Computer Lab 2 block 1

Biswas Kumar, bisku859

April 01, 2020

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

The code input and references obtained from the lecture slides.(Since, it is easy to call with popular name, the stored vector or value has been kept with same name to the extent possible)

```
## Warning in RNGkind("Mersenne-Twister", "Inversion", "Rounding"): non-uniform
## 'Rounding' sampler used
```

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.

```
##  Misclasification rate (train data):  0.212
```

```
## .
##  Misclasification rate (test data):   0.268
```

```
##  Misclasification rate (Train data):  0.238
```

```
## .
##  Misclasification rate (Test data) :  0.372
```

Comparing the above calculated misclassification rate for both Deviance and Gini measures of impurity, we found that Devinance has considerably lower misclassification rate. It is therefore, Deviance should be choosen for better results.

(Comment : As instructed, I rechecked my report with group report. I ran the models above by mistake on "test" data to reproduce the model for prediction. Deleted the same and hence corrected the mistake)

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.

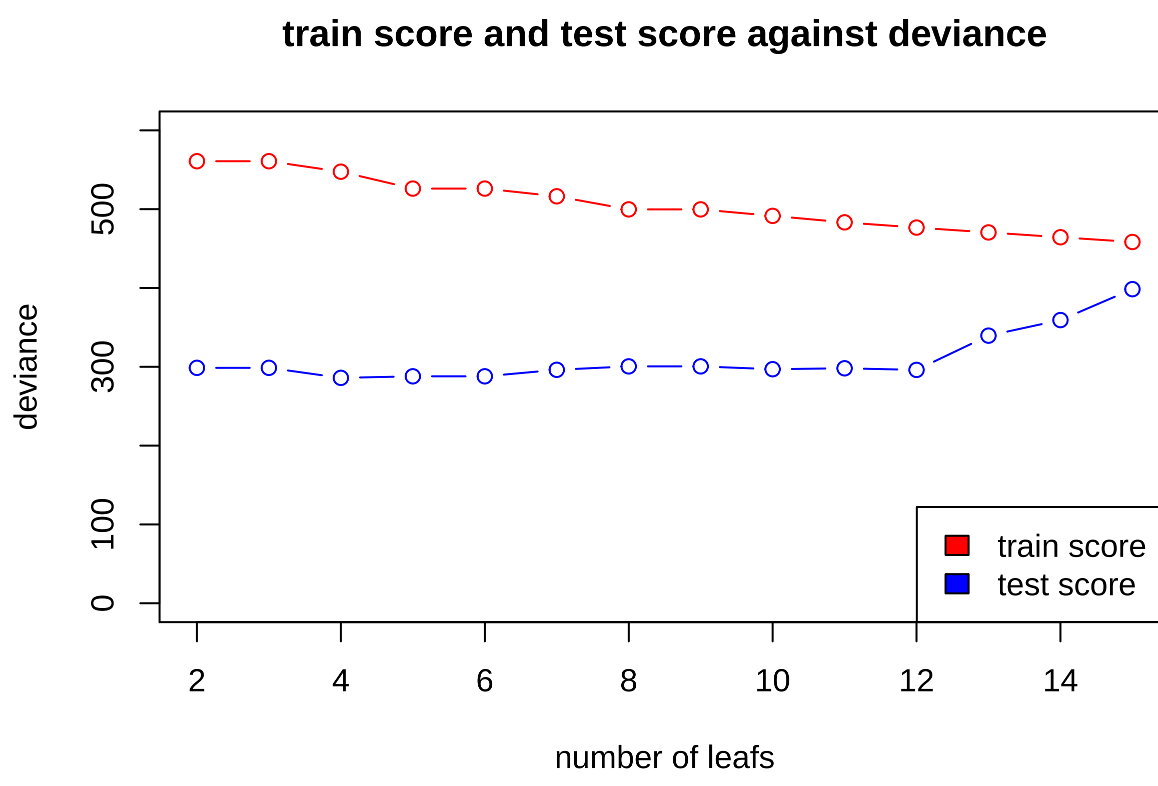
Selecting optimal tree by train/validation

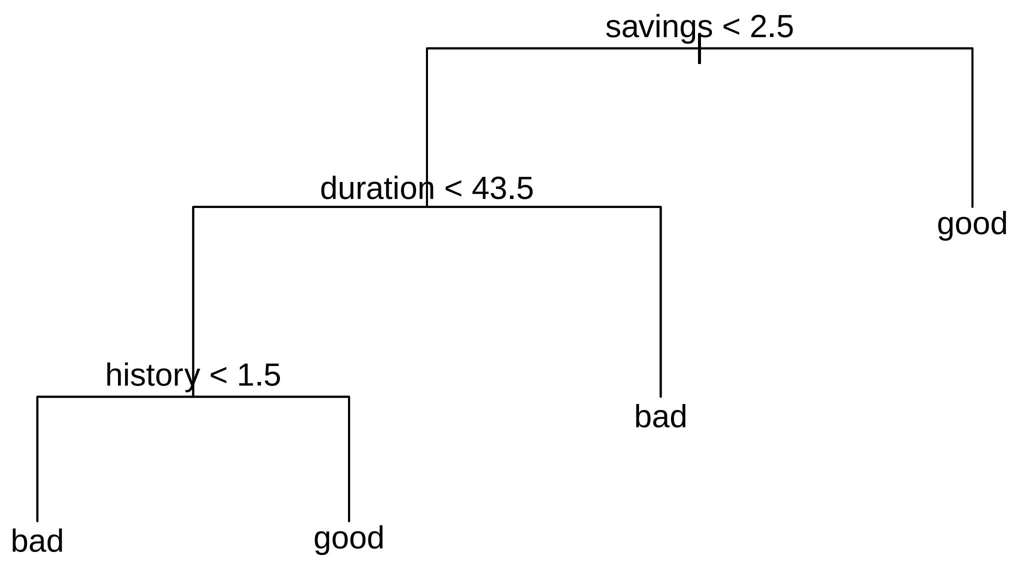
While referring to the lecture content, we get the optimal tree by train/validation as below : ## Steps

1. Finding the optimal tree depth using cv.res function
2. Generating trainscore and test score over relication method till optimal tree depth
3. Determining a nested sequence of subtrees of the supplied tree by recursively snipping off the least important splits by prune.tree() function
4. Plotting the trainscore & testscore to undestand the behaviour of the resultant datasets and interpretation
5. Calculation of misclassification rate of for the test data

```
## Warning in RNGkind("Mersenne-Twister", "Inversion", "Rounding"): non-uniform
## 'Rounding' sampler used
```

```
## $size
##  [1] 15 14 13 12 11 10  9  7  6  4  3  1
##
## $dev
##  [1] 817.7131 804.5495 804.0982 804.0982 792.6432 702.2004 702.2004 695.1741
##  [9] 669.3915 663.2478 640.7901 602.7423
##
## $k
##  [1]      -Inf  6.029735  6.112118  6.142687  6.635000  8.208589  8.231807
##  [8]  8.299712  9.806345 10.713529 13.285371 18.592320
##
## $method
## [1] "deviance"
##
## attr(,"class")
## [1] "prune"         "tree.sequence"
```


train score and test score against deviance
deviance
0
100
300
500
2
4
6
8
10
12
14
number of leafs
🟥 train score
🟦 test score



```
##        Yfit
##          bad good
##   bad    18   58
##   good    6  168
```

Upon the execution of cv.res, we got the size limit as 15. The optimal tree depth is the no. of leafs at which the model has least deviance. In the graph above. Looking at the graph above, we can say the optimum number of leafs of tree depth is 4.

comment: This was wrong in my previous reporting as I earlier considered size limit as optimum. The group report was too incomplete and didn't talked about optimum tree).

Now, the misclassification rate for test data as below :

```
##  Misclassification rate (test data) :  0.108
```

Upon running test data, we get the misclassification rate around 11 %.

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.

steps : 1.Conducting naiveBayes function on train dataset 2.Predicting the data and building up classification matrice using standard formula using TP,TN,FP,FN ratios.

