Lab2 Block1 Correction

Aman Kumar Nayak

12/29/2019

Assignment 2

Part 2.1

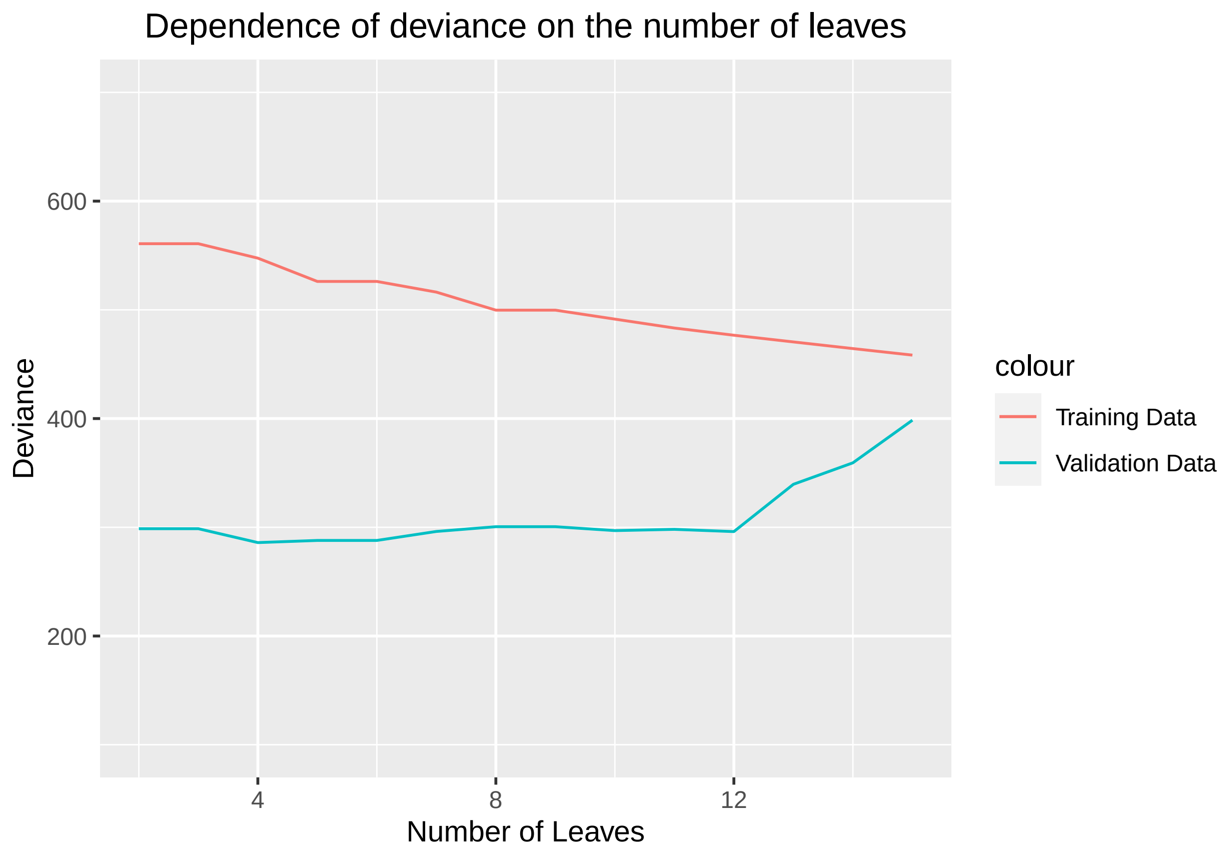
Divided data in train, validate and test as asked.

Part 2.2

```
## Misclassification rate for Training Data when Impurity measure is Deviance :  0.212

## Misclassification rate for Testing Data when Impurity measure is Deviance :  0.268

## Misclassification rate for Training Data when Impurity is measure is Gini Index :  0.238

## Misclassification rate for Testing Data when Impurity is measure is Gini Index :  0.372
```

When comparing Impurity measure methods, I could see that Misclassification Rate for Deviance as impurity measure is least among two, so selecting it for further implementation.

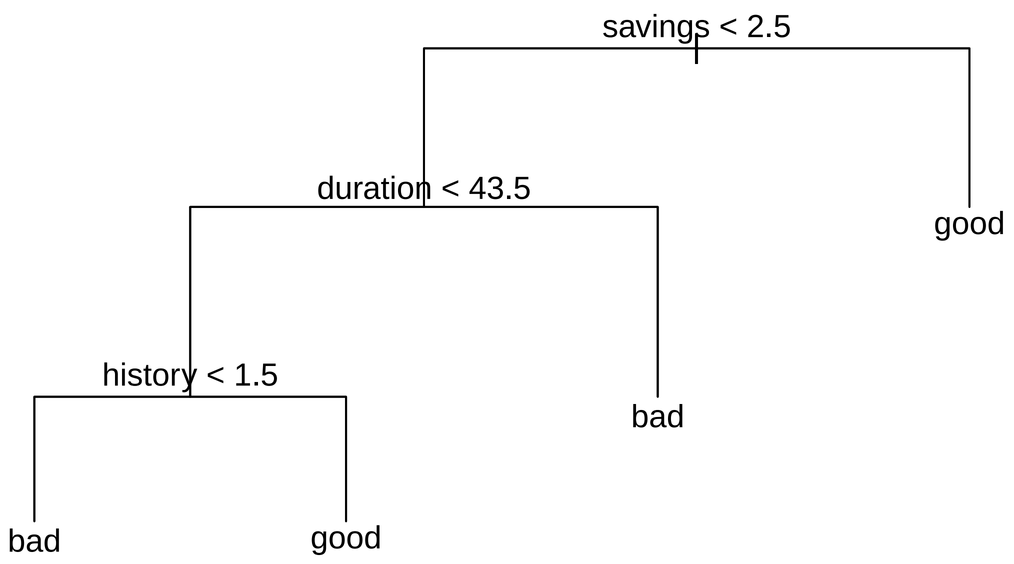
Part 2.3



Looking at graph for deviance dependency with training and validation data, we could see that even though deviance for training data is constantly decreasing but for validation data after number of leaves increased from 4, it shows continuous increase in deviance and from point 12 there is high spikes which signifies over fit model for increased number of leaves.

```
## Optimal number of leaves based on Miscal Rate of Validation Data is :  4
```

```
## Optimal Tree is
```



Looking at optimal tree plot we can say that *tree depth is 3*.

On analyzing tree we can say that if applicant have saving is more than 2.5 unit amount he is already marked as good manager of loan in terms of return of loan and can be given credit while if it is less than that we look for different values to validate in second order i.e. duration. If duration of credit is more than 43.5 unit time he/she is classified as bad creditor while if it less than that we would check for his/her credit history duration and in case if it found to be less than 1.5 unit time he/she is marked as bad one else as good creditor.

So in case saving is less than 2.5 units we would like to validate creditor based on his duration of loan followed by history data in order to classify him/her eligible for credit.

```
## Confusion Matrix for Optimal Decision Tree Train data is as below
```

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

```
## Miscalculation Rate for Optimal Decision Tree Train data is :  0.252
```

```
## Confusion Matrix for Decision Tree Test data is as below
```

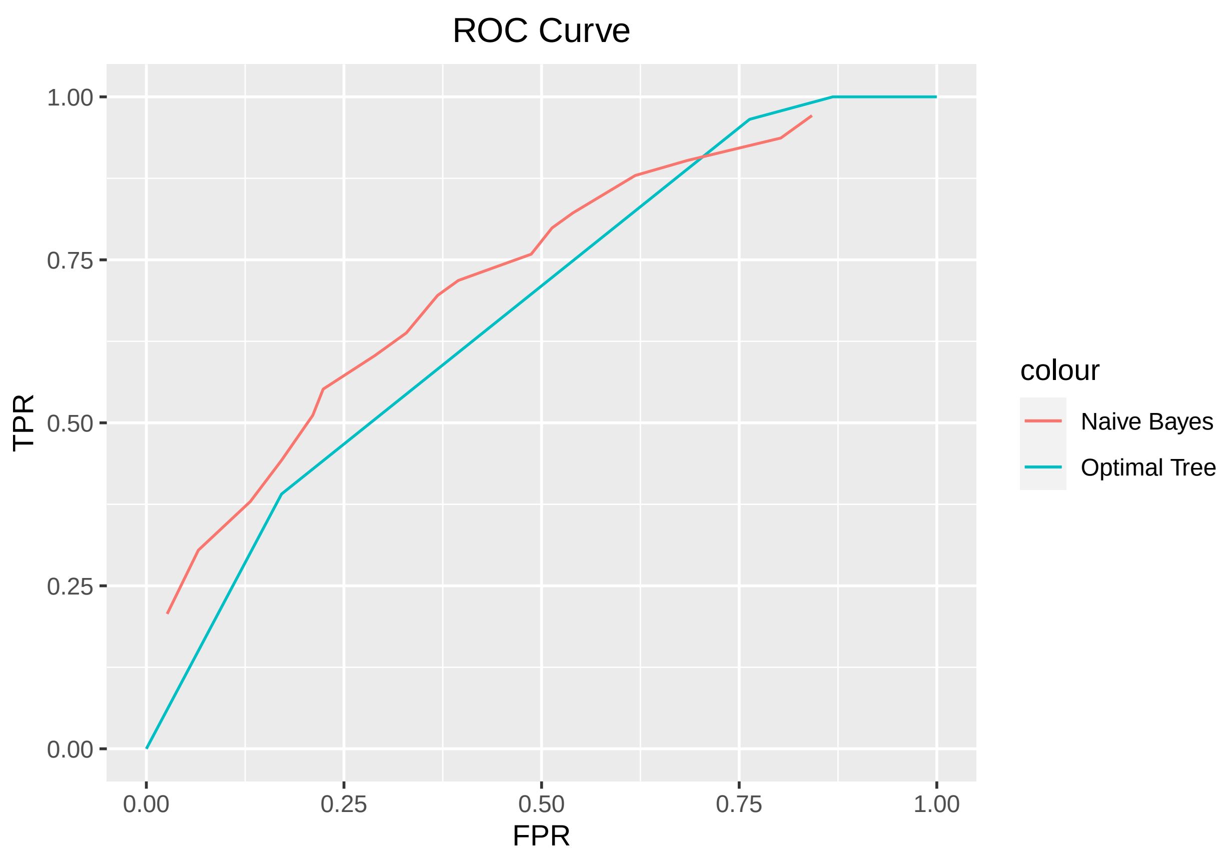
```
##
## testPred bad good
##     bad   18    6
##     good  58  168
## Miscalculation Rate for Decision Tree Test data is :  0.256
```

Part 2.4

```
## Naive Bayes

## Confusion Matrix for Naive Bayes Train Data

##
## nTrainPred bad good
##       bad   95   98
##       good  52  255

## Misclassification Rate for Naive Bayes Train Data is  0.3

## Confusion Matrix for Naive Bayes Test Data

##
## nTestPred bad good
##      bad   46   49
##      good  30  125

## Misclassification Rate for Naive Bayes Test Data is  0.316
```

On comparing misclassification rate of Naive Bayes with optimal tree's training and test data, it can be concluded that Optimal tree perform better also the classification of bad creditors as good one is extremely low for Optimal tree when compared.

