Block 1 - Lab 2

Roshni Sundaramurthy

9 December 2018

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. 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.

2.1

```
################################# Assignment 2 ###############################################
#load data and divide into training/validation/test as 50/25/25
library(xlsx)
credit_data <- read.xlsx("creditscoring.xls",1)
n=dim(credit_data)[1]
set.seed(12345)
id=sample(1:n, floor(n*0.5))
train=credit_data[id,]
id1=setdiff(1:n, id)
set.seed(12345)
id2=sample(id1, floor(n*0.25))
valid=credit_data[id2,]
id3=setdiff(id1,id2)
test=credit_data[id3,]
```

2.2

```
## [1] "Using Deviance"

## Confusion Matrix for test data :

##       pred_test
##         bad good
##   bad   28   48
##   good  19  155

## Misclassification rate of test data :  0.268

## Confusion Matrix for train data :

##       pred_train
##         bad good
##   bad   61   86
##   good  20  333

## Misclassification rate of train data :  0.212

## [1] "Using Gini Index"

## Confusion Matrix for test data :
```

```
##       pred_test1
##        bad good
##   bad   18   58
##   good  33  141
```

```
## Misclassification rate of test data :  0.364
```

```
## Confusion Matrix for train data :
```

```
##       pred_train1
##        bad good
##   bad   66   81
##   good  40  313
```

```
## Misclassification rate of train data :  0.242
```

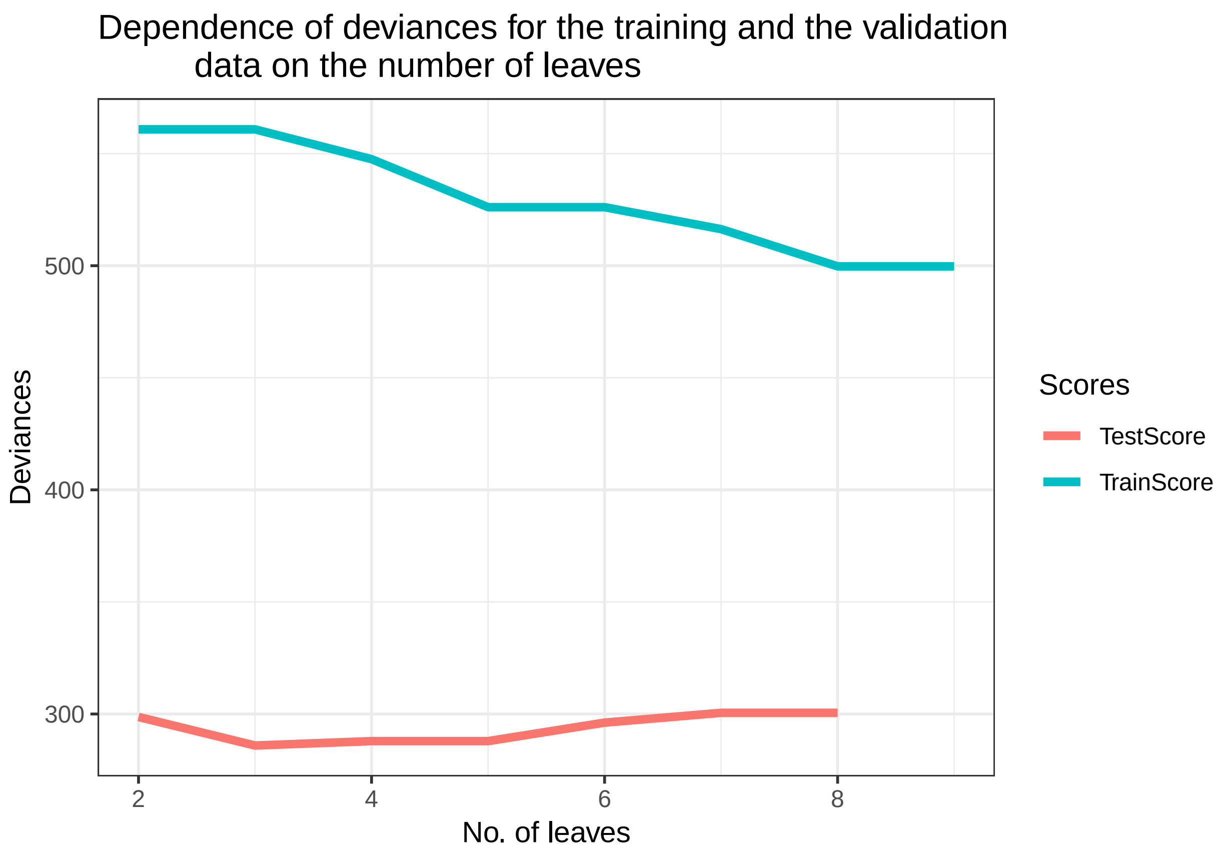
Analysis:

a) Using Deviance measure: The misclassification error for test data (0.268) is greater than train data (0.212). This is obvious that train data has been fitted and so error is lesser than test data. For test data predictions, 155 customers have managed their loans well. For train data, it is 333 customers.

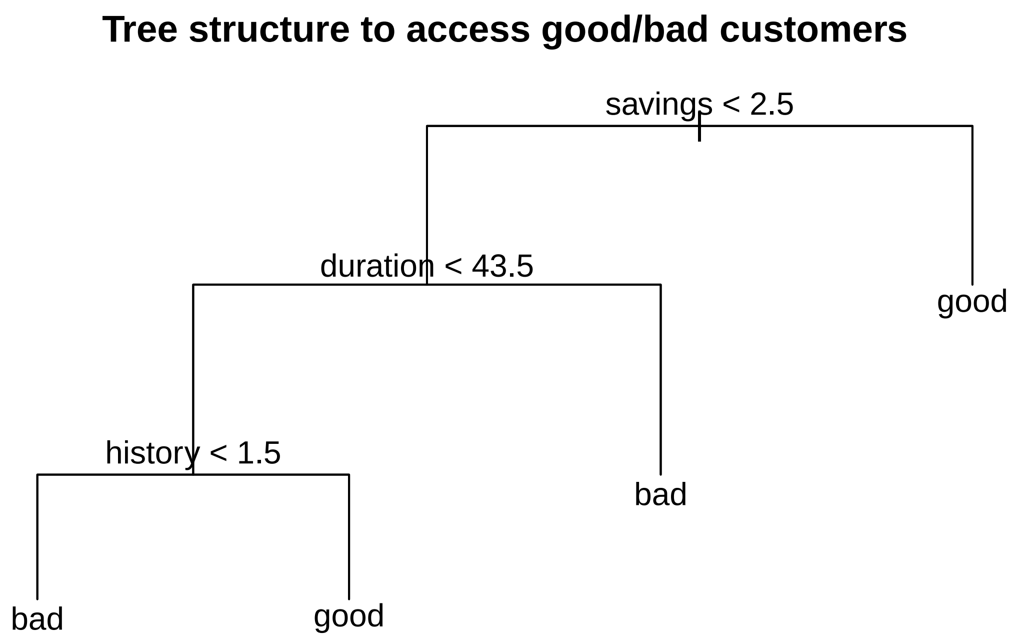
b) Using gini index measure: The misclassification error for test data (0.364) is greater than train data (0.242). For test data, 141 customers and for train, it is 313 customers who have managed their loans well. These misclassification rates are greater than the one which we got using deviance measure.

From the above measures, the *deviance measure provided the better results.* So I have chosen this model for the following steps.

2.3



```
## Misclassification rate of test data :  0.256
```



Analysis:

The training and validation sets of the data are choosen to get the optimal tree depth. The plot of dependence of deviances for the training and the validation data on the number of leaves is plotted. The pattern of deviance for test score is random. The deviance for train score seems to be decreasing with increase in number of leaves. So, we can conclude that the best number of leaves for both scores is 12 and the depth of it is 4. The misclassification error rate is 0.256. The tree structure shows that the customer with savings less than 2.5, and history of repaying loans with less than 1.5 seems to be a good customer.

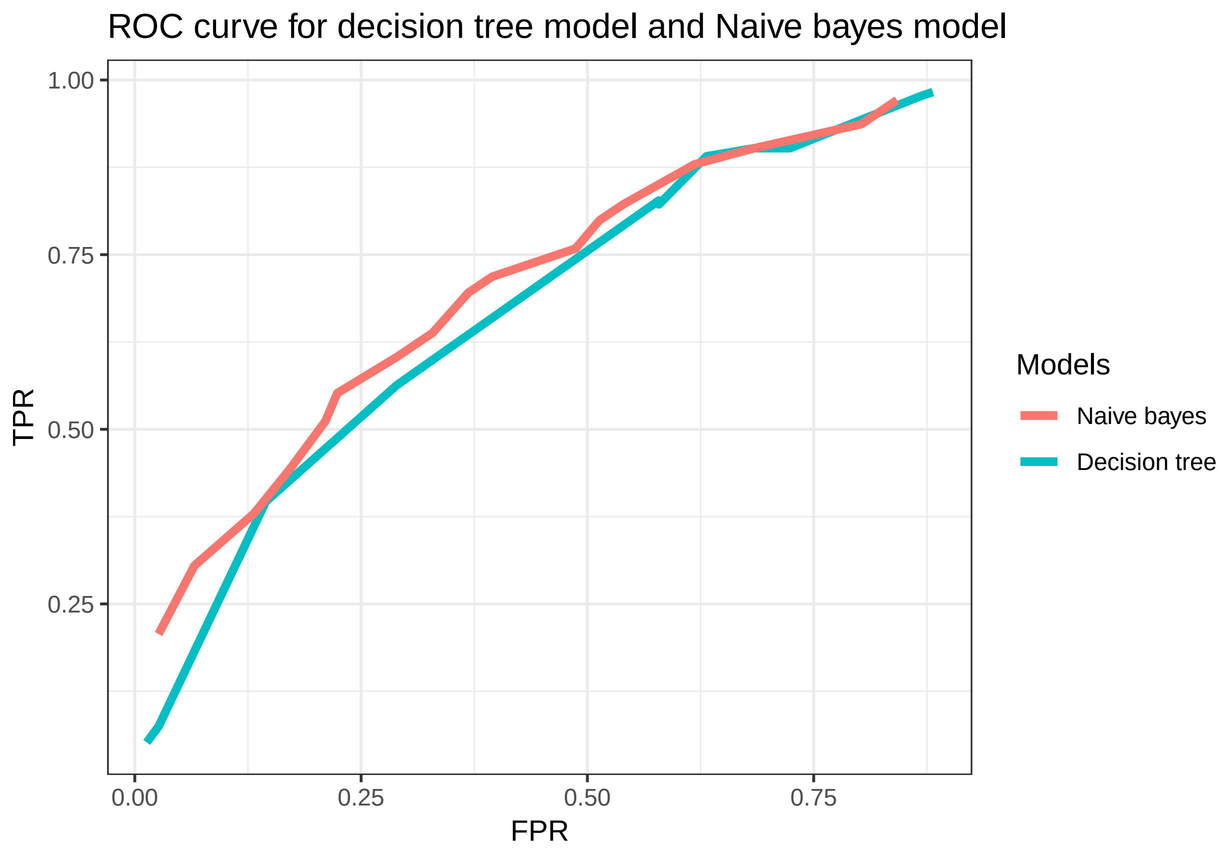
2.4

```
## Confusion Matrix for test data :

##       naive_pred_test
##        bad good
##   bad   46   30
##   good  49  125

## Misclassification rate of test data :  0.316

## Confusion Matrix for train data :

##       naive_pred_train
##        bad good
##   bad   95   52
##   good  98  255
## Misclassification rate of train data :  0.3
```