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

## confusion Matrix Train    Misclasification rate  0.3

##       Yfit
##        bad good
##   bad   95   52
##   good  98  255

## TruePR 0.7223796     TrueNR 0.6462585    FalsePR 0.3537415   FalseNR 0.2776204

##       Yfit
##        bad good
##   bad   46   30
##   good  49  125

## confusion Matrix Test     Misclasification rate  0.316

##       Yfit
##        bad good
##   bad   46   30
##   good  49  125

## TruePR 0.7183908     TrueNR 0.6052632    FalsePR 0.3947368   FalseNR 0.2816092
```

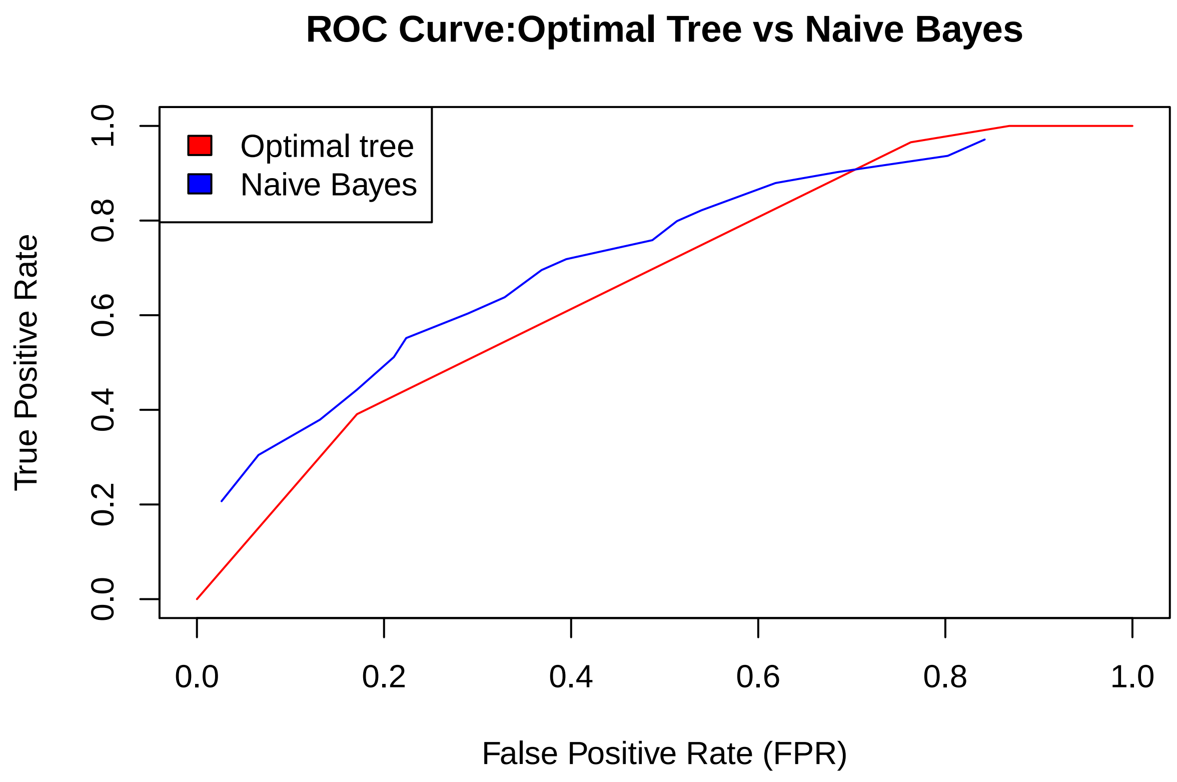
Referring to the above confusion matices and misclassification rate, we can observe that the misclassification rate is almost similar in training as well as test data. On test it is around 0.316(i.e 31.6%).

On comparision with misclassification rate of step 3 above (2.3) which has a misclassification error of only 11% , we can interpret that the optimal tree based classification is better than Naïve Bayes model as it gives less misclassification in this case.

5. Use the optimal tree and the Naïve Bayes model to classify the test data by using the given principle. Compute the TPR and FPR values for the two models and plot the corresponding ROC curves. Conclusion?

Solution

Since the plotted tree graph over train data has lowest devianvce value at no.of trees =4, the same shall be considered as optimal tree value.



Conclusionm of ROC curve : Looking at the graph we can say that the True Positive rate (TPR) of Naive Bayes is higher than Optimal tree model for most of the time against same FPR. It is therefore , we can say that Naive Bayes model works better for the given environment.

6. Repeat Naive Bayes classification as it was in step 4 but use the given loss matrix and report the confusion matrix for the training and test data.

```
## [1] "Loss matrix"
```

```
##      good bad
## good    0   1
## bad    10   0
```

```
## confusion Matrix (Train data)
```

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

```
## Misclassification rate_ train 0.546
```

```
## confusion Matrix (test data)
##       option
##        bad good
##   bad   71    5
##   good 122   52
```