Part 2.5



Post analysis ROC curve, it can be stated that Naive Bayes is better classifier for given data as area under curve for Naive Bayes is more compared to Optimal Tree.

Part 2.6

```
## Confusion Matrix for Naive Bayes Training Data with given Loss Matrix
```

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

```
## Miscalculation Rate for Naive Bayes Training Data with given Loss Matrix is : 0.546
```

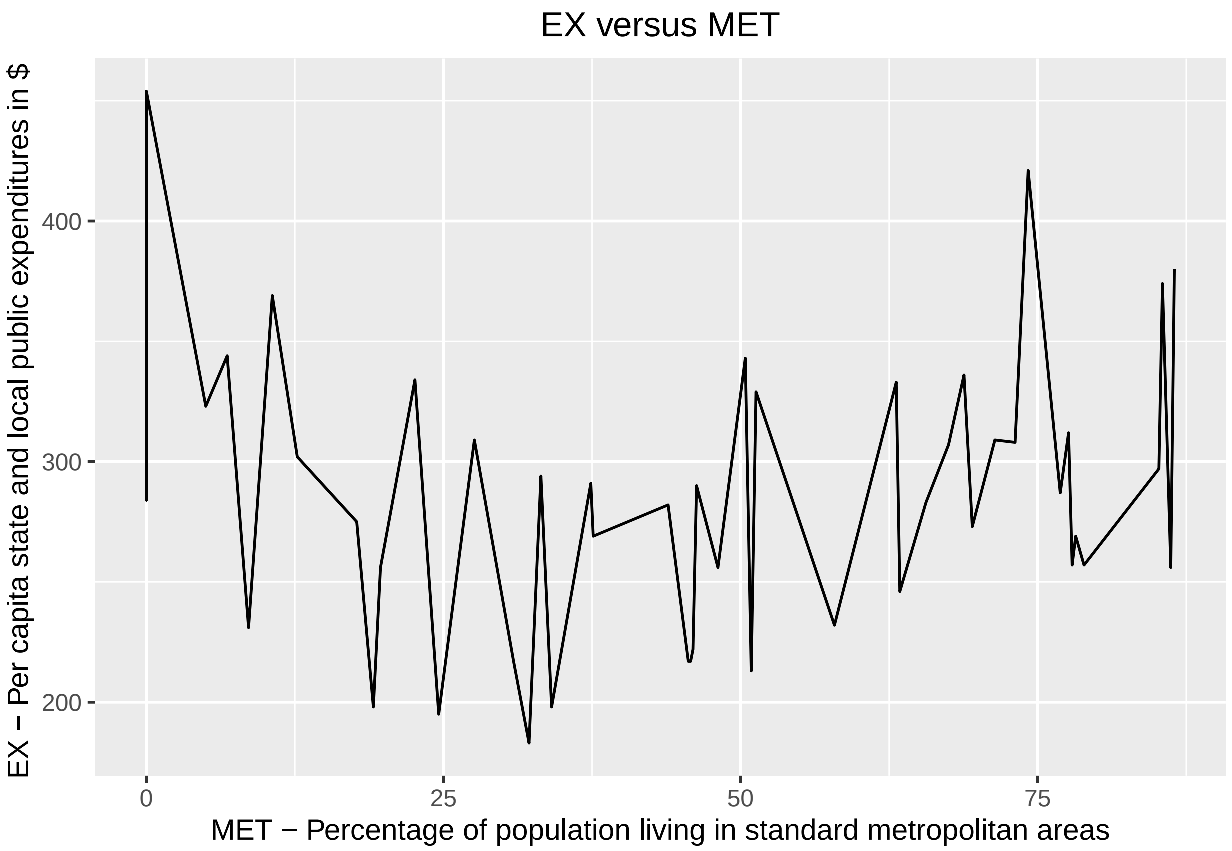
```
## Confusion Matrix for Naive Bayes Test Data with given Loss Matrix
```

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

```
## Miscalculation Rate for Naive Bayes Test Data with given Loss Matrix is : 0.508
```

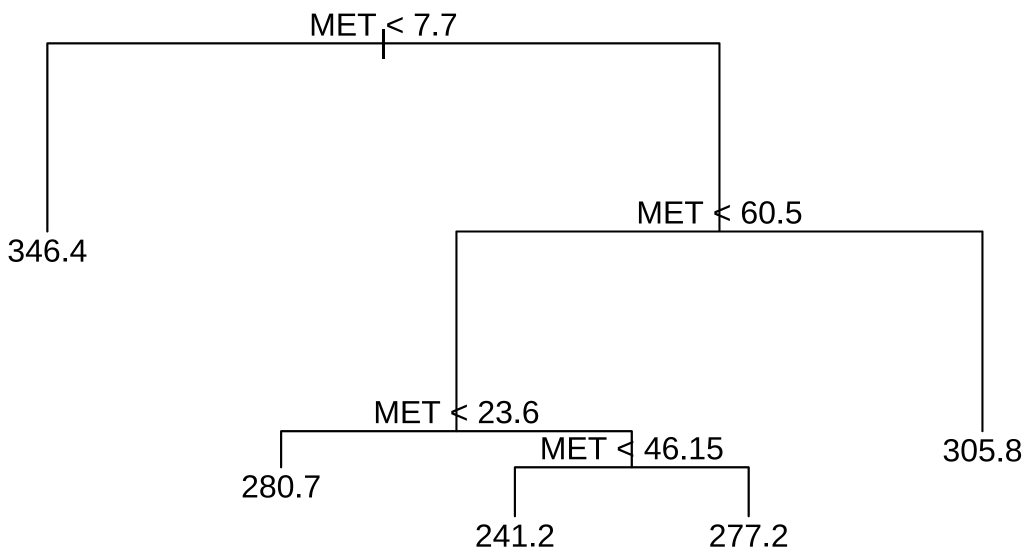
Post applying Loss function it can be seen that accuracy of prediction is decreased (which in inverse of misclassification rate) but objective to applying loss function was to reduce false classification of good creditor as bad one and thus that number is significantly reduced while raise in misclassification rate can be explained by increased bad creditor as good one.

Assignment 3



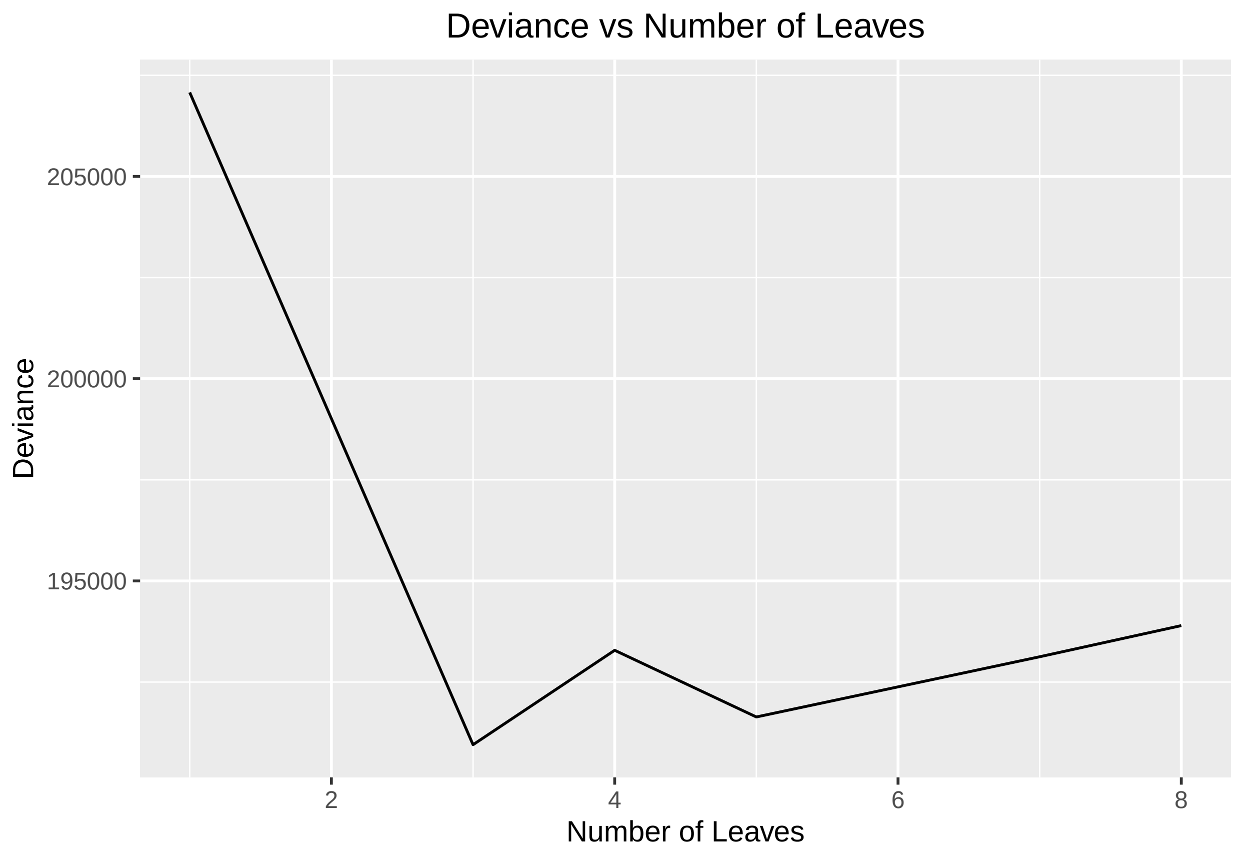
Part 3.1

```
## Decision Tree for given parameters
```



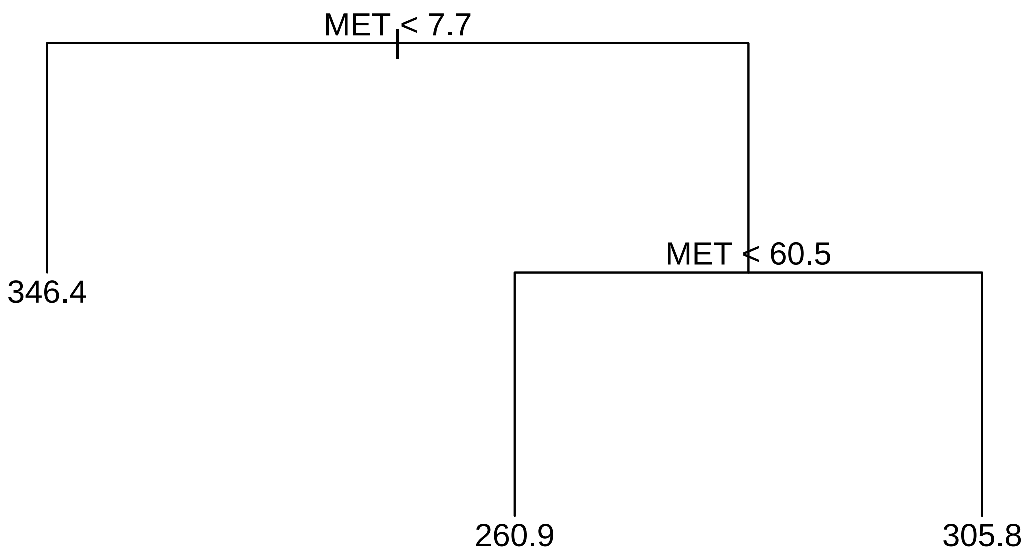
Since we have two parameters, secession tree based model can be used here for prediction of EX based on values of MET.

