TDDE01 - Labs Block 2

Olof Simander (olosi122), Gustaf Lindgren (gusli281), Anton Jervebo (antje840)

Dec 2022

Writing: Olof Simander
Code: Olof Simander
Analysis: Olof Simander, Gustaf Lindgren, Anton Jervebo
Results discussed by: Olof Simander, Gustaf Lindgren, Anton Jervebo

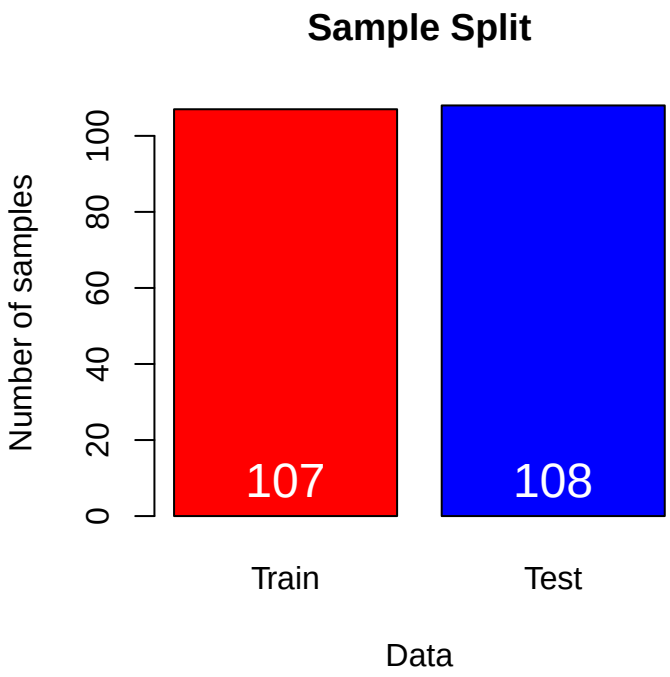
Task 0

Divide data randomly into train and test (50/50) by using the codes from the lectures.

```
library(dplyr)
```

```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
tecator1 = read.csv("tecator.csv", header = TRUE)
n = nrow(tecator1)

# Set up training data
set.seed(12345)
id = sample(1:n, floor(n*0.5))
train = tecator1[id,]
train = train %>% select(Channel1:Channel100,Fat)
id_dif = setdiff(1:n, id)

# Set up test data
test = tecator1[id_dif,]
test = test %>% select(Channel1:Channel100,Fat)

barpl = barplot(c(Train = nrow(train), Test = nrow(test)), main = "Sample Split", xlab = "Data", ylab =
text(barpl,0, c(Train = nrow(train), Test = nrow(test)), cex = 1.5, pos = 3, col = "white")
```



Task 1

- Create a linear regression model based on Channels and predicting Fat content. Use training data and estimate MSE for both training and test data.
- Report the underlying probabilistic model, fit the linear regression to the training data and estimate the training and test errors.

$$P(y^{(i)}|x^{(i)}) = \frac{1}{\sqrt{2\pi\sigma}} exp(-\frac{(y^{(i)} - \theta^T x^{(i)})^2}{2\sigma^2})$$

```
m1 = lm(Fat~., train)

preds_train = predict(m1, train, type = "response")
MSE_train = mean((train$Fat - preds_train)^2)
cat("Training data MSE:",MSE_train, "\n")
```

```
## Training data MSE: 0.005709117
```

```
cat("Correlation for training data:", cor(train["Fat"], preds_train)^2, "\n")
```

```
## Correlation for training data: 0.9999637
```

```
preds_test = predict(m1, test, type = "response")
MSE_test = mean((test$Fat - preds_test)^2)
cat("Test data MSE:",MSE_test, "\n")
```

```
## Test data MSE: 722.4294
```

```
cat("Correlation for test data:", cor(test["Fat"], preds_test)^2)
```

```
## Correlation for test data: 0.4086971
```

- Comment on the quality of fit and prediction and therefore on the quality of model.

There is a large discrepancy between the training and test data in terms of MSE where the test data performs considerably worse than the training data. This is an indication that the linear regression model. This becomes even more clear when comparing the R-squared metrics: 0.9999637 and 0.4086971 for the training and test data respective.

