TDDE01. Lab2. Group B24 report.

# **Statement of contribution**

Firstly, general analysis was performed teamwise. Approaches and strategies of solving the tasks were elaborated teamwise as well.

Student Elena was responsible for code writing and problem solving for the Assignment 1. Student Anton was responsible for code writing and problem solving for the Assignment 2. Student Elham was responsible for code writing and problem solving for the Assignment 3.

After completion of coding stage group analyzed the results together and peer reviewed the results of each other’s work. Finally, each student corrected their solutions according to the received reviews from groupmates.

# **Assignment 1. Explicit regularization.**

Task 1: Raw data description: the “tecator.csv” 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.

Columns containing levels of moisture, protein and sample number were removed from the dataset because they are not involved into analysis.

The raw data was divided by 50/50 to train and test data.

Assuming that Fat can be modeled as a linear regression in which Channels are used as features, we have the following probabilistic model:

The linear regression was fitted to the training data and training and test errors were estimated:

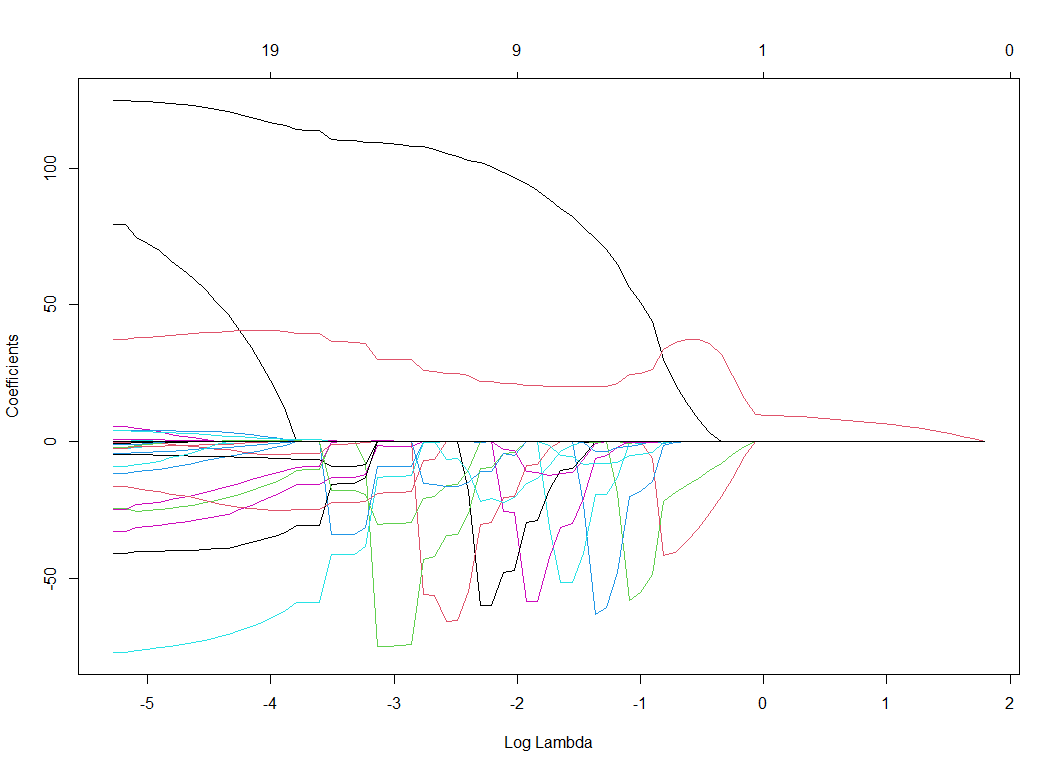
|  |  |
| --- | --- |
| Mean squared error (train data) | Mean squared error (test data) |
| 0.005709117 | 982.2478 |

The model fits the training data well, but as seen from the value of test error, the model is overfitted and too complex. It memorized the train dataset but performed poorly on test dataset.

Task 2: Assuming that Fat can be modeled as a LASSO regression with Channels as features, we have the following cost function to optimize:

Where > 0 is penalty factor, y is target, x is features, is model parameters.

Task 3: The LASSO regression model was fitted to the training data (alpha=1).