Analysis:

Here it seems a small variation in misclassification error for both test (0.316) and train data (0.3). 125 customers for test data and 255 customers for train data have managed their loans well. When compared to step 3 (Decision tree), the error rate for both train and test data is greater for naive bayes. Therefore, Decision tree classifier is the best in this case.

2.5



Analysis:

Using various threshold values, corresponding TPR (True Positive Rate) and FPR (False Positive Rate) are calculated for both decision tree model and naive bayes model and plotted to get ROC curve. From plot, it is observed that naive bayes classifier has higher TPR than the decision tree. And it has greatest Area Under Curve (AUC). Hereby, we can conclude that Naive bayes classifier is the best classifier.

2.6

```
## [1] "Confusion Matrix for test data : "

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

```
## Misclassification rate of train data :  0.508
```

```
## [1] "Confusion Matrix for train data : "
```

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

```
## Misclassification rate of train data :  0.546
```

Analysis:

Results from naive bayes with loss matrix:

For test data, 52 customers have been classified as good in managing loans and the error rate is 0.508. But train data, 90 customers have been classified as good in managing loans and the error rate is 0.546. Here the error rate seems to be higher for both train and test data. When compared to step 2.4, the number of customers managed their loans well is large. And also the error rate seems to be less in 2.4. As we used loss matrix with 10*bad, the error rate gets increased.

Assignment 3. Uncertainty estimation

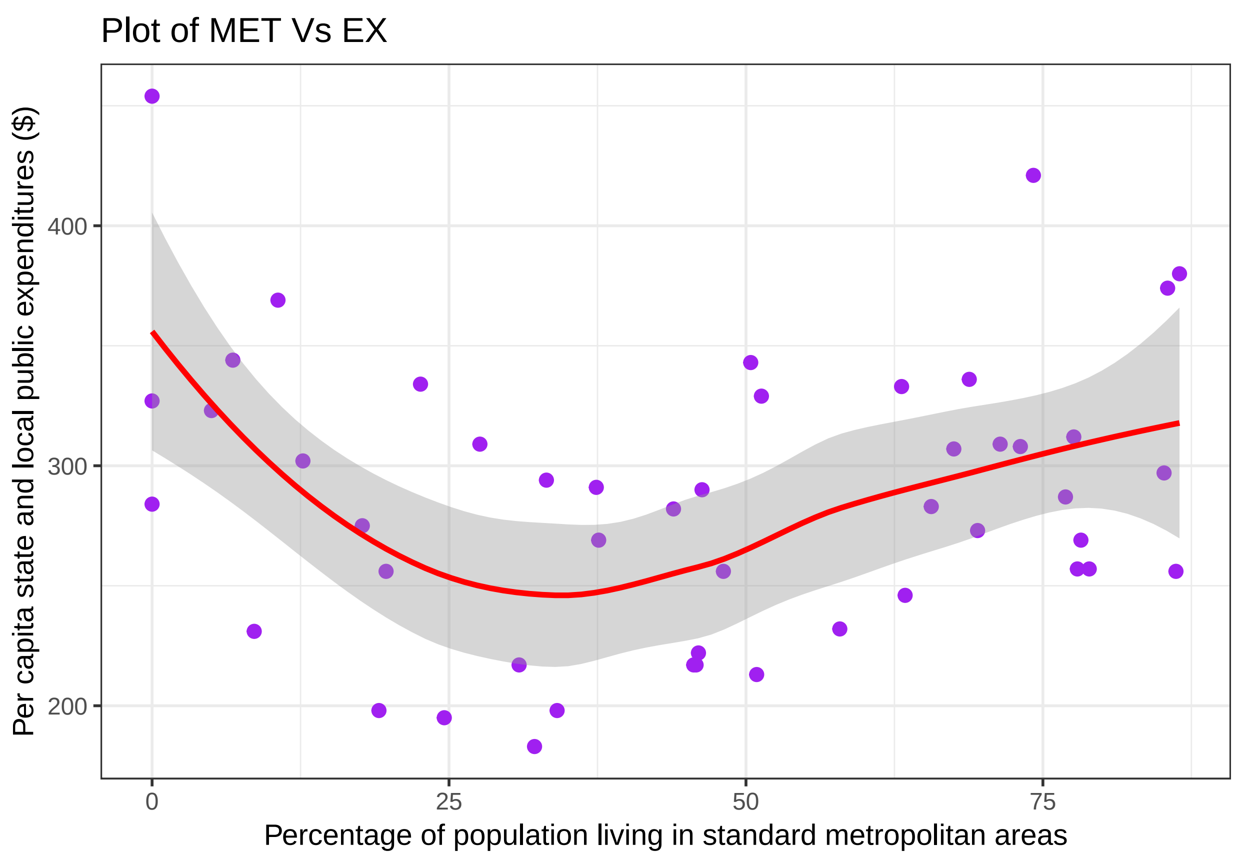
The data file State.csv contains per capita state and local public expenditures and associated state demographic and economic characteristics, 1960, and there are variables

. MET: Percentage of population living in standard metropolitan areas

. EX: Per capita state and local public expenditures ($)

3.1

