Computer lab 1 block 1: correction

Laura Julià Melis

12/20/2019

CORRECTION:

Task:"Correct mistakes from the group report into your individual report and highlight the corrected text. If some parts that are incorrect in the group report are already correct in your individual report, highlight the correct text and add some extra text that explains why the group solution was wrong."

Legend of colors:

- Corrected text: red.
- Already correct things in my individual report but not in the group report: blue.
- Correct things in the group report but not in my individual report: green.

Assignment 1. Spam classification with nearest neighbors.

The data file spambase.xlsx contains information about the frequency of various words, characters etc for a total of 2740 e-mails. Furthermore, these e-mails have been manually classified as spams (spam = 1) or regular e-mails (spam = 0)

1. Import the data into R and divide it into training and test sets (50%/50%) by using the following code:

To read xlsx data, we call read_xlsx function from readxl package. Then we divide the data into training and testing data.

See Appendix.

2. Use logistic regression (functions glm(), predict()) to classify the classification principle

$$\hat{Y} = 1 \text{ if } p(Y = 1|X) > 0.5, \text{ otherwise } \hat{Y} = 0$$

and report the confusion matrices (use table()) and the misclassification rates for training and test data. Analyse the obtained results.

The predictions have been corrected by changing the argument "data" to "newdata"

- Confusion matrices.

```
##    fitted_response_test
##       0   1
##   0 808 143
##   1  92 327
```

```
##    fitted_response_train
##       0   1
##   0 804 127
##   1  93 346
```

(In the table() function, the first argument is for the rows and the second argument is for the columns. So, in order to use the normal convention for confusion matrices, in this corrected report, all the confusion matrices will be created with true labels on the rows and predicted labels on the columns)

It can be observed that the number of observations classified correctly (number of cases in the diagonal of the matrix) is similar in both confussion matrices but we can notice a little difference in favour of the train confussion matrix: the number of non-spam mails (0) classified as spam (1) is a bit greater in the test matrix.

- Misclassification rates.

For the test data:

```
## [1] 0.1715328
```

For the training data:

```
## [1] 0.1605839
```

The misclassification rate for the test set is a bit greater than the one for the training set.

3. Use logistic regression to classify the test data by the classification principle

$$\hat{Y} = 1 \text{ if } p(Y = 1|X) > 0.8, \text{ otherwise } \hat{Y} = 0$$

and report the confusion matrices (use table()) and the misclassification rates for training and test data. Compare the results. What effect did the new rule have?

- Confusion matrices.

```
##    fitted_response_test
##       0   1
##   0 931  20
##   1 314 105
```

```
##    fitted_response_train
##       0   1
##   0 921  10
##   1 333 106
```

In this case, where a probability of 0.8 has been used as the classification principle, the number of observations in the diagonal (well classified) is greater than before for the both data sets (the matrices are corrected now).

- Misclassification rates.

For the test data:

```
## [1] 0.2437956
```

For the training data:

```
## [1] 0.250365
```

Both rates are corrected as well.

In this occasion, both missclassification rates are greater than before but it's worth mentioning that the number of cases being non-spam that would be classified as spam has decreased a lot so this principle is better than the one in the second question.

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

- Misclassification rate for the test data:

```
## [1] 0.3131387
```

- Misclassification rate for the training data:

```
## [1] 0.1671533
```

If we compare these misclassification rates with the ones obtained in question 2 we can observe that the rate values for the training set are almost the same (0.161 and 0.167) while the rate for the test set is around 0.14 points bigger when using 30-nn model than the linear regression model. So, the results obtained with this new model are worse.

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

- Misclassification rate for the test data:

```
## [1] 0.3591241
```

- Misclassification rate for the training data:

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

When choosing K=1, what we are doing is choosing only one point (the closest one), so when making predictions the model will always classify the observation to that point (because it will have probability 1). For this reason, the missclassification rate for the training set is 0 but the rate for the test set is greater than the one using K=4. Basically, the model is over-fitting very much. (In the group report we didn't mention that the model was overfitting because we started doubting about if we could use the term when we don't have a regression model)

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

1. Implement an R function that performs feature selection (best subset selection) in linear regression by using k-fold cross-validation without using any specialized function like lm() (use only basic R functions). Your function should depend on:

```
X: matrix containing X measurements.
Y: vector containing Y measurements
Nfolds: number of folds in the cross-validation.
```

