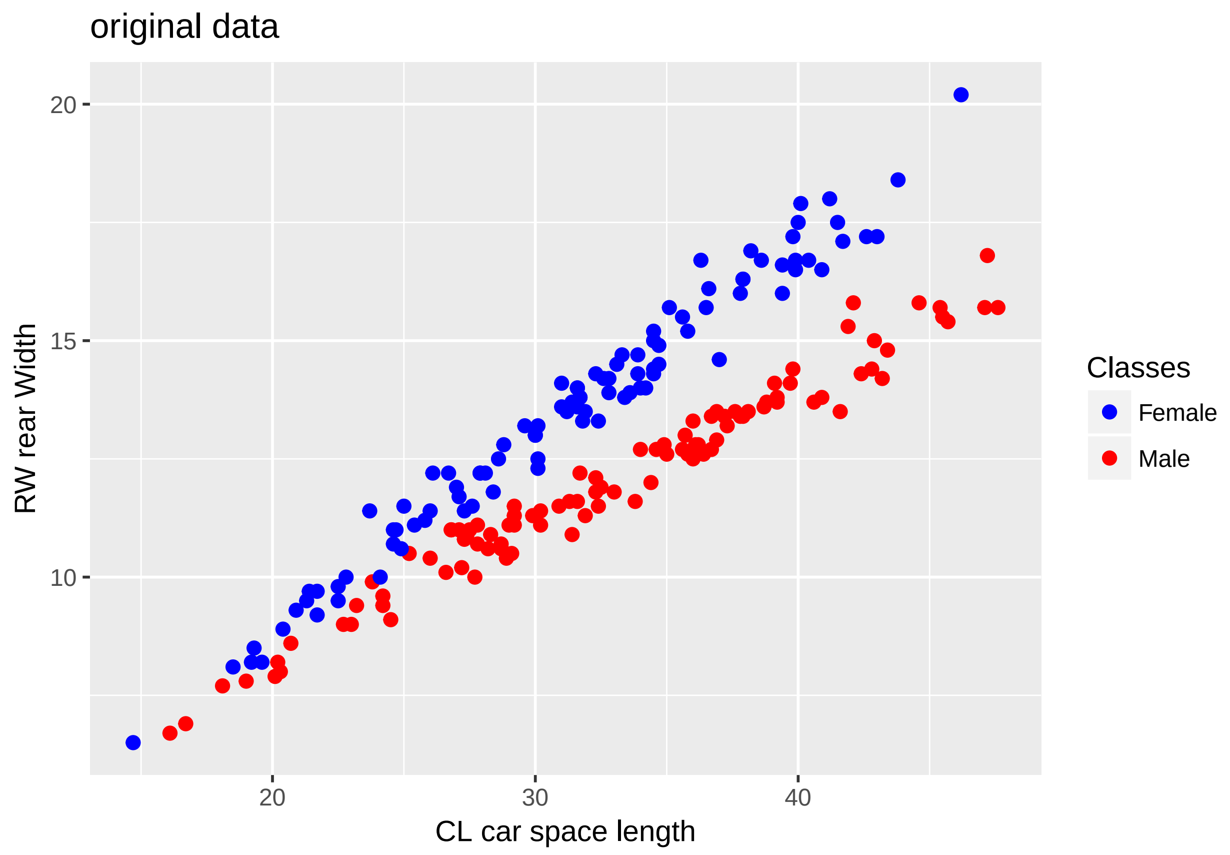
Group8_lab2

Group8

November 29, 2017

Assignment 1 LDA and Logistic Regression

Part 1



Yes , it is easy to classify the data by linear discriminant function because from the plot it can be observed that the two classes have cooperative distribution having equal covariance.

Part 2

Discriminant function coefficients for male

```
## [1]   2.5658514  -0.2138144 -12.5634175
```

Discriminant function coefficients for Female

```
## [1]   8.248698  -2.161318 -22.428769
```

Decision boundary Equation

$RW * W_{1R} + CL * W_{1C} + W_{01} = RW * W_{2R} + CL * W_{2C} + W_{02}$

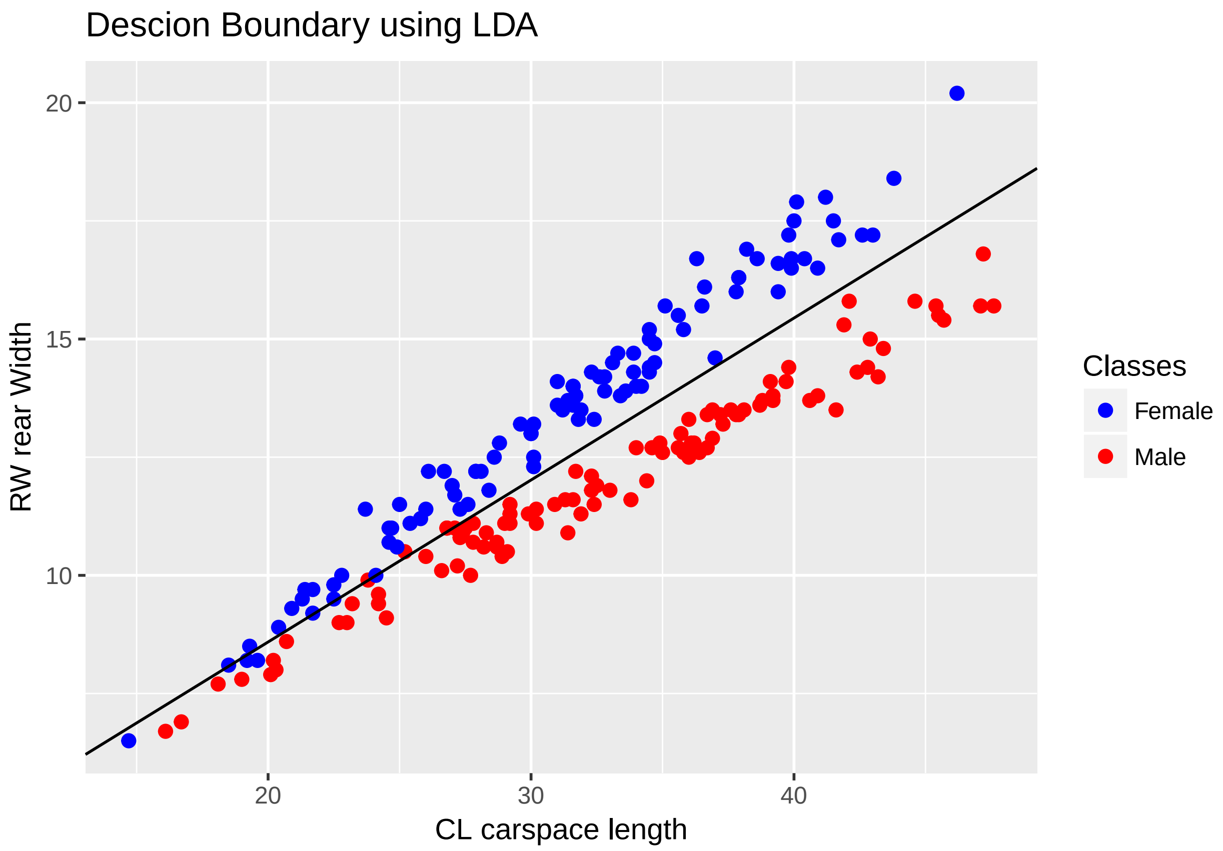
Final equation in term or RW as a Function of CL

