Lab2 Block 1 Report

Samia Butt

Assignment 2. Analysis of credit scoring

2.1

Devided data in 3 parts e.g. Train, Validation and Test by 50%, 25% and 25% consectively.

2.2 Gini and Deviance Missclassification rates

Gini Missclassification rate for training/test

```
##
## train.gini.predict good bad
##               good  312  84
##               bad    41  63
```

Gini Missclassification rate for the train data is 0.25

```
##
## test.gini.predict good bad
##              good  142  55
##              bad    30  23
```

Gini Missclassification rate for the test data is 0.34

Deviance Missclassification rate for training/test

```
##
## train.dev.predict good bad
##              good  333  86
##              bad    20  61
```

Deviance Missclassification rate for the train data is 0.212

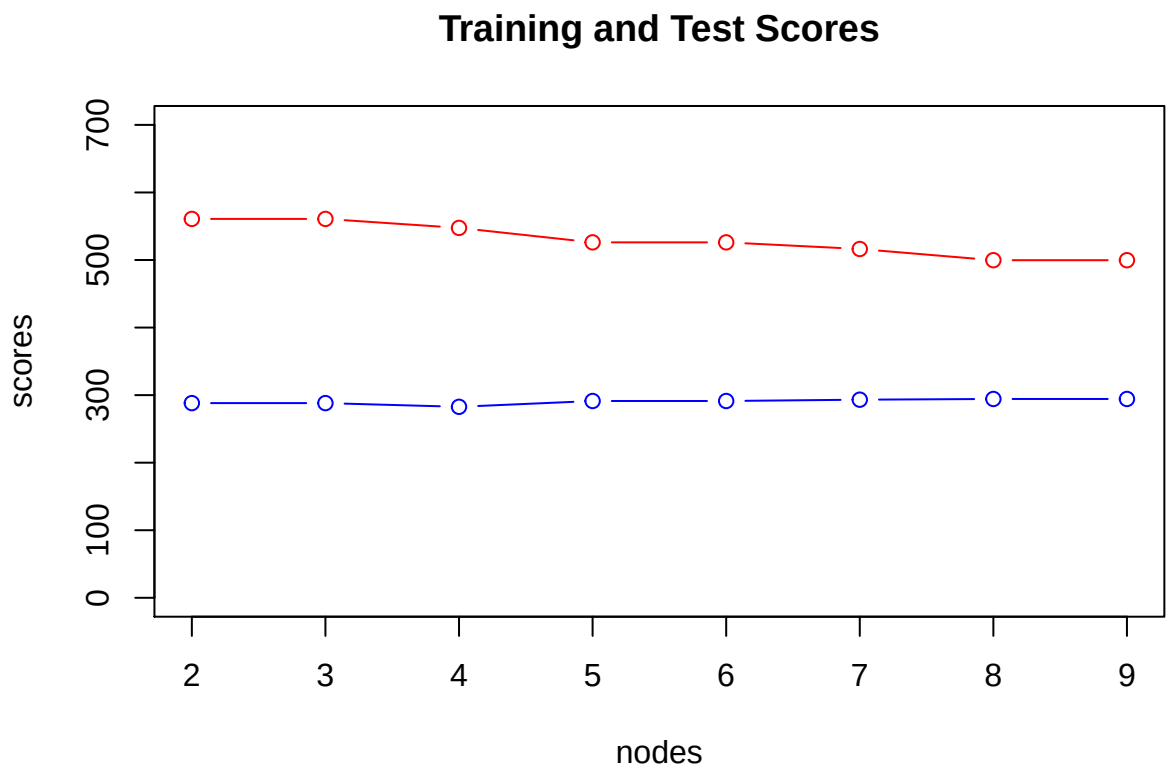
```
##
## test.dev.predict good bad
##             good  155  54
##             bad    17  24
```

Deviance Missclassification rate for the test data is 0.284

Comparison

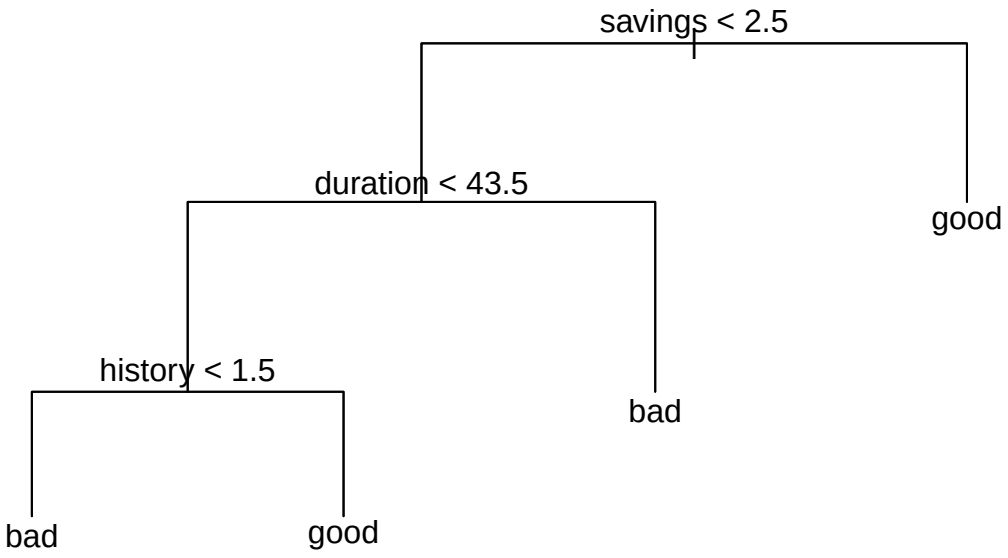
We will select the model using deviance impurity because it has less FPR as compare to gini. We have to focus on FPR in our scenario because if we give loan to the customers who were not able to mantain their loans it will cause big loss.

2.3 Choosing Optimal Tree



Optimal Tree Size 4

```
##
## Classification tree:
## snip.tree(tree = treemodel, nodes = c(5L, 3L, 9L))
## Variables actually used in tree construction:
## [1] "savings"  "duration" "history"
## Number of terminal nodes:  4
## Residual mean deviance:  1.117 = 547.5 / 490
## Misclassification error rate: 0.251 = 124 / 494
```



Information Interpretation of the tree structure: Above plotted tree is the optimal tree of depth 3 with 4 leafs/terminal nodes and it has used 3 variables e.g. duration, histroy and savings. While the root nodes of the sub tree which has been snipped off from the main tree are 5, 3 and 9. Misclassfication rate of the overall model is 0.251.

Misclassification rate for the pruned tree

```
##       fittedvalue
##        good bad
##   good  341  12
##   bad   114  33
```

Misclassification rate for the training data on Optimal tree is 0.252

```
##       testfittedvalue
##        good bad
##   good  166   6
##   bad    59  19
```

Misclassification rate for the test data on Optimal tree is 0.26

As we can see above pruned tree for test data misclassficiation rate is less than the deviance tree misclassification rate obtained in step(2.2), so we will say that pruned tree is much better than the deviance tree in our case.

2.4 Naive Bays Confusion matrix and missclassification rate

Confusion Matrix for train data

```
##       tainfit
```

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

Missclassification rate for the train data is 0.3

Confusion Matrix for validation data

```
##       testfit
##        good bad
##   good  117  58
##   bad    37  38
```

