732A99 - Lab 2, block 1 Group L4

Keshav Padiyar, Jacob Welander & Henrik Olofsson

06 December, 2020

Contents

statement of contribution

Keshav Padiyar: Assignment 2

Jacob Welander: Assignment 3

Henrik Olofsson: Assignment 1

Assignment 1

Background

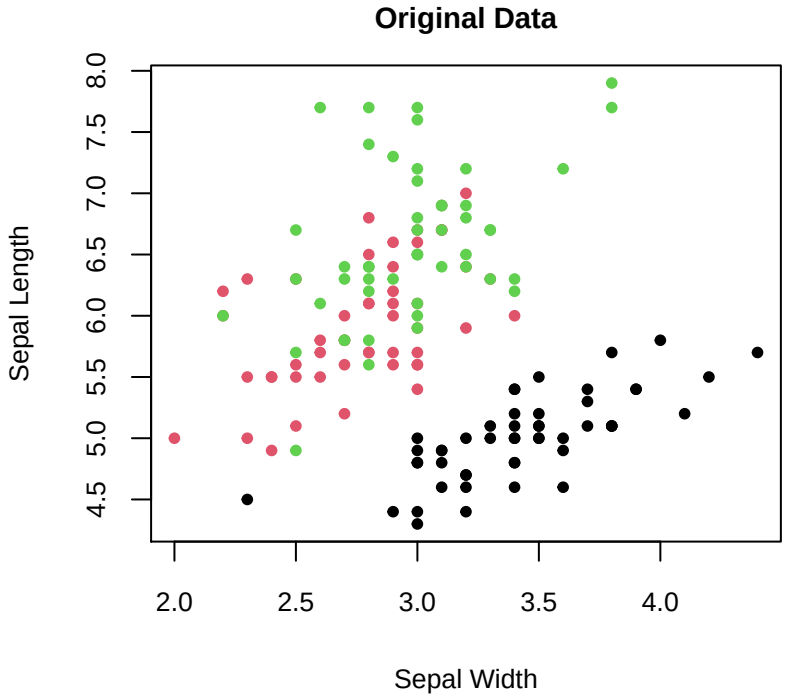
R data file "iris" (present in the default R installation) shows the measurements in centimeters of the variables sepal length and width and petal length and width, respectively, for 50 flowers from each of 3 species of iris. The species are Iris setosa, versicolor, and virginica.

In this assignment, we will perform Linear Discriminant Analysis (LDA) and logarithmic regression on the well-known iris dataset. We load this data conveniently from the R installation.

Solutions

1. Make a scatterplot of Sepal Width versus Sepal Length where observations are colored by Species. Do you think that this data is easy to classify by linear discriminant analysis? Motivate your answer.

See the plot below. From this plot alone, it is evident that there is a lot of overlap wrt two of the classes, however the third one could be easily separated. Without any additional information, it seems hard to find a good separation with just these two variables. The reason for this is that the class centroids/means are close and there seems to be a lot of variance/scatter, making the ratio $\frac{(\mu_1-\mu_2)^2}{s_1^2+s_2^2}$ very small no matter the rotation of the discriminator.



2. Use basic R functions only to implement Linear Discriminant Analysis between the three species based on variables Sepal Length and Sepal Width:

a. Compute mean, covariance matrices (use cov()) and prior probabilities per class and report them

```
## Means

##            Sepal.Length Sepal.Width
## setosa            5.006       3.428
## versicolor        5.936       2.770
## virginica         6.588       2.974

## Covariance Matrices

## target: setosa
##              Sepal.Length Sepal.Width
## Sepal.Length   0.12424898  0.09921633
## Sepal.Width    0.09921633  0.14368980
## ------------------------------------------------------------
## target: versicolor
##              Sepal.Length Sepal.Width
## Sepal.Length   0.26643265  0.08518367
## Sepal.Width    0.08518367  0.09846939
## ------------------------------------------------------------
## target: virginica
##              Sepal.Length Sepal.Width
## Sepal.Length   0.40434286  0.09376327
## Sepal.Width    0.09376327  0.10400408

## Prior Probabilities

## target
##     setosa versicolor  virginica
##  0.3333333  0.3333333  0.3333333
```

b. Compute overall (pooled) covariance matrix and report it

```
## Pooled Covariance Matrix

##              Sepal.Length Sepal.Width
## Sepal.Length   0.26500816  0.09272109
## Sepal.Width    0.09272109  0.11538776
```

c. Report the probabilistic model for the LDA

The probabilistic model is

$$\boldsymbol{x}|y = C_i, \mu_i, \Sigma \sim N(\mu_i, \Sigma)$$

and

$$y|\pi \sim Multinomial(\pi_1, ..., \pi_K)$$

where

- we assume that $\Sigma_i = \Sigma \forall i$
- π_k is estimated from the data as $\frac{N_k}{N}$
- $\hat{\mu}_k = \frac{1}{N_k} \sum_{i:y_i=k} x_i$
- $\hat{\mathbf{\Sigma}}_k = \frac{1}{N_k} \sum_{i:y_i=k} (x_i - \hat{\mu}_k)(x_i - \hat{\mu}_k)^T$

d. Compute discriminant functions for each class

We can compute the estimates for w and w_0:

$$w_{0k} = -\frac{1}{2}\boldsymbol{\mu}_k^T \Sigma^{-1} \boldsymbol{\mu}_k + log\pi_k$$

and:

$$w_k = \Sigma^{-1} \boldsymbol{\mu}_k$$

which gives:

```
## $w0
##       w01       w02       w03
## -65.32281 -70.47563 -85.68398
##
## $w
##                      w1        w2        w3
## Sepal.Length 11.81827 19.475666 22.037717
## Sepal.Width  20.21183  8.356129  8.065317
```

The discriminant functions are formed as follows:

$$\delta_k(x) = x^T \Sigma^{-1} \boldsymbol{\mu}_k - \frac{1}{2}\boldsymbol{\mu}_k^T \Sigma^{-1} \boldsymbol{\mu}_k + log\pi_k$$

We can substitute and get the three linear discriminant functions:

$$\delta_1(x) = w_1^T x + w_{01}$$
$$\delta_2(x) = w_2^T x + w_{02}$$
$$\delta_3(x) = w_3^T x + w_{03}$$

Together with the coefficients given above.

e. Compute equations of decision boundaries between classes and report them

The decision boundaries are obtained where the discriminant functions are equal, simply equating $\delta(1) = \delta(2) = \delta(3)$ pairwise and moving all terms to one side gives us the boundaries. It is then possible to simply calculate the result for a point and make a decision based on the sign of the result.

$(w_1 - w_2)x + (w_{01} - w_{02}) = 0$

$(w_1 - w_3)x + (w_{01} - w_{03}) = 0$

$(w_2 - w_3)x + (w_{02} - w_{03}) = 0$

This can be interpreted: $w_1 - w_2$ gives the resulting vector (-7.66, 11.86), x is the vector of input features (Sepal.Length, Sepal.Width), $w_{01} - w_{02}$ is 5.15 and so on.

These are the computed boundaries:

```
##       L1_diff L2_diff Constant_diff
## d1-d2   -7.66   11.86          5.15
## d1-d3  -10.22   12.15         20.36
## d2-d3   -2.56    0.29         15.21
```

Do estimated covariance matrices seem to fulfill LDA assumptions?

