Computer lab 2 block 1

Laura Julià Melis

12/08/2019

Assignment 2. Analysis of credit scoring.

The data file creditscoring.xls contains data retrieved from a database in a private enterprise. Each row contains information about one customer. The variable good/bad indicates how the customers have managed their loans. The other features are potential predictors. Your 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.

1. Import the data to R and divide into training/validation/test as 50/25/25: use data partitioning code specified in Lecture 1e.

See Appendix

2. Fit a decision tree to the training data by using the following measures of impurity (a. Deviance, b. Gini index) and report the misclassification rates for the training and test data. Choose the measure providing the better results for the following steps.

We have fitted two decision trees using the function `tree()` from the package **tree**, choosing `split = "deviance"` and `split = "gini"` in each one.

- Missclassification rates for the fitted tree using the deviance:

```
##   Test.data Training.data
## 1     0.268         0.212
```

- Missclassification rates for the fitted tree using the Gini index:

```
##   Test.data Training.data
## 1     0.368          0.24
```

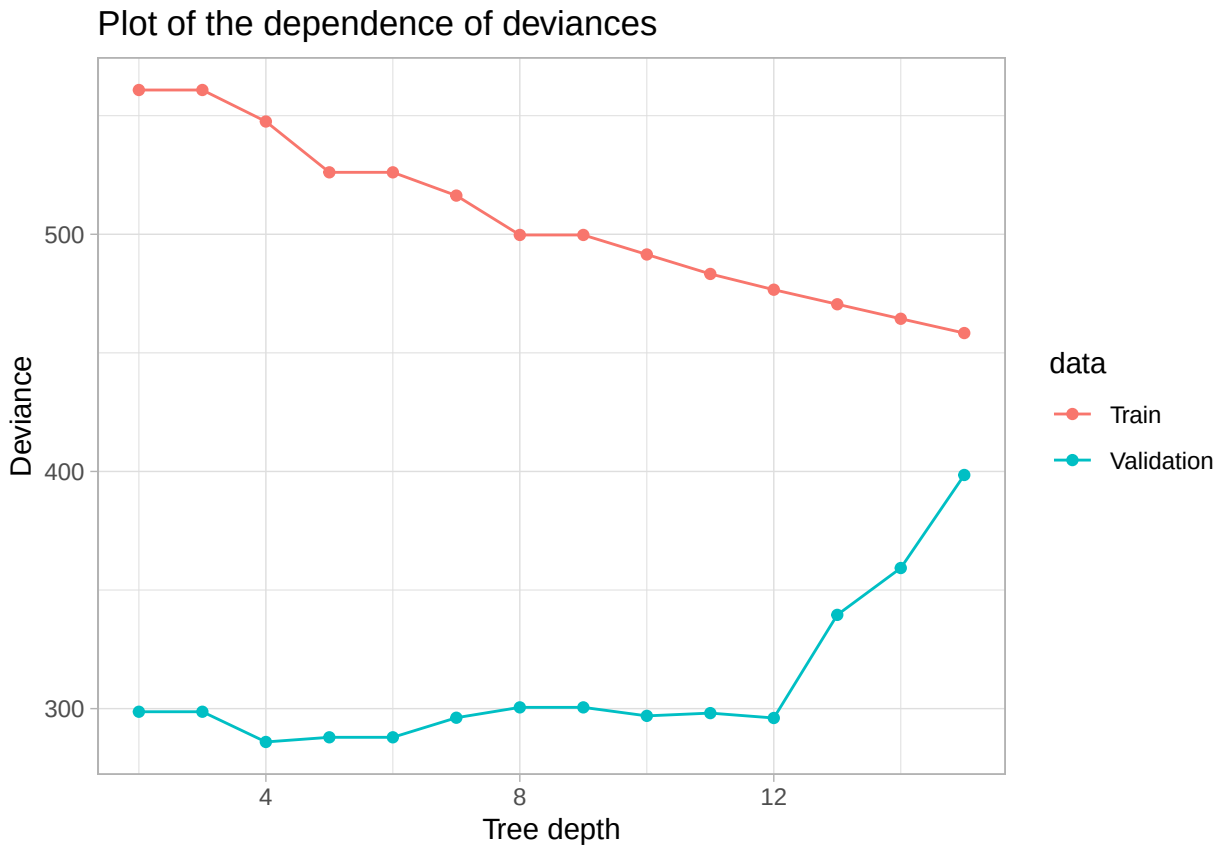
Given the missclassification rates, the best performance is the one obtained by the tree fitted with the deviance as measure of impurity. For both trees the rate for the training data is quite similar but the error rate for the test data is smaller in the deviance case.

For this reason, from now on we will use the tree with `deviance`.

3. Use training and validation sets to choose the optimal tree depth. Present the graphs of the dependence of deviances for the training and the validation data on the number of leaves. Report the optimal tree, report it's depth and the variables used by the tree. Interpret the information provided by the tree structure. Estimate the misclassification rate for the test data.

In `summary(fit)` we have obtained that the number of terminal nodes is 15, so we have used the `prune.tree()` function with the fitted model (using the deviance) in the **tree** argument and values $2, 3, \ldots, 15$ in the **best** argument. So, 14 different trees with $2, 3, \ldots, 15$ terminal nodes in each one has been considered and the deviance of each model has been calculated with the `deviance()` function.

Then, the results obtained are shown in the plot below:

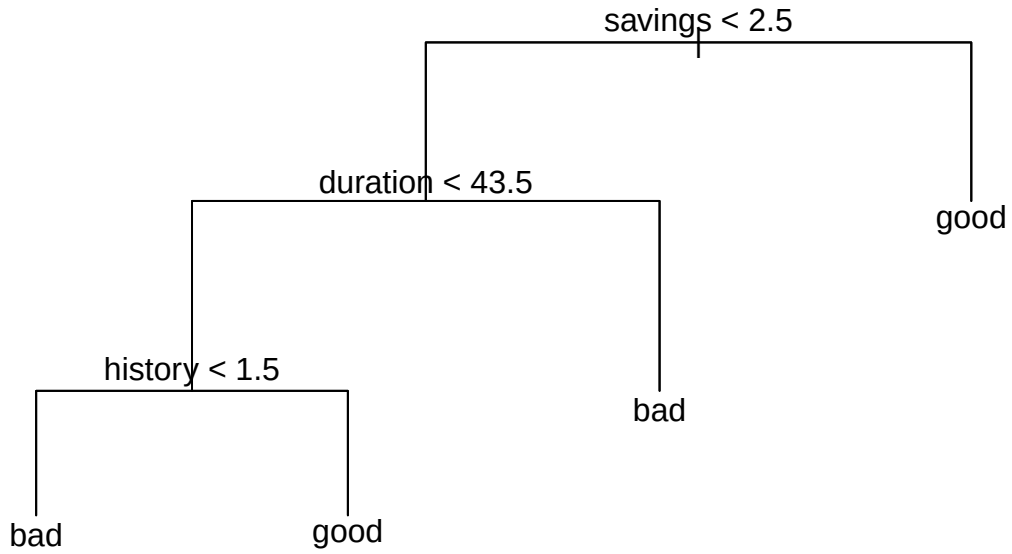


We observe that in the validation data, the smallest deviance is obtained when the tree depth is 4 (4 terminal nodes). We can also confirm that with the following command:

```
which(testScore==min(testScore[2:12]))
```

```
## [1] 4
```

We can draw the optimal tree with the `plot()` function



This three has 4 terminal nodes and the variables used are: duration, history and savings.

Finally, we will estimate the misclassification rate for the test data:

```
## [1] 0.256
```

We have reduced the misclassification rate (compared to the tree chosen in question 2). Now, only 25.6% of the observations are classificated wrongly.

4. Use training data to perform classification using Naïve Bayes and report the confusion matrices and misclassification rates for the training and for the test data. Compare the results with those from step 3.

We have fitted the required model using the training data and the function `naiveBayes()` from the package **e1071**.

- Confusion matrices.

```
##       fitted_test
##        bad good
##   bad   46   30
##   good  49  125
```

```
##       fitted_train
##        bad good
##   bad   95   52
##   good  98  255
```

- Misclassification rates.

```
## For the test data: 0.316
```

```
## For the training data: 0.3
```

The misclassification rate for the test data is 0.06 units larger for the Naïve Bayes model than the decision tree with 4 terminal nodes.

