Computer lab 2

Martynas Lukosevicius, Alejo Perez Gomez, Shwetha Vandagadde Chandramouly

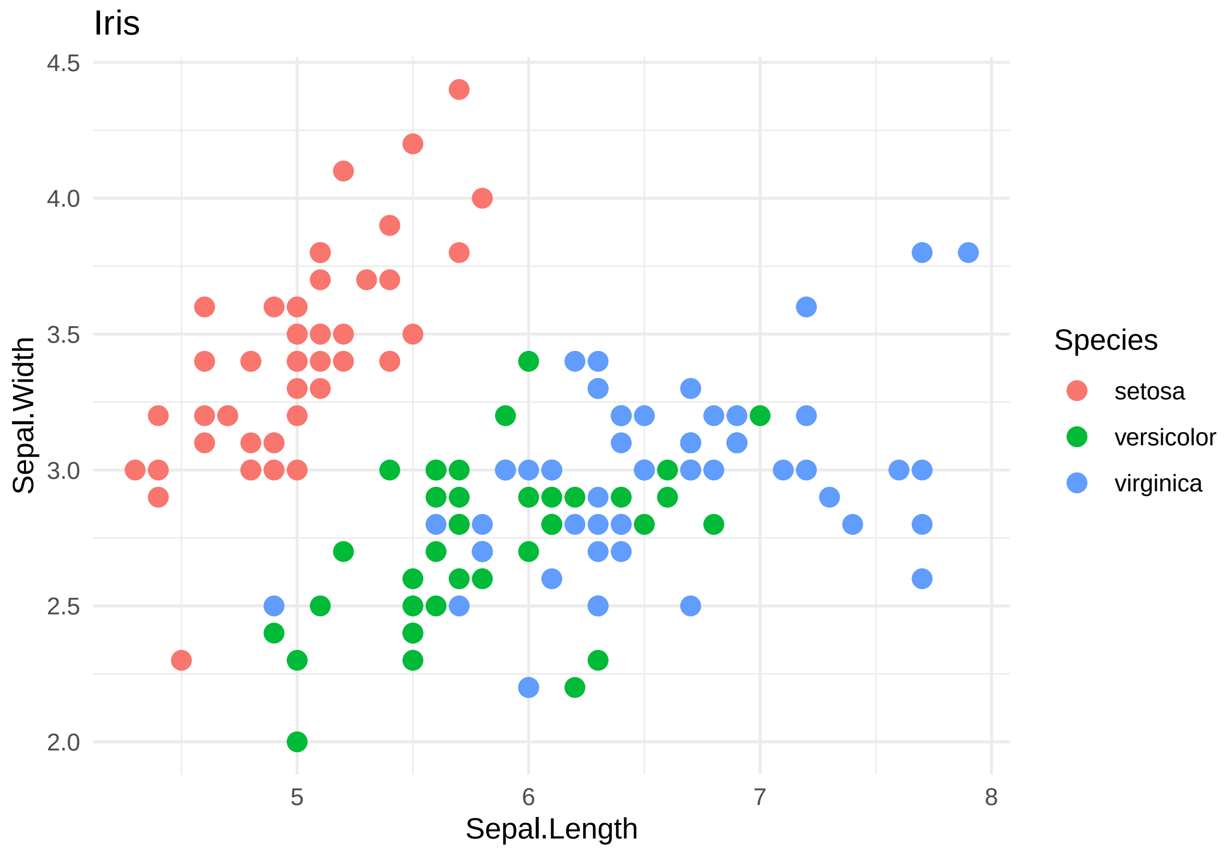
06/12/2020

Statement of Contribution

- Assignment 1 - Martynas Lukosevicius
- Assignment 2 - Shwetha Vandagadde Chandramouly
- Assignment 3 - Alejo Perez Gomez

Assignment 1. LDA and logistic regression

1.



It is not easy to classify by LDA because as we can see from scatter plot versicolor overlay virginica. I expect that misclasification rate will be high because part of the virginica data will be predicted as versicolor and vice versa. We can see that setosa was classified correctly

2.

a)

Setosa: mean - $\begin{bmatrix}5.006\\3.428\end{bmatrix}$, covariance - $\begin{bmatrix}0.124 & 0.099\\0.099 & 0.144\end{bmatrix}$, $\pi_{setosa} - 0.3333333$

Virginica: mean - $\begin{bmatrix}6.588\\2.974\end{bmatrix}$, covariance - $\begin{bmatrix}0.404 & 0.094\\0.094 & 0.104\end{bmatrix}$, $\pi_{virginica} - 0.3333333$

Versicolor: mean - $\begin{bmatrix}5.936\\2.77\end{bmatrix}$, covariance - $\begin{bmatrix}0.266 & 0.085\\0.085 & 0.098\end{bmatrix}$, $\pi_{versicolor} - 0.3333333$

b)

Pooled covariance - $\begin{bmatrix}0.219 & 0.09\\0.09 & 0.114\end{bmatrix}$

c)

Probabilistic model for LDA:

$$P(y = C_i|X, w) \propto P(X|Y = C_i, w)P(Y = C_i|w)$$
$$P(X|Y = C_i, w) \sim N(\mu_i, \Sigma)$$
$$P(Y = C_i|w) = \pi_i$$

$$P(y = C_i|X, w) \propto exp[(\Sigma^{-1}\mu_i)^T X - \frac{1}{2}\mu_i^T \Sigma^{-1}\mu_i + log(\pi_i)] = exp[w_i X + w_{0i}]$$

Where $w_i - (\Sigma^{-1}\mu_i)^T$ and $w_{0i} - -\frac{1}{2}\mu_i^T\Sigma^{-1}\mu_i + log(\pi_i)$

d)

discriminant function $\delta_k(x) = x^T\Sigma^{-1}\mu_k - \frac{1}{2}\mu_k^T + log(\pi_k)$

Setosa:

$$\delta_{setosa}(x) = x^T \begin{bmatrix}15.50155\\17.92285\end{bmatrix} - 70.61876$$

Versicolor:

$$\delta_{Versicolor}(x) = x^T \begin{bmatrix}25.309346\\4.365498\end{bmatrix} - 82.26297$$

Virginica:

$$\delta_{Virginica}(x) = x^T \begin{bmatrix}28.625671\\3.537537\end{bmatrix} - 100.6519$$

```r
discrim <- function(x,a){
  constant <- (-1/2) * t(a$mean) %*% solve(pcov) %*% a$mean + log(a$prior)
  nonconstant <- t(x) %*% solve(pcov) %*% a$mean
  return(nonconstant+constant)
}
```

e)