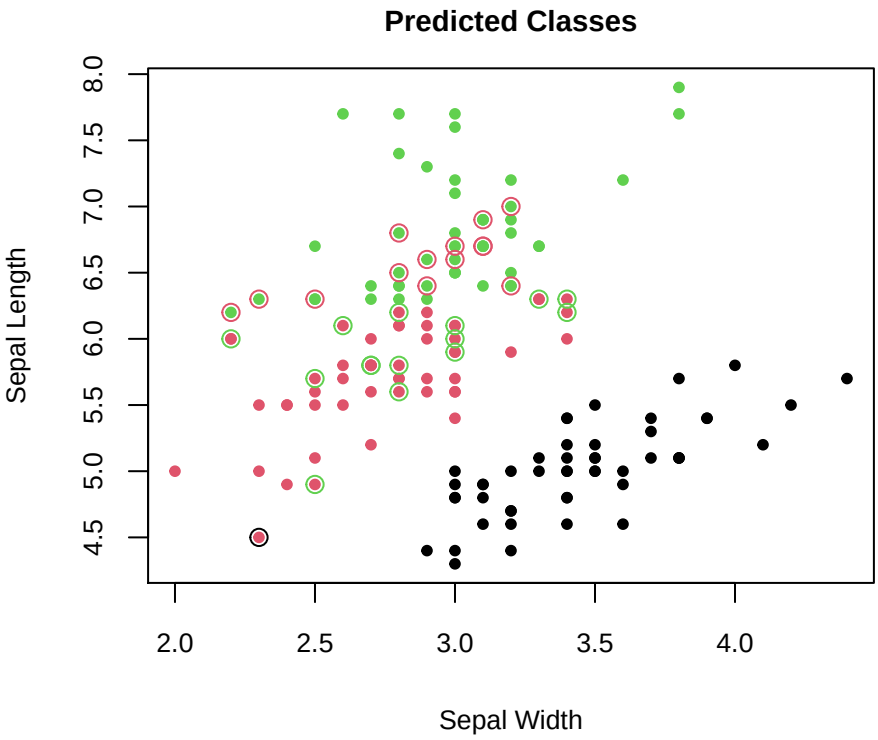
Not quite, as we can see the three Σ_k differ:

```
## Covariance matrix for species setosa

##              Sepal.Length Sepal.Width
## Sepal.Length         0.12        0.10
## Sepal.Width          0.10        0.14

## Covariance matrix for species versicolor

##              Sepal.Length Sepal.Width
## Sepal.Length         0.27        0.09
## Sepal.Width          0.09        0.10

## Covariance matrix for species virginica

##              Sepal.Length Sepal.Width
## Sepal.Length         0.40        0.09
## Sepal.Width          0.09        0.10
```

This should give rise to quadratic terms in our computations, indicating that the LDA might not be a great fit for the data.

3. Use discriminant functions from step 2 to predict the species from the original data and make a scatterplot of Sepal Length versus Sepal Width in which color shows the predicted Species.

To make predictions, we simply input our feature data into the discriminant functions and pick the class of whichever gives the highest discriminant score. The scatterplots below show the predicted values from our own implementation and the one from lda function. The colour of each dot shows the predicted value. Misclassified points are circled with their true class colour.



Estimate the misclassification rate of the prediction. Comment on the quality of classification. Afterwards, perform the LDA analysis with lda() function and investigate whether you obtain the same test error by using this package. Should it be same?

The misclassification rates and confusion matrices are presented below, and are equal for the two methods. The resulting error rate of 0.2 is not very impressive, but considering the overlap of two of the classes, it was to be expected. The same error rate is obtained from the `lda()` method from `MASS` package and the same classification is made as previously for each observation.

The `lda()` method probably does additional checks and possibly corrections like making the covariance matrices spherical. This could *possibly* create a different result under different circumstances. It may also use another method to for the inversion of the covariance matrix. Other than that, there is no randomness or parameter settings that would change the result in this case.

```
## Confusion matrix for own LDA implementation
```

```
##             Predicted
## True         setosa versicolor virginica
##   setosa         49          1         0
##   versicolor      0         36        14
##   virginica       0         15        35
```

```
## Error rate for own LDA implementation
```

```
## [1] 0.2
```

```
## Confusion matrix for lda()
```

```
##             Predicted
```

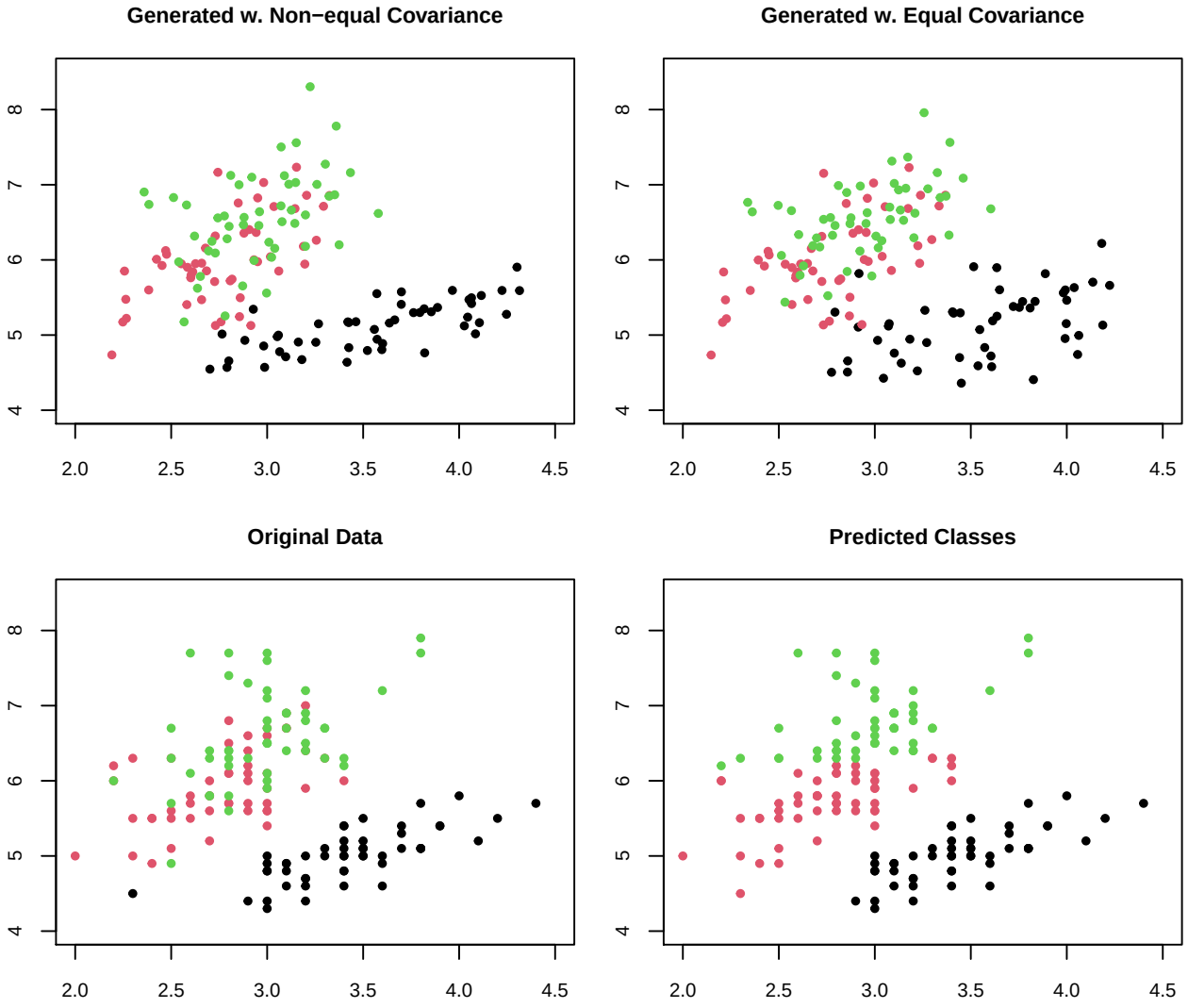
```
## True         setosa versicolor virginica
##   setosa         49          1         0
##   versicolor      0         36        14
##   virginica       0         15        35
```

```
## Error rate for lda()
```

```
## [1] 0.2
```

4. Use Models reported in 2c to generate new data of this kind with the same total number of cases as in the original data (hint: use sample() and rmvnorm() from package mvtnorm). Make a scatterplot of the same kind as in step 1 but for the new data and compare it with the plots for the original and the predicted data. Conclusions?