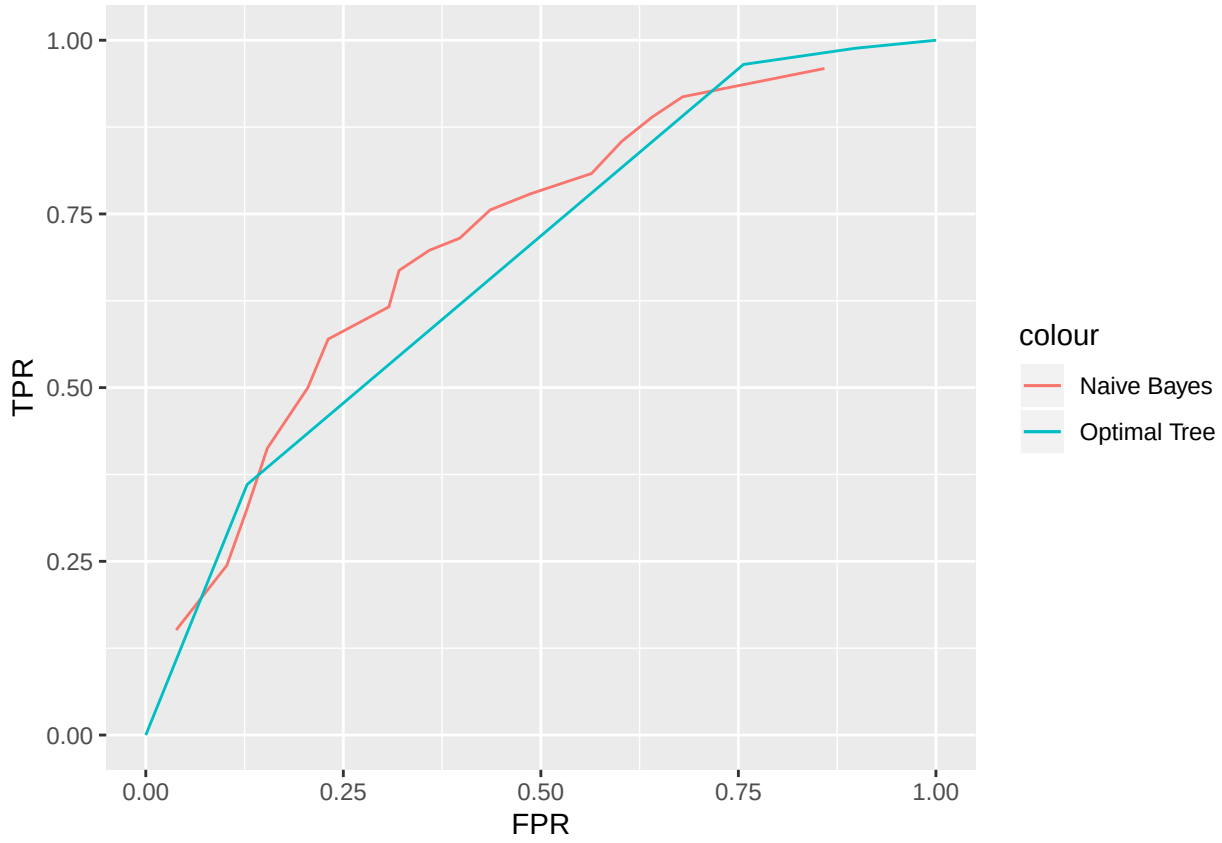
Missclassification rate for the test data is 0.32

Comparison

Model_Type	DataType	FPR
Optimal Tree	Train	0.7755102
Optimal Tree	Test	0.7564103
Naive Bays	Train	0.3537415
Naive Bays	Test	0.4933333

As FPR rate of Naive bayes is less than optimal tree so we will conclude that Naive bayes is better than the optimal tree .

2.5 ROC Graph



According to the above ROC graphs, as NaiveBayes has higher area under the curve (highre TPR rate) and

Table 1: Loss and No Loss FPR Comparison

tr_loss_FPR	tr_nloss_FPR	test_loss_FPR	test_nloss_FPR
0.0680272	0.3537415	0.1025641	0.4933333

Optimal tree has lower area under the curve (lower TPR as compare to NaiveBayes). As our trarget is to choose the model which has the higher TPR, so we conclude that the NaiveBayes is better than the optimal tree.

2.6 Loss matrix implementation

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

```
## [1] 0.06802721
```

```
##       testpredict
##        good bad
##   good   41 131
##   bad     8  70
```

```
## [1] 0.1025641
```

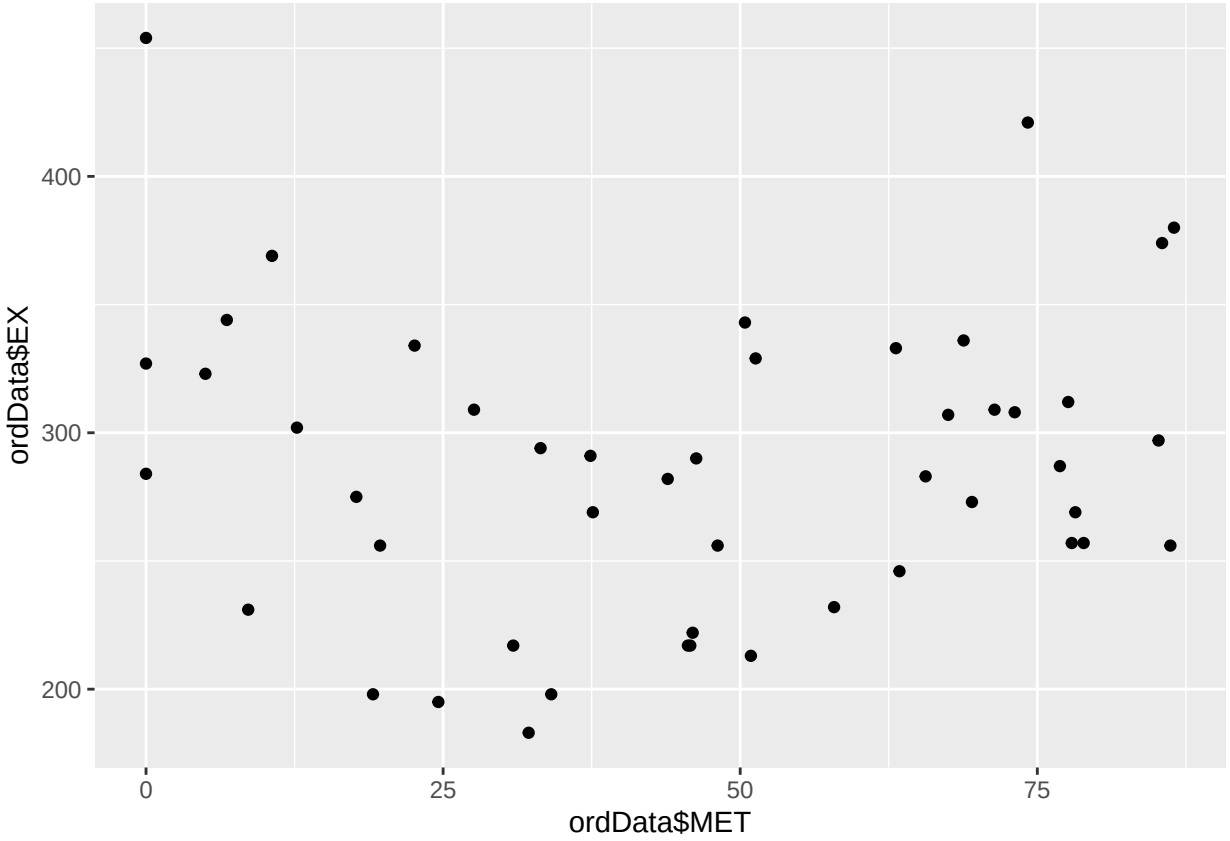
Comparison

Compare the results with the results from step 4. When we use NaiveBayes with the loss matrix, FPR decreasing and as in our scenario we are concerned to choose a method with less FPR, so we will conclude that NaiveBayes with loss matrxi is better as compare to no loss matrix from step 4.

discuss how the rates has changed and why. By using Loss matrix we restrict our classification to wrongly specify bad as good. In our case we are using 10 times restircition for bad, so in this way we decrease the number of bad to be classified as good.

Assignment 3. Uncertainty estimation

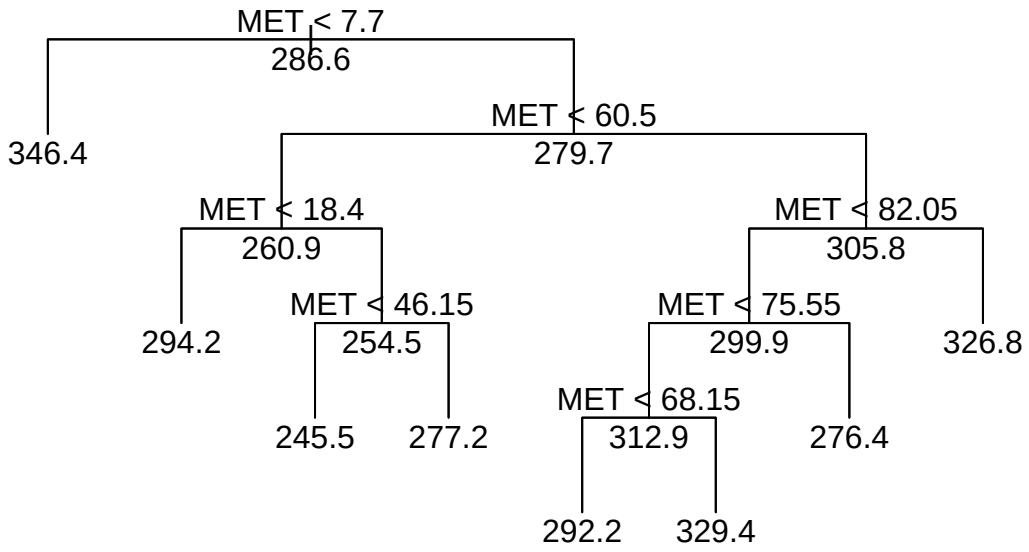
3.1 Reorder and ploting the original data



From the above ploted original data we can see that there is no linear relation between MET and EX while we can clearly notice that there is a non-lieaner relationship between them. From the baove plotted data behavior we can say that polynomial model might work correctly.