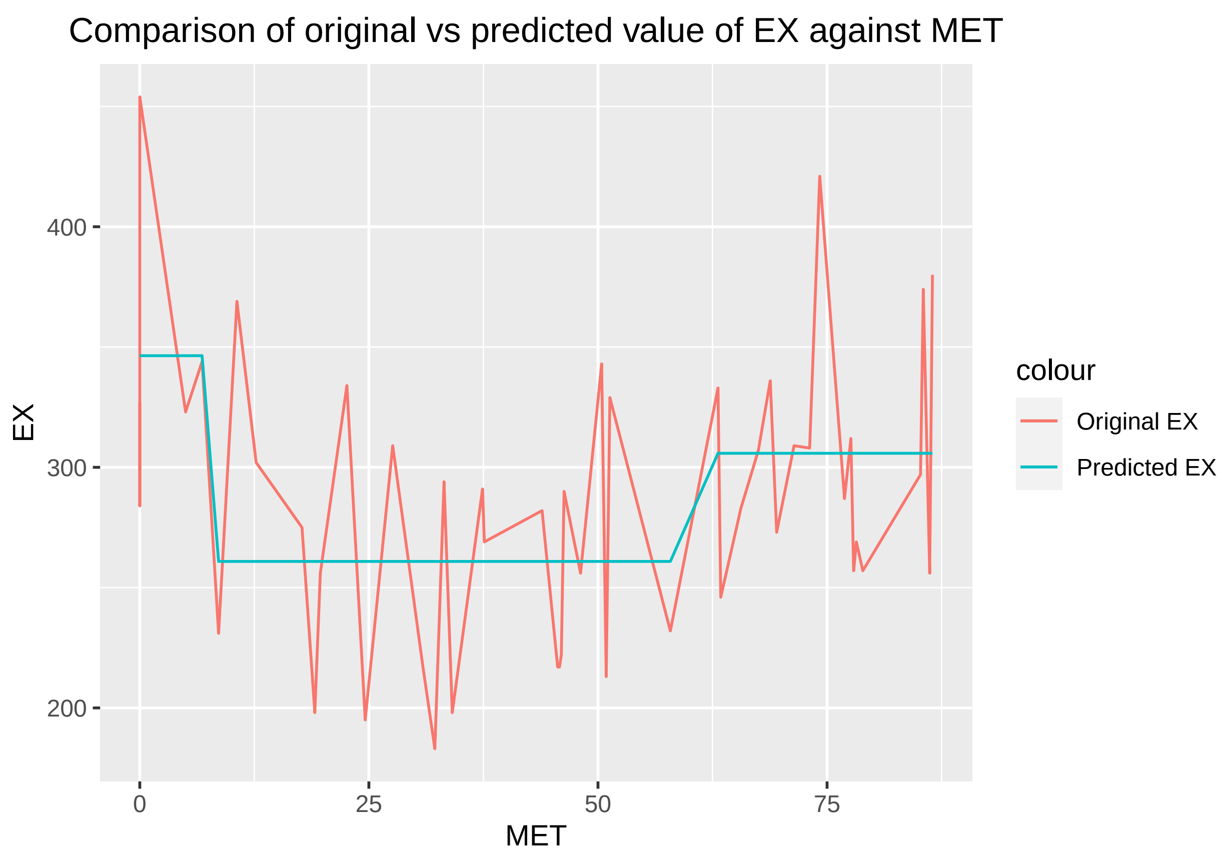
Part 3.2

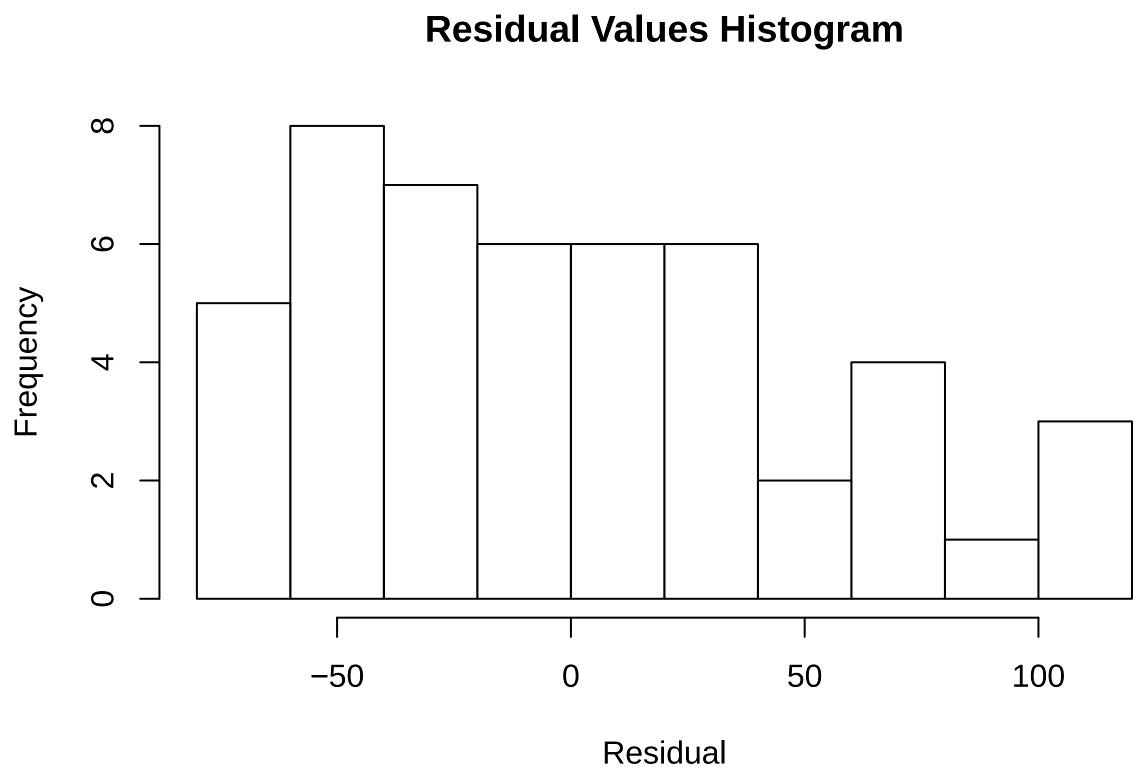


```
## Looking at Deviance vs Number of tree plot, it can be seen that minimum deviance is obtained when
##     number of leaves is  3 .
```

```
## Optimal Tree with asked requirement
```


MET < 7.7
346.4
MET < 60.5
260.9
305.8


Comparison of original vs predicted value of EX against MET
EX
400
300
200
0
25
50
75
MET
colour
Original EX
Predicted EX

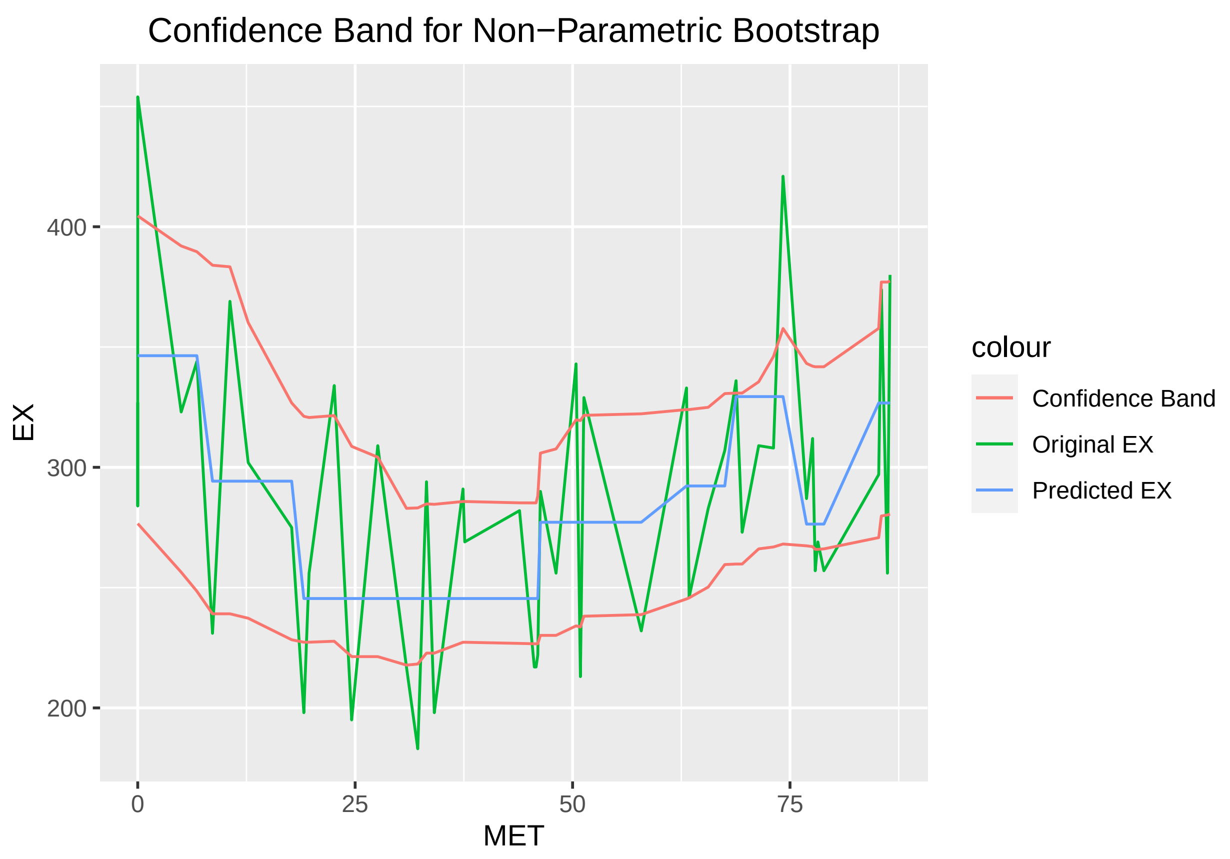


Distribution of Residual

While looking at the above graph, it can observe that it is right skewed and have a wide distribution. Distribution of residual resembel gama distribution.