The generated data can be seen below. The top left plot depicts data generated without the assumption of equal covariance, while the top right one does make that assumption. Especially for the black class, the assumption of equal covariances produces a more spherical sample than in the original data. Considering our decision boundaries from before (see the predicted values above), we should still see a large misclassification error if we were to use LDA on this data.



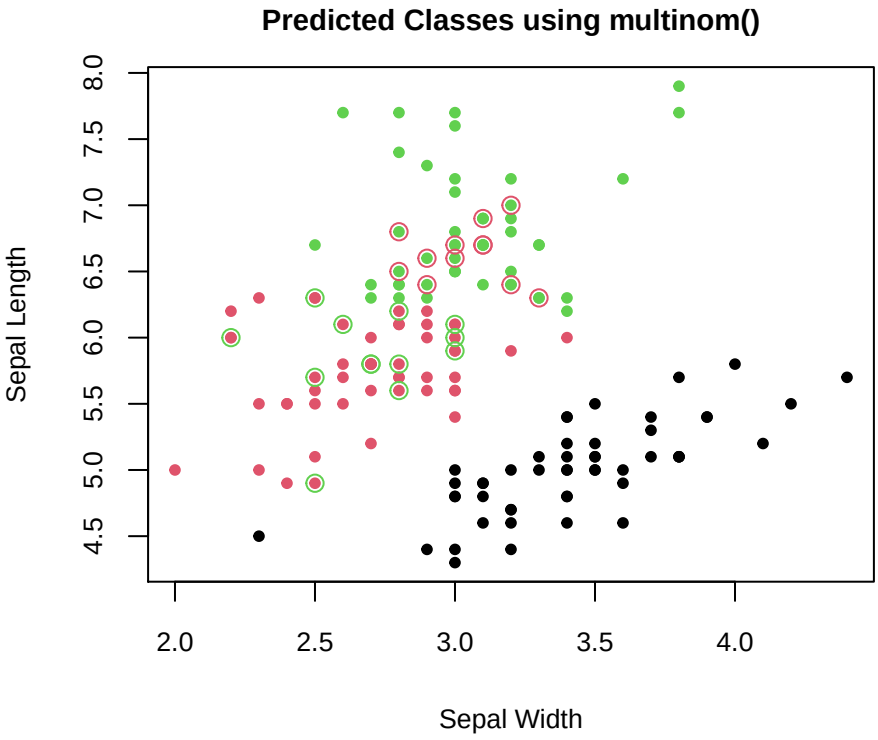
5. Make a similar kind of classification by logistic regression (use function multinom() from nnet package), plot the classified data and compute the misclassification error. Compare these results with the LDA results.

```
## Confusion matrix for multinom()
```

```
##             Predicted
## True         setosa versicolor virginica
##   setosa         50          0         0
##   versicolor      0         38        12
##   virginica       0         13        37
```

```
## Error rate for multinom()
```

```
## [1] 0.1666667
```



These results are slightly better than for the LDA method. Specifically, even the outlier among the black class in the plot above is classified correctly. The main problem is still the large overlap between the green and red classes above, where no linear separation will ever be able to fully separate the classes without error.

Assignment 2. Decision trees and Naïve Bayes for bank marketing

1. Import the data to R, remove variable "duration" and divide into training/validation/test as 40/30/30

2. Fit decision trees to the training data so that you change the default settings one by one (i.e. not simultaneously):

a. Decision Tree with default settings.

```
## Confusion Matrix: Train

##      predicted
## true     no   yes
##   no  15795   223
##   yes  1673   393

##  Misclassifincation Error in train:  0.1048441

## Confusion Matrix: Valid

##      predicted
## true     no   yes
##   no  11772   153
##   yes  1329   309

##  Misclassifincation Error in Valid:  0.1092679
```

b. Decision Tree with smallest allowed node size equal to 7000.

```
## Confusion Matrix: Train

##      predicted
## true     no   yes
##   no  15795   223
##   yes  1673   393

##  Misclassifincation Error in train:  0.1048441

## Confusion Matrix: Valid

##      predicted
## true     no   yes
##   no  11772   153
##   yes  1329   309

##  Misclassifincation Error in Valid:  0.1092679
```

c. Decision trees minimum deviance to 0.0005.

```
## Confusion Matrix: Train

##      predicted
## true     no   yes
##   no  15821   197
##   yes  1503   563

##  Misclassifincation Error in train:  0.09400575

## Confusion Matrix: Valid

##      predicted
## true     no   yes
##   no  11715   210
##   yes  1308   330

##  Misclassifincation Error in Valid:  0.1119221
```

Report the misclassification rates for the training and validation data. Which model is the best one among these three? Report how changing the deviance and node size affected the size of the trees and explain why

Below table shows the misclassication rates for training and validation data. Observing the metrics showed in table, tree with default setting is best out of three. The said tree has same error rate but slightly less mean residual deviance when compared with the tree with *minsize = 7000* and has least error rate compared to the tree with *mindev=0.0005*.

Effect of modifying minsisze: This parameter Controls the number of nodes by considering the number of observations that a node can have. By setting this number to 7000 we have restricted the tree to grow only upto 5 terminal nodes. By comparing this tree with the default tree we could see that due to reduced splits, the mean residual deviance is slightly high.

Effect of modifying mindev: The within node deviance must be equal to mindev times the deviance in the root node, for the node to split. Smaller the value of mindev larger the tree size. But we observe that given the fully grown tree, model overfits. Even execution time is also slightly more compared to the other trees in the table.

Table 1: Model Observations

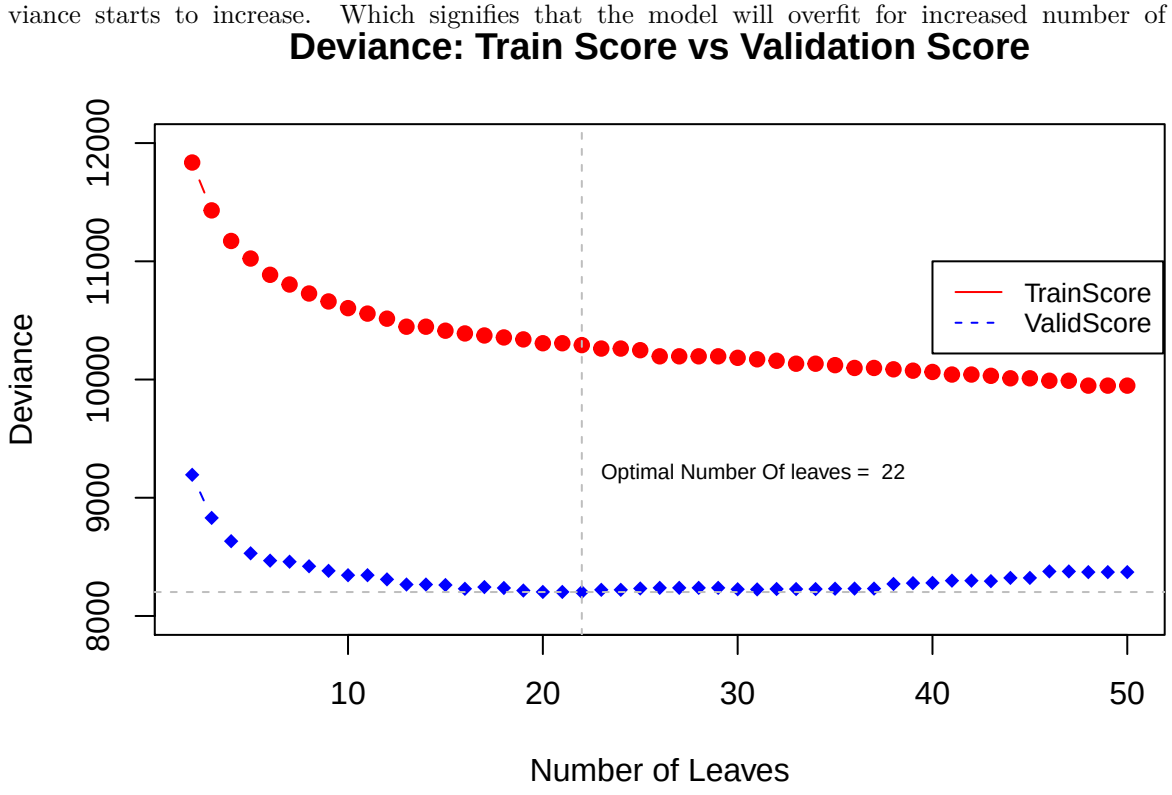
Tree.Setting	Residual.mean.deviance	Train.Error.Rate	Valid.Error.Rate	Execution.Time
Default	0.6021626	0.1048441	0.1092679	0.155
minsize=7000	0.6097382	0.1048441	0.1092679	0.120
mindev=0.0005	0.5212852	0.0940058	0.1119221	0.348