5. Use the optimal tree and the Naïve Bayes model to classify the test data by using the following principle:

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

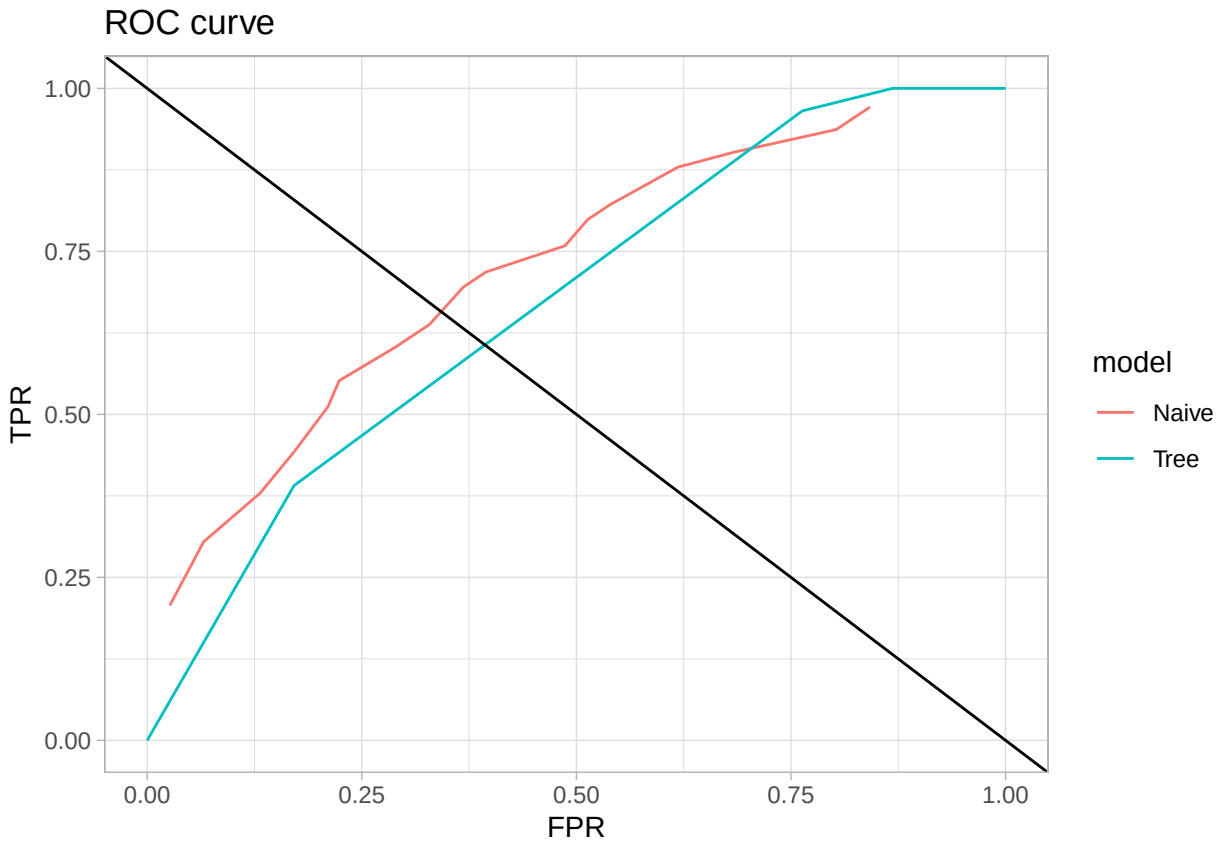
where $\pi = 0.05, 0.1, 0.15, ...0.9, 0.95$. Compute the TPR and FPR values for the two models and plot the corresponding ROC curves. Conclusion?

- Table TPR and FPR values:

```
##      pi model    TPR    FPR   pi model    TPR    FPR
## 1  0.05 Naive 0.9713 0.8421 0.05  Tree 1.0000 1.0000
## 2  0.10 Naive 0.9368 0.8026 0.10  Tree 1.0000 1.0000
## 3  0.15 Naive 0.9253 0.7632 0.15  Tree 1.0000 1.0000
## 4  0.20 Naive 0.9023 0.6842 0.20  Tree 1.0000 0.8684
## 5  0.25 Naive 0.8793 0.6184 0.25  Tree 1.0000 0.8684
## 6  0.30 Naive 0.8218 0.5395 0.30  Tree 1.0000 0.8684
## 7  0.35 Naive 0.7989 0.5132 0.35  Tree 0.9655 0.7632
## 8  0.40 Naive 0.7586 0.4868 0.40  Tree 0.9655 0.7632
## 9  0.45 Naive 0.7356 0.4342 0.45  Tree 0.9655 0.7632
## 10 0.50 Naive 0.7184 0.3947 0.50  Tree 0.9655 0.7632
## 11 0.55 Naive 0.6954 0.3684 0.55  Tree 0.9655 0.7632
## 12 0.60 Naive 0.6379 0.3289 0.60  Tree 0.9655 0.7632
## 13 0.65 Naive 0.6034 0.2895 0.65  Tree 0.9655 0.7632
## 14 0.70 Naive 0.5517 0.2237 0.70  Tree 0.9655 0.7632
## 15 0.75 Naive 0.5115 0.2105 0.75  Tree 0.3908 0.1711
## 16 0.80 Naive 0.4425 0.1711 0.80  Tree 0.3908 0.1711
```

```
## 17 0.85 Naive 0.3793 0.1316 0.85  Tree 0.0000 0.0000
## 18 0.90 Naive 0.3046 0.0658 0.90  Tree 0.0000 0.0000
## 19 0.95 Naive 0.2069 0.0263 0.95  Tree 0.0000 0.0000
```

- Plot of FPR vs TPR (ROC curve).



How to interpret the ROC CURVE?: Same FPR, higher TPR $\rightarrow$ better classifier.

So we can observe that, for almost al the curve, the Naive Bayes model has a higher TPR than the Tree model.

6. Repeat Naïve Bayes classification as it was in step 4 but use the following loss matrix:

$$L = \begin{pmatrix} 0 & 1 \\ 10 & 0 \end{pmatrix}$$

and report the confusion matrix for the training and test data. Compare the results with the results from step 4 and discuss how the rates has changed and why.

Rule to classify an observation:

$$\frac{P(C_1|x)}{P(C_2|x)} > \frac{L_{21}}{L_{12}} \rightarrow \text{predict } y \text{ as } C_1$$

In our case:

$$\frac{P(\text{“good"}|x)}{P(\text{“bad"}|x)} > 10 \rightarrow \text{predict } y \text{ as “good"}$$

- Confusion matrices.

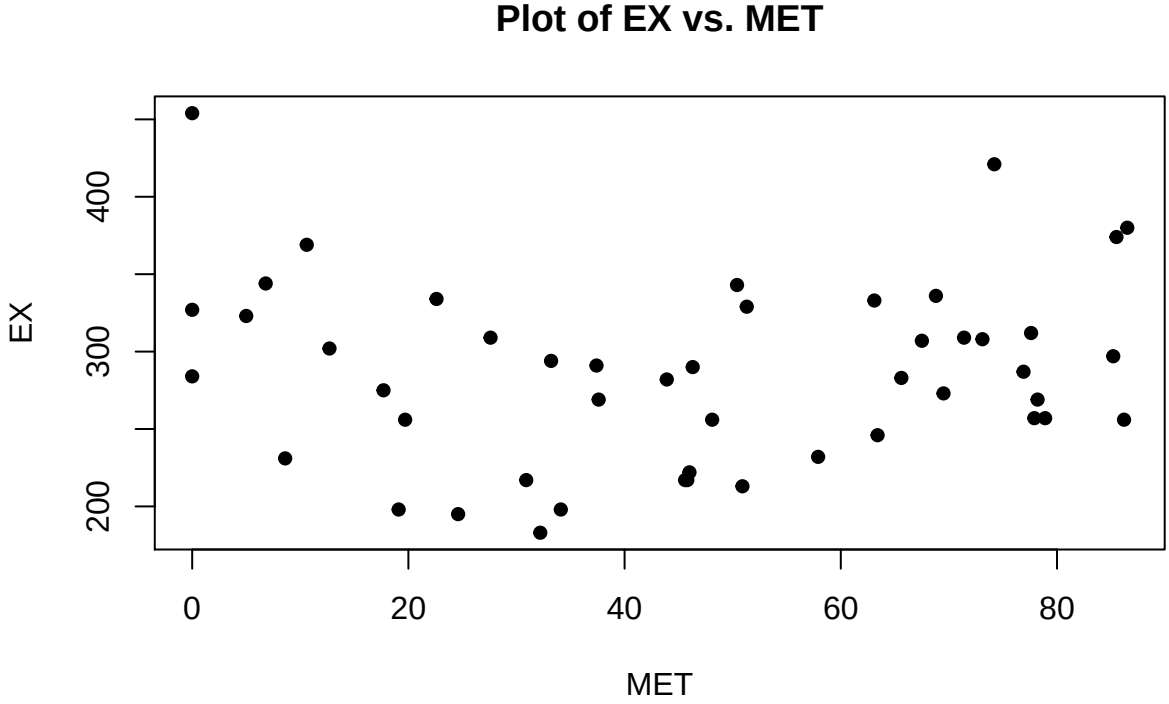
```
##       fitted_class_test
##        bad good
##   bad   71    5
##   good 122   52
```

```
##       fitted_class_train
##        bad good
##   bad  137   10
##   good 263   90
```

