# Introduction to Machine Learning
## *TDDE01 - Lab2*

Author: *David Tran - davtr766*

## I. Introduction

As part of the TDDE01 course, Introduction to Machine Learning, the students are obliged to complete a series of three labs in order to complete the course. The purpose of this lab is to learn more about decision trees, nave bayes and principal components.

## II. Assignment 2 - Analysis of credit scoring

In the first assignment a data file *creditscoring.csv* is given containing data about customers with several features that combined determines how the customers have managed their loans in terms of the variable *good* or *bad*. The task is to derive a prediction model that can be used to predict whether or not a new customer is likely to pay back the loan.

### A. 2.1

The first task is to divide the data into training, validation and test with the distribution of 50%, 25% and 25%.

```
data = read.csv2("creditscoring.csv")

# Exercise 1
#Divide data into training/validation/test
    as 50/25/25.
n=dim(data)[1]
set.seed(12345)
id = sample(1:n, floor(n*0.5))
train = data[id,] # 50% training.
validation_test = data[-id,] # 50%
    validation/test.

n1 = dim(validation_test)[1]
new_id = sample(1:n1, floor(n1*0.5))
validation = validation_test[new_id,] # 25%
    validation.
test = validation_test[-new_id,] # 25% test
```

### B. 2.2

The next task is to fit a decision tree to the training data using the impurity measures deviance and gini index.

*1) Missclassification rate for deviance:*
Training data: 0.212
Test data: 0.284

*2) Missclassification rate for gini index:*
Training data: 0.25
Test data: 0.348

The result implies that the impurity measure deviance provides better results.

### C. 2.3

The next task is to use the training and validation sets to determine the optimal tree depth. The following plot shows the dependence of deviances for the training and validation data on the number of leaves:

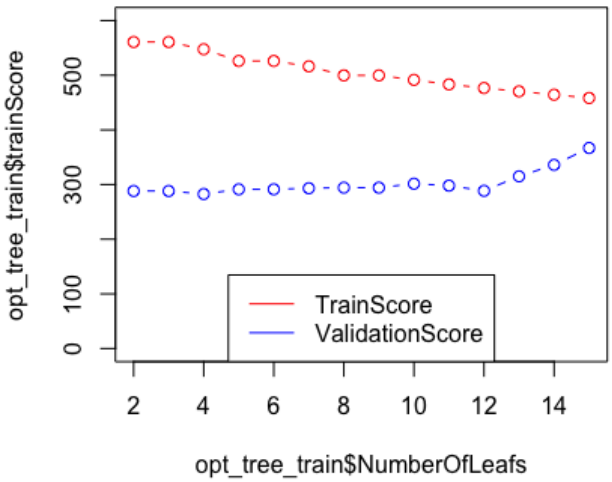


Fig. 1. Dependences of deviances

.

By interpreting the plot one can say that the optimal number of leaves are 4 because the goal is to have the lowest amount of deviance in the blue

curve. Therefore, the tree model used is pruned with number of terminal nodes equal to 4. The optimal tree is presented below:

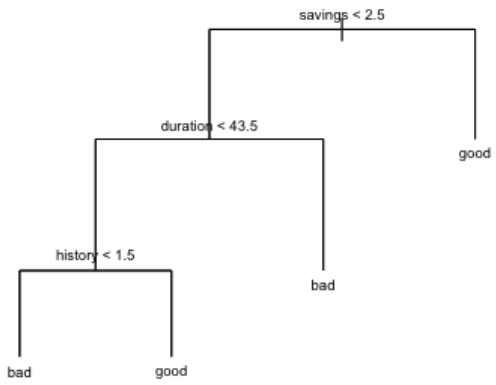


Fig. 2. Optimal tree

The depth of the tree is three and every node has a condition which determines if the customer is good or bad. The missclassification rate for the test data is 0.26.

### D. 2.4

The next task is to use the training data to perform classification using Nave Bayes. The confusion matrix for training data and test data is presented below:

```
Confusion matrix for training data
       bad good
  bad   95   52
  good  98  255
Missclassification rate:  0.3


Confusion matrix for test data
       bad good
  bad   47   31
  good  49  123
Missclassification rate: 0.32
```

The conclusion is that since the missclassification rate is lower for the tree model compared to Naves Bayes then for this particular data, the tree model performs better.