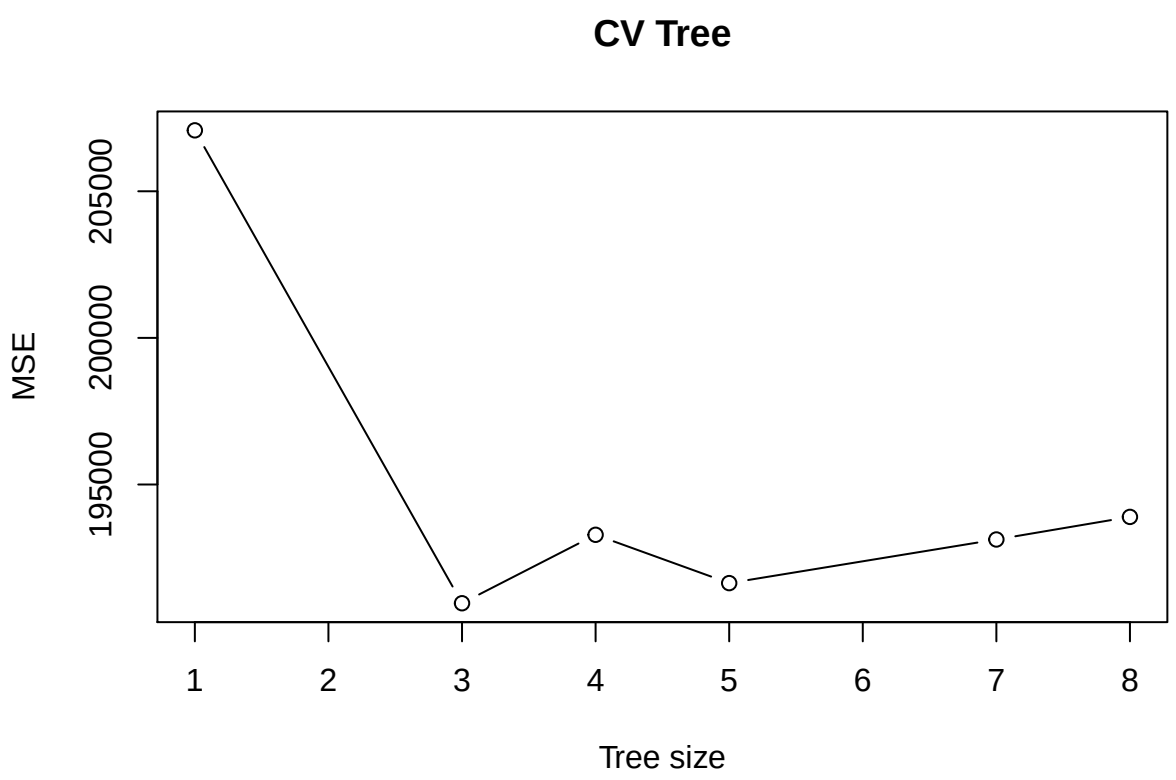
3.2.1 Regression tree model

In the below regression tree model we have selected minimum 8 leafs and target as EX wgile the feature as MET.



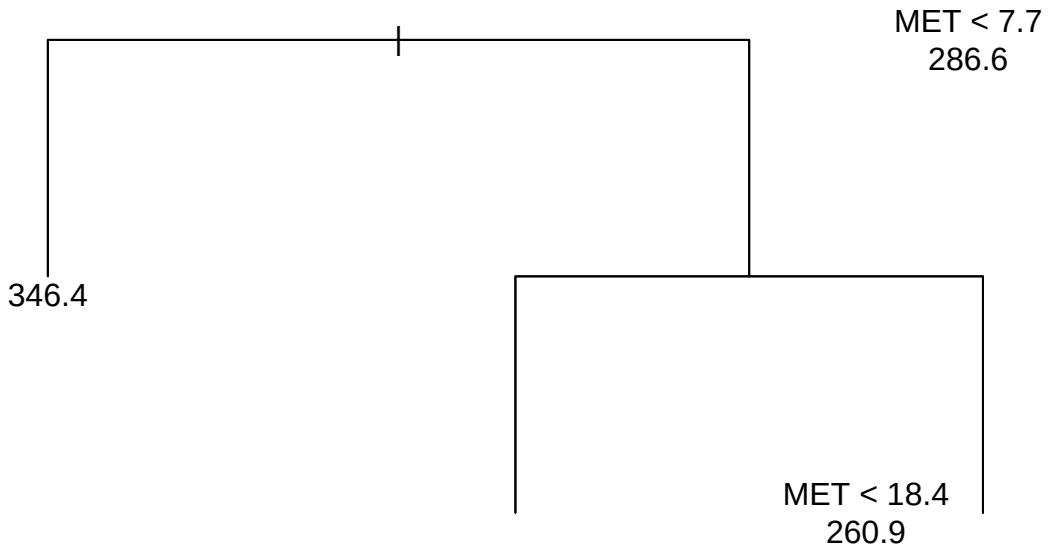
3.2.2 CV Tree Size vs Residuals

Below graph shows that at the begning when the tree size is 1, MSE is very high while the for the tree size 3 we have lowest MSE after this MSE starts increasing again with the increase of tree size.




```
## integer(0)
```

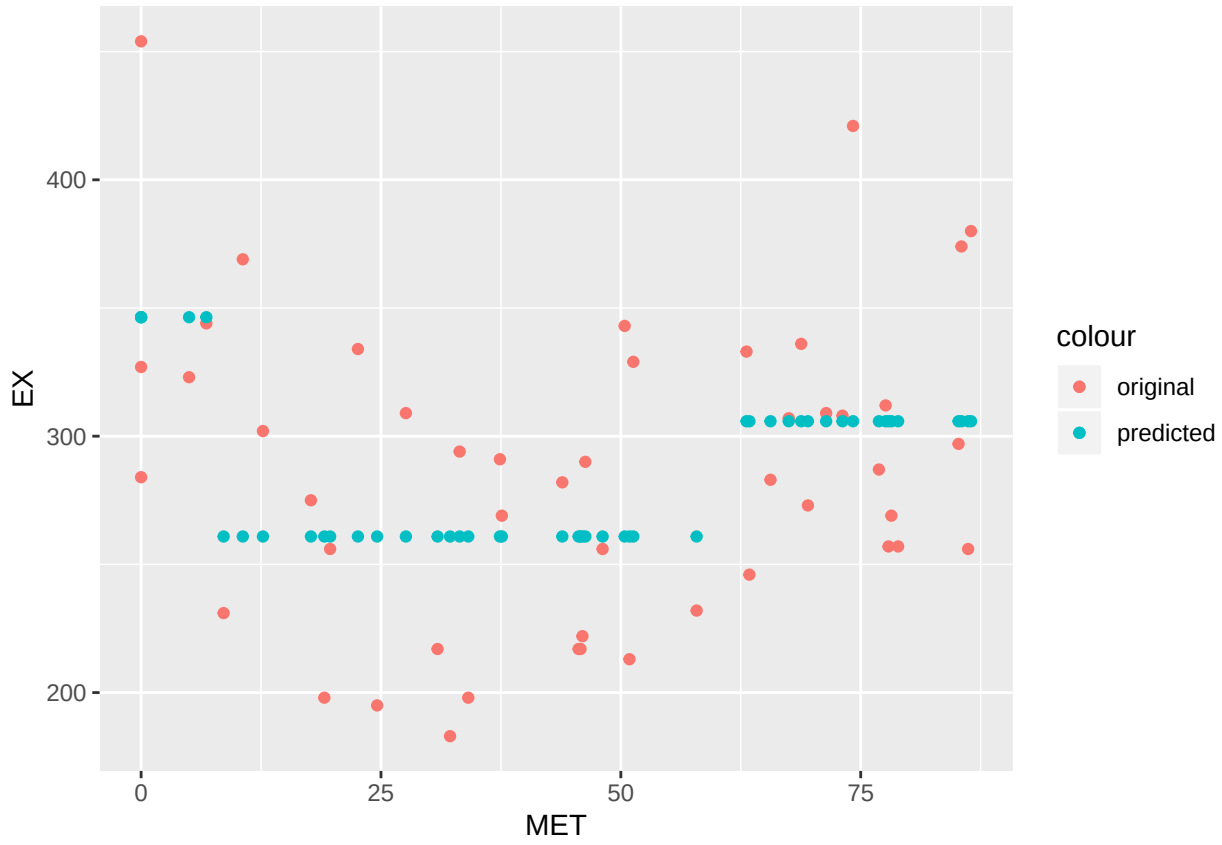
3.2.3 Optimal Tree



The above optimal tree is made by selecting the minimum leaves 3 using cross validation.