Quality of Fit From residuals histogram, it can be seen that they are highly density in -50-0 which conveys that model predicts less than the actual values and ideally, residuals should be zero or should be around zero. Thus it can be concluded that the current model is not a very good fit.

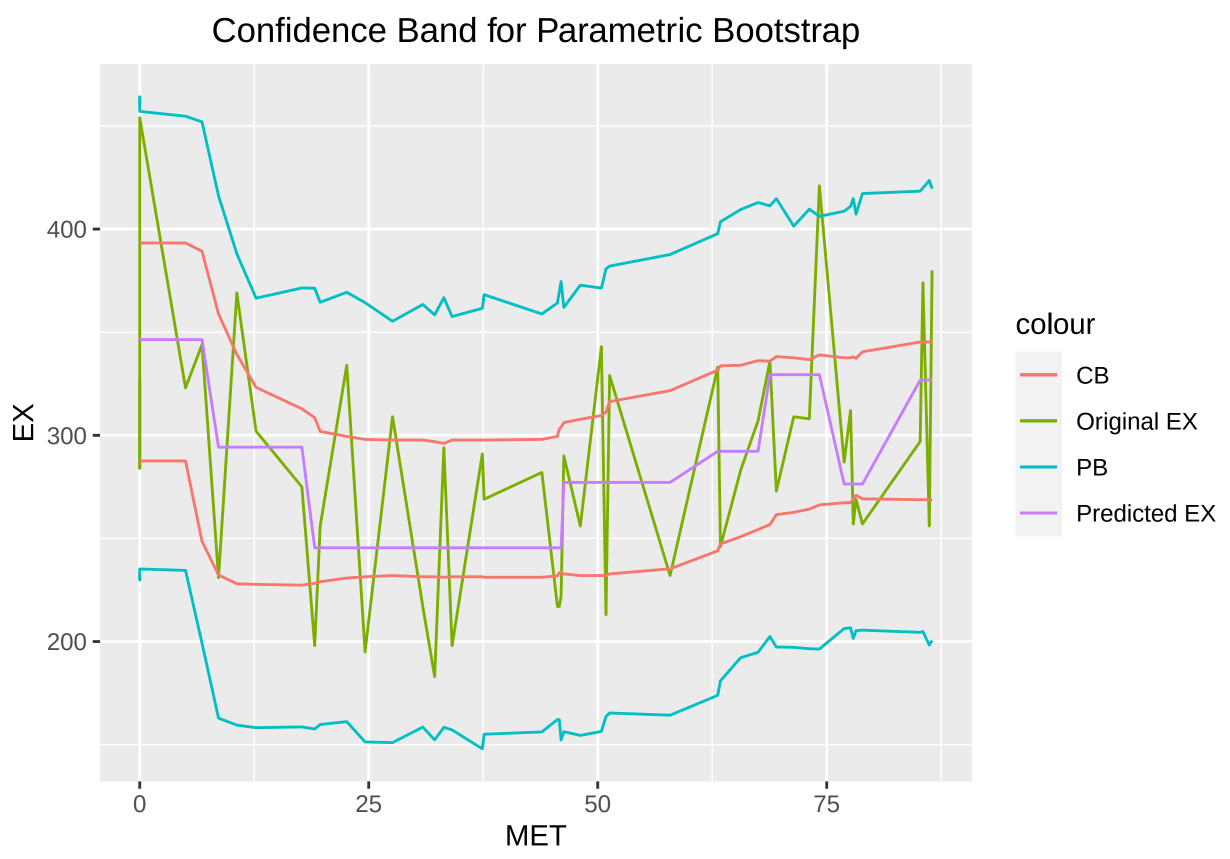
Part 3.3



It can be seen that 95% confidence band is bumpy in context to bootstrapped sample, also presence of various outliers can be seen because re-sampling is done based on the discrete distribution of prediction values thus discrete prediction points are obtained and hence bumpy confidence band.

Since distribution of error not know non-parametric confidence interval is reliable.

Part 3.4



From the above plot, it can be seen that confidence interval is smooth when compared with non-parametric bootstrap confidence interval since it was assumed that data is normally distributed and thus predictions are assumed to be continuous.

We can see only one point lies outside prediction band which is approximately within 5% as prediction band is expected to cover 95% of predictions.

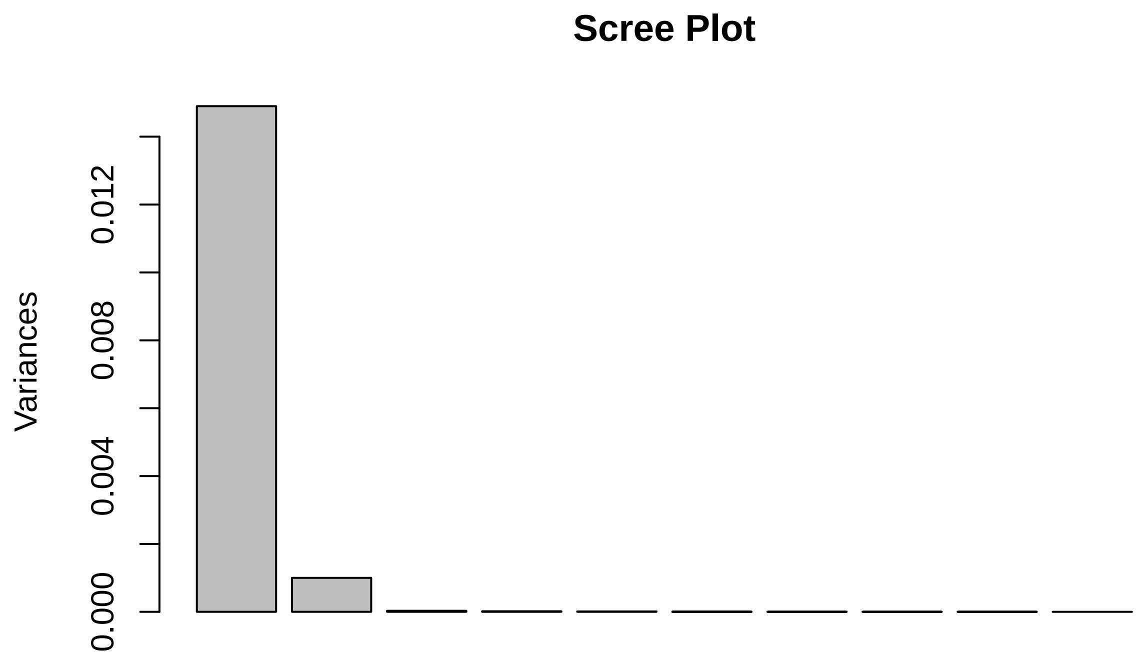
Part 3.5

Parametric bootstrap performs better than non-parametric bootstrap given assumed distribution of parametric bootstrap is correct. At the same time, the histogram of residuals is not normally distributed; thus using non-parametric bootstrap is the better choice here.