### E. 2.5

The last task of this assignment is to use the loss matrix for the Nave Bayes classifcation below:

$$L = \begin{matrix} \\ Observed \\ \\ \end{matrix} \begin{matrix} Predicted \\ good \\ bad \end{matrix} \begin{pmatrix} 0 & 1 \\ 10 & 0 \end{pmatrix}$$

Fig. 3. Loss Matrix

The confusion matrix for training data and test data is presented below:

```
Confusion matrix for training data
       bad good
  bad  137   10
  good 263   90
Missclassification rate:  0.546


Confusion matrix for test data
       bad good
  bad   70    8
  good 131   41
Missclassification rate: 0.556
```

This is done because the importance factor is larger if a customer is entitled a loan even though the customer does not meet the standard qualifications. Therefore, it is necessary to penalize the false negatives.

## III. Assignment 3 - Uncertainty Estimation

In the second assignment, a data file *State.csv* is given containing data about per capita state,local public expenditures and associated state demographic and economic characteristics. The important variables to take note are:

1) MET: Percentage of population living in standard metropolitan areas.
2) EX: Per capita state and local public expenditures ($)

### A. 3.1

The first task of this assignment is to reorder the data with respect to the increase of MET and afterwards plot EX vervus MET. The plot is presented below:

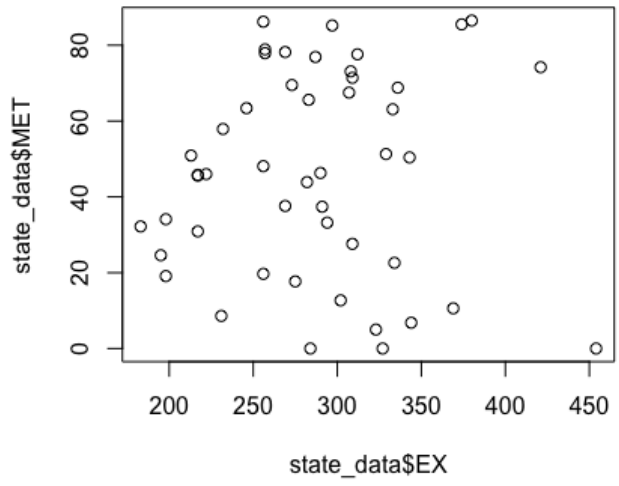


Fig. 4. EX versus MET

From the plot in *Fig.4*, one can conclude that a classification tree model may be appropriate for this kind of data.

*B. 3.2*

The next step is fit the model with *tree()* and then perform cross-validation to determine the appropriate number of leaves.. The following plot below:

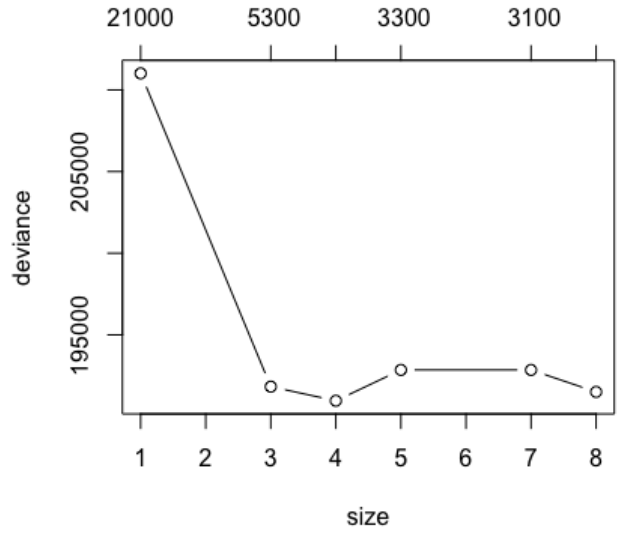


Fig. 5. Deviance vs Size

determines the appropriate number of leaves and in this case it is 4. Therefore, the fitted tree model can be pruned with size equal to 4. The reported tree is presented below:

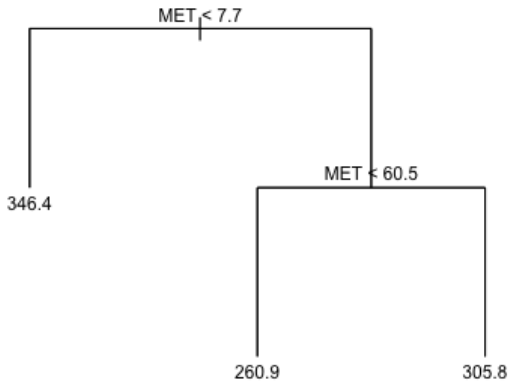


Fig. 6. Selected tree

The plot for the original and fitted data is given below:

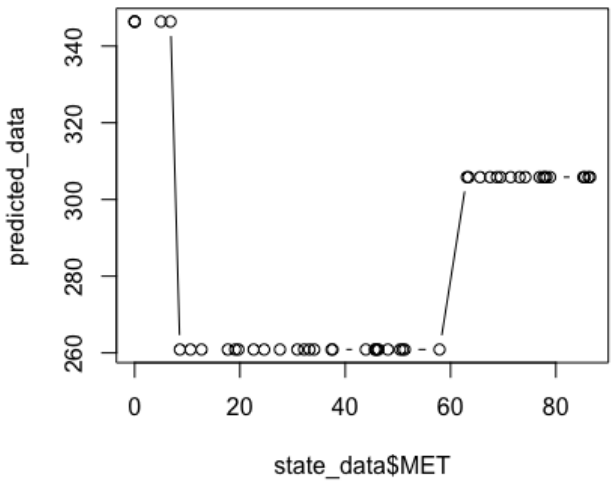


Fig. 7. Plot of original and fitted data

The histogram of residuals plot are presented below:

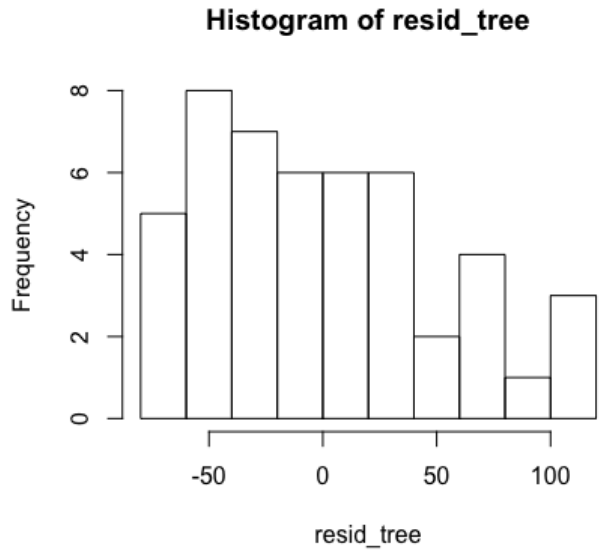


Fig. 8. Histogram of residuals

By interpreting the plot it would have been better if the histogram resembled a normal distribution. In this case, it is more likely to underestimate with a small error of -50 ˜20.

### C. 3.3

The next step is to use a non-parametric bootstrap to plot the 95% confidence bands for the regression tree model. The result is presented below:

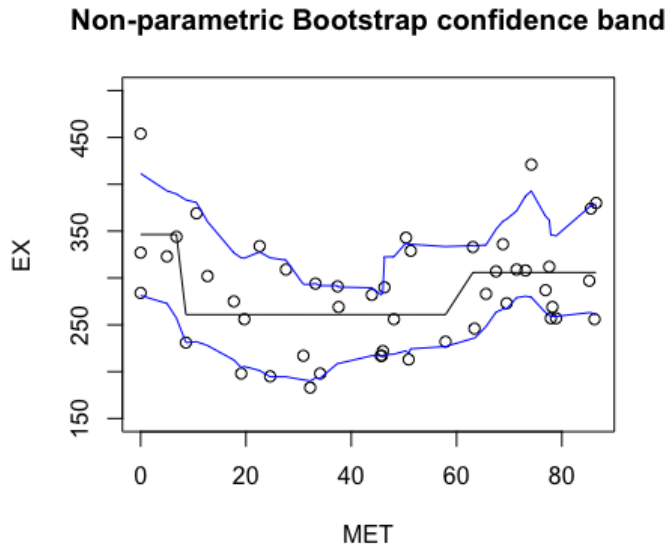


Fig. 9. Histogram of residuals

The width of the confidence band increases at around MET 20-30. This means our level of uncertainty increases when MET = 20-30. The reason why the confidence bands are bumpy is because it is not dependent on the distribution but rather on the data values.