$RW = \frac{(CL(W_{2C} - W_{1C}) + W_{02} - W_{02})}{W_{1R} - W_{2R}}$

$intercept = \frac{CL(W_{2C} - W_{1C})}{W_{1R} - W_{2R}}$

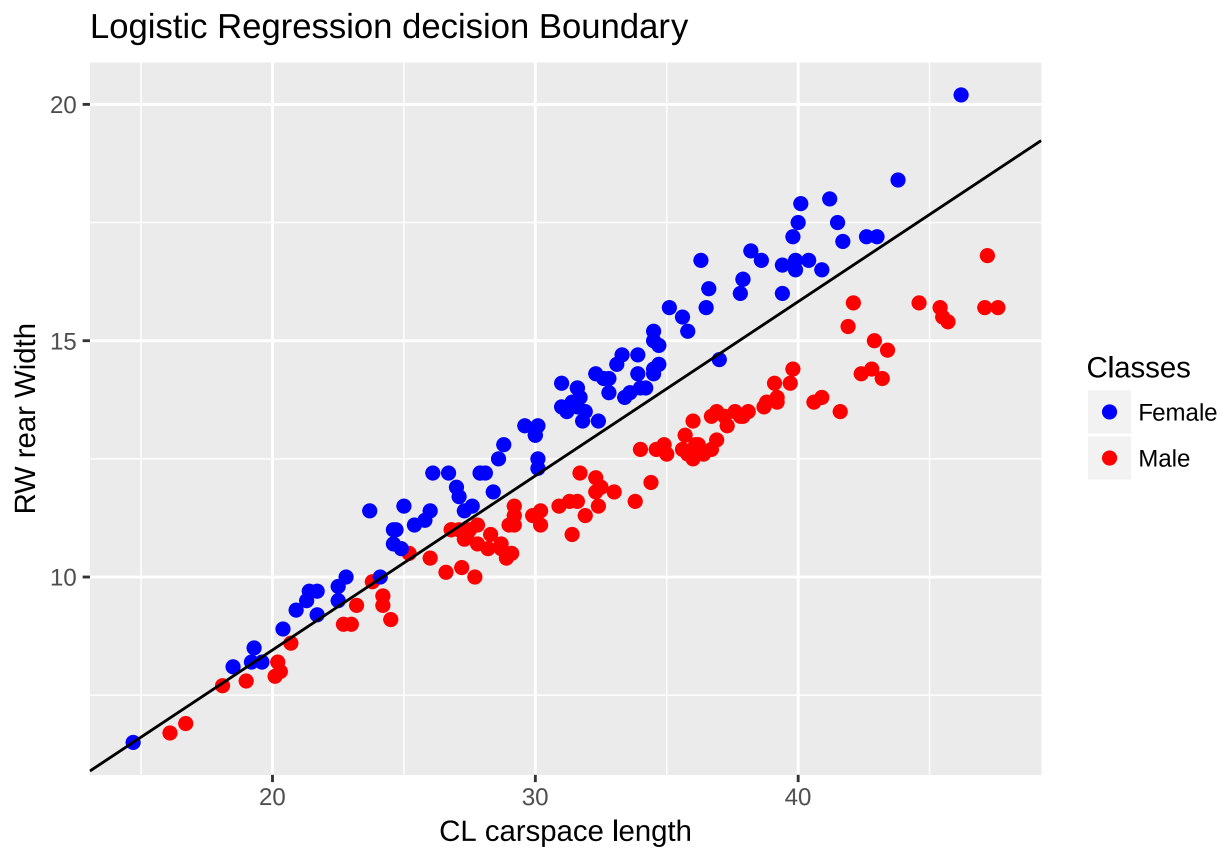
$slope = \frac{W_{02} - W_{02}}{W_{1R} - W_{2R}}$

Part 3



It is evident from the plot that decision boundary is separating the data between the two classes classifying points belonging to each class on the respective side of the boundary. There are some points which are misclassified but since there is not a lot of misclassified data that is misclassification rate is low therefore it can be assumed that the quality of fit is good.

Part 4



Decision boundary equation for Logistic Regression

$RW(W_{RW}) + CL(W_{CL}) + W_{01} = 0.5$

In terms of RW

$RW = \frac{-CL*W_{CL}}{W_{RW}} - \frac{W_{01}}{W_{RW}} + \frac{0.5}{W_{RW}}$

$intercept = \frac{-CL(W_{CL})}{W_{RW}}$ $slope = -\frac{W_{01}}{W_{RW}}$

Comaprision

Apparently there are more misclassified points for female in LDA while on applying logistic regression, the female points are classified better than that of male. One more thing which can be observed from the plot is that for LDA, there are very few points which lie on the decision boundary while for Logistic Regression, there are comparatively more points which lie on the decision boundary that iis which are neither classified as male nor as female.

Assignment 2. Analysis of Credit Scoring

Part 2

Misclassification for Training Data

Missclassification Rate of Deviance

```
##
## devi_yfit bad good
##      bad   61   20
##      good  86  333
```

```
## [1] 0.212
```

Missclassification rate Gini

```
##
## gini_yfit bad good
##      bad   68   36
##      good  79  317
```

```
## [1] 0.23
```

Misclssification for Test Data

Missclassification Rate of Deviance

```
##
## devi_yfit bad good
##      bad   24   17
##      good  54  155
```

```
## [1] 0.284
```

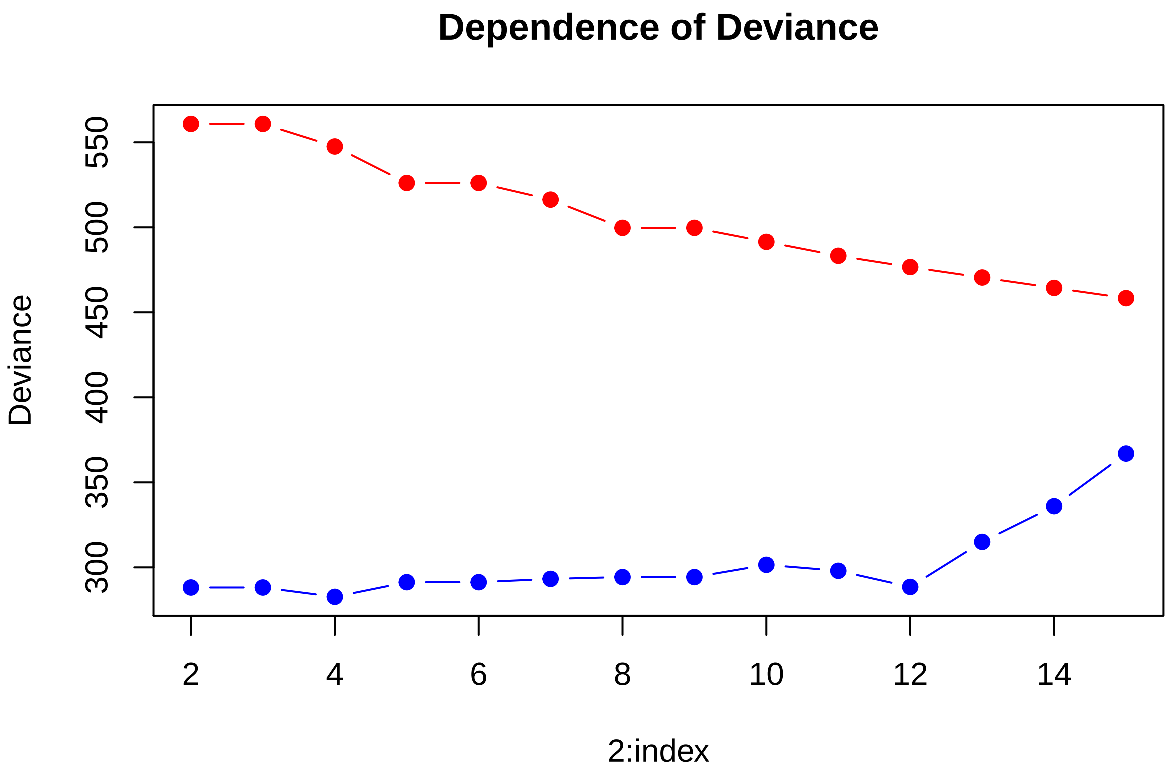
Missclassification Rate of Gini

```
##
## gini_yfit bad good
##      bad   23   30
##      good  55  142
```

```
## [1] 0.34
```