```
################################ Assignment 3 ############################################
library(ggplot2)
State_data <- read.csv2("State.csv",sep = ";")
State_df <- State_data[order(State_data$MET), ]
ggplot(State_df)+geom_point(aes(MET,EX),colour="purple",size=2)+
  geom_smooth(aes(MET,EX),colour="red")+
  theme_bw() +ggtitle("Plot of MET Vs EX")+
    xlab("Percentage of population living in standard metropolitan areas")+
    ylab("Per capita state and local public expenditures ($)")
```



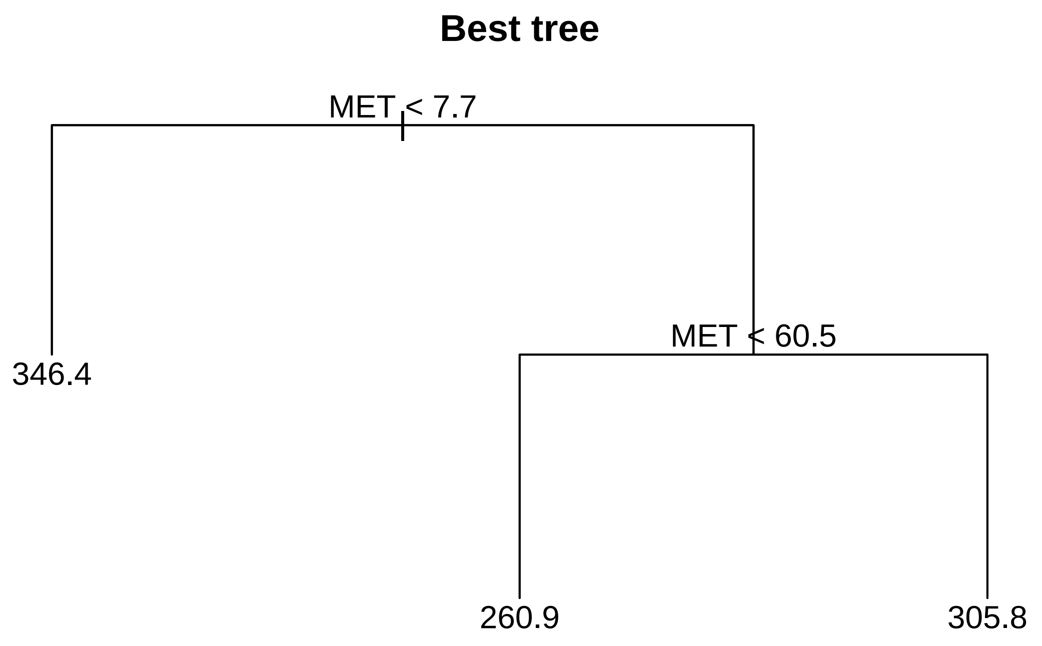
Analysis:

The observations in the plot is regarding the metropolitan ratio and the local public expenditures ($) for several states. The data does not seems to be having any linear relationship between the features. So, Multivariate Adaptive Regression Splines models can be appropriate to use here.

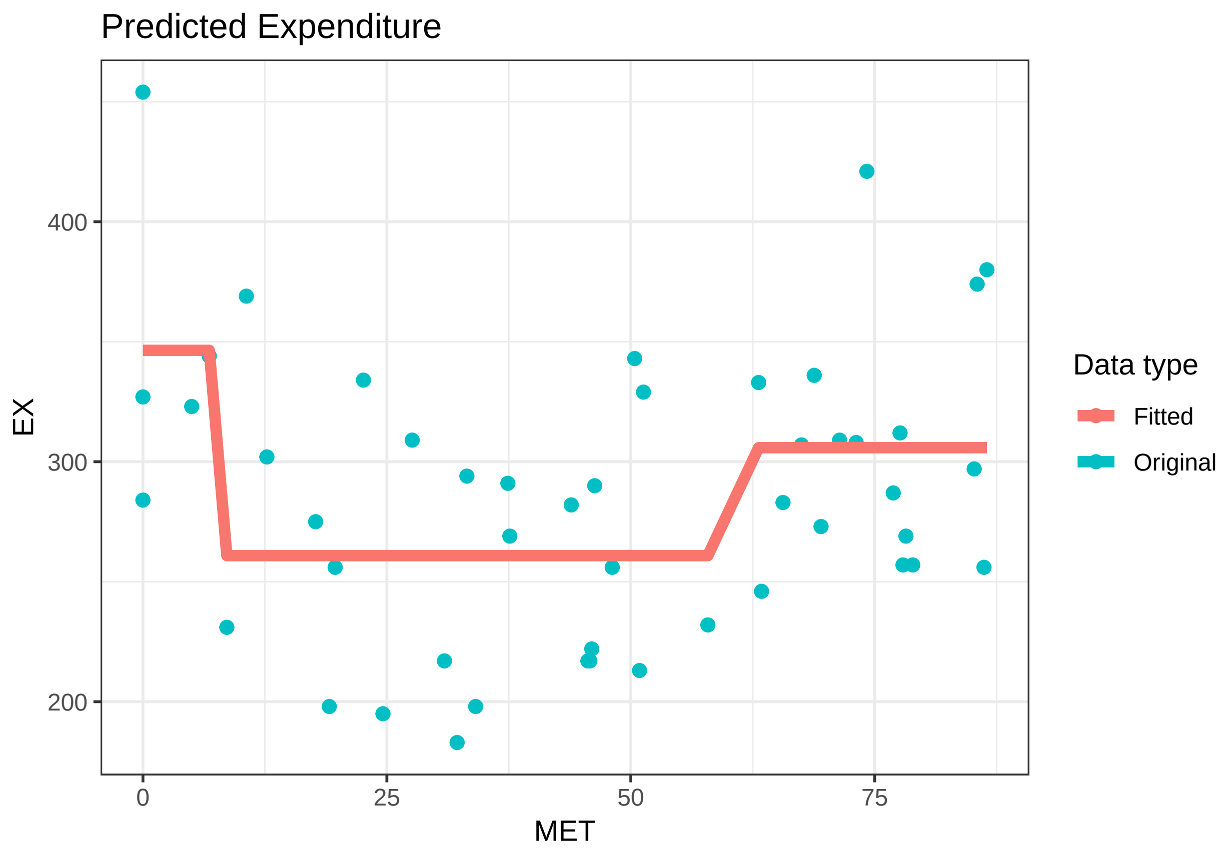
Steps by MARS:

- This method transform variables via "basis functions"
- Perform stepwise procedure on the transformed variables
- Build the model out
- Prune back using cross-validation

3.2

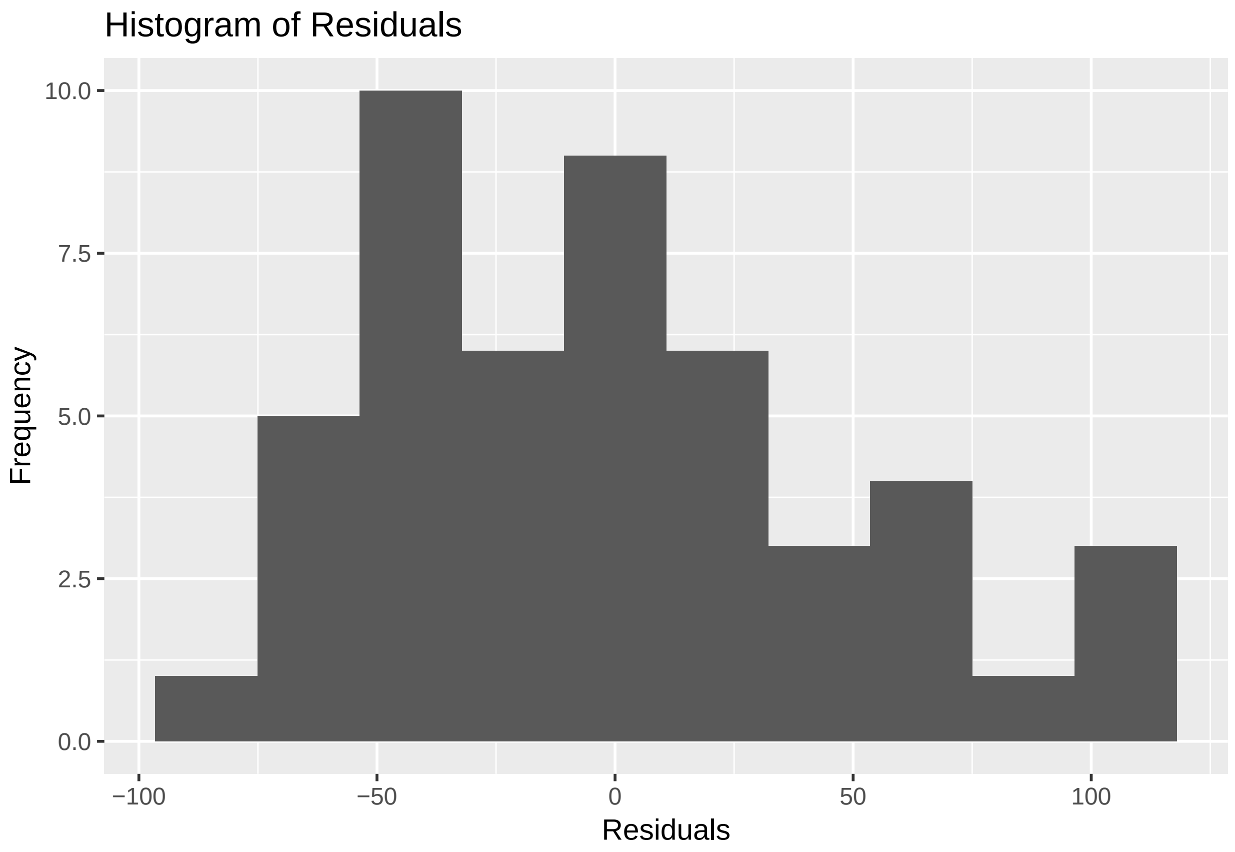


Best tree
MET < 7.7
MET < 60.5
346.4
260.9
305.8



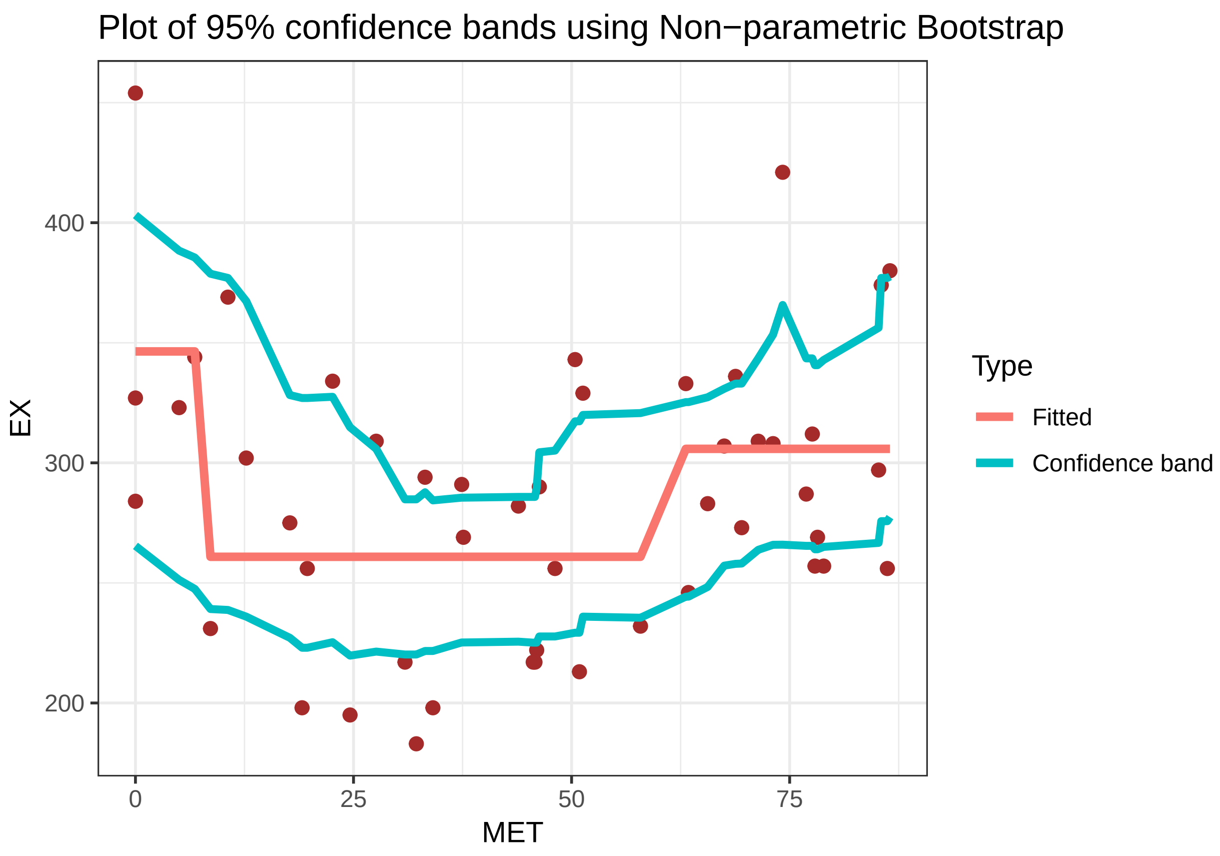
Predicted Expenditure
EX
400
300
200
0
25
50
75
MET
Data type
Fitted
Original

The optimal regression tree is found by pruning the original tree. Then we predict the observed data with the optimal tree. When this predicted optimal tree is plotted with the original data, it seems that fitted value is better between 200 and 300 of expenditure. But still, the model is not performing good. The model just predict some in the right way because its around the mean value. Hence, the quality of the fit is not obtained to the expected level.



The residuals are plotted as histogram. This is nothing but the distance between the original data and the predicted data. From the plot, we can say that it is not normally distributed. The frequency remains high for residuals with positive values i.e, 0 to 100, when compared to negative values i.e, 0 to -100.

3.3