In this case, the number of good customers predicted as good has been reduced in comparison to the confusion matrices in question (4) which happened because when using the loss matrix, our rule to classify a customer as "good" is "the propability to classify a customer as good has to be 10 times higher than the probability of being bad". So we are being more strict.

Assignment 3. Uncertainty estimation.

1. Reorder your data with respect to the increase of MET and plot EX versus MET. Discuss what kind of model can be appropriate here. Use the reordered data in steps 2-5.



The observations (black dots) are quite sparsed in the plot, so a linear regression model wouldn't be appropiate. Maybe, the best option is to fit a regression tree, because the target variable is numeric.

2. Use package tree and fit a regression tree model with target EX and feature MET in which the number of the leaves is selected by cross-validation, use the entire data set and set minimum number of observations in a leaf equal to 8 (setting minsize in tree.control). Report the selected tree. Plot the original and the fitted data and histogram of residuals. Comment on the distribution of the residuals and the quality of the fit.

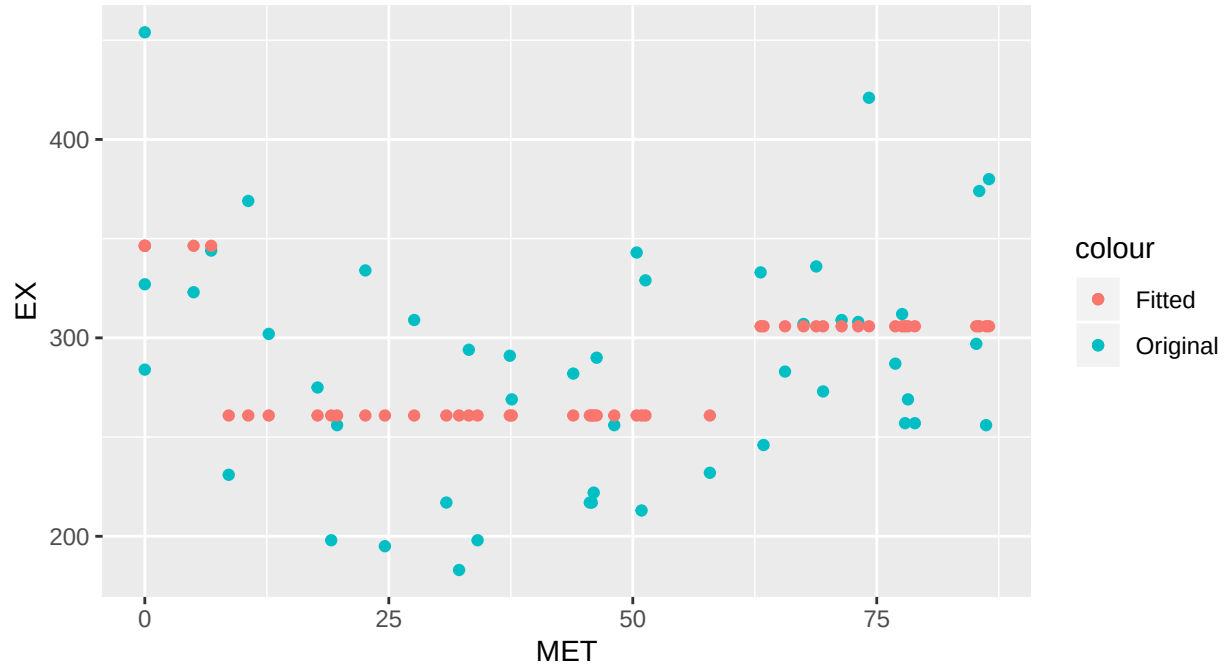
In order to fit the regression tree with at least 8 observations in each leaf, it has been used the option `minsize` in the control argument. Then. the `cv.tree` function has been used to selet the optimal number of leaves by cross-validation. This value can be seen executing the following code: `cv.tree$size[which(cv.tree$dev==min(cv.tree$dev))]`.

So the optimal number of leaves is 3 and we can report this tree:

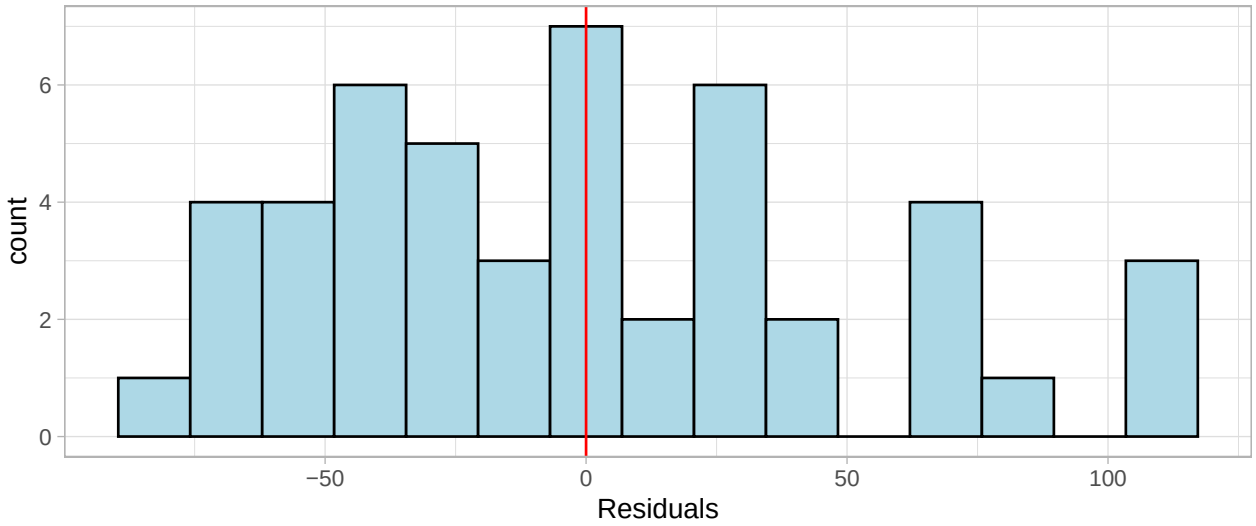
```
## node), split, n, deviance, yval
##       * denotes terminal node
##
```

```
## 1) root 48 162500 286.6
##   2) MET < 7.7 5  16400 346.4 *
##   3) MET > 7.7 43 126100 279.7
##     6) MET < 60.5 25  65660 260.9 *
##     7) MET > 60.5 18  39330 305.8 *
```

We will plot the original and the fitted data:



The histogram of the residuals is the following:



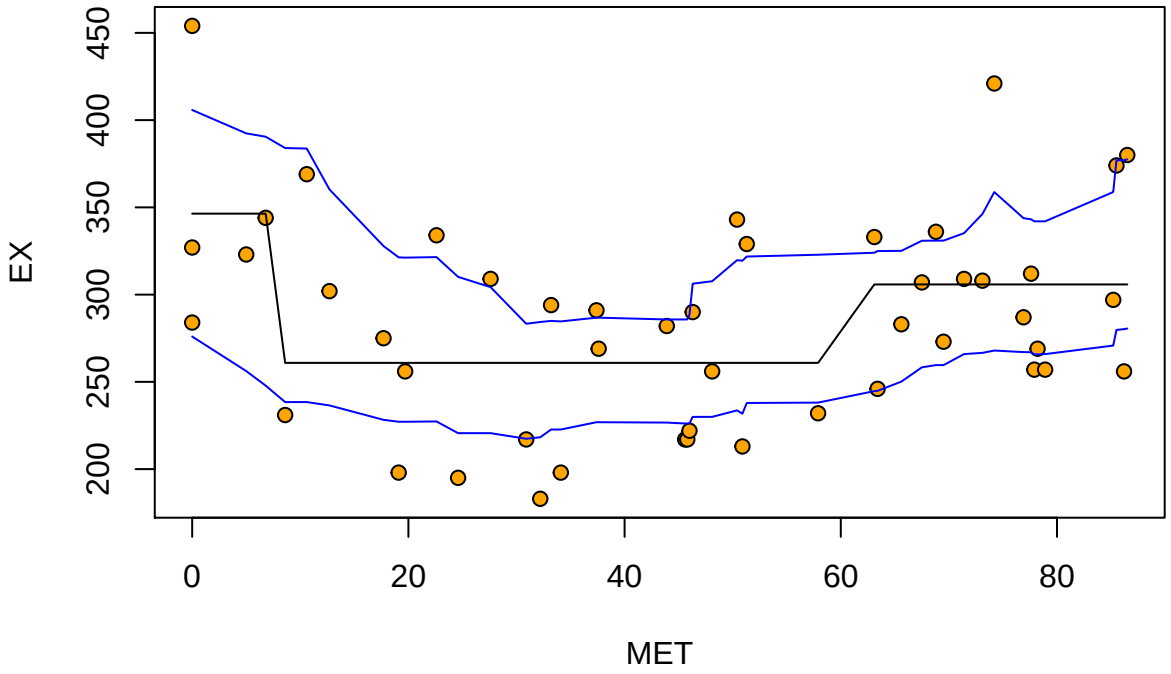
We observe from the histogram that the residuals don't follow a normal distribution.

3. Compute and plot the 95% confidence bands for the regression tree model from step 2 (fit a regression tree with the same settings and the same number of leaves as in step 2 to the resampled data) by using a non-parametric bootstrap. Comment whether the band is smooth or bumpy and try to explain why. Consider the width of the confidence band and comment whether results of the regression model in step 2 seem to be reliable.

We have used the `boot()` function in order to generate 1000 bootstrap replicates. The statistic of interest in our case is the vector of predictions obtained in each regression tree (fitted with all the bootstrap sample).

Once we have had the 1000 vectors of predictions the `envelope()` function has been used to compute the confidence bands.

Then we can plot the observations with the estimate line (in black) and the confidence intervals (blue lines):



We observe that the bands' shapes are bumpy: the lines are rising and falling irregularly. This might happen because the data does not follow a clear distribution and does not have an evident trend.

Also, the width of the confidence band is not including almost all the points and we expect to have 95% of the points inside. More than 5% of the points are outisde, so we can conclude that the regression tree model with 3 leaves considered as optimal in the step (2) is a not a very good model for this data.

The width of the confidence band is wide because we don't have many observations (only 48). If we had considered more data points, the confidence bands would be near the curve (fitted black line) so most of the orange dots would be outside the confidence bands.

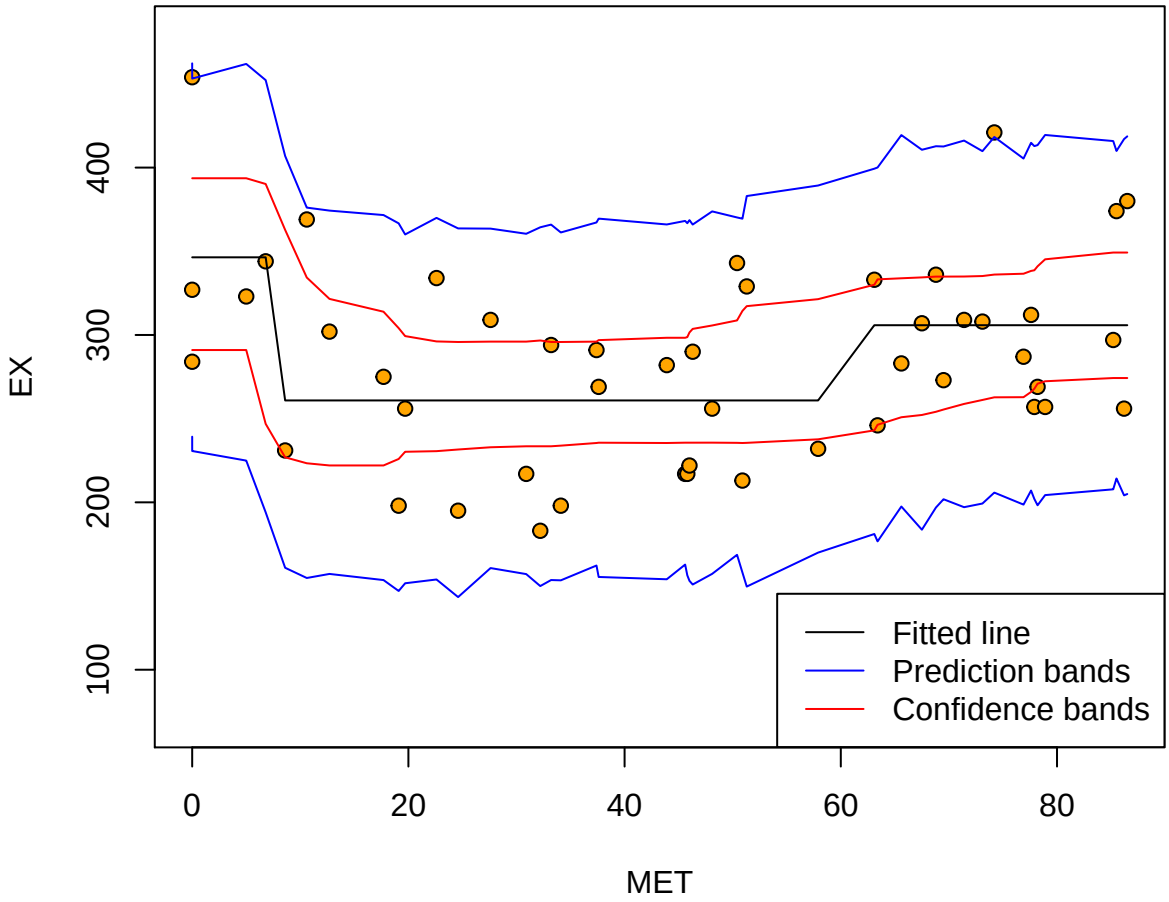
4. Compute and plot the 95% confidence and prediction bands the regression tree model from step 2 (fit a regression tree with the same settings and the same number of leaves as in step 2 to the resampled data) by using a parametric bootstrap, assume $Y \sim N(\mu_i, \sigma^2)$ where μ_i are labels in the tree leaves and σ^2 is the residual variance. Consider the width of the confidence band and comment whether results of the regression model in step 2 seem to be reliable. Does it look like only 5% of data are outside the prediction band? Should it be?

Steps followed to compute parametric bootstrap confidence and prediction bands:

1. Compute value MLE that estimates model parameters from the data: the maximum likelihoood estimates of the parameters in our case are computed with the function `prune.tree()` because our tree regression model is pruned with only 5 leaves.
2. Write function "ran.gen" that depends on data and MLE and which generates new data. This function returns new values for the target variable "EX", generated from a normal distribution of μ = predictions of MLE and σ = residuals of MLE.

3. Write function "statistic" that depend on data which will be generated by ran.gen and should return the estimator. In this case two different statistics have been computed: prediction and confidence.
4. Make bootstrap for each statistic. We have used the `boot()`function for this in order to generate 1000 parametric bootstrap replicates of each statistic.
5. Compute the confidence and prediction bands with `envelope()`.

