Group_A11_Lab2

Obaid,Sridhar,Naveen

10 December 2018

Contents

Assignment 2 :Analysis of credit scoring

1. Importing and Pre-processing credit

The data is read and divided into training,test and validation dataset in the proportion of 50/25/25 respectively.

2. Decision Tree

1. Deviance as Split measure

- **Train Data**
  - This model has missclassification rate of 22.8% which means 22.8% of the data were predicted wrongly.
  - The model has accuracy of 77.2% which means 77.2% of the data were predicted accurately.
  - The model has 321 and 60 true positive and true negative respectively.
  - The precision is 76.9% which means 76.9% of the predicted value for good were right.
  - The recall is 94.9% which means 94.5% of the actual good was properly predicted by the model.

```
## Confusion Matrix:
```

```
##         Predicted
## Expected bad good
##     bad   61   86
##     good  20  333
```

```
##
## Missclassification Rate: 0.212
```

- **Test Data**
  - This model has missclassification rate of 30% which means 30% of the data were predicted wrongly.
  - The model has accuracy of 70% which means 70% of the data were predicted accurately.
  - The model has 149 and 23 true positive and true negative respectively.
  - The precision is 74.87% which means 74.87% of the predicted value for good were right.
  - The recall is 85.6% which means 85.6% of the actual good was properly predicted by the model.

```
## Confusion Matrix:
```

```
##         Predicted
## Expected bad good
##     bad   28   48
##     good  19  155
```

```
##
## Missclassification Rate: 0.268
```

2. Gini as split measure

- **Train Data**
  - This model has missclassification rate of 21.25% which means 21.25% of the data were predicted wrongly.
  - The model has accuracy of 78.74% which means 78.74% of the data were predicted accurately.
  - The model has 305 and 84 true positive and true negative respectively.
  - The precision is 80.9% which means 80.9% of the predicted value for good were right.
  - The recall is 90.2% which means 90.2% of the actual good was properly predicted by the model.

```
## Confusion Matrix:
```

```
##         Predicted
## Expected bad good
##     bad   66   81
##     good  38  315
```

```
##
## Missclassification Rate: 0.238
```

- **Test Data**
  - This model has missclassification rate of 29.55% which means 29.55% of the data were predicted wrongly.
  - The model has accuracy of 70.44% which means 70.44% of the data were predicted accurately.
  - The model has 152 and 22 true positive and true negative respectively.
  - The precision is 74.87% which means 74.87% of the predicted value for good were right.
  - The recall is 87.3% which means 87.3% of the actual good was properly predicted by the model.

```
## Confusion Matrix:
```

```
##         Predicted
## Expected bad good
##     bad   18   58
##     good  35  139
```

```
##
## Missclassification Rate: 0.372
```

Below table gives the comparison for deviance and gini by different metrics. Deviance as a measure of impurity gives better decision tree than gini. So for below steps, I will use deviance as a measure for splitting the trees

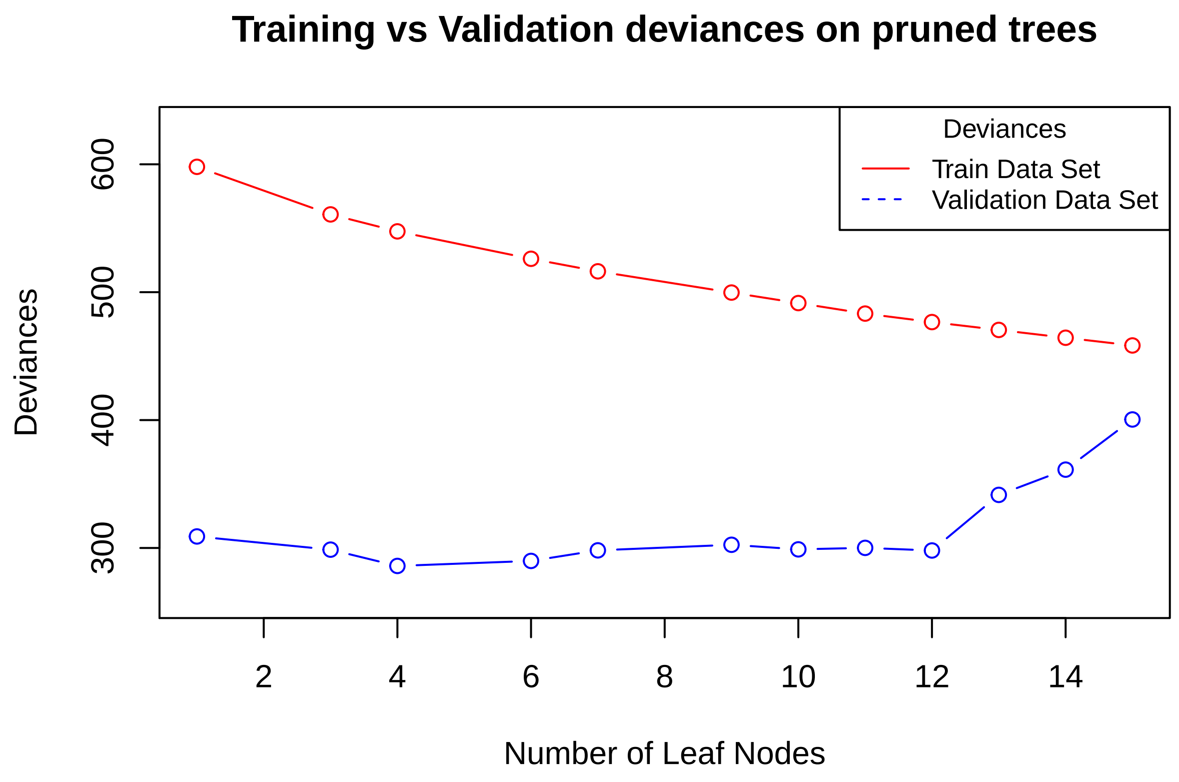
```
##                Precision    Recall Accuracy
## Deviance_Train 0.7947494 0.9433428    0.788
## Deviance_Test  0.7635468 0.8908046    0.732
## Gini_Train     0.7954545 0.8923513    0.762
## Gini_Test      0.7055838 0.7988506    0.628
```

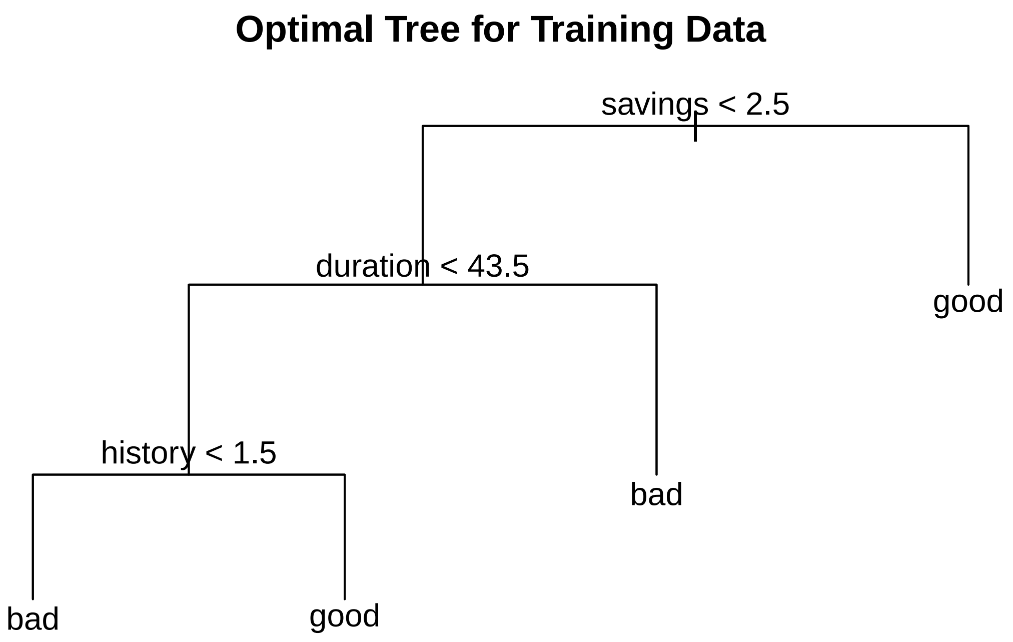
3. Optimal Tree Depth

The training error decreases with increase in leaf node because it decreases misclassification but perform poorly on vali data set. According to the plot, I chose optimal tree with leaf node as 4 because at 4 the valid deviance is the minimum and thereafter the error steadily increases.

The variable used by train data to create splits is **savings**, **duration** and **history** .

The first variable used by the model is split which classify good if saving>2.5. It seems the first criteria used by model is to identify people who has minimum saving with themself. If they do have minimum savings then it might mean that the person has good monetary income to repay the loan. If the saving<2.5 then the classfication is based on the time period where he has not paid the interest/principle for loan. If thepart of the loan is not paid even after 43.5(months/weeks) then the person is serious defaulter. Given the savings < 2.5 and duration < 43.5 , the next criteria used is history. May be this is to identify the past history of the person. Here history may be a past history score. Even if the savings of the person is not great but the history score is better then that means the person is most likely to pay up the loan.