Assignment 4

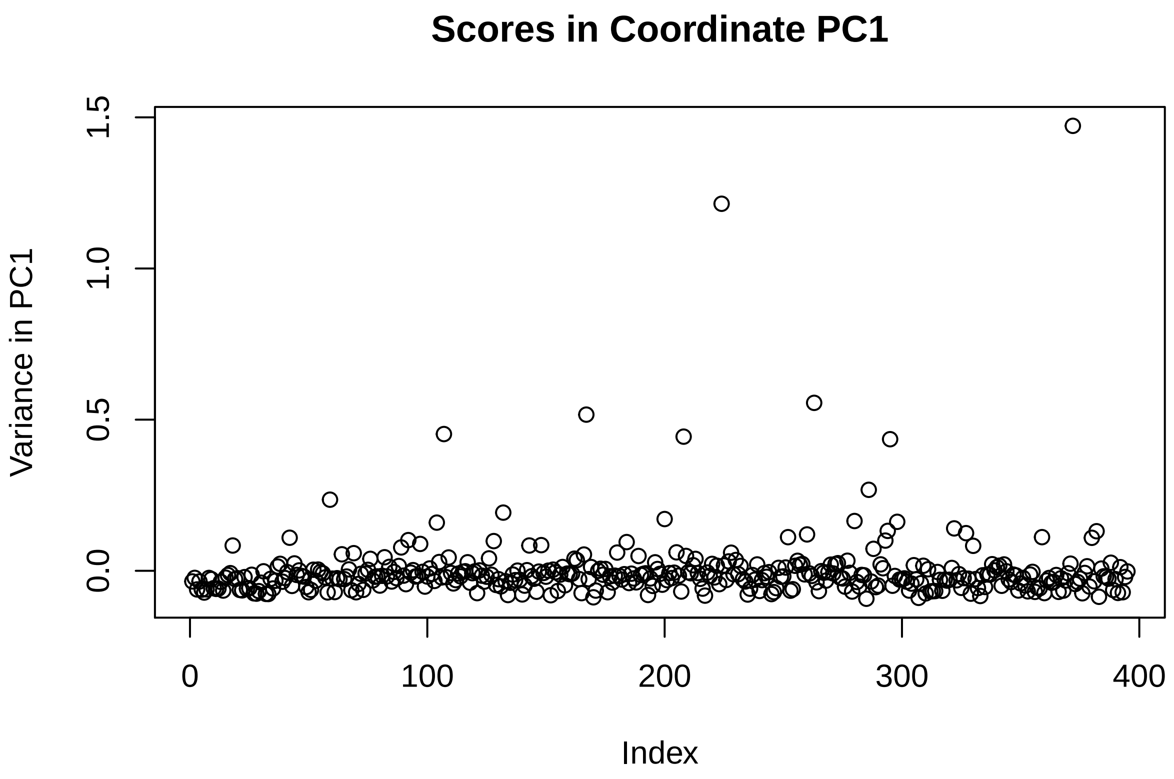
Part 4.1

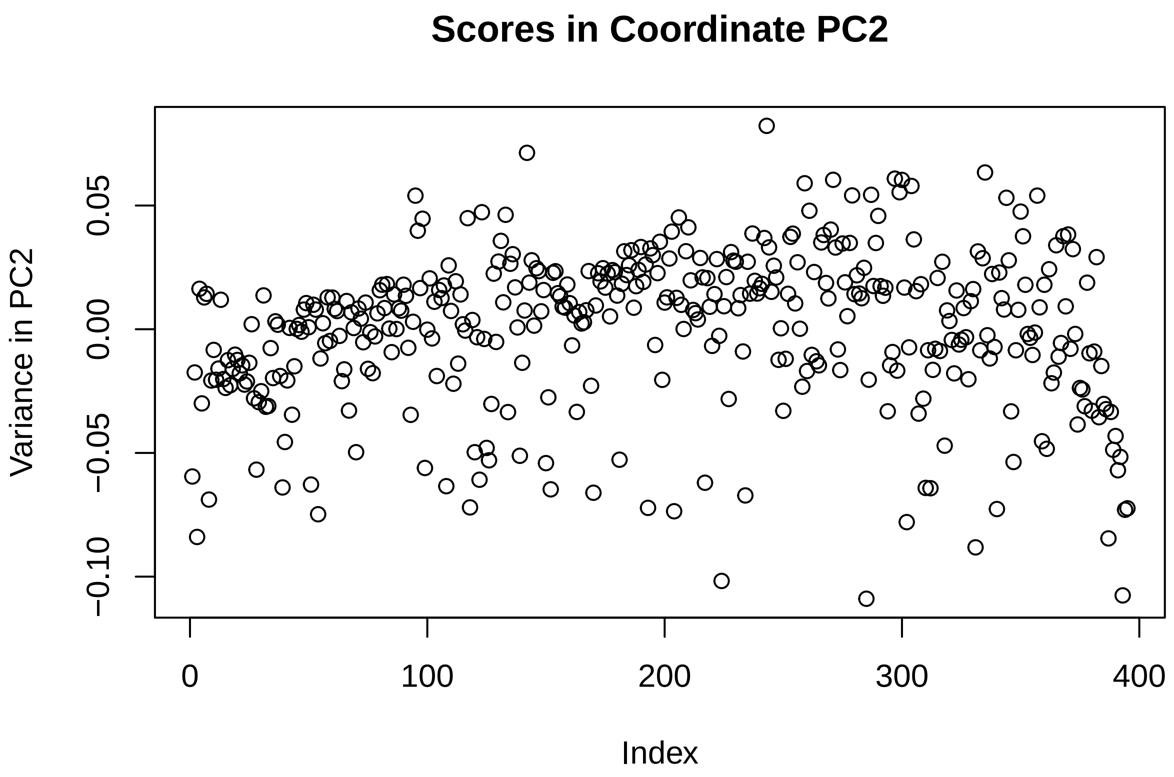
```
## Variation Explained by components

##  [1] "93.332" "6.263"  "0.185"  "0.101"  "0.068"  "0.025"  "0.009"  "0.003"
##  [9] "0.003"  "0.002"  "0.001"  "0.001"  "0.001"  "0.001"  "0.000"  "0.000"
## [17] "0.000"  "0.000"  "0.000"  "0.000"  "0.000"  "0.000"  "0.000"  "0.000"
## [25] "0.000"  "0.000"  "0.000"  "0.000"  "0.000"  "0.000"  "0.000"  "0.000"
## [33] "0.000"  "0.000"  "0.000"  "0.000"  "0.000"  "0.000"  "0.000"  "0.000"
## [41] "0.000"  "0.000"  "0.000"  "0.000"  "0.000"  "0.000"  "0.000"  "0.000"
## [49] "0.000"  "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"  "0.000"
## [65] "0.000"  "0.000"  "0.000"  "0.000"  "0.000"  "0.000"  "0.000"  "0.000"
##  [73] "0.000"  "0.000"  "0.000"  "0.000"  "0.000"  "0.000"  "0.000"  "0.000"
##  [81] "0.000"  "0.000"  "0.000"  "0.000"  "0.000"  "0.000"  "0.000"  "0.000"
##  [89] "0.000"  "0.000"  "0.000"  "0.000"  "0.000"  "0.000"  "0.000"  "0.000"
##  [97] "0.000"  "0.000"  "0.000"  "0.000"  "0.000"  "0.000"  "0.000"  "0.000"
## [105] "0.000"  "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"  "0.000"
## [121] "0.000"  "0.000"  "0.000"  "0.000"  "0.000"  "0.000"
```



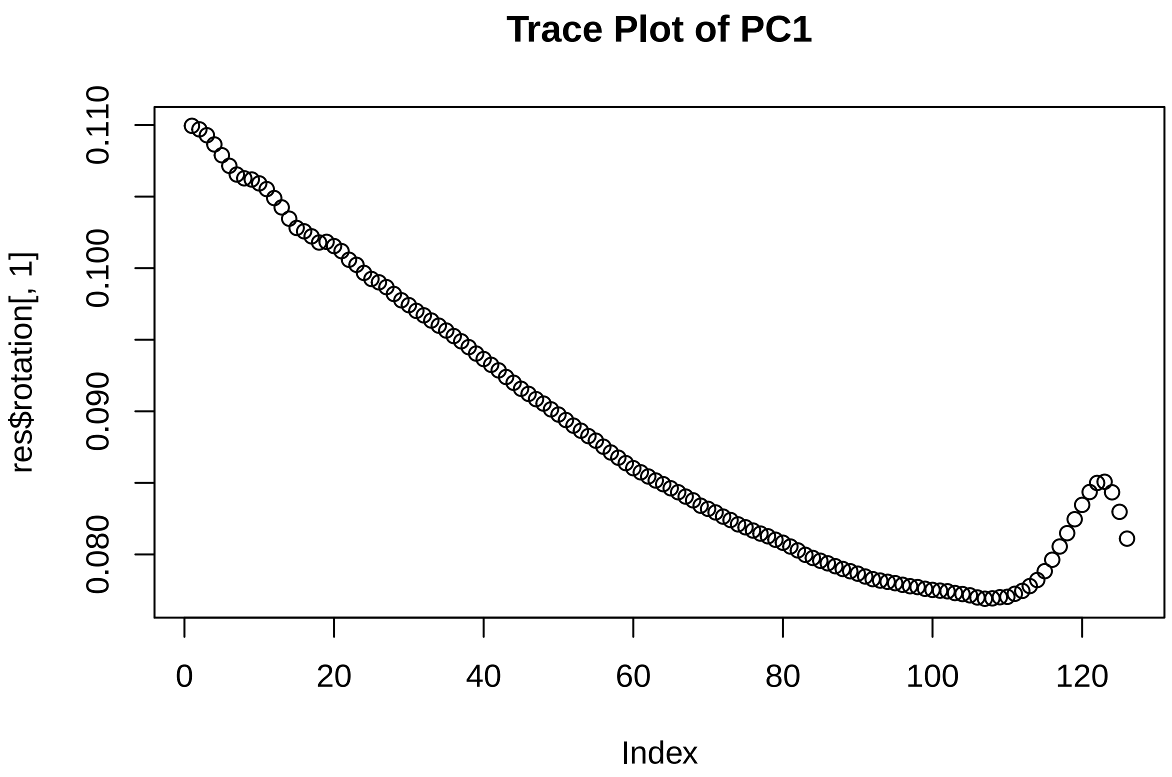
It can be seen that with principle component (PC) 1 and 2 we can explain 93.32 and 6.263 percentage of variations respectively.

Plot shows that 2 PC, PC1 and PC2 need to be extracted in order to explain at least 99% of variation.

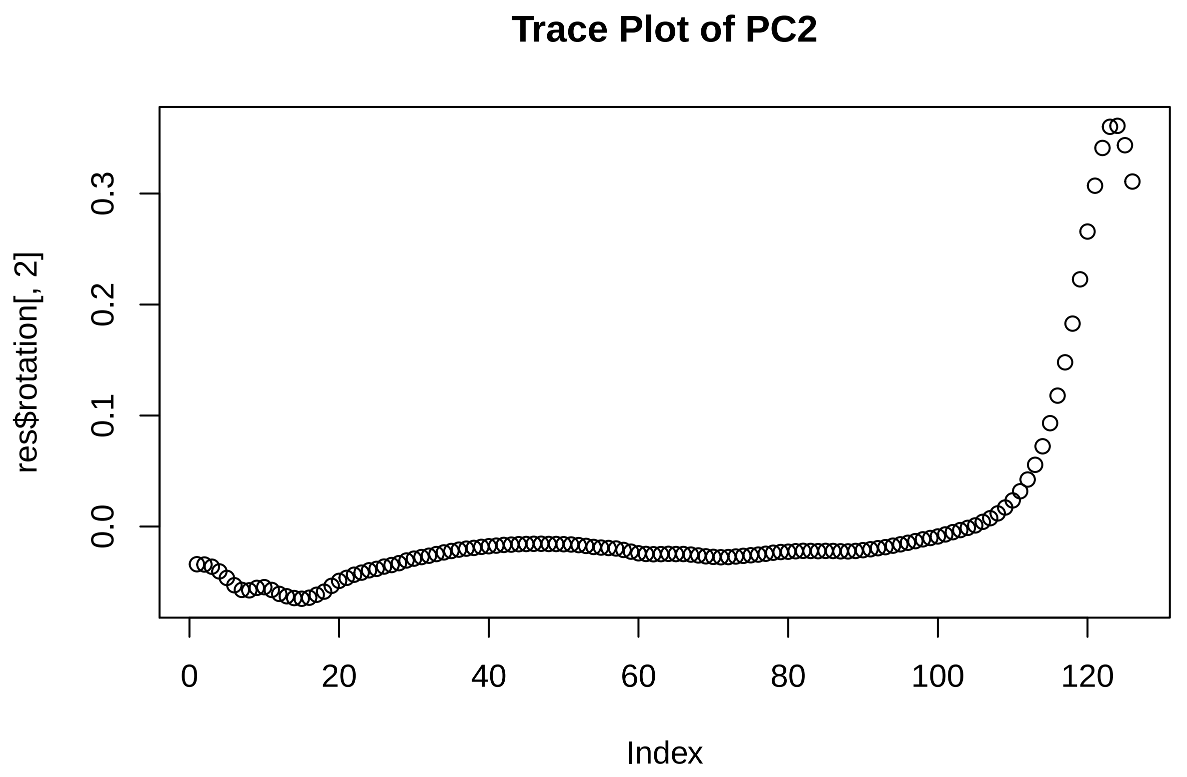

Scores in Coordinate PC1
Variance in PC1
Index
0
100
200
300
400
0.0
0.5
1.0
1.5



Based on presence of outliers for PC1, it can be seen that there are some unusual diesel fuel present here.