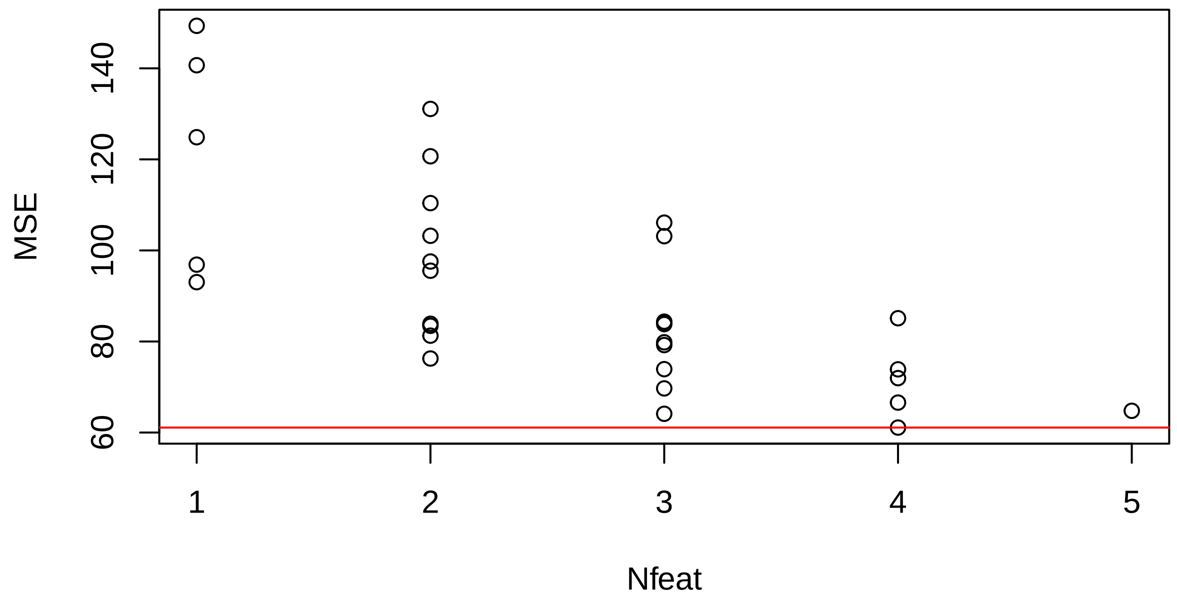
You may assume in your code that matrix X has 5 columns. The function should plot the CV scores computed for various feature subsets against the number of features, and it should also return the optimal subset of features and the corresponding cross-validation (CV) score. Before splitting into folds, the data should be permuted, and the seed 12345 should be used for that purpose.

See Appendix.

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] 61.08948
##
## $Features
## [1] 1 0 1 1 1
```

We were not using the randomized order of observations in your code so now, inside the loop of Nfolds instead of using X and Y, I'm usinf $X1$ and $Y1$. Then, the results are a bit different:

In the plot we can observe the Mean Squared Errors associated to all the different possibilities of fitting a model. The red line indicates the minimum MSE so we can see that the number of features included in the fitted model with the lowest MSE is 4.

Also, from the list output we can observe that the optimal subset of features is (1 0 1 1 1) so we can write the optimal model as:

$$\text{Fertility} = \beta_0 + \beta_1 \cdot \text{Agriculture} + \beta_2 \cdot \text{Education} + \beta_3 \cdot \text{Catholic} + \beta_4 \cdot \text{Infant.Mortality}$$

Given that