So, the plot of the 95% confidence and prediction bands:

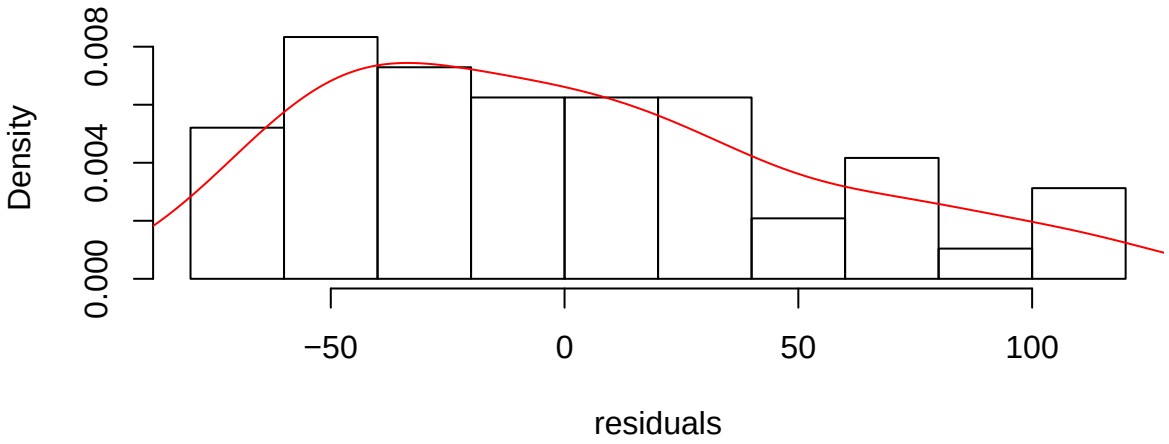


From the plot we observe that the prediction bands are quite wider than the confidence bands and that almost all the points are inside these bands. This is totally normal because the 95% confidence bands enclose the area that contains the true curve (at 95% confidence) while the 95% prediction bands takes in the area that one can **expect** to enclose 95% of future data points.

Also, we can see that there are only 2 points that are outside the prediction bands (given that we have 48 points, that is the 2.4%), this is less than the 5% and it is expected to be like this because, assuming that Y follows a normal distribution, only a maximum of 5% of new points can be outside.

5. Consider the histogram of residuals from step 2 and suggest what kind of bootstrap is actually more appropriate here.

If we draw the density curve in the histogram of the residuals, we obtain the following plot:



Which makes us think that a Gamma distribution could be appropiate in this case.

Assignment 4. Principal components.

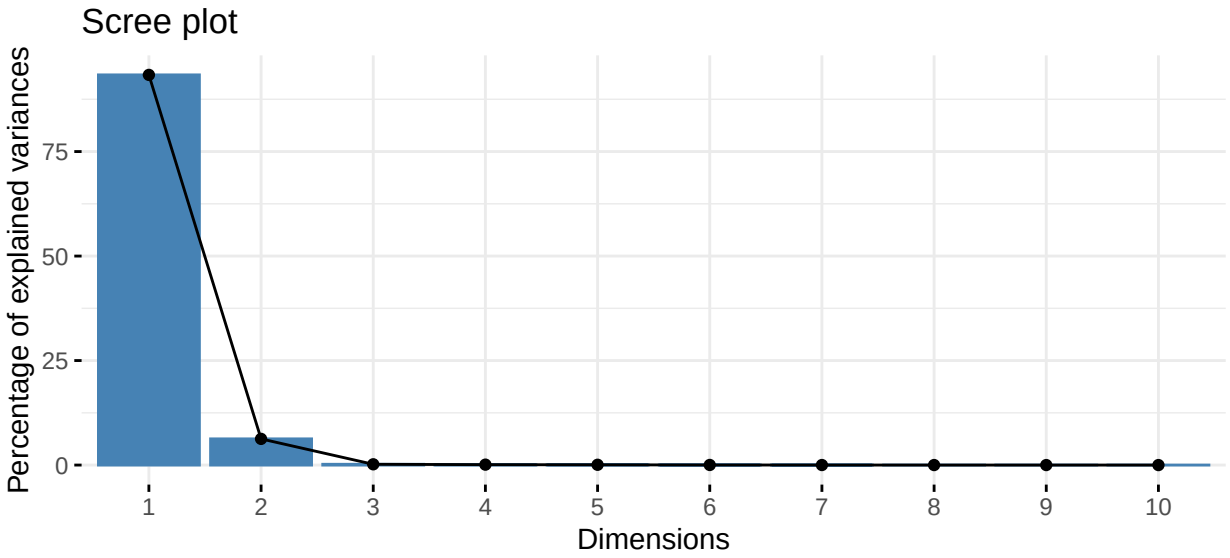
1. Conduct a standard PCA by using the feature space and provide a plot explaining how much variation is explained by each feature. Does the plot show how many PC should be extracted? Select the minimal number of components explaining at least 99% of the total variance. Provide also a plot of the scores in the coordinates (PC1, PC2). Are there unusual diesel fuels according to this plot?

The Principal Component Analysis for all the different near-infrared spectra levels (variables 750 to 1000 from the original data frame) has been performed with the `prcomp()` function. In the following table it is possible to see how much variation is explained by each of the first five principal components as well as the cumulative proportion of variance.

```
##                            PC1     PC2     PC3     PC4     PC5
## Standard deviation     0.12206 0.03162 0.00544 0.00401 0.00330
## Proportion of Variance 0.93332 0.06263 0.00185 0.00101 0.00068
## Cumulative Proportion  0.93332 0.99596 0.99781 0.99882 0.99950
```

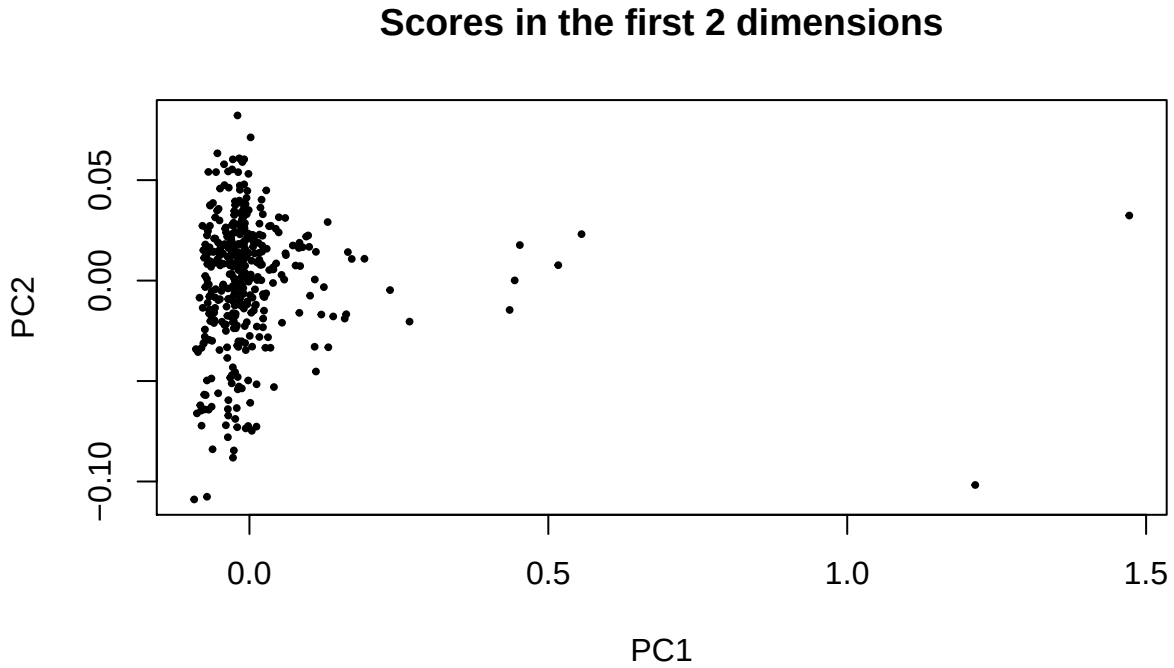
From the table we can observe that the first two principal components explain 99.6% of the total variance.