Task 2

- Report he cost function that should be used to optimize the model.

$$\frac{1}{n} \cdot \sum_{i=1}^{n} \sum_{j=1}^{100} (y_i - \theta_0 - \theta_1 x_{i1} - \theta_2 x_{i2} - \ldots - \theta_{100} x_{i100})^2 + \lambda \sum_{j=1}^{100} |\theta_j|$$

This is the cost function we want to minimize to find the optimal model.

Task 3

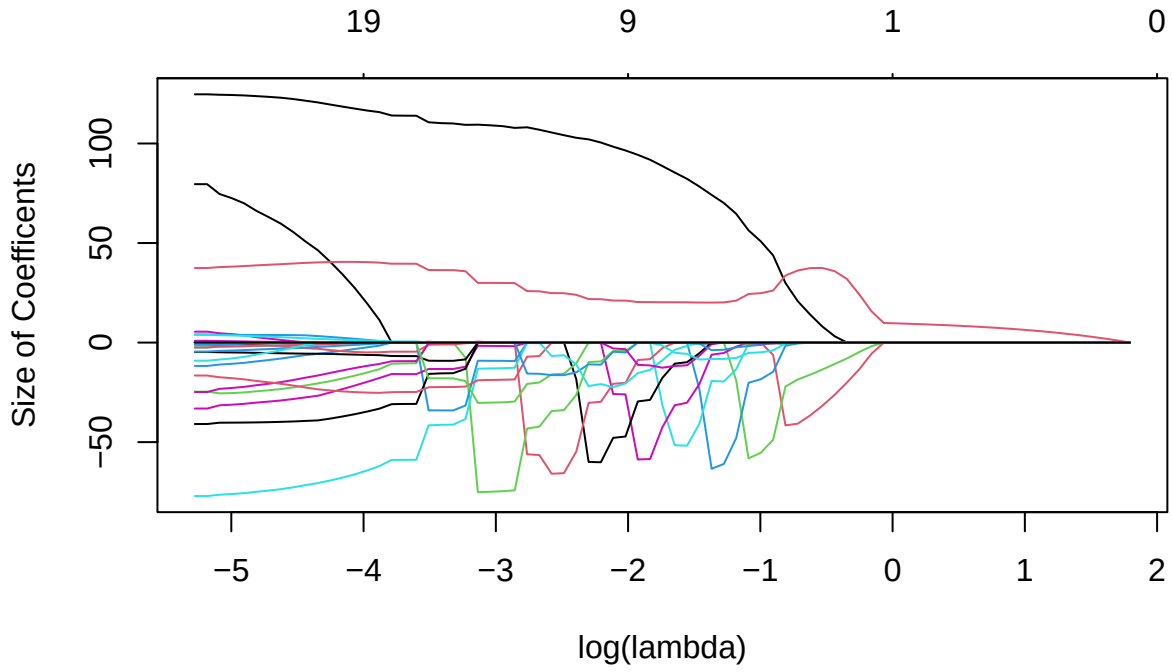
- Create a LASSO regression model based on the previous training data, also predicting the Fat content. Fit the LASSO regression model to the training data.
- Present a plot illustrating how the regression coefficients depend on the log of penalty factor $log(\lambda)$ and interpret this plot. What value of the penalty factor can be chosen if we want to select a model with only three features?

```
library(glmnet)
```

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

```
## Loaded glmnet 4.1-4
```

```
m_l = glmnet(as.matrix(train[,1:100]), as.matrix(train[,101]), family = "gaussian", alpha = 1)

# Plot results
par(oma=c(0,0,2,0))
plot(m_l, xvar = "lambda", xlab = "log(lambda)", ylab = "Size of Coefficents")
title(main = "Lasso", outer = TRUE)
```




```
# The positions with three non-zero coefficents are 22,23,24
cat("The lambdas with df = 3 are:", which(m_l$df == 3), "\n")
```

```
## The lambdas with df = 3 are: 22 23 24
```