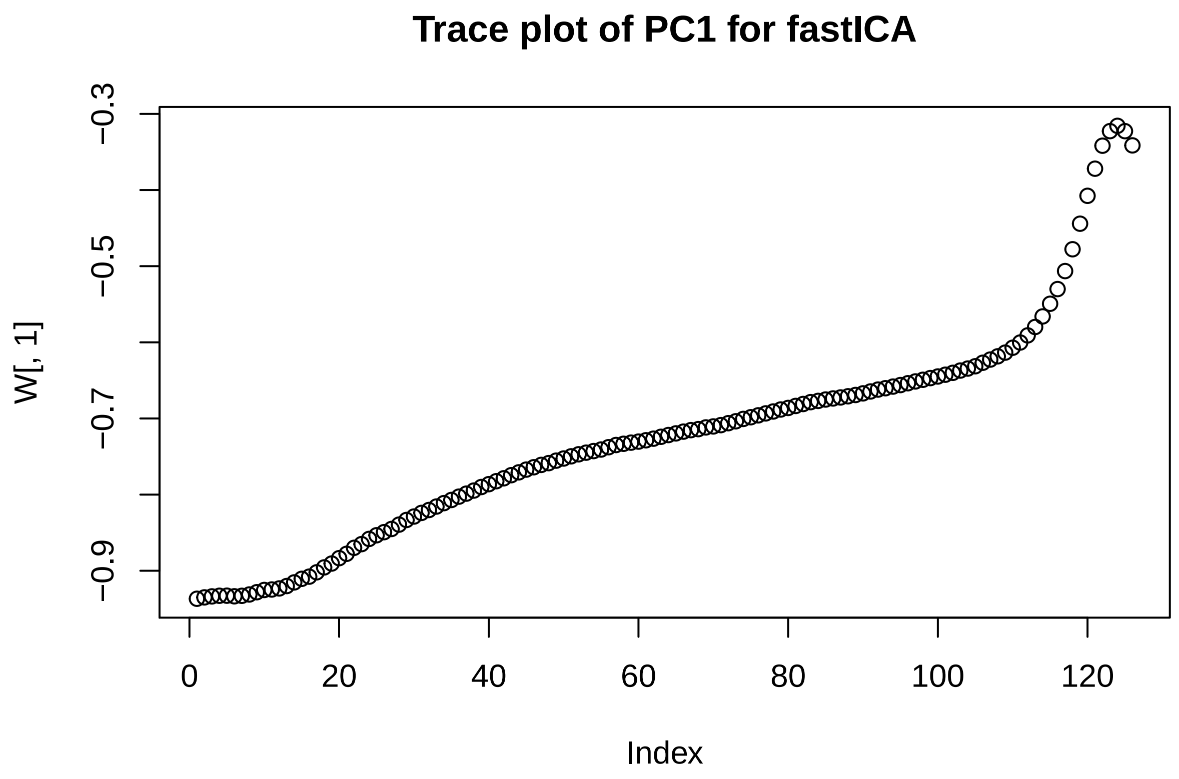

Trace Plot of PC1
res$rotation[, 1]
0.080
0.090
0.100
0.110
0
20
40
60
80
100
120
Index

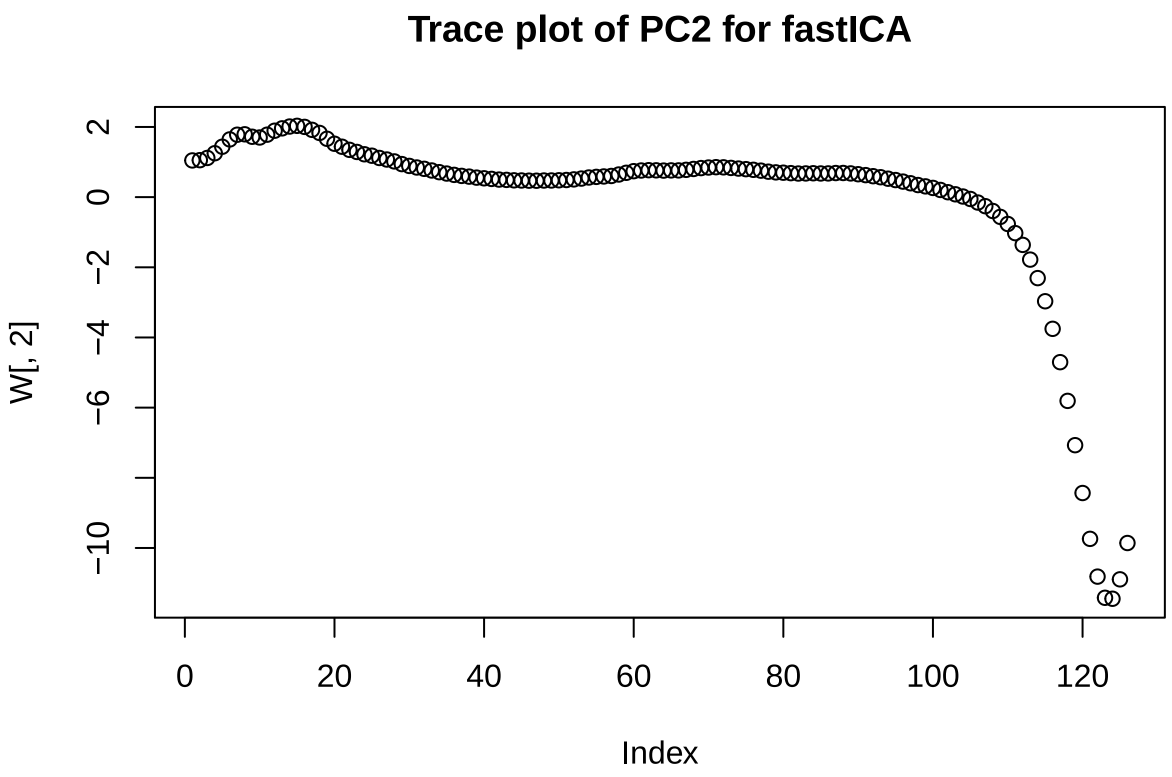
Part 4.2



It can be seen that PC2 is mainly explained by least number of features

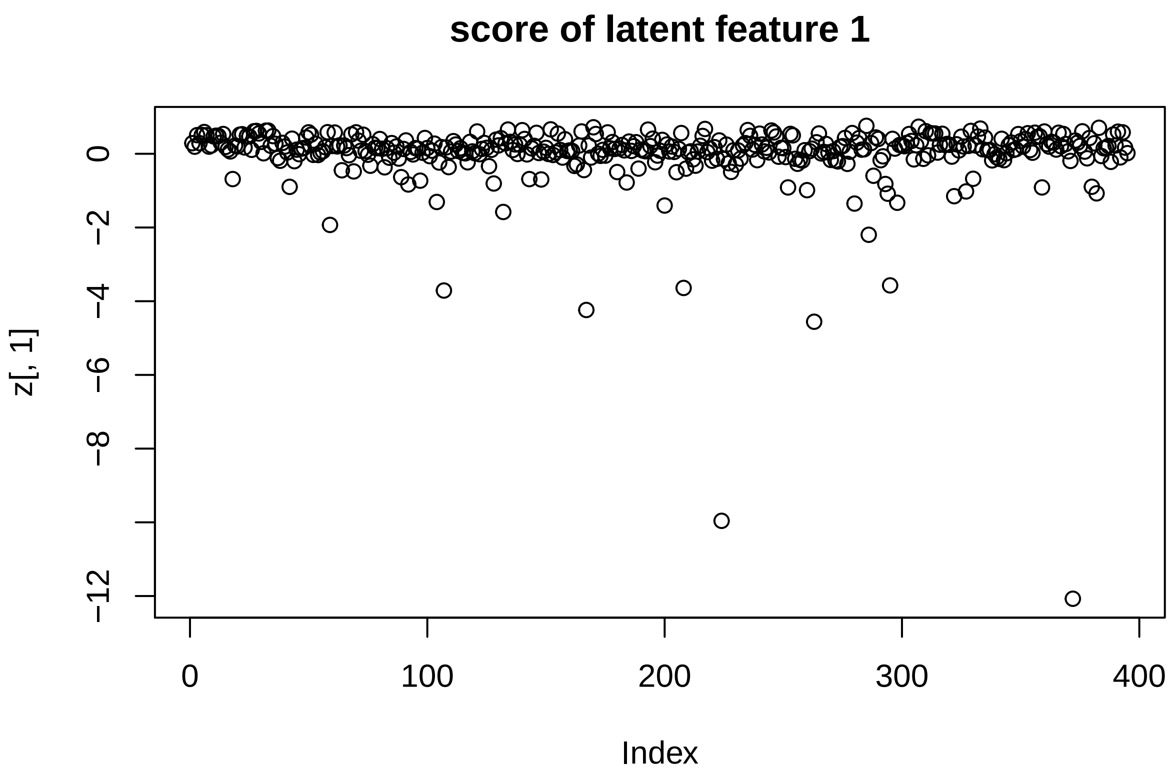
Part 4.3 a

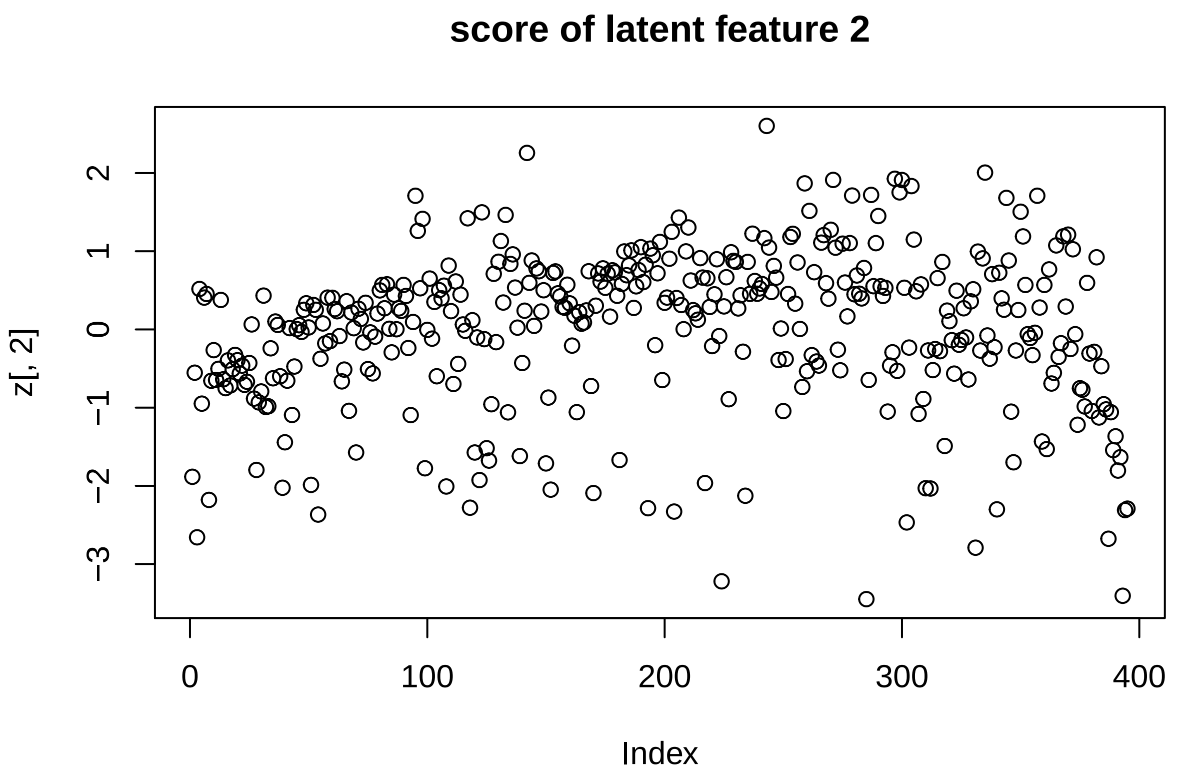

Trace plot of PC1 for fastICA
W[, 1]
−0.3
−0.5
−0.7
−0.9
0
20
40
60
80
100
120
Index



On comparing two methods and obtained Trace plots it can be seen that around 60% variation is explained by PC1 while remain target is achieved by PC2 unlike earlier one in which PC1 explain 93.3% variation.

