Machine Learning Assignment 01 - Group A15

Zuxiang Li (zuxli371), Marcos F. Mourao (marfr825), Agustín Valencia (aguva779)

23/11/2019

Good report overall, well done!

Acknowledgements

The assignments were solved mainly under Lab time and study meetings. Although we worked together, every team member has written his own individual report by himself and this document summarize the best of all three individual reports.

First question is based on Marcos' work, third one is based on Zuxiang's report, and fourth one is based on Agustín's solution. After compiling them the report was polished by all team members.

Assignment 1. Spam classification with nearest neighbors

2. Use logistic regression to classify the training and test data by the classification principle $\hat{Y} = 1$ if $p(Y = 1|X) > 0.5$, otherwise $\hat{Y} = 0$ and report the confusion matrices and the misclassification rates for train and test data. Analyze the obtained results.

Confusion matrix for training data:

```
##                Rate    Value
## 1     True positive 73.15011
## 2     True negative 89.63211
## 3    False positive 26.84989
## 4    False negative 10.36789
## 5 Misclassification 16.05839
```

```
##     predicted
## true   0   1
##    0 804 127
##    1  93 346
```

Confusion matrix for testing data:

```
##                Rate    Value
## 1     True positive 69.57447
## 2     True negative 89.77778
## 3    False positive 30.42553
## 4    False negative 10.22222
## 5 Misclassification 17.15328
```

```
##     predicted
## true   0   1
##    0 808 143
##    1  92 327
```

The confusion matrix measure how well our model is performing under its training. This model performed similarly for both training and testing data. Although having similar performance scores, the true positive rate drops around 4% from tranining to testing data. The misclassification rate is somewhat low (~17%), but the false positive rate is high (~30%). This means that real mail is often being flagged as spam. To reduce this, we can increase the classification principle, as we can see below.

3. Use logistic regression to classify the test data by the classification principle $\hat{Y} = 1$ if $p(Y = 1|X) > 0.8$, otherwise $\hat{Y} = 0$

Confusion matrix for training data:

```
##                Rate    Value
## 1     True positive 91.37931
## 2     True negative 73.44498
## 3    False positive  8.62069
## 4    False negative 26.55502
## 5 Misclassification 25.03650
```

```
##     predicted
## true   0   1
##    0 921  10
##    1 333 106
```

Confusion matrix for testing data:

```
##                Rate    Value
## 1     True positive 84.00000
## 2     True negative 74.77912
## 3    False positive 16.00000
## 4    False negative 25.22088
## 5 Misclassification 24.37956
```

```
##     predicted
## true   0   1
##    0 931  20
##    1 314 105
```

Very good analysis!

Even though the misclassification rate went up with the new criteria, it is still preferred over the previous criteria. In this particular classification scenario (spam or not spam), classifying "real mail" as spam is much worse than classifying spam as "real mail". From a user point of view, having some spam in one's inbox is better than missing "real mail" that went to the spam box. Here, the key is analysing the false positive rate, which is much smaller for the new criteria of $p(Y = 1|X) > 0.8$.

4. Use standard kknn() with K = 30 from package *kknn*, report the misclassification rates for the training and test data and compare the results with step 2.

Confusion matrix for training data:

```
##                Rate     Value
## 1     True positive 70.428016
## 2     True negative 91.004673
## 3    False positive 29.571984
## 4    False negative  8.995327
## 5 Misclassification 16.715328
```

```
##     predicted
## true   0   1
##    0 779 152
##    1  77 362
```

Confusion matrix for testing data:

```
##                Rate    Value
## 1     True positive 48.97541
## 2     True negative 79.59184
## 3    False positive 51.02459
```

```
## 4    False negative 20.40816
## 5 Misclassification 31.31387
```

```
##     predicted
## true   0   1
##    0 702 249
##    1 180 239
```

The new model has worse results than the logistic regression with significantly higher misclassification rates for both test and training data. This approach also results in a higher number of false positives and false negatives.

Misclassification rate for training data is not significantly high.