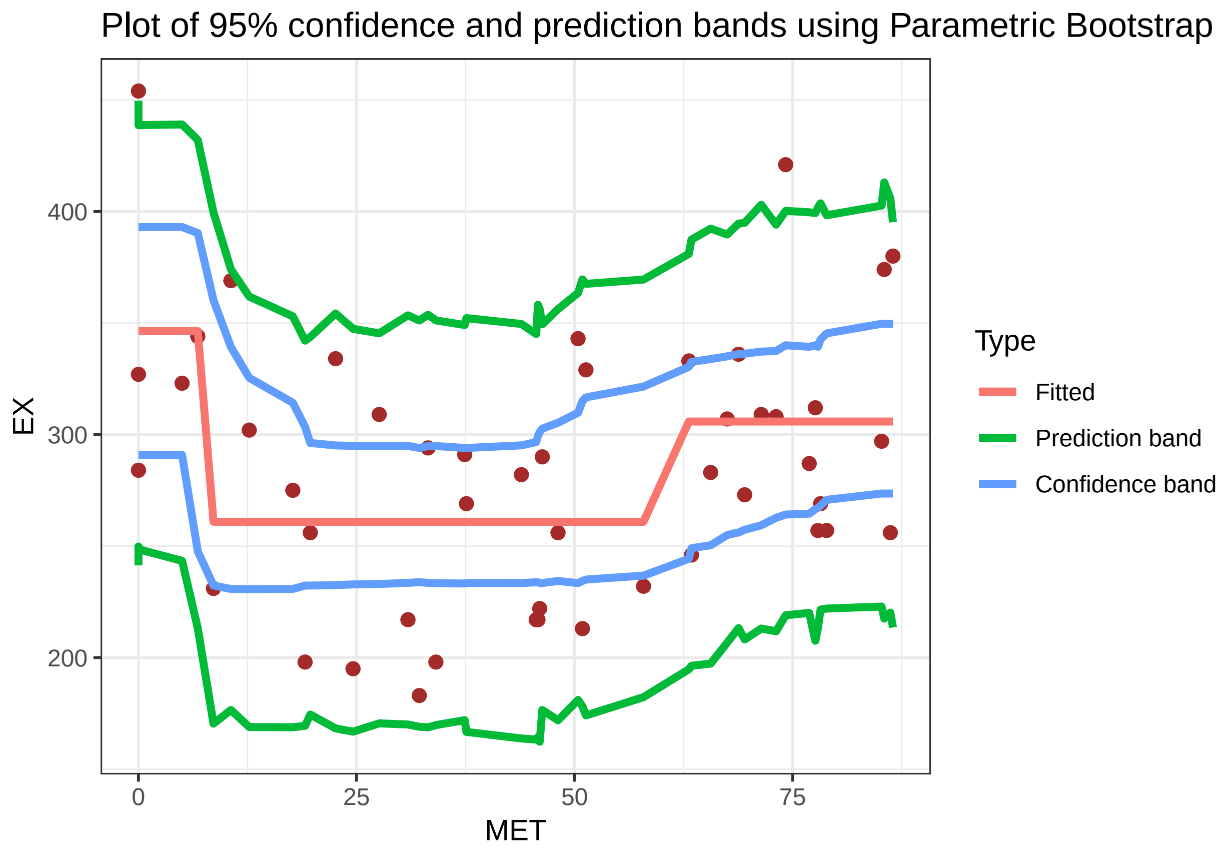


Analysis:

As the above steps does not describe the data well, we are using boot strap method. Since, we don't know the distribution from the above steps, we are trying non parametric boot strapping.The nonparametric bootstrap allows you to estimate the sampling distribution (usually) as precisely as needed without specifying that the sampling distribution is normal, or gamma, or so on.

The data points seems to be lying outside of the confidence bands (95% CI). These bands are not smooth as the predictor lies between the 3 leaves and also the width of the bands reduces as it progresses. The results of the regression model in step 2 does not seem to be reliable because of the confidence bands width.

3.4



Analysis:

Now, we are considering $Y \sim N(\mu_i, \sigma^2)$, μ_i are labels in the tree leaves and σ^2 is the residual variance. It denotes that the target variable is normally distributed. Now, we can consider using parametric bootstrap method.

Now, the prediction band plays a major role. It almost covers all data points. And also the width of the bands are narrower and the results of the regression model in step 2 seem to be reliable due to this reason. It looks like only 5% of data are outside the prediction band. Therefore the prediction seems to be reasonable and quite accurate in this method.

3.5

Analysis:

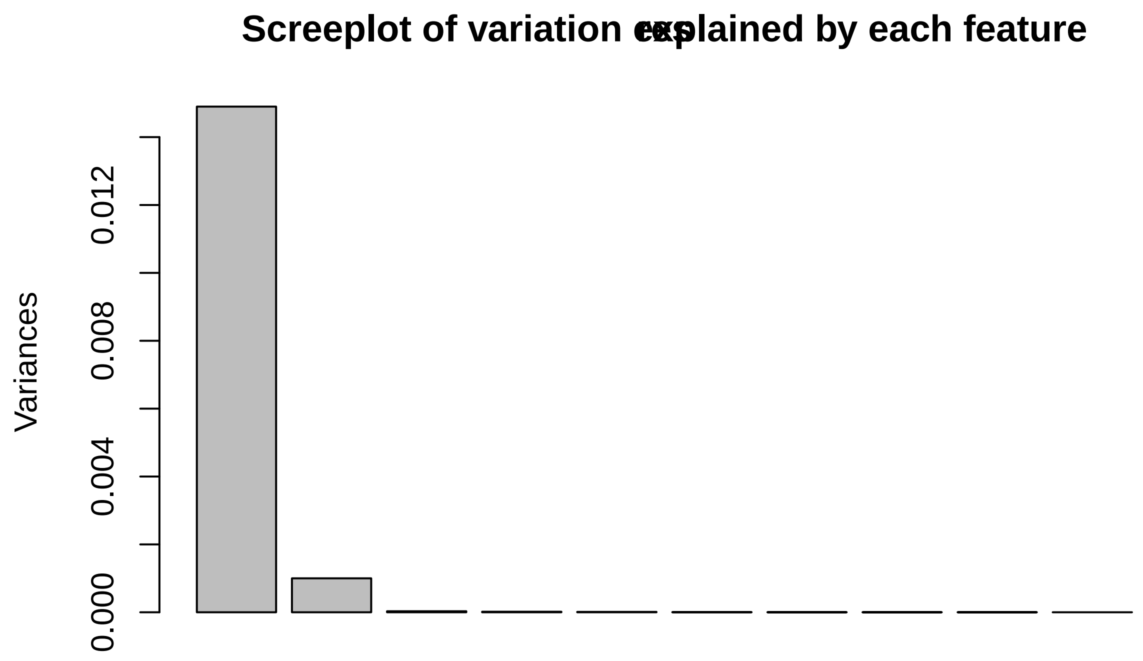
From the histogram obtained from previous steps, the parametric bootstrapping is actually appropriate to use here because, new samples have been generated in parametric bootstrap method. While in non parametric bootstrapping, samples were from the original data set. And also the confidence bands does not seems to be good in later method. So, we prefer parametric bootstrap for this data.

Assignment 4. Principal components

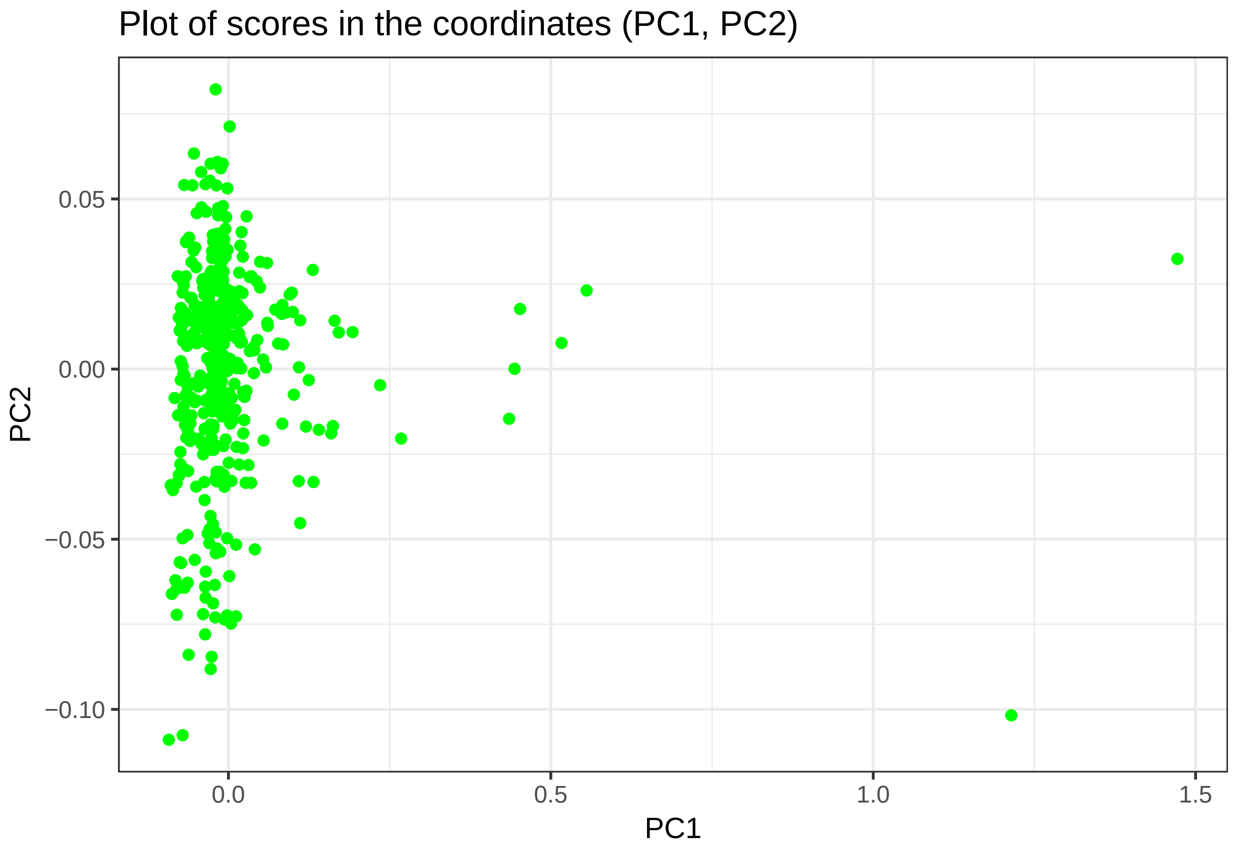
The data file NIRspectra.csv contains near-infrared spectra and viscosity levels for a collection of diesel fuels. The task is to investigate how the measured spectra can be used to predict the viscosity.

4.1