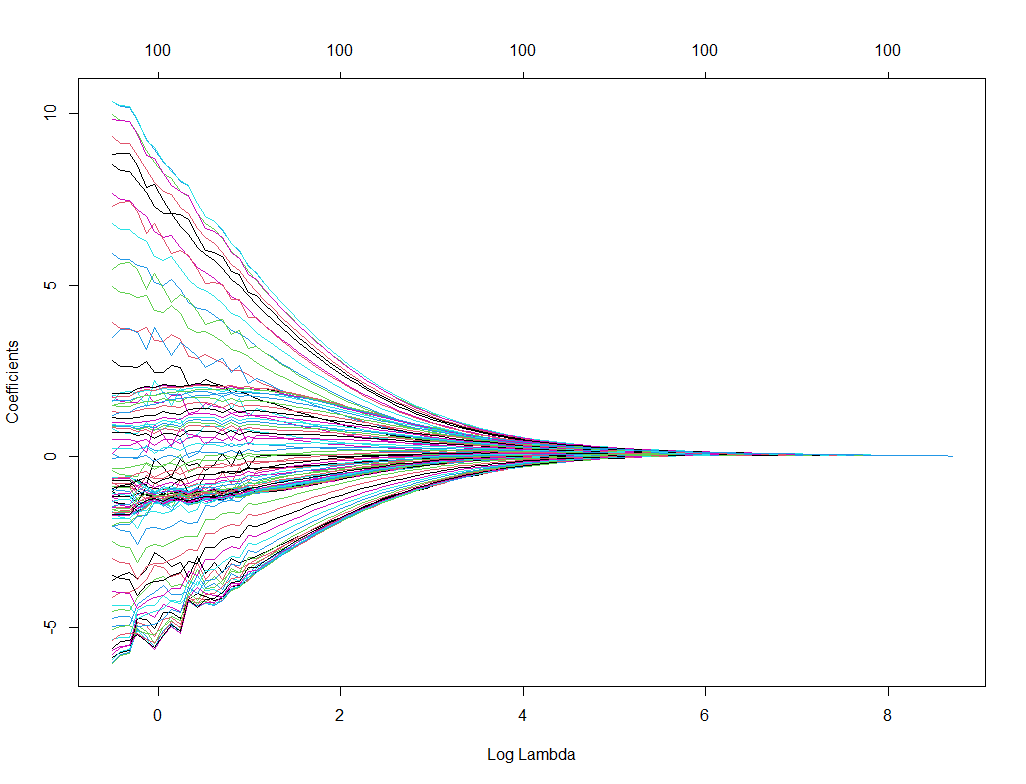
Figure 1. Dependency of the regression coefficients on the log of penalty factor (log 𝜆). LASSO.

When increasing the penalty factor, we shrink the number of variables in the model. L1 regularization term forces parameters to zero and thus can be used as a feature selection method.

To select a model with only three features one can pick 𝜆 = 0.853045182.

Task 4: After that we applied the Ridge regression to the data (alpha=0).

Figure 2. Dependency of the regression coefficients on the log of penalty factor (log 𝜆). Ridge.