5. Repeat step 4 for K=1 and compare results with step 4. What effects does the decrease of K lead to and why?

Confusion matrix for training data:

```
##                Rate Value
## 1     True positive   100
## 2     True negative   100
## 3    False positive     0
## 4    False negative     0
## 5 Misclassification     0
```

```
##     predicted
## true   0   1
##    0 931   0
##    1   0 439
```

Confusion matrix for testing data:

```
##                Rate    Value
## 1     True positive 43.25323
## 2     True negative 77.68396
## 3    False positive 56.74677
## 4    False negative 22.31604
## 5 Misclassification 35.91241
```

```
##     predicted
## true   0   1
##    0 644 307
##    1 185 234
```

In this case, decreasing K to 1 in our model overfitted the data. That is, the predicted values according to the model are exaclty the training values. When predicting the training data itself, the misclassification rate was 0%, as expected.

Good!

While using the testing data, however, this leads to worse performance: higher misclassification, false positives, and false negatives rates.

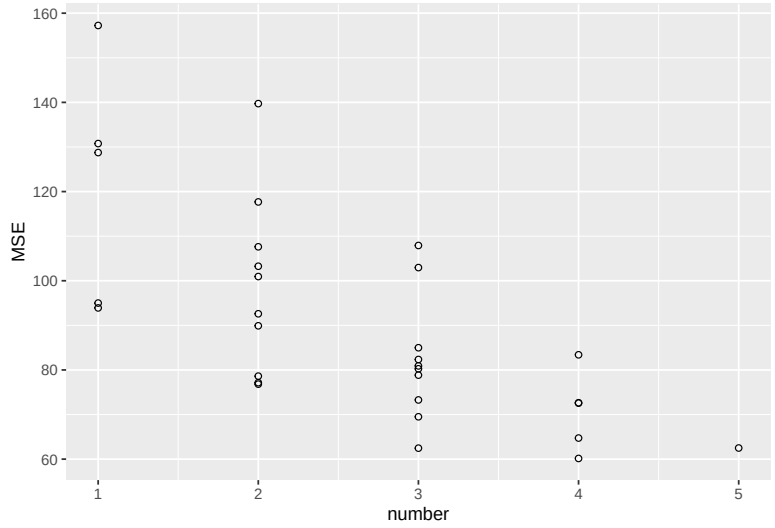
Assignment 3. Feature selection by cross-validation in a linear model.

2. Test your function on data set swiss available in the standard R repository:

- Fertility should be Y
- All other variables should be X
- Nfolds should be 5

Report the resulting plot and interpret it. Report the optimal subset of features and comment whether it is reasonable that these specific features have largest impact on the target.

```
## $CV
## [1] 60.15763
##
## $Features
## [1] 1 0 1 1 1
##
## $plot
```

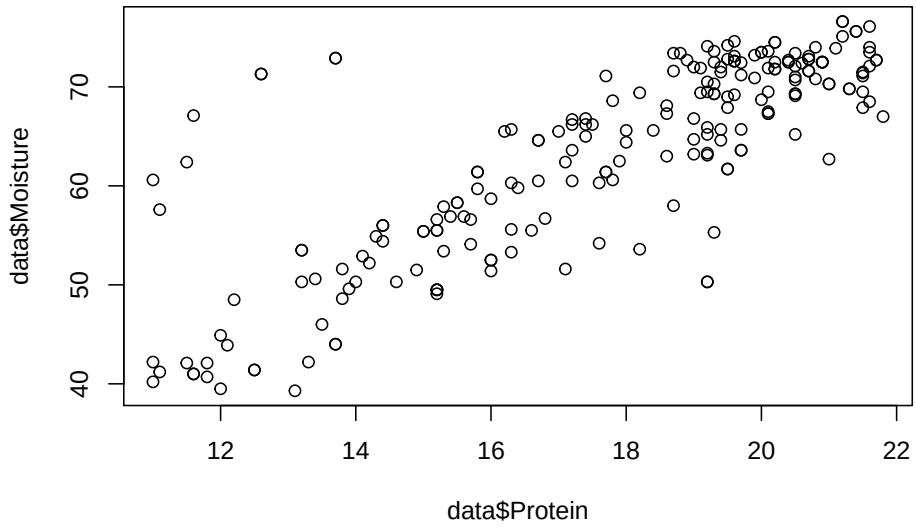


From the MSE plot it can be observed that when the number of features under consideration is increased, the MSE tends to decrease. This indicates that, in this case, using simple models underfits the data, though using too many can also overfit the data and make the MSE scores increase again.