```
##   [1] "93.332" "6.263"  "0.185"  "0.101"  "0.068"  "0.025"  "0.009"
##   [8] "0.003"  "0.003"  "0.002"  "0.001"  "0.001"  "0.001"  "0.001"
##  [15] "0.000"  "0.000"  "0.000"  "0.000"  "0.000"  "0.000"  "0.000"
##  [22] "0.000"  "0.000"  "0.000"  "0.000"  "0.000"  "0.000"  "0.000"
##  [29] "0.000"  "0.000"  "0.000"  "0.000"  "0.000"  "0.000"  "0.000"
##  [36] "0.000"  "0.000"  "0.000"  "0.000"  "0.000"  "0.000"  "0.000"
##  [43] "0.000"  "0.000"  "0.000"  "0.000"  "0.000"  "0.000"  "0.000"
##  [50] "0.000"  "0.000"  "0.000"  "0.000"  "0.000"  "0.000"  "0.000"
##  [57] "0.000"  "0.000"  "0.000"  "0.000"  "0.000"  "0.000"  "0.000"
##  [64] "0.000"  "0.000"  "0.000"  "0.000"  "0.000"  "0.000"  "0.000"
##  [71] "0.000"  "0.000"  "0.000"  "0.000"  "0.000"  "0.000"  "0.000"
##  [78] "0.000"  "0.000"  "0.000"  "0.000"  "0.000"  "0.000"  "0.000"
##  [85] "0.000"  "0.000"  "0.000"  "0.000"  "0.000"  "0.000"  "0.000"
##  [92] "0.000"  "0.000"  "0.000"  "0.000"  "0.000"  "0.000"  "0.000"
##  [99] "0.000"  "0.000"  "0.000"  "0.000"  "0.000"  "0.000"  "0.000"
## [106] "0.000"  "0.000"  "0.000"  "0.000"  "0.000"  "0.000"  "0.000"
## [113] "0.000"  "0.000"  "0.000"  "0.000"  "0.000"  "0.000"  "0.000"
## [120] "0.000"  "0.000"  "0.000"  "0.000"  "0.000"  "0.000"  "0.000"
```

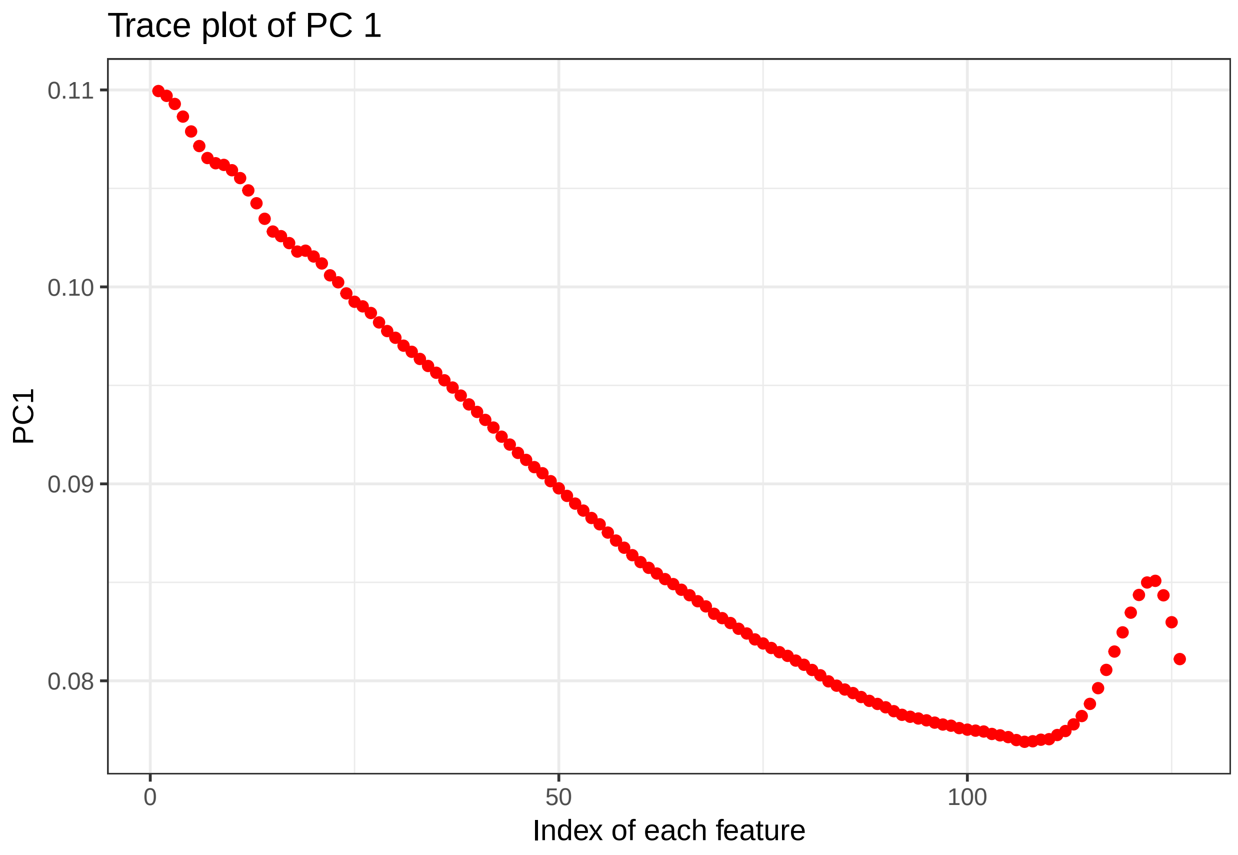


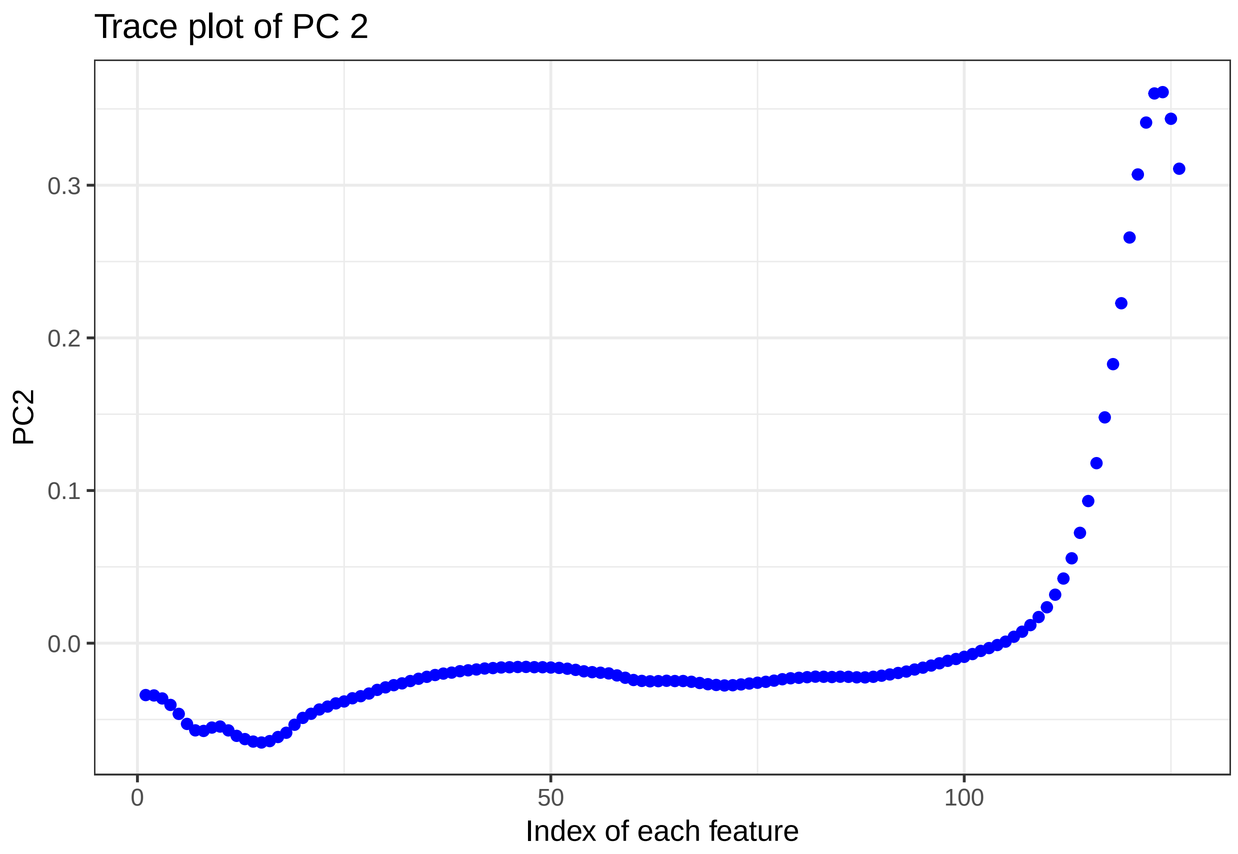
The screeplot shows that the two features (measured spectra) captures the most variation. So, the minimal number of components explaining at least 99% of the total variance is PC1 and PC2.



Data in (PC1, PC2) - scores (Z) as given in lecture slides. The scatter plot is plotted between PC1 and PC2. There are unusual diesel fuels according to this plot as it clearly shows that there are a few outliers. Especially, the two outliers between 1.0 and 1.5 in x axis. This can be classified as unusual diesel fuels.

4.2


Trace plot of PC 1
PC1
0.11
0.10
0.09
0.08
0
50
100
Index of each feature


Trace plot of PC 2
PC2
0.3
0.2
0.1
0.0
0
50
100
Index of each feature

Analysis:

The correlation will be strong for high loadings of principle components for given features. From both plots, it seems that the loadings of PC2 differs greatly because for PC2, the maximum loading is 0.3609 and minimum loading is -0.0650. For PC1, the loadings do not differ much as it has 0.1099 as max loading and 0.0769 as min loading. It is observed that the principal components are not explained by a few original features but with many features.

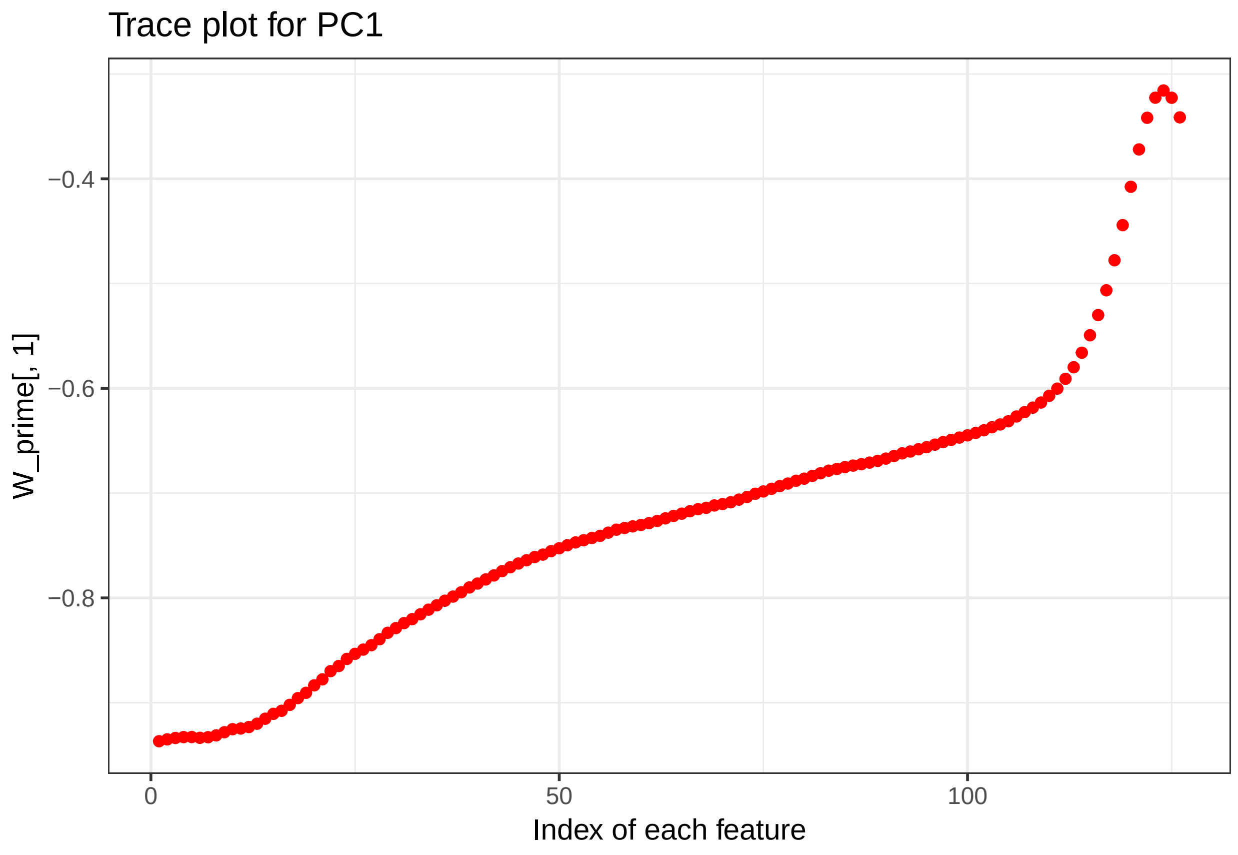
4.3

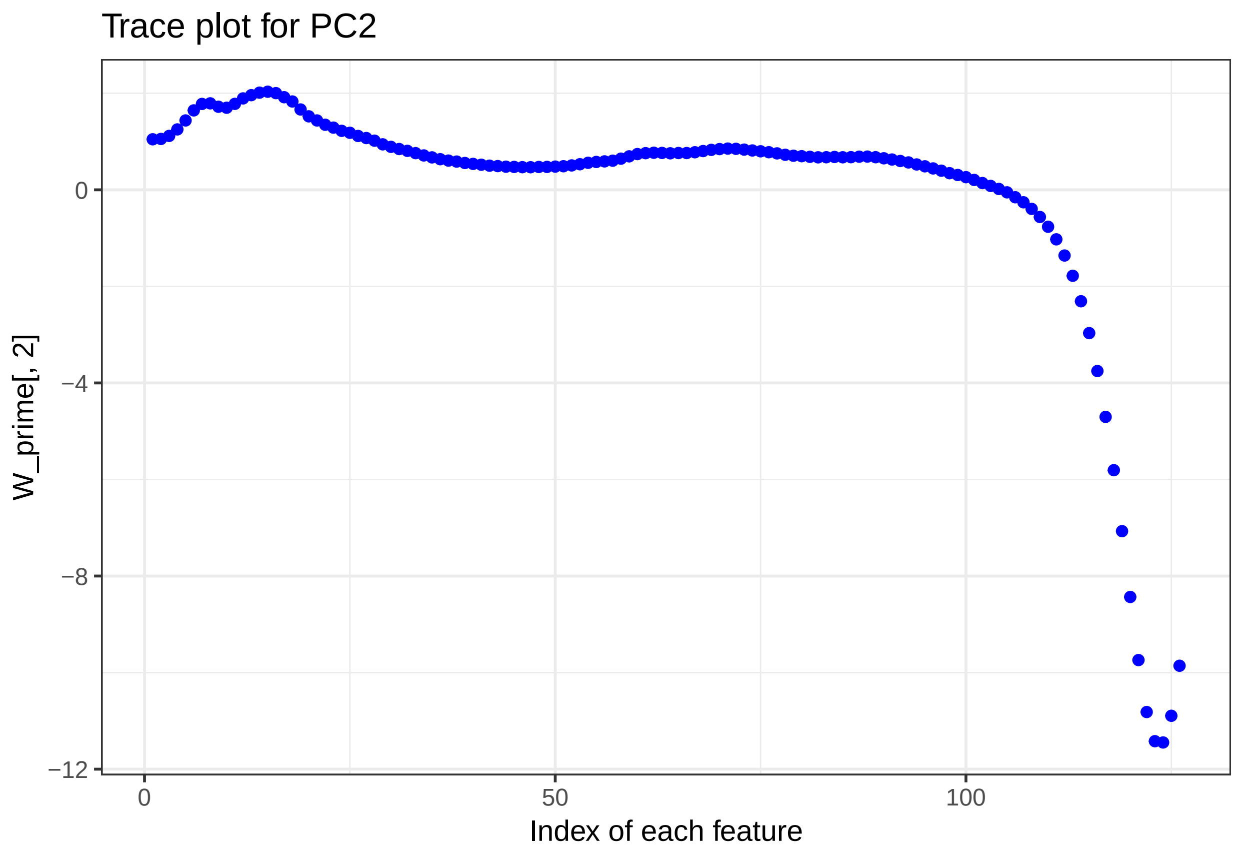
The Independent Component Analysis with the number of components selected in step 1 is performed with NIR_data.

The W prime matrix is computed as follows:

```
#Computing W_prime matrix
W_prime <- NIR_ICA$K %*% NIR_ICA$W
```

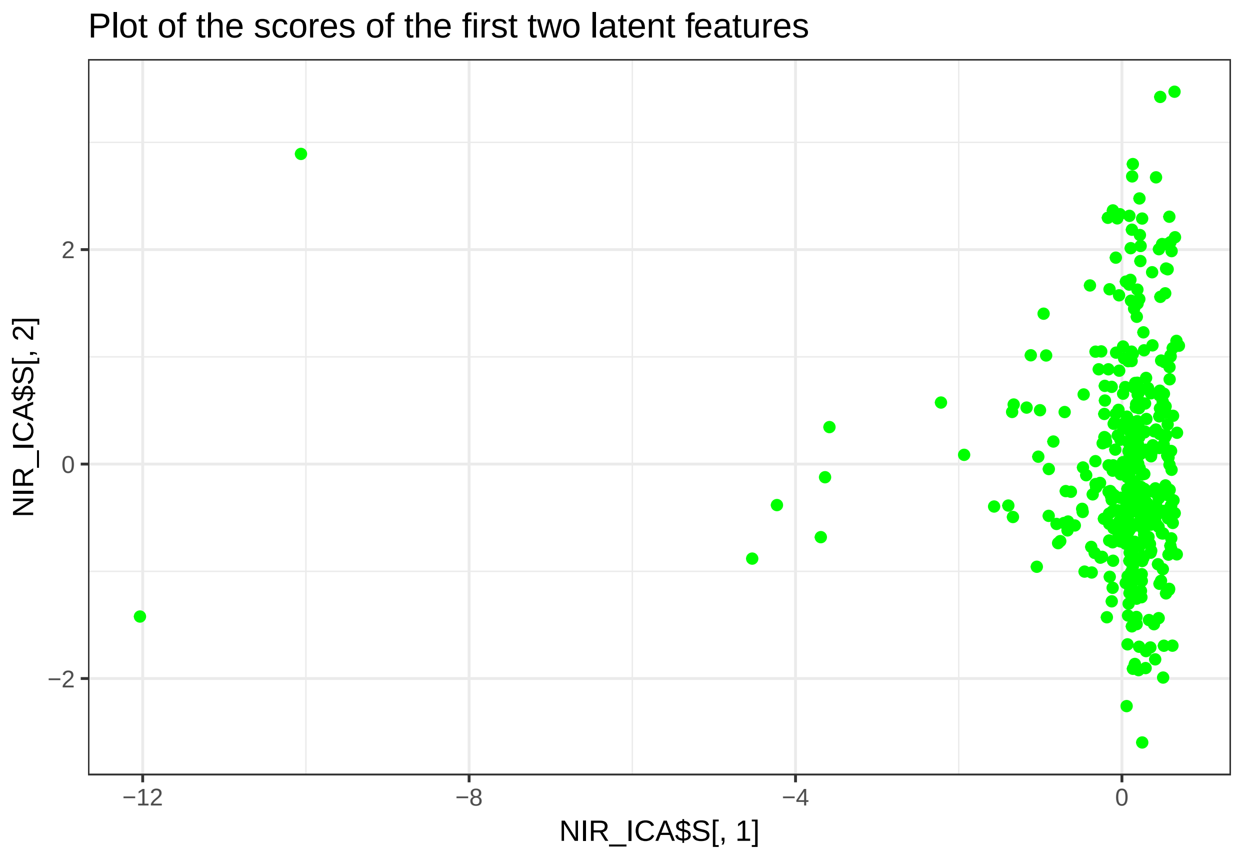
Columns of W prime in the form of trace plots:





When we compare these two plots with step 1, it seems that these plots are just inverted version of PC1 and PC2 plots. This is because, we are measuring independence instead of correlation. So, almost the W prime matrix have the same values as the loading components but in a reversed form.

The plot of scores of the first two latent features is plotted:



Analysis:

It has been observed that, again this plot is reversed when compared to step 1, as we seen before in separate trace plots of components. But we can observe some differences in the unit measures. There are some unusual diesel fuels according to this plot as it clearly shows that there are a few outliers as in PCA. Especially, the two outliers between -10 and -12 in x axis. This can be classified as unusual diesel fuels in ICA method.

Appendix