```
## Misclassification rate_test 0.508
```

The misclassification rate of Naive Bayes classification based on loss matrix is in very high range (more than 50%) when compared with 30% range in step 4(without loss matrix).This suggests that the model with loss matrix gives worse result and should not be preferred for the given conditions.

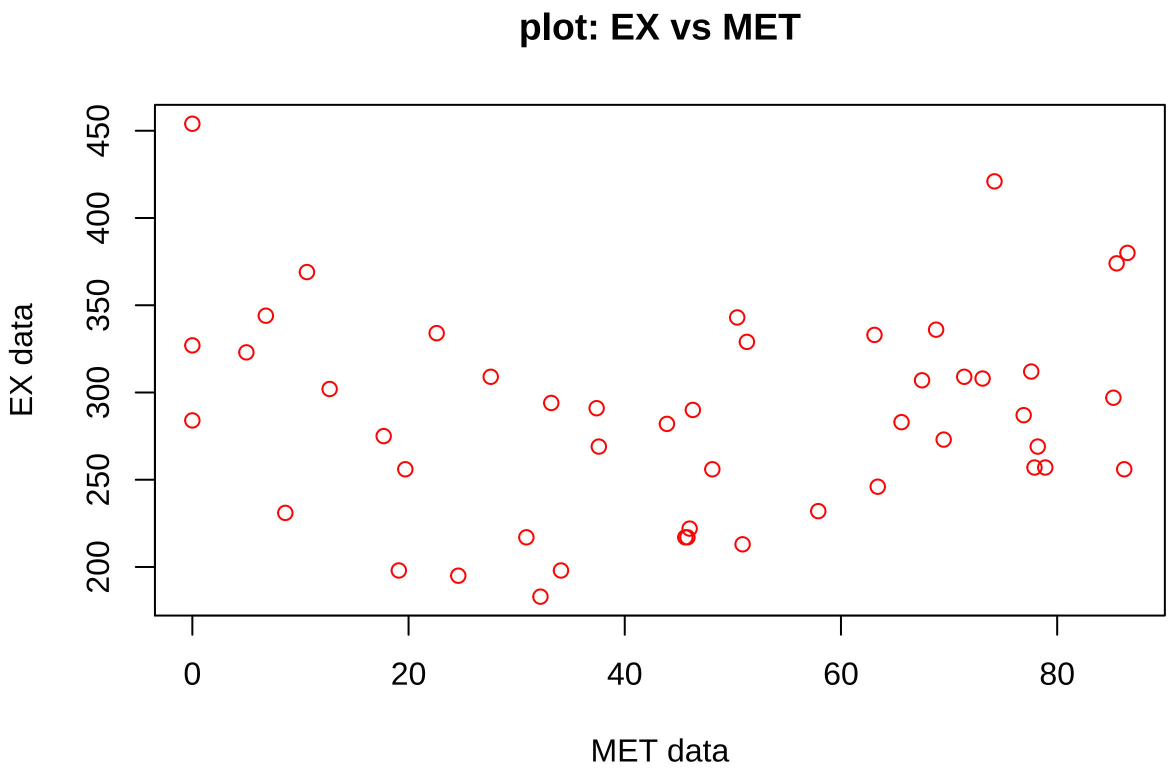
The most prominent reason seems to be the penalty associated with loss matrix that was imposed on the model, especially for False positive (10) which is 10 times the False negative(1).

Comment : I have rectified the model as earlier I have predicted on train data by mistake

Assignment 3.Uncertainty estimation

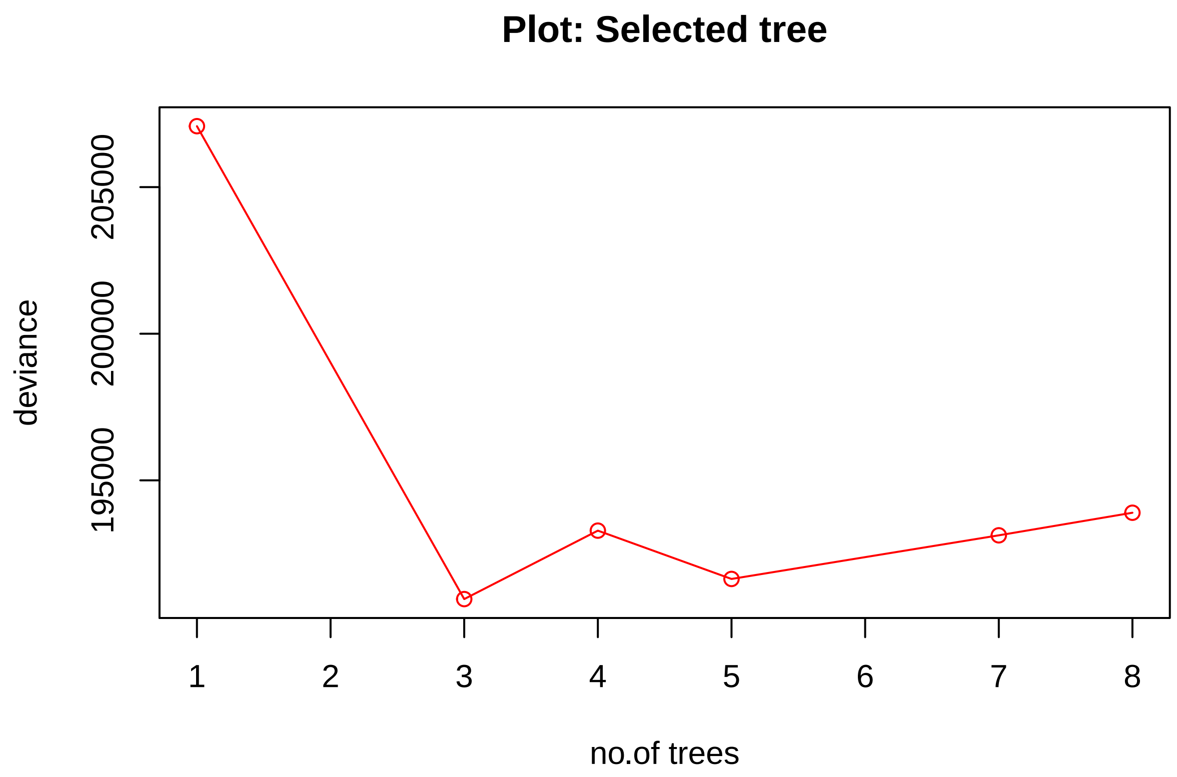
The data file State.csv contains per capita state and local public expenditures and associated state demographic and economic characteristics, 1960, and there are variables MET: Percentage of population living in standard metropolitan areas EX: Per capita state and local public expenditures

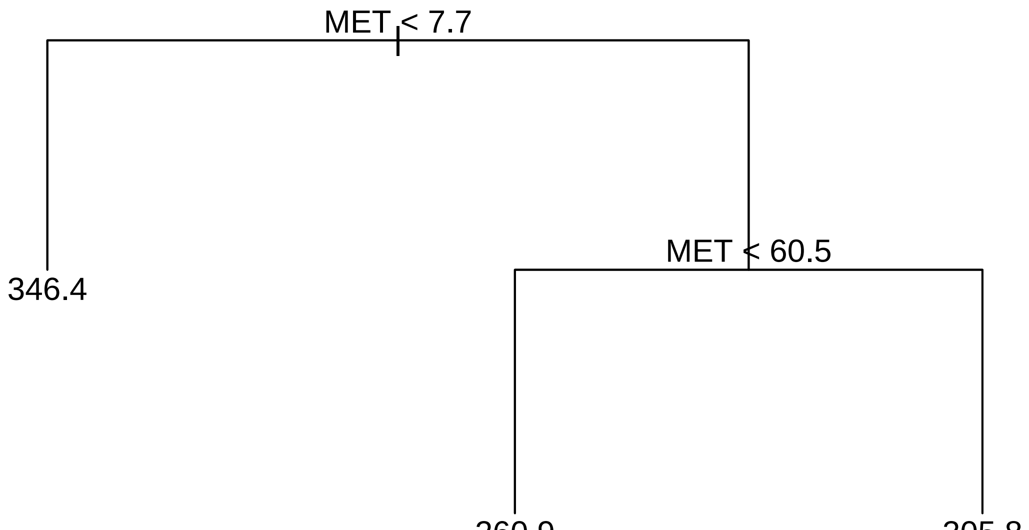
1. Reorder your data with respect to the increase of MET and plot EX versus MET. Discuss what kind of model can be appropriate here. Use the reordered data in steps 2 to 5.

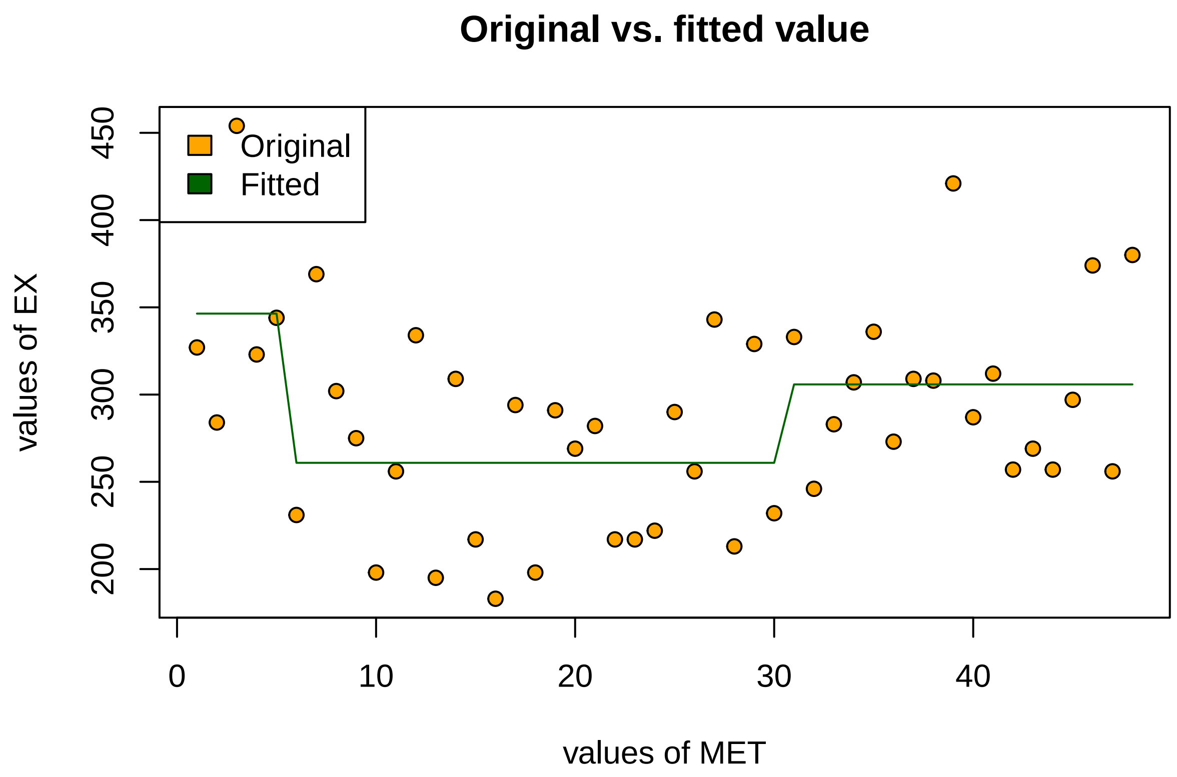