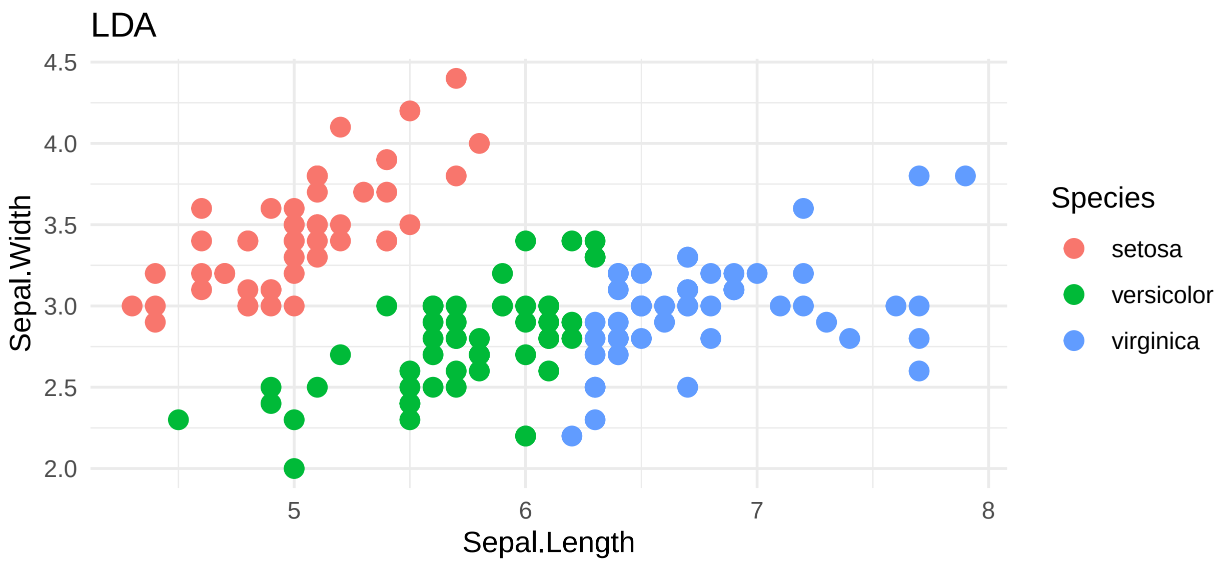
$$(w_k - w_i)x + (w_{0k} - w_{0i}) = 0$$

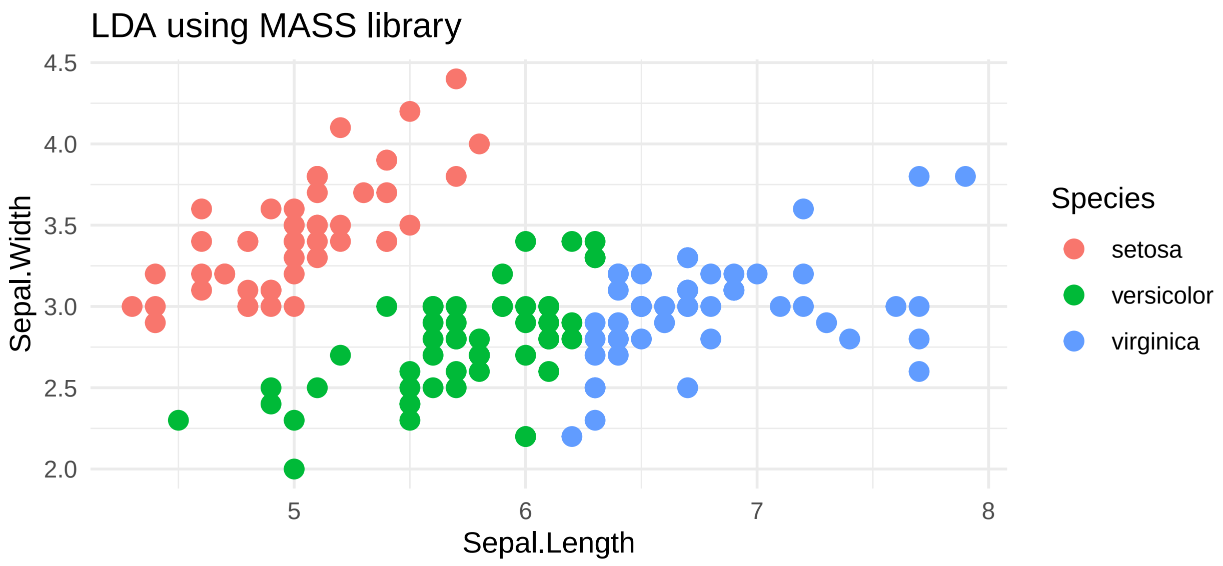
decision boundaries:

- Setosa - Versicolor: $\begin{pmatrix} -9.8077917 \\ 13.5573492 \end{pmatrix} x + (-11.6442029) = 0$
- Virginica - Versicolor: $\begin{pmatrix} 3.3163254 \\ -0.827961 \end{pmatrix} x + (18.3889254) = 0$
- Setosa - Virginica: $\begin{pmatrix} -13.124117 \\ 14.3853102 \end{pmatrix} x + (-30.0331283) = 0$

LDA assume that $\Sigma_i = \Sigma$, However it is not the case in this situation

3.