Also, we will plot the percentage of explained variation for each dimension (PC1 to PC10) with the function`fviz_eig()` from the **factoextra** package:



Again, we see that the first principal component explains more than 90% of the variance. Also, it is made evident that we should only extract 2 PC's in order to explain almost all the variation in our data (nearly without losing information).

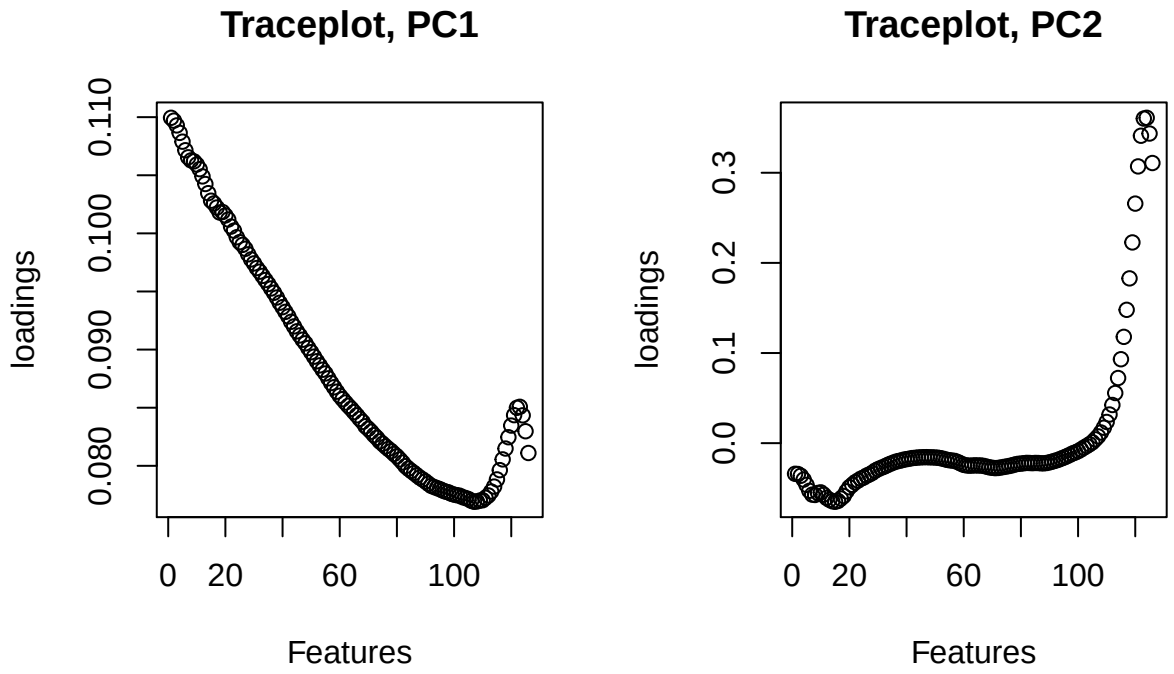
Finally, we will plot the scores in in the coordinates (PC1, PC2).



According to this plot there are unusual diesel fuels (outliers), those with high values in the x axis (PC1). More or less we can say that the unusual diesel fuels have a PC1 score greater than 0.4.

2. Make trace plots of the loadings of the components selected in step 1. Is there any principle component that is explained by mainly a few original features?

Since in question (1) we selected PC1 and PC2, now we will make a trace plot for each one:




```
## Variables with higest loadings in PC1:

##      X750      X752      X754      X756      X758      X760
## 0.1099439 0.1096993 0.1092886 0.1086458 0.1078912 0.1071514

## Variables with higest loadings in PC2:

##      X996      X994      X998      X992     X1000      X990
## 0.3609749 0.3601228 0.3435218 0.3410492 0.3108059 0.3070516
```

However, the variables with highest loadings don't differ too much from the rest (as observed in the plots), so we can't choose only a few original features that explain PC1 or PC2.

3. Perform Independent Component Analysis with the number of components selected in step 1 (set seed 12345). Check the documentation for the fastICA method in R and do the following:

a. Compute $W' = K \cdot W$ and present the columns of W' in form of the trace plots. Compare with the trace plots in step 2 and make conclusions. What kind of measure is represented by the matrix W'?

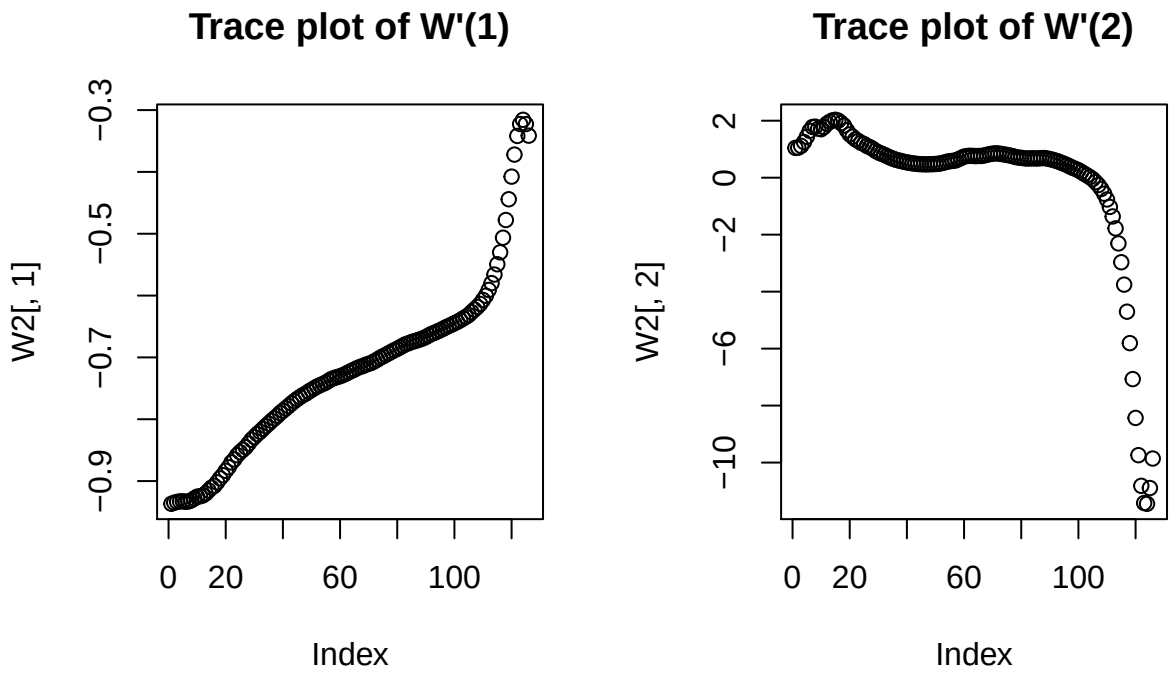
Matrix definitions, obtained from `fastICA()` function:

```
K: pre-whitening matrix that projects data onto the first n.comp principal components.
W: estimated un-mixing matrix that maximizes the non-gaussianity of the sources.
```