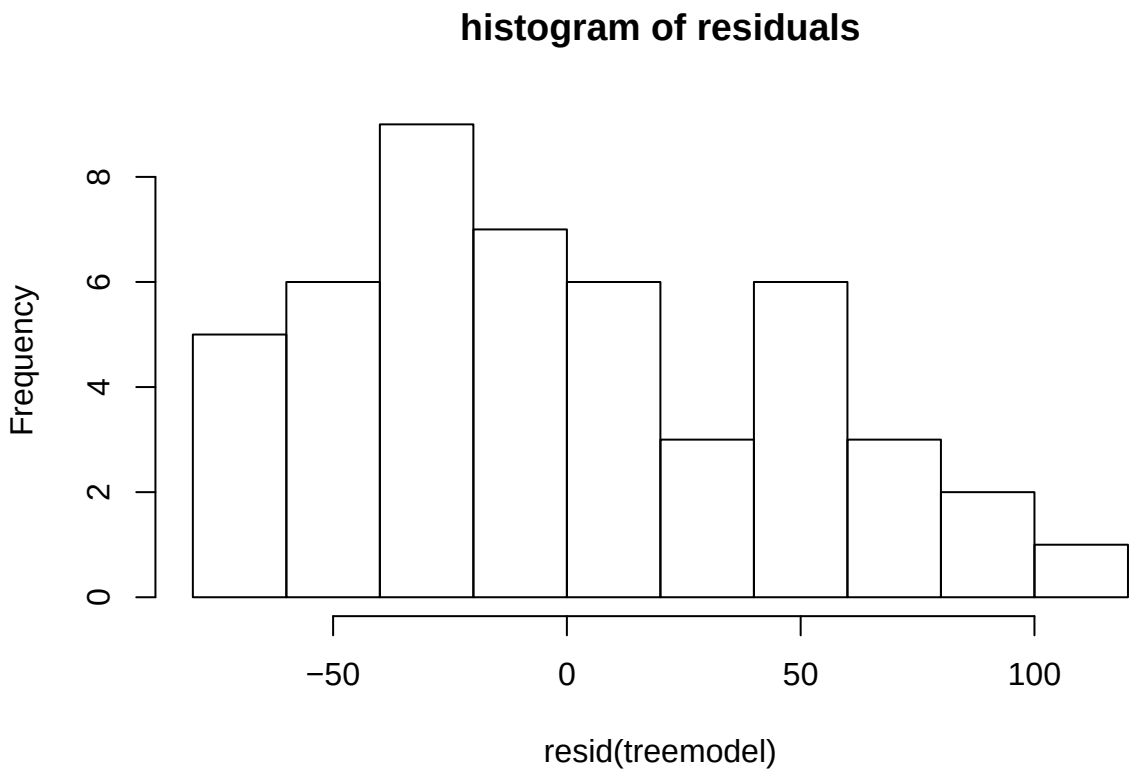
3.2.4 Original Vs Predicted data

After pruning the original tree we have predicted the values using pruned model and then ploted both the original and the predicted data.



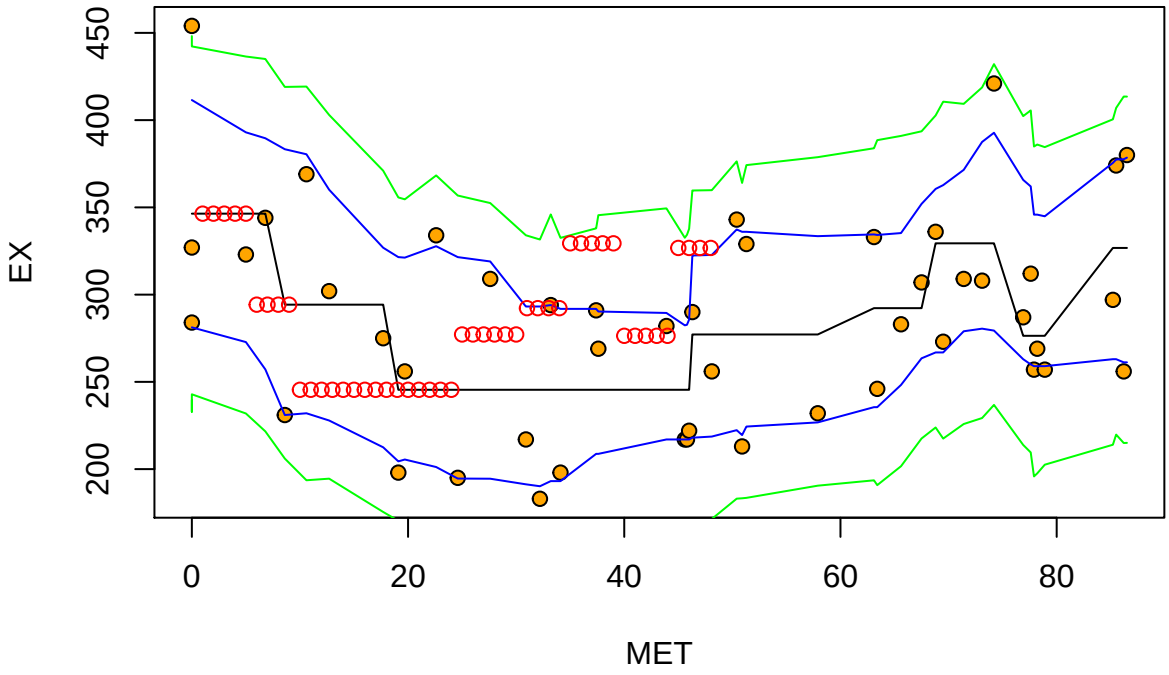
3.2.5 Residuals Histogram

Below histogram shows the residuals of the original tree.



We can see that, it looks like right skewed distributed histogram, while a falwless model has the zero centered, normally distiributed histogram. So we can conclude that our tree model didt work well.

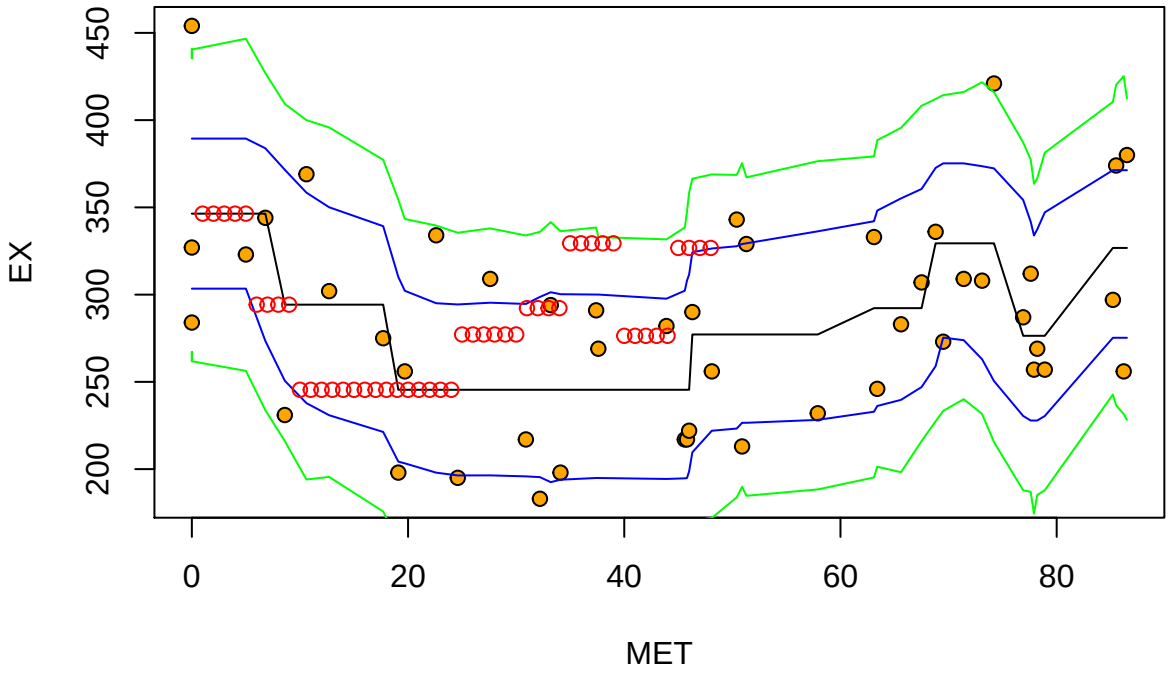
3.3 95% Confidence band for non-parametric Bootstrap



Comment whether the band is smooth or bumpy and try to explain why. In the above plot, blue line is the confidence band and the green line is the prediction band. As we can see that our confidence band is quite bumpy and wider because our data is scattered. According to the 95% confidence band we can see that there are some points out side of the confidence band.

Consider the width of the confidence band and comment whether results of the regression model in step 2 seem to be reliable. By comparing our 3.2.4 result with the above confidence band we can see that our predictions in 3.2.4 are within the confidence band but our confidence band is quite bumpy and wide. It is difficult to guess the distribution of the data. So we can't exactly conclude about the reliablity.

3.4 95% Confidence band for Parametric Bootstrap



Consider the width of the confidence band and comment whether results of the regression model in step 2 seem to be reliable. In the above Parametric Bootstrap plot, blue line is the confidence band and the green line is the prediction band. As we can see that our Parametric Bootstrap confidence band is smoother and thin, as compare to the non-parametric bootstrap confidence band. According to the 95% confidence band we can see that there are also some points, out side of the confidence band.

