```
#### Lab 1 - Assignment 4 - Linear regression and regularization

### Libraries
library(MASS)
library(glmnet)

### Setup
data.tecator = read.csv("tecator.csv", dec=',')
n = dim(data.tecator)[1]
data.tecator = data.tecator[-n,]
set.seed(12345)
ids = sample(1:n, floor(n/2))
train = data.tecator[ids,]
y.train = train$Moisture
test = data.tecator[-ids,]
y.test = test$Moisture
DEGREES = 6

### Functions

MSE = function(Y, Yfit){
  return(mean((Y-Yfit)^2))
}

### Implementation
## Task 1
plot(data.tecator$Protein, data.tecator$Moisture,
     main="Protein vs. Moisture", xlab="Protein", ylab="Moisture")
# Comment: This data can be described by a linear model, with the
exception to some outliers

## Task 2
# Mi = B0+B1*p+B2*p²+...+Bi*p^i+ε where ε ~N(0,σ²)
# It is appropriate to use the MSE criterion since it takes both bias
and variance into account
# However, the training data has a couple of outliers that with weigh
heavily

MSEs.train = numeric(DEGREES)
MSEs.test = numeric(DEGREES)

## Task 3 - Fit models
for(i in 1:DEGREES){
  # Fit model
  linear.model = lm(Moisture~poly(Protein, i), data=train)

  # Train
  pred.train = linear.model$fitted.values
  MSEs.train[i] = MSE(y.train, pred.train)

  # Test
  pred.test = predict(linear.model, test)
  MSEs.test[i] = MSE(y.test, pred.test)
}
```

```
plot(MSEs.train, type="l", col="green", ylim=c(31, 34),
     main="MSEs", xlab="Polynomial degree i", ylab="MSE value")
lines(MSEs.test, col="blue")
legend("bottomleft", legend=c("Training", "Test"),
col=c("green","blue"), lty=1)
# Comment: The MSE for training set strictly decreases with higher
polynomial degree -> reduced bias
# The MSE for the test set is at its lowest for a strictly linear
model, suggests that the variance increases
# It also suggests that a higher polynomial degree results in
overfitting of the training data

#M1 = lm(Moisture ~ Protein, data=training)
#M2 = lm(Moisture ~ Protein+I(Protein^2), data=training)
#etc.

## Task 4 - Reduce the linear model's features used to make prediction
using stepAIC, AIC variable selection

# Create formula
formula = "Fat~.-Sample-Protein-Moisture"
# channels = !colnames(data.tecator) %in% c("Sample", "Fat",
"Protein", "Moisture")
# channels.string = paste(colnames(data.tecator[,channels]), collapse
= "+")
# formula = paste("Fat~", channels.string)

# Fit model
linear.model = lm(formula, data=data.tecator)

# Reduce model
linear.model.AIC = stepAIC(linear.model, data=data.tecator,
direction="both", trace=F)
# Comment:
# 63 variables selected out of 100, indiciates that 63 variables are
enough to describe the data sufficiently

## Task 5 - Regularization using Ridge regression model
# Scale values
channels.scaled = as.matrix(scale(data.tecator[2:101]))
fat.scaled = scale(data.tecator$Fat)
# Fit model
ridge.model = glmnet(channels.scaled, fat.scaled, family="gaussian",
alpha=0)
# Plot
plot(ridge.model, xvar="lambda", main="Ridge Regression")
# Comment:
# All 100 variables selected, the coffecients value reduce with larger
lambdas
# The coffecients' values converge towards 0 in an exponential pattern

## Task 6 - Regularization using LASSO regression
# Fit model
lasso.model = glmnet(channels.scaled, fat.scaled, family="gaussian",
```

```
alpha=1)
# Plot
plot(lasso.model, xvar="lambda", TRUE, main="Lasso Regression")
# Comment:
# A lot fewer variables selected straight away
# A lot of coffecients value are equal to 0 straight away
# Channel 41's coffecient is non-zero longer than the rest
# Both converge to 0 but in different patterns
# Both have larger penalty with larger lambdas

## Task 7 - Cross-Validation
lasso.model.cv = cv.glmnet(channels.scaled, fat.scaled,
family="gaussian", alpha=1, lambda=seq(0,1,0.001))
plot(lasso.model.cv, xvar="lambda", label=TRUE, main="Cross-Validation
LASSO", xlab="LogLambda")
lambda.min = lasso.model.cv$lambda.min # "best" lambda = 0, no
penalty/regularization
# Comment;
# The MSE gets higher with higher penalty for lambda
# However, the performance is not that much higher from 100 variables
to eg. 9 variables

## Task 8
# In step 4 the stepAIC algorithm chose 63 variables, whereas the
optimal LASSO
# model in step 7 chose 100 variables. The stepAIC algorithm does not
necessarily
# choose the optimal model. It just tries to find a model with a
better score until the
# next model isn't better.
```