Training vs Validation deviances on pruned trees
Deviances
Train Data Set
Validation Data Set
Deviances
300
400
500
600
2
4
6
8
10
12
14
Number of Leaf Nodes



```
## Depth of Optimal Tree for training data: 3
```

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

```
## [1] "root"            "savings < 2.5"   "duration < 43.5" "history < 1.5"
## [5] "history > 1.5"   "duration > 43.5" "savings > 2.5"
```

- **Prediction on Test Data Using Optimal Tree**
  - This model has missclassification rate of 25.6% which means 25.6% of the data were predicted wrongly.
  - The model has accuracy of 74.4% which means 74.4% of the data were predicted accurately.
  - The model has 168 and 18 true positive and true negative respectively.
  - The precision is 74.33% which means 74.33% of the predicted value for 'good' class were right.
  - The recall is 96.5% which means 96.5% of the actual 'good' class was properly predicted by the model.
  - Here is precision is low and the recall is pretty high. The metric is undesirable for the financial institution. The recall being high, the bank will classfiy most of the people as good. In classfying most of the people as good for loan ,it might include most of the defaulters automatically.

```
##
##
## Confusion Matrix:
```

```
##         Predicted
## Expected bad good
##      bad  18   58
##     good   6  168
```

Table 1: Comparison of Naive bayes vs Optimla Tree

	Missclass	Precision	Recall	Accuracy
Train_optimal_tree	0.252	0.7494505	0.9660057	0.748
Test_optimal_tree	0.256	0.7433628	0.9655172	0.744
Train_by_naive	0.300	0.8306189	0.7223796	0.700
Test_by_naive	0.316	0.8064516	0.7183908	0.684

```
##
## Missclassification Rate: 0.256
```

4. Naive Baiyes and comparison with optimal tree

- **Analysis**
  - Under naive bayes, train data has 255 and 95 true positive and true negative respectively.
  - Under naive bayes, test data has 125 and 46 true positive and true negative respectively.
  - For naive bayes, missclassifaction of test is more than train so the accuracy is less for test than train. Moreover the precision and recall is more for train than test. One of the probable reason why naive bayes does poorly on test data is because it considers the variables as indpendent which might not be true in the case of bank where several factors are dependent on each other to classify a person as 'good' or 'bad' for the loan.
  - Based on the metrics shown in table,optimal tree can be better for the bank because its recall for "good" class is lesser than naive bayes which means its tendency to classify a person as defaulter is more than optimal tree due to which the naive bayes has poor missclassfication than optimal tree. In my opinion naive bayes is better than optimal tree.

```
##
##
## Confusion Matrix for Train Data:
```

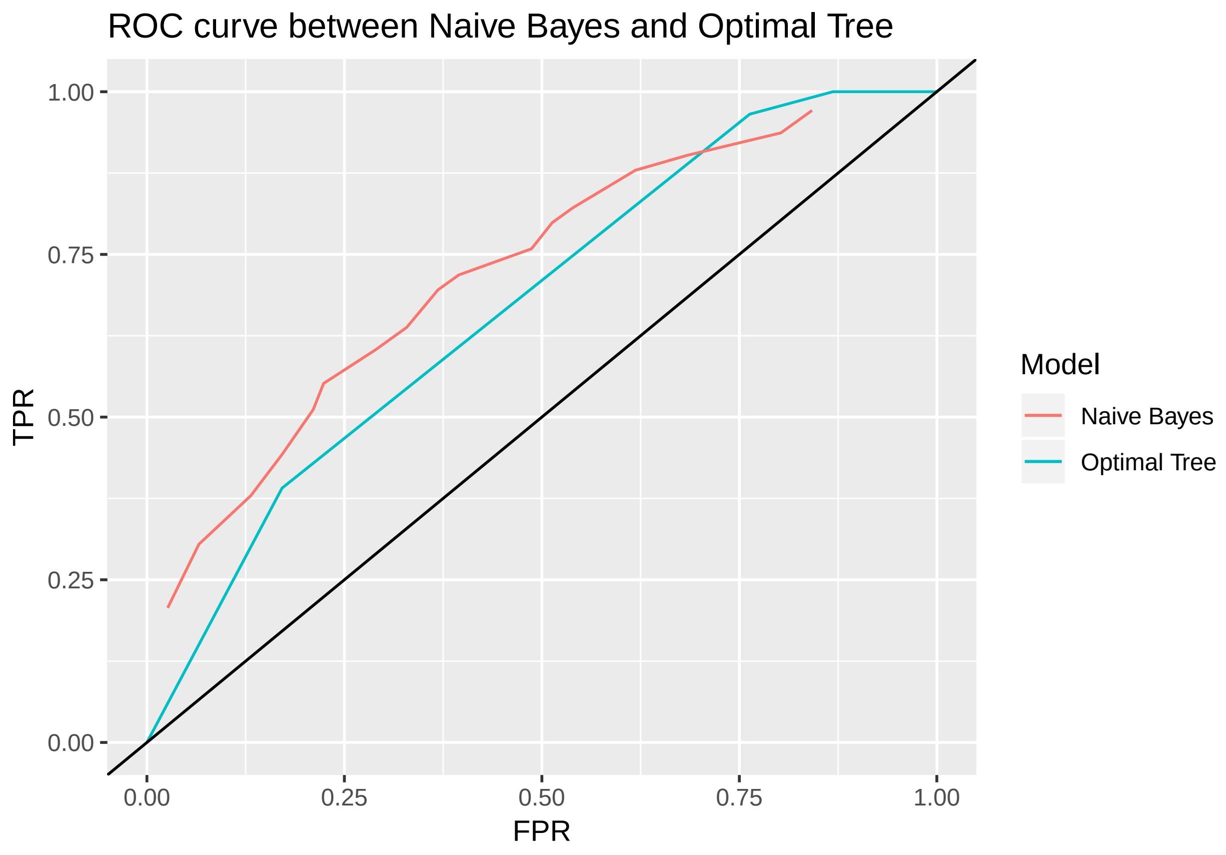
```
##         Predicted
## Expected bad good
##     bad   95   52
##     good  98  255
```

```
##
##
## Confusion Matrix for Test Data:
```

```
##         Predicted
## Expected bad good
##     bad   46   30
##     good  49  125
```

5. Optimal tree and the Na?ve Bayes model

Supporting the answer from question 4, we can see the naive bayes is better than optimal tree because the area uder ROC curve is more for Niave bayes than optimal tree.



6. Naive Bayes using Loss Matrix

- **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.

```
##
##
## Confusion Matrix for Train Data:
```

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

```
##
##
## Confusion Matrix for Test Data:
```

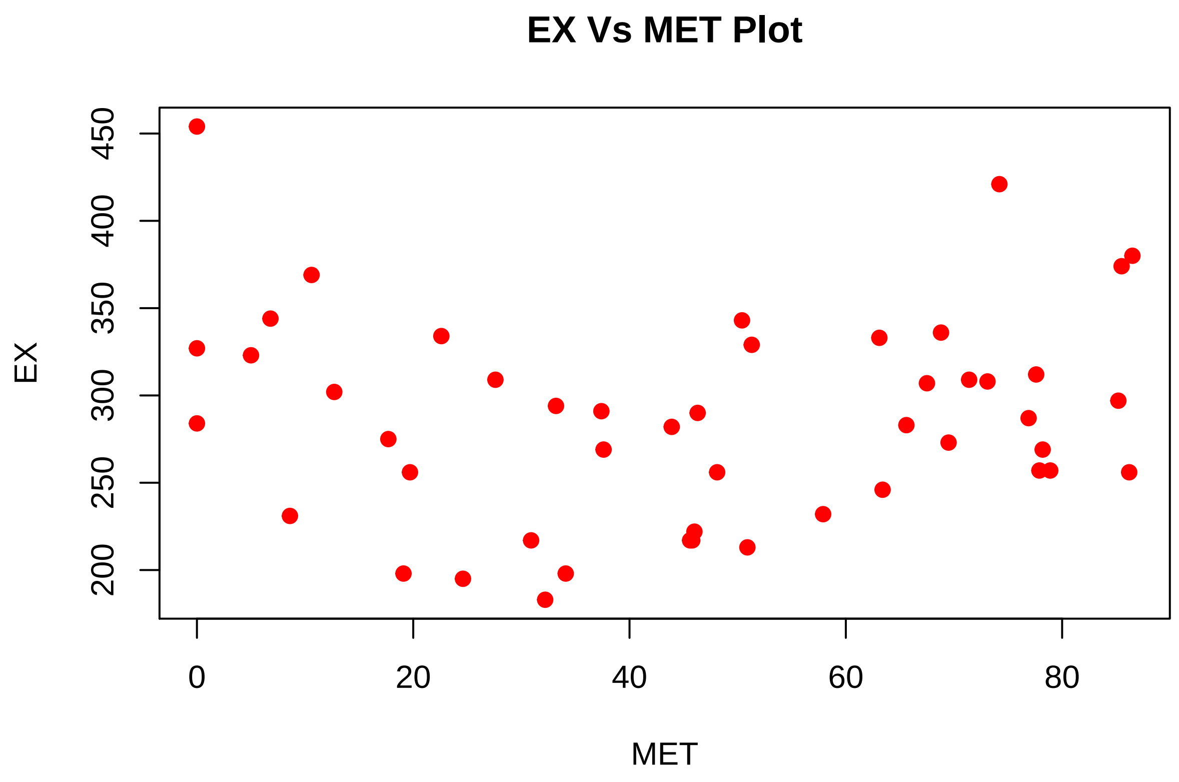
Table 2: Naive Bayes by loss matrix

	Missclass	Precision	Recall	Accuracy
Train_Naive_loss	0.546	0.9000000	0.2549575	0.454
Test_Naive_loss	0.508	0.9122807	0.2988506	0.492

