Lab2

Sridhar

10 December 2018

Contents

Assignment 2. Analysis of credit scoring

1. Reading Data and spliting into Train, Valid, Test

2. Decision Tree

```
## [1] "Deviance Impurity Measure - "
```

```
## [1] "Performance on Train -"
```

```
## Misclassification Rate :  0.78
```

```
##       p1
##        bad good
##   bad   94   53
##   good  57  296
```

```
## [1] "Performance on Test -"
```

```
## Misclassification Rate :  0.728
```

```
##       p2
##        bad good
##   bad   48   28
##   good  40  134
```

Uning the deviance impurity measure the classify focuses on accuracy. The classifier has high accuracy on both Train and Test data, but the precision will be low, as there are many of the good cases being classied as bad. If the focus of the classifier is accuracy then this would be a good impurity measure to choose.

```
## [1] "Gini Impurity Measure - "
```

```
## [1] "Performance on Train -"
```

```
## Misclassification Rate :  0.766
```

```
##       p1
##        bad good
##   bad   67   80
##   good  37  316
```

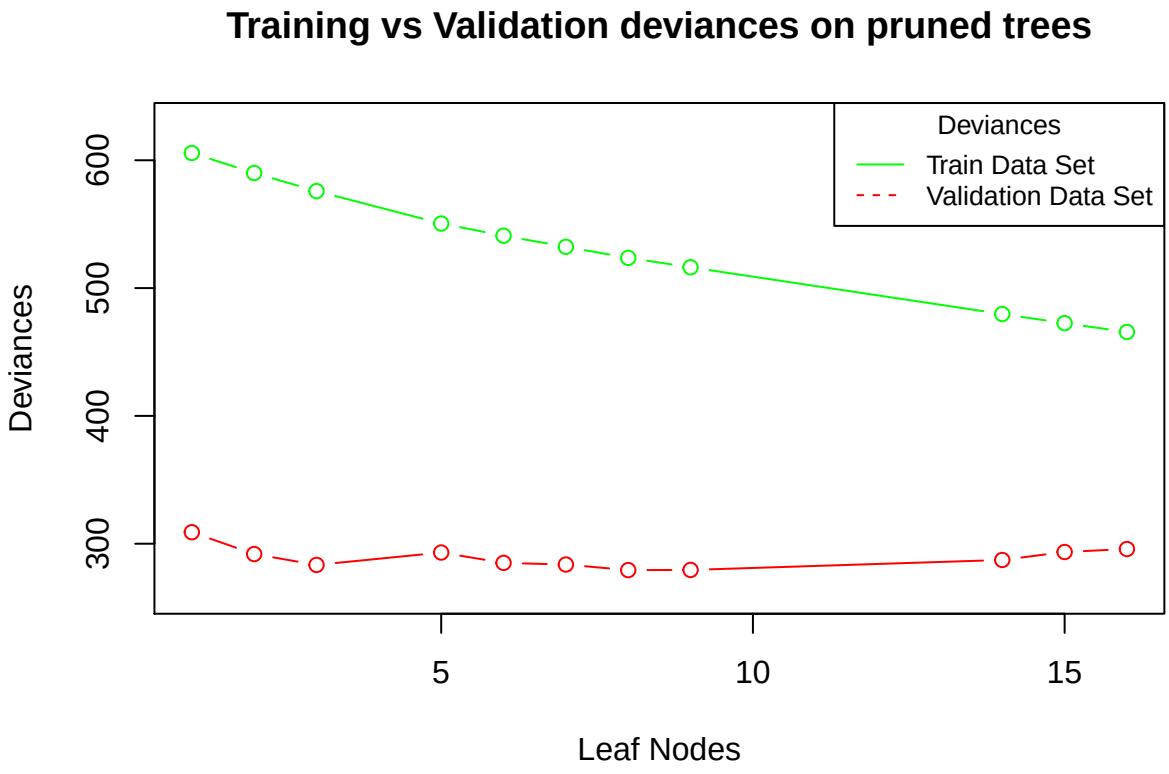
```
## [1] "Performance on test -"
```

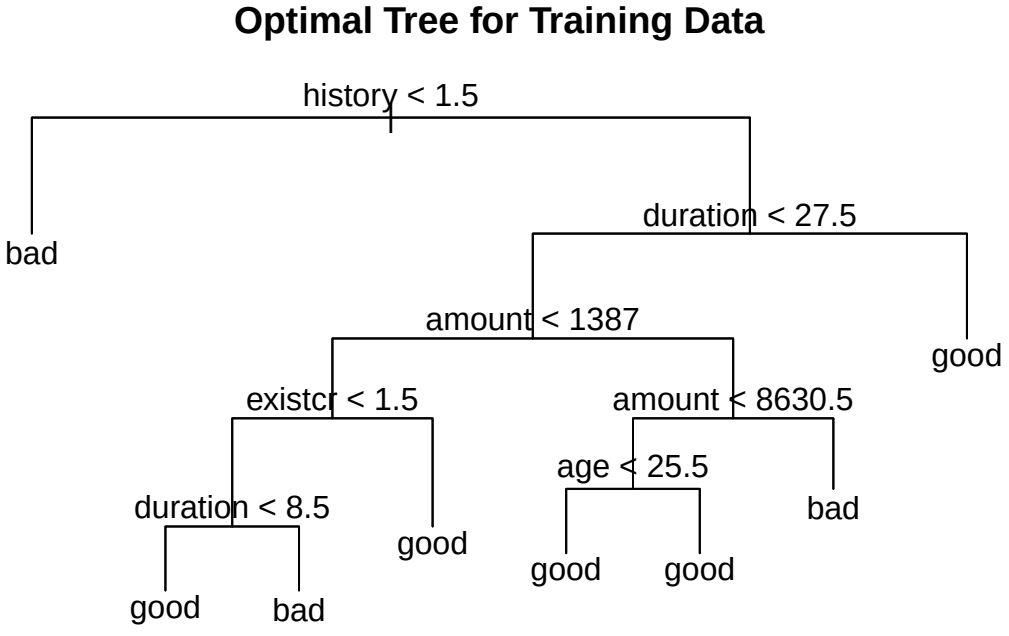
```
## Misclassification Rate :  0.62
```

```
##       p2
##        bad good
##   bad   15   61
##   good  34  140
```

These are the results I got using the Gini impurity measure. Gini has lower accuracy as compared to the Deviance model as this focuses on increasing the precision of the system. Using gini the number of good customers being marked as bad is reduced, but this also reduces the number of correct bad predictions as the classifier increased the threshold for a customer to be classified as bad. This is good if the focus of the required classifier is precision.

3. Finding Optimal Tree





```
## Depth of Optimal Tree for training data: 5
```

```
## Number of Leaf Nodes in Optimal Tree for training data: 8
```

```
##
## Labels used by training data:
```

```
##  [1] "root"            "history < 1.5"   "history > 1.5"
##  [4] "duration < 27.5" "amount < 1387"   "existcr < 1.5"
##  [7] "duration < 8.5"  "duration > 8.5"  "existcr > 1.5"
## [10] "amount > 1387"   "amount < 8630.5" "age < 25.5"
## [13] "age > 25.5"      "amount > 8630.5" "duration > 27.5"
```

Since the deviance impurity measure model had better accuracy I will be using that in this question.

The optimal tree found had depth 5 with 8 leaf nodes. This is the size of the tree that had the lowest error on th validation set.

4. Naive Baiyes and comparison with optimal tree

```
## [1] "Naive Bayes Model - "
```

```
## [1] "Performance on Train -"
```

```
## Misclassification Rate :  0.7
```

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

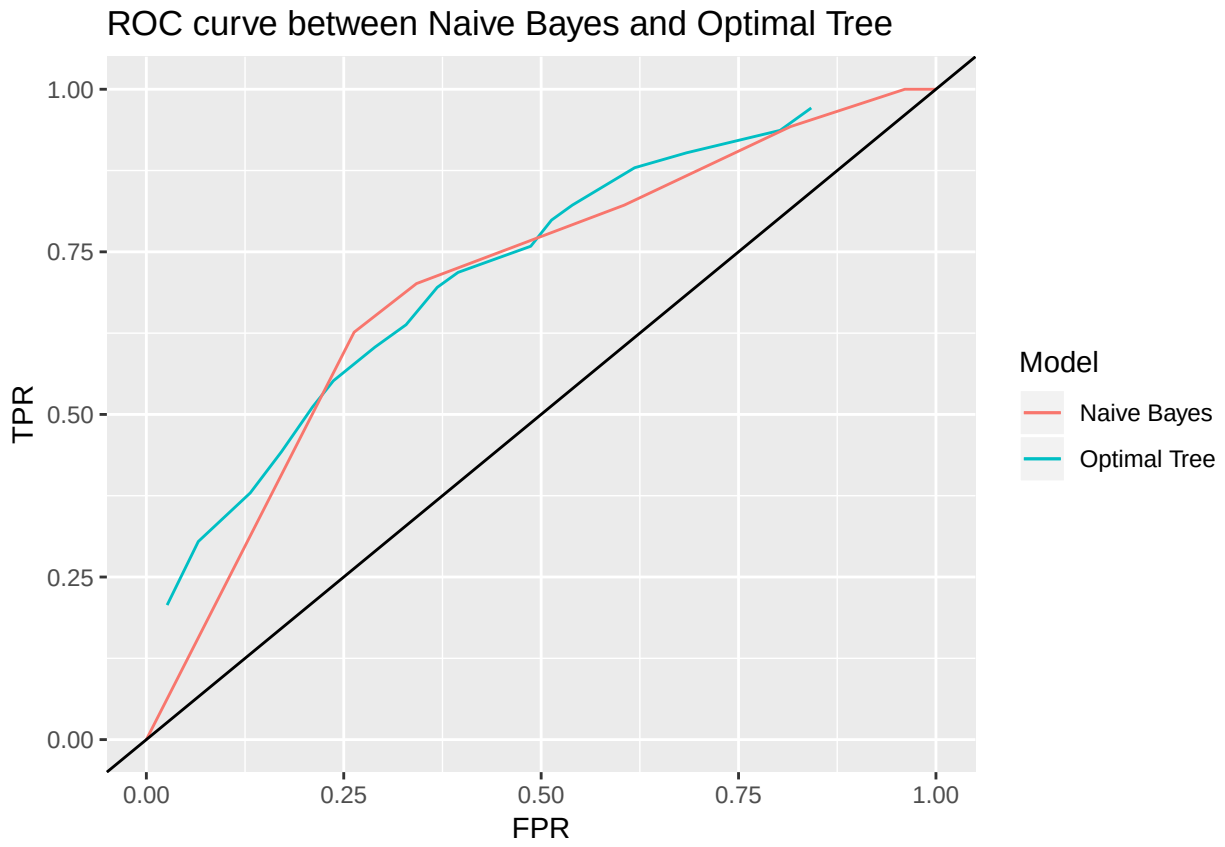
```
## [1] "Performance on Test -"
```

```
## Misclassification Rate :  0.684
```

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

Naive bayes model has much lower accuracy compared to the Decision Tree model using Deviance impurity measure. But this model has less difference in its preformance between train and test set. The decision tree model seems to have overfit the train data, but it is not the case with naive bayes model.