3. Use training and validation sets to choose the optimal tree depth in the model 2c: study the trees up to 50 leaves.

Present a graph of the dependence of deviances for the training and the validation data on the number of leaves and interpret this graph. Report the optimal amount of leaves and which variables seem to be most important for decision making in this tree. Interpret the information provided by the tree structure (not

everything but most important findings). Estimate the confusion matrix and misclassification rate for the test data and comment whether the model has a good predictive power.

The deviance dependence graph shows that, deviance for training data is decreasing continuously as the number of leaves increases. Where as in validation data, the deviance seems to be decreasing at first till the optimal number of leaves (number of leaves = 22), there after the validation deviance starts to increase. Which signifies that the model will overfit for increased number of leaves.




```
## Optimal Number Of Leaves:  22
```

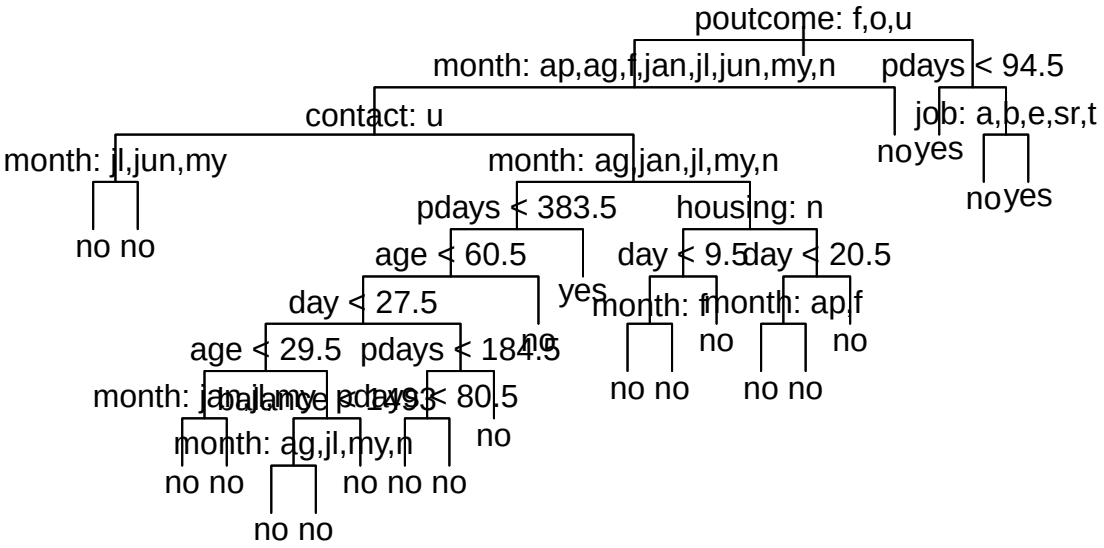
```
## Most Contributing Variables:  poutcome month contact pdays age day balance housing job
```

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

```
##      predicted
## true     no   yes
##   no  11872   107
##   yes  1371   214
```

```
##  Misclassifincation Error in Test:  0.1089649
```

Observing the above results - miscalssification rate in test and the residual mean deviance, the tree pruned to optimal number of leaves has good predictive power.



The above optimal tree graph contains 22 leaf nodes and depth = 11. Out of 15 input variables, the tree considers only 9 variables for decision making. Analyzing the tree further,

Split 1: tree splits based on outcome of the previous marketing campaign (*poutcome*) where there were 4 outcomes, out of which outcome success had more client subscribed to the term deposit hence we could see 2 branches at the root node of the graph - right branch for successful outcome and left for the rest.

Tracing the right most branch:

Split 2: The second split takes place based on the number of days that passed by after the client was last contacted from a previous campaign. If the campaign and customer contacting happens within 94.5 days, there is a high probability that the client would subscribe for the term deposit and turns out to be the leaf. The lower probability branch since it has not reached min deviance limit goes for further split.

Split 3: The final split on this branch happens based on the job of the client. Tree identifies that the clients belonging to admin,blue-collar,entrepreneur,services and technician category are highly likely to opt term deposit option and rest are not. This branch ends into two leaf nodes.

Remaining branches happen in the similar fashion.

4. Perform a decision tree classification of the test data with the following loss matrix

$$L = observed \begin{matrix} no \\ yes \end{matrix} \overset{Predicted}{\begin{pmatrix} 0 & 1 \\ 5 & 0 \end{pmatrix}}$$

and report the confusion matrix for the test data. Compare the results with the results from step 3 and discuss how the rates has changed and why.

The loss matrix used to weigh the mis-classifications. In case of two class problems, False Positive and True Negative are the errors which needed to be weighed based on the type, which causes more of a loss.

We have used the above mentioned loss matrix which alters the probabilities. Here we are increasing the probabilities of Class Yes to avoid mis-classification of clients into NO bucket. Hence the resulting classification gets altered. Same can be observed from the below mentioned confusion matrices. Error rate is more that is because we increased the probability of class yes using loss matrix.

```
##  Misclassifincation Error in Test with Loss Matrix:  0.1268063
```

```
## Optimal Tree Confusion Matrix
```

```
##      predicted
## true     no   yes
##   no  11872   107
##   yes  1371   214
```

```
## 5-1 Loss Matrix Tree Confusion Matrix
```

```
##      predicted
## true     no   yes
##   no  11030   949
##   yes   771   814
```

5. Use the optimal tree and the Naïve Bayes model to classify the test data by using the following principle:

$\hat{Y} = 1\ if\ p(Y = good|X) > \pi,\ otherwise \hat{Y} = 0$ where π=0.05, 0.1, 0.15, ...0.9, 9.95. Compute the TPR and FPR values for the two models and plot the corresponding ROC curves. Conclusion?

```
## Classification Using Naive Bayes
```

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

```
##      predicted
## true     no   yes
##   no  14468  1550
##   yes  1271   795
```

