alpha=1,family="gaussian")
plot(cvlasso_model)
```