Since the misclassification rate for both training and test data for Gini is less therefore it is a better measure.

Part 3



```
##
## Classification tree:
## snip.tree(tree = tree_deviance, 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
```

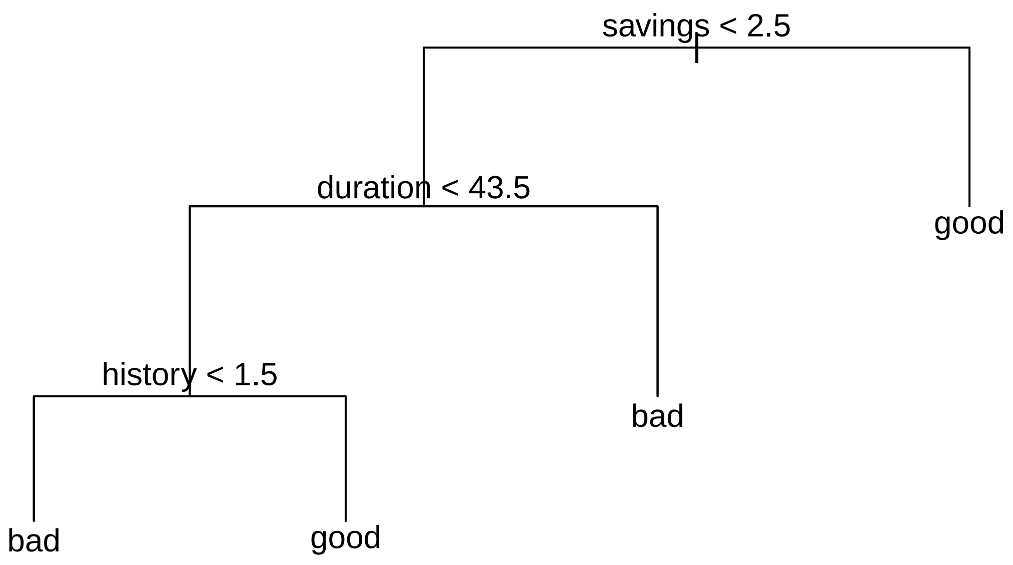
Optimal depth of tree is 4 and the variables selected by the tree are `savings`, `duration` and `history`

```
## [1] "Confusion Matrix"
```

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

```
## [1] "Misclassification rate"
```

```
## [1] 0.26
```



According to the selected tree, if savings is greater than 2.5 then its good else bad. Moreover, if the duration of savings is less than 43.5 months then its good else bad and if history of savings is maintained for less more than 1.5 then its good.

Part 4

```
##
## nav_test bad good
##     bad   47   49
##     good  31  123
```

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

```
## Misclassification train data value Using Naive Bayes is: 0.3
```

```
## Misclassification test data value Using Naive Bayes is: 0.32
```

The misclassification rate calculated by Naive Bayes in step 4 for test data is `0.32` which is greater than that of misclassification calculated by the decision tree in step 3 that is `0.26`

```
##       nav_train
##        FALSE TRUE
##   bad    137   10
##   good   263   90
```

```
## [1] 0.546
```

```
##       nav_test
##        FALSE TRUE
##   bad     70    8
##   good   131   41
```

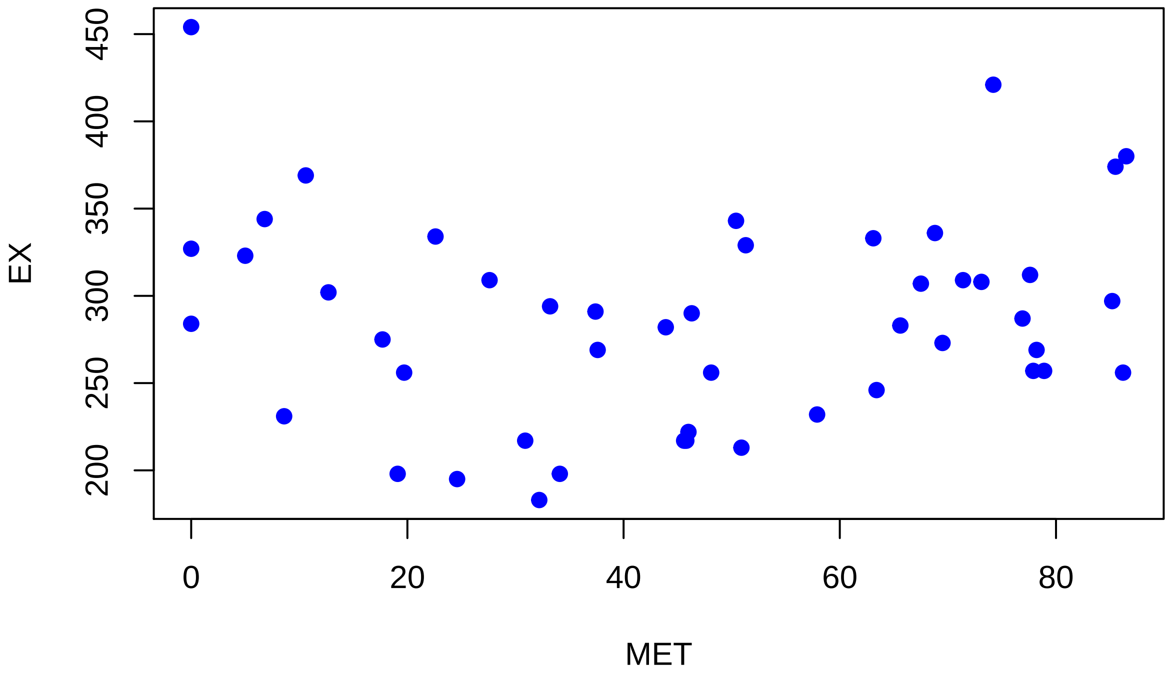
```
## [1] 0.556
```

The misclassification rate calculated using the given loss function is `0.546` for training data and `0.556` for test data which is much greater than the one calculated by Naive Bayes without using loss function that is

`0.3` for training data and `0.32` for test data. The increase in misclassification is due to the introduction of loss function giving a threshold for the true and false values.