By comparing our 3.2.4 result with the above Parametric Bootstrap confidence band we can see that our predictions are within the confidence band but this time our confidence band is quite smooth and thin. But in paramtertic bootstrap we have assumed that we have normally distributed data. However our assumption is not correct, we can see this from the non-normal distributed residuals. so we can say that our result of the resgression model in step 2 is not reliable.

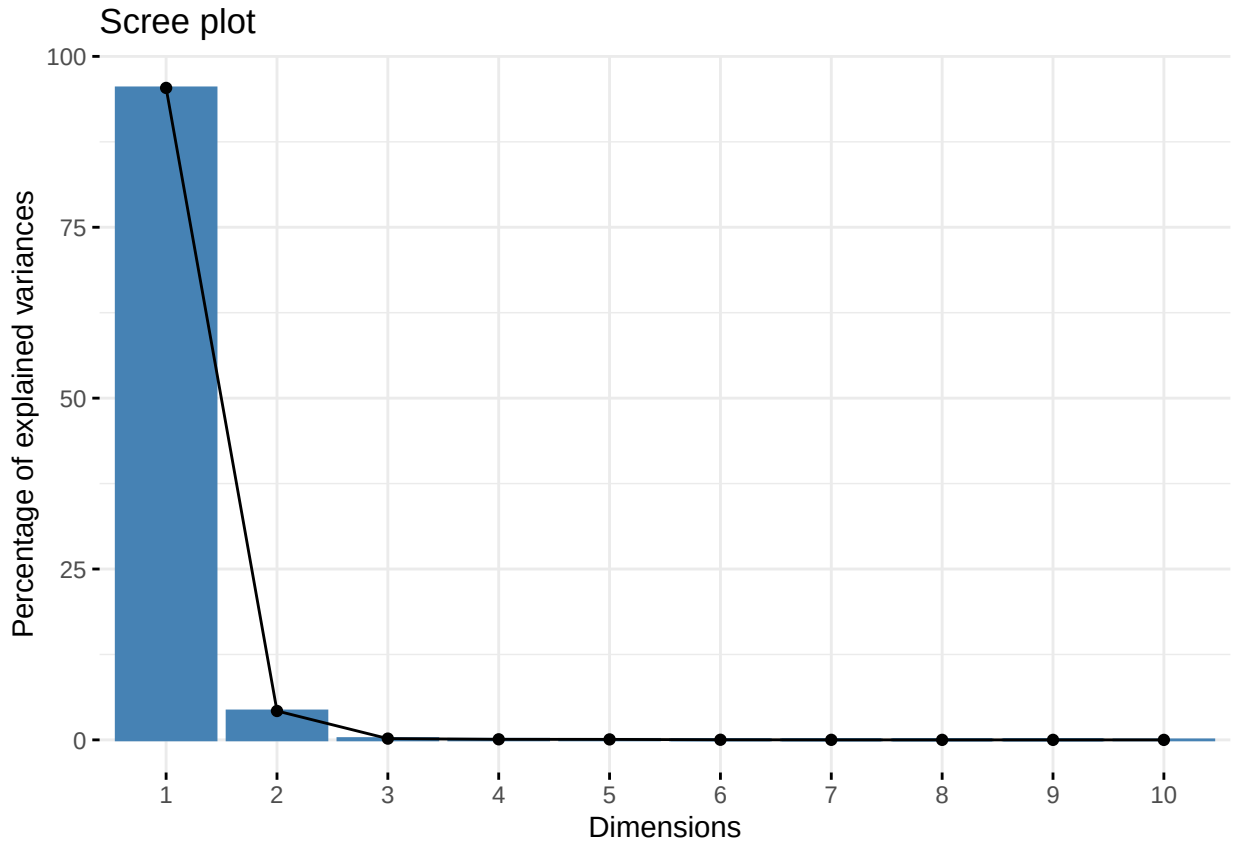
Does it look like only 5% of data are outside the prediction band? Should it be? As we can see that very small number of points are outside of the prediction band so we can say that only 5% of the data is not covered by the prediction band. It sounds reasonable that our 5% values are outside because we are expecting 95% of the values should be inside.

3.5 Comparison

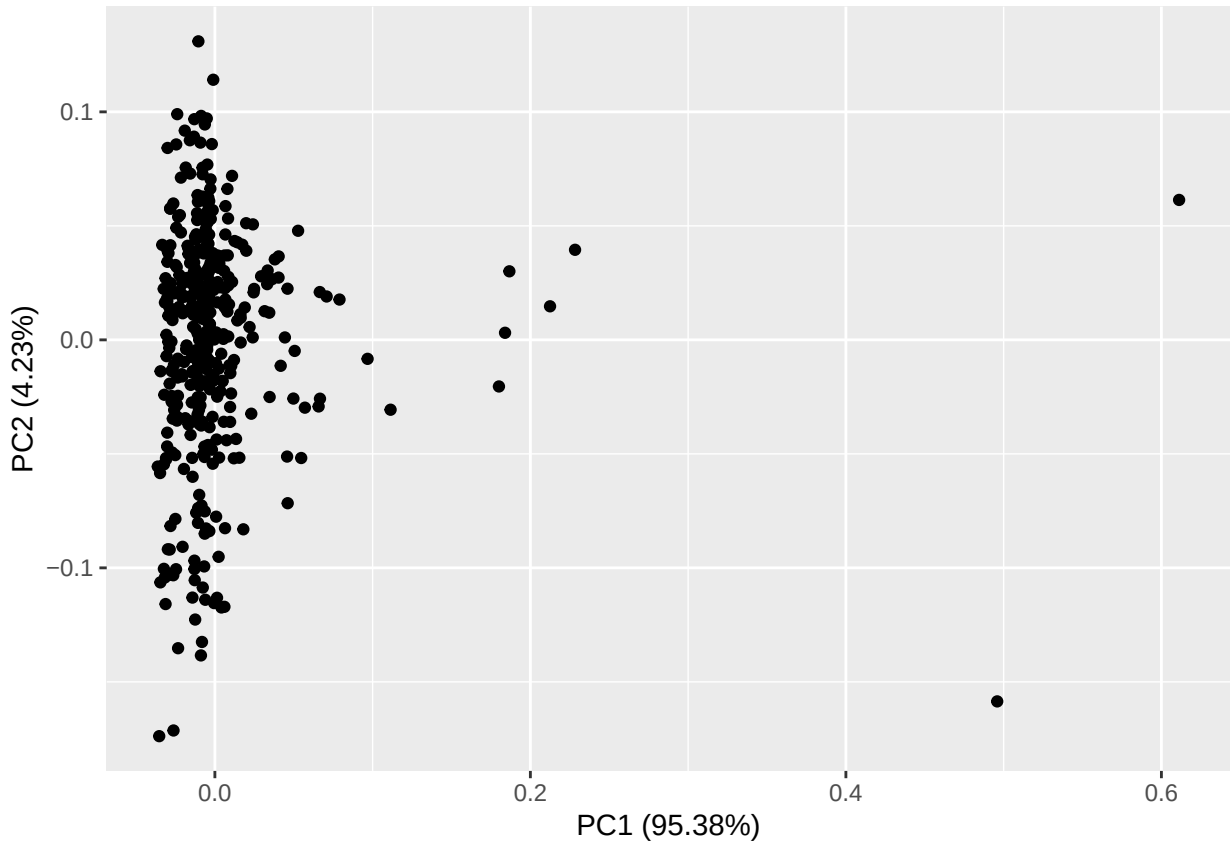
By comparing above both parametric and non-parametric bootstrap, the parametric bootstrap is not reasonable because we dont know the real distribution of data and we have assumed that we have normally distributted data so we can conclude that non-parametric is reasonable as comapre to parametric.