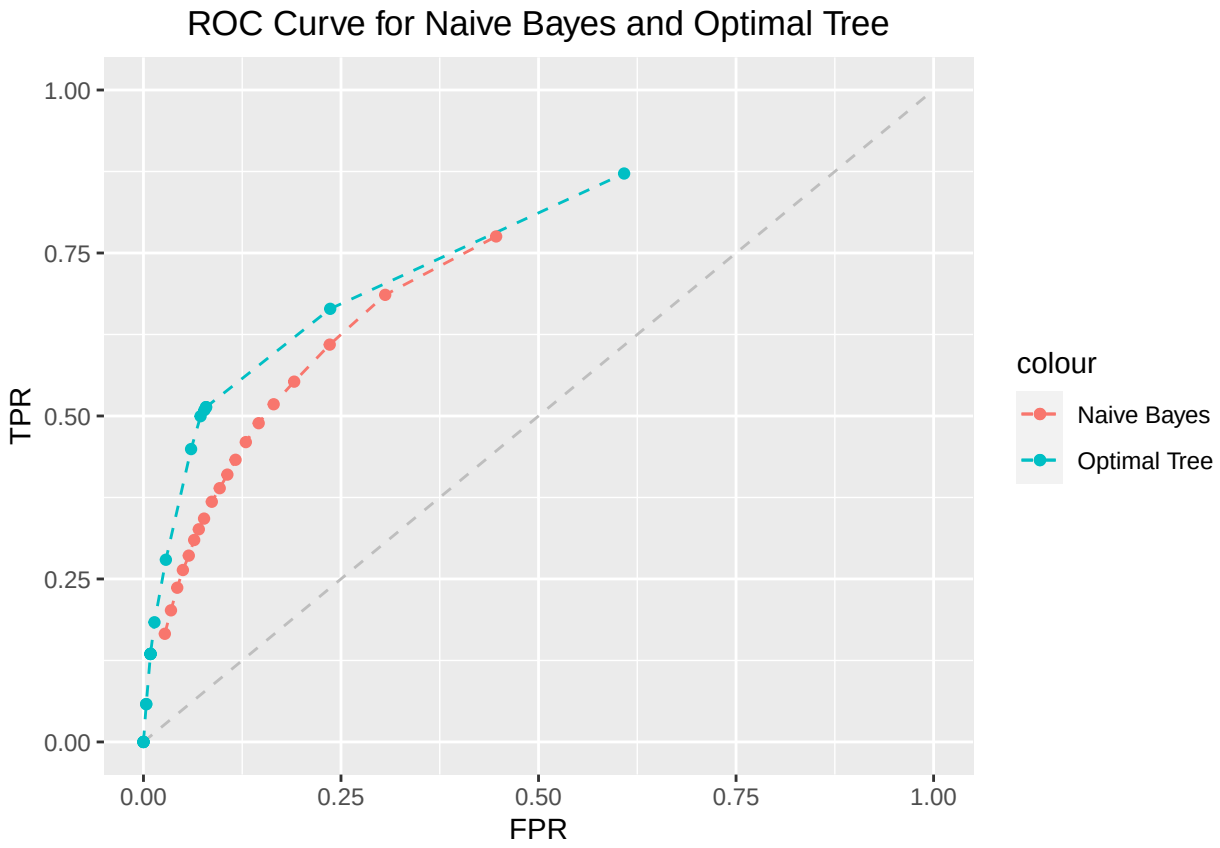
```
##  Misclassifincation Error in Train:  0.1559942
```

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

```
##      predicted
## true     no   yes
##   no  10823  1156
##   yes   968   617
```

```
##  Misclassifincation Error in Test:  0.156591
```

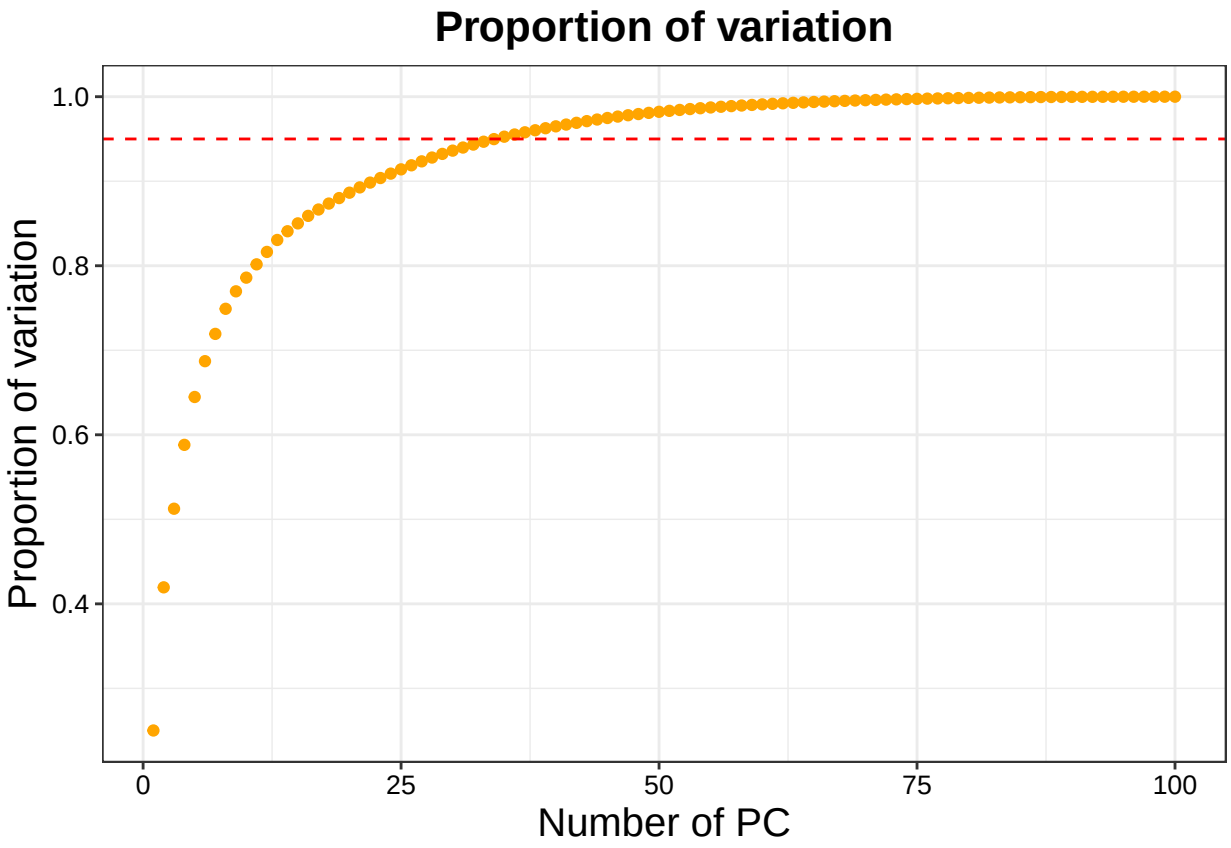
Observing the below ROC curves for Naive Bayes and Optimal Decision tree models, the area under the optimal tree is larger than the Naive Bayes model. Hence, Optimal Tree is better classifier for the given data.


ROC Curve for Naive Bayes and Optimal Tree
1.00
0.75
0.50
0.25
0.00
TPR
0.00
0.25
0.50
0.75
1.00
FPR
colour
Naive Bayes
Optimal Tree

1. Principal components for crime level analysis

1.

Scale all variables except of ViolentCrimesPerPop and implement PCA by using function eigen().




```
## [1] 0.4195296
```

Report how many features are needed to obtain at least 95% of variance in the data.