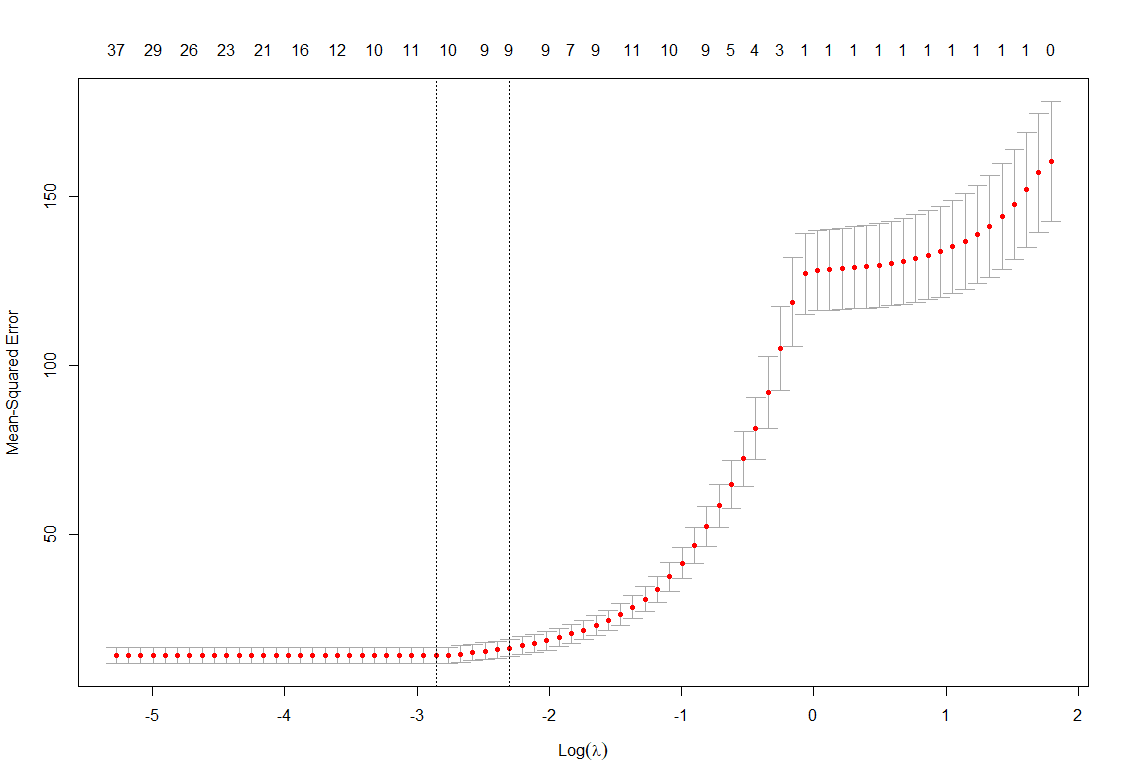


L2 regularization term pushes parameters to small values while 𝜆 increases. Unlike L1, it doesn’t force them necessarily to zero. In contrast to Ridge, the LASSO regularization will actually set less-important variables to zero and thus helps to simplify the model and throw away less important parameters.

It may be beneficial to use L1 with a high-dimensionality data.

Task 5: Cross-validation process with the default number of folds was used to compute the optimal LASSO model

Figure 3. Cross-validation.



The dotted line on the left represents the optimal 𝜆 which minimizes cross-validation error. The dotted line on the right represents the point with one standard error of optimal 𝜆. After that error grows.

As it can be seen from the cross-validation results, optimal 𝜆 = 0.05745 (with the log 𝜆 = -2.85684027469).

To understand the number of variables with optimal 𝜆 we built a LASSO model with alpha=1, lambda=0.05745. and figured out that 7 variables were used in the optimal model.

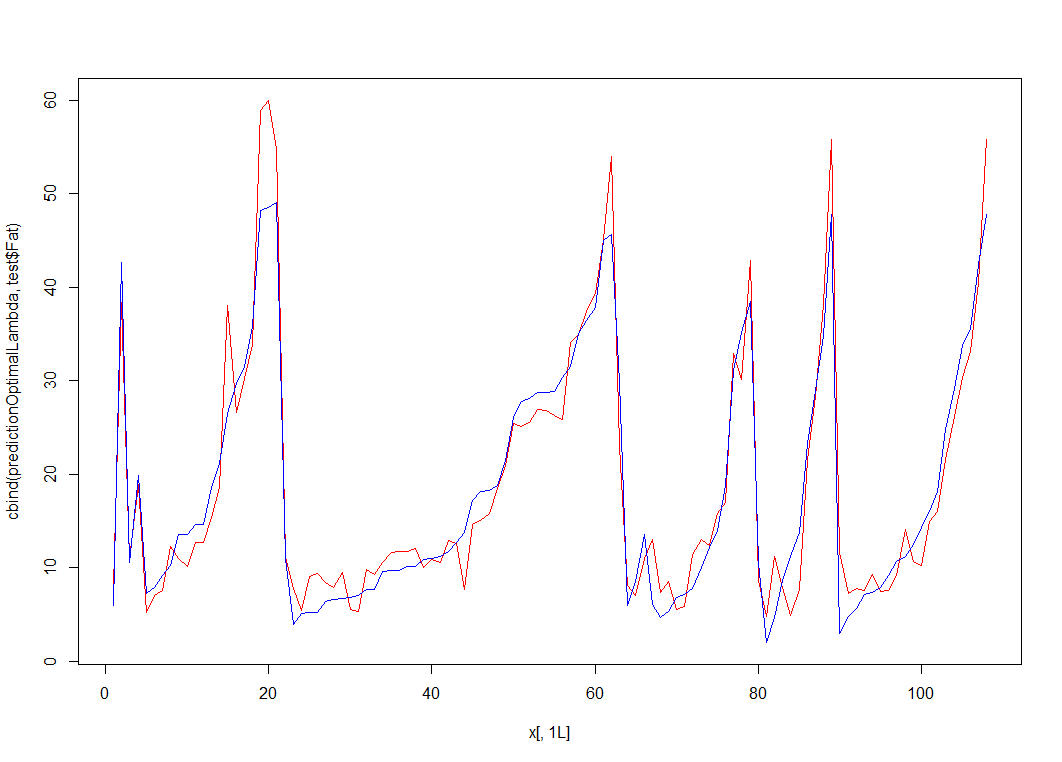
To understand if the optimal 𝜆 gives better prediction that 𝜆 with log 𝜆 = -4, we directly compared the error for those two 𝜆 values:

|  |  |
| --- | --- |
| Error value for optimal 𝜆 | Error value for 𝜆 with log 𝜆 = -4 |
| 14.21175 | 14.22138 |

Although the value of the error for optimal 𝜆 is less than for log 𝜆 = -4, there is probably no significant statistical difference as the values are very close to each other.

Finally, we created a plot of the original test versus predicted test values for the model corresponding to optimal lambda.

Figure 4. Test data versus prediction with optimal 𝜆.



It can be seen from the graph that the optimal model (red line) predicts the test data (blue line) quite well. If we check the MSE value for optimal model and compare it with the very first linear regression model, one can see significant improvement:

|  |  |
| --- | --- |
| MSE (optimal model) | MSE (linear regression) |
| 13.2998 | 982.2478 |

# **Appendix. Code.**

Assignment 1.

#########################################Preparations##########################################

tecator=read.csv("tecator.csv")

#REMOVE UNNECESSARY COLUMNS FIRST

tecator$Protein=c()

tecator$Moisture=c()

tecator$Sample=c()

#dividing the train and test

set.seed(12345)

n=nrow(tecator)

id=sample(1:n, floor(n\*0.5))

train=tecator[id,]

test=tecator[-id,]

#model based on training data

linearModel=lm(Fat~., data=train)

sum=summary(linearModel)

############################################Task 1.###########################################

#MSE for train and test

MSEtrain=mean(sum$residuals^2)

#calculate MSE for test data set

predictionTest=predict(linearModel,newdata=test,interval = "prediction")

MSEtest=mean((test$Fat-predictionTest)^2)

###########################################Task 2.############################################

#cost function - see the report

#########################################Task 3.############################################

library(glmnet)

library(dplyr)

#creating lasso regression model

x=as.matrix(train%>%select(-Fat))

y=as.matrix(train%>%select(Fat))

LASSOmodel=glmnet(x, y, alpha=1)

plot(LASSOmodel, xvar = "lambda")

#########################################Task 4.############################################

#Ridge regression and plot of lambda

ridgeRegressionModel=glmnet(x, y, alpha=0)

plot(ridgeRegressionModel, xvar="lambda")

#####################################Task 5.################################################

#cross validation

modelCrossValidation <- cv.glmnet(x=x, y=y, alpha=1, family="gaussian")

plot(modelCrossValidation)

#lambda min is 0.05745

print(modelCrossValidation$lambda.min)

#build a model with minLambda and see how many variables are there

LASSOmodelOptimalLambda=glmnet(x, y, alpha=1,lambda=modelCrossValidation$lambda.min)

print(LASSOmodelOptimalLambda$beta)

#there are 7 variables for the model with optimal lambda

# 0.01831563888 is lambda for log λ = −4, it is located somewhere between 63 and 64 position in lambda array of cross-validation dataset

print(modelCrossValidation$cvm[63])

print(modelCrossValidation$cvm[64])

print(modelCrossValidation$cvm[51])

print(modelCrossValidation$cvsd[63])

print(modelCrossValidation$cvsd[64])

print(modelCrossValidation$cvsd[51])

#plot of the original test versus predicted test values for the model corresponding to optimal lambda

x1=as.matrix(test%>%select(-Fat))

predictionOptimalLambda=predict(object = LASSOmodelOptimalLambda,newx=x1,s="lambda.min")

matplot(y=cbind(predictionOptimalLambda,test$Fat),type="l",col=c("red","blue"),lty=c(1,1))

#check mse for linear regression model in task 1 and compare with mse for optimal model

MSEOptimalLambda=mean((predictionOptimalLambda-test$Fat)^2)

print(MSEOptimalLambda)