Looking at the plot we can see that there is no direct relationship or straight line certainly wont justify the fit. It seems we need to model with some degree of linear regression or try to fit regression tree model for good prediction.

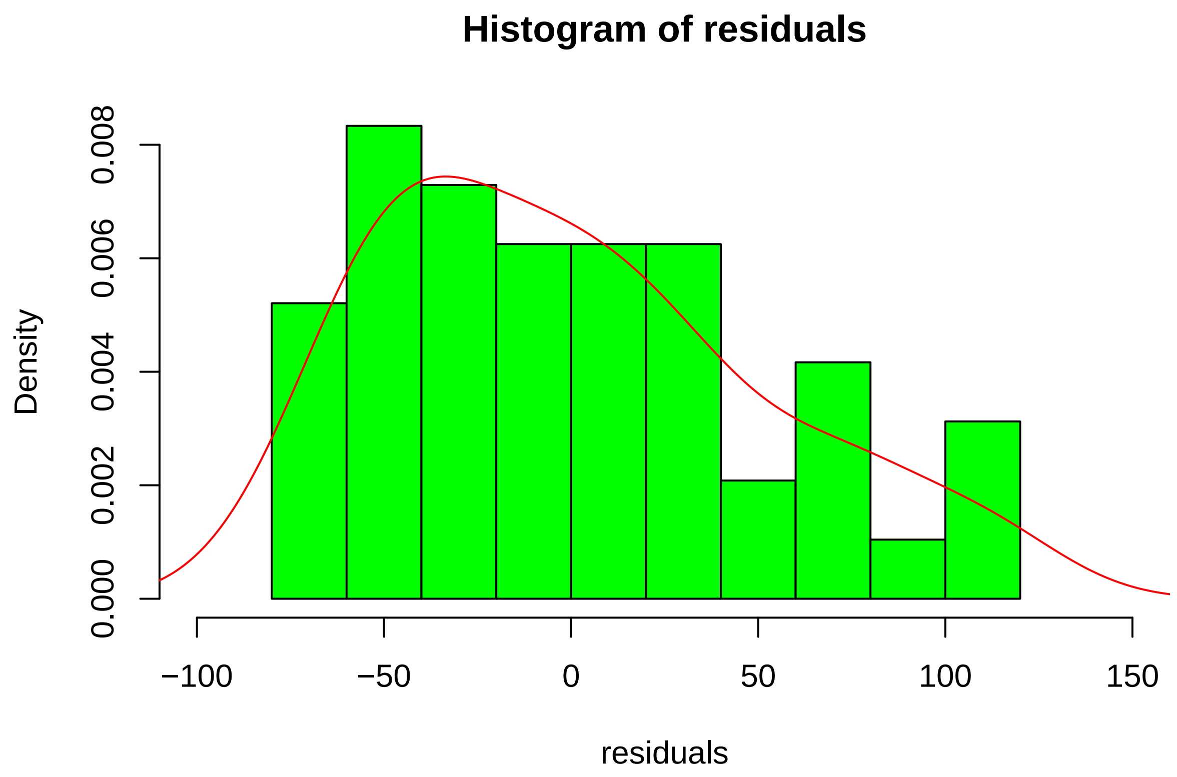
2 **Use package tree and fit a regression tree model with target EX and feature MET in which the number of the leaves is selected by cross-validation, use the entire data set and set minimum number of observations in a leaf equal to 8 (setting minsize in tree.control). Report the selected tree. Plot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







Original vs. fitted value
Original
Fitted
values of EX
values of MET
0
10
20
30
40
200
250
300
350
400
450

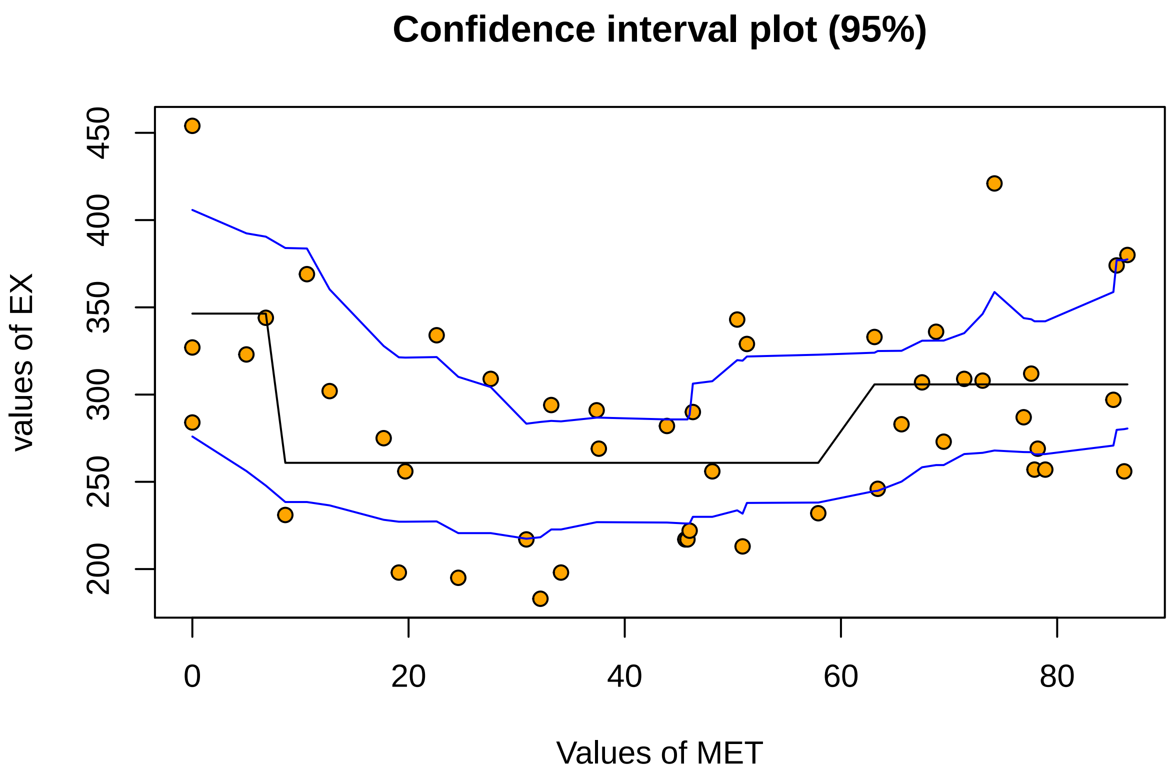


Looking at plot of original vs fitted, we see that the fitted is majorly divided into three levels.Also,it does not captures all the extreme values and generalise the fit overall.These levels majorly corresponds to the usage of mean values of data at leaves in the model for the same branch of tree.The other observation is that the histogram of residuals weighs towards left and on comparision of gamma distribution , it eventually does not represents a normal distribution.

We can say that the model captures the data averagely, it is not a very good fit for the given data set .

##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.