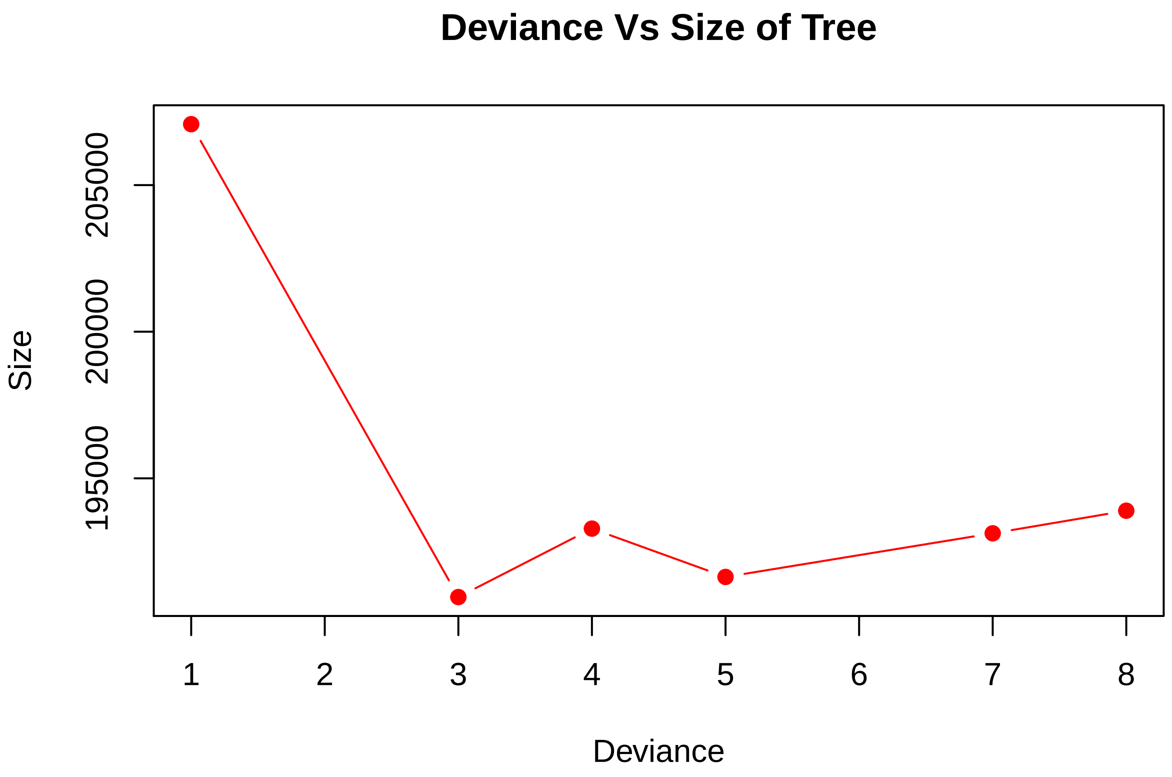
Assignment 3 Uncertainty estimation

Part 1

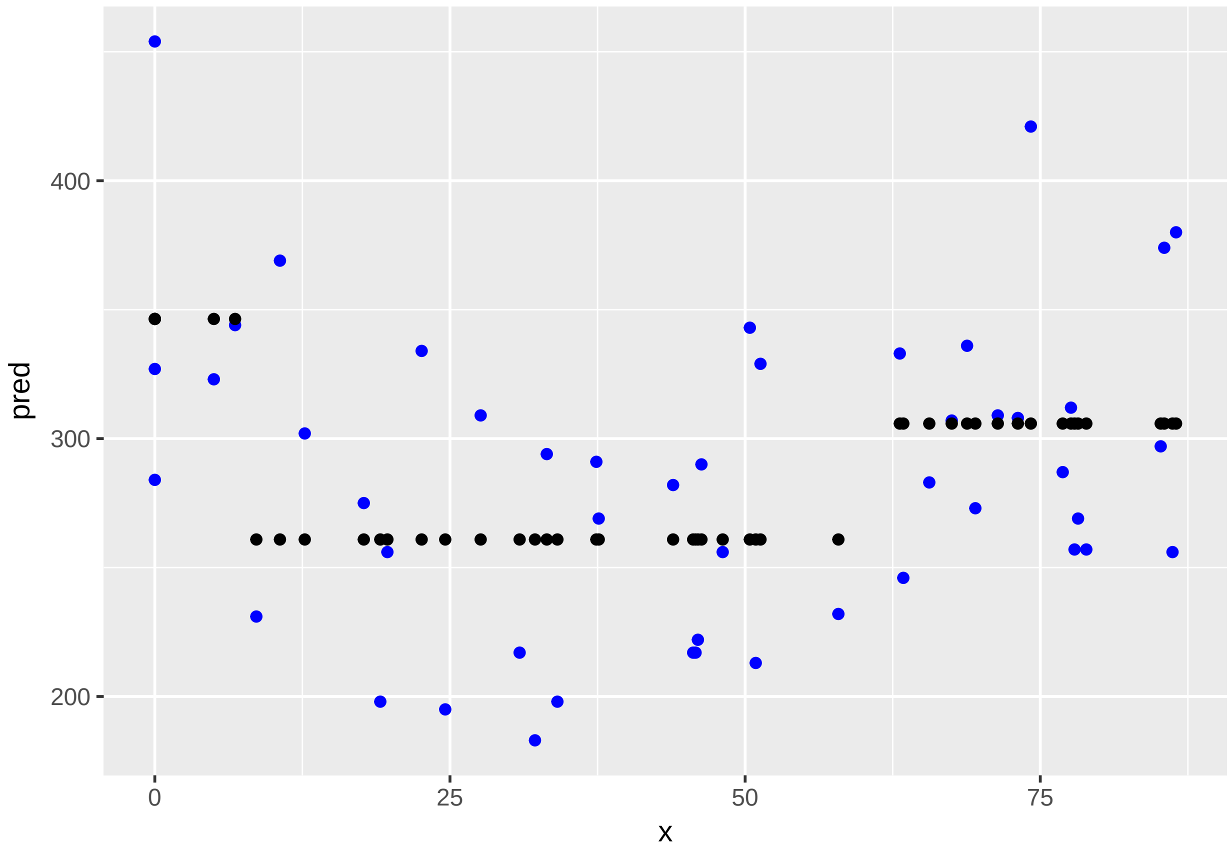


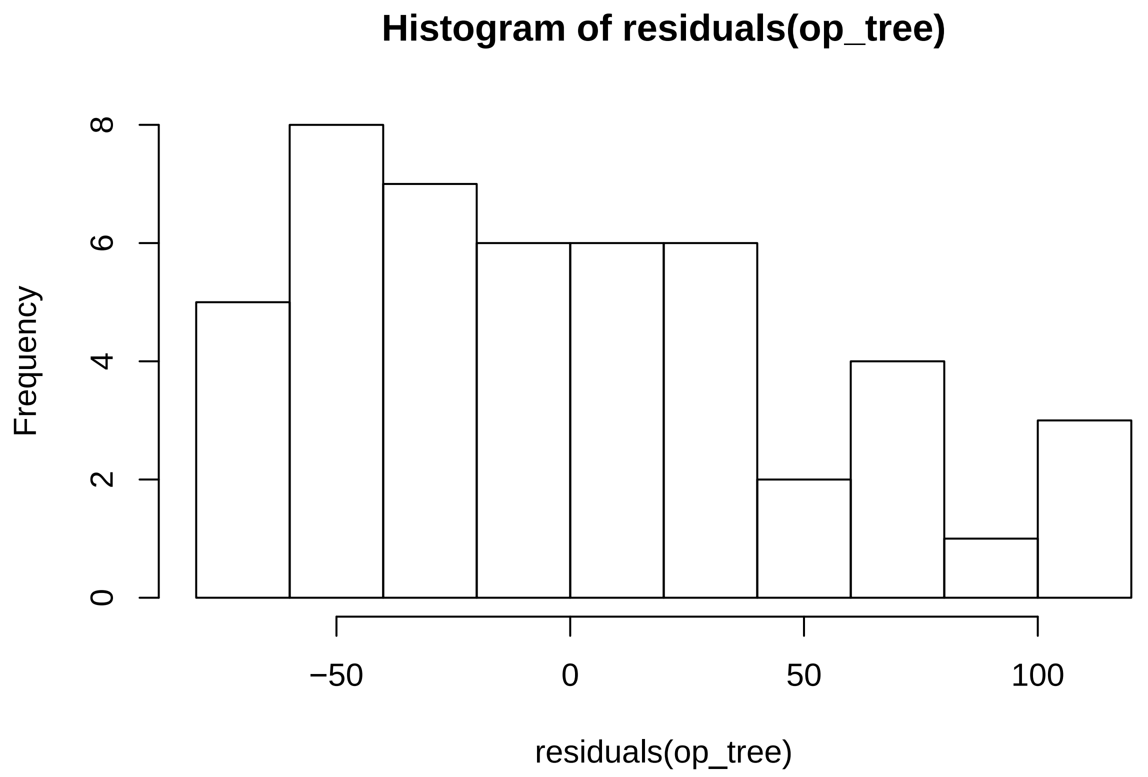
From the plot it can be observed that the data is scattered