```r
################################# Assignment 2 ##############################################
#load data and divide into training/validation/test as 50/25/25
library(xlsx)
credit_data <- read.xlsx("creditscoring.xls",1)
n=dim(credit_data)[1]
set.seed(12345)
id=sample(1:n, floor(n*0.5))
train=credit_data[id,]
id1=setdiff(1:n, id)
set.seed(12345)
id2=sample(id1, floor(n*0.25))
valid=credit_data[id2,]
id3=setdiff(id1,id2)
test=credit_data[id3,]
```

```r
#Fit a decision tree to the training data
library(tree)

set.seed(12345)
paste( "Using Deviance")
decision_tree_dev_fit <- tree(good_bad~., data=train, split = c("deviance"))

#Using predict on test data
pred_test <- predict(decision_tree_dev_fit, test, type="class")

#Confusion matrix for test data
conf_matrix_test <- table(test$good_bad, pred_test)
cat(paste("Confusion Matrix for test data : "))
conf_matrix_test

#Misclassification rate for test data
misclassification_rate_test <- 1 - sum(diag(conf_matrix_test))/sum(conf_matrix_test)
cat(paste("Misclassification rate of test data : ", misclassification_rate_test,"\n"))

#Using predict on train data
pred_train <- predict(decision_tree_dev_fit, train, type="class")

#Confusion matrix for train data
conf_matrix_train <- table(train$good_bad,pred_train)
cat(paste("Confusion Matrix for train data : "))
conf_matrix_train

#Misclassification rate for train data
misclassification_rate_train <- 1 - sum(diag(conf_matrix_train))/sum(conf_matrix_train)
cat(paste("Misclassification rate of train data : ", misclassification_rate_train,"\n"))

paste ("Using Gini Index")
decision_tree_gin_fit <- tree(good_bad~., data=train, split = c("gini"))

#Using predict on test data
pred_test1 <- predict(decision_tree_gin_fit, test, type="class")

#Confusion matrix for test data
conf_matrix_test1 <- table(test$good_bad,pred_test1)
# or with(test, table(pred_test, good_bad))
cat(paste("Confusion Matrix for test data : "))
conf_matrix_test1

#Misclassification rate for test data
misclassification_rate_test1 <- 1 - sum(diag(conf_matrix_test1))/sum(conf_matrix_test1)
cat(paste("Misclassification rate of test data : ", misclassification_rate_test1,"\n"))

#Using predict on train data
pred_train1 <- predict(decision_tree_gin_fit, train, type="class")

#Confusion matrix for train data

conf_matrix_train1 <- table(train$good_bad,pred_train1)
```

```r
cat(paste("Confusion Matrix for train data : "))
conf_matrix_train1

#Misclassification rate for train data
misclassification_rate_train1 <- 1 - sum(diag(conf_matrix_train1))/sum(conf_matrix_train1)
cat(paste("Misclassification rate of train data : ", misclassification_rate_train1,"\n"))

# Selecting optimal tree
trainScore<-rep(0,9)
testScore<-rep(0,9)
library(rpart)
for(i in 2:9) {
  prunedTree<-prune.tree(decision_tree_dev_fit,best=i)
  pred<-predict(prunedTree, newdata=valid, type="tree")
  trainScore[i]<-deviance(prunedTree)
  testScore[i]<-deviance(pred)
}
library(ggplot2)
ggdat<-data.frame(TrainScore=trainScore[2:9],TestScore=testScore[2:9])
ggplot(ggdat)+
  geom_line(aes(x=2:9, y=TrainScore,color="green"),size=1.5)+
  geom_line(aes(x=2:9, y=TestScore[2:9],color="blue"),size=1.5)+
  ggtitle("Dependence of deviances for the training and the validation
          data on the number of leaves")+xlab("No. of leaves")+ ylab("Deviances")+
  scale_color_discrete(name = "Scores", labels = c("TestScore", "TrainScore"))+theme_bw()

# prune the tree to the required depth
prune_for_test <- prune.tree(decision_tree_dev_fit, best = 4)

# misclassification rate for best pruned tree
predict_prune_test <- predict(prune_for_test, test, type = "class")
conf_prune_tree_test <- table(test$good_bad, predict_prune_test)

#Misclassification rate for test data
misclassification_rate <- 1 - sum(diag(conf_prune_tree_test))/sum(conf_prune_tree_test)
cat(paste("Misclassification rate of test data : ", misclassification_rate,"\n"))

# plot of tree structure
plot(prune_for_test)
text(prune_for_test)
title("Tree structure to access good/bad customers")
#classification using Naïve Bayes
library(MASS)
library(e1071)
set.seed(12345)
naiveBayes_fit <- naiveBayes(good_bad~., data=train)
#naiveBayes_fit

#Using predict on test data
naive_pred_test <- predict(naiveBayes_fit, test, type = "class")

#Confusion matrix for test data
conf_matrix_test2 <- table(test$good_bad,naive_pred_test)
```

```r
cat(paste("Confusion Matrix for test data : "))
conf_matrix_test2

#Misclassification rate for test data
misclassification_rate_test2 <- 1 - sum(diag(conf_matrix_test2))/sum(conf_matrix_test2)
cat(paste("Misclassification rate of test data : ", misclassification_rate_test2,"\n"))

#Using predict on train data
naive_pred_train <- predict(naiveBayes_fit, train, type="class")

#Confusion matrix for train data
conf_matrix_train2 <- table(train$good_bad,naive_pred_train)
cat(paste("Confusion Matrix for train data : "))
conf_matrix_train2

#Misclassification rate for train data
misclassification_rate_train2 <- 1 - sum(diag(conf_matrix_train2))/sum(conf_matrix_train2)
cat(paste("Misclassification rate of train data : ", misclassification_rate_train2,"\n"))
#Use the optimal tree and the Naïve Bayes model to classify the test data
#Classification of train data

#Decision tree
pred_test <- predict(decision_tree_dev_fit, test, type="vector")
pi<-seq(0.05,0.95,0.05)

pred_tre <- lapply(pi, function(x){
    ifelse(as.numeric(pred_test[,2]) > x, 1,0)
  }
)

pred_tre1 <- lapply(pred_tre, function(x){
    ifelse(x == 0,"bad","good")
  }
)

# confus matrix
matrixx <- lapply(pred_tre1, function(x){
  table(test$good_bad,x)
})

tpr <- lapply(matrixx, function(x){
  x[2,2]/ sum(x[2,])
})

fpr <- lapply(matrixx, function(x){
  x[1,2]/ sum(x[1,])
})