Assignment 4. Principal components

4.1 Percentage of Explained Variances



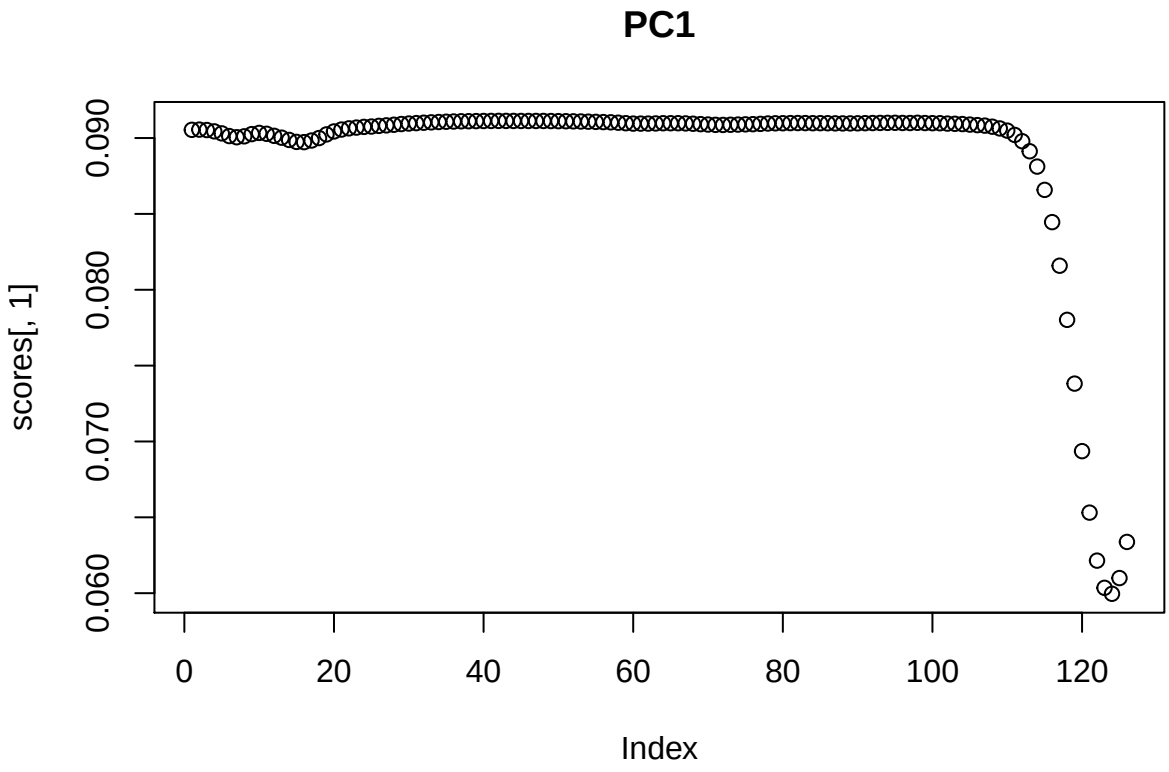
Does the plot show how many PC should be extracted? Above plot clearly shows that first two PCs have higher percentage of explained variance. So We have chosen first two principal components because the cumulative sum of their percentage is more than 99%.

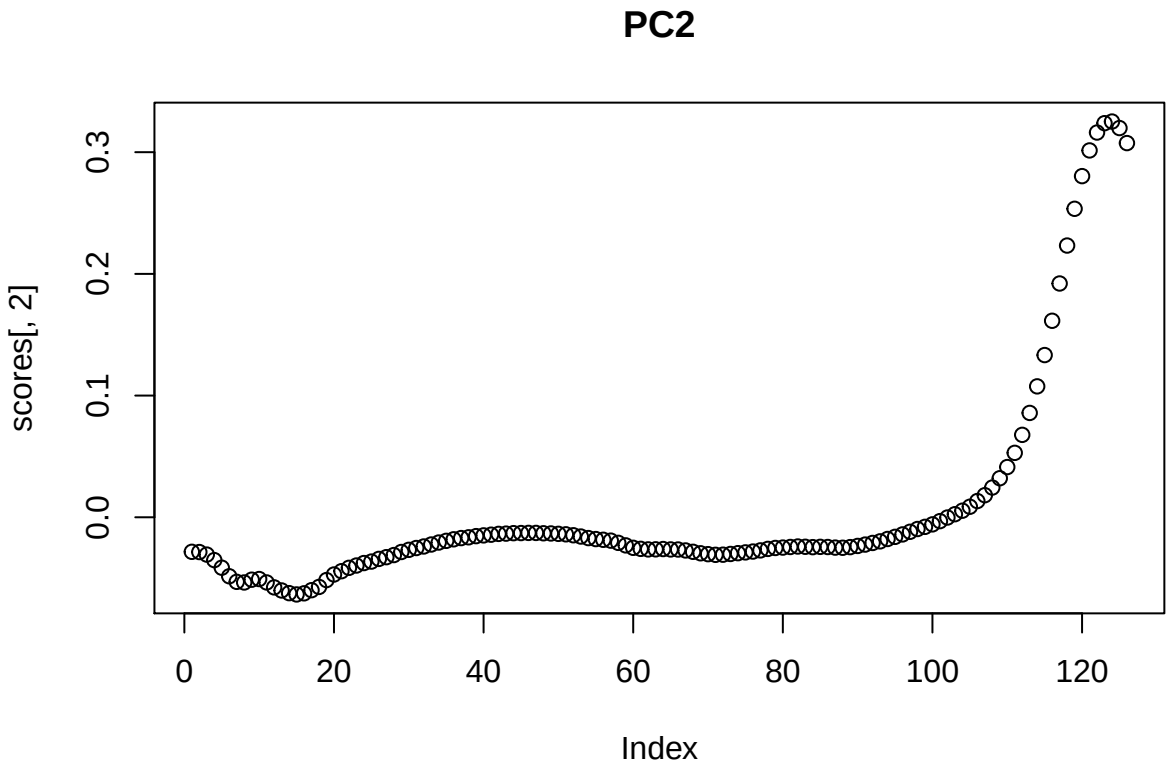


PC1 vs PC2

Are there unusual diesel fuels according to this plot? According to the above plot we can see that almost all the data is on left side while we have some outliers on extreme right.