```
Fertility:          Ig, 'common standardized fertility measure'
Agriculture:         % of males involved in agriculture as occupation
Examination:         % draftees receiving highest mark on army examination
Education:          % education beyond primary school for draftees.
Catholic:            % 'catholic' (as opposed to 'protestant').
Infant.Mortality:    live births who live less than 1 year.
```

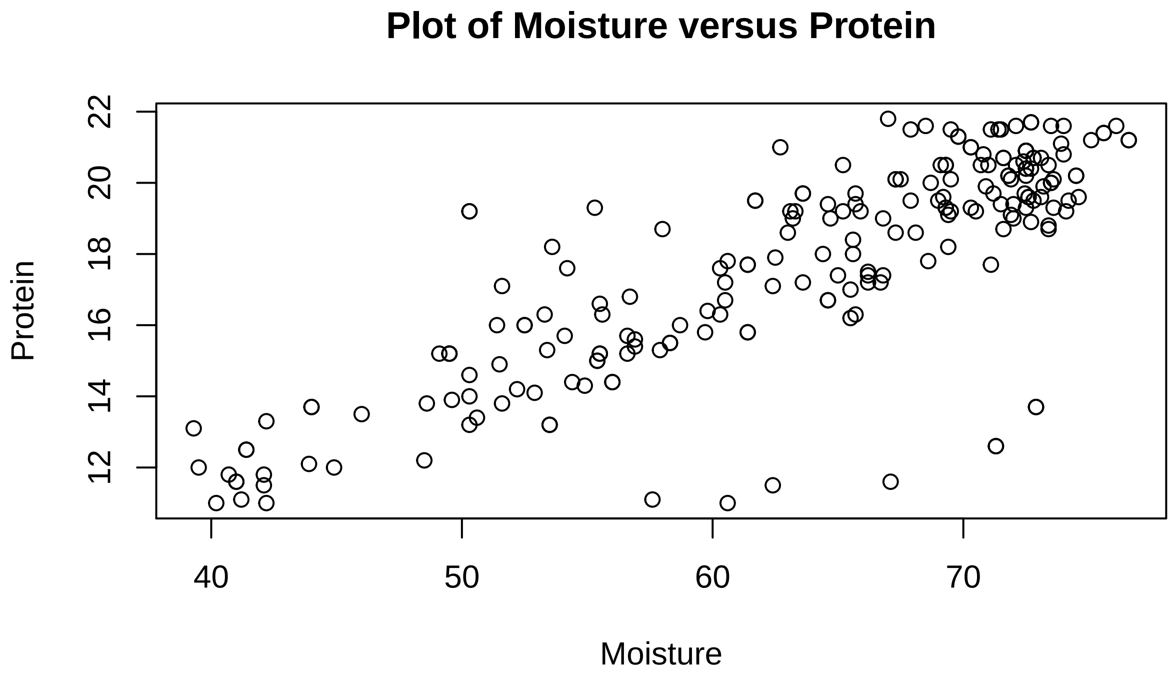
It is reasonable that Agriculture, Education, Religion (being catholic) or Infant mortality are variables that have a largest impact on fertility than having a good grade in the army exam.

Assignment 4. Linear regression and regularization.

The Excel file tecator.xlsx contains the results of study aimed to investigate whether a near infrared absorbance spectrum can be used to predict the fat content of samples of meat. For each meat sample the data consists of a 100 channel spectrum of absorbance records and the levels of moisture (water), fat and protein. The absorbance is -log10 of the transmittance

measured by the spectrometer. The moisture, fat and protein are determined by analytic chemistry.

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



As can be seen from the plot, for most of the points, as Protein values increase the Moisture values do too. And this increasing relationship is linear, with some outliers though in the down right side, so a linear model whould probably fit the data but not so well (I didn't mention in my individual report that we can see some outliers so that the linear model could not be the best one but we mentioned it in the group report).