In $W' = K.W$, W' represent project W to the first n principle components.


score of latent feature 1
z[, 1]
0
−2
−4
−6
−8
−12
0
100
200
300
400
Index



Again post observing score values for latent feature 1 and 2, it can be said based on latent feature 1, there are some unusual diesel fuel are present as per outliers.

Appendix

```
knitr::opts_chunk$set(echo = TRUE)
library("readxl")
library(MASS)
library(e1071)
library(tree)
library(grid)
library(readr)
library(tidyverse)
library(boot)
library(fastICA)
library(ggplot2)
loanData = data.frame(read_excel("/home/aman/Downloads/ML/2/creditscoring.xls"))

#Divide data between Train, Validate And Test

#Train
loanData$good_bad = as.factor(loanData$good_bad)
n = dim(loanData)[1]
suppressWarnings(RNGversion("3.5.9"))
set.seed(12345)
id = sample(1:n , floor(n*.5))
train = loanData[id,]
```

```r
#Validate
id1 = setdiff(1:n,id)
suppressWarnings(RNGversion("3.5.9"))
set.seed(12345)
id2 = sample(id1 , floor(n*.25))
valid = loanData[id2 , ]

#Test
id3 = setdiff(id1 , id2)
test = loanData[id3 , ]
fit_Deviance = tree(as.factor(good_bad)~. , data = train , split = "deviance")
# summary(fit_Deviance)
#  plot(fit_Deviance)
#  text(fit_Deviance , pretty = 0)

#Train Data

deviancePredTrain = predict(fit_Deviance , newdata = train , type = "class")

miscalTrain = table(train$good_bad , deviancePredTrain)

miscalRateTrain  = 1 - (sum(diag(miscalTrain)) / sum(miscalTrain))


#Test Data
deviancePred = predict(fit_Deviance , newdata = test , type = "class")

miscal = table(test$good_bad , deviancePred)

miscalRate  = 1 - (sum(diag(miscal)) / sum(miscal))

cat("Misclassification rate for Training Data when Impurity measure is Deviance : ", miscalRateTrain ,

cat("\n")

cat("Misclassification rate for Testing Data when Impurity measure is Deviance : ", miscalRate , "\n")

fit_gini = tree(as.factor(good_bad)~. , data = train , split = "gini")

#Train
giniPredTrain = predict(fit_gini , newdata = train , type = "class")

miscalGiniTrain = table(train$good_bad , giniPredTrain)

miscalRateGiniTrain  = 1 - (sum(diag(miscalGiniTrain)) / sum(miscalGiniTrain))


#Test
giniPred = predict(fit_gini , newdata = test , type = "class")

miscalGini = table(test$good_bad , giniPred)

miscalRateGini  = 1 - (sum(diag(miscalGini)) / sum(miscalGini))
```

```r
cat("\n")

cat("Misclassification rate for Training Data when Impurity is measure is Gini Index : ", miscalRateGini

cat("\n")

cat("Misclassification rate for Testing Data when Impurity is measure is Gini Index : ", miscalRateGini


suppressWarnings(RNGversion("3.5.1"))
set.seed(12345)
optimalTree = cv.tree(fit_Deviance)
#plot(optimalTree$size , optimalTree$dev , type = "b" , col = "red")

#optimal number of leaves
oLeaves = max(optimalTree$size)

#based on Train And Validate Data

trainScore = vector(length = (as.numeric(oLeaves)))
testScore  = vector(length = (as.numeric(oLeaves)))
newMisCal  = vector(length = (as.numeric(oLeaves)))

for(i in 2:oLeaves)
  {
    pruneTree = prune.tree(fit_Deviance , best = i)
    newPred = predict(pruneTree , newdata = valid , type = "tree")

    #calculate deviance for test and valid data
    trainScore[i] = deviance(pruneTree)
    testScore[i]  = deviance(newPred)

    #miscal Part
    newPred1 = predict(pruneTree , newdata = valid , type = "class")
    misCalTable = table(newPred1 , valid$good_bad)
    newMisCal[i] = 1 - sum(diag(misCalTable)/sum(misCalTable))
  }

#newMisCal = newMisCal[-1]

#lowestMisCal = which(newMisCal == min(newMisCal) & as.numeric(newMisCal) != 0)

#plot(x = 2:oLeaves , y = newMisCal[-1])

#which(newMisCal == min(newMisCal[-1]))

# plot(2:oLeaves , trainScore[2:oLeaves], type = "b" , col = "red" , ylim = c(0,600) , main = "Dependen
# data on the number of leaves" , xlab = "Number of Leaves" , ylab = "Deviances")
# points(2:oLeaves, testScore[2:oLeaves], type = "b" , col = "blue")

ggplot()+
geom_line(aes(x = 2:oLeaves , y = trainScore[2:oLeaves] , color = "Training Data"))+
geom_line(aes(x = 2:oLeaves , y = testScore[2:oLeaves]  , color = "Validation Data"))+
ylab("Deviance") + xlab("Number of Leaves") +
ggtitle("Dependence of deviance on the number of leaves")+
ylim(100,700)+
theme(plot.title = element_text(hjust = 0.5))




terminalNode = which(newMisCal == min(newMisCal[-1]))

cat("\n")
cat("Optimal number of leaves based on Miscal Rate of Validation Data is : " , terminalNode[1] ,"\n")
cat("\n")

optimalTree = prune.tree(fit_Deviance , best = terminalNode[1])

cat("Optimal Tree is" , "\n")
plot(optimalTree , main = "Optimal Tree")
text(optimalTree , pretty = 0)
cat("\n")



#misCal test data (also calculated for Train comparission in step 4)
#Train
treeTrainPred = predict(optimalTree , newdata = train , type = "class")
tTrainMCT = table(treeTrainPred , as.factor(train$good_bad))
tTrainMCR = 1 - (sum(diag(tTrainMCT)) / sum(tTrainMCT))

cat("\n")
cat("Confusion Matrix for Optimal Decision Tree Train data is as below","\n")
tTrainMCT
cat("\n")

cat("Miscalculation Rate for Optimal Decision Tree Train data is : " , tTrainMCR )
cat("\n")
cat("\n")
#Test
testPred = predict(optimalTree , newdata = test , type = "class")
MCT = table(testPred , as.factor(test$good_bad))
MCR = 1 - (sum(diag(MCT)) / sum(MCT))

cat("\n")
cat("Confusion Matrix for Decision Tree Test data is as below","\n")
MCT

cat("\n")
cat("Miscalculation Rate for Decision Tree Test data is : " , MCR )
cat("\n")

cat("\n")
cat("Naive Bayes" , "\n")
```

```r
NaiveFit = naiveBayes(good_bad ~. , data = train)

nTrainPred = predict(NaiveFit , newdata = train)

nTrainMCT = table(nTrainPred , train$good_bad)

nTrainMCR = 1 - (sum(diag(nTrainMCT)) / sum(nTrainMCT))

cat("\n")
cat("Confusion Matrix for Naive Bayes Train Data" , "\n")
nTrainMCT

cat("\n")
cat("Misclassification Rate for Naive Bayes Train Data is " , nTrainMCR ,  "\n")

nTestPred = predict(NaiveFit , newdata = test)

nTestMCT = table(nTestPred , test$good_bad)

nTestMCR = 1 - (sum(diag(nTestMCT)) / sum(nTestMCT))

cat("\n")
cat("Confusion Matrix for Naive Bayes Test Data" , "\n")
nTestMCT

cat("\n")
cat("Misclassification Rate for Naive Bayes Test Data is " , nTestMCR ,  "\n")

#ROC : Receiver Operating Characteristics
#TPR = TP / (TP+FN)
#FPR = FP / (TN + FP)

pi = seq(0.05 , 0.95 , 0.05)

otTPR = c()
otFPR = c()
#otR = c()
nTPR = c()
nFPR = c()
#nR = c()

#testPred > optimal Tree Test
#recalculated for numerical value

testPred = as.data.frame(predict(optimalTree , newdata = test , type = "vector"))

#nTestPred > Naive Bayes Test
nTestPred = predict(NaiveFit , newdata = test , type = "raw")

for(i in 1:length(pi))
{
 #Optimal Tree
 treeCondition = ifelse(testPred[,2] > pi[i] , "good" , "bad")
```

```r
  treeCondition = factor(treeCondition , levels = c("bad","good"))
  tMCT = table(treeCondition , test$good_bad)
  otTPR[i] = as.integer(tMCT[2,2])/sum(tMCT[,2])
  otFPR[i] = as.integer(tMCT[2,1])/sum(tMCT[,1])

  #Naive Bayes
  naiveCondition = ifelse(nTestPred[,2] > pi[i] , "good" , "bad")
  nMCT = table(naiveCondition , test$good_bad)
  nTPR[i] = as.integer(nMCT[2,2])/sum(nMCT[,2])
  nFPR[i] = as.integer(nMCT[2,1])/sum(nMCT[,1])
}

#ROC Curve
ggplot()+
geom_line(aes(x = otFPR , y = otTPR , color = "Optimal Tree"))+
geom_line(aes(x = nFPR , y = nTPR , color = "Naive Bayes"))+
ylab("TPR") + xlab("FPR") + ggtitle("ROC Curve")+
theme(plot.title = element_text(hjust = 0.5))


#creating asked loss matrix
#Change corrected for byrow
lossMatrix = matrix(data = c(0 , 10 , 1 , 0) , nrow = 2 , ncol = 2 ,
                    #byrow = TRUE
                    )

#Train
nLMfit = predict(NaiveFit , newdata = train, type = "raw")
#head(nLMfit , 5)

nLMfit[,2] = nLMfit[,2]*lossMatrix[1,2]
nLMfit[,1] = nLMfit[,1]*lossMatrix[2,1]
cond = ifelse(nLMfit[,2] > nLMfit[,1] , "good" , "bad")
lmNTMCT = table(cond , train$good_bad)
lmNTMCR = 1 - (sum(diag(lmNTMCT))/sum(lmNTMCT))

cat("\n")
cat("Confusion Matrix for Naive Bayes Training Data with given Loss Matrix")
lmNTMCT

cat("\n")
cat("Miscalculation Rate for Naive Bayes Training Data with given Loss Matrix is :",
    lmNTMCR , "\n" )

cat("\n")
cat("\n")

#Test Data
nLMfitTest = predict(NaiveFit , newdata = test , type = "raw")
nLMfitTest[,2] = nLMfitTest[,2]*lossMatrix[1,2]
nLMfitTest[,1] = nLMfitTest[,1]*lossMatrix[2,1]
cond = ifelse(nLMfitTest[,2] > nLMfitTest[,1] , "good" , "bad")
lmTestMCT = table(cond , test$good_bad)
```

```r
lmTestMCR = 1 - (sum(diag(lmTestMCT))/sum(lmTestMCT))