that is variance is high thus for the this type of data, decision trees would be the appropriate method

Part 2



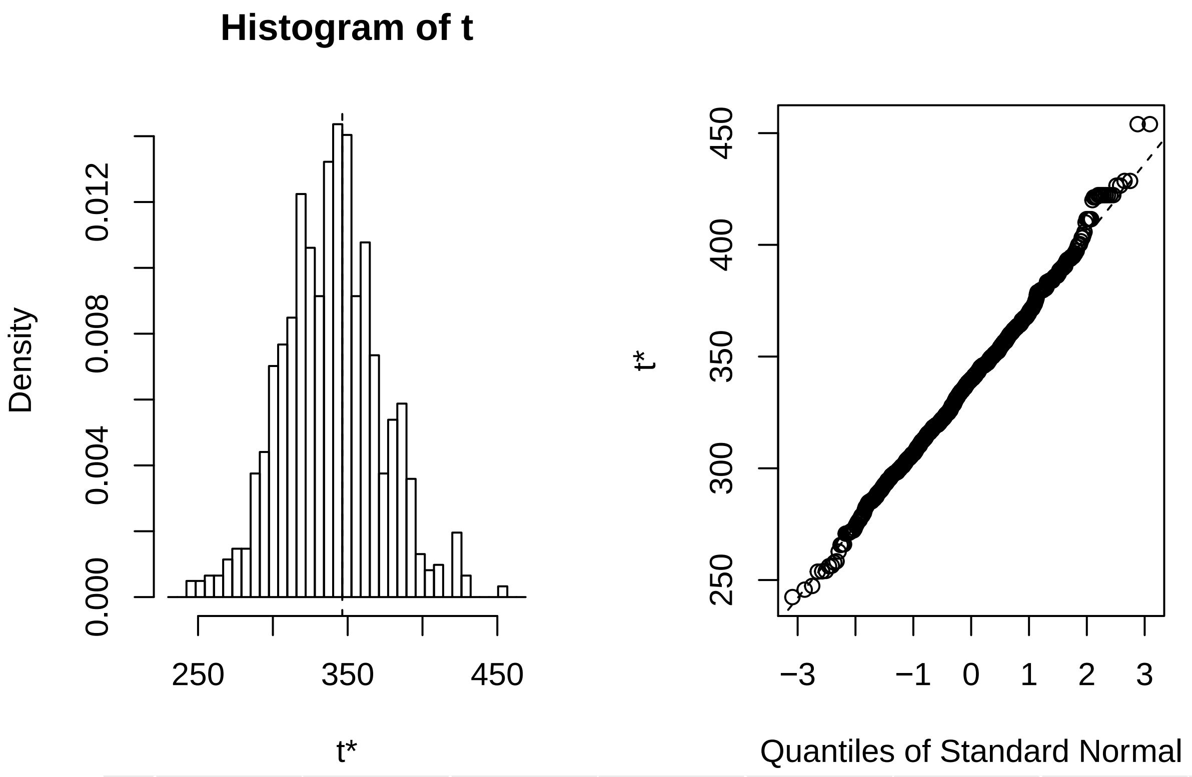
Selected tree is 3 which is selected on the basis of lowest value of deviance