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?

A probabilistic model that describes M_i would the following:

$$\text{Moisture}_i = \beta_0 + \beta_1 \text{Protein}^1 + \cdots + \beta_i \text{Protein}^i + \epsilon_i$$

with $\epsilon_i \sim N(\mu, \sigma^2)$ (We didn't mention in the group nor in my individual group the distibution and its parameters).

Also, the probabilistic model can be written as:

$$\text{Moisture} \sim N(\beta_0 + \beta_1 \text{Protein}^1 + \cdots + \beta_i \text{Protein}^i, \sigma^2)$$

The mean squared error indicates how close a fitted regression model is to the data by taking the distances from the points (Y_i) to the fitted points $(\hat{Y}_i)$ and squareing them:

$$MSE = \frac{1}{n} \sum_{i=1}^{n} (Y_i - \hat{Y}_i)^2$$

But with a polynomial function of the independent variable, the model can get very complex and overfit (the model would be too close to the training points because it would also capture the noise) meaning that the distance between the test data points and the predicted points would increase (and that would be bad, we don't want that).

Also, for a normally distributed model minimizing MSE results in the Maximum Likelihood Estimation, and this is the reason for why the MSE is appropiate in this case.

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

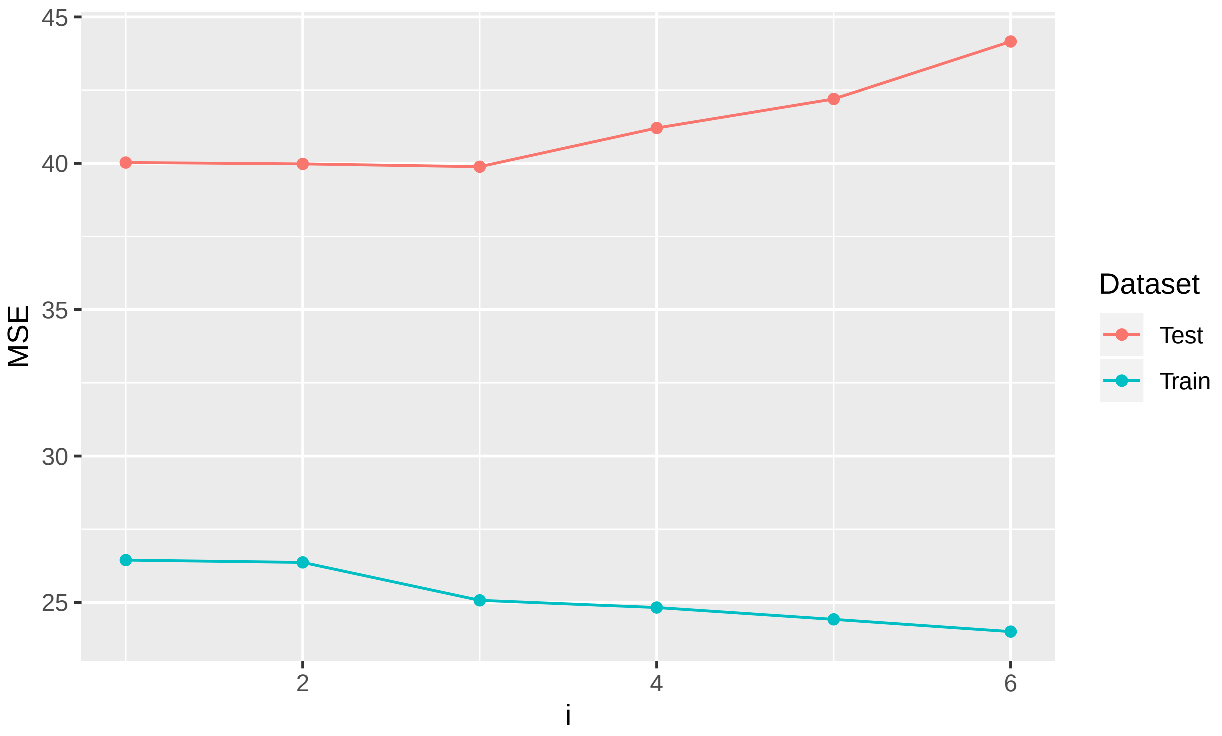
Instead of using the poly() function, I've created a new data set in which the independent variables are $\text{Protein}, \text{Protein}^1, \ldots, \text{Protein}^6$

- MSE table for the training and the test datasets.