5. Optimal tree and the Naïve Bayes model



We can see the naive bayes is a little better than optimal tree because the area uder ROC curve is more for Niave bayes than optimal tree.

6. Naive Bayes using Loss Matrix

```
## [1] "Naive Bayes Model - "
```

```
## [1] "Performance on Train -"
```

```
## Misclassification Rate :  0.454
```

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

```
## [1] "Performance on Test -"
```

```
## Misclassification Rate :  0.492
```

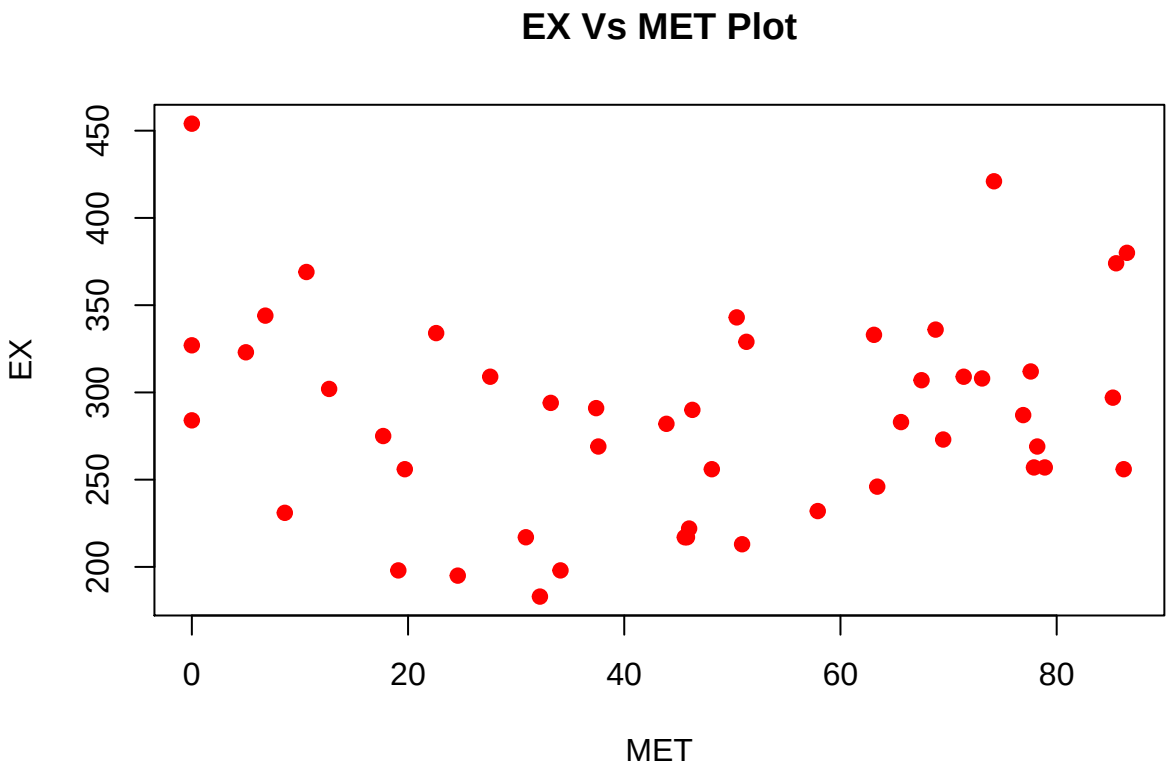
```
##
##        bad good
##   bad   71    5
##   good 122   52
```

Analysis + With loss matrix, train data has 90 and 137 true positive and true negative respectively. + With loss matrix, test data has 52 and 71 true positive and true negative respectively. + With this loss matrix where the probability that is customer if good is 10 times greater than the bad , it classify most of the people

as bad. Here the accuracy and missclassifictaion for both test and train is high and recall is pretty low. That means wit classy most of the people as bad which is in way good for a robust bank to ensure that there are less defaulter. + Previously the naive bayes has high accuracy and high recall for both test and train as compared to current model. But still this model i can say is better for the bank because it can help bank losse less but on the down side it might push away the potential customer. The tradeoff needs to be settled by bank.

Assignment 3. Uncertainty estimation

1. Plot EX versus MET using ordered dataset

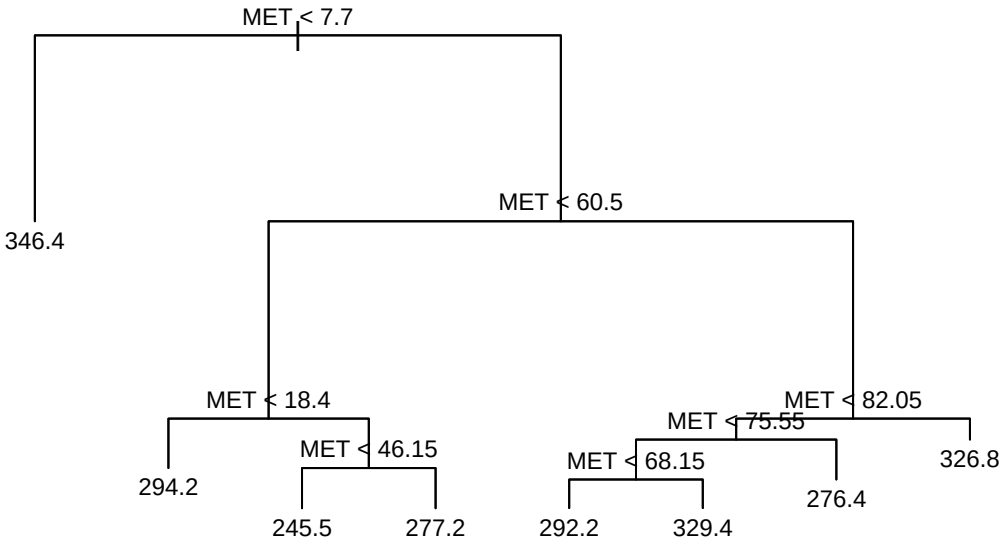


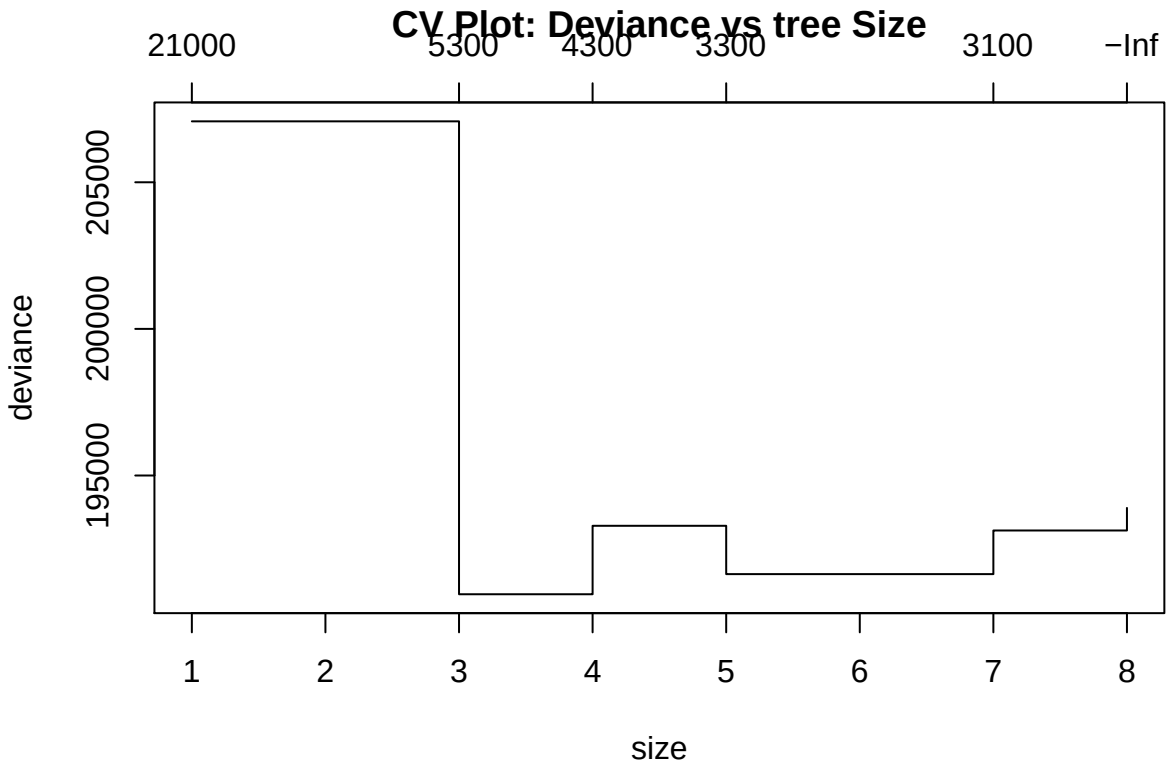
As we can see in the above plot, the data is not corelated. A straight line fit wont be a good idea in this scenario. A line similar to the cos function would be the best fit according to me, but any polynomial finction would be a good fit.