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

Assignment 3. Uncertainty estimation

1. Plot EX versus MET using ordered dataset

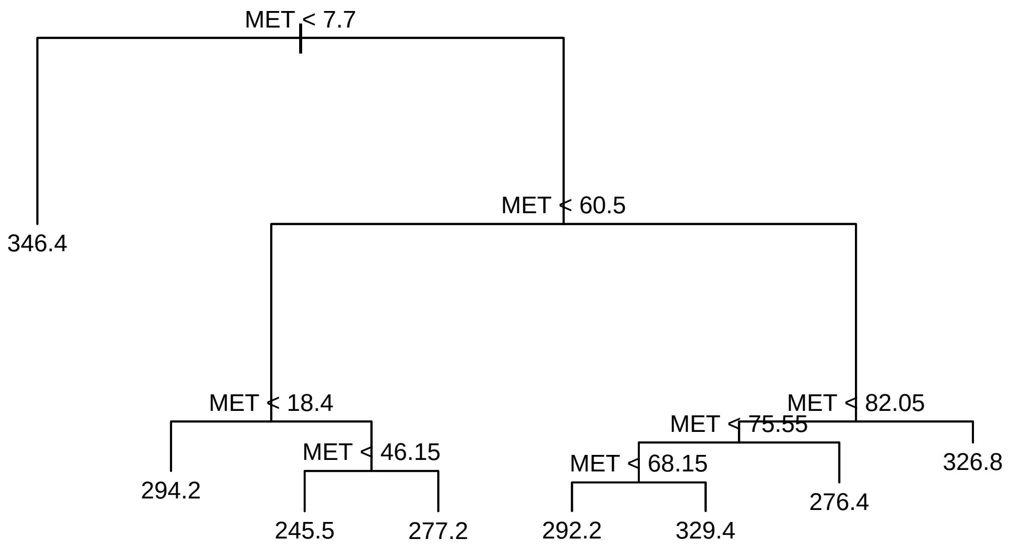


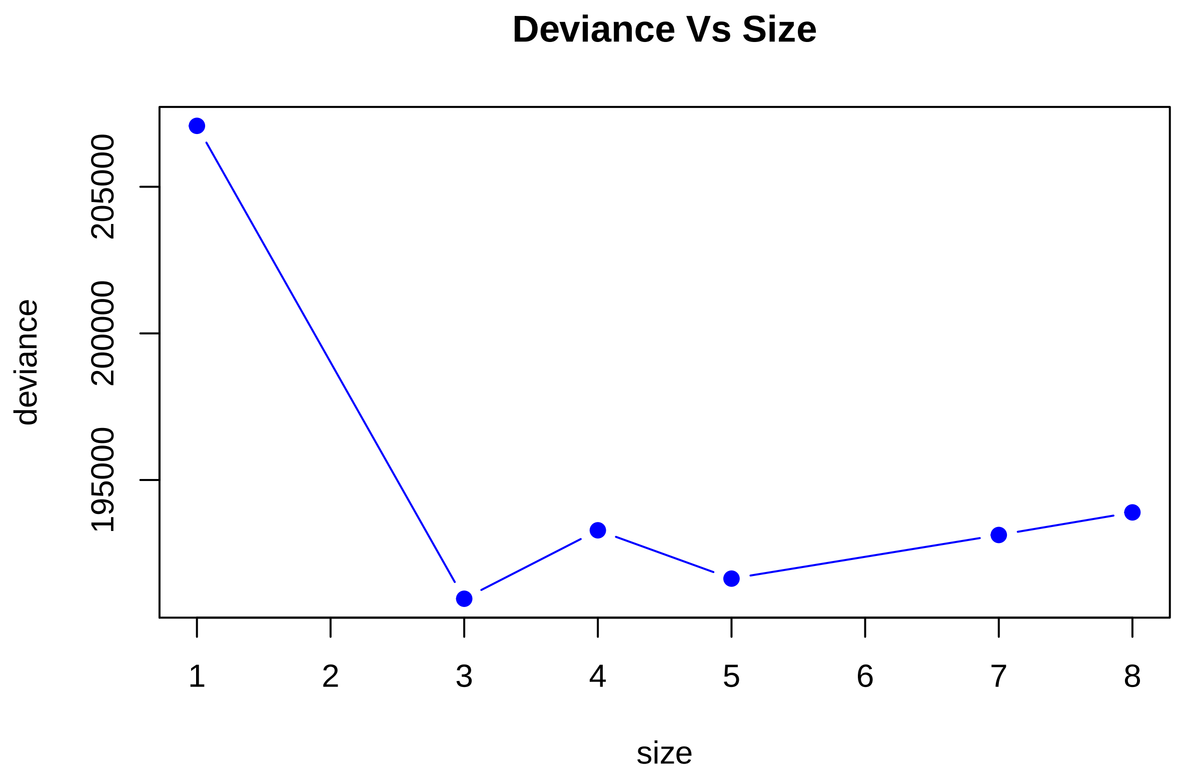
As we can see in the above plot, there is high variance among data as the data points are scattered. So linear of polynomial regression will not be a good fit to it. We think, decision trees would be good to fit for this data.

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

a) Selection of tree using cross validation

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

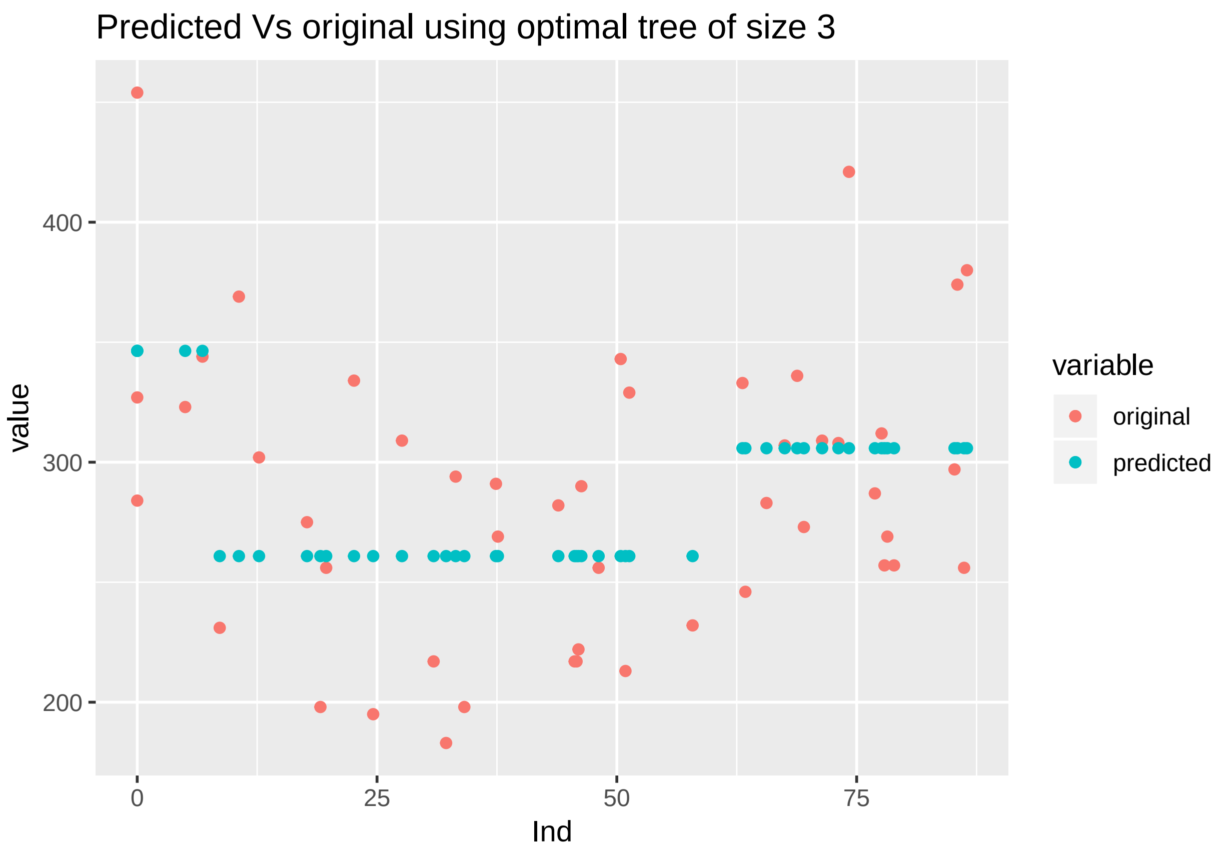




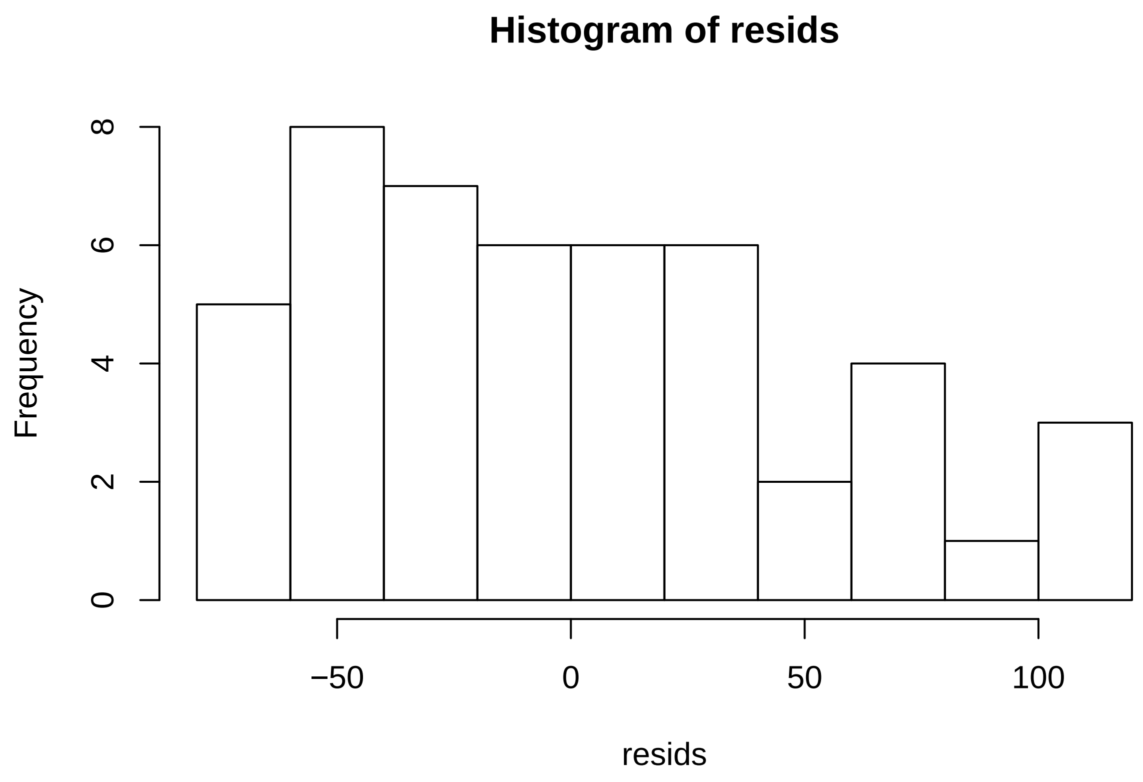
```
##
##  Optimal tree: 3
```

As see from the CV plot of deviance vs size, the least deviance(174057.6) is at 3, therefore best size is 3.