```
##   i mse_test mse_train
## 1 1 40.02562  26.44494
## 2 2 39.97895  26.36590
## 3 3 39.88347  25.07017
## 4 4 41.20548  24.82475
## 5 5 42.19681  24.41987
## 6 6 44.16041  24.00218
```

- Plot:

For a better comparison, the MSE's are now graphed in the same plot:



According to the plots, the best model is the one with $i = 3$ (third degree polynomial):

$$\text{Moisture} = \beta_0 + \beta_1 Protein + \beta_2 Protein^2 + \beta_3 Protein^3$$

We see this from the *Test dataset* plot: the lowest MSE (39.88347) is the one we get when $i = 3$.

In order to interpret this plot, first we will look at the Training dataset line , where it can be seen that as the complexity increases the MSE decreases. In other words, the larger i, the closer to the training data points the model fits and the smaller the mean squared error. But if we look at the Test dataset line , we can observe that from i = 1 to i = 3 the MSE decreases a little bit (the increase in the complexity of the model is good because the predictions obtain are better every time) but for values of i higher than 3, the MSE increases meaning that the model might be too complex and that it overfits.

In terms of bias-variance tradeoff, we can interpret the combination of both plots by observing that when the model is less complex (when i is small), the bias (expected value of the difference between the fitted and the real values) is high and the variance is "low" while when i is greater, the variance is higher and the bias lower. So the optimal model complexity is the one obtained by reducing variance at the cost of introducing some bias, and this happens when i = 3.

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.

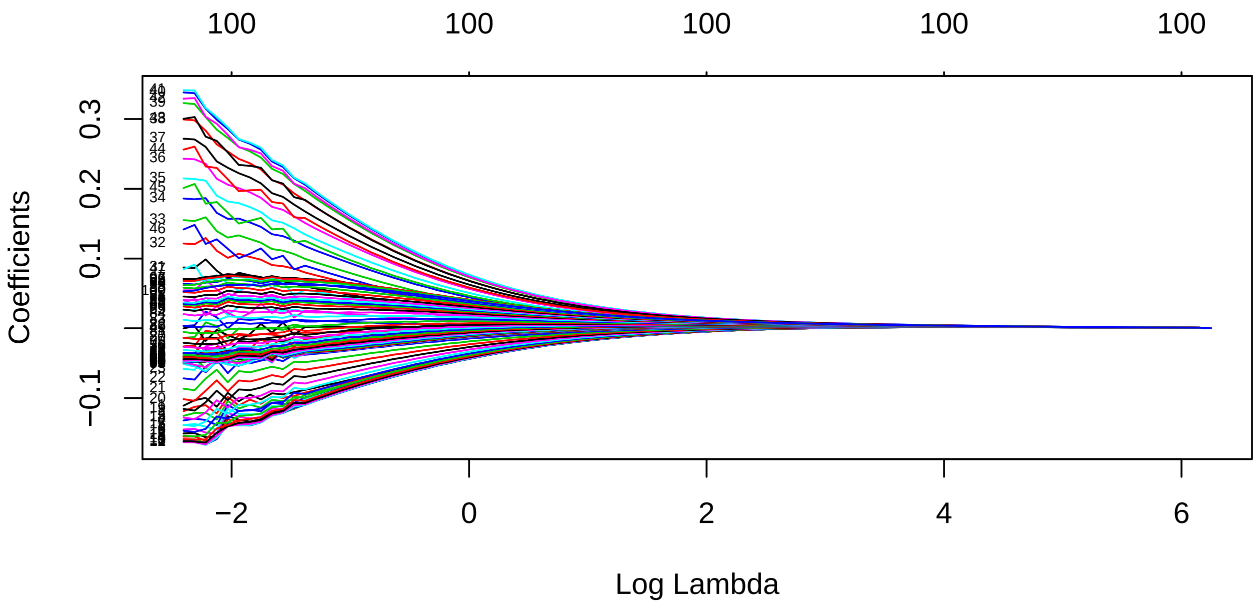
We used the *direction* argument to indicate that the we want the stepwise method to combine backward and forward selection.