### D. 3.4

The next step is to use a parametric bootstrap:

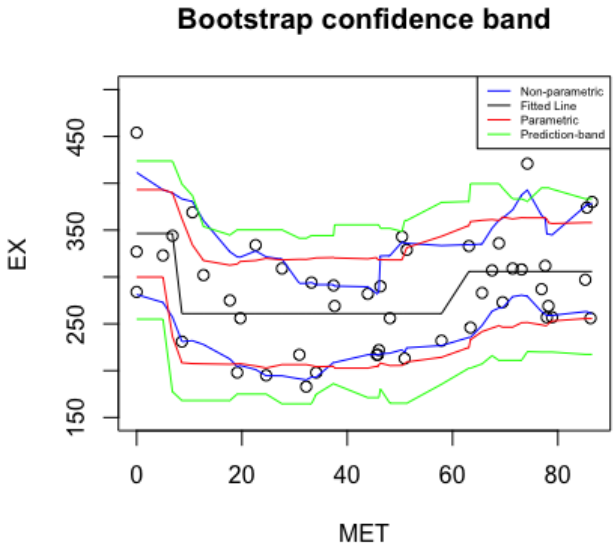


Fig. 10. Bootstrap confidence bands

which is the red curve. One can observe that the parametric bootstrap confidence band is more smooth compared to the non-parametric because it is based on the model and not the data values. From the plot the conclusion is that it is reliable because less than 5% are outside the prediction band (green curve).

When considering the histogram of residuals(Fig.8), the non-parametric bootstrap would be more appropriate in this case since parametric bootstrap assumes that we know the normal distribution is correct and by interpreting the histogram it does not look like a normal distribution.

## IV. Assignment 4 - Principal Components

For the last assignment of this lab, we are given a data file *NIRSpecta.csv* containing near-infrared spectra and the viscosity levels for a collection of different diesel fuels. The main task of this assignment is to investigate how measured spectra can be used to predict the viscosity.

### A. 4.1

The first task is to conduct a standard PCA by using feature space and then provide a plot explaining how much variation is explained by each feature. The mentioned plot can be seen below:

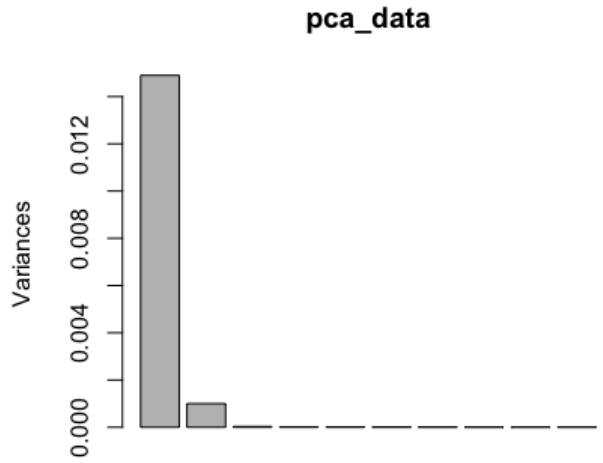


Fig. 11. Variance vs number of PC

The plot shows the variances against the number of the principal component and we can clearly see that using two principal components will extract at least 99% of the total variance. The next plot is the scores in the coordinates PC1 and PC2:

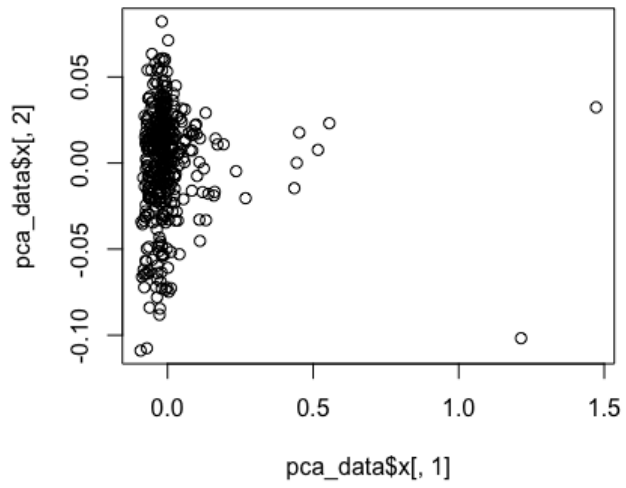


Fig. 12. Score plot

By interpreting *Fig. 12* one can conclude that there are unusual diesel fuels according to the plot that deviates from the PC axis. This means that those particular diesel fuels that deviates from the axis are not heavily impacted.

### B. 4.2

Next step is to trace plot of the loadings of the selected components(PC1 and PC2):

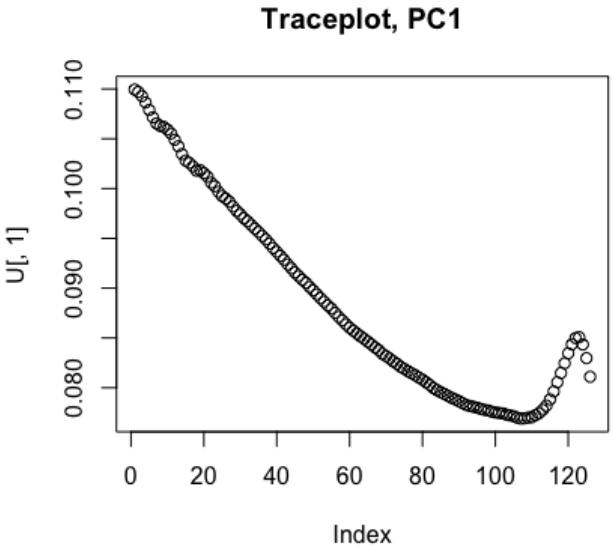


Fig. 13. Trace plot PC1

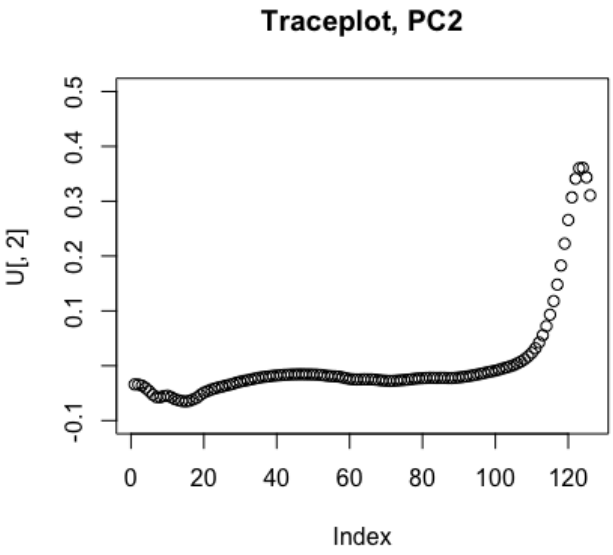


Fig. 14. Trace plot PC2

In the above plots one can conclude that *Fig.14* PC2 is explained by mainly the last 10 features because the last features have a larger contribution to PC2 in comparison with the other features.

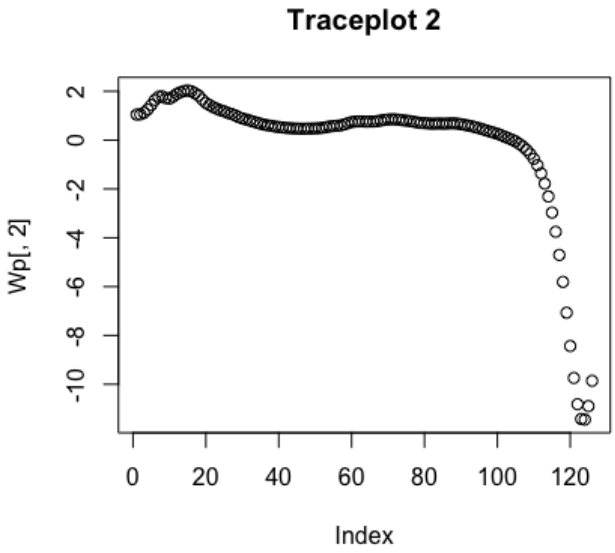


Fig. 16. Trace plot PC2 ICA

*C. 4.3*

The next step is to perform an Indenpendent Component Analysis with the number of components selected in the previous step(two principal components). The function *fastICA()* is used and then compute *W' = K * W* and the following trace plot are presented below:

The plot from the previous task *B. 4.2* and these ones looks similar, one can say the plots looks reversed or upside-down. *K* is a pre-whitening matrix where the data projects into *n* first principal components and *W* is the un-mixing matrix. This gives us *Wp* which essentially is a un-mixing matrix but the difference is that the data is projected onto the independentprinciple components. The score plot for the first two latent features is presented below:

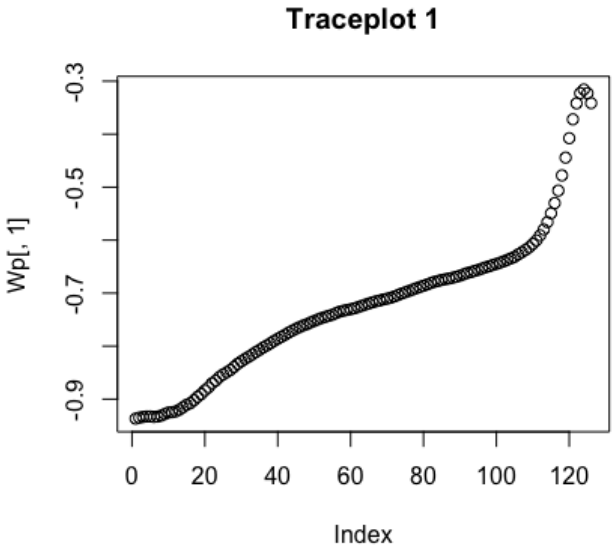


Fig. 15. Trace plot PC1 ICA

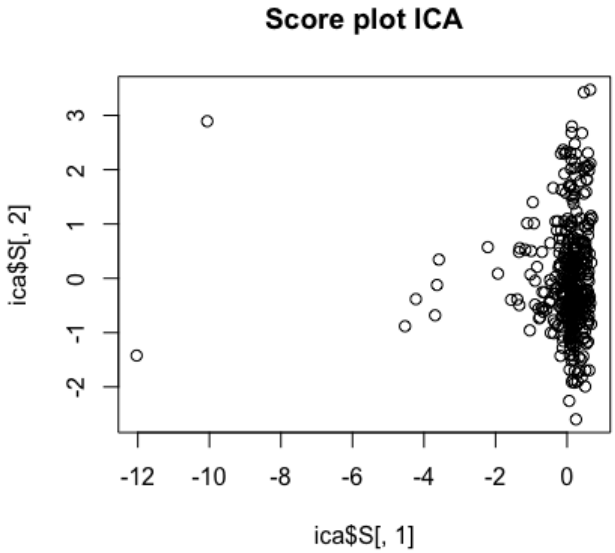


Fig. 17. Score plot ICA

This score plot compared to *Fig.12* is essentially

the same but reversed. This is expected since the trace plots were similar and reversed also.

*D. 4.4*

The last step is to fit a Principle Components Regression model in which number of components is selected by cross-validation. The plot of the dependence of the mean-square predicted error on the number of components is shown below:

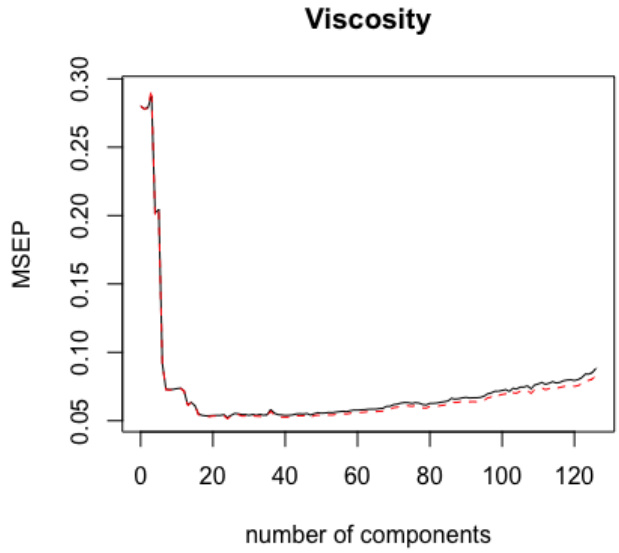


Fig. 18. Validation plot

From this plot we want to strive for a small mean-square predicted error. Therefore, the optimal number of components is 20.