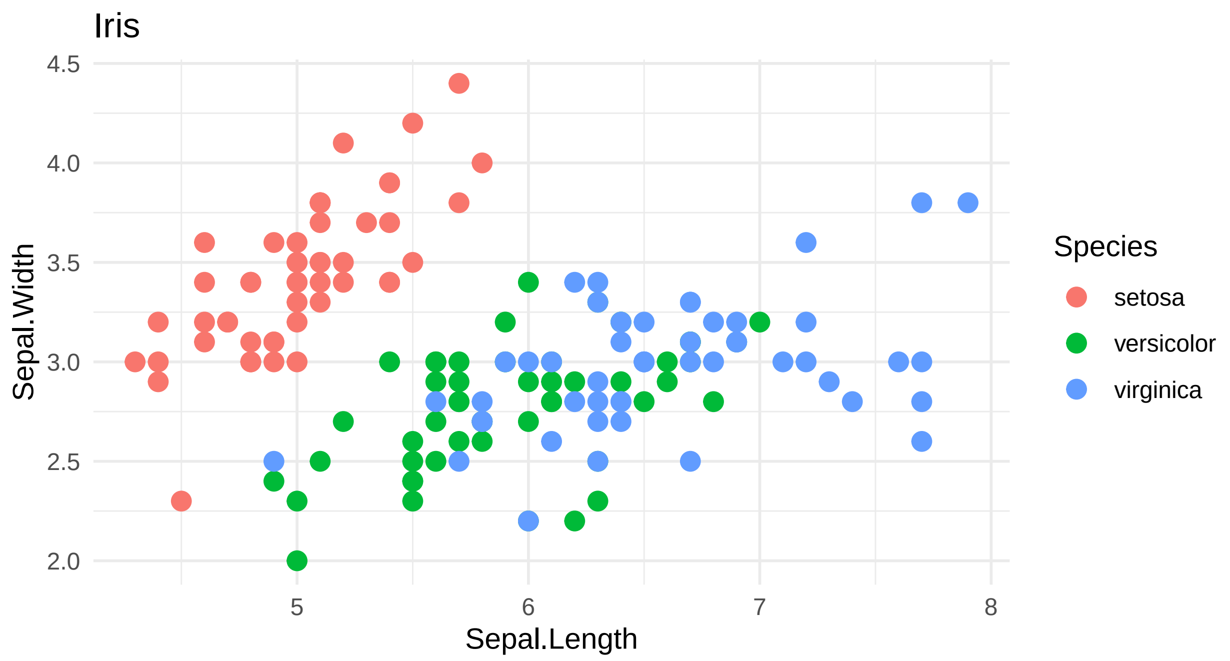


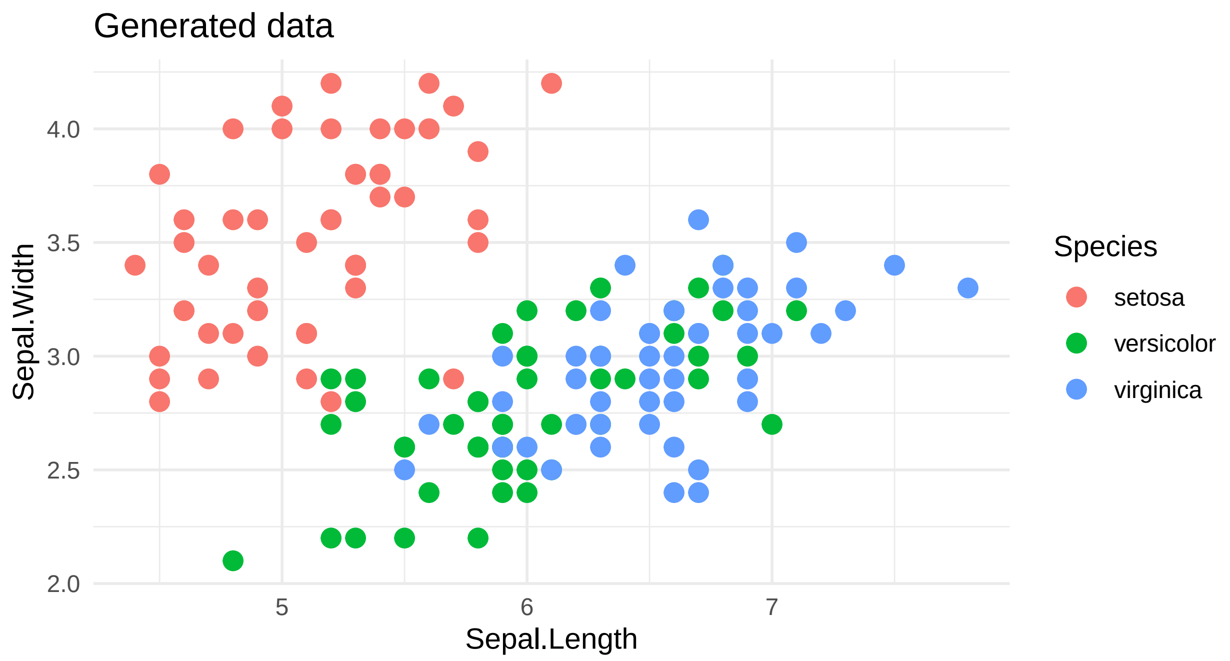
Missclasification rate of LDA: 0.2

Missclasification rate of LDA using MASS library: 0.2

Test errors are the same, Classification methods are identical so and the results are identical

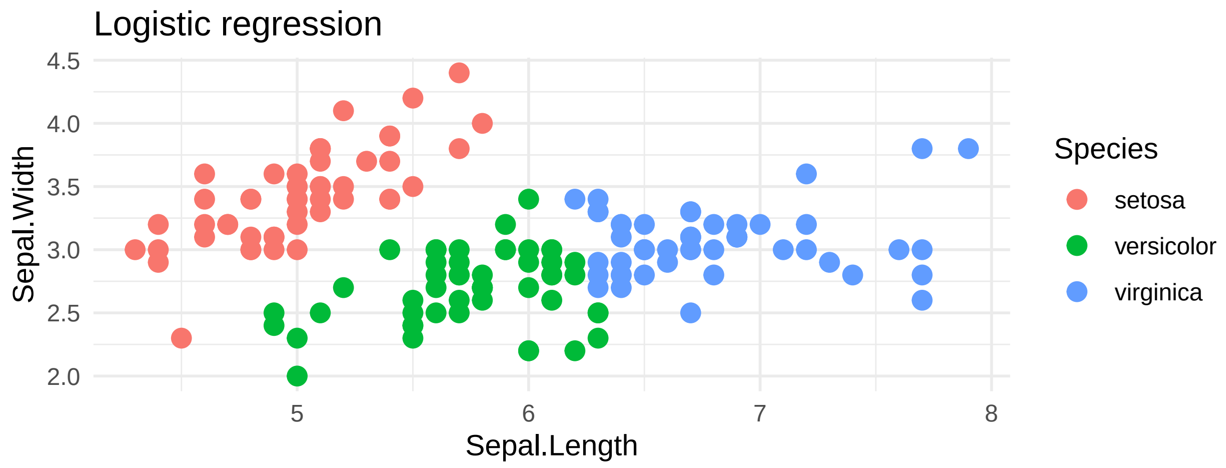
4.

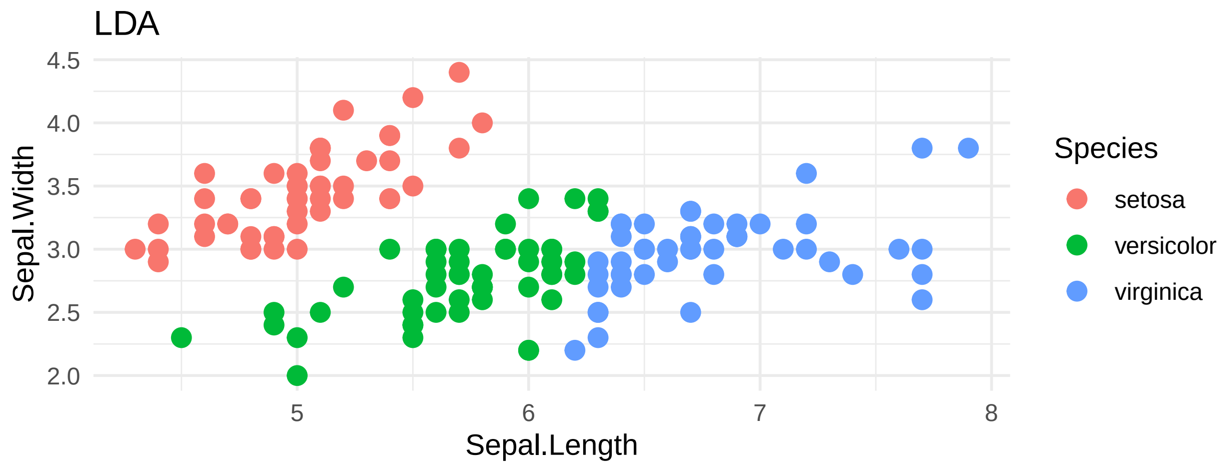




From Plots we can see generated data is spread equally, it is because of LDA assumption that covariances are equal.

5.





	Misclasification rate
LDA	0.2000000
LR	0.1666667

From misclasification rate we can see that logistic regresion performed slightly better than LRA. however it might have overfitted

Assignment 2. Decision trees and Naïve Bayes for bank marketing

1.

Partitioning the data into train , test and validation.

```r
bank = read.csv("bank-full.csv", header = TRUE, sep = ";",stringsAsFactors = TRUE)

bank = bank[,-12]
n = nrow(bank)
set.seed(12345)
id1=sample(1:n, floor(n*0.4))
train=bank[id1,]
```

```
d2 = bank[-id1,]
n2 = nrow(d2)
id2=sample(1:n2, floor(n2*0.5))
test=d2[id2,]
validate=d2[-id2,]
```

2.