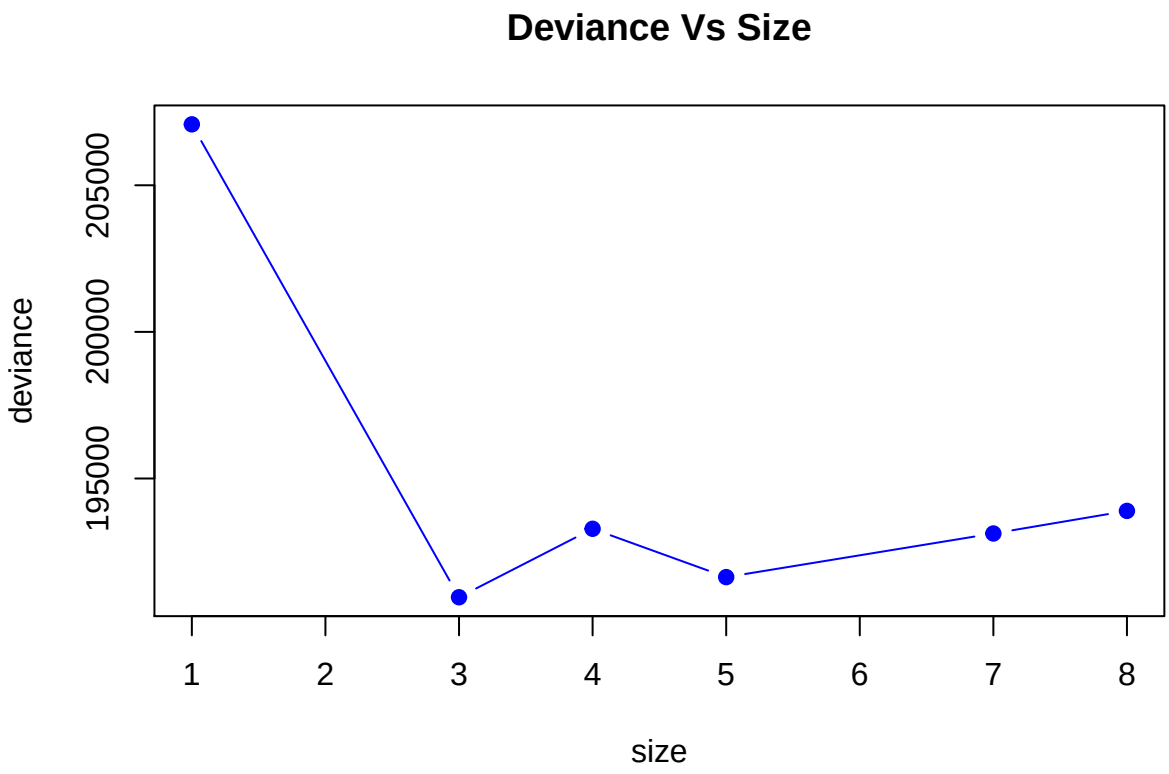
2. Fitting regression tree model using cross-validation method for leaves

a) Selection of tree using cross validation

```
##
## Fitted Tree:
```


MET < 7.7
346.4
MET < 60.5
MET < 18.4
294.2
MET < 46.15
245.5
277.2
MET < 82.05
MET < 75.55
MET < 68.15
292.2
329.4
276.4
326.8

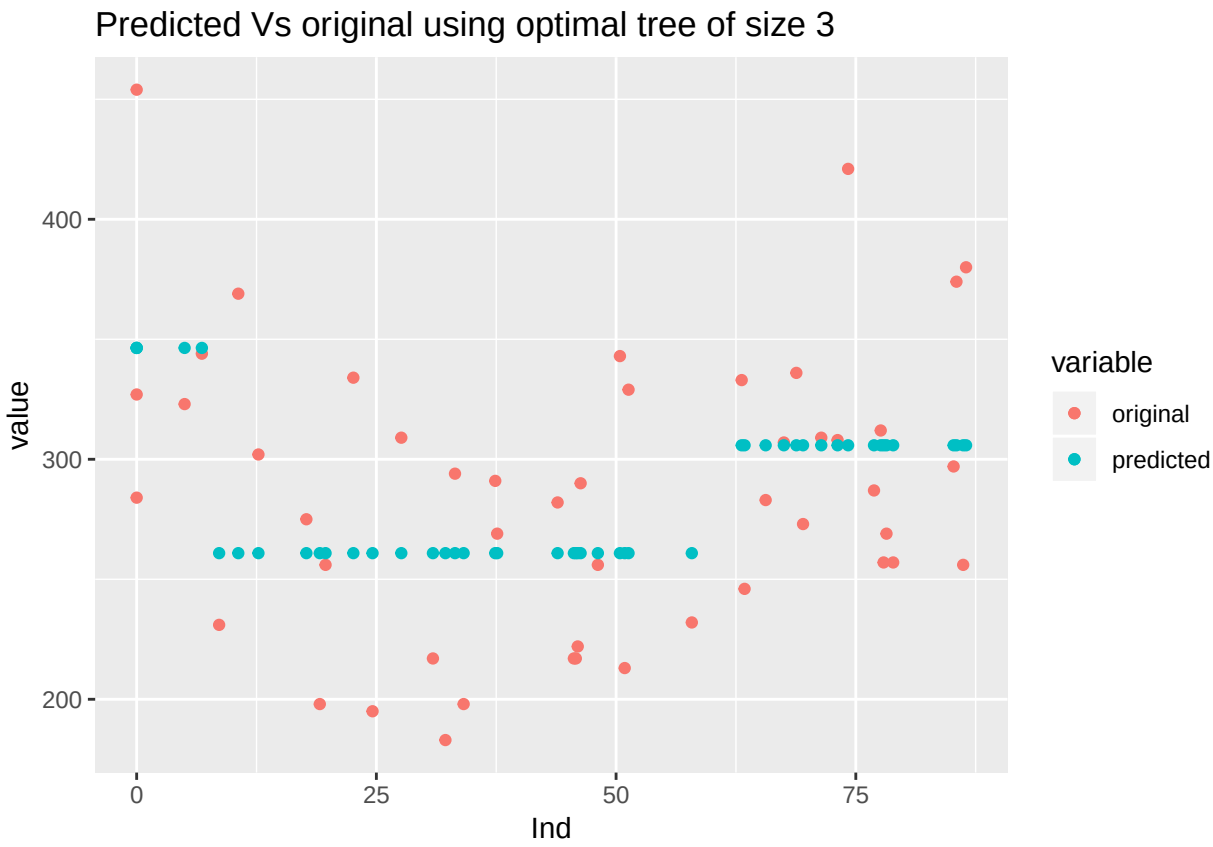

CV Plot: Deviance vs tree Size
21000
5300
4300
3300
3100
−Inf
deviance
195000
200000
205000
1
2
3
4
5
6
7
8
size



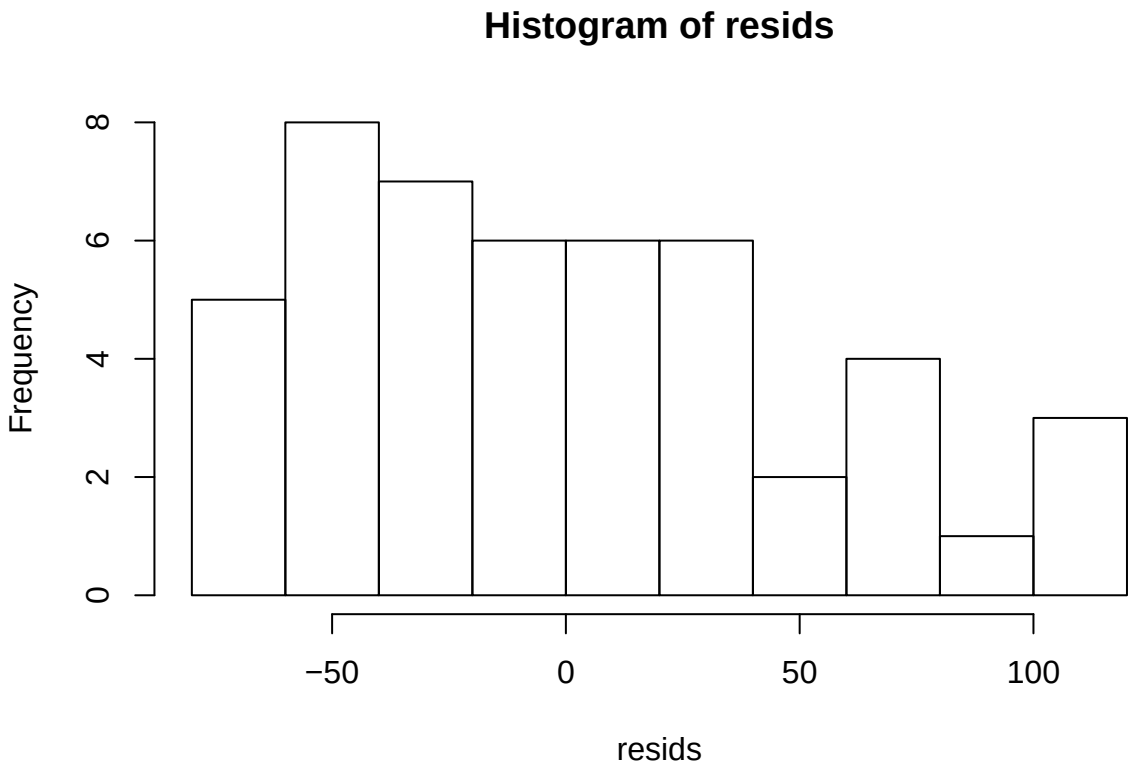

```
##
##  Optimal tree: 3
```

The CV plot of deviance vs size clearly shows that the least deviance(174057.6) is at 3, therefore best size is 3.