The minimum MSE score is reached at 4 features and its value is 60.15763. The best features subset is $x = (1, 0, 1, 1, 1)$ which comprise `agriculture`, `education`, `religion` and `infant mortality` data. As `examination` feature is dropped it can be intepreted that marks on army examinations do not contribute to explain Fertility changes.

Assignment 4. Linear regression and regularization

1. Import data and create a plot of Moisture versus Protein. Do you think these data are described well by a linear model?



By the plot, although there are some outliers, it seems that the data could be approximated by a linear model.

2. Consider model M_i in which Moisture is normally distributed, and the expected Moisture is a polynomial function of Protein including the polynomial terms up to power i (i.e M_1 is a linear model, M_2 is a quadratic model and so on). Report a probabilistic model that describes M_i. Why is it appropriate to use MSE criterion when fitting this model to a training data?

$$M_i = \sum_{j=0}^{i} \beta_j x^j + \varepsilon$$

$$i = 1, \cdots, 6$$

$$\varepsilon \sim N(\mu, \sigma^2)$$

\mu should be 0 here

With increasing model complexity (higher n polynomial degrees), the model tends to be overfitted, capturing noise from the training data. An overfitted model would produce inferior results (higher variance) when predicting for the test data. The MSE criterion therefore can tell how well fit (or under/overfit) the model is to the data.

But, so can mean absolute error or SSE. Why is MSE better?

3. Divide the data (50/50) and fit models $M_i, i = 1, \cdots, 6$. For each model, record the training and validation MSE and present a plot showing how training and validation MSE depend on i. Which model is best according to this plot? How do MSE values change and why? Interpret this picture in bias-variance tradeoff.

The plot shows a decreasing MSE score for the training data. This is expected because the more complex models fit the training data "better" by capturing its noise. By analysing the MSE score for the training data only, one could erroneously think that the more complex the model, the better. However, this is not always the case, as we can see in the graph.

High complexity models overfit the data and results in high variance and low bias. On the other hand, simplistic models have low variance and high bias. This MSE plot helps the choice of the most balanced model bias-variance wise. In this case, a 3rd degree polynomial model.

4. Perform variable selection of a linear model in which Fat is response and Channel1-Channel100 are predictors by using stepAIC. Comment on how many variables were selected.

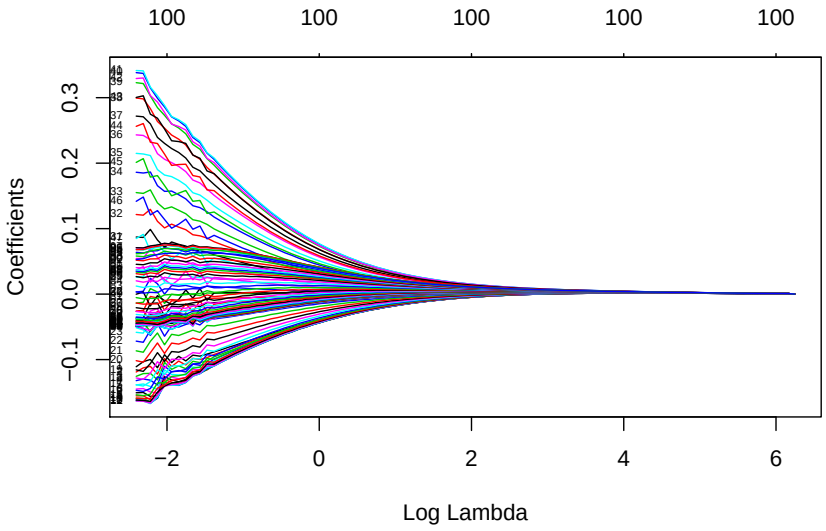
After running stepAIC we get that the amount of selected variables is :

```
## There were selected 64 variables
```

Thus, taking into account that one of them is the intercept, we have 63 selected variables out of 100.

Good catch!