```
## Warning in prune.tree(tree_fit, best = 3): best is bigger than tree size
```

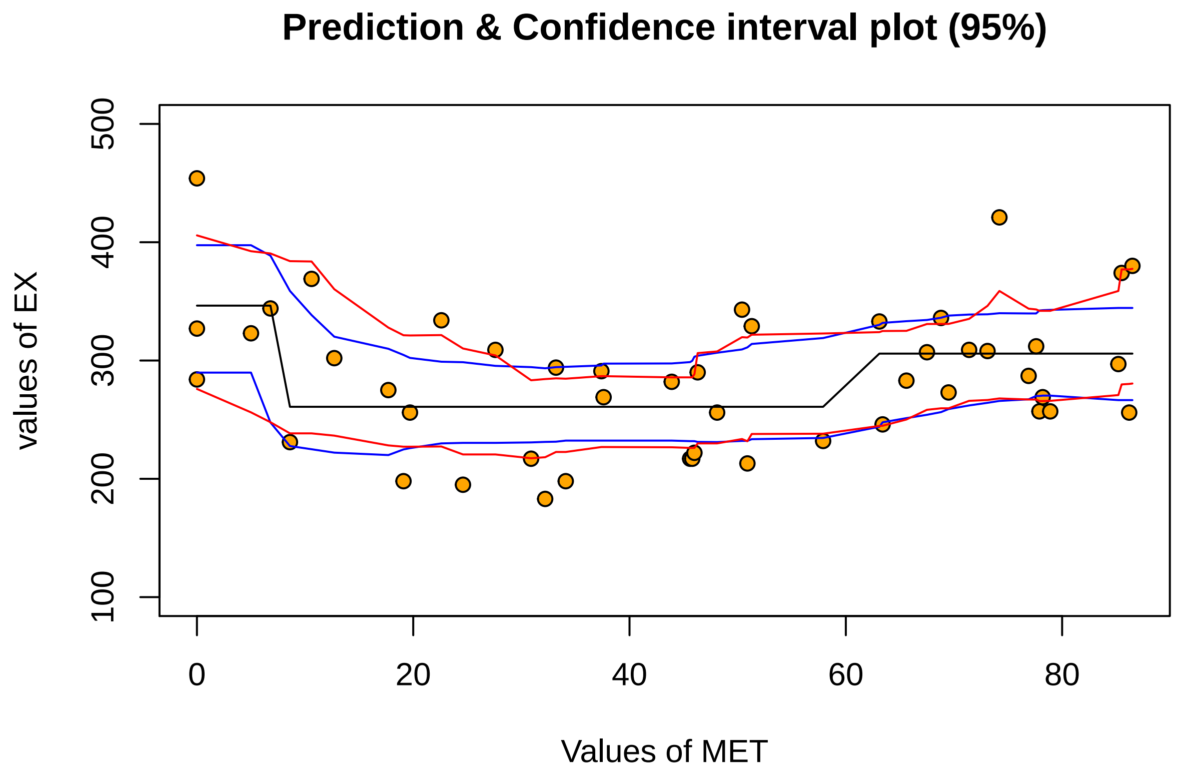


The plot is now updated with the 95% confidence bands. The upper blue line represents upper band while the lower line signifies the lower band.

The confidence band is also bumpy which highlights the relevance of histogram, weighs towards left and not a normal distribution.

Since, the predicted line is completely and entirely captured within the confidence band, we can say that results of the regression model in step 2 seem to be reliable.

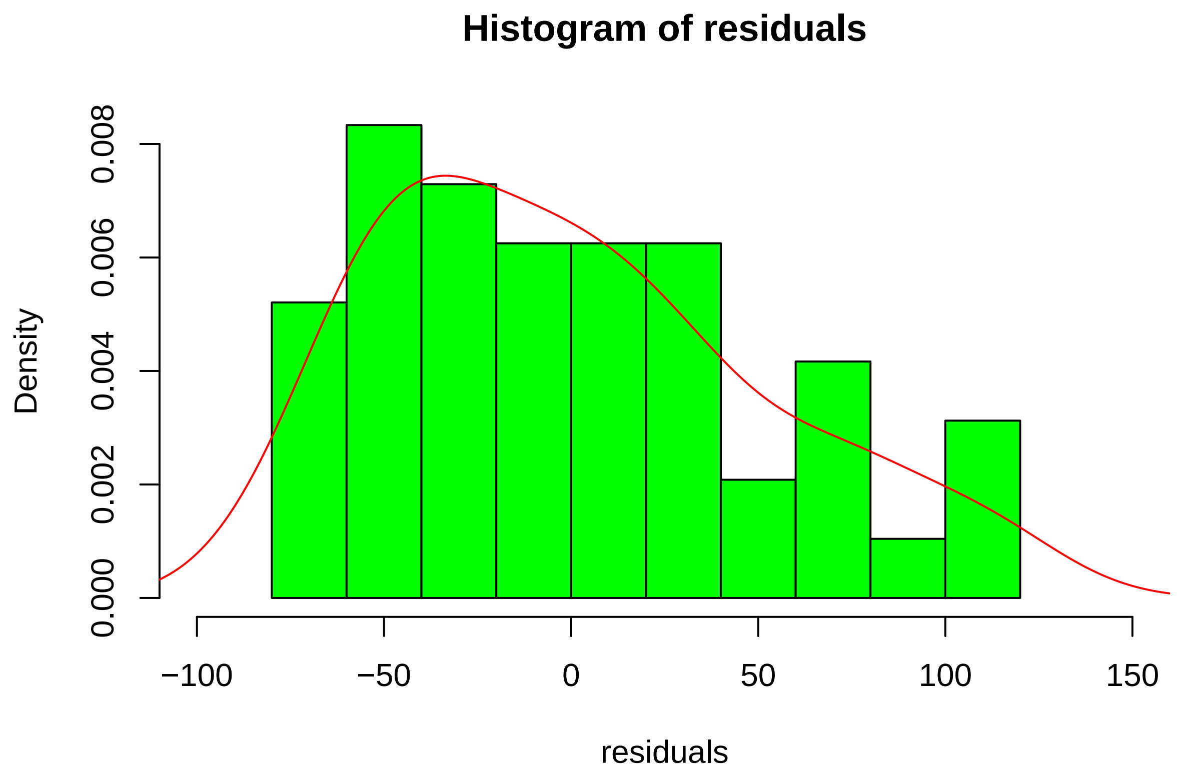
4. Compute and plot the 95% confidence and prediction bands the regression tree model from step 2 (fit a regression tree with the same settings and the same number of leaves as in step 2 to the resampled data) by using a parametric bootstrap. Consider the width of the confidence band and comment whether 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?



Looking at the plot, the width of confidence interval plot has shrunken which has a wide impact as more original values are now out of band. Though it still captures the predicted bumpy line well, it misses out more observation values. It is therefore, the confidence band is no more reliable for the purpose.

On the other hand, the prediction band looks really great. Visually, only one data point look outside of the band.Hence we can confidently say that only 5% of less of data are outside the prediction band.

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



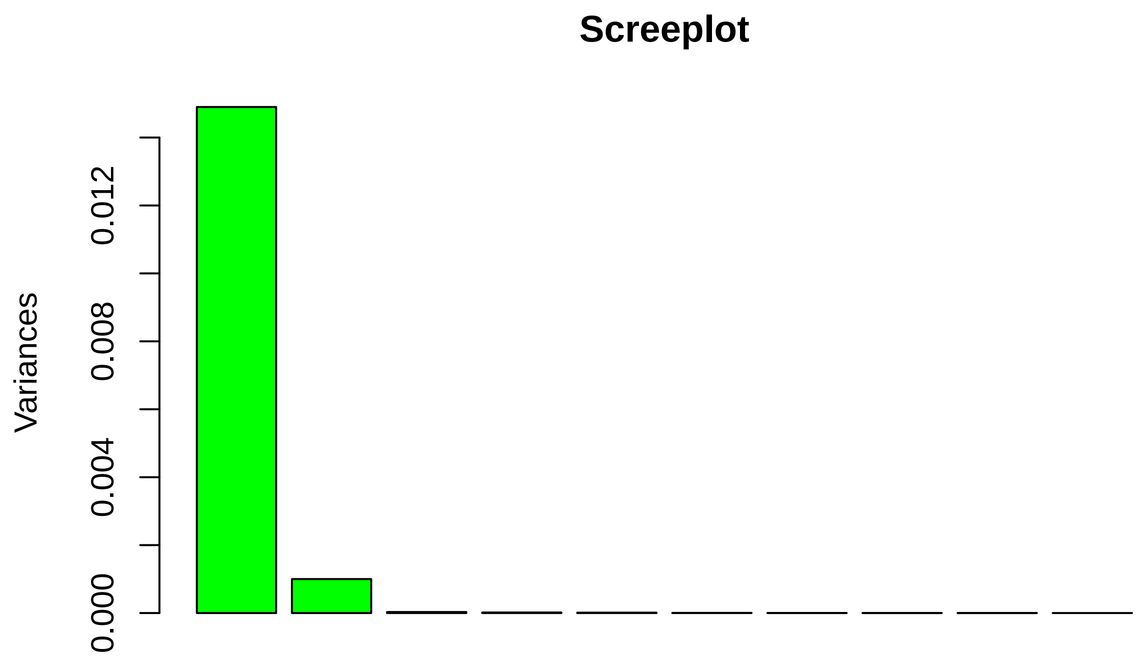
It seems that a non-parametric bootstrap will be more appropriate here as it be applied to any deterministic estimator, distribution-free.

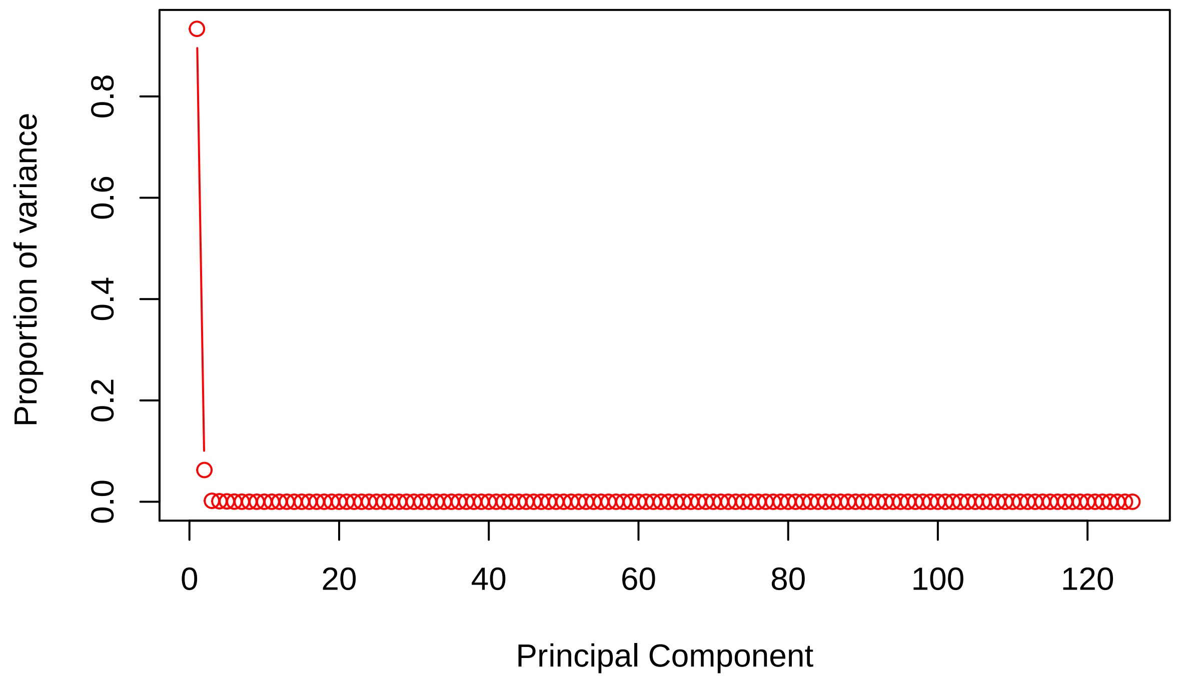
Assignment 4. Principal components

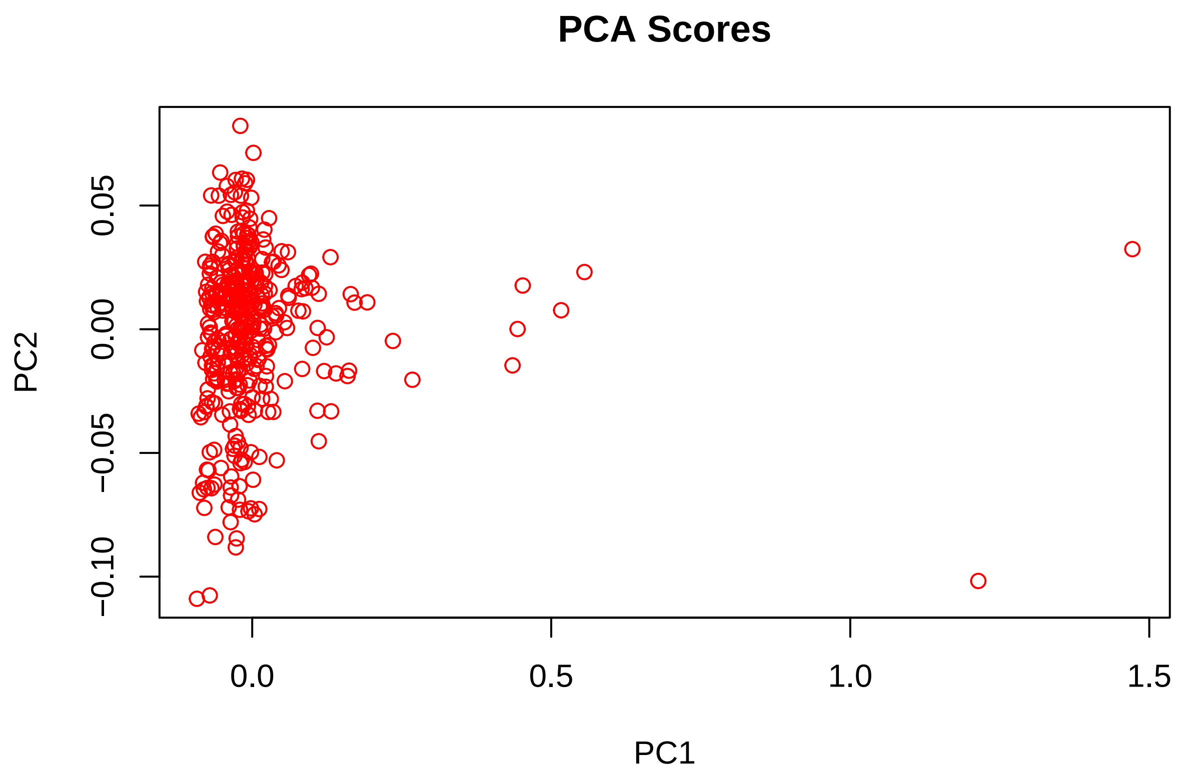
The data file NIRspectra.csv contains near-infrared spectra and viscosity levels for a collection of diesel fuels. Your task is to investigate how the measured spectra can be used to predict the viscosity.

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?