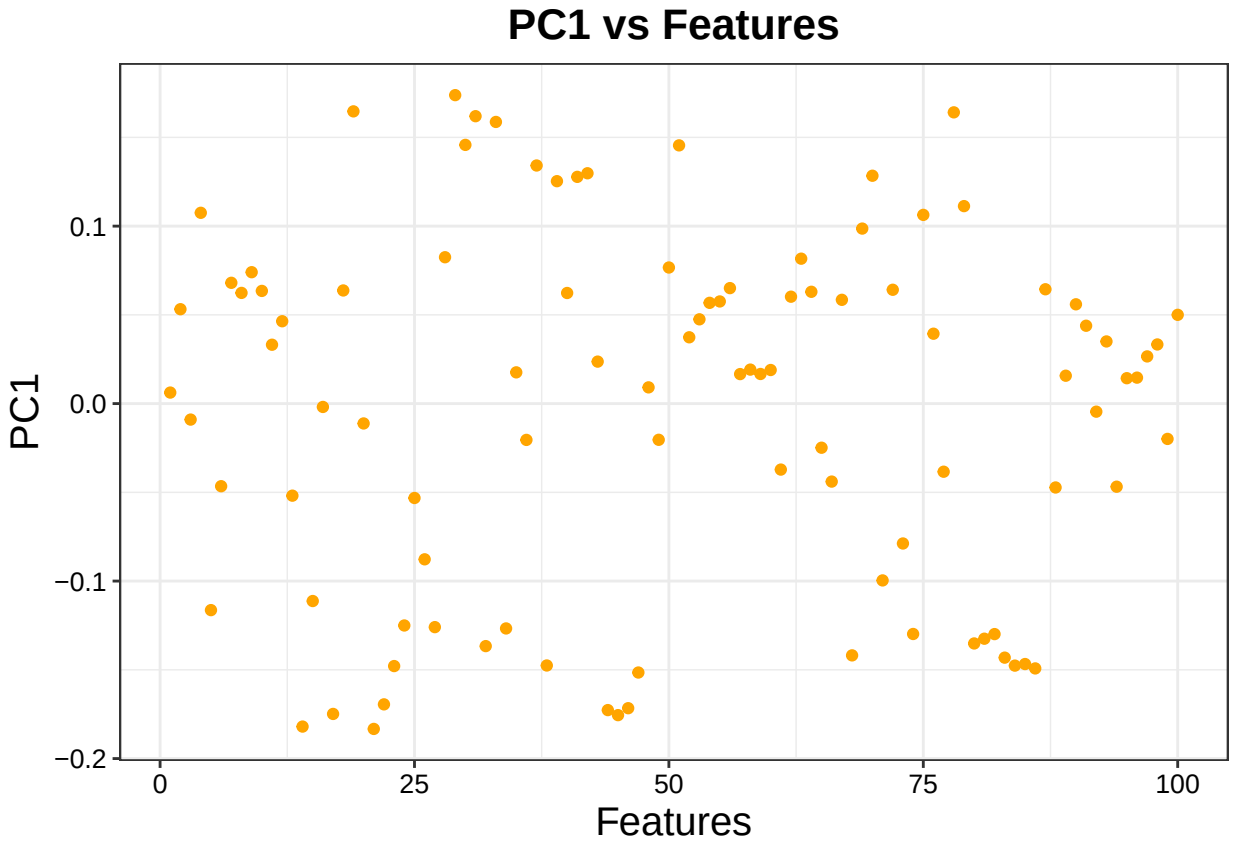
As seen in the graph with the cumulative proportion of variation above, the number of features needed to achieve at least 95% of the variance in the data are 35.

What is the proportion of variation explained by each of the first two principal components?

The two Principal component PC1 and PC2 explains about 42% of the variation of the data set.

2

Repeat PCA analysis by using princomp() function and make the score plot of the first principle component. Do many features have a notable contribution to this component? Report which 5 features contribute mostly (by the absolute value) to the first principle component.



	x
medFamInc	0.1833080
medIncome	0.1819830
PctKids2Par	0.1755423
pctWInvInc	0.1748683
PctPopUnderPov	0.1737978

From the graph above the first principal component is visualized against the features reviling no pattern, but some clusters is identified such as income results in a high absolute score on PC1, but also a cluster with a lower score on PC1 that contains features regarding information about the communities such as $PctSameCity85$ and $PctUsePubTrans$ etc.

From the table one can see that the 5 features with most contribution are $medFamInc$, $medIncome$, $PctKids2Par$, $pctWInvInc$ and $PctPopUnderPov$.

The definition of each feature is :

- medFamInc: median family income
- medIncome: median household income
- PctKids2Par: percentage of kids in family housing with two parents
- pctWInvInc: percentage of households with investment / rent income in 1989

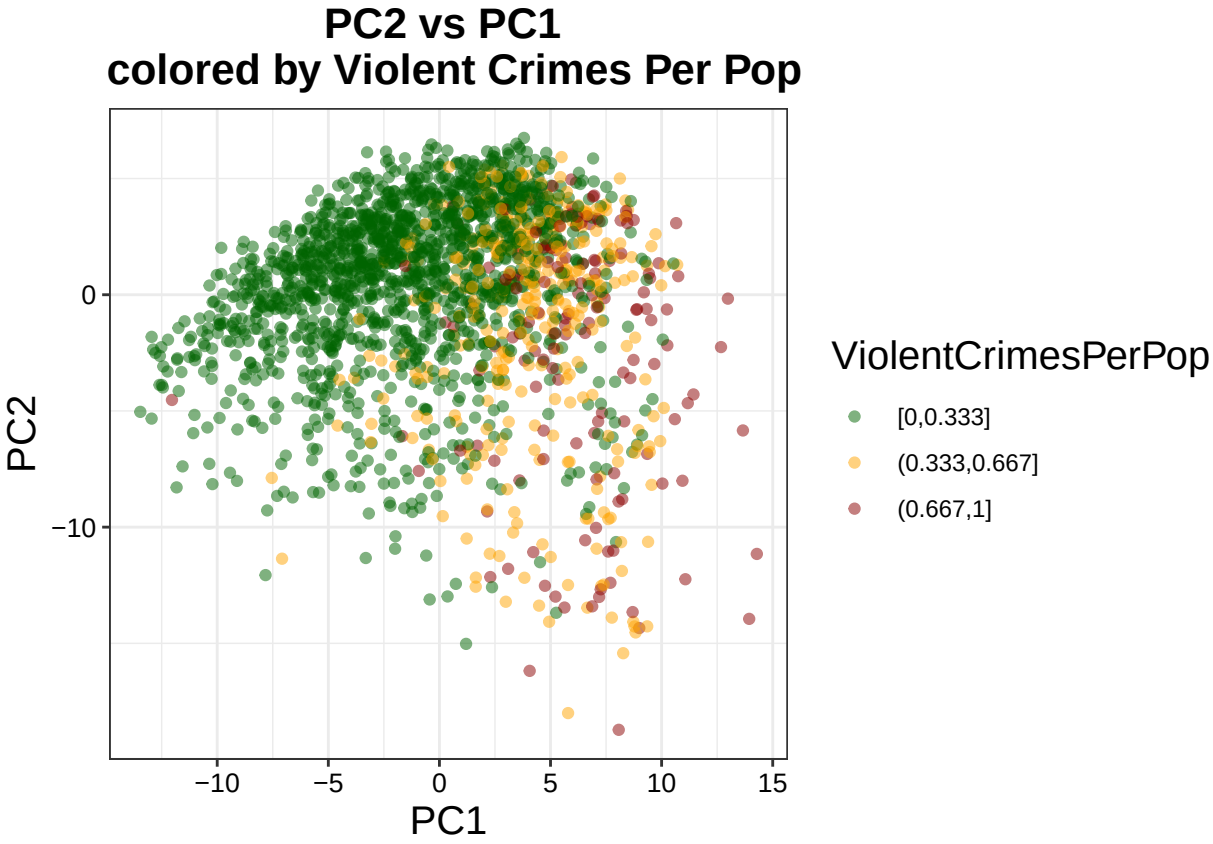
- PctPopUnderPov: percentage of people under the poverty level

Four out of five of the features seems to be connected to income or the economic welfare of the household. This would indicate that low income communities tends to have a large impact the violent crimes per population. The one feature not directly connected to income is the percentage of kids in family housing with two parents. The feature that contains percentage of people under the poverty level seems to be the only principal component that has a positive score of the five most contribution features.