```
## [1] 64
```

From `length(fit$coefficients)` we can observe that $64 - 1 = 63$ variables have been selected and the model.

In my individual report I calculated the AIC value for the model but it is not necessary so in the group report we didn't included it, so I have removed the line "and the model has an AIC value of 707.6913".

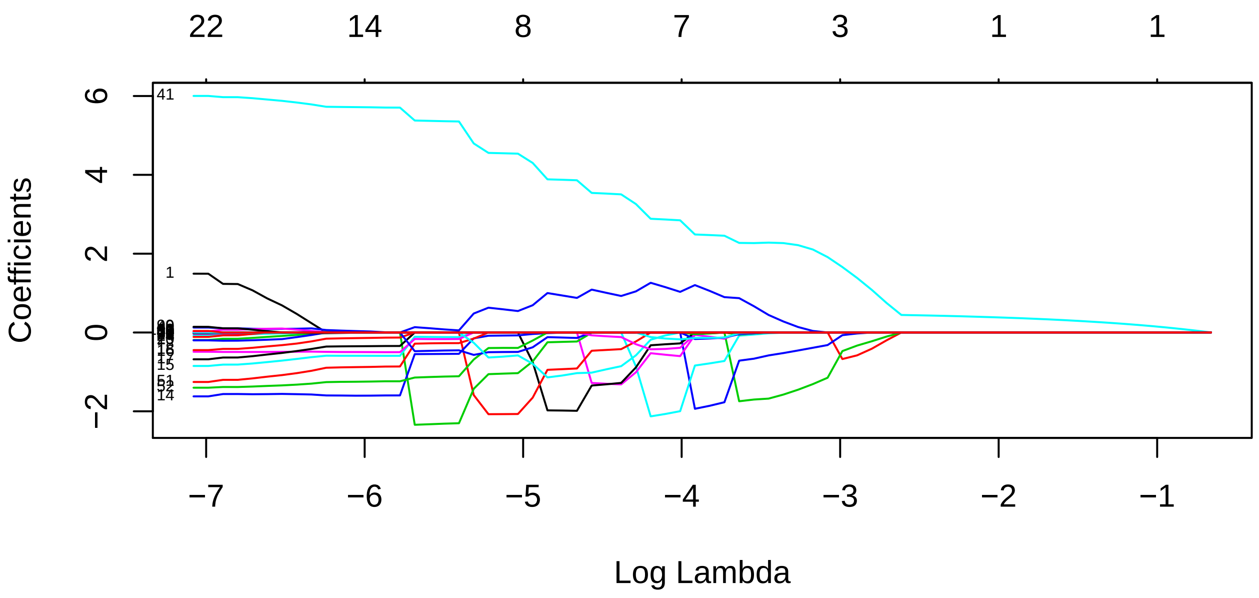
5. Fit a Ridge regression model with the same predictor and response variables. Present a plot showing how model coefficients depend on the log of the penalty factor λ and report how he coefficients change with λ.



Lambda is a tuning parameter that tries to penalize the coefficients in order to get only the important variables and not overfit. Thus, it shrinks coefficients towards zero for non-important variables.

In the plot we see different combinations of lambdas on 100 variables. Ridge regression does not remove any variables, it just shrinks their coefficients. We see that as the lambda increases, all the coefficients converge to zero and that's why we don't choose very big lambdas because then we will have underfit and the coefficients will be smaller than they ought to be to get the best predictive accuracy.

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

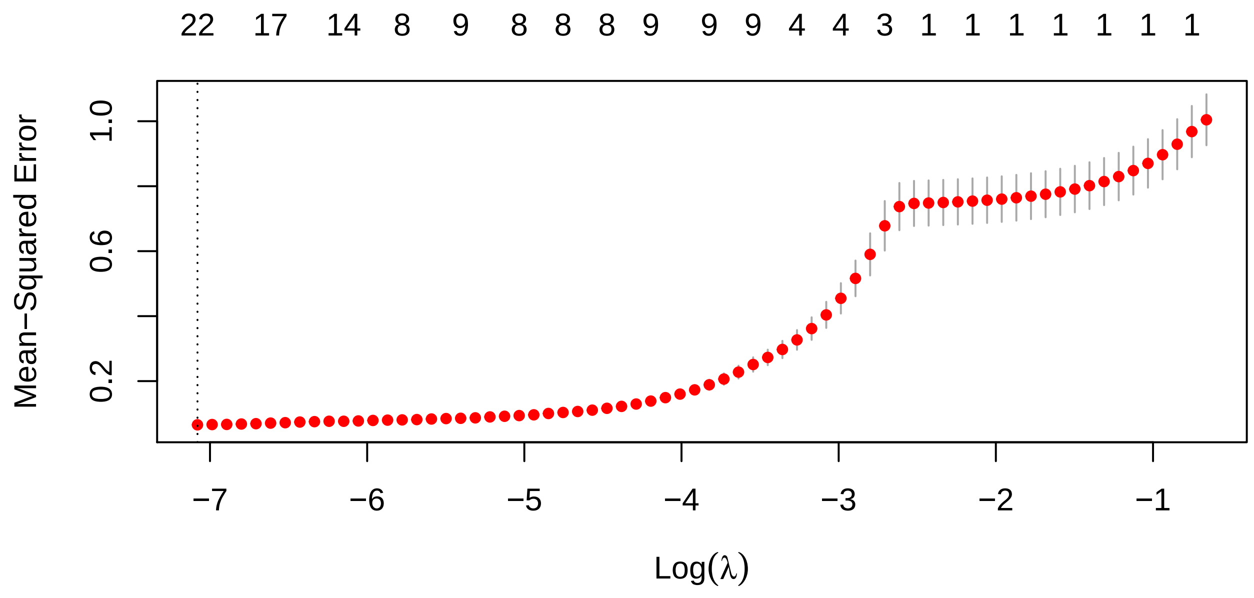


The main difference between LASSO and ridge regression is that in LASSO some coefficients can become zero and eliminated from the model, whereas in ridge regression there is no elimination of coefficients, just shrinkage. Larger penalties result in coefficient values closer to zero, which is the ideal for producing simpler models. LASSO produces sparse solutions as can be seen from the plot

I have removed the following line: "We observe that with LASSO the coefficients converge to zero faster compared to Ridge regression." because it doesn't make sense to compare the lambdas of each regression since the lambdas are 2 different parameters.

If we choose $\log \lambda = 0$, for example, almost all the coefficients in the Ridge Regression plot are non-zero while all the coefficients in the LASSO plot are 0.

7. Use cross-validation to find the optimal LASSO model (make sure that case $\lambda = 0$ is also considered by the procedure).



In the plot, the numbers in the top are the number of nonzero coefficient estimates. The lowest point in the curve indicates the optimal lambda and this exact value is:

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

Also, we can consider `lambda.1se` which is the "largest value of lambda such that error is within 1 standard error of the minimum" so, it eliminates some variables while still achieving comparable MSE. (We didn't consider this value in either the group report nor my report) This value is:

```
lasso_cv$lambda.1se
```

```
## [1] 0.0008421867
```

We can compare both values and its results in the following table (from the cv.glmnet() function):

```
lasso_cv
```

```
##
## Call:  cv.glmnet(x = as.matrix(covariates), y = response, lambda = lambdas_to_try,      alpha = 1, f
##
## Measure: Mean-Squared Error
##
##        Lambda Measure       SE Nonzero
## min 0.0000000 0.05932 0.006345     100
## 1se 0.0008422 0.06538 0.006221      22
```

So in the model using the optimal lambda ($\lambda = 0$) we will have all the variables.

In my individual report I had different values because actually I didn't consider the possibility of $\lambda = 0$

8. Compare the results from steps 4 and 7.

In question 4 we used `stepAIC()` to perform the stepwise method with both backward and forward types of selection variables and as result we obtained a model with 63 variables while in question 7 the `cv.glmnet()` function has been used to perform cross-validation and find the optimal LASSO model, and as a result a model with $\lambda = 0$ and all 100 variables has been obtained.

To compare these models, we will calculate the AIC values for each one.

```
##                           AIC
## step AIC             707.6913
## lasso cross_validated 173.1461
```

We can observe that the AIC associated to the stepAIC model is much greater, even though that model is quite more simple than the cross-validated lasso because in lasso no parameters have been dropped out.

In my individual report I compared both models using the R^2 values instead of the AIC's but when working with the group we thought that it was not necessary to calculate any measurement in order to compare the models.

Appendix

Assignment 1. Spam classification with nearest neighbors.

Question 1.

```r
# Importing the data:
library(readxl)
data <- read_excel("spambase.xlsx")

# Dividing it into training and test sets:
n=dim(data)[1]
set.seed(12345)
id=sample(1:n, floor(n*0.5))
train=data[id,]
test=data[-id,]
```

Question 2.

```r
# Logistic regression model:
fit1 <- glm(Spam~., data= train, family = binomial)

# Predictions:
fitted_prob_test <- predict(fit1, newdata=test,  type ="response") # type response to get probabilities
fitted_prob_train<- predict(fit1, newdata=train,  type ="response")

fitted_response_train <- ifelse(fitted_prob_train > 0.5, 1, 0)
fitted_response_test <- ifelse(fitted_prob_test > 0.5, 1, 0)

# Confusion matrices.
(confusion_matrix_test <- table(test$Spam, fitted_response_test))
(confusion_matrix_train <- table(train$Spam, fitted_response_train))

# Misclassification rates.
mean(fitted_response_test != test$Spam)
mean(fitted_response_train != train$Spam)
```

Question 3.

```r
fitted_response_train <- ifelse(fitted_prob_train > 0.8, 1, 0)
fitted_response_test <- ifelse(fitted_prob_test > 0.8, 1, 0)

# Confusion matrices.
(confusion_matrix_test <- table(test$Spam, fitted_response_test))
(confusion_matrix_train <- table(train$Spam, fitted_response_train))

# Misclassification rates
```

```r
mean(fitted_response_test != test$Spam)
mean(fitted_response_train != train$Spam)
```

Question 4.

```r
library(kknn)
knn_test_30 <- kknn(Spam~., train, test, k=30)
knn_train_30 <- kknn(Spam~., train, train, k=30)

knn_response_train_30 <- ifelse(knn_train_30$fitted.values > 0.5, 1, 0)
knn_response_test_30 <- ifelse(knn_test_30$fitted.values > 0.5, 1, 0)

# Misclassification rates
mean(knn_response_test_30 != test$Spam) # for the test data
mean(knn_response_train_30 != train$Spam) # for the training data
```

Question 5.

```r
knn_test_1 <- kknn(Spam~., train, test, k=1)
knn_train_1 <- kknn(Spam~., train, train, k=1)

knn_response_train_1 <- ifelse(knn_train_1$fitted.values > 0.5, 1, 0)
knn_response_test_1 <- ifelse(knn_test_1$fitted.values > 0.5, 1, 0)

# Misclassification rates
mean(knn_response_test_1 != test$Spam) # for the test data
mean(knn_response_train_1 != train$Spam) # for the training data
```

Assignment 3.

Question 1.

```r
mylin=function(X,Y, Xpred){
  Xpred1 = cbind(1,Xpred)
  X1 = cbind(1, X)
  beta = solve( t(X1) %*% X1) %*% t(X1) %*% Y # obtained using the "training" data matrix X
  Res = Xpred1%*%beta # y_hat for the "test" data
  return(Res)
}

myCV=function(X,Y,Nfolds){
  n=length(Y) # number of observations (rows)
  p=ncol(X) # number of covariates (variables or columns)
  set.seed(12345)
  ind=sample(n,n) # indexes are randomized
  X1=X[ind,] # randomize the order of the observations
  Y1=Y[ind]
  sF=floor(n/Nfolds) # number of observations inside each fold
  MSE=numeric(2^p-1) # vector of the length of 2^p-1 combinations
  Nfeat=numeric(2^p-1)
  Features=list() # features that will be selected
  curr=0 # current
  folds_obs <- cut(1:n,breaks=Nfolds,labels=FALSE)