```
##   [1] "93.33" "6.26"  "0.19"  "0.10"  "0.07"  "0.02"  "0.01"  "0.00"  "0.00"
##  [10] "0.00"  "0.00"  "0.00"  "0.00"  "0.00"  "0.00"  "0.00"  "0.00"  "0.00"
##  [19] "0.00"  "0.00"  "0.00"  "0.00"  "0.00"  "0.00"  "0.00"  "0.00"  "0.00"
##  [28] "0.00"  "0.00"  "0.00"  "0.00"  "0.00"  "0.00"  "0.00"  "0.00"  "0.00"
##  [37] "0.00"  "0.00"  "0.00"  "0.00"  "0.00"  "0.00"  "0.00"  "0.00"  "0.00"
##  [46] "0.00"  "0.00"  "0.00"  "0.00"  "0.00"  "0.00"  "0.00"  "0.00"  "0.00"
##  [55] "0.00"  "0.00"  "0.00"  "0.00"  "0.00"  "0.00"  "0.00"  "0.00"  "0.00"
##  [64] "0.00"  "0.00"  "0.00"  "0.00"  "0.00"  "0.00"  "0.00"  "0.00"  "0.00"
##  [73] "0.00"  "0.00"  "0.00"  "0.00"  "0.00"  "0.00"  "0.00"  "0.00"  "0.00"
##  [82] "0.00"  "0.00"  "0.00"  "0.00"  "0.00"  "0.00"  "0.00"  "0.00"  "0.00"
##  [91] "0.00"  "0.00"  "0.00"  "0.00"  "0.00"  "0.00"  "0.00"  "0.00"  "0.00"
## [100] "0.00"  "0.00"  "0.00"  "0.00"  "0.00"  "0.00"  "0.00"  "0.00"  "0.00"
## [109] "0.00"  "0.00"  "0.00"  "0.00"  "0.00"  "0.00"  "0.00"  "0.00"  "0.00"
## [118] "0.00"  "0.00"  "0.00"  "0.00"  "0.00"  "0.00"  "0.00"  "0.00"  "0.00"
```




Proportion of variance
0.0
0.2
0.4
0.6
0.8
0
20
40
60
80
100
120
Principal Component



PCA basically gives us idea of components which are significant for further operation.It can be also looked as a technique for feature extraction (from multiple) for relevant analysis.

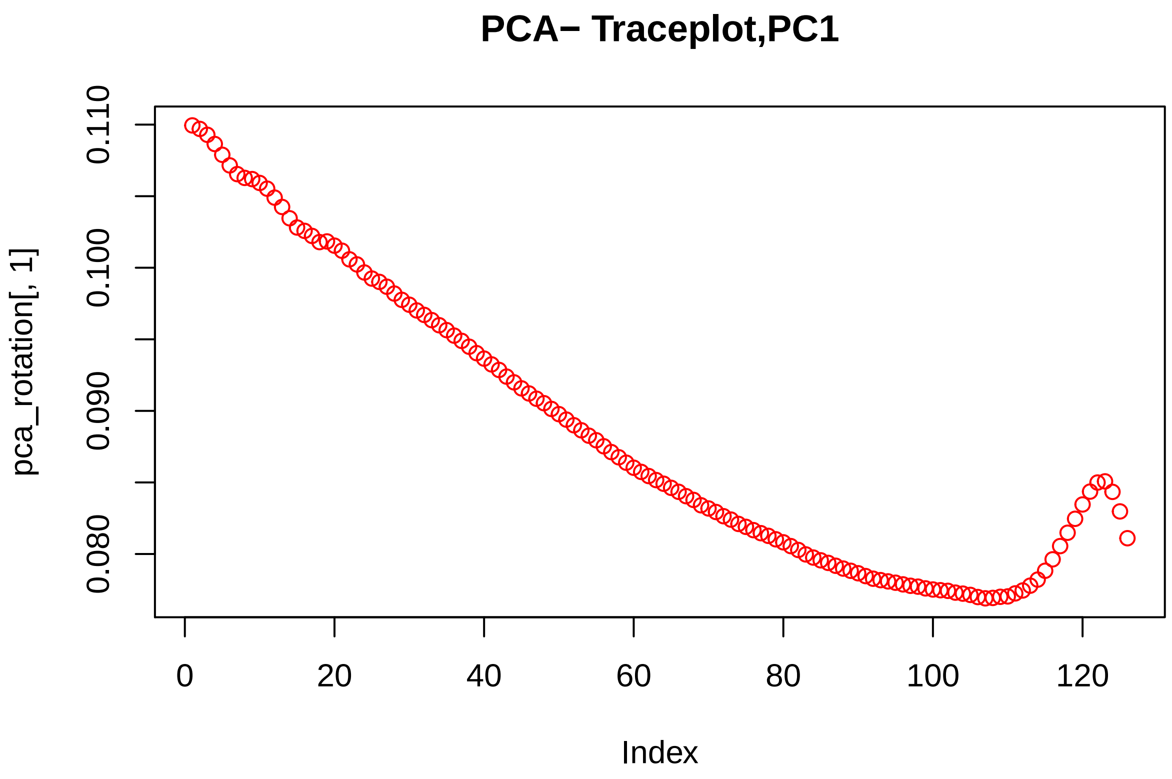
The weight distribution highlights that PC1 and PC2 adds upto 99.59% (93.33+6.26) of the total variance.It is therefore PC1 and PC2 can be extracted or illustrates around 99.6% of total variance.In short, using PCA, we try to find out the major component which explain the variability of data overall.

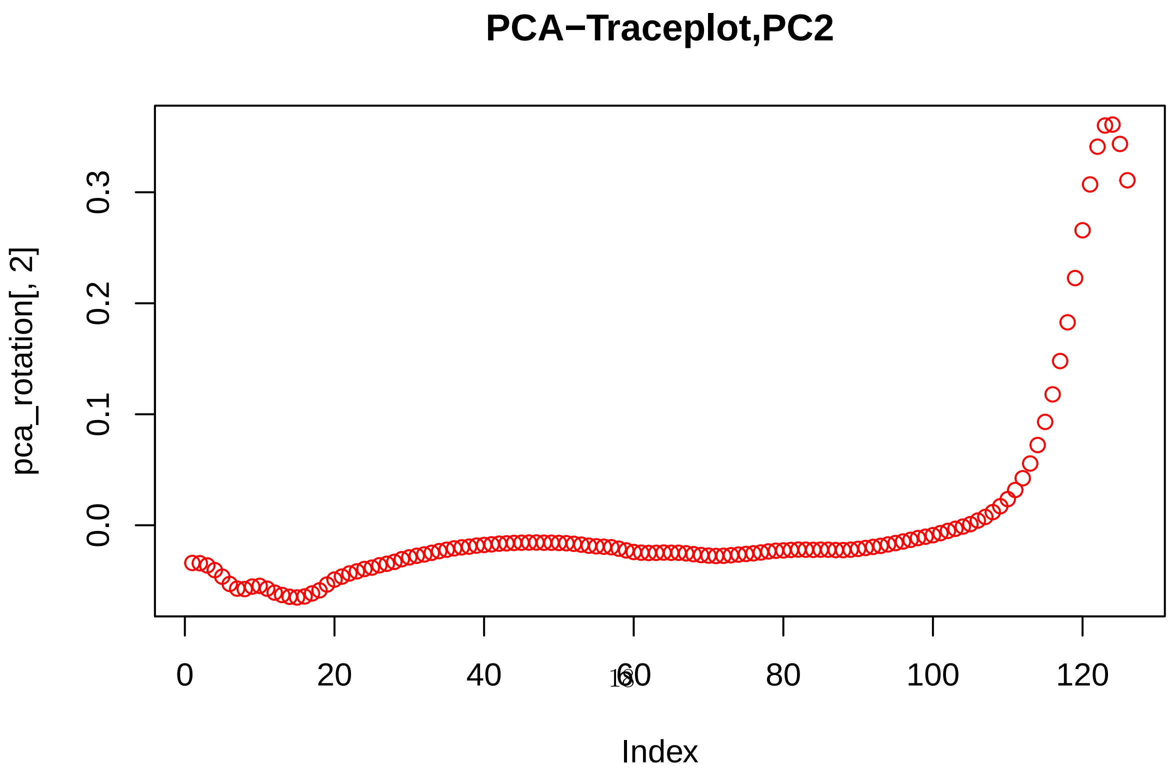
Now, looking at the plot , we can make out that all data points are in clusters with respect to features, except few(majorly 2 unusual diesel fuels) on the extreme right.

The screeplot which represents percentage of variations, highlights that majorly 2 are showing huge variances and rest are flat.This means that PC1 and PC2 (Principal component 1 and 2)can explain the major variance of the entire range.

Comment (The group report and plots were correct but didnot explained the PCA and related plots)

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?





The traceplots here signifies the importance of features towards relevance of Principal component.Higher the rotation (i.e loadings), more significantly it conttributes towards PC.

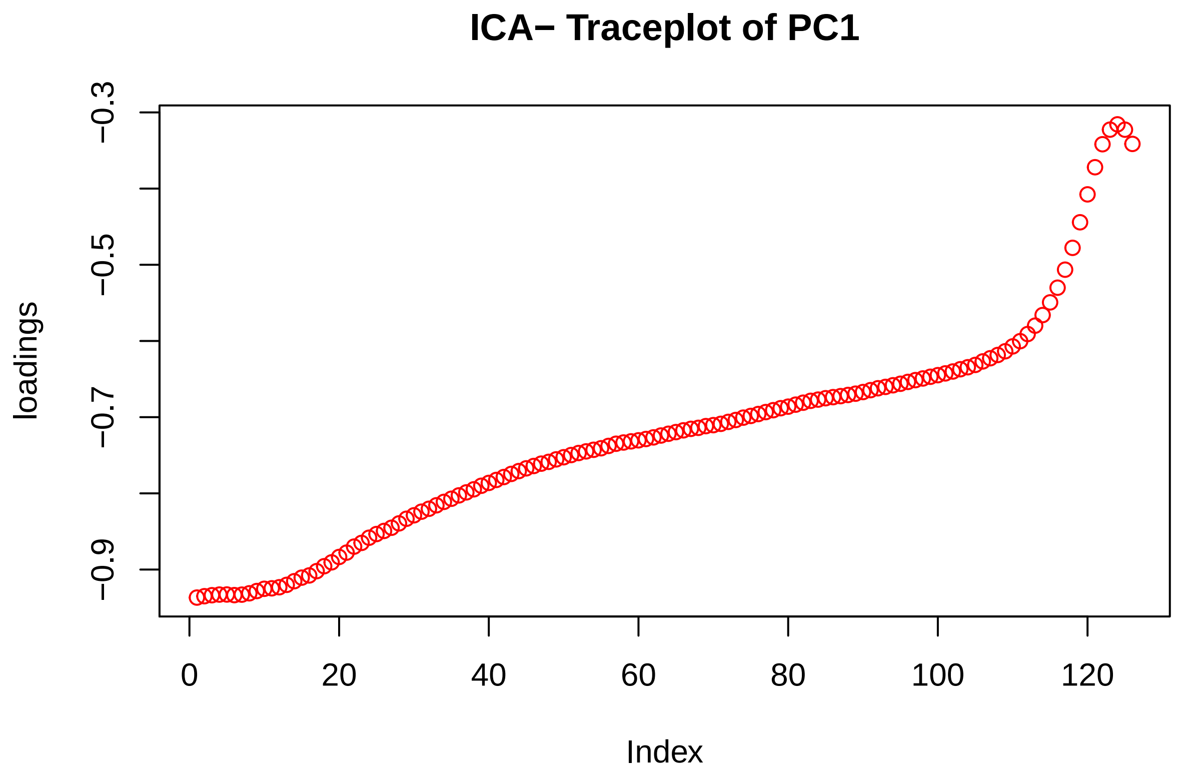
PC1 has more features that contributes when compared to PC2. Many of the original features of PC2 are almost zero and therefore it fails to signifies PC majorly.They seems to be orthagonal in nature.

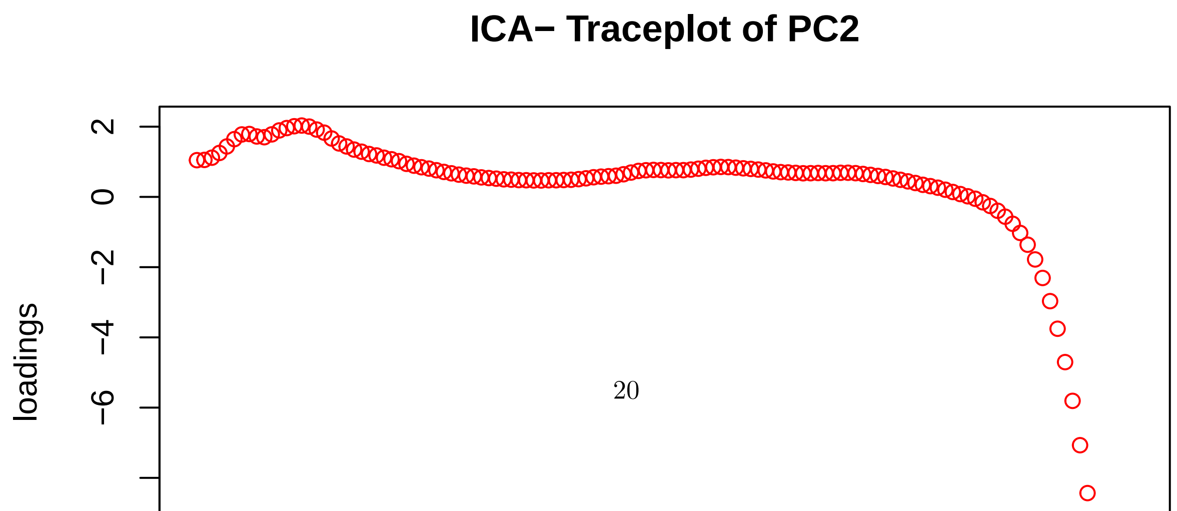
Comment (The plots in group report was correct but didnot explained plots and observations)

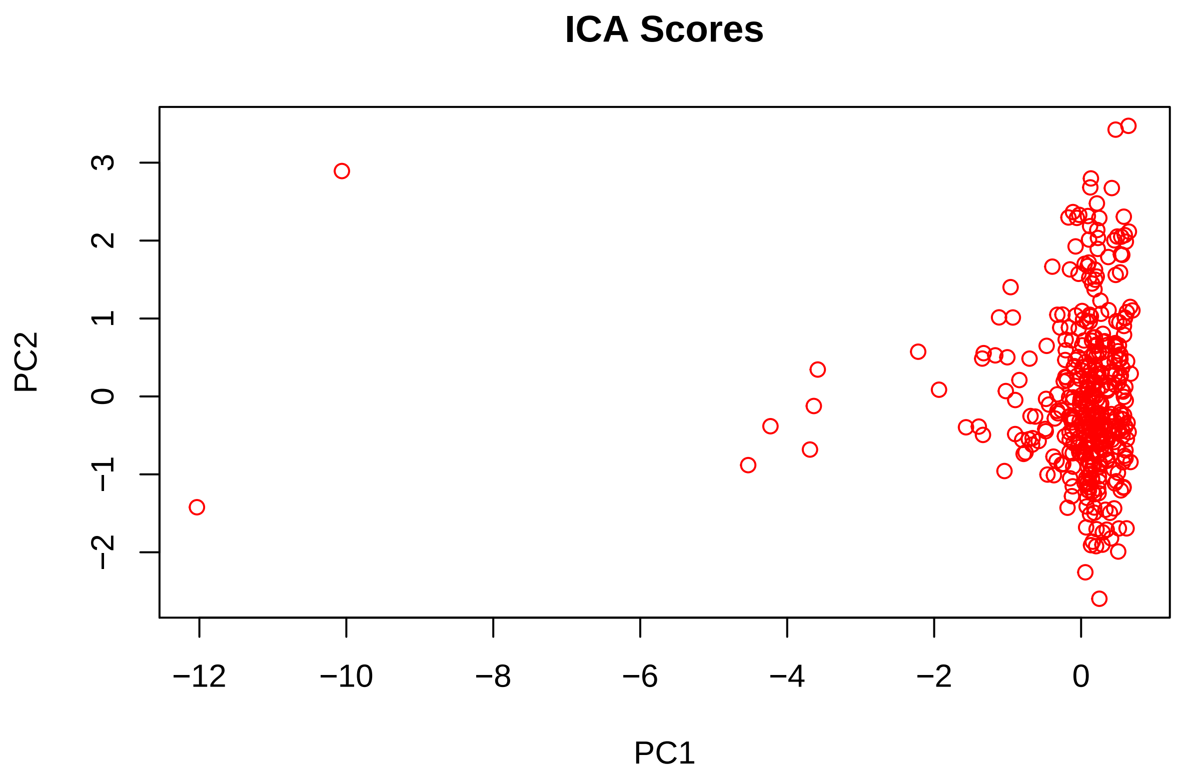
3. Perform Independent Component Analysis with the number of components selected in step 1 (set seed 12345). Check the documentation for the fastICA method in R and do the following:

a. Compute W_dash= KW and present the columns W_dash in form of the trace plots. Compare the trace plots in step 2 and make conclusions. Whats is represented by the matrix W_dash ?

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







The trace plots of PCA and ICA for both components looks like inverted to each other and on different scale.The inverted graph shape basically depicts the core nature of PCA (explains highest or maximum variances)and ICA(explains Independent or un-correlated).

The matrix W_dash reflects the correlation of features with Principal component 1 and 2 , once the ICA model is generated.

It may be noted that ICA or Independent Component Analysis is a re-generative model which maximizes independence.We can therefore see the inverted mirror images of PCA and ICA score plot analysis as well but on diffenet scale (one in positive and other on negative scale).

Comment : The exercise was missing in the submitted group report

Code Appendix