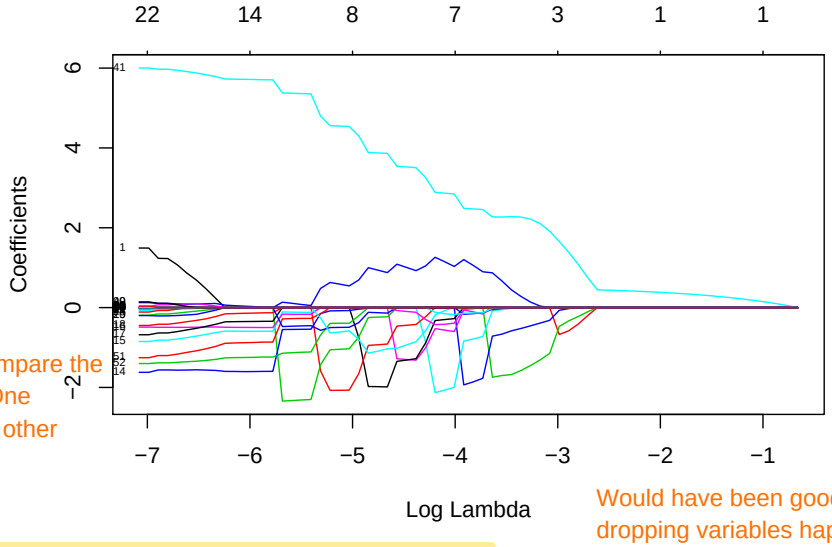
5. Fit a Ridge regression model with the same predictor and response variables. Present a plot showing how model coefficients depend on the log of the penalty factor λ and report how the coefficients change with λ



It can be seen that, when using Ridge regression, for increasing values of λ, the coefficients tend to zero, though the amount of parameters is still 100 since the penalty factor in this regression does not converge to zero, so no actual paramaters are being dropped.

Convergence is not the same as equality.

6. Repeat step 5 but fit with LASSO instead of the Ridge regression and compare the plots from steps 5 and 6. Conclusions?



It doesn't make much sense to compare the lambda values for the 2 models. One penalizes the squared values and other penalizes the absolute values.

Would have been good if you explained why this kind of dropping variables happens.

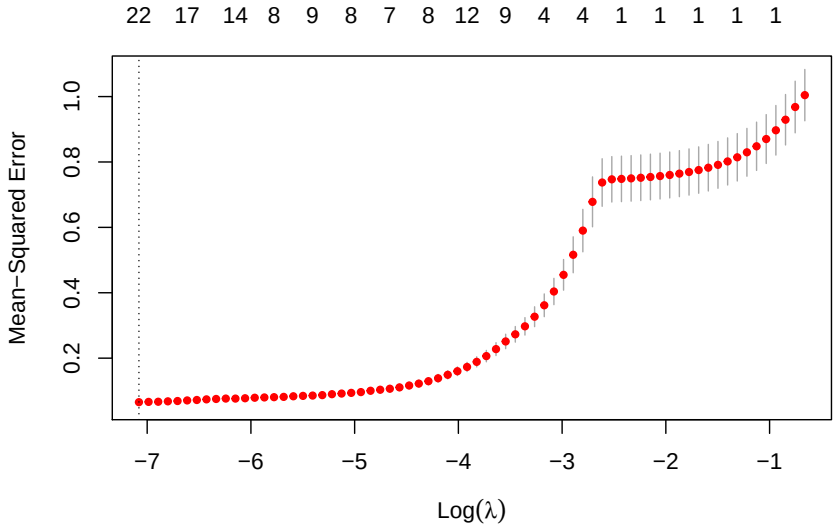
LASSO converges to zero much faster than Ridge regression and it also drops variables.

7. Use cross-validation to find optimal LASSO model (make sure that case $\lambda = 0$ is also considered by the procedure), report the optimal λ and how many variables were chosen by the model and make conclusions. Present also a plot showing the dependence of the CV score and comment how the CV score changes λ

```
## The best performance was at lambda =  0
```

```
##
## Call:  cv.glmnet(x = as.matrix(covariates), y = response, lambda = lambdas,      alpha = 1, family =
```

```
##
## Measure: Mean-Squared Error
##
##        Lambda Measure       SE Nonzero
## min 0.0000000 0.05932 0.006345     100
## 1se 0.0008422 0.06538 0.006221      22
```



From the cross-validated model it is seen that the λ value for the minimum obtained MSE score is at $\lambda_{min} = 0$. The model has 100 non-zero parameters. An increase in $\log(\lambda)$ increases MSE.