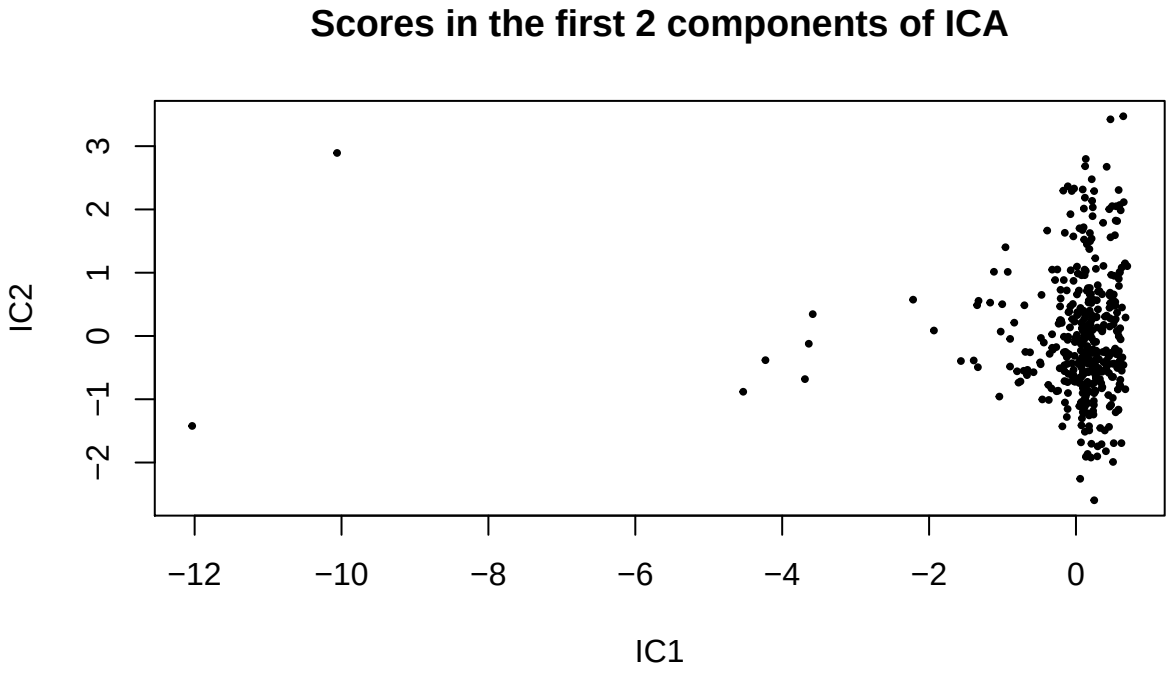
We use `fastica()` with the data frame containung the variables 750 to 1000 from the original data set (*NIRspectra*) and indicating in the **n.comp** argument that the number of components to be extracted is 2.

So, the output W matrix is a 2x2 matrix and K has 126 rows and 2 columns. Then, when computing $W' = K \cdot W$ we only need to use the command "%*%" and the obtained matrix (W') will have 126x2 dimensions.

We will plot the W' values in each principal component:



b. Make a plot of the scores of the first two latent features and compare it with the score plot from step 1.



Appendix

Assignment 2. Analysis of credit scoring.

Question 1.

```
## ----- ASSIGNMENT 2 ----- ##
# Changing the version:
RNGversion('3.5.1')
```

```r
# Importing the data:
library(readxl)
data <- read_excel("creditscoring.xls")
data$good_bad <- as.factor(data$good_bad)

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

id1=setdiff(1:n, id)
set.seed(12345)
id2=sample(id1, floor(n*0.25))
valid=data[id2,]

id3=setdiff(id1,id2)
test=data[id3,]
```

Question 2.

```r
library(tree)

# 2.1. Fitting the models.
fit_dev = tree(good_bad~., data=train, split = "deviance") # Impurity measure: Deviance
fit_gini = tree(good_bad~., data=train, split = "gini") # Impurity measure: Gini index

# 2.2. Predictions.
## For the test data:
fitted_dev_test <- predict(fit_dev, newdata = test, type = "class")
fitted_gini_test <- predict(fit_gini, newdata = test, type = "class")

## For the training data:
fitted_dev_train <- predict(fit_dev, newdata = train, type = "class")
fitted_gini_train <- predict(fit_gini, newdata = train, type = "class")

# 2.3. Misclassification rates.
## For the test data:
mis_rate_dev_test <- mean(fitted_dev_test != test$good_bad)
mis_rate_gini_test <- mean(fitted_gini_test != test$good_bad)

## For the training data:
mis_rate_dev_train <- mean(fitted_dev_train != train$good_bad)
mis_rate_gini_train <- mean(fitted_gini_train != train$good_bad)

# Results:
data.frame(cbind("Test data"=mis_rate_dev_test,"Training data"=mis_rate_dev_train))
data.frame(cbind("Test data"=mis_rate_gini_test,"Training data"=mis_rate_gini_train))
```

Question 3.

```r
library(ggplot2)

# 3.1. Fitting the model
```

```r
fit = tree(good_bad~., data=train, split = "deviance")
# summary(fit) -> Number of terminal nodes is 12

# 3.2. Pruning trees with difference depths
trainScore=rep(0,15)
testScore=rep(0,15)

for(i in 2:15) {
  prunedTree=prune.tree(fit,best=i)
  pred=predict(prunedTree, newdata=valid, type="tree")

  trainScore[i]=deviance(prunedTree)
  testScore[i]=deviance(pred)
}

# 3.3. Plot of deviance vs. depth
df <- data.frame(depth=2:15, score= c(trainScore[2:15],testScore[2:15]),
                 data= rep(c("Train", "Validation"), each=14))

ggplot(df, aes(x = depth, y = score, group = data, color = data)) +
  geom_point() + geom_line() + ggtitle("Plot of the dependence of deviances") +
  xlab("Tree depth") + ylab("Deviance") + theme_light()

# Smallest deviance
which(testScore==min(testScore[2:15]))

# 3.4. Optimal tree.
finalTree=prune.tree(fit, best=4)
{plot(finalTree)
text(finalTree, pretty=0)}

# 3.5. Misclassification rate for the test data.
fitted_best_test=predict(finalTree, newdata=test, type="class") # predictions
# table(fitted_best_test, test$good_bad) confussion matrix
mean(fitted_best_test != test$good_bad) # error rate
```

Question 4.

```r
library(e1071)

# 4.1. Fitting the model.
fit_naive=naiveBayes(good_bad~., data=train)

# 4.2. Predictions.
fitted_test <- predict(fit_naive, newdata = test, type = "class") # For the test data
fitted_train <- predict(fit_naive, newdata = train, type = "class") # For the training data

# 4.3. Confusion matrices.
table(test$good_bad, fitted_test)
table(train$good_bad, fitted_train)

# 4.4. Misclassification rates.
cat("For the test data:", mean(fitted_test != test$good_bad),"\n")
cat("For the training data:", mean(fitted_train != train$good_bad),"\n")
```

Question 5.

```
# 5.1. Classification models:
## Naive Bayes model
fit_naive=naiveBayes(good_bad~., data=train)

## Optimal tree (4 terminal nodes)
fit = tree(good_bad~., data=train, split = "deviance")
fit_optimal=prune.tree(fit, best=4)

# 5.2. Predictions:
## Naive Bayes
fitted_naive <- predict(fit_naive, newdata = test, type="raw") # type raw to get probabilities

## Optimal tree
fitted_tree <- predict(fit_optimal, newdata = test)

# 5.3. Classification using the principle.
pi <- seq(0.05, 0.95, by=0.05)

fitted_naive_class <- data.frame(matrix(ncol = length(pi), nrow = nrow(test)))
fitted_tree_class <- data.frame(matrix(ncol = length(pi), nrow = nrow(test)))
colnames(fitted_naive_class) <- pi
colnames(fitted_tree_class) <- pi

for(i in 1:length(pi)){
  fitted_naive_class[,i] <- ifelse(fitted_naive[,2] > pi[i], 1, 0)
  fitted_tree_class[,i] <- ifelse(fitted_tree[,2] > pi[i], 1, 0)
}

# 5.4. Computation of TPR and FPR: TPR=TP/(TP+FN) and FPR=FP/(FP+TN)
library(EvaluationMeasures)
real_values <- ifelse( test$good_bad == "good", 1, 0)

result <- data.frame(matrix(ncol = 4, nrow = length(pi)*2))
colnames(result) <- c("pi", "model", "TPR", "FPR")
result$pi <- rep(pi,2)
result$model <- rep(c("Naive", "Tree"),each=length(pi))

for(i in 1:ncol(fitted_naive_class)){
  result[i,3] <- EvaluationMeasures.TPR(Real= real_values, Predicted = fitted_naive_class[,i],
                                        Positive = 1)
  result[i+19,3] <- EvaluationMeasures.TPR(Real= real_values, Predicted = fitted_tree_class[,i],
                                           Positive = 1)

  result[i,4] <- EvaluationMeasures.FPR(Real= real_values, Predicted = fitted_naive_class[,i],
                                        Positive = 1)
  result[i+19,4] <- EvaluationMeasures.FPR(Real= real_values, Predicted = fitted_tree_class[,i],
                                           Positive = 1)
}

#  Table TPR and FPR values:
cbind(result[1:19,], result[20:38,])
```

```r
# 5.5. ROC curve.
ggplot(result, aes(x = FPR, y=TPR)) + geom_line(aes(colour = model)) +
  geom_abline(intercept=1, slope =-1) + xlim(0,1)+
  ggtitle("ROC curve") + theme_light()
```

Question 6.

```r
# 6.1. Naive Bayes model.
fit =naiveBayes(good_bad~., data=train)

# 6.2. Predictions
# Probabilities (type= "raw")
fitted_prob_test <- predict(fit_naive, newdata = test, type="raw") # test data
fitted_prob_train <- predict(fit_naive, newdata = train, type="raw") # training data

# 6.3. Classification using the loss matrix.
fitted_class_test <- ifelse(fitted_prob_test[,2]/fitted_prob_test[,1] > 10, "good", "bad")
fitted_class_train <- ifelse(fitted_prob_train[,2]/fitted_prob_train[,1] > 10, "good", "bad")

# 6.4. Confusion matrices.
table(test$good_bad, fitted_class_test)
table(train$good_bad, fitted_class_train)
```

Assignment 3. Uncertainty estimation.