```
knitr::opts_chunk$set(echo = TRUE)
library("readxl")
library(tree)
library(rpart)
library(rpart.plot)
library(MASS)
library(e1071)
library("boot")
library(fastICA)
library("dplyr")
```

```r
#as illustrated in lecture slide 1e

data <- read_excel("creditscoring.xls")
#data
data$good_bad <- as.factor(data$good_bad)
n=dim(data)[1]
RNGversion('3.5.1') # rounding sampler
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,]
# Deviance misclassification on training data
dev<- tree(formula= good_bad ~. , data= train,split = "deviance")
dev_projection <- predict(dev,newdata =train, type="class")
#dev_projection
plan=table(train$good_bad, dev_projection)
#True Positive, True Negative, False Positive & False Negative calculation
TrueP=plan[2,2];TrueN=plan[1,1];FalseN=plan[2,1];FalseP=plan[1,2]
cat(" Misclasification rate (train data): ",(FalseP + FalseN)/ sum(plan))



# Deviance misclassification on test data
# dev<- tree(formula= good_bad ~. , data= test,split = "deviance") ,removed the incorrect data set
dev_projection <- as.factor(dev_projection)
dev_projection <- predict(dev,newdata =test, type="class")
#dev_projection
plan=table(test$good_bad, dev_projection)
#True Positive, True Negative, False Positive & False Negative calculation
TrueP=plan[2,2];TrueN=plan[1,1];FalseN=plan[2,1];FalseP=plan[1,2]
cat(".\n Misclasification rate (test data):  ",(FalseP + FalseN)/ sum(plan))



# Gini misclassification on training data
gini<- tree(formula= good_bad ~. , data= train,split = "gini")
gini_projection <- predict(gini,newdata =train, type="class")
#gini_projection
plan=table(train$good_bad, gini_projection)
#True Positive, True Negative, False Positive & False Negative calculation
TrueP=plan[2,2];TrueN=plan[1,1];FalseN=plan[2,1];FalseP=plan[1,2]
cat(" Misclasification rate (Train data): ",(FalseP + FalseN)/ sum(plan))
# Misclassification_error <- (FalseP + FalseN)/ sum(plan)



# Gini misclassification on test data
# gini<- tree(formula= good_bad ~., data= test,split = "gini"), removed the incorrect data set
```

```r
gini_projection <- predict(gini,newdata =test, type="class")
#gini_projection
plan=table(test$good_bad, gini_projection)
#True Positive, True Negative, False Positive & False Negative calculation
TrueP=plan[2,2];TrueN=plan[1,1];FalseN=plan[2,1];FalseP=plan[1,2]
cat(".\n Misclasification rate (Test data) : ",(FalseP + FalseN)/ sum(plan))
# Misclassification_error <- (FalseP + FalseN)/ sum(plan)

fit=tree(formula= good_bad ~., data=train)
RNGversion('3.5.1')
set.seed(12345)
cv.res=cv.tree(fit)
cv.res
fit=tree(formula= good_bad ~., data=train)
# we will incorporate the same (size)to find the trainscore and test score.
#The coding methodology is referred from the lecture materials.
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)
}
 fit <- tree(formula= good_bad ~., data=train)
 #rpart.plot(tree, box.palette="RdBu", shadow.col="gray",nn=TRUE)

plot(2:15, trainScore[2:15], type="b", col="red",ylim=c(0,600),
     main="train score and test score against deviance",
     xlab="number of leafs",ylab="deviance")
points(2:15, testScore[2:15], type="b", col="blue")
legend('bottomright',legend=c("train score ","test score"),fill=c("red","blue"))

# Optimal tree calculation : As per lecture slide
# best is minimum deviance  of testscore
finalTree<-prune.tree(fit,best=which( testScore== min(testScore[2:15])))
plot(finalTree)
text(finalTree,pretty = 0)
Yfit=predict(finalTree, newdata=test,type="class")
table(test$good_bad,Yfit)



datatest<- tree(formula= good_bad ~. , data= test)
datap <- predict(datatest,newdata =test, type="class")
#datap
plan=table(test$good_bad, datap)
#True Positive, True Negative, False Positive & False Negative calculation
TrueP=plan[2,2];TrueN=plan[1,1];FalseN=plan[2,1];FalseP=plan[1,2]
cat("\tMisclassification rate (test data) : ",(FalseP + FalseN)/ sum(plan))
```

```r
fit=naiveBayes(formula= good_bad ~., data=train)
Yfit=predict(fit, newdata=train,type="class")
tab=table(train$good_bad, Yfit)
tab
TrueP=tab[2,2];TrueN=tab[1,1];FalseN=tab[2,1];FalseP=tab[1,2]
cat("confusion Matrix Train ","\t Misclasification rate ",
    (FalseP + FalseN)/sum(tab))
tab
cat("TruePR",TrueP/(TrueP+FalseN),
    "\tTrueNR", TrueN/(TrueN+FalseP),
    "\tFalsePR",FalseP/(FalseP+TrueN),
    "\tFalseNR",FalseN/(FalseN+TrueP))

# fit=naiveBayes(formula= good_bad ~., data=test)   # update  train only on training data
Yfit=predict(fit, newdata=test,type="class")
tab=table(test$good_bad, Yfit)
tab
TrueP=tab[2,2];TrueN=tab[1,1];
FalseN=tab[2,1];FalseP=tab[1,2]
cat("confusion Matrix Test ","\t Misclasification rate ",
    (FalseP + FalseN)/sum(tab))
tab
cat("TruePR",TrueP/(TrueP+FalseN), "\tTrueNR",
    TrueN/(TrueN+FalseP),
    "\tFalsePR",FalseP/(FalseP+TrueN),
    "\tFalseNR",FalseN/(FalseN+TrueP))


pi=seq(0.05,0.95,0.05) # value of all pi
optimal_fit=tree(formula= good_bad ~., data=train)
opt_tree=prune.tree(optimal_fit,best=4) #optimal tree value is 4
pred=predict(opt_tree, newdata=test)

tpr=vector(length=length(pi))
fpr=vector(length=length(pi))
for(d in 1:length(pi)){
  y_hat<-c()
for(i in 1:length(pred[,2])){
y_hat[i]<-if(pred[i,2]>pi[d]){"good"}else{"bad"}}
y_hat<-factor(y_hat,levels=c("bad","good"))
tab=table(test$good_bad,y_hat)
tab
TrueP=tab[2,2];TrueN=tab[1,1];FalseN=tab[2,1];FalseP=tab[1,2]

tpr[d]<-(TrueP/(TrueP+FalseN))
fpr[d]<-(FalseP/(FalseP+TrueN))

class(test$good_bad)
class(y_hat)
head(as.character(test$good_bad))
head(y_hat)
}
# not printing tpr and fpr values anymore on the report
```

```r
#cat("\nTrue Positve Rate_Optimal tree model","\n",tpr) # tpr values
#cat("\nFalse Positve Rate_Optimal tree model","\n",fpr) # fpr values


#Naive Bayes model
naive_fit=naiveBayes(formula= good_bad ~., data=train)
naive_predict=predict(naive_fit, newdata=test,"raw")



tpr_naive=vector(length=length(pi))
fpr_naive=vector(length=length(pi))
for(d in 1:length(pi)){
  y_hat<-c()
for(i in 1:length(naive_predict[,2])){
y_hat[i]<-if(naive_predict[i,2]>pi[d]){"good"}else{"bad"}}
y_hat<-factor(y_hat,levels=c("bad","good"))
tab=table(test$good_bad,y_hat)
tab
TrueP=tab[2,2];TrueN=tab[1,1];FalseN=tab[2,1];FalseP=tab[1,2]
#cat("confusion Matrix Test ","\t Misclasification rate ",(FalseP + FalseN)/sum(tab))
tpr_naive[d]<-(TrueP/(TrueP+FalseN))
fpr_naive[d]<-(FalseP/(FalseP+TrueN))

class(test$good_bad)
class(y_hat)
head(as.character(test$good_bad))
head(y_hat)
}
# not printing tpr and fpr values anymore on the report
#cat("\nTrue Positve Rate_Naive","\n",tpr_naive) # tpr values
#cat("\nFalse Positve Rate_Naive","\n",fpr_naive) # fpr values