b) Plot of original and fitted data

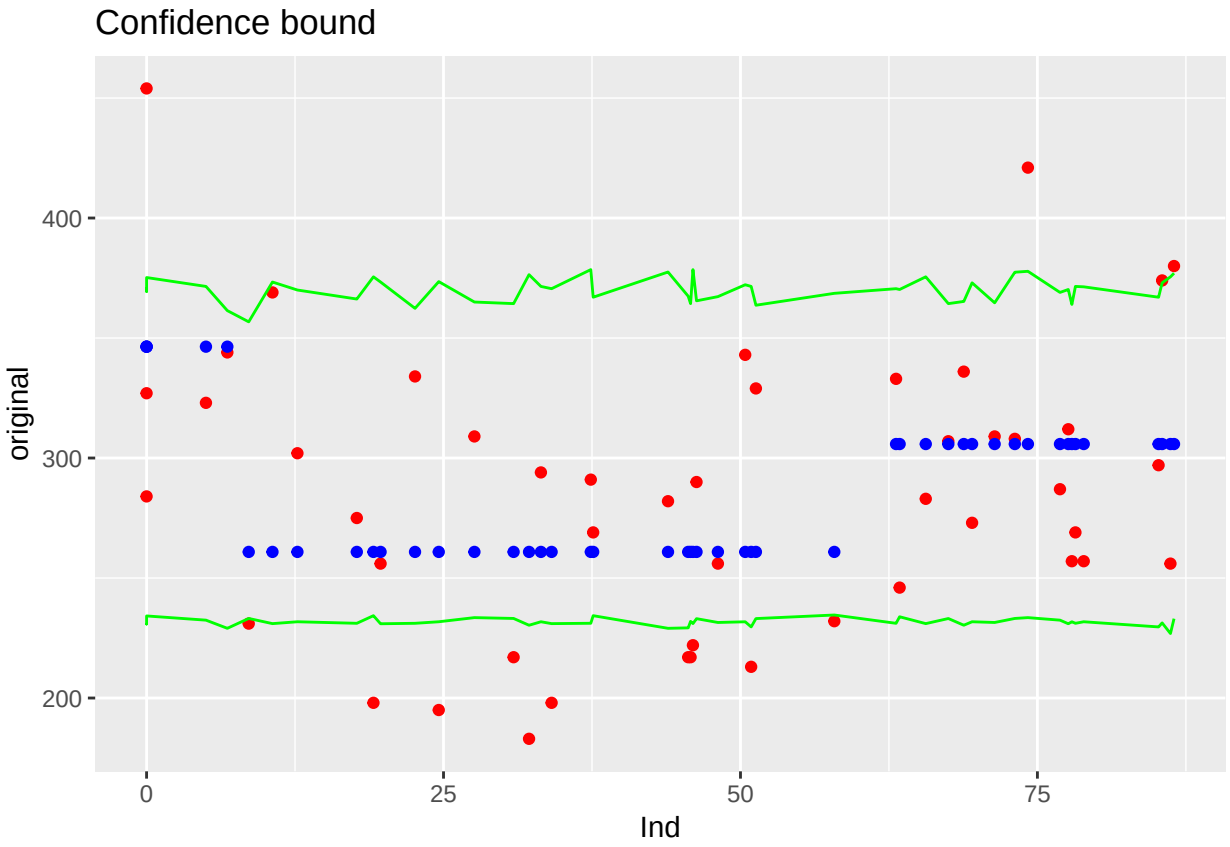


c) Histogramm of residuals



The distribution of the residuals is skewed towards left. Usually a model works best when the residuals are normally distributed. This is not a good fit to the data and it can be improved.

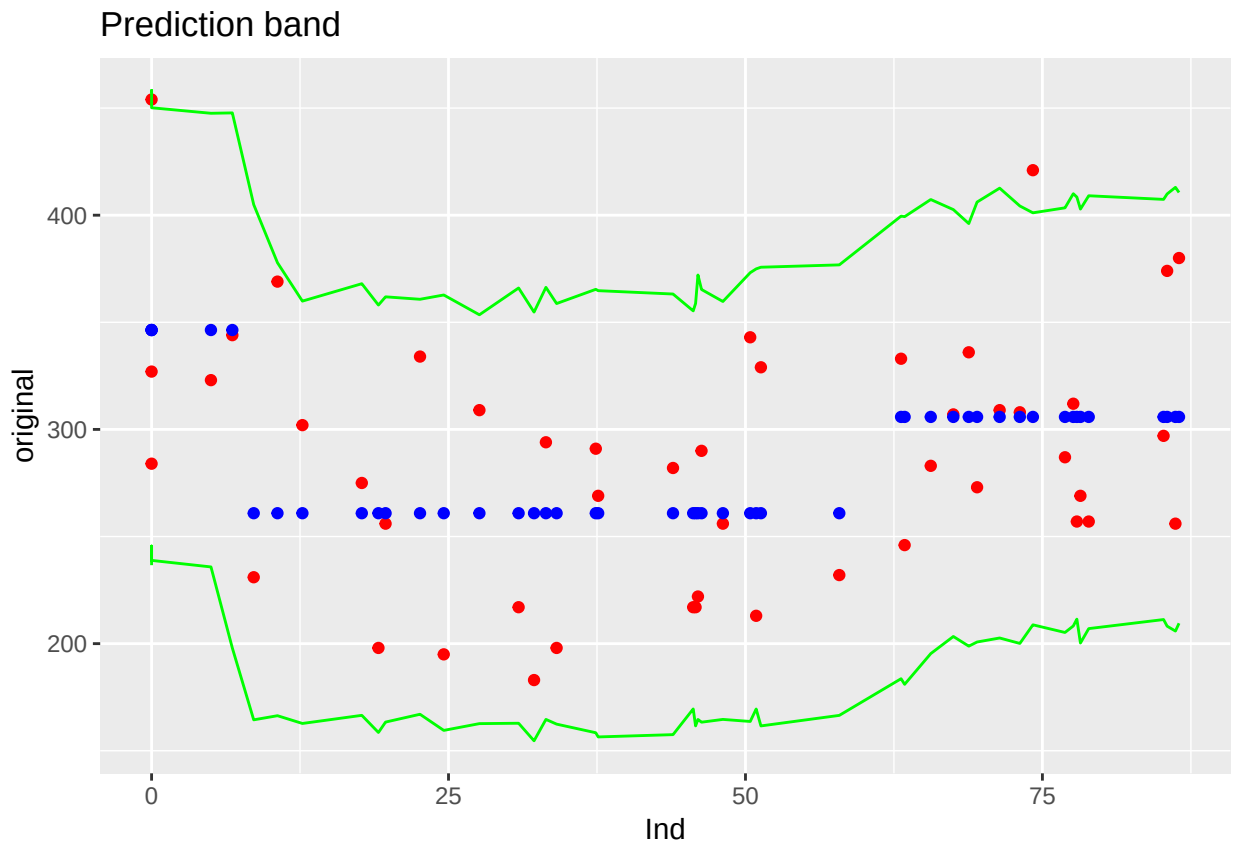
3. 95% confidence bands for the regression tree model by using non-parametric bootstrap



The confidence interval calculated using non-parametric bootstrap generalizes the data quiet well. The confidence inverval calculated in this case is very bumpy, it could be because of the way we are calculating it. This is a combination of intervals calculated for different bootstrap samples, so the bumpy interval may be because of that.

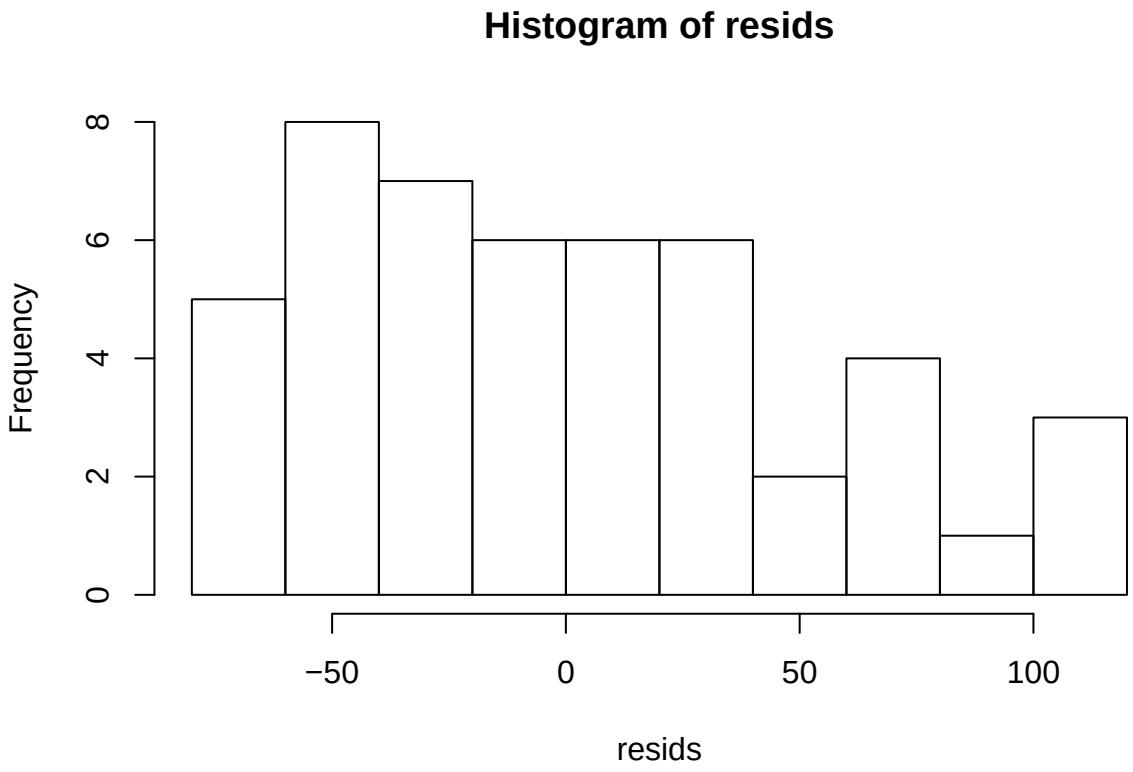
The confidence bands in this case are very close to the model we fit in step 2. Given that the distribution of the data is unknown this seems to be a good fit.