Comment whether these features have anything in common and whether they may have a logical relationship to the crime level.

Most of the variables are connected to the income of the communities which is one of the key factor in a societies welfare.

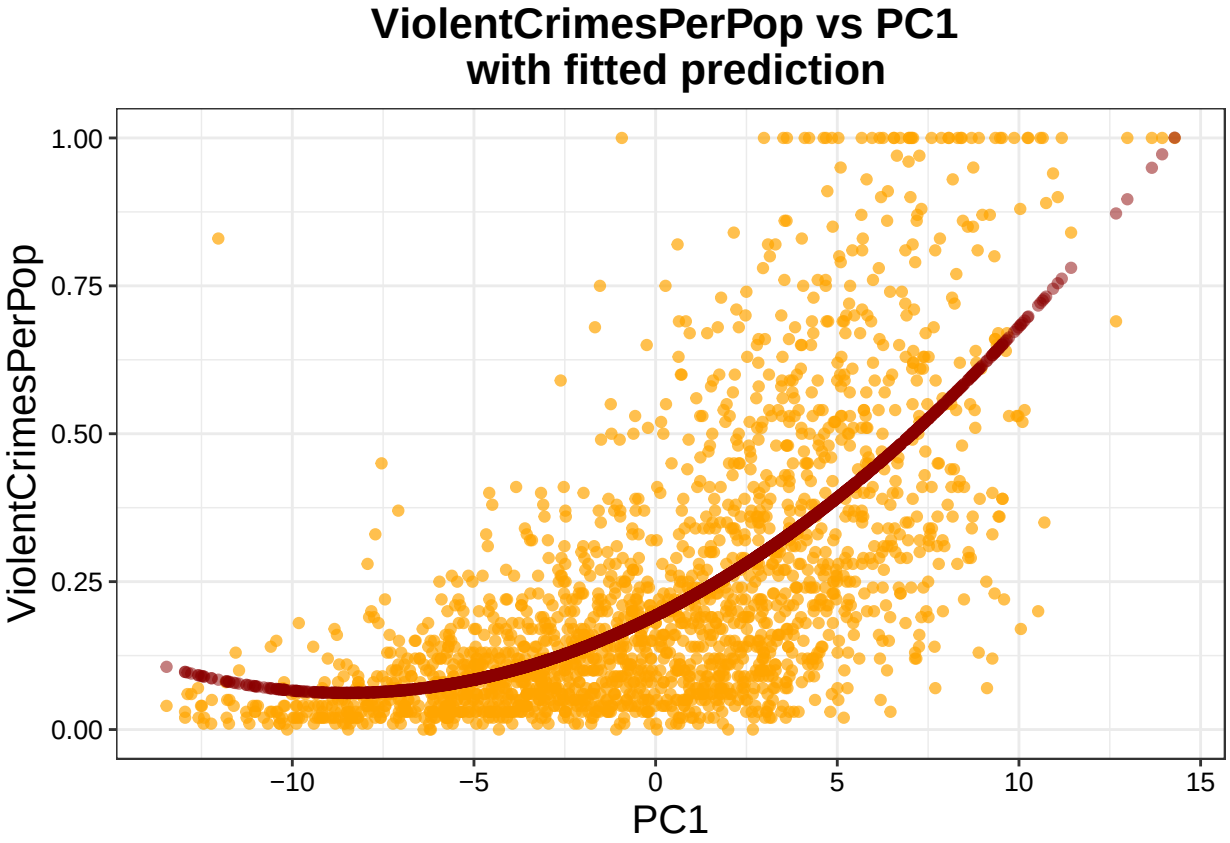
Also provide a plot of the PC scores in the coordinates (PC1, PC2) in which the color of the points is given by ViolentCrimesPerPop.



The graph above visualizes the two principal component that explains 42% of the variation where the target variable *ViolentCrimesPerPop* is divided into three intervals and is displayed with color. One can identify communities with high level of violent crimes per population with high positive values on PC1 with a large spread throughout PC2.

3

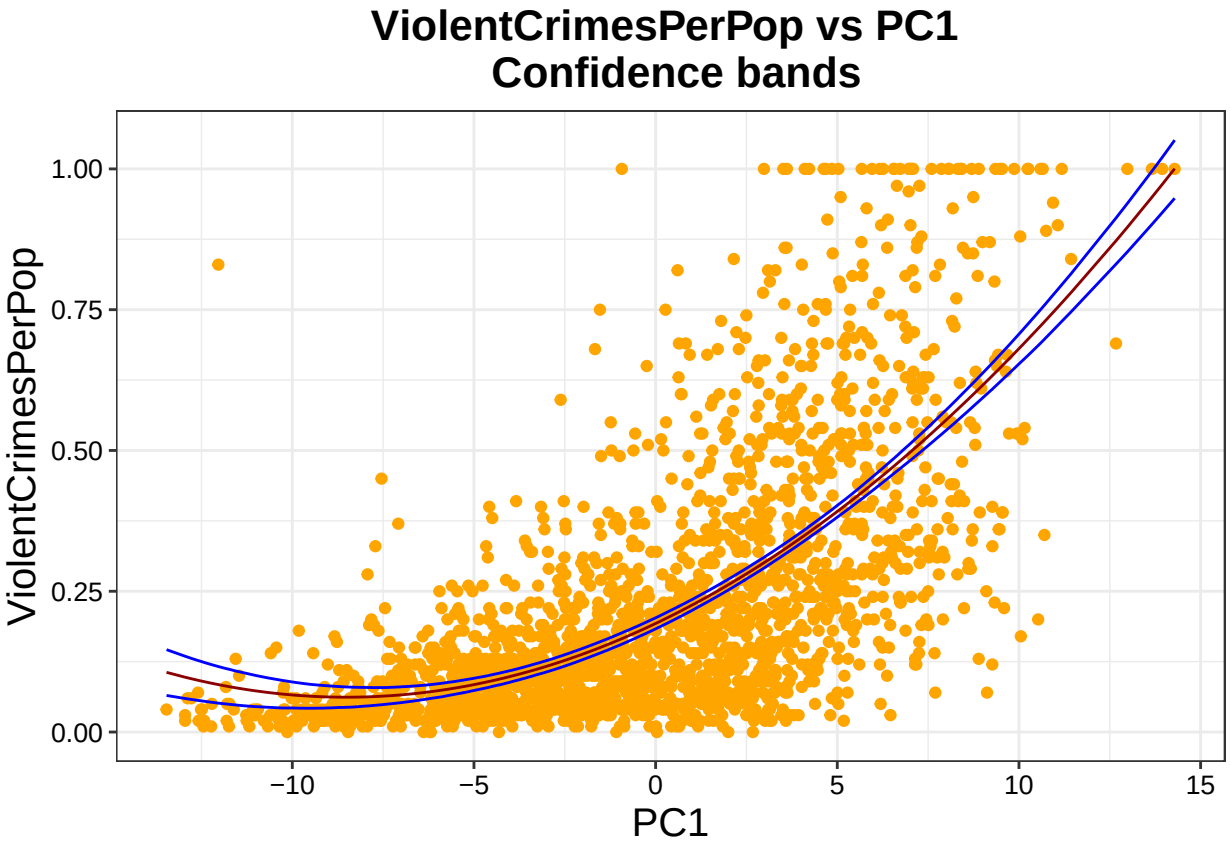
Assume a second order polynomial regression model in which ViolentCrimesPerPop is target and PC1 is the feature. Compute this model using lm() function (hint: use poly() function within the formula), make a scatterplot of the target versus the feature and present also the predicted values in this plot.