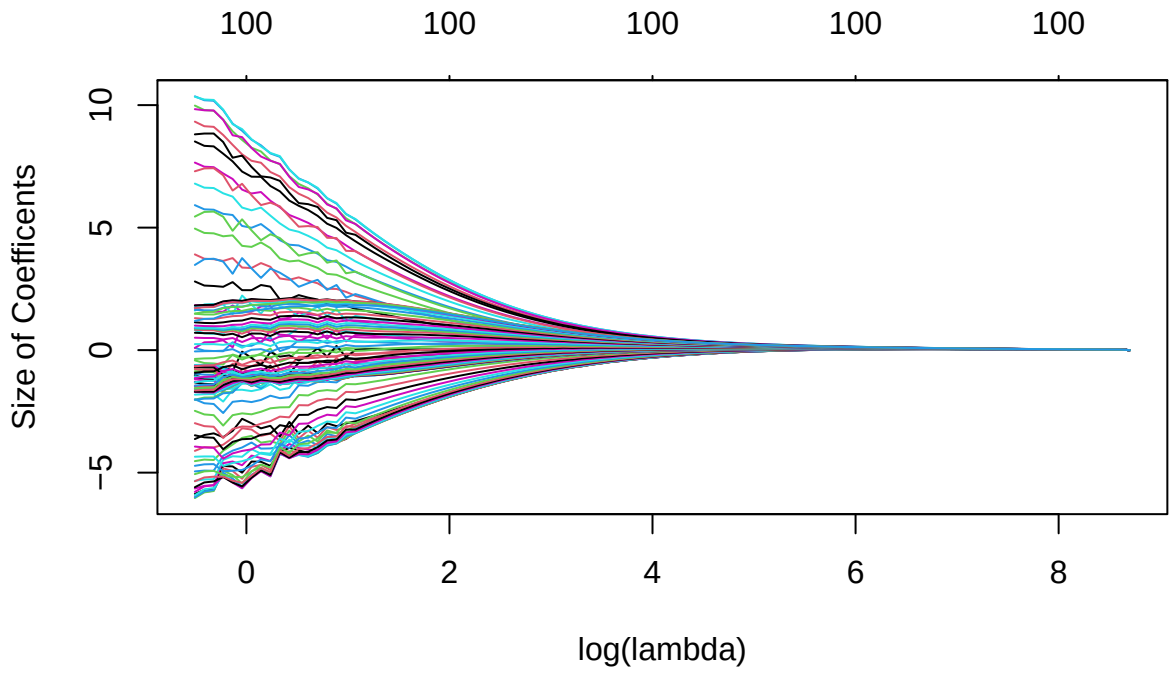
```
cat("The corresponding lambda values are:",m_l$lambda[22:24],".\n", "These can be selected to have exac
```

```
## The corresponding lambda values are: 0.8530452 0.777263 0.7082131 .
##  These can be selected to have exaclty 3 features in the model.
```

Task 4

- Repeat step 3 but fit Ridge instead of the LASSO regression and compare the plots from steps 3 and 4.

```
m_r = glmnet(as.matrix(train[,1:100]), as.matrix(train[,101]), family = "gaussian", alpha = 0)

# Plot results
par(oma=c(0,0,2,0))
plot(m_r, xvar = "lambda", xlab = "log(lambda)", ylab = "Size of Coefficents")
title(main = "Ridge", outer = TRUE)
```