b) Plot of original and fitted data

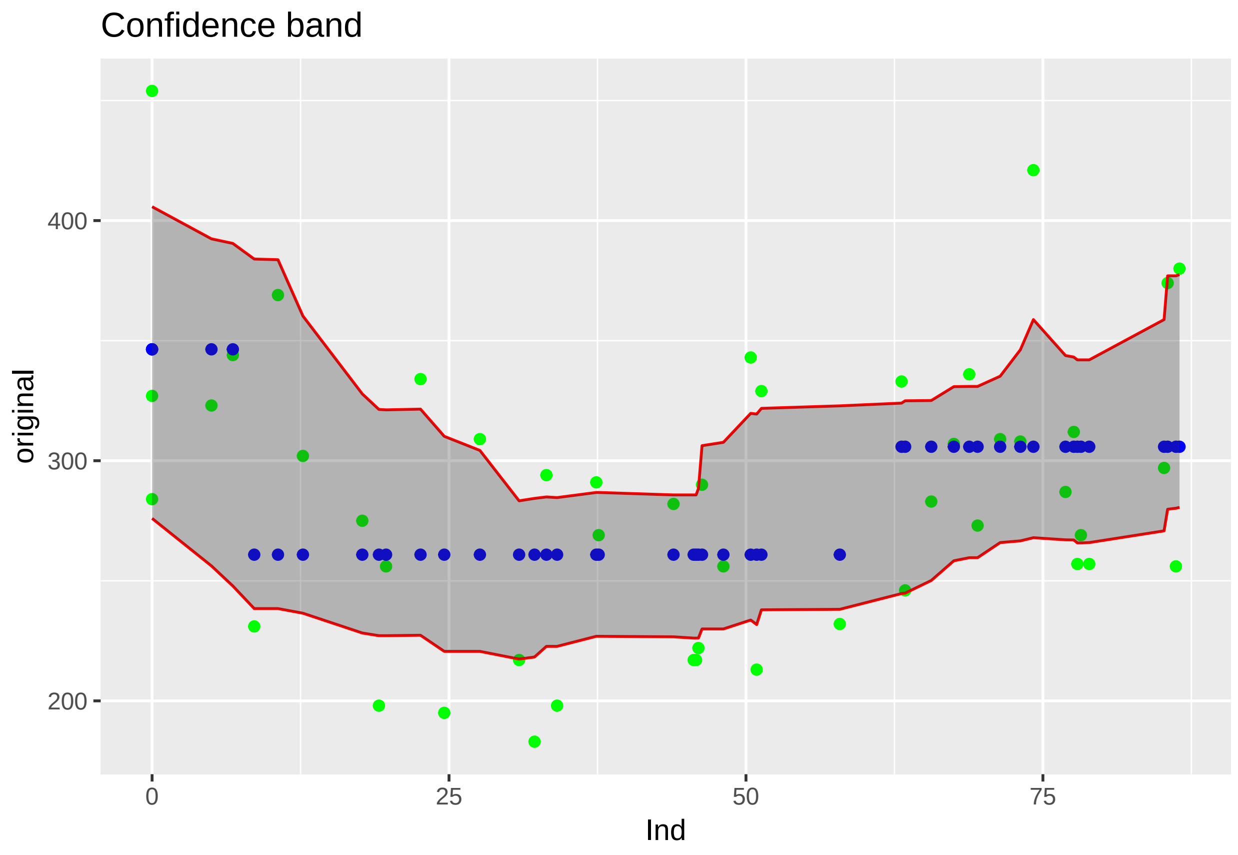


c) Histogramm of residuals



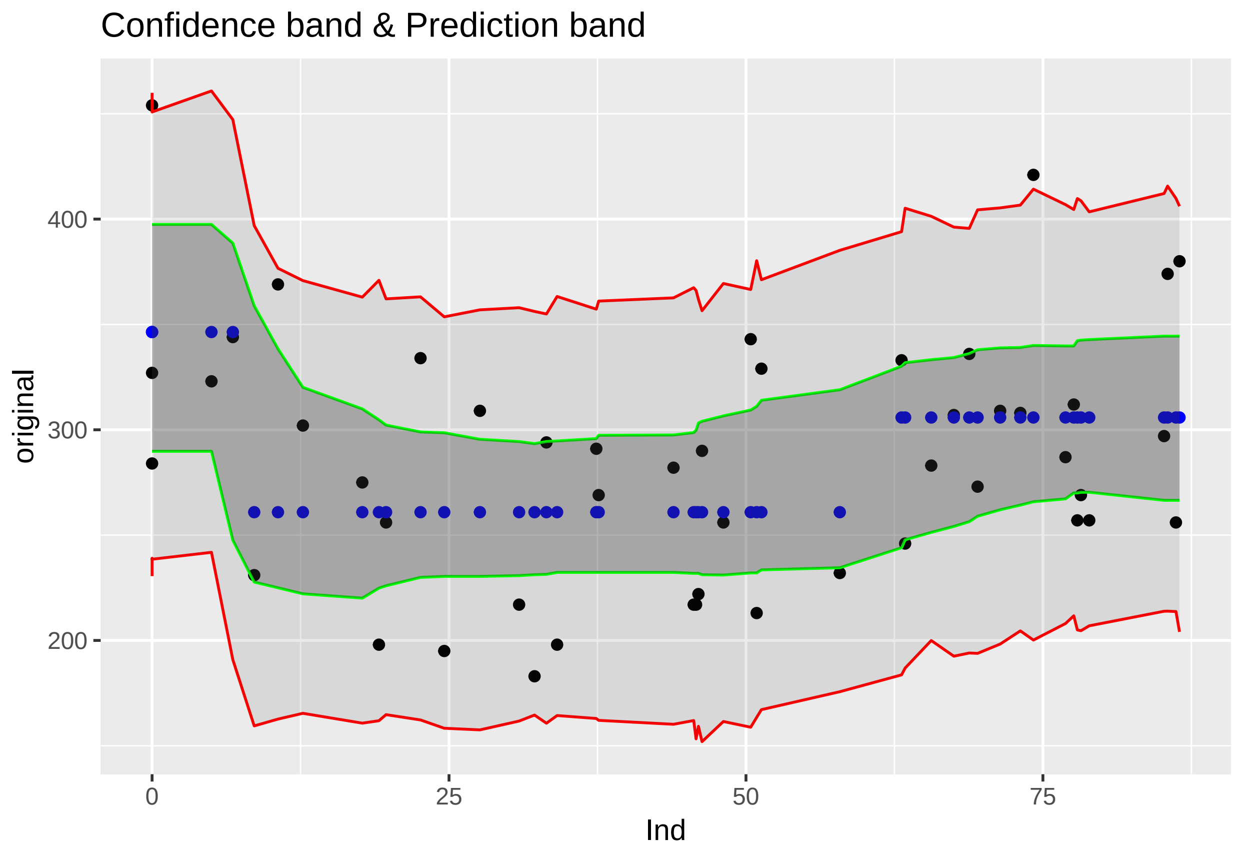
Residuals are not normally distributed and , in generally, models work better with more symmetrical or bell shaped distribution of residuals. This means ,in our case, fitting can be improved.

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



The band is not smooth, instead it is bumpy. The reason being, it is combination of different intervals calculated for different bootstrap iterations. The predictions we made lies inside the confidence band therefore, the model in step 2 is reliable.

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



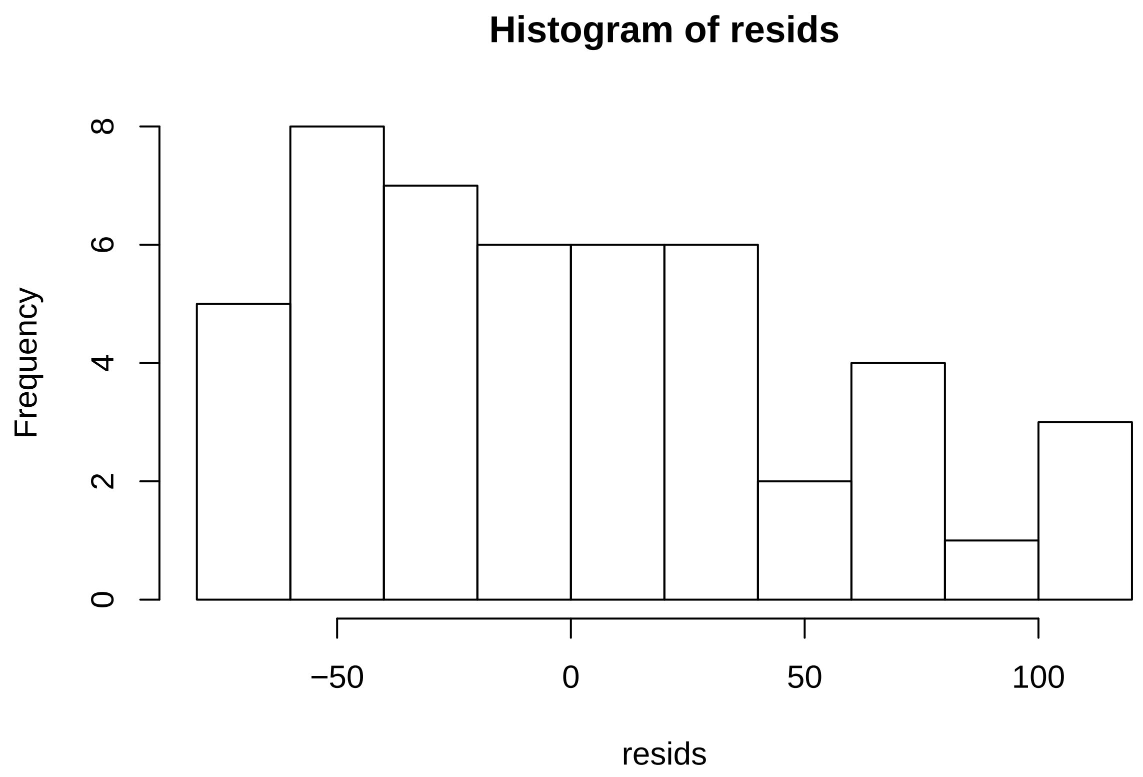
The green line represents the confidence band and red line represents the prediction band.