4. 95% confidence bands for the regression tree model by using parametric bootstrap



This is the plot of the confidence interval calculated using parametric bootstrap. This is a much better confidence interval though the cofidence band for is still bumpy. As the predctions we made in step 2 lie inside the two confidence intervals we got therefore the model in step 2 appears reliable.

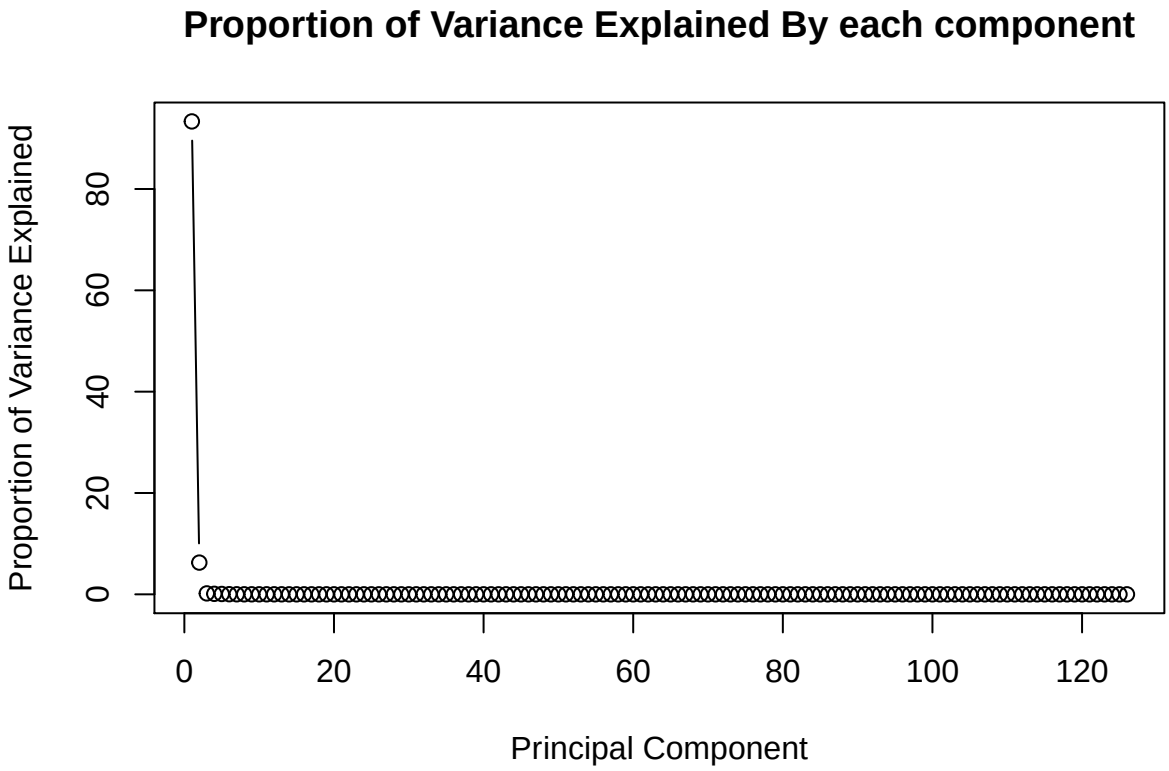
5. Analysis of histogram of residuals

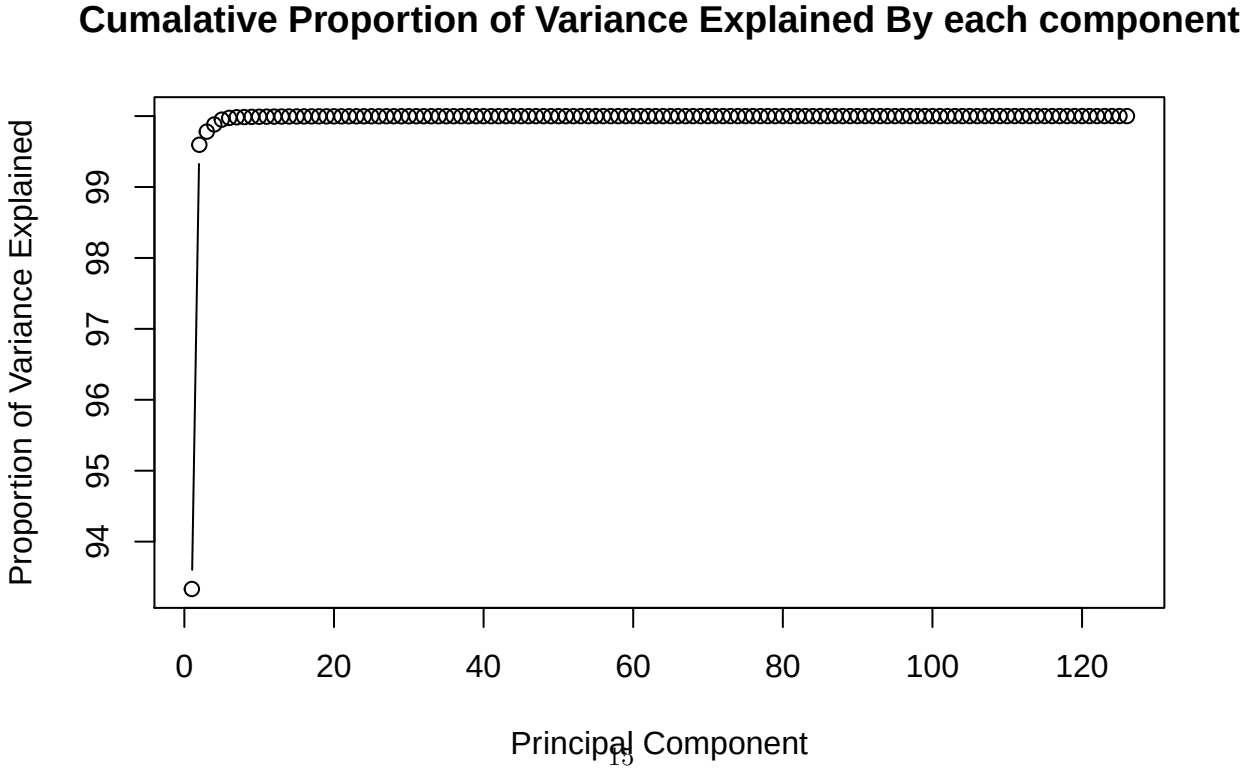


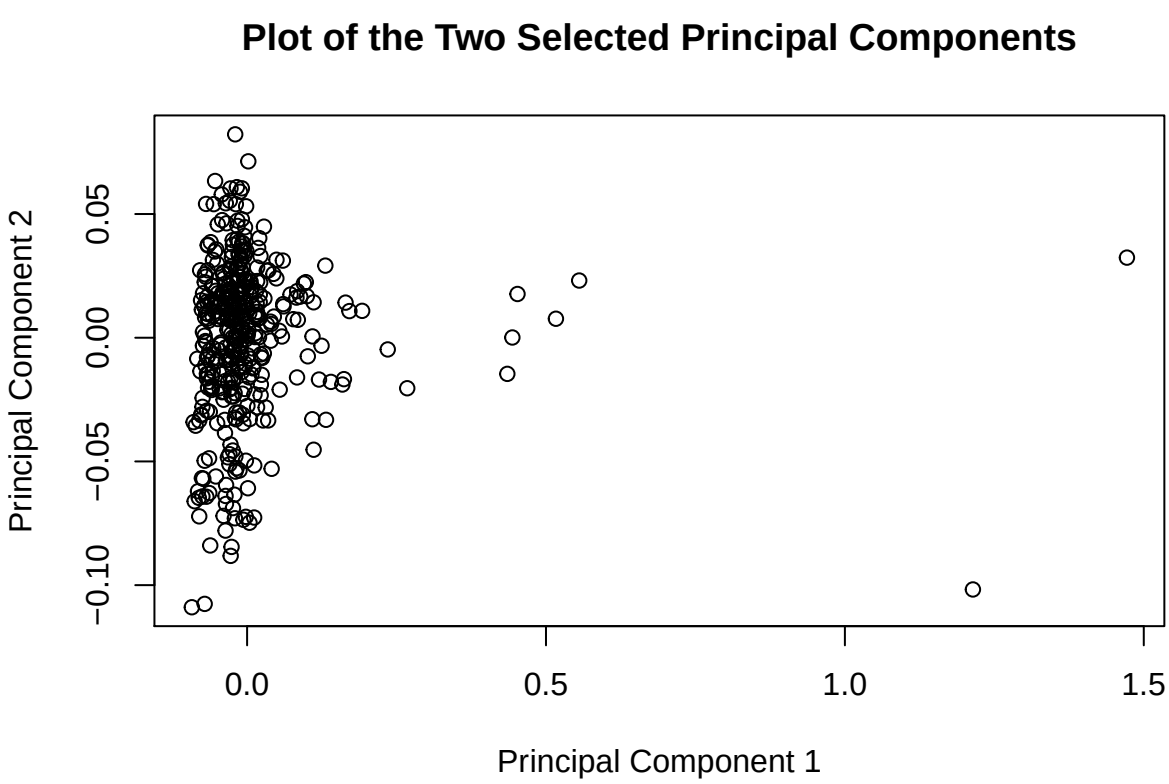
The histogram shows that parametric bootstrap is better than non-parametric bootstrap for this dataset. It fits the data very well and I think it would be good for data with skewed residuals.