Fitting decision tree to training data

```
library(tree)
dt_default = tree(y~.,data = train)
dt_size = tree(y~.,data = train, control = tree.control(nrow(train),minsize = 7000))
dt_dev = tree(y~.,data = train, control = tree.control(nrow(train),mindev = 0.0005))
```

Training data missclassification rate:

Default fit : 0.1048441

Min node size is 7000 fit : 0.1048441

Min deviance is .0005 fit : 0.0936187

Validation data missclassification rate:

Default fit : 0.1116927

Min node size is 7000 fit : 0.1116927

Min deviance is .0005 fit : 0.112946

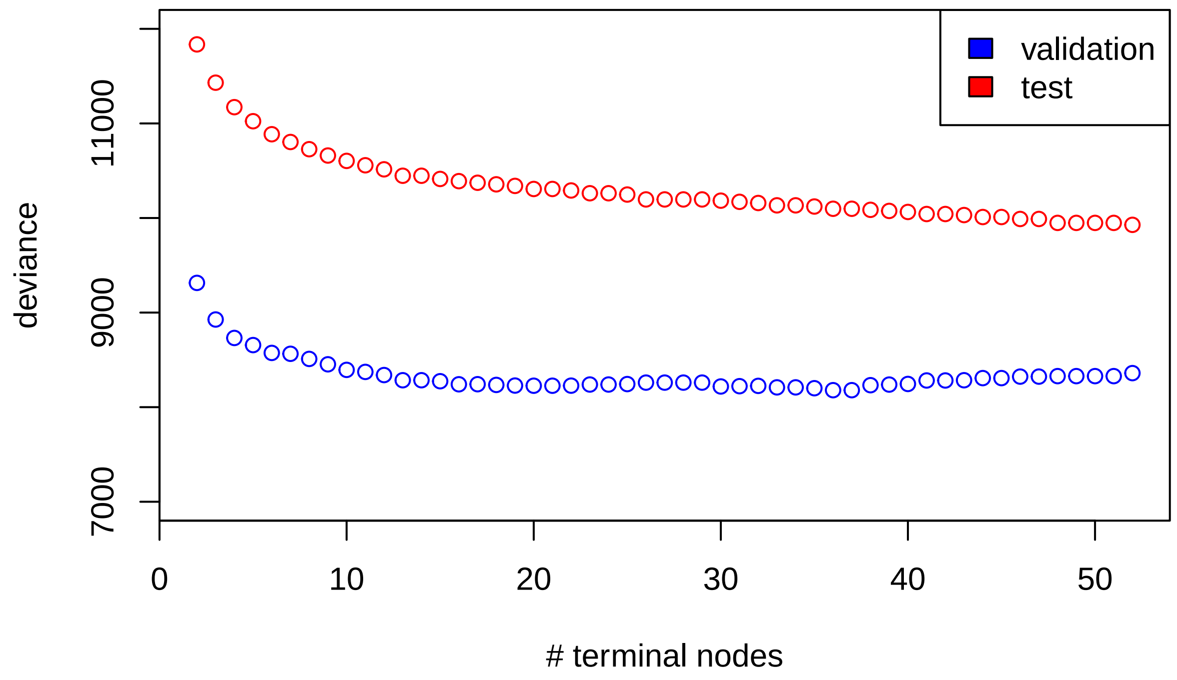
Choosing the best of three fits : Though the misclassification error was least for mindev=0.0005 fit on train data ,it has large tree of 150 terminal nodes, this leads to a overfit tree. As a result of this , on fitting the validation data , the misclassification error rate increases more in this when compared to other two trees. Model a and b , both have same misclassification error , however since the "b" has only 5 terminal nodes whereas "a" has 6 terminal nodes , its better to choose the simpler model ie b. But if we are allowed to find the optimal number of leaves to avoid overfitting, then c would be the best model.

In our case , setting the deviance very small ie 0.0005, made the tree grow more deeper as a large tree with 150 terminal nodes , this did reduce the misclassification on the training data, but ended up overfitting for validation data.

Effect of deviance and nodesize on the tree size : Decreasing the deviance leads to increase in tree size. Increase in the nodesize leads to decrease in the tree size.

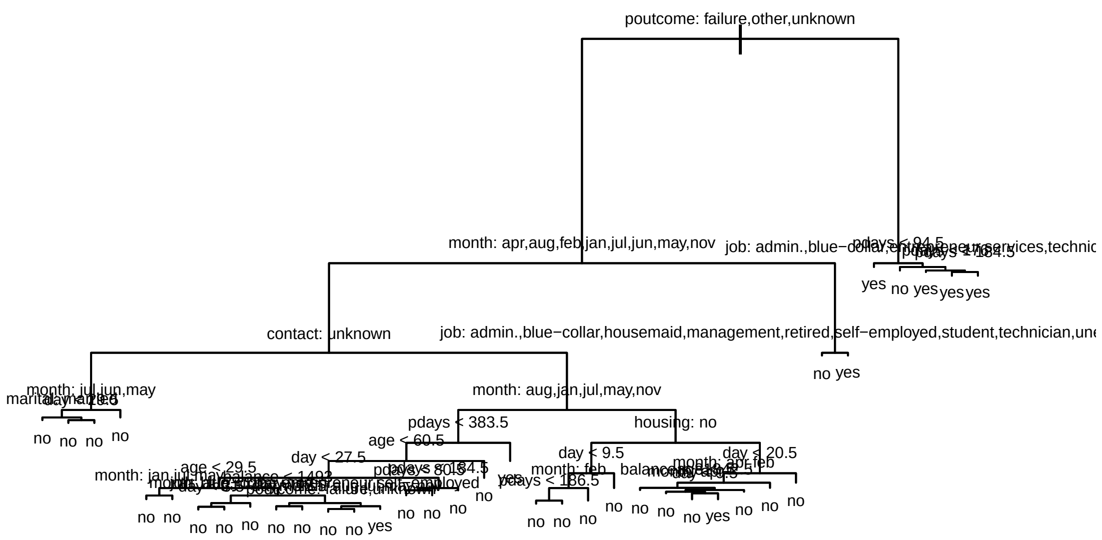
3.

Selecting optimal tree by training and validation



Optimal amount of leaves = 36

```
optimal_tree = prune.tree(dt_dev, best = 36)
plot(optimal_tree)
text(optimal_tree, pretty = 0, cex = 0.5)
```


poutcome: failure,other,unknown
month: apr,aug,feb,jan,jul,jun,may,nov
job: admin.,blue−collar,entrepreneur,services,technic
pdays < 94.5
pdays < 184.5
yes
no
yes
yes
yes
contact: unknown
job: admin.,blue−collar,housemaid,management,retired,self−employed,student,technician,une
no
yes
month: jul,jun,may
month: aug,jan,jul,may,nov
pdays < 383.5
housing: no
age < 60.5
day < 27.5
age < 29.5
day < 9.5
day < 20.5
month: feb
month: apr,feb
pdays < 186.5
month: jan,jul,may
balance < 1492
no
yes

Variable : poutcome is the most important for decision making in this tree.