# APPENDIX

Listing 1. Assignment 2

```
teactor_data = read.csv2("tecator.csv")
plot(teactor_data$Moisture, teactor_data$Protein, xlab = "Moisture", ylab = "Protein")

#Exercise 2

#Exercise 3
#Divide the data into training and validation set(50%/50%).
n=dim(teactor_data)[1]
id=sample(1:n, floor(n*0.5))
train=teactor_data[id,]
test=teactor_data[-id,]

#Fit model Mi, i = 1,...,6.
MSE = function(predict, obs){ #Prediction vector, Observation vector.
  return(mean((predict - obs)^2))
}

MSE_training = numeric(6)
MSE_test = numeric(6)
Moisture = teactor_data$Moisture
Protein = teactor_data$Protein
for(i in 1:6){
  model = lm(Moisture~poly(Protein,i), data = train)
  model_predict_train = predict.lm(model)
  model_predict_test = predict.lm(model, data = test)
  MSE_training[i] = MSE(unname(model_predict_train), train$Moisture)
  MSE_test[i] = MSE(unname(model_predict_test), test$Moisture)
}
x = seq(from = 1, to = 6)
plot(x, MSE_training, col = "red", type = "l")
par(new = TRUE)
plot(x, MSE_test, col = "blue", type = "l")

#Ex4
library(MASS)
lm_fat = lm(Fat~., data=teactor_data[, 2:102]) #First column and last columns are not features.
step = stepAIC(lm_fat)

#Ex5
library(glmnet)
covariates=scale(teactor_data[,2:101]) #Features
response=scale(teactor_data[, 102]) #Fat
model0=glmnet(as.matrix(covariates), response, alpha=0,family="gaussian")
plot(model0, xvar="lambda", label=TRUE)

#Ex6
lasso = glmnet(as.matrix(covariates), response, alpha=1,family="gaussian") #alpha = 1, gaussian
    => LASSO
plot(lasso, xvar="lambda", label=TRUE)

#Ex7
cv_lasso = cv.glmnet(as.matrix(covariates), response, alpha=1,family="gaussian")
cv_lasso$lambda.min #Get minimum lambda
plot(cv_lasso)
coef(cv_lasso, s="lambda.min")
```