Question 1.

```r
## ----- ASSIGNMENT 3 ----- ##
# Importing the data:
data <- read.table("State.csv", header=TRUE, sep=";", dec=",")

# Data reordered by MET
data <- data[order(data$MET),]

# Plot
plot(data$MET, data$EX, xlab="MET", ylab="EX", main="Plot of EX vs. MET", pch=19, cex=0.85)
```

Question 2.

```r
# 2.1. Fitting a regression tree model.
fit <- tree(EX~ MET , data=data, control=tree.control(nobs= nrow(data), minsize = 8))

# 2.2. Selecting the optimal number of leaves by cross-validation.
set.seed(12345)
cv.tree=cv.tree(fit)

# 2.3. Fitting the optimal regression tree.
fit_optimal <- prune.tree(fit, best=cv.tree$size[which(cv.tree$dev==min(cv.tree$dev))])
fit_optimal

# 2.4. Plot of the original and the fitted data.
ggplot()+ geom_point(mapping = aes(x=data$MET,y=data$EX,color='Original'))+ geom_point(mapping = aes(x=

# 2.5 Histogram of residuals.
pred <- predict(fit_optimal, newdata = data)
resid <- data$EX-pred
```

```r
df <- as.data.frame(resid)

ggplot(df, aes(x=resid)) + geom_histogram(color="black", fill="lightblue", bins=15)+
  theme_light() + xlab("Residuals")+ geom_vline(xintercept = 0, col="red")
```

Question 3.

```r
# 3. Non-parametric bootstrap CI
library(boot)

# 3.1. First we need to compute bootstrap samples
f=function(data, ind){
  # extract bootstrap sample
  data1 = data[ind,]

  # fit regression tree model
  fit = tree(EX~ MET , data=data1, control=tree.control(nobs= nrow(data1), minsize = 8))
  fit_optimal <- prune.tree(fit, best=3)

  # predict values from the original data
  predictions = predict(fit_optimal, newdata=data)

  return(predictions)
}

# 3.2. And now we can make bootstrap:
res=boot(data, f, R=1000)

# 3.3. Bootstrap cofidence bands
e=envelope(res) #compute confidence bands

# 3.4. Plotting the predicted line and 95% confidence bands
fit <- tree(EX~ MET , data=data, control=tree.control(nobs= nrow(data), minsize = 8))
cv.tree=cv.tree(fit)
fit_optimal <- prune.tree(fit, best=3)

predictions =predict(fit_optimal)

{plot(data$MET, data$EX, xlab="MET", ylab="EX", pch=21, bg="orange")
points(data$MET,predictions,type="l") #plot fitted line
#plot cofidence bands
points(data$MET,e$point[2,], type="l", col="blue")
points(data$MET,e$point[1,], type="l", col="blue")}
```

Question 4.

```r
# 4. Parametric bootstrap confidence and prediction bands
# 4.1. Compute value mle that estimates model parameters from the data:
fit <- tree(EX~ MET , data=data, control=tree.control(nobs= nrow(data), minsize = 8))
cv.tree=cv.tree(fit)
mle <- prune.tree(fit, best=3)

# 4.2. Write function ran.gen that depends on data and mle and which generates new data.
rng=function(data, mle) {
  data1=data.frame(Ex=data$EX, MET=data$MET)
```

```
  n=length(data$EX)

  #generate new EX (mle$residuals doesn't exist for prune.tree)
  data1$EX=rnorm(n,predict(mle, newdata=data1),sd(residuals(mle)))
  return(data1)
}

# 4.3. Write function "statistic" that depend on data which will be generated by ran.gen
#      and should return the estimator.
# 4.3.1. Bootstrap samples for prediction bands
f_pred=function(data1){
  #fit linear model
  fit = tree(EX~ MET , data=data1, control=tree.control(nobs= nrow(data1), minsize = 8))
  fit_optimal <- prune.tree(fit, best=3)

  #predict values for all MET values from the original data
  n=length(data$EX)
  predictions_pred = rnorm(n,predict(fit_optimal, newdata=data), sd(residuals(mle)))
  return(predictions_pred)
}

# 4.3.2. Bootstrap samples for condicence bands
f_conf=function(data1){
  # fit regression tree model
  fit = tree(EX~ MET , data=data1, control=tree.control(nobs= nrow(data1), minsize = 8))
  fit_optimal <- prune.tree(fit, best=3)

  # predict values from the original data
  predictions_conf = predict(fit_optimal, newdata=data)

  return(predictions_conf)
}

# 4.4 Now we can make bootstrap for each statistic:
res_pred=boot(data, statistic=f_pred, R=1000, mle=mle, ran.gen=rng, sim="parametric")
res_conf=boot(data, statistic=f_conf, R=1000, mle=mle, ran.gen=rng, sim="parametric")

# 4.5. Parametric bootstrap bands
e_pred=envelope(res_pred) # compute prediction bands
e_conf=envelope(res_conf) # compute confidence bands

# 4.5. Plotting the predicted line and 95% confidence and prediction bands
estimated_line =predict(fit_optimal)

{plot(data$MET, data$EX, xlab="MET", ylab="EX", pch=21, bg="orange", ylim=c(70,480))
points(data$MET,estimated_line,type="l") #plot fitted line
#plot prediction bands
points(data$MET,e_pred$point[2,], type="l", col="blue")
points(data$MET,e_pred$point[1,], type="l", col="blue")
#plot confidence bands
points(data$MET,e_conf$point[2,], type="l", col="red")
points(data$MET,e_conf$point[1,], type="l", col="red")
legend("bottomright",legend=c("Fitted line","Prediction bands", "Confidence bands"),
```

```
       lty = c(1,1,1), col=c("black","blue", "red"))}
```

Question 5.

```
dens = density(resid)
hist(resid, freq=F, xlab="residuals") #probability densities
lines(dens,col="red")
```

Assignment 4. Principal components.

Question 1.

```
## ----- ASSIGNMENT 4 ----- ##
# Importing the data:
data4 <- read.table("NIRspectra.csv", header=TRUE, sep=";", dec=",")

# Standard PCA
data_pca <- data4[,-127] #  the feature space
pca <- prcomp(data_pca)
round(summary(pca)$importance[,1:5],5)

# Plot of the variation explained by each PC.
library(factoextra)
fviz_eig(pca)

# Plot of the scores:
plot(pca$x[,1], pca$x[,2], pch=19, cex=0.4, xlab="PC1",
     ylab="PC2", main="Scores in the first 2 dimensions")
```

Question 2.

```
U= pca$rotation
par(mfrow=c(1,2))
plot(U[,1], main="Traceplot, PC1", xlab="Features",ylab="loadings")
plot(U[,2],main="Traceplot, PC2", xlab="Features", ylab="loadings")

cat("Variables with higest loadings in PC1:\n")
head(sort(U[,1], decreasing = TRUE))
cat("Variables with higest loadings in PC2:\n")
head(sort(U[,2], decreasing = TRUE))
```

Question 3a.

```
library(fastICA)
set.seed(12345)

ICA <- fastICA(data_pca,2)
W2 <- ICA$K %*% ICA$W

par(mfrow=c(1,2))
plot(W2[,1], main = "Trace plot of W'(1)")
plot(W2[,2], main = "Trace plot of W'(2)")
```

Question 3b.

```
plot(ICA$S[,1], ICA$S[,2], pch=19, cex=0.4, xlab="PC1",
     ylab="PC2", main="Scores in the first 2 PC's of ICA")
```