  #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
            indices <- ind[which(folds_obs==k)] #indeces of the observations in fold k

#MISSING:implement cross-validation for model with features in "model" and iteration i.
            X_mylin <- X1[-indices,which(model==1)]
            XPred_mylin <- X1[indices,which(model==1)]
            Y_mylin <- Y1[-indices]

#MISSING:Get the predicted values for fold k, Ypred, and the original values for fold 'k', Yp.
              Ypred <- mylin(X_mylin, Y_mylin, XPred_mylin)
              Yp <- Y1[indices]

              SSE=SSE+sum((Ypred-Yp)^2)
            }
            curr=curr+1
            MSE[curr]=SSE/n
            Nfeat[curr]=sum(model)
            Features[[curr]]=model

          }

  plot(Nfeat, MSE) #MISSING: plot MSE against number of features
  abline(h=MSE[which.min(MSE)], col="red")

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

Question 2.

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

Assignment 4.

Question 1.

```
# Importing the data:
data <- read_excel("tecator.xlsx")
plot(data$Moisture, data$Protein, main="Plot of Moisture versus Protein",
     xlab="Moisture", ylab="Protein")
```

Question 3.

```r
# Creating new dataset with the new independent variables:
data2 <- data.frame( Moisture = data$Moisture, P1 = data$Protein,
                     P2 = data$Protein^2, P3 = data$Protein^3,
                     P4 = data$Protein^4, P5 = data$Protein^5,
                     P6 = data$Protein^6)
# Dividing the data into training and test sets:
n=dim(data2)[1]
set.seed(12345)
id=sample(1:n, floor(n*0.5))
train=data2[id,]
test=data2[-id,]

# Fitting the models and calculating MSE's:
mse_test <- vector()
mse_train <- vector()
for(i in 2:7){
  fit <- lm(train$Moisture ~ ., data = train[,1:i])
  mse_test[i-1] <- mean((predict(fit, test)- test$Moisture)^2)
  mse_train[i-1] <- mean((predict(fit, train)- train$Moisture)^2)
}

# MSE table for the training and the test datasets
(df <- as.data.frame(cbind(i=1:6,mse_test, mse_train)))

# MSE in a plot
library(ggplot2)

ggplot(data = df, aes( x = i)) +geom_point(data = df,
                                    aes( x = i, y = mse_train, color="Train")) +
  geom_line(data = df, aes( x = i, y = mse_train, color="Train")) +
  geom_point(data = df, aes( x = i, y = mse_test, color="Test")) +
  geom_line(data = df, aes( x = i, y = mse_test, color="Test")) +
  labs(x="i", y="MSE", colour="Dataset")
```

Question 4.

```r
library(MASS)
df = data[-c(1, 103, 104)]
model<- lm(Fat ~ . , data = df)
fit <- stepAIC(model, direction = "both", trace = F)
length(fit$coefficients)
```

Question 5.

```r
# First we have to scale de variables.
response <- scale(df[,101])
covariates <- scale(df[,1:100])

# Now we can fit the ridge regression model.
library(glmnet)
ridge <- glmnet(as.matrix(covariates), response, alpha=0,
                family="gaussian")
plot(ridge, xvar="lambda", label=TRUE)
```

Question 6.

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

Question 7.

```
lambdas_to_try<- append(lasso$lambda,0)
lasso_cv <- cv.glmnet(as.matrix(covariates), response, alpha = 1, lambda = lambdas_to_try, family="gaus
plot(lasso_cv)

lasso_cv$lambda.min #  value of lambda that gives minimum mean cross-validated error.
lasso_cv$lambda.1se
lasso_cv
```

Question 8.

```
# AIC value for the stepAIC() optimal model obtained:
AIC_step <- AIC(fit)

# LASSO model fitted with optimal lambda value
fit_lasso_cv <- glmnet(as.matrix(covariates), response,
                       lambda = lasso_cv$lambda.min)

# AIC for the CV lasso model:
tLL <- fit_lasso_cv$nulldev - deviance(fit_lasso_cv)
k <- fit_lasso_cv$df
n <- fit_lasso_cv$nobs
AIC_lasso <- -tLL+2*k+2*k*(k+1)/(n-k-1)

# COMPARISON TABLE:
aics <- cbind("AIC" = c(AIC_step, AIC_lasso))
rownames(aics) <- c("step AIC", "lasso cross_validated")
print(aics)
```