The cofidence band for parametirc bootstrap is also bumpy.

As the predctions we made in step 2 lie inside the confidence band therefore the model in step 2 appears reliable.

As we can see formm the plot above, the prediction band contains almost all the data except some which is almost 5%.

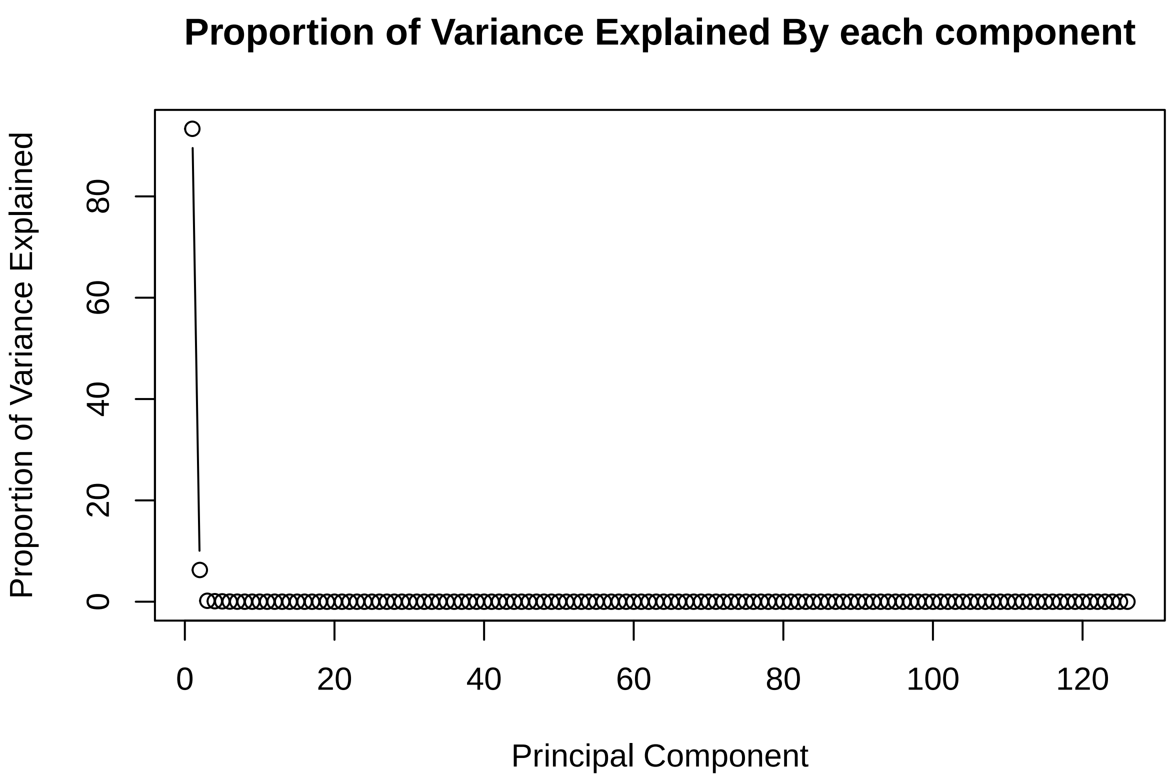
5. Analysis of histogram of residuals

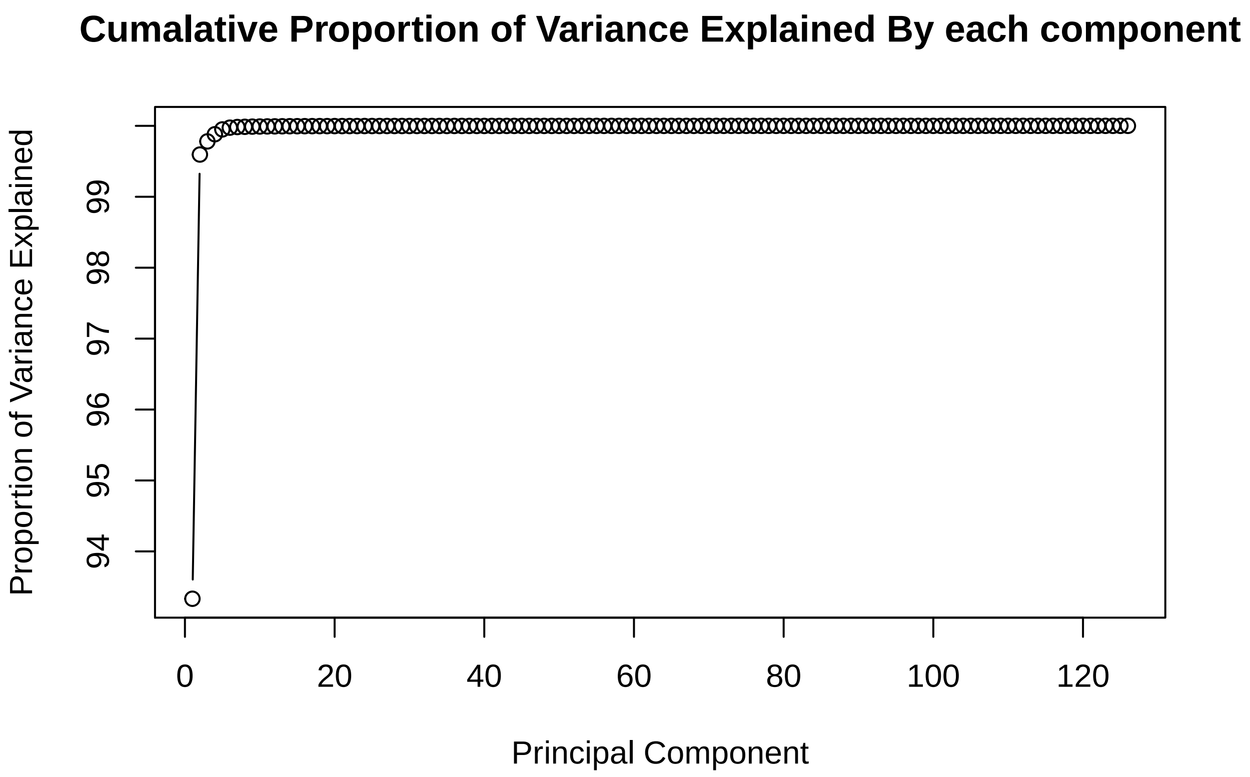


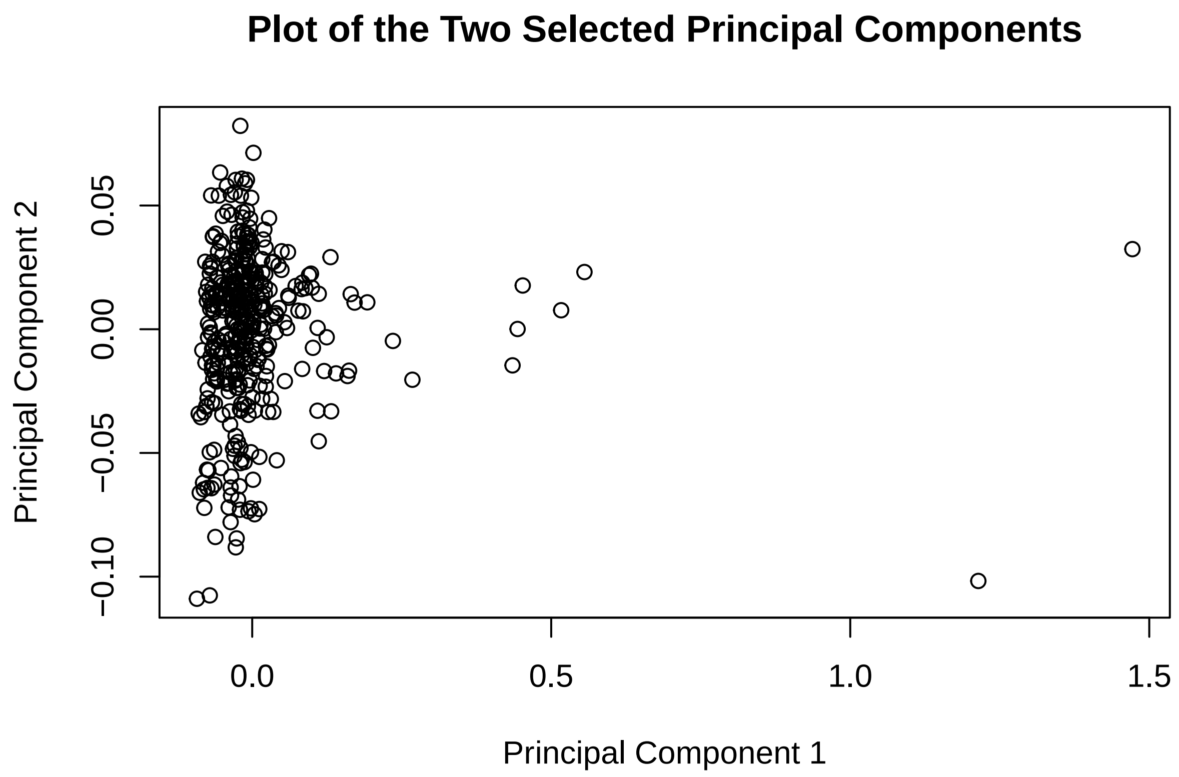
The histograms shows that, parametric booststrap is better than non-parametric bootstraping in this case. Because, as we saw in above graphs, the band for non-parametric bootstraping does not fit the data well.