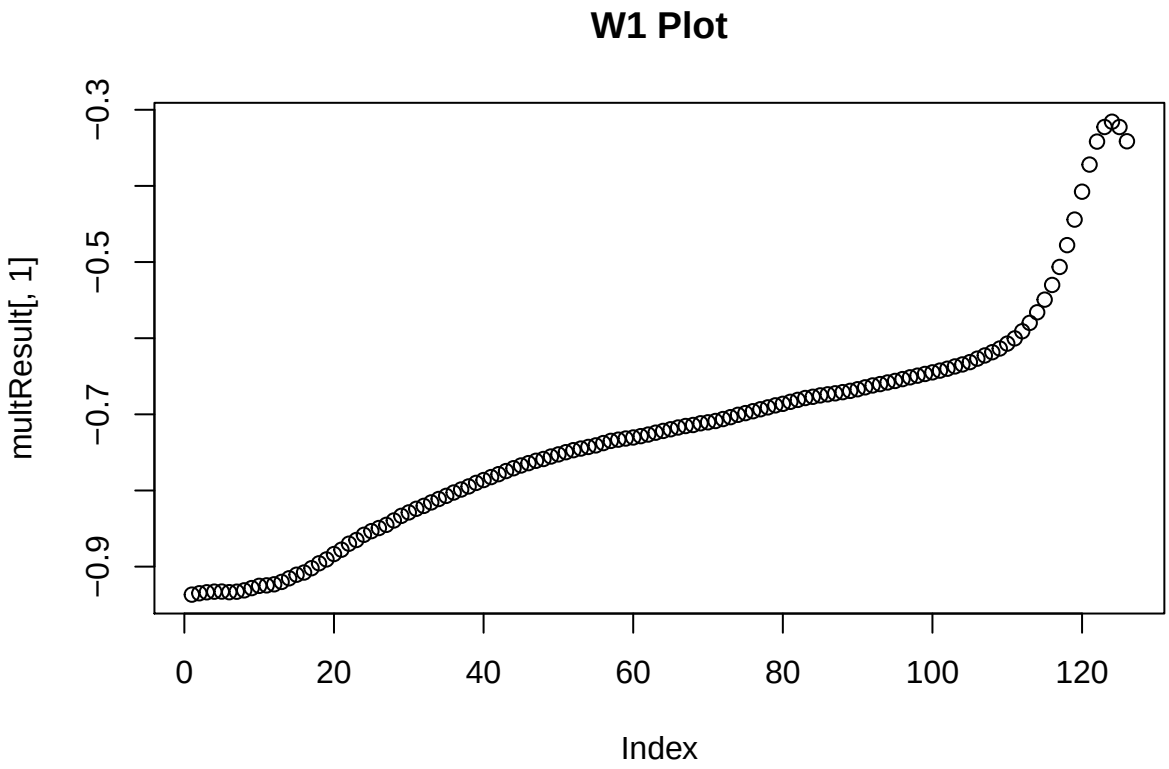
4.2 Trace plot

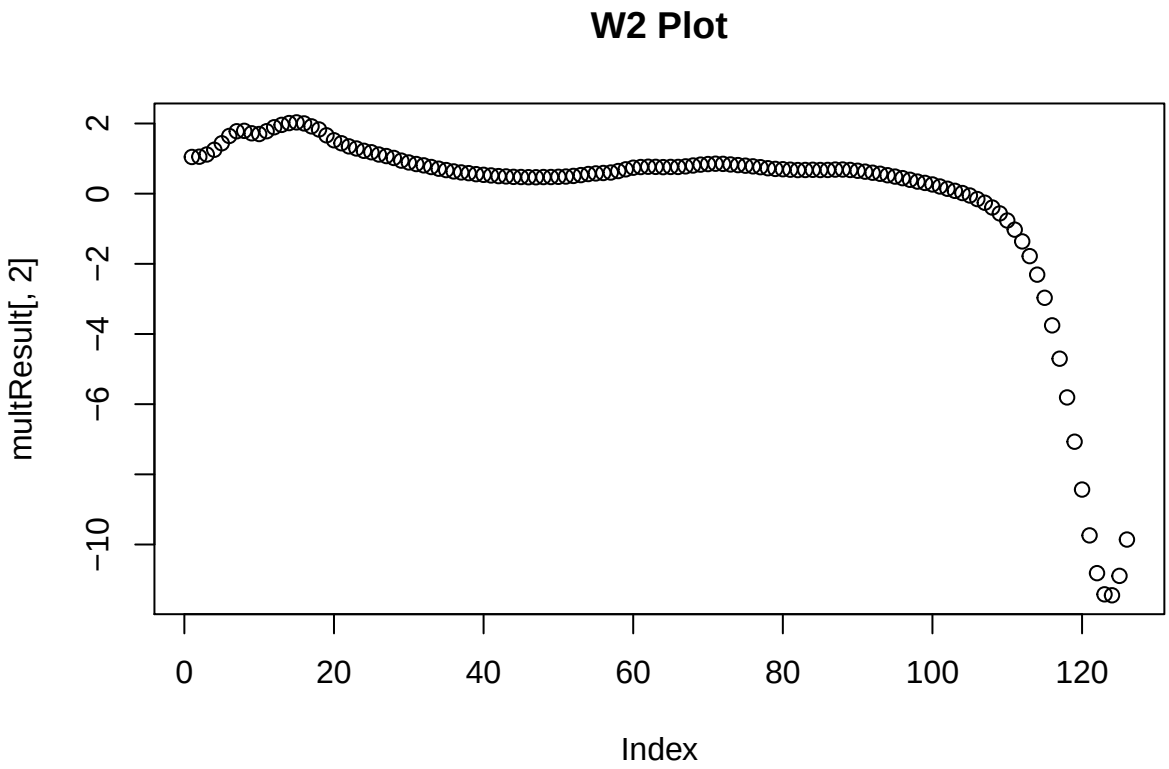




Is there any principle component that is explained by mainly a few original features? As we can see in the above PC1 and PC2 trace plots, PC1 has more imapct as compare to PC2. Which means PC2 is explained by few original features.

4.3 W1 Vs W2

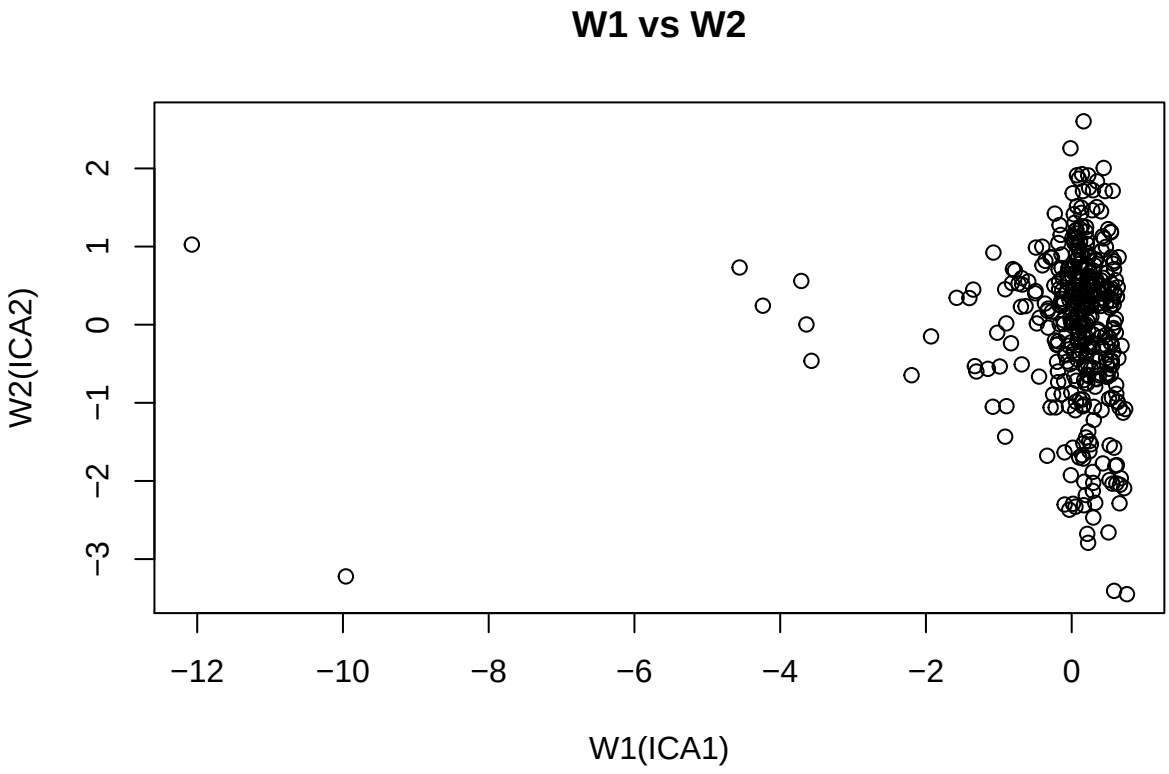

W1 Plot
multResult[, 1]
−0.3
−0.5
−0.7
−0.9
0
20
40
60
80
100
120
Index


W2 Plot
multResult[, 2]
2
0
−2
−4
−6
−10
0
20
40
60
80
100
120
Index

a.1 Compare with the trace plots in step 2 and make conclusions. By comparing trace plots from 4.2(PC) with above W trace plots, we can see it is almost the same like in step2. So we can conclude that in our case ICA has the same dependence on original data as the PCA has.

a.2 What kind of measure is represented by the matrix W'? By using fastICA, we have implemented parallel algorithem to exctract two independent components. In result we got unmixing component(W), which maximizes the independendence of data in the direction of principle components.

b. Compar it with the score plot from step 1



We can see in the W1 vs W2 graph that it is almost the same like in step1 PC1 vs PC2. So we can conclude that in our case ICA has the same result as the PCA has.