#combining both curve as below
plot(x=fpr,y=tpr,type="l",col="red",xlab="False Positive Rate (FPR)",
     ylab = "True Positive Rate",main="ROC Curve:Optimal Tree vs Naive Bayes")
lines(fpr_naive,tpr_naive,type="l",col="blue")
legend('topleft',legend=c("Optimal tree","Naive Bayes"),fill=c("red","blue"))

naive_fit=naiveBayes(formula= good_bad ~., data=train)
naive_predict=predict(naive_fit, newdata=train,"raw")
#naive_predict
#creation of loss matrix as given in exercise
loss_matrix= matrix(c(0,10,1,0),ncol =2)
rownames(loss_matrix)=c("good","bad")
colnames(loss_matrix)=c("good","bad")
print("Loss matrix")
loss_matrix

option_1=naive_predict[,1]/naive_predict[,2]
option_2=loss_matrix[1,2]/loss_matrix[2,1]
option<-c()
for(i in 1:length(option_1)){
```

```
option[i]<-if(option_1[i] > option_2){"bad"}else{"good"}}


option_final<-factor(option,levels=c("bad","good"))
tab=table(train$good_bad,option)
misclassification_rate<- 1-(sum(diag(tab))/sum(tab))
cat("confusion Matrix (Train data) ","\n")
tab
cat("Misclassification rate_ train", misclassification_rate,"\n")

# Naives model of test data

# comment : deleted wrong model which i built on test data by mistake

naive_predict=predict(naive_fit, newdata=test,"raw")
#naive_predict
#creation of loss matrix as given in exercise
#loss_matrix= matrix(c(0,10,1,0),ncol =2)
#loss_matrix
option_1<-naive_predict[,1]/naive_predict[,2]
option_2<-loss_matrix[1,2]/loss_matrix[2,1]
option<-c()
i<-1
while(i<=length(option_1))
{option[i]<-if(option_1[i]> option_2){"bad"}else{"good"}
  i<-i+1}

#option
option<-factor(option,levels=c("bad","good"))

#length(test$good_bad)

tab=table(test$good_bad,option)
misclassification_rate<- 1-(sum(diag(tab))/sum(tab))
cat("confusion Matrix (test data)","\n")
tab
cat("Misclassification rate_test", misclassification_rate)



data <- read.csv2("State.csv")
#Re-ordering of data (dataframe) wrt increase in MET
data_reorder<-data[order(data$MET),]
#data_reorder
plot(x=data_reorder$MET,y=data_reorder$EX,xlab = "MET data",
     ylab = "EX data",main="plot: EX vs MET",col="red")


#Regression tree model using tree package,target EX and feature MET
set.seed(12345)
tree_fit=tree(formula = EX~MET,data=data_reorder,
              control=tree.control(minsize=8,nobs = length(data_reorder[,1])))
#Cross valivation for selection of number of leaves
```

```r
cv_tree<-cv.tree(tree_fit)
#plot of selected tree
plot(x=cv_tree$size,y=cv_tree$dev,xlab="no.of trees",
     ylab="deviance",main="Plot: Selected tree",type="o",col="red")
# It can be observed from graph that 3 leaves is the minimum as well as best for selection
prunedTree=prune.tree(tree_fit,best=3)
pred=predict(prunedTree, newdata=data_reorder) #fitted value
plot(prunedTree)
text(prunedTree,pretty = 0)
Yfit=predict(prunedTree)

plot(x=c(1:nrow(data_reorder)),y=data_reorder$EX,pch=21,
     bg="orange",
     xlab = "values of MET",ylab = "values of EX",
     main = "Original vs. fitted value")
points(Yfit,col="dark green",type="l")
legend('topleft',legend=c("Original","Fitted"),
       fill=c("orange","dark green"))


residuals=as.numeric(resid(prunedTree)) # original - predicted
hist(residuals,col = "green",xlim = c(-100,150),prob=TRUE)# residuals against density
# as per lecture note
lines(density(residuals),col="red")

# computing bootstrap samples, from the lecture slide bootstrap- non -parametric bootstrap
f=function(data,ind){
data1=data[ind,]# extract bootstrap sample
#fit linear model
tree_fit=tree(formula = EX~MET,data=data1,
              control=tree.control(minsize=8,nobs = length(data_reorder[,1])))
prunedTree=prune.tree(tree_fit,best=3)
pred=predict(prunedTree, newdata=data_reorder) #fitted value
return(pred)
}
res=boot(data_reorder, f, R=1000) #make bootstrap
#Bootstrap cofidence bands for linearmodel
e=envelope(res) #compute confidence bands 95%
#fit linear model
tree_fit=tree(formula = EX~MET,data=data_reorder,
              control=tree.control(minsize=8,nobs = length(data_reorder[,1])))
prunedTree=prune.tree(tree_fit,best=3)
pred=predict(prunedTree, newdata=data_reorder) #fitted value
plot(x=data_reorder$MET,y=data_reorder$EX, pch=21,
     bg="orange",main="Confidence interval plot (95%)",
     xlab="Values of MET",ylab="values of EX")
points(data_reorder$MET,pred,type="l") #plot fitted line
points(data_reorder$MET,e$point[2,], type="l", col="blue")
points(data_reorder$MET,e$point[1,], type="l", col="blue")



# parametric bootstrap as illustrated in lecture slide
```

```r
#area = Vlaues of MET, Price= values of EX, data_reorder = data1
 #fit linear model
tree_fit=tree(formula = EX~MET,data=data_reorder,
              control=tree.control(minsize=8,nobs = length(data_reorder[,1])))
mle=prune.tree(tree_fit,best=3)
rng=function(data, mle) {
  data1=data.frame(EX=data_reorder$EX, MET=data_reorder$MET)
n=length(data_reorder$EX)
#generate new EX
data1$EX=rnorm(n,predict(mle, newdata=data1),sd(resid(mle)))
return(data1)
}

f1=function(data1){
  #fit linear model on data 1 now
tree_fit=tree(formula = EX~MET,data=data1,
              control=tree.control(minsize=8,nobs=length(data_reorder[,1])))
res=prune.tree(tree_fit,best=3)
#predict values for all Area values from the original data
Ex_Predict=predict(res,newdata=data_reorder)
return(Ex_Predict)
                  }
res=boot(data_reorder,statistic=f1, R=1000, mle=mle,ran.gen=rng,sim="parametric")
e2 <- envelope(res)
#fit linear model
tree_fit=tree(formula = EX~MET,data=data_reorder,
              control=tree.control(minsize=8,nobs = length(data_reorder[,1])))
mle=prune.tree(tree_fit,best=3)
f1=function(data1){
  tree_fit=tree(formula = EX~MET,
                data=data1,
                control=tree.control(minsize=8,nobs=length(data_reorder[,1])))
  res=prune.tree(tree_fit,best=3)
#res=lm(Price~Area, data=data1) #fit linear model
#predict values for all Area values from the original data
predict_EX=predict(res,newdata=data_reorder)
n=length(data_reorder$EX)
predictedEX=rnorm(n,predict_EX,sd(resid(mle)))
return(predictedEX)
}
res=boot(data_reorder, statistic=f1,
         R=1000,mle=mle,ran.gen=rng, sim="parametric")
#plotting of prediction and confidence intervals (95%)
plot(x=data_reorder$MET,y=data_reorder$EX,
     pch=21, bg="orange",main="Prediction & Confidence interval plot (95%)"
     ,xlab="Values of MET",ylab="values of EX",ylim=c(100,500))
points(data_reorder$MET,pred,type="l") #plot fitted line
points(data_reorder$MET,e2$point[2,], type="l", col="blue")
points(data_reorder$MET,e2$point[1,], type="l", col="blue")
points(data_reorder$MET,e$point[2,], type="l", col="red")
points(data_reorder$MET,e$point[1,], type="l", col="red")
 hist(residuals,col = "green",xlim = c(-100,150),prob=TRUE)
lines(density(residuals),col="red")
```

```r
data<-read.csv2("NIRspectra.csv")
set.seed(12345)
#head(data)
#filtering out or removing viscocity and saving back to data
data<-data %>% select(-Viscosity)
pca_std<-prcomp(data)
# calculating lambda, eigenvalues
lambda<-(pca_std$sdev)^2
# data set to analyse proportion of variance (in percentage)
sprintf("%2.2f", lambda/sum(lambda)*100)
#Screeplot of PCA standard result
screeplot(pca_std,col="green",main="Screeplot")
#Variations contibution data
plot(lambda/sum(lambda),xlab="Principal Component",
     ylab="Proportion of variance ",type="b",col="red")
# plot column 1 (PC1) vs column 2 (PC2)of x in pca

PC1=pca_std$x[,1]
PC2=pca_std$x[,2]

plot(PC1,PC2,main="PCA Scores",col="red")





#
pca_rotation<-pca_std$rotation
plot(pca_rotation[,1],main = "PCA- Traceplot,PC1",col="red")
plot(pca_rotation[,2],main = "PCA-Traceplot,PC2",col="red")


ICA_res<- fastICA(data,n.comp=2) # 2 numbers of components to be extracted
# computation
K<-ICA_res$K
W<-ICA_res$W
W_dash <- K %*% W  #(matrix multiplication)
# plot of the scores of the first latent feature, first column
plot(W_dash[,1], main="ICA- Traceplot of PC1", ylab="loadings",col="red")
# plot of the scores of the second latent feature, second column
plot(W_dash[,2], main="ICA- Traceplot of PC2", ylab="loadings",col="red")
# Comparative plot with step 1 (scores comparision)
PC1_S1<-ICA_res$S[,1]
PC2_S2<-ICA_res$S[,2]
plot(PC1_S1,PC2_S2,main="ICA Scores",col="red",xlab = "PC1",ylab = "PC2")
```