Assignment 4

1 Principal Component Analysis



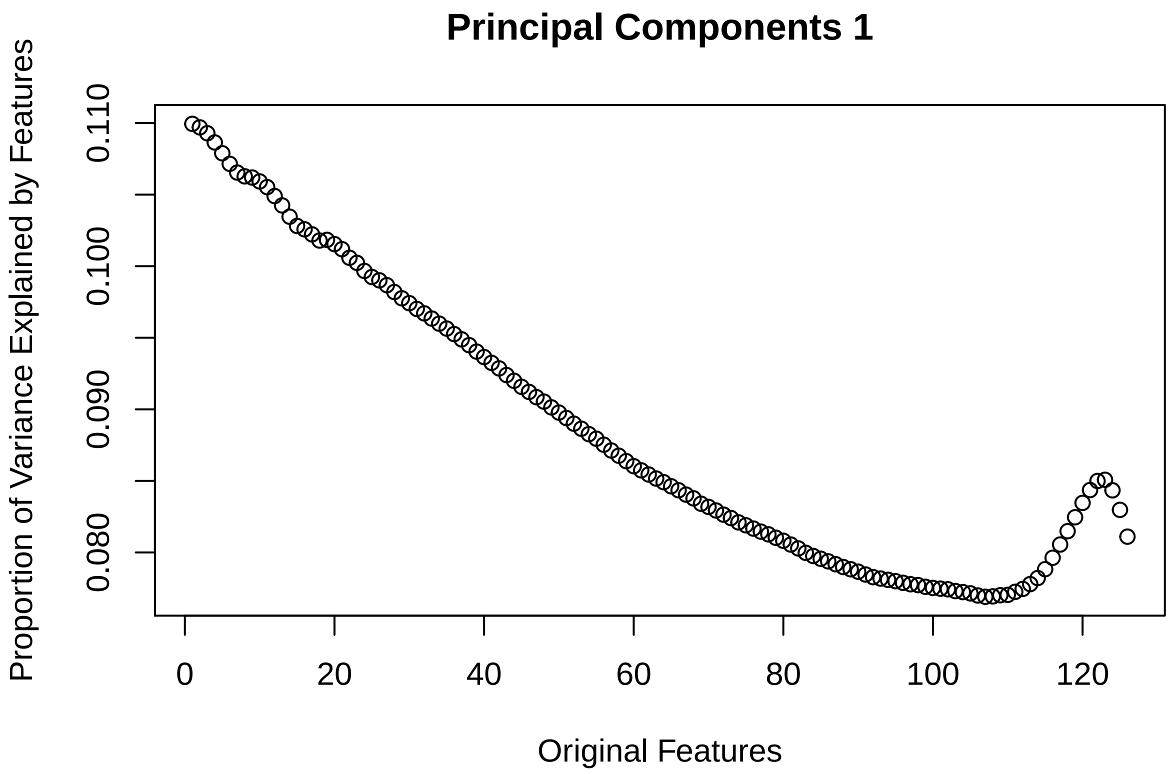

Cumalative Proportion of Variance Explained By each component
Proportion of Variance Explained
94
95
96
97
98
99
0
20
40
60
80
100
120
Principal Component

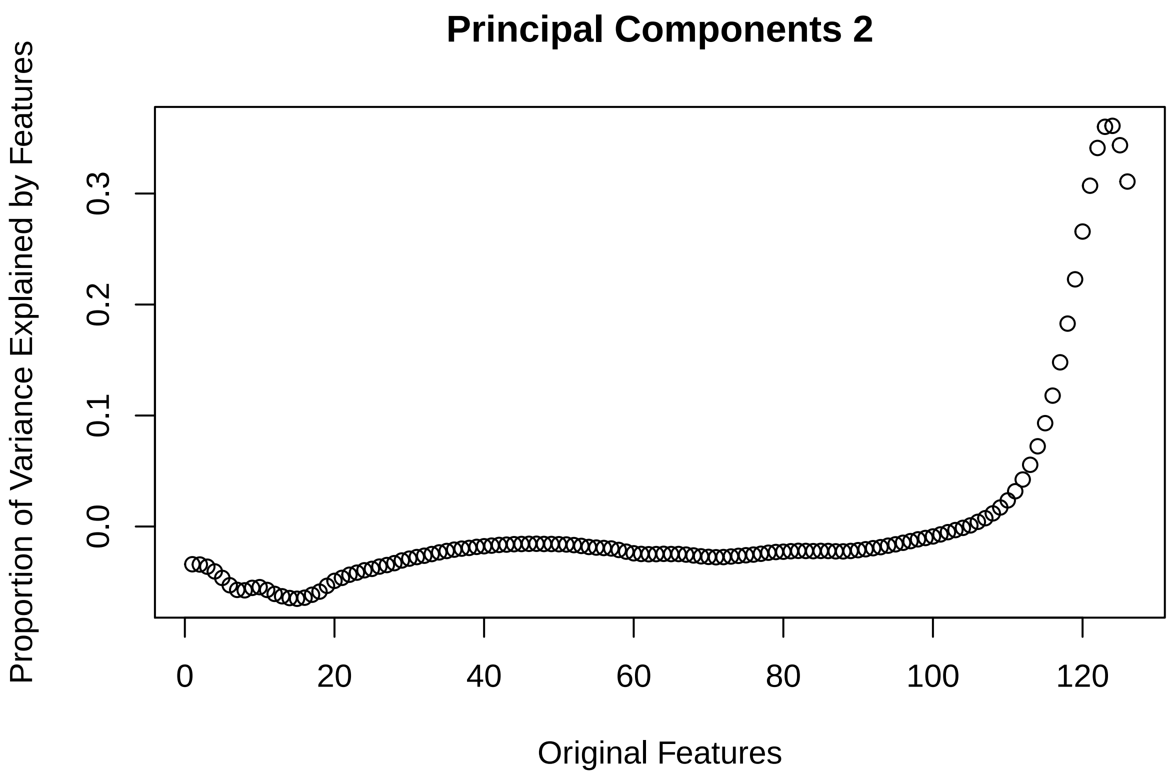


The first graph shows clearly that the first two principal components explain more than 99% of the total variance of the data, so we selected the first two principal components. We also made a cumalative percentage variance plot for the data and it showed the same thing, the first two components explain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