Tree sturcture : variable included = poutcome, month, contact, marital, day, pdays, age, balance, job and housing Number of leaves = 37 First variable considered in the tree is "poutcome", the partition is made by taking the condition poutcome = failure, other or unknown, if this is true , the data goes to left side to the next condition on variable month. If the poutcome was not satisfied , ie say poutcome was "success" then the data flows to right part , where next node condition is on pdays < 94.5. The tree continues until the number of leaves are 37.

Confusion matrix and missclassification rate for test data.

confusion matrix :

	no	yes
no	11868	122
yes	1347	226

Missclassification rate : 0.1083094

we can see that the misclassification rate for the test data has reduced for the optimal tree. Hence this is a better fit compared to the previously tried tree fits.

4.

Loss Matrix :

	no	yes
no	0	1
yes	5	0

Confusion matrix :

	no	yes
no	10965	1025
yes	745	828

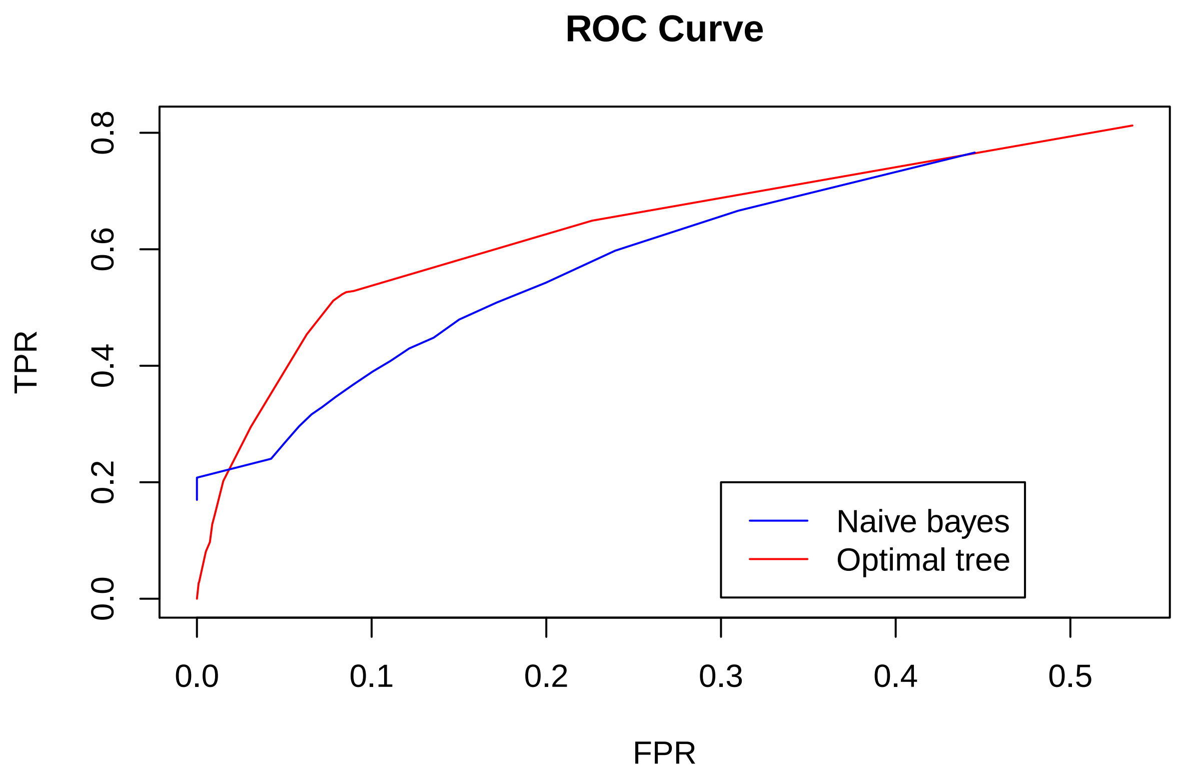
Missclassification rate : 0.1305021

Here in the loss function we can see that , penalty for predicting observed yes as no is 5 and no as yes is 1. So on applying loss function , as expected , the misclassification of an observed yes as no is reduced in the confusion matrix here. Previously observed yes predicted as no was 1347 , and now it is 745. However the missclassification error rate has increased.

We can try different loss matrix (assigning the loss function with suitable costs for the respective senario) and choose the one which gives the lowest misclassification rate.

5.

Fitting naive bayes model, computing TPR and FPR for both models and plotting the ROC curve



Conclusion : Area under the curve is more for Optimal tree fit hence this is the best classifier.

Assignment 3

1.

First we will scale all the variables and implement the PCA. We will not include the variable State as it is categorical data and we will only apply scaling and PCA on continuous variables.

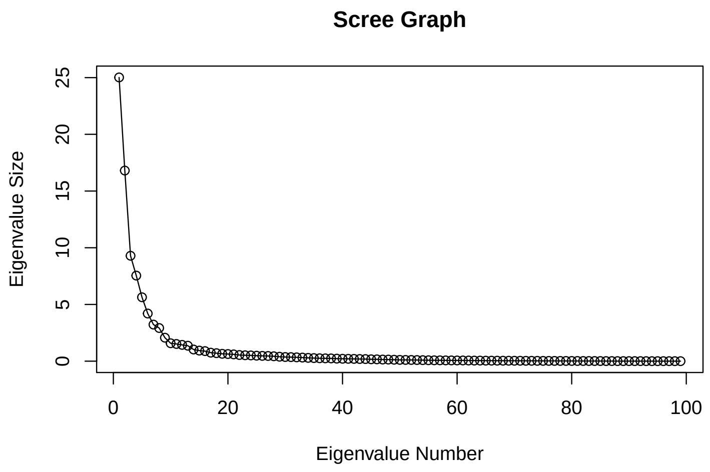
```r
## Load data and convert to numeric
data_com <- sapply(data_com, as.numeric)

#Scaling features excluding state and target
X <- scale(data_com[,c(-1, -101)])
cov_mat <- cov(X)

cov_mat_eigen <- eigen(cov_mat)

#proportion of variance explained for each component
proportion_variation <- cov_mat_eigen$values/sum(cov_mat_eigen$values)

plot(cov_mat_eigen$values, xlab = 'Eigenvalue Number', ylab = 'Eigenvalue Size',
     main = 'Scree Graph')
lines(cov_mat_eigen$values)
```



The percentage of the variance accounted by the first two principal components is 42.2%. We calculated that out of summing the two first terms in our vector of variance explanations, dividing eigen values by total variance.

To achieve an explanation of the variance of 0.955%, 35 PC are needed.

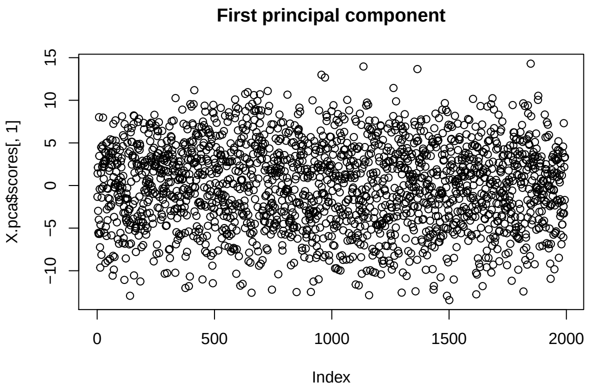
2.