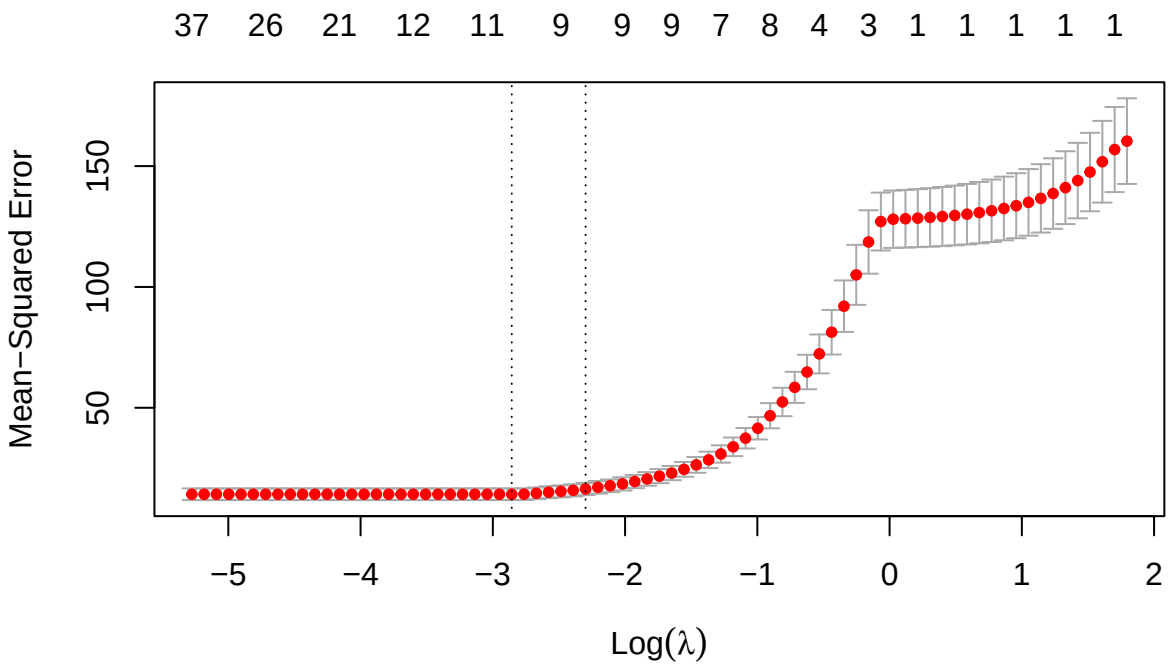


Comparing the plots, it is clear that the ridge regression does not completely eliminate coefficients. This is why we cannot find a model that has only three degrees of freedom. However, because of the penalty factor of $\lambda \cdot \sum_j \theta^2 \ \forall \ j$ we can see that the higher values for lambda does in fact lower the size of the coefficients overall.

Task 5

- Use cross-validation with default number of folds to compute the optimal LASSO model. Present a plot showing the dependence of the CV score on *log* λ and comment how the CV score changes with *log* λ.
- Report the optimal λ and how many variables were chosen in this model.
- Does the information displayed in the plot suggest that the optimal λ value result in a statistically significant better prediction than *log* $\lambda = -4$?
- Create a scatter plot of the original test versus predicted test values for the model corresponding to optimal lambda and comment whether the model predictions are good.

```
m_l = cv.glmnet(as.matrix(train[,1:100]), as.matrix(train[,101]), family = "gaussian", alpha = 1)

# Plot the results
par(oma=c(0,0,2,0))
plot(m_l)
title(main = "Lasso Optimization", outer = TRUE)
```




```
# Evaluation comparing to previous model
cat("The optimal lambda is:", m_l$lambda.min, "\n")
```

```
## The optimal lambda is: 0.05744535
```

```
cat("There are", m_l$nzero[51], "selected channels.", "\n")
```