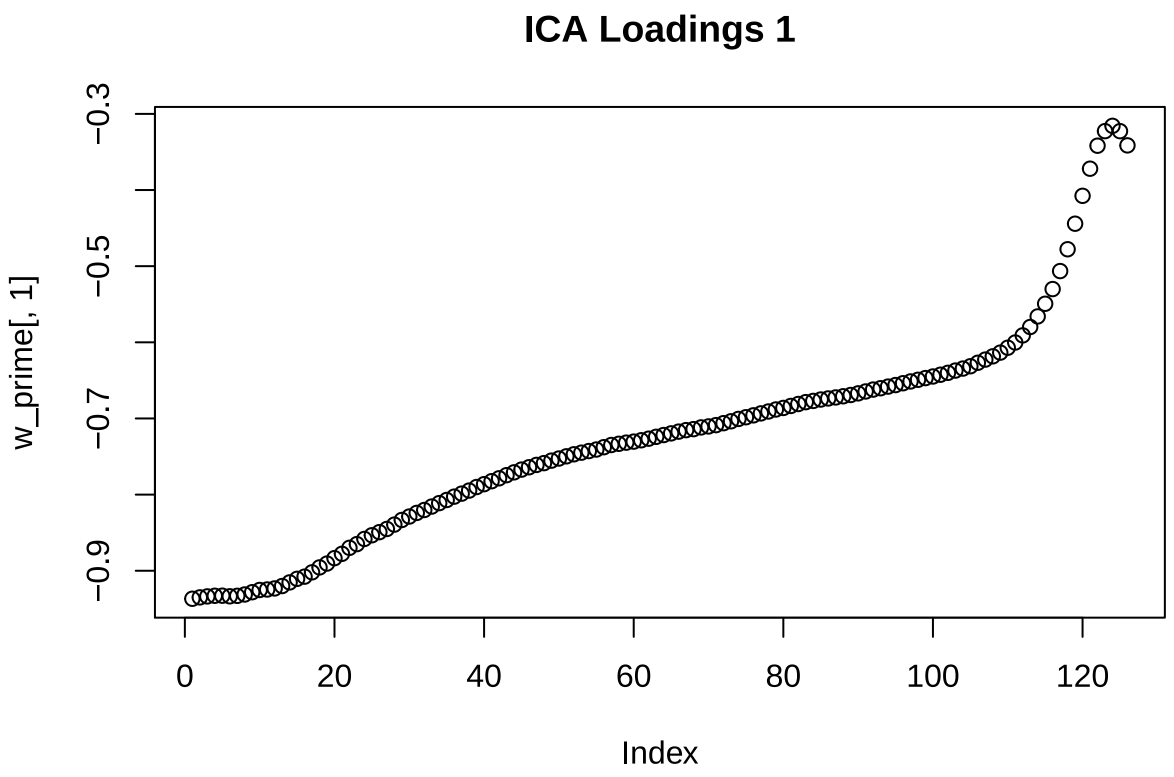


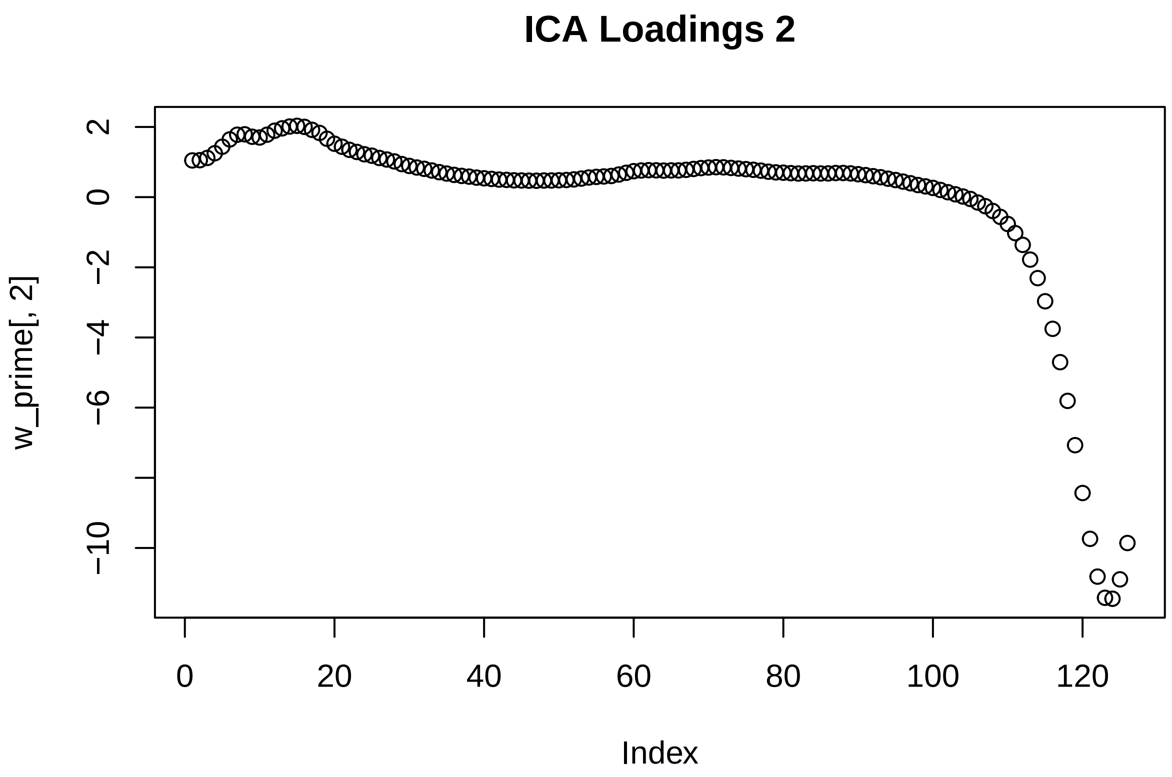


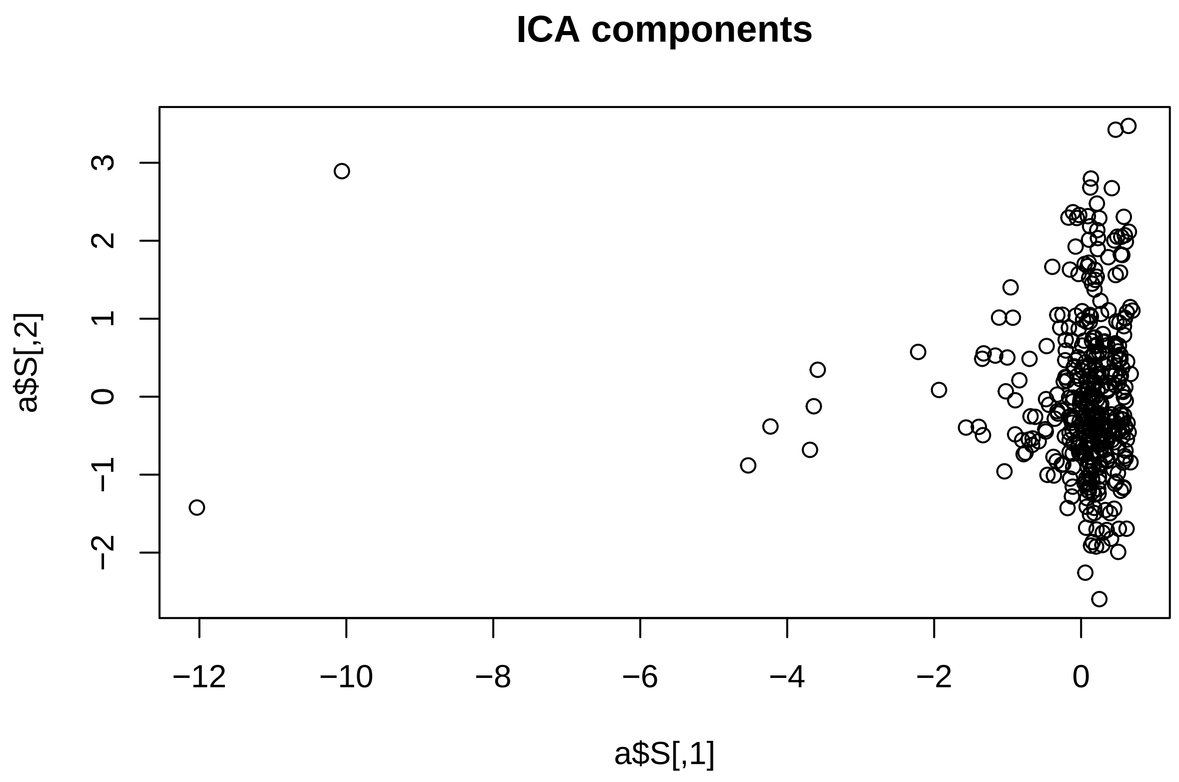
These are the trace plots showing how much contribution each of the original features make in the calculation of that principal component. The principal component 2(shown in the second plot) is explained by just a few of the original features. The features from 115 to 123 make majority of the contribution 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 1
w_prime[, 1]
−0.3
−0.5
−0.7
−0.9
0
20
40
60
80
100
120
Index


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



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.