In this section PCA will be used but with function `princomp()`

```r
X.pca <- princomp(X)
```

Now the plot for the first principal component will be shown.

```r
plot(X.pca$scores[,1], main="First principal component" )
```



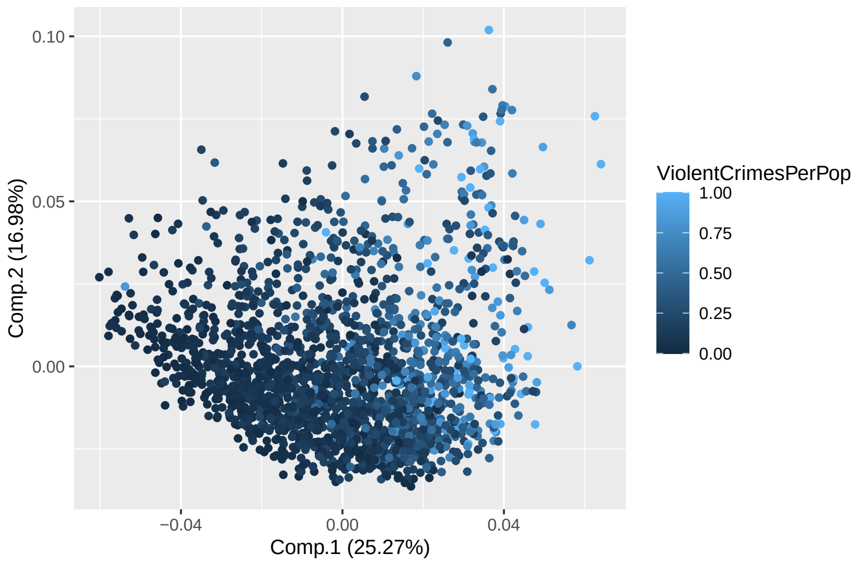
There are a total of 13 features that contribute notably to the first PC (using a threshold of 0.15) are the following (in absolute value).

	Value of contribution
medFamInc	0.1832453
medIncome	0.1819115
PctKids2Par	0.1755956
pctWInvInc	0.1749060
PctPopUnderPov	0.1738039
PctFam2Par	0.1727358
PctYoungKids2Par	0.1716485
perCapInc	0.1694056
pctWPubAsst	0.1647161
PctHousNoPhone	0.1640624
PctNotHSGrad	0.1619471
PctUnemployed	0.1587331
PctTeen2Par	0.1515750

The 5 most contributing features in regards to the first PC are **median family income, median household income, percentage of kids in family housing with two parents, percentage of households with investment and percentage of people under the poverty level**. These features can be arguably related to the crime level in communities. We could assume a correlation between these variables and criminality. It can be likely then, that an individual that happen to commit a crime registers poor values in terms of some of the aforementioned features disregarding a cause-effect approach. In addition, that could lead us to think that those variables can connected when relating them to criminality.

Here is presented a plot of 2 fist PCA colored according the violent crimes per population. As can be seen, we could separate observations in clusters if we establish a threshold for the criminality rate.

PC2 vs PC1



3.

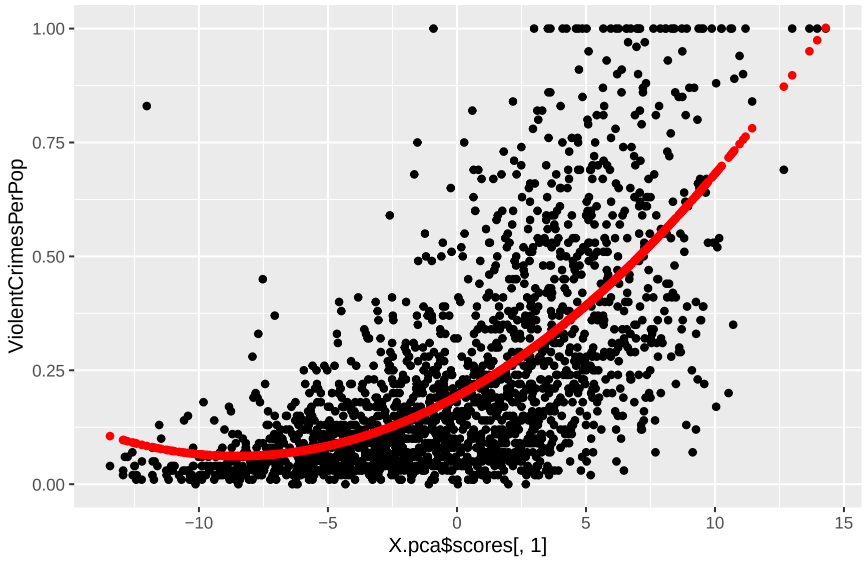
In this section, a quadratic linear model is to be applied using the violent crime rate as a target and PC1 as feature.

```r
X_data <- as.data.frame(cbind(X, data_com[,101]))
colnames(X_data)[100] <- "ViolentCrimesPerPop"

model = lm(ViolentCrimesPerPop ~ poly(X.pca$scores[,1], degree = 2), data= X_data)
```

A plot is shown down below presenting the observations scattered in black (PC1 vs Violent Crimes per Population) and the predictions in red. It is visible how the model has been able to capture the underlying relationship between two variables emulating the curve inside the cloud of points.

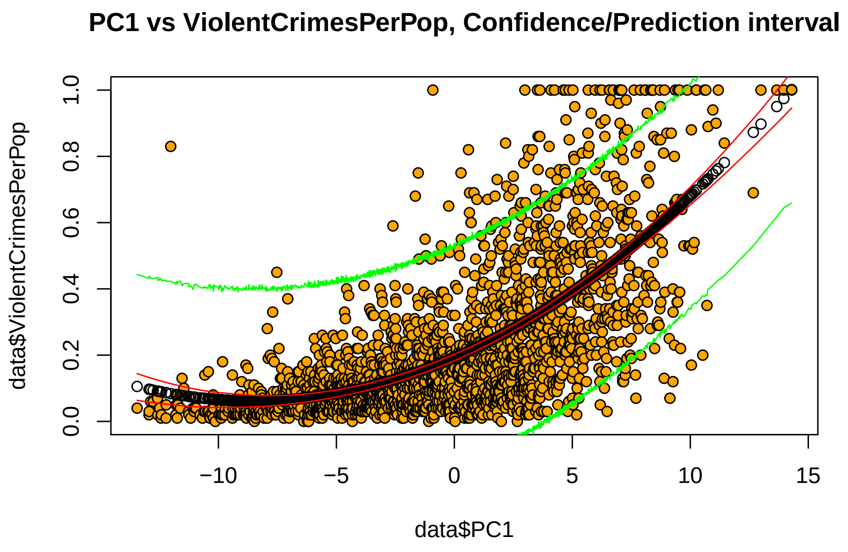
Violent Crimes per Population vs PC1



4.