Assignment 4. Principal Components

1 Principal Component Analysis



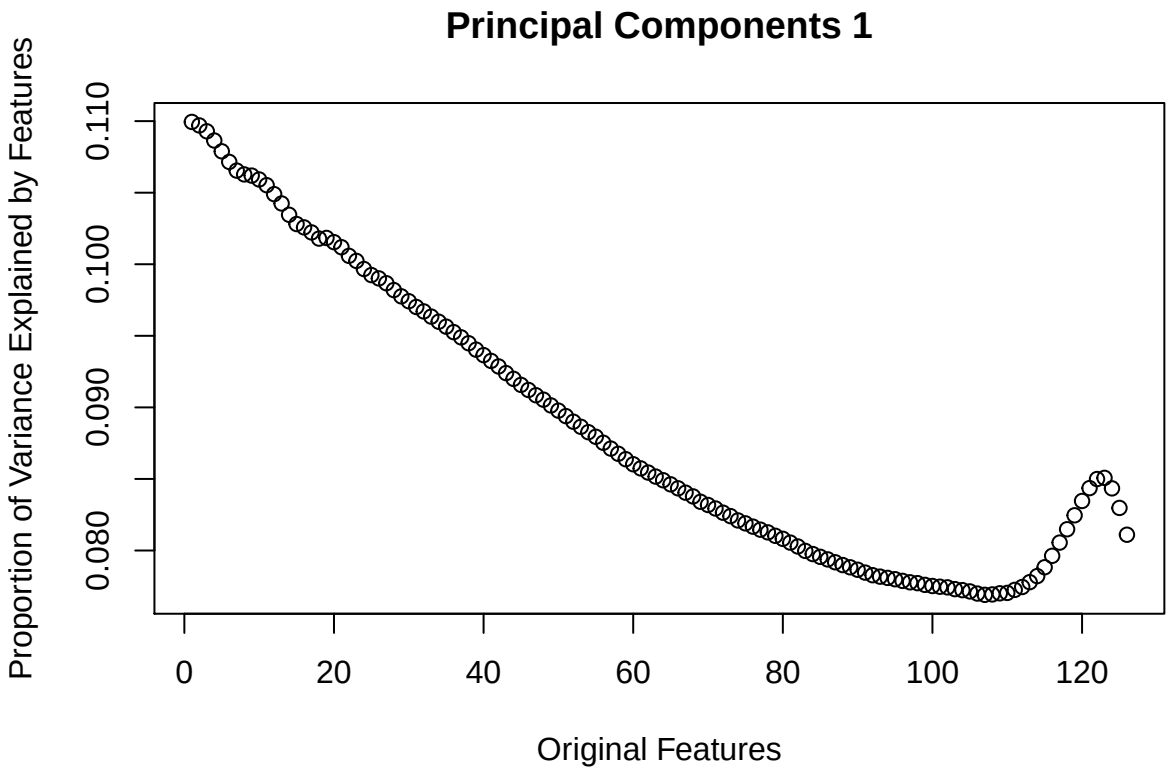


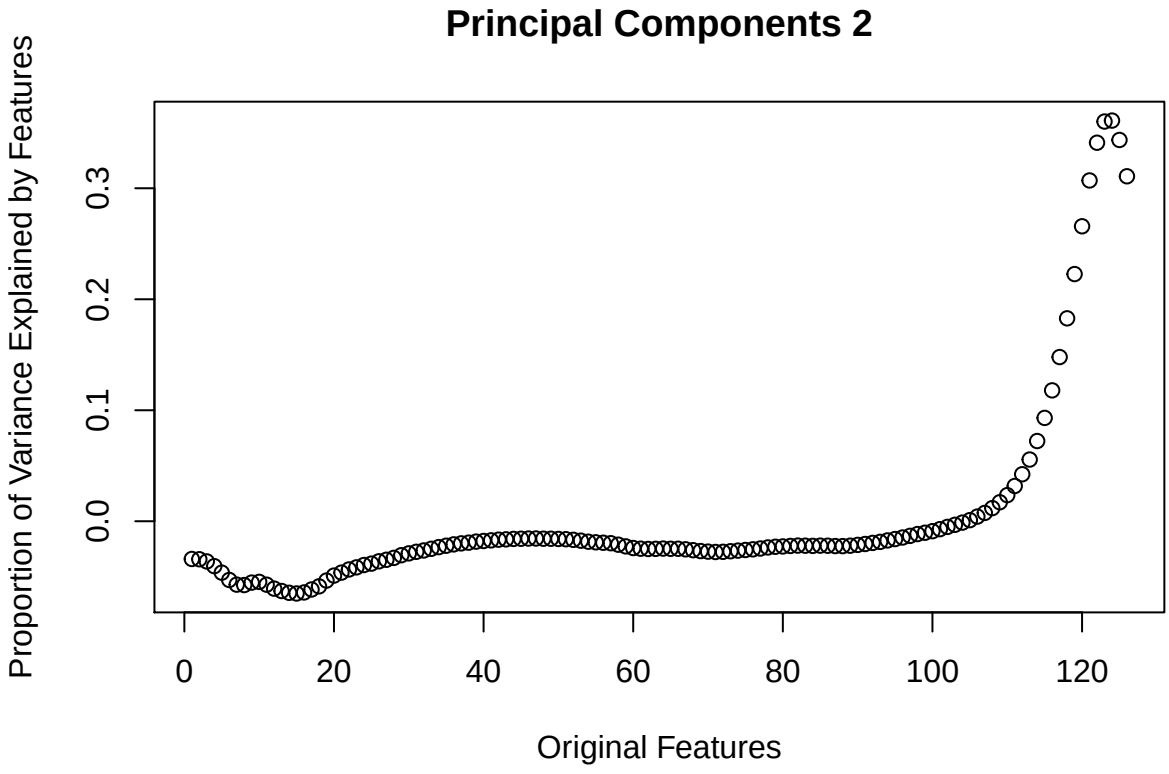


The first graph shows clearly that the first two principal components explain more than 99% of the total variance of the data, so we selected the first two principal components. We also made a cumalative percentage variance plot for the data and it showed the same thing, the first two components explain more than 99% of the total variance.

We than made a scatter plot of the two principal components. This plot had most of the components clustered together near the left side of the plot. There are two outliers near the right side of the plot both far apart from each other. These are the unusual diesal fuels according to this plot.