Listing 2. Assignment 3

```
rm(list=ls())
library(tree)
library(boot)
set.seed(12345)
state_data = read.csv2("State.csv")

#Exercise 1.
state_data <- state_data[order(state_data$MET),] #Reorder data with respect to increase of "MET"
plot(state_data$EX, state_data$MET) #Plot EX versus MET

#Exercise 2.
#Appropriate model: Classification trees.
nr_obs = dim(state_data)[1]
tree_model <- tree(EX~MET, data = state_data, control = tree.control(nr_obs, minsize = 8)) #Fit
    model.
cv_model = cv.tree(tree_model) #Perform cross-validation.
plot(cv_model, type = "b")

#Selected tree from cross-validation.
bestNrLeafs = cv_model$size[cv_model$dev == min(cv_model$dev)]
selected_tree = prune.tree(tree_model, best = bestNrLeafs)
summary(selected_tree)
plot(selected_tree)
text(selected_tree, cex=.75)


#Plot the orignal data and the fitted data.
predicted_data = predict(selected_tree, state_data)
plot(state_data$MET, predicted_data, type = "b")

#Plot the histogram residuals.
resid_tree = resid(selected_tree) #state_data$EX - predicted_data
hist(resid_tree)
#Residual: Difference between data and fitted data.
#Comment: Would have been better if histogram resemble a normal distribution which would have
    been good. Our histogram
#is not good.

#Exercise 3.
library(boot)
#Get bootstrap samples.
f = function(state_data, index){
  data1 = state_data[index,] #Extract bootstrap samples.
  res_tree = tree(EX~MET, data = data1, control = tree.control(nr_obs, minsize = 8))
  pruned_tree = prune.tree(res_tree, bestNrLeafs)
  pred = predict(pruned_tree,newdata = state_data)
  return(pred)
}
res = boot(state_data, f, R = 1000)
confidence_band = envelope(res, level = 0.95)
plot(state_data$MET, state_data$EX, main = "Bootstrap confidence band", xlab = "MET", ylab = "
    EX", ylim = range(150,500))
points(state_data$MET, predicted_data, type = "l")
points(state_data$MET, confidence_band$point[2,], type = "l", col = "blue")
points(state_data$MET, confidence_band$point[1,], type = "l", col = "blue")
#The width of the confidence band increases at around MET ~ 20-30. This means our level of
    uncertainty increases
#when MET = 20-30.

#Exercise 4
mle = selected_tree

rng = function(state_data, mle){
  data1 = data.frame(EX = state_data$EX, MET = state_data$MET)
```

```
  n = length(state_data$EX)
  data1$EX = rnorm(n, predict(mle, newdata = data1), sd(resid(mle)))
  return(data1)
}

f2 = function(data1){
  res_tree = tree(EX~MET, data = data1, control = tree.control(nr_obs, minsize = 8))
  pruned_tree = prune.tree(res_tree, best = bestNrLeafs)
  pred = predict(res_tree,newdata = data1)
  return(pred)
}

res = boot(state_data, statistic = f2, R= 1000, mle = mle, ran.gen = rng, sim = "parametric")
confidence_band_par = envelope(res, level = 0.95)
points(state_data$MET, confidence_band_par$point[2,], type = "l", col = "red")
points(state_data$MET, confidence_band_par$point[1,], type = "l", col = "red")
points(state_data$MET, confidence_band_par$overall[2,], type = "l", col = "green") #Pred.band
points(state_data$MET, confidence_band_par$overall[1,], type = "l", col = "green") #Pred.band
legend("topright", lty = c(1,1), col = c("Blue", "Black","Red", "Green"), legend = c("Non-
    parametric", "Fitted Line", "Parametric", "Prediction-band"), cex = 0.50)
#Comment: The confidence band for the parametric bootstrap is less bumpy compared to the non-
    parametric bootstrap.
#The confidence bands are depended on the fitted model and the standard deviation and not the
    data values.


#Exercise 5
#Non-parametric bootstrap would be more appropriate in this case since parametric bootstrap
    assumes that we know the
#normal distribution is correct, which again, is not correct..
```

Listing 3. Assignment 4

```
rm(list=ls())
library(pls)
set.seed(12345)
spectra_data = read.csv2("NIRSpectra.csv")
data1 = spectra_data
data1$Viscosity = c()
pca_data = prcomp(data1)
lambda = pca_data$sdev^2
#Eigenvalues
print(lambda)
#Proportion of variation
sprintf("%2.3f", lambda/sum(lambda)*100)
screeplot(pca_data)
plot(pca_data$x[,1], pca_data$x[,2])
#There are unusual diesel fuels according to the plot that deviates from the PC axis with
    similar viscocity.

#Exercise 2
U = pca_data$rotation
plot(U[,1], main = "Traceplot, PC1")
plot(U[,2], main = "Traceplot, PC2", ylim = c(-0.1,0.5))
# PC2 is explained by mainly a few original features.

#Exercise 3
library(fastICA)
ica = fastICA(data1,2,alg.typ = "parallel", fun = "logcosh", alpha = 1, method = "R", row.norm
    = FALSE, maxit = 200,tol = 0.0001, verbose = TRUE)
Wp = ica$K %*% ica$W #K is prewhitening matrix, W is estimated matrix.
plot(Wp[,1], main = "Traceplot 1")
plot(Wp[,2], main = "Traceplot 2")
#Wp measures

plot(ica$S[,1], ica$S[,2], main = "Score plot ICA")


#Exercise 4
visc_pcr <- pcr(Viscosity~., data = spectra_data, validation = "CV")
validationplot(visc_pcr, val.type = "MSEP")
```