You could also consider lambda.1se

8. Compare the results from steps 4 and 7.

For (4) after performing a stepAIC over the model, 36 variables were dropped, getting a final model with only 63 variables (plus the intercept). Nonetheless, for (7) after cross-validating the LASSO model it has been found that the best performance regarding MSE scores is at $\lambda = 0$ which implies no penalization, thus, no parameters will be dropped.

Appendix A - Code chunks

Code used for Assignment 1

1.1

```
#import and read data
data <- read.xlsx("./data/spambase.xlsx")
#dividing data
n=dim(data)[1]
set.seed(12345)
id=sample(1:n, floor(n*0.5))
#train
train=data[id,]
#test
test=data[-id,]

confusion <- function(data, criteria, pred) {
    #criteria
    predicted <- as.integer(pred > criteria)

    #confusion matrix
    true <- data$Spam
    conf <- table(true, predicted)

    #misclassification rate
    total <- sum(conf)
    tp <- conf[2,2]
    tn <- conf[1,1]
    fp <- conf[1,2]
    fn <- conf[2,1]
    miss <- 100 * (fp + fn) / total

    #other rates
    true_positive <- 100 * tp/(tp + fp)
    true_negative <- 100 * tn/(tn + fn)
    false_positive <- 100 * fp/(tp + fp)
    false_negative <- 100 * fn/(tn + fn)

    df <- data.frame(Rate = c("True positive", "True negative", "False positive", "False negative", "Mis
                     Value = c(true_positive, true_negative, false_positive, false_negative, miss))

    print(df)
    return(conf)
}
```

1.2

```
#model
form <- Spam ~ .
model <- glm(formula = form, data = train, family = "binomial")
#train
pred <- predict(model, train, type = "response")
confusion(data = train, criteria = 0.5, pred = pred)
```

```
#test
pred <- predict(model, test, type = "response")
confusion(data = test, criteria = 0.5, pred = pred)
```

1.3

```
#train
pred <- predict(model, train, type = "response")
confusion(data = train, criteria = 0.8, pred = pred)
#test
pred <- predict(model, test, type = "response")
confusion(data = test, criteria = 0.8, pred = pred)
```

1.4

```
model <- train.kknn(formula = form, data = train, ks = 30)


#train
pred <- predict(model, train)
confusion(data = train, criteria = 0.5, pred = pred)
#test
pred <- predict(model, test)
confusion(data = test, criteria = 0.5, pred = pred)
```

1.5

```
model <- train.kknn(formula = form, data = train, ks = 1)

#train
pred <- predict(model, train)
confusion(data = train, criteria = 0.5, pred = pred)
#test
pred <- predict(model, test)
confusion(data = test, criteria = 0.5, pred = pred)
```

Code used for Assignment 3

```
#linear regression
mylin=function(X,Y, Xpred){
  X1=cbind(1,X)
  beta=solve(t(X1)%*%X1)%*%t(X1)%*%Y
  Xpred1=cbind(1,Xpred)
 #MISSING: check formulas for linear regression and compute beta
  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)
          if (sum(model)==0) next()
          SSE=0

          for (k in 1:Nfolds){
            #MISSING: compute which indices should belong to current fold
            current_feat=which(model==1)
            #MISSING: implement cross-validation for model with features in "model"
            #and iteration i.
            begin_pos=(k-1)*sF+1
            if(k==Nfolds){
              end_pos=length(Y1)
            }else{
              end_pos=k*sF
            }

            test_X=X1[begin_pos:end_pos,current_feat]

            train_X=X1[-begin_pos:-end_pos,current_feat]

            train_Y=Y1[-begin_pos:-end_pos]

            Ypred=mylin(train_X,train_Y,test_X)

            #MISSING: Get the predicted values for fold 'k', Ypred, and the original
            #values for folf 'k', Yp.
            Yp=Y1[begin_pos:end_pos]
            SSE=SSE+sum((Ypred-Yp)^2)
          }
          curr=curr+1
          MSE[curr]=SSE/n
          Nfeat[curr]=sum(model)
          Features[[curr]]=model
        }
#MISSING: plot MSE against number of features
library(ggplot2)
df<-data.frame(number=c(),MSE=c())
```

```
  for(i in 1:length(Features)){
    tmp=data.frame(number=sum(Features[[i]]),MSE=MSE[i])
    df=rbind(df,tmp)
  }
  plot1<-ggplot(df,aes(x=number,y=MSE))+geom_point(shape=21)
  #return(plot1)
  i=which.min(MSE)
  return(list(CV=MSE[i], Features=Features[[i]],plot=plot1))
}

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

Code used for Assignment 4

4.1