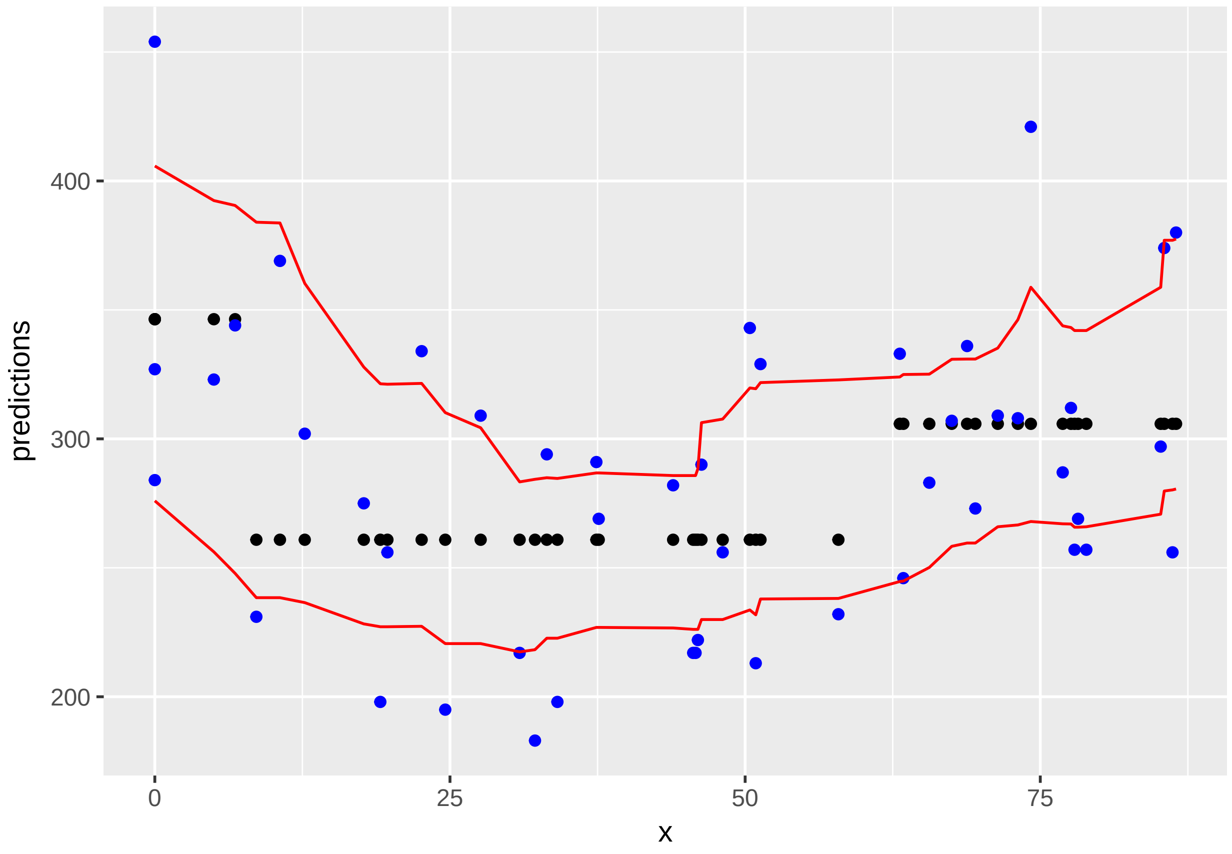




Residuals seems to be marginally distributed across the data. According to the histogram, residuals represents that fitting of data can be improved that is the residuals can be reduced if better fitting is applied.

Part 3

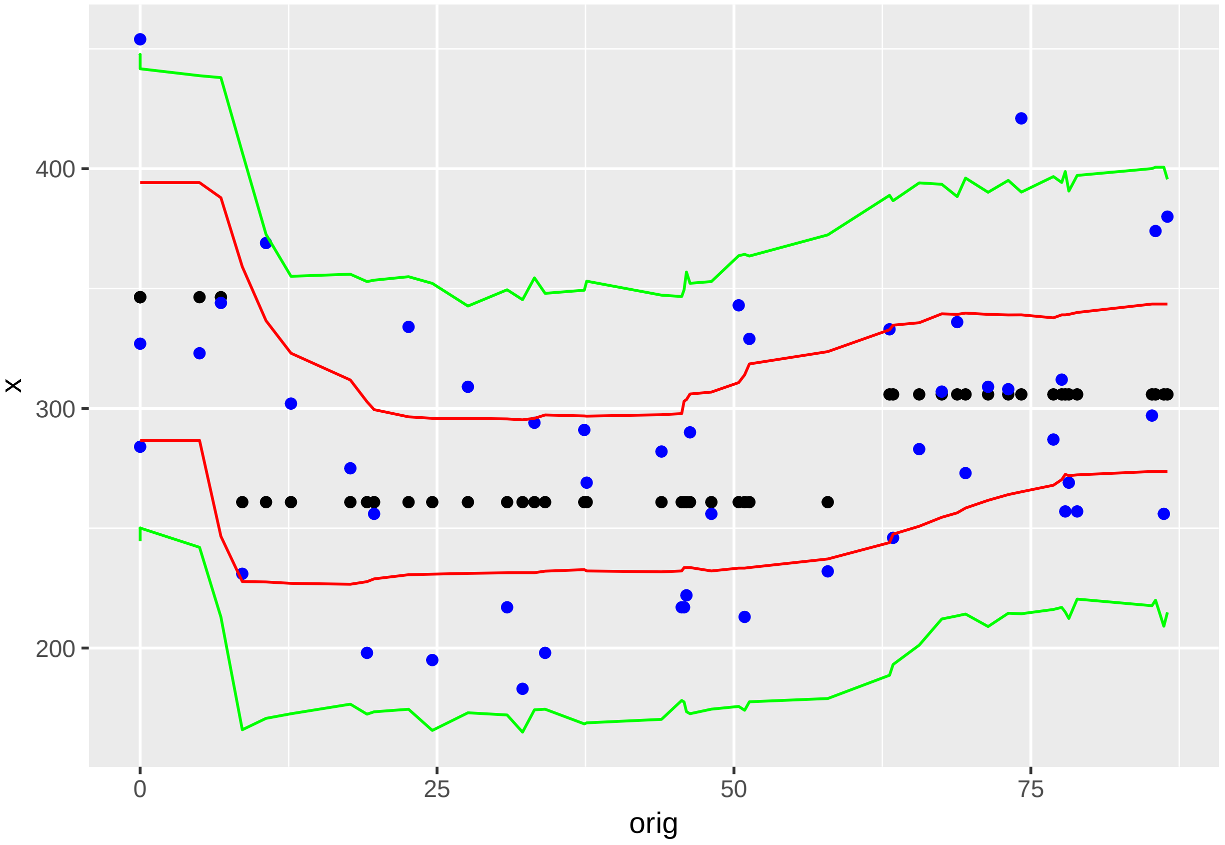




Confidence band is the bootstrap percentile interval which is combination of different confidence intervals computed for different replicates of bootstrap.

The confidence band for the parametric bootstrap is bumpy due to the impact of bias on bootstrap.

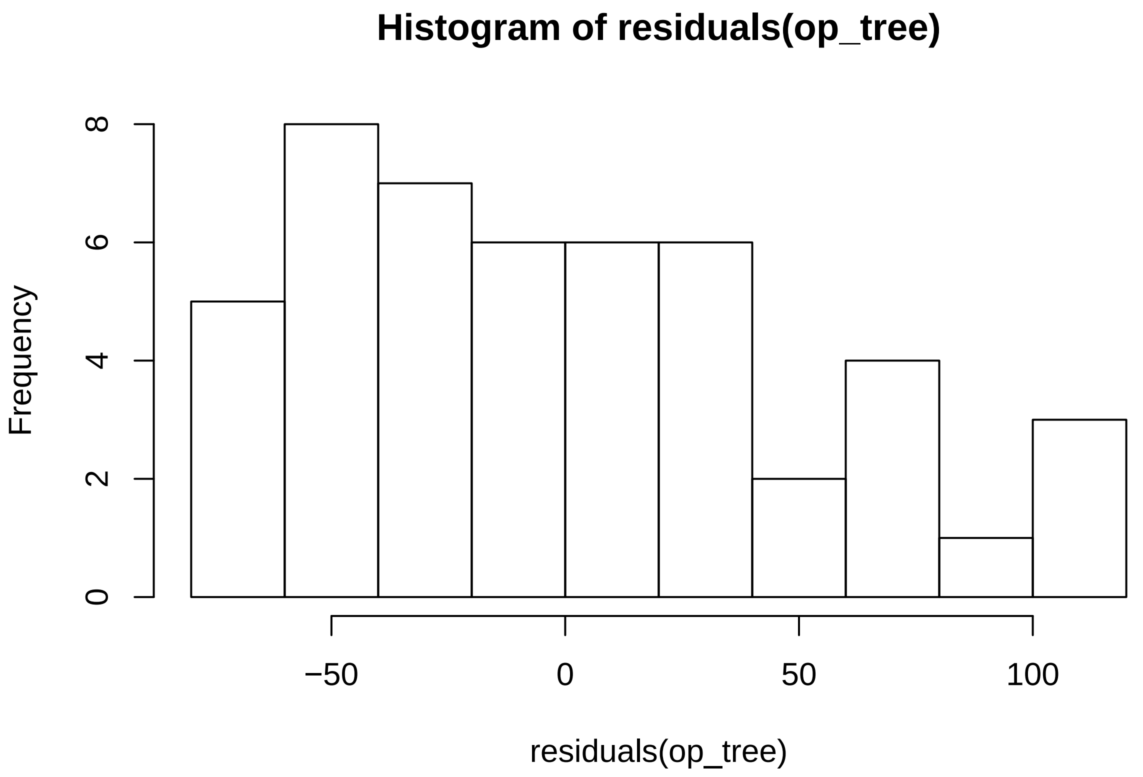
Considering the width of confidence interval, the result of regression model computed in part 2 appears to be reliable as it appears to lie within the confidence band.