Parametric Bootstrap will be used in this point to estimate the confidence and prediction bands of the model in section 3. The prediction interval is a type of confidence interval used for predictions in regression procedures. It represents the range of values that predicts the value of a new observation, based on our existing model, where our predicted value is expected to fall. Meanwhile, the confidence interval is an estimated range of values based on what is observed centered in a sampled parameter.

As for the prediction interval (green), the plotted lines envelop and border the cloud of point observations. The plot of confidence interval, in turn, shows lines that emulate the curve of the fitted predictions as well, approaching it far closer, constraining the limits of possible predictions with this quadratic model. The lines (red) are closer to the predicted values in middle range of the slope, whereas as ViolentCrimesPerPop increases or decreases.



Appendix

```
knitr::opts_chunk$set(echo = TRUE)
library(datasets)
y <- iris

library(ggplot2)
plotiris <- ggplot(iris, aes(x=Sepal.Length, y=Sepal.Width, color=Species)) +
  geom_point(size=3) + theme_minimal() + ggtitle("Iris")
plotiris

a2 <- function(x){
y1 <- iris[as.numeric(iris$Species) == x, ][1:2]
mean1 <- apply(y1, 2,mean)
covar <- cov(y1)
prior <- dim(y1)[1]/ dim(iris)[1]

return(list(data = y1, mean = unname(mean1), covariance = covar, prior = prior, n = dim(y1)[1]))
}
```

```r
pooledcov <- function(a,b,c){
  temp <- (a$cov * a$n) + (b$cov * b$n) + (b$cov * b$n)
  return(temp/dim(iris)[1])
}

Setosa <- a2(1)
Versicolor <- a2(2)
virginica <- a2(3)

# print("Setosa")
# print(Setosa[c(-1,-5)])
#
# print("Versicolor")
# print(Versicolor[c(-1,-5)])
#
# print("Virginica")
# print(virginica[c(-1,-5)])

write_matex <- function(x) {
  begin <- "\\begin{bmatrix}"
  end <- "\\end{bmatrix}"
  X <-
    apply(x, 1, function(x) {
      paste(
        paste(x, collapse = "&"),
        "\\\\"
      )
    })
  writeLines(c(begin, X, end))
}


pcov <- pooledcov(Setosa,Versicolor,virginica)
discrim <- function(x,a){
  constant <- (-1/2) * t(a$mean) %*% solve(pcov) %*% a$mean + log(a$prior)
  nonconstant <- t(x) %*% solve(pcov) %*% a$mean
  return(nonconstant+constant)
}
w0 <- function(a){
  constant <- (-1/2) * t(a$mean) %*% solve(pcov) %*% a$mean + log(a$prior)
  return(constant)
}

w <- function(a){
  nonconstant <- solve(pcov) %*% a$mean
  return(nonconstant)
}

predictLDA <- function(x){
  values <- c(discrim(x, Setosa), discrim(x,Versicolor), discrim(x,virginica))
  return(which.max(values))
}
```

```r
res <- cbind(iris[1:2],cut(apply(iris[1:2], 1, predictLDA),3, c("setosa","versicolor","virginica")))
colnames(res) <- c("Sepal.Length", "Sepal.Width", "Species")
plotcustom <- ggplot(res, aes(x=Sepal.Length, y=Sepal.Width, color=Species)) +
  geom_point(size=3) + theme_minimal() + ggtitle("LDA")

# a <- table(iris$Species, res$Species)
# knitr::kable(a, caption = "Confusion matrix ")


library(MASS)
fitLDA <- lda(Species ~ Sepal.Length + Sepal.Width, data = iris)
resLDA <- cbind(iris[1:2], predict(fitLDA, iris[1:2]))
colnames(resLDA) <- c("Sepal.Length", "Sepal.Width", "Species")

library(gridExtra)

plotLDA <- ggplot(resLDA, aes(x=Sepal.Length, y=Sepal.Width, color=Species)) +
  geom_point(size=3) + theme_minimal() + ggtitle("LDA using MASS library")

grid.arrange(plotcustom, plotLDA, nrow=2)
library(mvtnorm)
set.seed(12345)
gen <- rbind(rmvnorm(50, Setosa$mean, pcov),
             rmvnorm(50, Versicolor$mean, pcov),
             rmvnorm(50, virginica$mean, pcov))
gen <- round(gen,1)
gen <- as.data.frame(cbind(gen, c(rep("setosa",50), rep("versicolor",50), rep("virginica",50) )))
colnames(gen) <- c("Sepal.Length", "Sepal.Width", "Species")

plotfromequation <- ggplot(gen, aes(x=as.numeric(Sepal.Length), y=as.numeric(Sepal.Width), color=Specie
  geom_point(size=3) + ggtitle("Generated data")  + theme_minimal() + xlab("Sepal.Length") + ylab("Sepal
grid.arrange(plotiris, plotfromequation , nrow=2)
library(nnet)
fitlr <- multinom(Species ~ Sepal.Length + Sepal.Width, data = iris)
reslr <- predict(fitlr,iris[1:2])

reslr <- cbind(iris[1:2], reslr, deparse.level = 1)
names(reslr)[3] <- c("Species")

plotlr <- ggplot(reslr, aes(x=Sepal.Length, y=Sepal.Width, color=Species)) +
  geom_point(size=3) + ggtitle("Logistic regression")  + theme_minimal()

grid.arrange(plotlr,  plotcustom,  nrow=2)
table <- matrix(c(mean(iris$Species != resLDA$Species),mean(iris$Species != reslr$Species)))
colnames(table) <- c("Misclasification rate")
row.names(table) <- c("LDA", "LR")
knitr::kable(table)
bank = read.csv("bank-full.csv", header = TRUE, sep = ";",stringsAsFactors = TRUE)

bank = bank[,-12]
n = nrow(bank)
set.seed(12345)
id1=sample(1:n, floor(n*0.4))
```

```r
train=bank[id1,]
d2 = bank[-id1,]
n2 = nrow(d2)
id2=sample(1:n2, floor(n2*0.5))
test=d2[id2,]
validate=d2[-id2,]

library(tree)
dt_default = tree(y~.,data = train)
dt_size = tree(y~.,data = train, control = tree.control(nrow(train),minsize = 7000))
dt_dev = tree(y~.,data = train, control = tree.control(nrow(train),mindev = 0.0005))
#summary(dt_default)
#summary(dt_size)
#summary(dt_dev)

mce_default_train = summary(dt_default)$misclass[1]/summary(dt_default)$misclass[2]
mce_size_train = summary(dt_size)$misclass[1]/summary(dt_size)$misclass[2]
mce_dev_train = summary(dt_dev)$misclass[1]/summary(dt_dev)$misclass[2]

y_default = predict(dt_default, validate, type = "class")
y_size = predict(dt_size, validate, type = "class")
y_dev = predict(dt_dev, validate, type = "class")

missclass = function(y,y1){
  n = length(y)
  return( 1 - sum(diag(table(y,y1)))/n)
}