```
## Import data and plot Moisture vs Protein.
data <- read.xlsx("data/tecator.xlsx")
plot(data$Protein, data$Moisture)
```

4.3

```
# Creating data sets
p <- data$Protein
y <- data$Moisture
P <- data.frame(
    Y = y,
    P1 = p,
    P2 = p^2,
    P3 = p^3,
    P4 = p^4,
    P5 = p^5,
    P6 = p^6
)

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

# Training models
M1 <- lm(Y ~ ., data = train[,1:2])
M2 <- lm(Y ~ ., data = train[,1:3])
M3 <- lm(Y ~ ., data = train[,1:4])
M4 <- lm(Y ~ ., data = train[,1:5])
M5 <- lm(Y ~ ., data = train[,1:6])
M6 <- lm(Y ~ ., data = train[,1:7])

## Train Scores
eval_model <- function(model, data) {
  pred <- predict(model, data)
  errors <- pred - data$Y
```

```
  MSE <- mean(errors^2)
  return(MSE)
}

MSE_train <- c()
MSE_train[1] <- eval_model(M1, train)
MSE_train[2] <- eval_model(M2, train)
MSE_train[3] <- eval_model(M3, train)
MSE_train[4] <- eval_model(M4, train)
MSE_train[5] <- eval_model(M5, train)
MSE_train[6] <- eval_model(M6, train)

MSE_test <- c()
MSE_test[1] <- eval_model(M1, test)
MSE_test[2] <- eval_model(M2, test)
MSE_test[3] <- eval_model(M3, test)
MSE_test[4] <- eval_model(M4, test)
MSE_test[5] <- eval_model(M5, test)
MSE_test[6] <- eval_model(M6, test)

df <- data.frame(
    degree <- c(1:6),
    MSE_train,
    MSE_test
)
p <- ggplot()
p <- p +geom_point(data = df, aes( x = degree, y = MSE_train, color="Train")) +
    geom_line(data = df, aes( x = degree, y = MSE_train, color="Train"))
p <- p + geom_point(data = df, aes( x = degree, y = MSE_test, color="Test")) +
    geom_line(data = df, aes( x = degree, y = MSE_test, color="Test"))
p <- p + labs(y="MSE", colour="Dataset", title = "Mean Square Errors") +
      geom_line()
p
```

4.4

```
adhok_data <- data[,2:101]
Fat <- data$Fat
adhok_data <- cbind(adhok_data, Fat)
model <- lm(Fat ~ . , data=adhok_data)
step <- stepAIC(model, direction ="both")
selected_vars <- step$coefficients
cat("There were selected", length(selected_vars), "variables\n")
```

4.5

```
covariates <- scale(adhok_data[1:(length(adhok_data)-1)])
response <- scale(adhok_data$Fat)
model_ridge <- glmnet(covariates, response, alpha = 0, family = "gaussian")
plot(model_ridge, xvar="lambda", label=T)
```

4.6

```
model_lasso <- glmnet(covariates, response, alpha = 1, family = "gaussian")
plot(model_lasso, xvar="lambda", label=T)
```

4.7

```
lambdas <- append(model_lasso$lambda,0)
cv_model_lasso <- cv.glmnet(as.matrix(covariates), response, alpha=1,
                            family="gaussian", lambda=lambdas)
cat("The best performance was at lambda = ", cv_model_lasso$lambda.min, "\n")
cv_model_lasso
plot(cv_model_lasso)
```