cat("\n")
cat("Confusion Matrix for Naive Bayes Test Data with given Loss Matrix")
lmTestMCT

cat("\n")
cat("Miscalculation Rate for Naive Bayes Test Data with given Loss Matrix is :",
    lmTestMCR , "\n" )

#
# Change in misclassification rate, post increasing penalty on misclassification it can be seen that ev
stateData = data.frame(read.csv2("/home/aman/Downloads/ML/2/State.csv"))

#Reorder based on value of MET
stateData = stateData[order(stateData$MET),]

ggplot()+
geom_line(aes(y = stateData$EX , x = stateData$MET))+
#geom_label(color = "Ex = Per capita state and local public expenditures in $")+
xlab("MET - Percentage of population living in standard metropolitan areas") +
ylab("EX - Per capita state and local public expenditures in $ ") +
ggtitle("EX versus MET") +
theme(plot.title = element_text(hjust = 0.5))


tree123 = tree(EX~MET , data = stateData , split = "deviance")

cat("Decision Tree for given parameters")
plot(tree123)
text(tree123 , pretty = 0)

suppressWarnings(RNGversion("3.5.1"))
set.seed(12345)
library(tree)
stateTree = tree(EX~MET , data = stateData ,
              control =  tree.control(minsize = 8 , nobs = nrow(stateData)))
#plot(stateTree)
#set.seed(12345)

cvTree = cv.tree(stateTree)

ggplot()+
geom_line(aes(x = cvTree$size , y = cvTree$dev))+
ylab("Deviance") + xlab("Number of Leaves")  +
#ylim(160000,200000)
ggtitle("Deviance vs Number of Leaves") +
theme(plot.title = element_text(hjust = 0.5))

minDevLeave = 3
cat("Looking at Deviance vs Number of tree plot, it can be seen that minimum deviance is obtained when
    number of leaves is " , minDevLeave ,".")
```

```r
optimalT = prune.tree(stateTree , best = minDevLeave)

cat("Optimal Tree with asked requirement" , "\n")

plot(optimalT , main = "Optimal Tree with asked requirement")
text(optimalT , pretty = 0)

predictEX = predict(optimalT , newdata = stateData)
residualEX = stateData$EX - predictEX
#residualDF = data.frame(residualEX , stateData$MET)

ggplot()+
geom_line(aes(x = stateData$MET , y = stateData$EX , color = "Original EX"))+
geom_line(aes(x = stateData$MET , y = predictEX , color = "Predicted EX"))+
# geom_histogram(data = residualDF,
#           aes(x = stateData.MET , y = residualEX, color = "Residual") ,
#           stat = "identity") +
ylab("EX") + xlab("MET") +
ggtitle("Comparison of original vs predicted value of EX against MET") +
theme(plot.title = element_text(hjust = 0.5))

hist(residualEX , xlab = "Residual" ,  main = "Residual Values Histogram"
     #, probability = T
     )
#lines(density(residualEX) , col = "red" , lwd = 2)

#install.packages("goft")
# library(goft)
# c = cauchy_test(residualEX)
# c


#Miscalculation Rate

# T4MCT = table(predictEX , stateData$EX)
# T4MCR = 1 - (sum(diag(T4MCT))/sum(T4MCT))
#
# cat("\n")
# cat("Misclassification Rate is : ",T4MCR,"\n")

predictFun = function(data,index){
  bootData = data[index,]
  model = tree(EX ~ MET , data = bootData ,
               control =  tree.control(minsize = 8 , nobs = nrow(bootData)))
  pruneModel = prune.tree(model , best = minDevLeave)
  return(predict(pruneModel , newdata = data))
}

suppressWarnings(RNGversion("3.5.1"))
set.seed(12345)
bootStrap = boot(data = stateData , statistic = predictFun , R = 1000)

#confidence interval
```

```r
#CI = envelope(bootStrap , level = .95)
CI = t(apply(bootStrap$t, MARGIN = 2, quantile , probs = c(0.025 ,0.975)))


#cat("Percentile - 95% Confidence interval for Non-Parametric Bootstrap is in Range : " , CI$point[2] ,
#CI$normal[2]

#########
bootTree123 = tree(EX ~ MET , data = stateData , control = tree.control(nobs = 48 , minsize = 8))

predticedBT123 = predict(bootTree123)

ggplot()+
geom_line(aes(x = stateData$MET , y = stateData$EX , color = "Original EX"))+
geom_line(aes(x = stateData$MET , y = predticedBT123 , color = "Predicted EX"))+
geom_line(aes(x = stateData$MET , y = CI[,2] , color = "Confidence Band"))+
geom_line(aes(x = stateData$MET , y = CI[,1] , color = "Confidence Band"))+
ylab("EX") + xlab("MET") +
ggtitle("Confidence Band for Non-Parametric Bootstrap") +
theme(plot.title = element_text(hjust = 0.5))
#plot confidence band
#
# ggplot()+
# geom_line(aes(x = stateData$MET , y = stateData$EX , color = "Original EX"))+
# geom_line(aes(x = stateData$MET , y = predictEX , color = "Predicted EX"))+
# geom_line(aes(x = stateData$MET , y = CI$point[2,] , color = "Confidence Band"))+
# geom_line(aes(x = stateData$MET , y = CI$point[1,] , color = "Confidence Band"))+
# ylab("EX") + xlab("MET") +
# ggtitle("Confidence Band for Non-Parametric Bootstrap") +
# theme(plot.title = element_text(hjust = 0.5))

#Paramteric Bootstrap

#mle = lm(EX~MET , data = stateData)

mle = prune.tree(tree(EX ~ MET , stateData , control = tree.control(nobs = 48 , minsize = 8)) , best =


randomGen = function(data , mle){
  data_new = data.frame(EX = data$EX , MET = data$MET)
  n = length(data$EX)
  data_new$EX = rnorm(n , predict(mle, newdata = data_new) , sd(residuals(mle)))
  return(data_new)
}

#normFun = function(data){
#    bootData = data
#    model = tree(EX ~ MET , data = bootData ,
#                control =  tree.control(minsize = 8
#                                        , nobs = length(bootData$EX)))
#    pruneModel = prune.tree(model , best = minDevLeave)

    #predictEX = predict(pruneModel , newdata = bootData)
    #return(rnorm(n = length(stateData$EX) , predictEX , sd(mle$residuals)))
#    return(predict(pruneModel , newdata = bootData))
#  }

parFun_cb = function(data){
  model = prune.tree(tree(EX ~ MET , data , control = tree.control(nobs = 48 , minsize = 8)) , best = 3
  pred = predict(model , newdata = stateData)
  return(pred)
}

parFun_pb = function(data){
  model = prune.tree(tree(EX ~ MET , data , control = tree.control(nobs = 48 , minsize = 8)) , best = 3
  pred = predict(model , newdata = stateData)
  n = length(stateData$EX)
  resid = residuals(mle)
  preds = rnorm(n , pred , sd = sd(resid))
  return(preds)
}


# predictedBoot = boot(data = stateData , statistic = normFun , sim = "parametric"
# confBand = envelope(predictedBoot , level = .95)
#
# cat("Percentile - 95% Confidence interval for Parametric Bootstrap is in Range : " , confBand$point[2
#CIpoints = envelope(bootStrap)




boot_cb = boot(data = stateData , statistic = parFun_cb , sim = "parametric"

boot_pb = boot(data = stateData , statistic = parFun_pb , sim = "parametric"


cb_param = envelope(boot_cb , level = 0.95)

pb_param = t(apply(boot_pb$t, MARGIN = 2, quantile , probs = c(0.025 , 0.975)))

ggplot()+
geom_line(aes(x = stateData$MET , y = stateData$EX , color = "Original EX"))+
geom_line(aes(x = stateData$MET , y = predticedBT123 , color = "Predicted EX"))+
geom_line(aes(x = stateData$MET , y = cb_param$point[2,] , color = "CB"))+
geom_line(aes(x = stateData$MET , y = cb_param$point[1,] , color = "CB"))+
geom_line(aes(x = stateData$MET , y = pb_param[,2] , color = "PB"))+
geom_line(aes(x = stateData$MET , y = pb_param[,1] , color = "PB"))+
ylab("EX") + xlab("MET") +
ggtitle("Confidence Band for Parametric Bootstrap") +
theme(plot.title = element_text(hjust = 0.5))


NIRdata = data.frame(read.csv2("/home/aman/Downloads/ML/2/NIRSpectra.csv"))

#set target feature to null
```

```r
NIRdata$Viscosity = c()

res = prcomp(NIRdata)
eigenValues = (res$sdev)^2
cat("\n")
cat("Variation Explained by components" , "\n")
sprintf("%2.3f" , eigenValues/sum(eigenValues)*100)
cat("\n")

screeplot(res , main = "Scree Plot")


# plot(res$x[,1] , res$x[,2]
#       #, ylim = c(-.2 ,.2) , xlim = c(0 , .4)
#       ,
#       xlab = "PC1" , ylab = "PC2", main = "Scores in the coordinates PC1 vs PC2")

plot(res$x[,1] , xlab = "Index" , ylab = "Variance in PC1",
     main = "Scores in Coordinate PC1")

plot(res$x[,2] , xlab = "Index" , ylab = "Variance in PC2",
     main = "Scores in Coordinate PC2")



plot(res$rotation[,1] , main = "Trace Plot of PC1")

plot(res$rotation[,2] , main = "Trace Plot of PC2")

# se = seq(-.5 , .3 , .01)
# plot(res$rotation[,1] , main = "Trace Plot of PC1" , ylim = se)
# points(res$rotation[,2])

suppressWarnings(RNGversion("3.5.1"))
set.seed(12345)
ICA = fastICA(NIRdata,2)

#Wt
W = ICA$K %*% ICA$W

plot(W[,1], main = "Trace plot of PC1 for fastICA")
plot(W[,2], main = "Trace plot of PC2 for fastICA")
#plot latent features
v = solve(ICA$W)
z = ICA$X %*% W
z = z %*% v
plot(z[,1] , main = "score of latent feature 1")
plot(z[,2] , main = "score of latent feature 2")
```