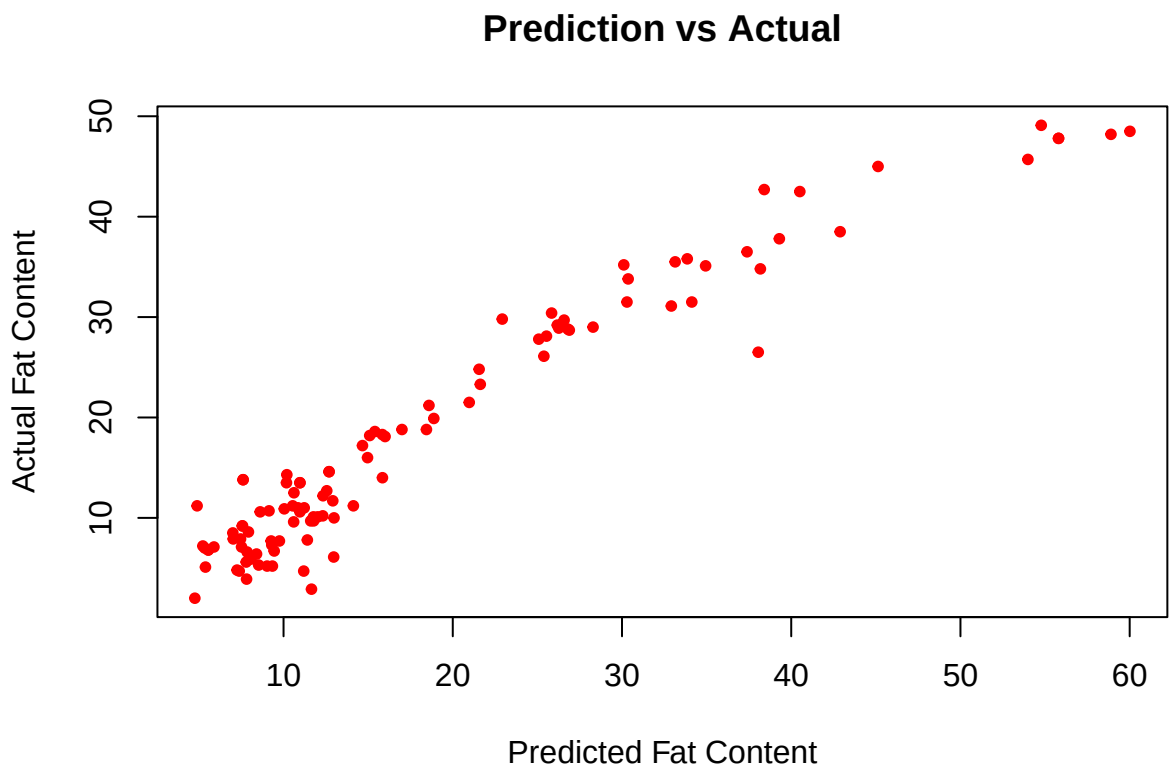
```
## There are 8 selected channels.
```

```
# Find what lambda values for which df == 1
#df_1 = (which(m_l$nzero==1))
#log(m_l$lambda[20])
```

Observing the LASSO optimization based on lambda with regards to MSE, the error is constant before the first dotted line from the left. This implies that increasing the number of variables, by lowering the regularization term, does not improve the model further. This would result in unnecessary complexity in the model. Beyond this point, until $log\ \lambda = 0.02712472$ and $df = 1$, the MSE is growing exponentially. This is reasonable because of decreasing complexity of the model should have a disproportionate impact on the performance of the model as the one-by-one exclusion of the maximum number features will have a larger and lager impact (i.e. the latter features removed will better predict the label, and their exclusion will have a larger negative impact on the MSE). Beyond $log\ \lambda = 0.02712472$, the momentum is lost and a new exponential increase is developing. This is because the model one has 1 feature left. Therfore, the increase in error from this point comes from the lowering of the coefficient for this last feature.

Based on the plot, we can conclude that $log\ \lambda = -4$ does not produce a statistically significant better result than the optimal $log\ \lambda = -2.856921$. The difference in MSE is minimal to the eye and compared to the remaining data points for higher values for lambda. This is evident because of the close proximity of the lowest and optimal lambda values in the plot.

```
m_min = glmnet(as.matrix(train[,1:100]), as.matrix(train[,101]), family = "gaussian", alpha = 1, lambda

# Create predictions for test data
pred_min = predict(m_min, newx = as.matrix(test[,1:100]), type = "response")

# Plot the the actual values against predictions
plot(x = pred_min, y = test[,101], main = "Prediction vs Actual", xlab = "Predicted Fat Content", ylab =
```




```
# Results for the test data
cat("The MSE for the optimal model is:", mean((test[,"Fat"] - pred_min)^2),"\n")
```

```
## The MSE for the optimal model is: 13.2998
```

```
cat("The correlation is:",cor(pred_min,test[,101]))
```

```
## The correlation is: 0.9644486
```

This model predicts the test data much better than the original linear regression model: MSE of 13.31354 versus 722.4294 respectively. This is a much better model since it does not over fit the data by including

all of the features in the data. This becomes more insuperable when comparing the correlation with the predictions and the label from the test results: LASSO model 0.9649745 versus for the linear regression model 0.4086971 for the test data.