2 Principal Component Analysis Loadings

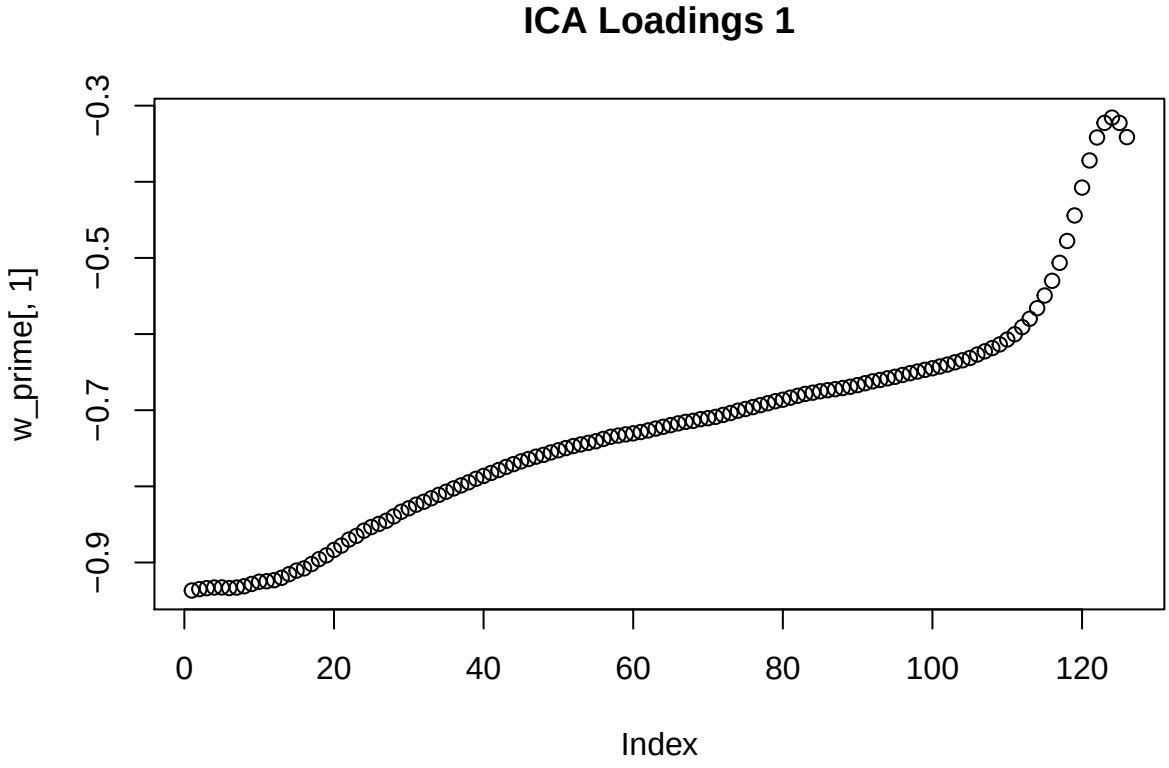


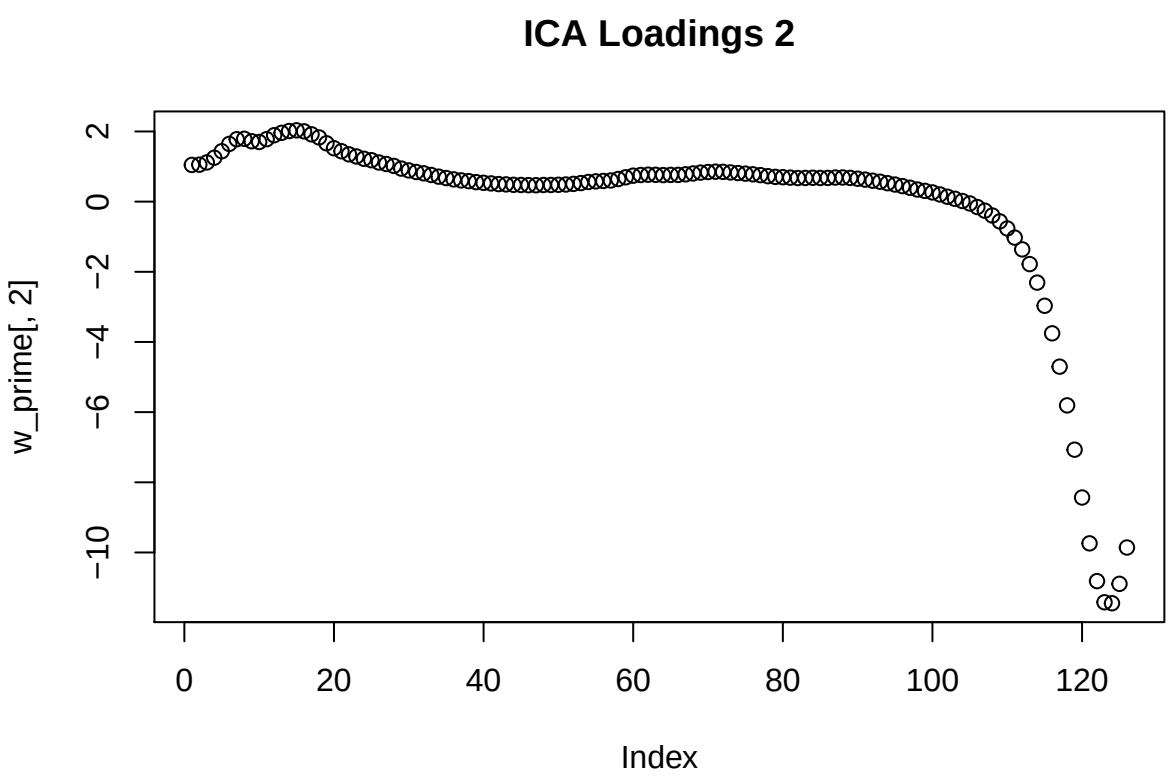


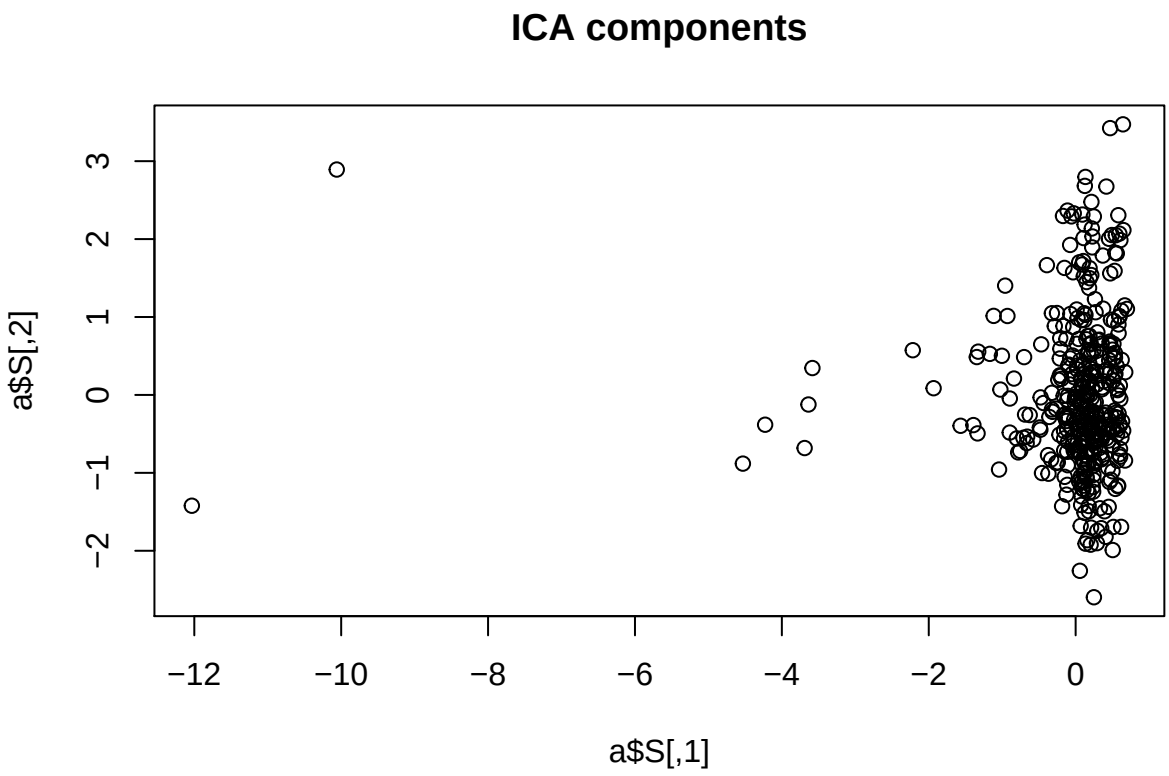
These are the trace plots showing how much contribution each of the original features make in the calculation of that principal component. The principal component 2(shown in the second plot) is explained by just a few of the original features. The features from 115 to 123 make majority of the contribution in explaining that feature.

3 Independent Component Analysis using fastICA

```
## Centering
## Whitening
## Symmetric FastICA using logcosh approx. to neg-entropy function
## Iteration 1 tol=0.019302
## Iteration 2 tol=0.013040
## Iteration 3 tol=0.002394
## Iteration 4 tol=0.000671
## Iteration 5 tol=0.000166
## Iteration 6 tol=0.000035
```




ICA Loadings 2
w_prime[, 2]
2
0
−2
−4
−6
−10
0
20
40
60
80
100
120
Index



Comparing the two trace plots with the ones in the previous step I think that each of the original features contribute more in calculation of the Independent components. The principal components are switched, like-

- PC1 (using PCA) is similar to PC2 (using ICA), and
- PC2 (unig PCA) is similar to PC1 (using ICA)

They have similar trace plots but the contribution of each of the original component is higher when using Independent Component Analysis.

The last plot is the plot of selected components using ICA. This looks like a mirror image to the one we had in the step 1, as the principal components are switched the plot is also mirrored along the Y-axis. The plot is exactly similar with the same outliers, just mirrored along the Y-axis.

We could say that ICA is negative of PCA. As the principal components calculated in ICA are negatively correlated with the original features of the data. This is the reason the plot of the two components is mirrored along the Y-axis.