The red line in the above plot represents the confidence band.The confidence band seems to enclose the fitted data from step 2 and thus the regression model in step 2 appears to be reliable.

The green lines represent the prediction band. It can be observed that the prediction band encloses almost all the points and only 5% or less data resides outside the prediction band. If the prediction band would enclose all points, it will indicate that there might be overfitting.

Part 5



On the basis of histogram plot from step 2, it can be suggested that the `parametric bootstrap` would be more appropriate model since for the given amount of data, non-parametric bootstrap might gives underfitted values. Moreover, the parametric bootstrap is more reliable if distribution is pre-determined regardless of the size of data.

APPENDIX

```
## Assignment 1

  library(ggplot2)
  data<- read.csv2("australian-crabs.csv" ,sep = ",",dec=".")
  p <- ggplot(data, aes(x=CL, y=RW)) + geom_point(aes(color=sex), size=2 ) +
    scale_color_manual (values =  c('blue', 'red')) +
    labs(x="CL carspace length", y="RW rear Width", colour="Classes") +
    ggtitle("original data")
  X<- data.frame(RW=data$RW , CL=data$CL )
  Y <- data$sex

  #1.2
  library(MASS)
  disc_fun=function(label, S)
  {
    X1=X[Y==label,]
    mean_v <- c(mean(X1$RW) ,mean(X1$CL))
    covaiance_mat_inverse <- solve(S)
    prior_prob <- nrow(X1) / nrow(X)
    w1 <- covaiance_mat_inverse %*% mean_v
    b1 <- ((-1/2) %*% t(mean_v) %*% covaiance_mat_inverse  %*% mean_v) + log(prior_prob)
    w1<- as.vector(w1)

    return(c(w1[1], w1[2], b1[1,1]))
  }

  X1=X[Y=="Male",]
  X2=X[Y=="Female",]

  S=cov(X1)*dim(X1)[1]+cov(X2)*dim(X2)[1]
  S=S/dim(X)[1]

  #discriminant function coefficients
  res1=disc_fun("Male",S)
  res2=disc_fun("Female",S)
  #1.2
  #decision boundary coefficients 'res'
  res <- c( -(res1[1]-res2[1]) , (res2[2]-res1[2]), (res2[3]-res1[3]))

  # classification
  d=res[1]*X[,1]+res[2]*X[,2]+res[3]
  Yfit=(d>0)
  plot(X[,1], X[,2], col=Yfit+1, xlab="CL", ylab="RW")

  #slope and intercept
  slope <- (res[2] / res[1] ) * -1
  intercept <- res[3] /res[1] * -1

  #1.3
  #plot decision boundary
  X<- cbind(X,sex=Y)
  p <- ggplot(X, aes(x=CL, y=RW)) + geom_point(aes(color=sex), size=2 ) +
    scale_color_manual (values =  c('blue', 'red')) +
    labs(x="CL carspace length", y="RW rear Width", colour="Classes") +
    geom_abline(slope = slope, intercept = intercept) +
    ggtitle("Descion Boundary LDA")

  #1.4
  logistic_fit <- glm(sex ~ RW + CL, data= X,
                      family='binomial') #glm fitting
  r.unlist <- as.numeric(unlist(logistic_fit$coefficients))
  r.unlist <- data.frame(intercept=r[1] ,
                         RW=r[2],
                         CL=r[3]) #convert data into datafreame
  glm_intercept <- (-r$intercept/r$RW) #+ (0.5/r$RW)
  glm_slope <- (-r$CL / r$RW ) #+ (0.5) / r$RW

  p <- ggplot(X, aes(x=CL, y=RW)) + geom_point(aes(color=sex), size=2 ) +
    scale_color_manual (values =  c('blue', 'red')) +
    labs(x="CL carspace length", y="RW rear Width", colour="Classes") +
    geom_abline(slope = glm_slope,
                intercept = glm_intercept) +
    ggtitle("Logistic Regression
            decision Boundary")

  ## Assignment 2

library(readxl)
library(tree)
library(e1071)

# Importing Data - 2.1
data <- read_excel("creditscoring.xls")
data <- as.data.frame(data)
data$good_bad <- as.factor(data$good_bad)

# Dividing Data into three Train(50%) Test(25%) Validation(25%)
n = nrow(data)
set.seed(12345)

n=dim(data)[1]

# 50% Training Data
id=sample(1:n, floor(n*0.5))
train=data[id,]

# 25% validation & testing Data
Sub_id = data[-id,]
m = dim(Sub_id)[1]
part1 = sample(1:m, floor(m*0.5))
validation = Sub_id[part1,]
testing = Sub_id[-part1,]


# Step 2

# Fitting data using Deviance and gini
tree_deviance = tree(as.factor(good_bad) ~ ., data = train, split = "deviance")
tree_gini = tree(as.factor(good_bad) ~ ., data = train, split = "gini")

# Prediction

## Misclassification for training data
devi_yfit = predict(tree_deviance, newdata = train,type="class")
gini_yfit = predict(tree_gini, newdata = train,type="class")

plot(tree_deviance)
plot(tree_gini)


devi_table = table(devi_yfit,train$good_bad)
gini_table = table(gini_yfit,train$good_bad)

devi_table
```

```r
# Missclassification rate Deviance
missclass_devi <- 1-sum(diag(devi_table))/sum(devi_table)
missclass_devi

gini_table
# Missclassification rate Gini
missclass_gini <- 1-sum(diag(gini_table))/sum(gini_table)
missclass_gini


## Misclssification for test data:
devi_yfit = predict(tree_deviance, newdata = testing,type="class")
gini_yfit = predict(tree_gini, newdata = testing,type="class")
plot(tree_deviance)
plot(tree_gini)


devi_table = table(devi_yfit,testing$good_bad)
gini_table = table(gini_yfit,testing$good_bad)

devi_table
# Missclassification rate Deviance
missclass_devi <- 1-sum(diag(devi_table))/sum(devi_table)
missclass_devi

gini_table
# Missclassification rate Gini
missclass_gini <- 1-sum(diag(gini_table))/sum(gini_table)
missclass_gini

### Step 3

index = summary(tree_deviance)[4]$size
trainScore = rep(0,index)
testScore = rep(0,index)

# Graph training and validation

for(i in 2:index) {
  prunedTree=prune.tree(tree_deviance,best=i)
  pred=predict(prunedTree, newdata=validation,type="tree")
  trainScore[i]=deviance(prunedTree)
  testScore[i]=deviance(pred)
}

plot(2:index,trainScore[2:index], col="Red",type = "b", main = "Dependence of Deviance",
     ylim=c(min(testScore[2:index]),max(trainScore)), pch=19, cex=1, ylab="Deviance")
points(2:index,testScore[2:index],col="Blue",type="b", pch=19, cex=1)