mce_default_val = missclass(validate$y,y_default)
mce_size_val = missclass(validate$y,y_size)
mce_dev_val =  missclass(validate$y,y_dev)


train_score = validate_score = rep(0,52)
for(i in 2:52){
  pruned_tree = prune.tree(dt_dev,best = i)
  pred = predict(pruned_tree, newdata=validate, type="tree")
  train_score[i] = deviance(pruned_tree)
  validate_score[i] = deviance(pred)
}
plot(2:52, train_score[2:52], type="p", col="red", ylim =c(7000,12000), xlab = "# terminal nodes", ylab
points(2:52, validate_score[2:52], type="p", col="blue")
legend("topright", c("validation", "test"), fill=c("blue", "red"))
optimal_tree = prune.tree(dt_dev, best = 36)
plot(optimal_tree)
text(optimal_tree, pretty = 0, cex = 0.5)

optimal_pred = predict(optimal_tree, test, type = "class")
confusion_matrix = table(test$y,optimal_pred)
missclassification_rate = missclass(test$y,optimal_pred)
knitr::kable(confusion_matrix)

loss_mat = t(matrix(c(0,1,5,0),2,2))
```

```r
row.names(loss_mat) = colnames(loss_mat) = c("no","yes")
op = predict(optimal_tree, test)
loss_fit = ifelse(loss_mat[1,2]*op[,1] > loss_mat[2,1]*op[,2] ,"no","yes")#1 is no , 2 is yes
loss_confusion_matrix = table(test$y,loss_fit)
m = missclass(test$y,loss_fit)
knitr::kable(loss_mat)
knitr::kable(loss_confusion_matrix)
library(e1071)
naive_model = naiveBayes(y~., train)
naive_y = predict(naive_model, test, type = "raw")

pi = seq(0.05,0.95,0.05)
prob_naive_yes = naive_y[,2]
prob_dt_yes = op[,2]
real_y = ifelse(test$y == "yes",1,0) # yes is 1 , no is 0
NAIVE = DT = matrix(0,ncol = 3,nrow = length(pi))


for(i in 1:length(pi)){
  dt_assign = ifelse(prob_dt_yes > pi[i],1,0)
  naive_assign = ifelse(prob_naive_yes > pi[i],1,0)

  cm_dt = table(real_y,dt_assign)
  cm_naive = table(real_y,naive_assign)

  if(all(dim(cm_dt) == c(2,2))== TRUE){
  tpr_dt = cm_dt[2,2]/sum(cm_dt[2,])
  } else {
  tpr_dt = 0
  }
  if(all(dim(cm_naive) == c(2,2))){
  tpr_naive = cm_naive[2,2]/sum(cm_naive[2,])
  } else {
  tpr_naive = 0
  }

  if(all(dim(cm_dt) == c(2,2))== TRUE){
  fpr_dt = cm_dt[1,2]/sum(cm_dt[1,])
  }else {
  fpr_dt = 0
  }
  if(all(dim(cm_dt) == c(2,2))== TRUE){
  fpr_naive = cm_naive[1,2]/sum(cm_naive[1,])
  }else {
  fpr_naive = 0
  }

  NAIVE[i,] = c(pi[i],tpr_naive,fpr_naive)
  DT[i,] = c(pi[i],tpr_dt,fpr_dt)


}
colnames(NAIVE) = c("pi","TPR","FPR")
colnames(DT) = c("pi","TPR","FPR")

#ROC curve
plot(DT[,3],DT[,2],ylab = "TPR",xlab = "FPR", type = "l", col = "red")
lines(NAIVE[,3],NAIVE[,2],col= "blue")
title("ROC Curve")
legend(0.3, 0.2, legend = c("Naive bayes","Optimal tree"), col=c("blue","red"), lty = c(1,1))

data_com=read.csv2("communities.csv", sep = ",")
library(ggplot2)
library(ggfortify)
## Load data and convert to numeric
data_com <- sapply(data_com, as.numeric)

#Scaling features excluding state and target
X <- scale(data_com[,c(-1, -101)])
cov_mat <- cov(X)

cov_mat_eigen <- eigen(cov_mat)

#proportion of variance explained for each component
proportion_variation <- cov_mat_eigen$values/sum(cov_mat_eigen$values)

plot(cov_mat_eigen$values, xlab = 'Eigenvalue Number', ylab = 'Eigenvalue Size',
     main = 'Scree Graph')
lines(cov_mat_eigen$values)


X.pca <- princomp(X)

plot(X.pca$scores[,1], main="First principal component" )

table <- as.matrix(sort(abs(X.pca$loadings[,1]), decreasing = TRUE)[1:13])
knitr::kable(as.matrix(sort(abs(X.pca$loadings[,1]), decreasing = TRUE)[1:13]),col.names = c("Value of

pca.plot <- autoplot(X.pca,)
autoplot(X.pca, data = data_com, colour = 'ViolentCrimesPerPop') + ggtitle("PC2 vs PC1")
X_data <- as.data.frame(cbind(X, data_com[,101]))
colnames(X_data)[100] <- "ViolentCrimesPerPop"

model = lm(ViolentCrimesPerPop ~ poly(X.pca$scores[,1], degree = 2), data= X_data)
ggplot(X_data, aes(X.pca$scores[,1], ViolentCrimesPerPop)) +
  geom_point() +
  geom_point(aes(x=X.pca$scores[,1], y=predict(model, X_data)), color="red") +
  ggtitle("Violent Crimes per Population vs PC1")



library(boot)

data <- as.data.frame(cbind(X.pca$scores[,1], X_data[100] ) )
colnames(data) <- c("PC1", "ViolentCrimesPerPop")
data <- data[order(data$PC1),]
```

```r
mle <- lm(ViolentCrimesPerPop ~ poly(PC1, degree = 2), data= data)


rng <- function(data, mle) {
  data1=data.frame(ViolentCrimesPerPop=data$ViolentCrimesPerPop, PC1=data$PC1)
  n=length(data$PC1)
  #generate new Price
  data1$ViolentCrimesPerPop=rnorm(n, predict(mle, newdata=data1), sd(mle$residuals))
  return(data1)
}

f1=function(data1){
  res=lm(ViolentCrimesPerPop ~ poly(PC1, degree = 2), data=data1) #fit linear model
  #predict values for all Area values from the original data
  y=predict(res,newdata=data)
  return(y)
}

res=boot(data, statistic=f1, R=1000, mle=mle,ran.gen=rng, sim="parametric")

e=envelope(res) # compute confidence bands
fit=mle
predicted_violence=predict(fit)

plot(data$PC1, data$ViolentCrimesPerPop, pch=21, bg="orange", main="PC1 vs ViolentCrimesPerPop, Confiden
points(data$PC1, predicted_violence) # plot fitted line

# plot confidence bands

points(data$PC1, e$point[2,], type="l", col="red")
points(data$PC1, e$point[1,], type="l", col="red")

####### Prediction interval ########

mle <- lm(ViolentCrimesPerPop ~ poly(PC1, degree = 2), data= data)

f1=function(data1){
  res=mle # fit
  # predict values for all PC1 values

  predicted_violence=predict(res,newdata=data)
  n=length(data$ViolentCrimesPerPop)
  predictedP=rnorm(n,predicted_violence,
                   sd(mle$residuals))
  return(predictedP)
}
res2=boot(data, statistic=f1, R=10000, mle=mle,ran.gen=rng, sim="parametric")

e2 = envelope(res2)


points(data$PC1, e2$point[2,], type="l", col="green")
points(data$PC1, e2$point[1,], type="l", col="green")
```