Appendix

```
knitr::opts_chunk$set(echo = TRUE)
library(fastICA)
library(ggplot2)
library(tree)
library(readxl)
library(e1071)
##1
```

```
cds = read_xls("creditscoring.xls")
#colSums(is.na(cds))
cds$purpose[which(is.na(cds$purpose))] = 0
cds$good_bad = as.factor(cds$good_bad)
#split data into train test
n=dim(cds)[1]
set.seed(12345)
id=sample(1:n, floor(n*0.5))
train=cds[id,]
id1=setdiff(1:n, id)
set.seed(12345)
id2=sample(id1, floor(n*0.25))
valid=cds[id2,]
id3=setdiff(id1,id2)
test=cds[id3,]
##tree deviance
print("Deviance Impurity Measure - ")
print("Performance on Train -")
dtm2_dev = tree(good_bad~., train, split = "deviance")
p1 = predict(dtm2_dev, train, type = "class")
cat("Misclassification Rate : ", sum(p1==train$good_bad)/(nrow(train)), "\n")
table(train$good_bad, p1)
print("Performance on Test -")
p2 = predict(dtm2_dev, test, type = "class")
cat("Misclassification Rate : ", sum(p2==test$good_bad)/(nrow(test)), "\n")
table(test$good_bad, p2)
##tree gini
print("Gini Impurity Measure - ")
print("Performance on Train -")
dtm2_gin = tree(good_bad~., train, split = "gini")
p1 = predict(dtm2_gin, train, type = "class")
cat("Misclassification Rate : ", sum(p1==train$good_bad)/(nrow(train)), "\n")
table(train$good_bad, p1)
print("Performance on test -")
p2 = predict(dtm2_gin, valid, type = "class")
cat("Misclassification Rate : ", sum(p2==test$good_bad)/(nrow(test)), "\n")
table(test$good_bad, p2)
##depth calc
pt_train <- prune.tree(dtm2_dev)
pt_valid <- prune.tree(dtm2_dev,newdata = valid)

{plot(pt_train$size, pt_train$dev, type="b", col="green",ylim=c(260,630),
      xlab="Leaf Nodes",ylab="Deviances")
  points(x=pt_valid$size,y=pt_valid$dev, type="b", col="red")
  title(main="Training vs Validation deviances on pruned trees")
  legend("topright",legend=c("Train Data Set", "Validation Data Set"),
         col=c("green","red"),lty=1:2, cex=0.8,title="Deviances")}

#Selecting the number of leaf node from the valid set for which the deviance error is least.
best_lf <-  pt_valid$size[which.min(pt_valid$dev)]

#Optinmal Tree based on leat deviance
pt_opt  <- prune.tree(dtm2_dev,best=best_lf)
```

```
nodes_train_opt <- as.numeric(rownames(pt_opt$frame))  #getnodes
depth_train <-  max(tree:::tree.depth(nodes_train_opt)) #get depth

{plot(pt_opt)
  text(pt_opt)
  title(main="Optimal Tree for Training Data")}

cat("Depth of Optimal Tree for training data:",depth_train,"\n")
cat("Number of Leaf Nodes in Optimal Tree for training data:",best_lf,"\n")

cat("\nLabels used by training data:\n")
tree:::labels.tree(pt_opt)
model = naiveBayes(good_bad~., train)
#train
print("Naive Bayes Model - ")
print("Performance on Train -")
pred = predict(model, train[,-ncol(train)])
cat("Misclassification Rate : ", sum(pred==train$good_bad)/(nrow(train)), "\n")
table(train$good_bad, pred)
#test
print("Performance on Test -")
pred = predict(model, test[,-ncol(test)])
cat("Misclassification Rate : ", sum(pred==test$good_bad)/(nrow(test)), "\n")
table(test$good_bad, pred)
#Naive Bayes
pies = seq(0.05, 0.95, 0.05)
results_nb = matrix(nrow = 0, ncol = 3)
results_ot = matrix(nrow = 0, ncol = 3)
for(pi in pies){
  pred = as.data.frame(predict(model, test[, -ncol(test)], type = "raw"))
  pred$res = ifelse(pred$good>pi, "good", "bad")
  misCl = sum(pred$res == test$good_bad)/(nrow(pred))
  m = (test$good_bad == "good")*1
  n = (pred$res == "good")*1
  tp = sum(m*n)
  fp = abs(sum(n)-tp)/sum(abs(m-1))
  tp = tp/(sum(m))
  results_nb = rbind(results_nb, c(misCl, tp, fp))

  pred = as.data.frame(predict(pt_opt, test[, -ncol(test)]))
  pred$res = ifelse(pred$good>pi, "good", "bad")
  misCl = sum(pred$res == test$good_bad)/(nrow(pred))
  m = (test$good_bad == "good")*1
  n = (pred$res == "good")*1
  tp = sum(m*n)
  fp = (abs(sum(n)-tp))/sum(abs(m-1))
  tp = tp/(sum(m))
  results_ot = rbind(results_ot, c(misCl, tp, fp))
}

results_nb = as.data.frame(results_nb)
colnames(results_nb) = c("MiscRate", "TP", "FP")
results_ot = as.data.frame(results_ot)
```

```r
colnames(results_ot) = c("MiscRate", "TP", "FP")

ggplot() + geom_line(data=results_nb,aes(x=FP,y=TP,color="red")) +
  geom_line(data=results_ot,aes(x=FP,y=TP,color="blue"))+ scale_color_discrete(name="Model",labels=c("Na
  geom_abline(intercept=0,slope=1)+
  xlab("FPR")+ylab("TPR")+ggtitle("ROC curve between Naive Bayes and Optimal Tree")
model = naiveBayes(good_bad~., train)
#train
print("Naive Bayes Model - ")
print("Performance on Train -")
pred = as.data.frame(predict(model, train[,-ncol(train)], type = "raw"))
pred$res = ifelse((pred$good)>(pred$bad)*10, "good", "bad")
cat("Misclassification Rate : ", sum(pred$res==train$good_bad)/(nrow(train)), "\n")
table(train$good_bad, pred$res)
#test
print("Performance on Test -")
pred = as.data.frame(predict(model, test[,-ncol(test)], type = "raw"))
pred$res = ifelse((pred$good)>(pred$bad)*10, "good", "bad")
cat("Misclassification Rate : ", sum(pred$res==test$good_bad)/(nrow(test)), "\n")
table(test$good_bad, pred$res)
set.seed(12345)

#functions
plotTree <- function(tree){
  plot(tree,main="Fitted tree")
  text(tree, cex=.75)
}

data <- read.table("State.csv",sep=";",header = TRUE)
colsNeeded <- c("EX","MET")
data <- data[colsNeeded]
data$MET <- gsub(',', '.', data$MET)
data$MET <- as.numeric(data$MET)

# reorder
data = data[order(data$MET),]

#plot
plot(EX ~ MET, data = data, pch = 19, cex = 1,col="red", main="EX Vs MET Plot")
#part 2
# fitting tree
model <- tree(formula = EX ~ MET,data = data,control =tree.control(nobs = nrow(data),minsize = 8))

cat("\nFitted Tree:")
plotTree(model)

cvModel <- cv.tree(model)
plot(cvModel,main="CV Plot: Deviance vs tree Size")
# Plotting cv tree
plot(cvModel$size, cvModel$dev, main = "Deviance Vs Size" ,
     xlab="size", ylab = "deviance", type="b",col="blue", pch= 19,cex=1)

# which size is better?
```

```
bestSize <- cvModel$size[which(cvModel$dev==min(cvModel$dev))]
cat("\n Optimal tree:",bestSize)

bestTree <- prune.tree(model,best = bestSize)

# predictions
preds = predict(bestTree,newdata=data)

# plot original and fitted data
results  <- data.frame(Ind=data$MET,original=data$EX,predicted=preds)

ggplot(results, aes(Ind, y = value, color = variable)) +
  geom_point(aes(y = original, col = "original")) +
  geom_point(aes(y = predicted, col = "predicted"))+
  ggtitle("Predicted Vs original using optimal tree of size 3")
#hsitogram
resids<- (data$EX - preds)
hist(resids)
#part 3
library(boot)
set.seed(12345)
bootstrap <- function(data,i){
  data  <- data[i,]

  model <- tree(EX ~ MET, data = data, control = tree.control(nobs = nrow(data),minsize = 8))
  bestTree <- prune.tree(model,best = bestSize)
  preds <- predict(bestTree,newdata=data)
  return(preds)
}
bootResults <- boot(data=data,statistic = bootstrap,R=1000)

confBound <- envelope(bootResults,level = 0.95)

upperLimits <- confBound$point[1,]
lowerLimits <- confBound$point[2,]

results["upper"] = upperLimits
results["lower"] = lowerLimits

ggplot(results, aes(Ind,original,predicted,upper,lower))+
  geom_point(aes(Ind,original),color="red")+
  geom_point(aes(Ind,predicted),color="blue")+
  geom_line(aes(Ind,upper),color="green")+
  geom_line(aes(Ind,lower),color="green")+
  ggtitle("Confidence bound")
# part4, parammetric bootstraping

bootStrapParam <- function(data,index){
  data <- data[index,]
  model <- tree(EX ~ MET, data = data, control = tree.control(nobs = nrow(data),minsize = 8))
  bestTree <- prune.tree(model,best = bestSize)
  preds <- predict(bestTree,newdata=data)
  resids <- data$EX - preds
```

```
  # each prediction is an estimation and can be used as mean,

  stDev <- sd(resids)
  preds<- rnorm(nrow(data),preds,stDev)
  return(preds)
}
ranGenFunc <- function(data,model){
  data$EX = rnorm(nrow(data), predict(model,newdata=data),sd(resid(model)))
  return(data)
}
bootResults <- boot(data,statistic = bootStrapParam , R=1000, mle=bestTree,sim="parametric",ran.gen = ra
confBound <- envelope(bootResults,level = 0.95)

upperLimits <- confBound$point[1,]
lowerLimits <- confBound$point[2,]

results["upperP"] = upperLimits
results["lowerP"] = lowerLimits
ggplot(results, aes(Ind,original,predicted,upperP,lowerP))+
  geom_point(aes(Ind,original),color="red")+
  geom_point(aes(Ind,predicted),color="blue")+
  geom_line(aes(Ind,upperP),color="green")+
  geom_line(aes(Ind,lowerP),color="green")+
  ggtitle("Prediction band")
hist(resids)
NIR_data = read.csv("NIRSpectra.csv", header = TRUE, sep = ';', dec = ',')
train = NIR_data[,-ncol(NIR_data)]
train_true = NIR_data[,ncol(NIR_data)]
NIR_pca = prcomp(train, center = TRUE, scale. = FALSE)
a = summary(NIR_pca)
df<- t(as.data.frame(a$importance))

q = sum(NIR_pca$sdev^2)
q1 = (NIR_pca$sdev^2/q) * 100

plot(q1, xlab = "Principal Component",
     ylab = "Proportion of Variance Explained",
     main = "Proportion of Variance Explained By each component",
     type = "b")
plot(cumsum(q1), xlab = "Principal Component",
     ylab = "Proportion of Variance Explained",
     main = "Cumalative Proportion of Variance Explained By each component",
     type = "b")

p = 0.9978
selected_pca = NIR_pca$x[,1:nrow(df[which(df[,"Cumulative Proportion"]<p),])]
plot(NIR_pca$x[,1], NIR_pca$x[,2], xlab = "Principal Component 1",
     ylab = "Principal Component 2",
     main = "Plot of the Two Selected Principal Components")
plot(NIR_pca$rotation[,1], xlab = "Original Features",
     ylab = "Proportion of Variance Explained by Features",
     main = "Principal Components 1")
plot(NIR_pca$rotation[,2], xlab = "Original Features",
```

```
     ylab = "Proportion of Variance Explained by Features",
     main = "Principal Components 2")
set.seed(12345)
a <- fastICA(train, 2, alg.typ = "parallel", fun = "logcosh", alpha = 1,
             method = "C", row.norm = FALSE, maxit = 200,
             tol = 0.0001, verbose = TRUE)
#par(mfrow = c(1, 3))
#plot(a$X, main = "Pre-processed data")
#plot(a$X %*% a$K, main = "PCA components")

w_prime=a$K %*% a$W

plot(w_prime[,1], main = "ICA Loadings 1")
plot(w_prime[,2], main = "ICA Loadings 2")

plot(a$S, main = "ICA components")
```