# Naive Bayes
pred_test1 <- predict(naiveBayes_fit, test, type="raw")

pred_tre1 <- lapply(pi, function(x){
    ifelse(as.numeric(pred_test1[,2]) > x, 1,0)
  }
)

pred_tre1_1 <- lapply(pred_tre1, function(x){
    ifelse(x == 0,"bad","good")
  }
)

# confus matrix
matrixx1 <- lapply(pred_tre1_1, function(x){
  table(test$good_bad,x)
})

tpr1 <- lapply(matrixx1, function(x){
  x[2,2]/ sum(x[2,])
})

fpr1 <- lapply(matrixx1, function(x){
  x[1,2]/ sum(x[1,])
})

tpr_fpr <- as.data.frame(pi)
tpr_fpr$TPR <- unlist(tpr)
tpr_fpr$FPR <- unlist(fpr)
tpr_fpr$TPR1 <- unlist(tpr1)
tpr_fpr$FPR1 <- unlist(fpr1)

names(tpr_fpr) <- c("Threshold","TPR_tree","FPR_tree","TPR_naive","FPR_naive")

# plotting ROC
ggplot(tpr_fpr)+
geom_line(aes(x=FPR_tree, y=TPR_tree,color="green"),size=1.5)+
  geom_line(aes(x=FPR_naive, y=TPR_naive,color="blue"),size=1.5)+
  ggtitle("ROC curve for decision tree model and Naive bayes model ")+xlab("FPR")+ylab("TPR")+
  scale_color_discrete(name = "Models", labels = c("Naive bayes","Decision tree"))+theme_bw()

# from 2.4, but changed "type"
#for test data
naive_pred_test <- predict(naiveBayes_fit, test, type = "raw")
naive_pred_test_classify <- ifelse(naive_pred_test[,2]>naive_pred_test[,1]*10,"good","bad")
L_matrix_test <- table(test$good_bad,naive_pred_test_classify)
paste("Confusion Matrix for test data : ")
L_matrix_test
#Misclassification rate for train data
misclass_rate_test <- 1 - sum(diag(L_matrix_test))/sum(L_matrix_test)
cat(paste("Misclassification rate of train data : ", misclass_rate_test))


#for train data
naive_pred_train <- predict(naiveBayes_fit, train, type = "raw")
naive_pred_train_classify <- ifelse(naive_pred_train[,2]*1>naive_pred_train[,1]*10,"good","bad")
L_matrix_train <- table(train$good_bad,naive_pred_train_classify)
paste("Confusion Matrix for train data : ")
```

```r
L_matrix_train
#Misclassification rate for train data
misclass_rate_train <- 1 - sum(diag(L_matrix_train))/sum(L_matrix_train)
cat(paste("Misclassification rate of train data : ", misclass_rate_train))
################################ Assignment 3 ###########################################
library(ggplot2)
State_data <- read.csv2("State.csv",sep = ";")
State_df <- State_data[order(State_data$MET), ]
ggplot(State_df)+geom_point(aes(MET,EX),colour="purple",size=2)+
  geom_smooth(aes(MET,EX),colour="red")+
  theme_bw() +ggtitle("Plot of MET Vs EX")+
    xlab("Percentage of population living in standard metropolitan areas")+
    ylab("Per capita state and local public expenditures ($)")
#n=nrow(State_data)
set.seed(12345)
# fitting a regression tree model with target EX and feature MET
reg_tree_fit <- tree(formula = EX ~ MET, data = State_data,
                     control = tree.control(nobs=nrow(State_data), minsize = 8))

# number of the leaves is selected by cross-validation
sel_leaf_cv <- cv.tree(reg_tree_fit)

# Plot of selected tree
#which.min(sel_leaf_cv$dev)  # 5
leaf <- sel_leaf_cv$size[5]
sel_tree <- prune.tree(reg_tree_fit, best = leaf)
plot(sel_tree)
title("Best tree")
text(sel_tree)

# Plot of Original and Fitted Data
sel_tree_pred <- predict(sel_tree)
ggplot(State_data) +
  geom_point(aes(x=MET, y=EX, color="green"),size=2) +
  geom_line(aes(x=MET, y=sel_tree_pred, color="blue"),size=2) +
  ggtitle("Predicted Expenditure") + theme_bw()+
  scale_color_discrete(name = "Data type", labels = c("Fitted","Original"))
# Histogram of Residuals
ggplot()+geom_histogram(aes(State_data$EX - sel_tree_pred),bins = 10)+
  xlab("Residuals")+ ylab("Frequency")+
  ggtitle("Histogram of Residuals")
library(boot)
# Computing bootstrap samples
f <- function(data, ind){
  data1 <- data[ind,]# extract bootstrap sample
  fit <- tree(EX ~ MET, data = data1, control = tree.control(nobs=nrow(State_data), minsize=8))
  final<-prune.tree(fit, best = 3)
  pred <- predict(final, newdata = State_data)
  return(pred)
}
res <- boot(State_data, f, R=1000) #make bootstrap
e <- envelope(res)  #compute confidence bands
```

```
# Plot of 95% confidence cands using Non-parametric Bootstrap
ggplot(State_data, aes(x=MET)) +
  geom_point(aes(y=EX), colour="brown", size=2) +
  geom_line(aes(y=res$t0, color="blue"),size=1.5) +
  geom_line(aes(y=e$point[2,], color="red"),size=1.4) +
  geom_line(aes(y=e$point[1,], color="red"),size=1.4) +
  ggtitle("Plot of 95% confidence bands using Non-parametric Bootstrap") +
  scale_color_discrete(name = "Type", labels = c("Fitted","Confidence band"))+
   theme_bw()

rng <- function(data, model) {
data1 <- data.frame(MET=data$MET, EX=data$EX) #database non ordered
n <- nrow(State_data)
data1$EX <- rnorm(n, predict(sel_tree, newdata = data1), sd(State_data$EX - sel_tree_pred))
return(data1)
}

# Computing bootstrap samples

f1 <- function(data1){
  fit <- tree(EX~MET, data = data1, control = tree.control(nobs = nrow(State_data), minsize=8))
  pruned_model <- prune.tree(fit, best = 3)
  final <- predict(pruned_model, newdata = State_data)
  return(final)
}

f2 <- function(data1){
  fit <- tree(EX~MET, data = data1, control = tree.control(nobs=nrow(State_data), minsize=8))
  pruned_model <- prune.tree(fit, best = 3)
  final <- predict(pruned_model, newdata = State_data)
  final_fit <- rnorm(nrow(data1), final, sd(resid(fit)))
  return(final_fit)
}

confidence_param <- boot(State_data, statistic=f1, R=1000, mle=sel_tree,
                         ran.gen=rng, sim="parametric")
e_conf <- envelope(confidence_param)

predicted_param <- boot(State_data, statistic=f2, R=1000, mle=sel_tree,
                        ran.gen=rng, sim="parametric")
e_pred <-envelope(predicted_param)

# Plot of 95% confidence and prediction bands using parametric bootstrap
ggplot(State_data, aes(x=MET)) +
  geom_point(aes(y=EX), colour="brown", size=2) +
  geom_line(aes(y=confidence_param$t0, color="blue"),size=1.4) +
  geom_line(aes(y=e_pred$point[2,], color="orange"),size=1.4) +
  geom_line(aes(y=e_pred$point[1,], color="orange"),size=1.4) +
  geom_line(aes(y=e_conf$point[2,], color="red"),size=1.4) +
  geom_line(aes(y=e_conf$point[1,], color="red"),size=1.4) +
  ggtitle("Plot of 95% confidence and prediction bands using Parametric Bootstrap")+
  scale_color_discrete(name = "Type", labels = c("Fitted","Prediction band","Confidence band"))+
  theme_bw()
```

```r
################################# Assignment 4 ###########################################
NIR_df <- read.csv("NIRspectra.csv",sep = ";",dec=",")
NIR_data<-NIR_df
#preprocessing
NIR_data$Viscosity<-c()

#pca
res<-prcomp(NIR_data)
#eigen values
lambda<-res$sdev^2
#proportion of variation, getting variance explained by each eigen vector
sprintf("%2.3f",lambda/sum(lambda)*100)
#plot
screeplot(res)
title("Screeplot of variation explained by each feature")
# top PC's and eigen values;
#Principal component loadings two_PC

two_PC<-res$rotation[,c(1,2)]
two_lambdas<-lambda[c(1,2)]

# plot of
ggplot(NIR_data)+geom_point(aes(x=res$x[,1],y=res$x[,2]),colour="green")+
  ggtitle("Plot of scores in the coordinates (PC1, PC2)")+theme_bw()+
  xlab("PC1")+ylab("PC2")

#PC1
ggplot()+geom_point(aes(x=1:nrow(two_PC),y=two_PC[,1]),colour="red")+
  ggtitle("Trace plot of PC 1")+xlab("Index of each feature")+
  ylab("PC1")+theme_bw()

#Pc2
ggplot()+geom_point(aes(x=1:nrow(two_PC),y=two_PC[,2]),colour="blue")+
  ggtitle("Trace plot of PC 2")+xlab("Index of each feature")+
  ylab("PC2")+theme_bw()
#4.3
library(fastICA)
set.seed(12345)
NIR_ICA <- fastICA(NIR_data, 2)
#Computing W_prime matrix
W_prime <- NIR_ICA$K %*% NIR_ICA$W
# Trace plots of two columns of W_prime

ggplot()+geom_point(aes(x=1:nrow(W_prime),y=W_prime[,1]),colour="red")+
  ggtitle("Trace plot for PC1")+theme_bw()+
  xlab("Index of each feature")

ggplot()+geom_point(aes(x=1:nrow(W_prime),y=W_prime[,2]),colour="blue")+
  ggtitle("Trace plot for PC2")+theme_bw()+
  xlab("Index of each feature")
# Plot of the scores of the first two latent features
ggplot()+geom_point(aes(NIR_ICA$S[,1],NIR_ICA$S[,2]),colour=c("green"))+
  ggtitle("Plot of the scores of the first two latent features")+
  theme_bw()
```