Appendix

```
knitr::opts_chunk$set(
    echo = FALSE,
    message = FALSE,
    warning = FALSE
)
library(reshape2)
library(fastICA)
library(ggplot2)
library(readxl)
library(tree)
```

```r
library(MASS)
library(e1071)
credit<-read_xls("creditscoring.xls")
#credit <- na.omit(credit)
credit$good_bad<-as.factor(credit$good_bad)

n=dim(credit)[1]
set.seed(12345)
id=sample(1:n, floor(n*0.5))
train_credit=credit[id,]

id1=setdiff(1:n, id)
set.seed(12345)
id2=sample(id1, floor(n*0.25))
valid_credit=credit[id2,]


id3=setdiff(id1,id2)
test_credit=credit[id3,]

comparmat<-matrix(0,nrow=0,ncol=3)
colnames(comparmat)<-c("Precision","Recall","Accuracy")

cumresult<-function(conf_mat,nr){
    missclass_train<-(conf_mat[1,2]+conf_mat[2,1])/nr
    precision <- (conf_mat[2,2])/(conf_mat[1,2]+conf_mat[2,2])
    recall <- (conf_mat[2,2])/(conf_mat[2,1]+conf_mat[2,2])
    accuracy <- (conf_mat[1,1]+conf_mat[2,2])/nr

    return(list(missclass_train,precision,recall,accuracy))

}

comparmat <- matrix(ncol=3,nrow=0)
colnames(comparmat) <- c("Precision","Recall","Accuracy")
#Split using deviance
model_fitde<-tree(good_bad~.,data=train_credit,split="deviance")

predict_train<-predict(model_fitde,train_credit,type="class")
comp_train<-data.frame("Expected"=train_credit$good_bad,"Predicted"=predict_train)
conf_mat<-table(comp_train)

results <- cumresult(conf_mat,nrow(train_credit))
comparmat<-rbind(comparmat,c(results[[2]],results[[3]],results[[4]]))

cat("Confusion Matrix:\n\n")
conf_mat
cat("\nMissclassification Rate:",results[[1]],"\n")

predict_test<-predict(model_fitde,test_credit,type="class")
comp_test<-data.frame("Expected"=test_credit$good_bad,"Predicted"=predict_test)
conf_mat<-table(comp_test)
```

```r
results <- cumresult(conf_mat,nrow(test_credit))
comparmat<-rbind(comparmat,c(results[[2]],results[[3]],results[[4]]))

cat("Confusion Matrix:\n\n")
conf_mat
cat("\nMissclassification Rate:",results[[1]],"\n")
#Split using gini
model_fitgi<-tree(good_bad~.,data=train_credit,split="gini")

predict_train<-predict(model_fitgi,train_credit,type="class")
comp_train<-data.frame("Expected"=train_credit$good_bad,"Predicted"=predict_train)
conf_mat<-table(comp_train)

results <- cumresult(conf_mat,nrow(train_credit))
comparmat<-rbind(comparmat,c(results[[2]],results[[3]],results[[4]]))

cat("Confusion Matrix:\n\n")
conf_mat
cat("\nMissclassification Rate:",results[[1]],"\n")
predict_test<-predict(model_fitgi,test_credit,type="class")
comp_test<-data.frame("Expected"=test_credit$good_bad,"Predicted"=predict_test)
conf_mat<-table(comp_test)

results <- cumresult(conf_mat,nrow(test_credit))
comparmat<-rbind(comparmat,c(results[[2]],results[[3]],results[[4]]))

cat("Confusion Matrix:\n\n")
conf_mat
cat("\nMissclassification Rate:",results[[1]],"\n")
rownames(comparmat)<-c("Deviance_Train","Deviance_Test","Gini_Train","Gini_Test")
comparmat
pruned_tree <- prune.tree(model_fitde)
pruned_valid_tree <- prune.tree(model_fitde,newdata = valid_credit)


{plot(pruned_tree$size, pruned_tree$dev, type="b", col="red",ylim=c(260,630),
      xlab="Number of Leaf Nodes",ylab="Deviances")
points(x=pruned_valid_tree$size,y=pruned_valid_tree$dev, type="b", col="blue")
title(main="Training vs Validation deviances on pruned trees")
legend("topright",legend=c("Train Data Set", "Validation Data Set"),
       col=c("red","blue"),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 <-  pruned_valid_tree$size[which.min(pruned_valid_tree$dev)]

#Optinmal Tree based on leat deviance
pruned_tree_opt  <- prune.tree(model_fitde,best=best_lf)

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

{plot(pruned_tree_opt)
text(pruned_tree_opt,pretty=0)
title(main="Optimal Tree for Training Data")}

cat("Depth of Optimal Tree for training data:",depth_train,"\n")

cat("\nLabels used by training data:\n")
tree:::labels.tree(pruned_tree_opt)
comparmat <- matrix(ncol=4,nrow=0)
colnames(comparmat) <- c("Missclass","Precision","Recall","Accuracy")

predict_train_opt <- predict(pruned_tree_opt, newdata=train_credit,type="class")
comp_test<-data.frame("Expected"=train_credit$good_bad,"Predicted"=predict_train_opt)
conf_mat<-table(comp_test)

#Function to calculate various measuring parameter
results <- cumresult(conf_mat,nrow(train_credit))
comparmat<-rbind(comparmat,c(results[[1]],results[[2]],
                             results[[3]],results[[4]]))

#Prediction on test data
predict_test=predict(pruned_tree_opt, newdata=test_credit,type="class")
comp_train<-data.frame("Expected"=test_credit$good_bad,"Predicted"=predict_test)
conf_mat<-table(comp_train)

results <- cumresult(conf_mat,nrow(test_credit))
comparmat<-rbind(comparmat,c(results[[1]],results[[2]],
                             results[[3]],results[[4]]))


cat("\n\nConfusion Matrix:\n\n")
conf_mat
cat("\nMissclassification Rate:",results[[1]],"\n")

model_naive <- naiveBayes(good_bad~.,train_credit)

predict_naivetrain<-predict(model_naive,newdata=train_credit)
comp_train<-data.frame("Expected"=train_credit$good_bad,"Predicted"=predict_naivetrain)

conf_mat1<-table(comp_train)
results <- cumresult(conf_mat1,nrow(train_credit))

comparmat<-rbind(comparmat,c(results[[1]],results[[2]]
                             ,results[[3]],results[[4]]))

predict_naivetest <- predict(model_naive,newdata=test_credit)
comp_test<-data.frame("Expected"=test_credit$good_bad,
                      "Predicted"=predict_naivetest)

conf_mat2<-table(comp_test)
results <- cumresult(conf_mat2,nrow(test_credit))

comparmat<-rbind(comparmat,c(results[[1]],results[[2]]
                             ,results[[3]],results[[4]]))
```

```r
rownames(comparmat) <- c ("Train_optimal_tree","Test_optimal_tree",
                          "Train_by_naive","Test_by_naive")
cat("\n\nConfusion Matrix for Train Data:\n\n")
conf_mat1

cat("\n\nConfusion Matrix for Test Data:\n\n")
conf_mat2

cat("\n\n")

kableExtra::kable(comparmat,
                  caption="Comparison of Naive bayes vs Optimla Tree")
#Naive Bayes
prob_seq<-seq(0.05,0.95,0.05)
predntestp <- predict(model_naive,newdata=test_credit,type = "raw")
pred_matrix<-matrix(nrow=0,ncol=length(prob_seq)+1)

orig_mat <- ifelse(test_credit$good_bad =="good",1,0)
for(i in 1:nrow(predntestp)) {
    pred_matrix<-rbind(pred_matrix,c(orig_mat[i],
                                     ifelse(predntestp[i,2]>prob_seq,1,0)))
}

pred_matrix<-as.data.frame(pred_matrix)

colnames(pred_matrix)[1] <- "Original Value"
for(i in 1:length(prob_seq)) {
    colnames(pred_matrix)[i+1] <- paste("Probability >",prob_seq[i])
}

TPR_naive <- apply(pred_matrix[,-1],2,
                   function(x)(sum(pred_matrix[,1]*x)/sum(pred_matrix[,1])))

FPR_naive <- apply(pred_matrix[,-1],2,
                   function(x)(sum(x)-
                               sum(pred_matrix[,1]*x))/sum(pred_matrix[,1]==0))

#Optimal Tree
pred_treopttes <- predict(pruned_tree_opt, newdata=test_credit)
pred_matrix1<-matrix(nrow=0,ncol=length(prob_seq)+1)

for(i in 1:nrow(pred_treopttes)) {
    pred_matrix1 <- rbind(pred_matrix1, c(orig_mat[i],
    ifelse(pred_treopttes[i, 2] > prob_seq, 1, 0)))
}
pred_matrix1 <- as.data.frame(pred_matrix1)

colnames(pred_matrix1)[1] <- "Original Value"

for(i in 1:length(prob_seq)) {
    colnames(pred_matrix1)[i+1] <- paste("Probability >",prob_seq[i])
}
```

```r
TPR_optimal <- apply(pred_matrix1[,-1],2,
                     function(x)(sum(pred_matrix1[,1]*x)/sum(pred_matrix1[,1])))
FPR_optimal <- apply(pred_matrix1[,-1],2,
                     function(x)(sum(x)-
                                    sum(pred_matrix1[,1]*x))/sum(pred_matrix1[,1]==0))

metricsdata <- as.data.frame(cbind("TPR_opt"=TPR_optimal,
                       "FPR_opt"=FPR_optimal,
                       "TPR_naive"= TPR_naive,
                       "FPR_naive"=FPR_naive))



ggplot() + geom_line(data=metricsdata,aes(x=FPR_opt,y=TPR_opt,color="red")) +
 geom_line(data=metricsdata,aes(x=FPR_naive,y=TPR_naive,color="blue"))+ scale_color_discrete(name="Model
    geom_abline(intercept=0,slope=1)+
    xlab("FPR")+ylab("TPR")+ggtitle("ROC curve between Naive Bayes and Optimal Tree")
comparmat2 <- matrix(ncol=4,nrow=0)
colnames(comparmat2) <- c("Missclass","Precision","Recall","Accuracy")

naive_lossmat <- naiveBayes(good_bad~., train_credit)
predict_train<-predict(naive_lossmat,train_credit,type="raw")
predict_test<-predict(naive_lossmat,test_credit,type="raw")

train_mat <- predict_train
test_mat <- predict_test

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

comp_train<-data.frame("Expected"=train_credit$good_bad,"Predicted"=train_mat)
conf_mattrain<-table(comp_train)

results<-cumresult(conf_mattrain,nrow(train_credit))

comparmat2<-rbind(comparmat2,c(results[[1]],results[[2]]
                             ,results[[3]],results[[4]]))

comp_test<-data.frame("Expected"=test_credit$good_bad,"Predicted"=test_mat)
conf_matest<-table(comp_test)

results<-cumresult(conf_matest,nrow(test_credit))

comparmat2<-rbind(comparmat2,c(results[[1]],results[[2]]
                             ,results[[3]],results[[4]]))

rownames(comparmat2)<-c("Train_Naive_loss","Test_Naive_loss")

cat("\n\nConfusion Matrix for Train Data:\n\n")
conf_mattrain

cat("\n\nConfusion Matrix for Test Data:\n\n")
conf_matest
```

```r
kableExtra::kable(comparmat2,caption="Naive Bayes by loss matrix")
set.seed(12345)

#Libraries
library(tree)
library(ggplot2)

#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)
# 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")
```

```r
#hsitogram
resids<- (data$EX - preds)
hist(resids)
#part 3
library(boot)

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

#resConf = boot.ci(bootResults, type="bca")

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="green")+
  geom_point(aes(Ind,predicted),color="blue")+
  geom_line(aes(Ind,upper),color="red")+
  geom_line(aes(Ind,lower),color="red")+
  geom_ribbon(aes(x=Ind,ymin= lower,ymax=upper),alpha=0.3)+
  ggtitle("Confidence band")
# part4, parammetric bootstraping

paramConf <- function(data){
  model = tree( EX ~ MET, data=data, control = tree.control(nobs = nrow(data),minsize = 8))
  bestTree = prune.tree(model, best=bestSize)
  preds<- predict(bestTree, data)
  return(preds)
}

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)
}

confBootResults = boot(data, statistic = paramConf, R=1000, mle = bestTree, ran.gen = ranGenFunc, sim =
confBoundParam = envelope(confBootResults, level=0.95)

predictBootResults <- boot(data,statistic = bootStrapParam , R=1000, mle=bestTree,sim="parametric",ran.g
PredBound <- envelope(predictBootResults,level = 0.95)


upperLimitsConf <- confBoundParam$point[1,]
lowerLimitsConf <- confBoundParam$point[2,]

upperLimitsPred <- PredBound$point[1,]
lowerLimitsPred <- PredBound$point[2,]

results["upperc"] = upperLimitsConf
results["lowerc"] = lowerLimitsConf
results["upperp"] = upperLimitsPred
results["lowerp"] = lowerLimitsPred

ggplot(results, aes(Ind,original,predicted,upperc,lowerc,upperp,lowerp))+
  geom_point(aes(Ind,original),color="black")+
  geom_point(aes(Ind,predicted),color="blue")+
  geom_line(aes(Ind,upperp),color="red")+
  geom_line(aes(Ind,lowerp),color="red")+
  geom_ribbon(aes(x=Ind,ymin= lowerp,ymax=upperp),alpha=0.1)+
  geom_line(aes(Ind,lowerc),color="green")+
  geom_line(aes(Ind,upperc),color="green")+
  geom_ribbon(aes(x=Ind,ymin= lowerc,ymax=upperc),alpha=0.3)+
  ggtitle("Confidence band & Prediction band") +
  scale_color_manual("Line.Color", values=c(red="red",green="green",blue="blue"),
                     labels=paste0("Int",1:3))
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")
```

```r
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")
```