Appendex

```
knitr::opts_chunk$set(echo = FALSE)
library(readxl)
library(e1071)
library(fastICA)
library(tree)
library(boot)
library(factoextra)
library(ggfortify)
library(kableExtra)
library(tidyverse)
```

```
RNGversion("3.5.1")
data = read_excel("creditscoring.xls")
#data$purpose[which(is.na(data$purpose))] = 0
data$good_bad = factor(data$good_bad, labels=c("good","bad"),levels =c("good","bad"))
n=dim(data)[1]
set.seed(12345)
id=sample(1:n, floor(n*0.5))
train=data[id,]
id1=setdiff(1:n, id)
id2=sample(id1, floor(n*0.25))
valid=data[id2,]
id3=setdiff(id1,id2)
test=data[id3,]

gini.tree = tree(good_bad~.,data = train, split = "gini")
dev.tree = tree(good_bad~.,data = train,split = "deviance")
train.gini.predict = predict(gini.tree,train,type = "class")
table(train.gini.predict, train$good_bad)
train.gini.msclf = mean(train.gini.predict != train$good_bad)
test.gini.predict = predict(gini.tree,test,type = "class")
table(test.gini.predict,test$good_bad)
test.gini.msclf = mean(test.gini.predict != test$good_bad)
train.dev.predict = predict(dev.tree,train,type = "class")
table(train.dev.predict, train$good_bad)
train.dev.msclf = mean(train.dev.predict != train$good_bad)

test.dev.predict = predict(dev.tree,test,type = "class")
table(test.dev.predict, test$good_bad )
test.dev.msclf = mean(test.dev.predict != test$good_bad)

set.seed(12345)
treemodel = tree(good_bad~.,data = train, split = "deviance")

scores<-sapply(2:9, function(i){

  pruneTree = prune.tree(treemodel,best = i)

  prunePredict = predict(pruneTree, valid, type = "tree")

  list(prunscore = deviance(pruneTree), predscore = deviance(prunePredict), node= i)

})

plot(unlist(scores[3,]), unlist(scores[1,]), type="b", col="red",xlab = "nodes",ylab = "scores", ylim=c
points(unlist(scores[3,]), unlist(scores[2,]), type="b", col="blue")

valdDevScore = unlist(scores[2,])
nodes = unlist(scores[3,])

bestSize = nodes[which.min(valdDevScore)]


prunedTreemodel=prune.tree(treemodel, best=4)
```

```
summary(prunedTreemodel)
plot(prunedTreemodel)
text(prunedTreemodel)
# Missclassification rate for the validation data
fittedvalue=predict(prunedTreemodel, newdata=train, type="class")
valconmatrix = table(train$good_bad,fittedvalue)
valconmatrix
treetrainfpr = valconmatrix[2,1]/sum(valconmatrix[2,])
trainMscfrate = mean(fittedvalue != train$good_bad)
# Missclassification rate for the test data
testfittedvalue=predict(prunedTreemodel, newdata=test, type="class")
conmatrix = table(test$good_bad,testfittedvalue)
conmatrix
treevalidtfpr = conmatrix[2,1]/sum(conmatrix[2,])
testMscfrate = mean(testfittedvalue != test$good_bad)

navbaseModel = naiveBayes(good_bad~., data=train)

#summary(navbaseModel)

tainfit=predict(navbaseModel, newdata=train, type = "class")
conf = table(train$good_bad,tainfit)
conf
nbtrainfpr = conf[2,1]/sum(conf[2,])

nbtrainMscfrate = mean(tainfit != train$good_bad)

testfit = predict(navbaseModel, newdata=test, type = "class")
nbconf  = table(valid$good_bad,testfit)
nbconf
nbvalidfpr = nbconf[2,1]/sum(nbconf[2,])
nbvalidMscfrate = mean(testfit != test$good_bad)


result = data.frame(Model_Type = c("Optimal Tree","Optimal Tree","Naive Bays","Naive Bays"),
DataType = c("Train","Test","Train","Test"),
FPR=c(treetrainfpr,
treevalidtfpr,
nbtrainfpr,
nbvalidfpr)
)

kable(result)
pivalue <- seq(from = 0.05, to = 0.95, by = 0.05)

comresutls<-sapply(pivalue, function(i){
  nbpredicts = predict(navbaseModel, test, type = "raw")

  nbpredict <- ifelse(nbpredicts[,1] > i, "good", "bad")
  nbConfusion <- table(test$good_bad,factor(nbpredict, levels = c("good","bad"),labels =c("good","bad"))

  nbTPRcal <- nbConfusion[1,1]/sum(nbConfusion[1,])
  nbFPRcal <- nbConfusion[2,1]/sum(nbConfusion[2,])
```

```
  prunepredicts = predict(prunedTreemodel, test)
  prunepredict <- ifelse(prunepredicts[,1] > i, "good", "bad")
  pruneConfusion <- table(test$good_bad,factor(prunepredict, levels = c("good","bad"),labels =c("good",

  pruneTPRcal <- pruneConfusion[1,1]/sum(pruneConfusion[1,])
  pruneFPRcal <- pruneConfusion[2,1]/sum(pruneConfusion[2,])

  data.frame(nbTPR =nbTPRcal ,nbFPR = nbFPRcal, pruneTPR =pruneTPRcal, pruneFPR=pruneFPRcal  )

})



datafr = data.frame(nbTPR =unlist(comresutls[1,]) ,nbFPR = unlist(comresutls[2,]), pruneTPR =unlist(com

ggplot(datafr)+geom_line(aes(y=nbTPR,x=nbFPR,col="Naive Bayes"))+geom_line(aes(y=pruneTPR,x=pruneFPR,co

nblossModel <- naiveBayes(good_bad~., train)

nblossPredict <- predict(nblossModel, train, type = "raw")
trainpred <- ifelse(nblossPredict[,1] > 10*nblossPredict[,2], "good", "bad")

trainpredict = factor(trainpred, levels = c("good","bad"),labels =c("good","bad"))


trainconf = table(train$good_bad, trainpredict)
trainconf
trainfpr <- trainconf[2,1]/sum(trainconf[2,])
trainfpr

nblossPredictedtest <- predict(nblossModel, test, type = "raw")

testpred <- ifelse(nblossPredictedtest[,1] > 10*nblossPredictedtest[,2], "good", "bad")

testpredict = factor(testpred, levels = c("good","bad"),labels =c("good","bad"))


testconf = table(test$good_bad, testpredict)
testconf
testfpr <- testconf[2,1]/sum(testconf[2,])
testfpr

datadfs = data.frame(tr_loss_FPR =trainfpr ,tr_nloss_FPR=nbtrainfpr, test_loss_FPR =testfpr ,test_nloss

kable(datadfs, caption = "Loss and No Loss FPR Comparison")
#=======================Assignment 3=======================================
data = read.csv2("State.csv")

#hist(data$EX)

#hist(data$MET)

ordData = data[order(data$MET),]
```

```r
ggplot(ordData,aes(ordData$MET,ordData$EX))+geom_point()

z=tree.control(nrow(ordData),minsize = 8)
treemodel = tree(EX~MET,ordData, control = z)

plot(treemodel, type = "uniform")
text(treemodel, all = TRUE)


# Task 2
cvtree = cv.tree(treemodel)

plot(cvtree$size,cvtree$dev,type="b",xlab = "Tree size",ylab = "MSE")+title("CV Tree")

#check minimum MSE
indexminmse = which.min(cvtree$dev)

#see which node has min mse
minsize = cvtree$size[indexminmse]

prunedTree = prune.tree(treemodel,best = minsize)

plot(prunedTree,type = "uniform")
text(treemodel, all = TRUE)
pred = predict(prunedTree,newdata = ordData)

df = cbind(ordData,pred)


ggplot(df,aes(x = MET, y = EX))+geom_point(aes(MET,EX,col="original"))+geom_point(aes(MET,pred, col="pr

hist(resid(treemodel),main = "histogram of residuals")
# task # 3

library(tree)
library(boot)

treefunc <- function(bootdata,indxes){

  newdata = bootdata[indxes,] # extract bootstrap sample

  tre = tree(EX~MET,data = newdata,control = z)

  fittedValue = predict(tre,ordData)
  return(fittedValue)
}
treefuncpred <- function(bootdata,indxes){

  newdata = bootdata[indxes,] # extract bootstrap sample

  tre = tree(EX~MET,data = newdata,control = z)

  fittedValue = predict(tre,ordData)
```

```
  n = length(ordData$EX)
  nmpredict <- rnorm(n,fittedValue, sd(residuals(tre)))
  return(nmpredict)
}

bootobj <- boot(data = ordData, statistic =  treefunc, R =1000)

bootobjpred <- boot(data = ordData, statistic =  treefuncpred, R =1000)

e=envelope(bootobj) #compute confidence bands
epred=envelope(bootobjpred)


bandpredict = predict(treemodel,ordData)

#ggplot(ordData,aes(ordData$MET,ordData$EX))+geom_point(aes(ordData$MET,ordData$EX,col = "original"))+g

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

#plot predicted cofidence bands
points(ordData$MET,epred$point[2,], type="l", col="green")
points(ordData$MET,epred$point[1,], type="l", col="green")

points(bandpredict,col="red")

# TASK 4 Non-paramteric bootstrap
ptreemodel = tree(EX~MET, data=ordData, control = z)

rng=function(data, ptreemodel) {

  data1=data.frame(EX=data$EX, MET=data$MET)
  n=length(data$EX)
  #generate new MET
  #ptest.MET <- data.frame(MET = data1$MET)
  data1$EX=rnorm(n,predict(ptreemodel, newdata=data1), sd(residuals(ptreemodel)))
  return(data1)
}
f1=function(data1){
  res=tree(EX~MET, data=data1, control = z) #fit regression tree model
  #predict values for all EX values from the original data
  exP=predict(res,newdata=ordData)

  return(exP)
}
predf =function(data1){
  res=tree(EX~MET, data=data1, control = z) #fit regression tree model
  #predict values for all EX values from the original data
  exP=predict(res,newdata=ordData)
   n = length(ordData$EX)
```

```
  nmpredict <- rnorm(n,exP, sd(residuals(res)))

  return(nmpredict)
}
pboot=boot(ordData, statistic=f1, R=1000,  mle=ptreemodel,ran.gen=rng, sim="parametric")

#plot(pboot)

predpboot=boot(ordData, statistic=predf, R=1000,  mle=ptreemodel,ran.gen=rng, sim="parametric")


paramE=envelope(pboot) #compute confidence bands
predparamE=envelope(predpboot) #compute predicted confidence bands


plot(ordData$MET, ordData$EX, pch=21, bg="orange", xlab = "MET", ylab="EX")
points(ordData$MET,bandpredict,type="l") #plot fitted line
#plot cofidence bands
points(ordData$MET,paramE$point[2,], type="l", col="blue")
points(ordData$MET,paramE$point[1,], type="l", col="blue")
#plot predicted cofidence bands
points(ordData$MET,predparamE$point[2,], type="l", col="green")
points(ordData$MET,predparamE$point[1,], type="l", col="green")

points(bandpredict,col="red")
#======================Assignment 4=========================================
data = read.csv2("NIRSpectra.csv")

subdata = data

subdata$Viscosity = c()

pcaData = prcomp(subdata,scale = TRUE,center = TRUE)

fviz_eig(pcaData)
# PC1 vs PC2
autoplot(pcaData)
# 4.2
scores<- pcaData$rotation
plot(scores[,1], main="PC1")
plot(scores[,2], main="PC2")
# 4.3
set.seed(12345)
resultica = fastICA(subdata,n.comp = 2,alg.typ = "parallel")

multResult = resultica$K %*% resultica$W
plot(multResult[,1],main = "W1 Plot")
plot(multResult[,2],main = "W2 Plot")

scores = resultica$X %*% resultica$K
plot(scores, xlab ="W1(ICA1)" ,ylab ="W2(ICA2)",  main = "W1 vs W2")
```