# misclassification rate for test data

missclass_test_t = prune.tree(tree_deviance, best = 4)
summary(missclass_test_t)
```

```
yfit = predict(missclass_test_t, newdata = testing, type="class")
valid_ = table(testing$good_bad,yfit)
print("Confusion Matrix")
valid_
mc <- 1-sum(diag(valid_))/sum(valid_)
print("Misclassification rate")
mc
plot(missclass_test_t)
text(missclass_test_t)

### Step 4

# Na?ve Bayes 2.4
naye = naiveBayes(good_bad ~., data=train)

nav_test = predict(naye, newdata = testing[,-ncol(testing)], type = "class")
nav_train = predict(naye,newdata = train[,-ncol(train)])

# Confusion Matrix Using Naive Bayes
nv_tbl_test = table(testing$good_bad,nav_test)
print(nv_tbl_test)
nv_tbl_train <- table(train$good_bad,nav_train)
print(nv_tbl_train)

# Missclassification train data value Using Naive Bayes
mc_nav_train <- 1-sum(diag(nv_tbl_train))/sum(nv_tbl_train)
cat("Misclassification train data value Using Naive Bayes is:",mc_nav_train)

# Missclassification test data value Using Naive Bayes
mc_nav_test <- 1-sum(diag(nv_tbl_test))/sum(nv_tbl_test)
cat("Misclassification test data value Using Naive Bayes is:",mc_nav_test)

### Step 5

# Naive Bayes With loss matrix 2.5
naye = naiveBayes(good_bad ~ ., data = train)

# Predicting using Naive
nav_test = predict(naye, testing[,-ncol(testing)] , type="raw")
nav_train = predict(naye, train[,-ncol(train)] , type="raw")

# applying loss matrix if greater then 10 True else False
nav_test =  (nav_test[, 2] / nav_test[, 1]) > 10
nav_train =  (nav_train[, 2] / nav_train[, 1]) > 10

# confusion matrix for train & test
naive_table = table(testing$good_bad,nav_test)
naive_table_train = table(train$good_bad,nav_train)

# missclasification for train & test
naive_table_train
1-sum(diag(naive_table_train))/sum(naive_table_train)
```

```r
naive_table
1-sum(diag(naive_table))/sum(naive_table)


# Assignment 3

library(tree)

# 3.1 Data import, reorder and Plot
set.seed(12345)

data = read.csv2("State.csv", header = TRUE)
data = data[order(data$MET),]
plot(EX ~ MET, data = data, pch = 19, cex = 1,col="blue")

# 3.2
set.seed(12345)

control_parameter = tree.control(nobs = nrow(data),minsize = 8)

fit_tree = tree(formula = EX ~ MET,data = data,control = control_parameter)
leave_fit = cv.tree(fit_tree)
plot(leave_fit$size, leave_fit$dev, main = "Deviance Vs Size of Tree" ,
     xlab="Deviance", ylab = "Size", type="b",col="red", pch= 19,cex=1)


op_tree = prune.tree(fit_tree,best = leave_fit$size[which.min(leave_fit$dev)])

fitted_val = predict(op_tree, newdata=data)

df = data.frame(x = data$MET, pred = fitted_val, ex_val = data$EX)
fit_original_plot = ggplot(df, aes(x, pred, ex_val)) +
  geom_point(aes(x,ex_val), colour = "blue") +
  geom_point(aes(x, pred))
fit_original_plot

hist(residuals(op_tree))


# 3.3 Non-Paramatric Bootstrap
library(boot)


f_np = function(data,index){
  sample = data[index,]
  Ctrl = tree.control(nrow(sample), minsize = 8)
  fit = tree( EX ~ MET, data=sample, control = Ctrl)
  optimal_tree = prune.tree(fit, best= leave_fit$size[which.min(leave_fit$dev)])
  return(predict(optimal_tree, newdata=data))
}

np_bs = boot(data, statistic = f_np, R=1000)
conf_bound = envelope(np_bs,level=0.95)
```

```
predictions = predict(op_tree,data)

plot(np_bs)

fig_data = data.frame(orig = data$EX, x=data$MET, pred=predictions,
                      upper=conf_bound$point[1,], lower=conf_bound$point[2,])
fig = ggplot(fig_data, aes(x,predictions,upper,lower))
p = fig + geom_point(aes(x, pred)) +
  geom_point(aes(x, orig),colour="blue") +
  geom_line(aes(x,upper),colour="red") +
  geom_line(aes(x,lower),colour="red")
p


# 3.4 Paramatric Bootstrap
set.seed(12345)

parama_conf = function(data){
  controll = tree.control(nrow(data), minsize = 8)
  fit = tree( EX ~ MET, data=data, control = controll)
  op_tree = prune.tree(fit, best=leave_fit$size[which.min(leave_fit$dev)])
  return(predict(op_tree, newdata=data))
}

param_predict = function(data){
  controll = tree.control(nrow(data), minsize = 8)
  fit = tree( EX ~ MET, data=data, control = controll)
  op_tree = prune.tree(fit, best=leave_fit$size[which.min(leave_fit$dev)])
  predictions = predict(op_tree, newdata=data)
  return(rnorm(nrow(data),predictions,sd(resid(fit))))
}

rnd = function(data, model){
  sample = data.frame(MET=data$MET, EX=data$EX)
  sample$EX = rnorm(nrow(data), predict(model,newdata=data),sd(resid(model)))
  return(sample)
}

set.seed(12345)
param_boot_conf = boot(data, statistic = parama_conf, R=1000, mle = op_tree,
                       ran.gen = rnd, sim = "parametric")
confidence_bound_param = envelope(param_boot_conf, level=0.95)

param_boot_predict = boot(data, statistic = param_predict, R=1000, mle = op_tree, ran.gen = rnd, sim =
prediction_bound_param = envelope(param_boot_predict, level=0.95)

plot(param_boot_conf)
plot(param_boot_predict)

predictions = predict(op_tree,data)
fig_data = data.frame(orig = data$EX, x=data$MET, pred=predictions,
                      upper_c=confidence_bound_param$point[1,],
                      lower_c=confidence_bound_param$point[2,],
```

```r
                      upper_p=prediction_bound_param$point[1,],
                      lower_p=prediction_bound_param$point[2,])

para_plot = ggplot(fig_data, aes(orig,x,pred,upper_c,lower_c, upper_p, lower_p))
para_plot = para_plot +
  geom_point(aes(x, pred)) +
  geom_point(aes(x, orig),colour="blue") +
  geom_line(aes(x,upper_c),colour="red") +
  geom_line(aes(x,lower_c),colour="red") +
  geom_line(aes(x,upper_p),colour="green")+
  geom_line(aes(x,lower_p),colour="green")
para_plot
```