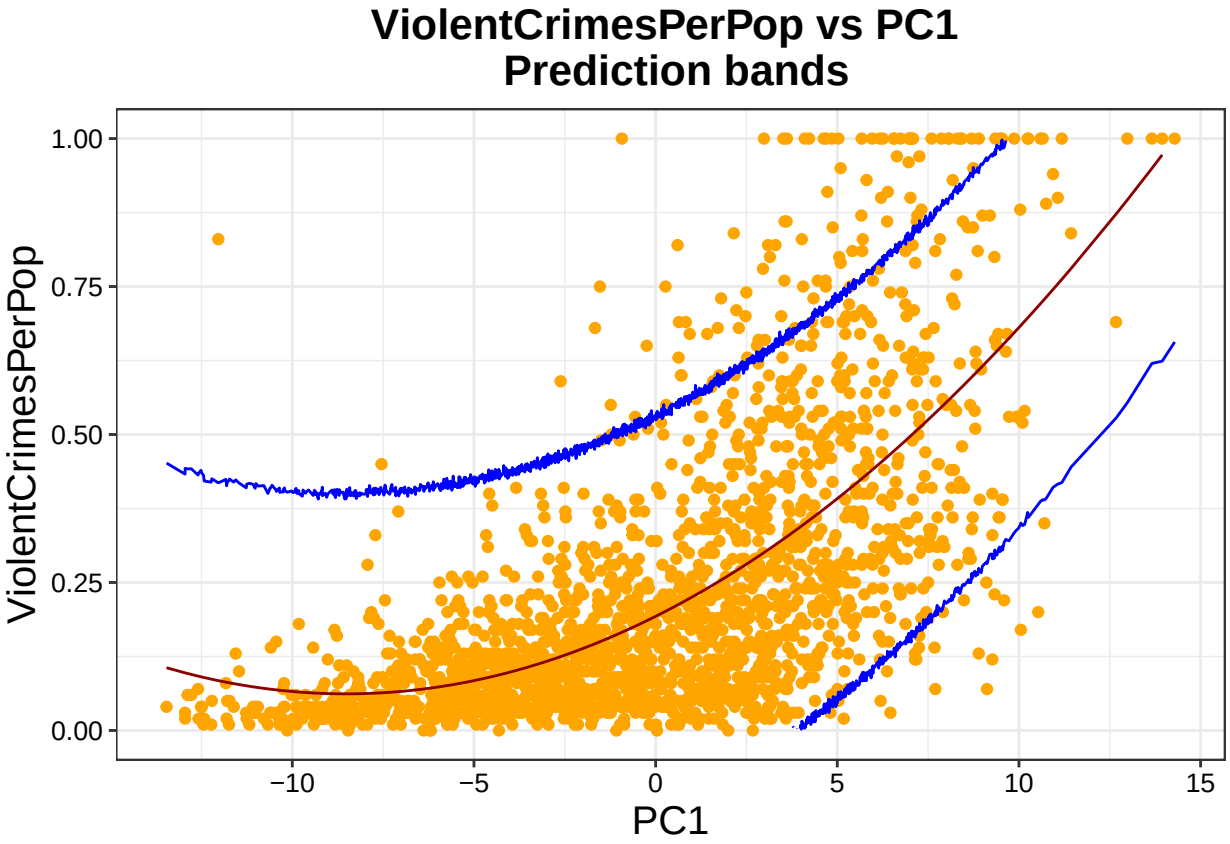
Can the target be well explained by this feature? According to the model R^2 value the PC1 explains ~45% of the variation in target value of ViolentCrimesPerPop.

Does the model seem to capture the connection between the target and the feature? In general from the previous plot one can draw the conclusion that higher value PC1 in a community leads to higher value of violent crimes per population. Due to the large variation throughout the target value and the features the polynomial fail to cover a lot observations that has lower values PC1 that is still contains high values of the target.

4 Use parametric bootstrap to estimate the confidence and prediction bands from the model from step 3 and add these bands into the plot from step 3. What can be concluded by looking at a) **confidence intervals?** b) **prediction intervals?**



According to the plot above it's concluded that that confidence interval with 95% certainty will contain the true mean of the population.



As can be seen in the plot above the prediction bands seems to have a larger interval compared to the confidence interval from the previous plot. This is due to the fact the prediction interval reveals where one can expect to observe a new data point if sampled, giving the interpretation that 95% of the new data points will be within the interval.

Code Appendix

```
### Assignment 1 ###
knitr::opts_chunk$set(echo = FALSE)
knitr::opts_chunk$set(message = TRUE)
library(MASS)
library(mvtnorm)
library(nnet)
# getting the data
data <- iris
target <- data$Species # slicing our target to a separate frame
features <- data[,1:2] # slicing the two features we are going to use

# setting plot parameters
par(mar = c(4.1, 4.1, 2.5, 2))

# plotting original data
plot(
  data$Sepal.Width,
  data$Sepal.Length,
  col = data$Species,
  pch = 20,
  main = "Original Data",
  xlab = "Sepal Width",
  ylab = "Sepal Length",
  cex.main = 0.9,
  cex.lab = 0.8,
  cex.axis = 0.8
)

p <- ncol(features) # number of predictors
N <- nrow(features) # nobs
C <- length(levels(target)) # nclasses

mu <- do.call("rbind", by(features, target, colMeans)) # list of mu's for each class and dimension
s <- by(features, target, cov) # covariance matrices for each class
pi <- (prop.table(table(target)))  # class probabilities

# output the descpriptives
message("Means")
print(mu)
message("Covariance Matrices")
print(s)
message("Prior Probabilities")
print(pi)


# calculating the ppoled covariance matrix
S <- matrix(0, nrow = p, ncol = p, dimnames = list(colnames(features), colnames(features)))
for (c in 1:C) {
  S <- S + matrix(nrow = p, unlist(Map('*',s[c],pi[c]))) # using Map to do list multiplication
}
```

```r
message("Pooled Covariance Matrix")
S


# function to get LDA coefficients
lincoef <- function(mu, S, pi, classes = C) {

  C <- classes
  w <- matrix(nrow = ncol(mu), ncol = C, dimnames = list(colnames(mu), paste0("w", 1:3)))
  w0 <- vector(length = C)
  names(w0) <- paste0("w0", 1:3)

  # simply translating the equations into code
  for (c in 1:C) {
      w0[c] <- -.5 * t(mu[c, ]) %*% solve(S) %*% mu[c, ] + log(as.matrix(pi[c]))
      w[,c] <- solve(S) %*% mu[c,]
  }

  return(list("w0" = w0, "w" = w))

}

m1 <- lincoef(mu, S, pi)

m1

# pairwise differences
b1 <- data.frame(t((m1$w[,1])-(m1$w[,2])), "w0diff" = m1$w0[1]-m1$w0[2])
b2 <- data.frame(t((m1$w[,1])-(m1$w[,3])), "w0diff" = m1$w0[1]-m1$w0[3])
b3 <- data.frame(t((m1$w[,2])-(m1$w[,3])), "w0diff" = m1$w0[2]-m1$w0[3])
b <- round(rbind("d1-d2" = b1, "d1-d3" = b2, "d2-d3" = b3),2)
colnames(b) <- c("L1_diff", "L2_diff", "Constant_diff")
b
message("Covariance matrix for species setosa")
round(s$setosa,2)

message("Covariance matrix for species versicolor")
round(s$versicolor,2)

message("Covariance matrix for species virginica")
round(s$virginica,2)

# function to get discriminant scores and predictions
discriminant <- function(x, coefs) {

  w <- t(as.matrix(coefs$w))
  w0 <- coefs$w0
  x <- t(as.matrix(x))
  d <- matrix(nrow = ncol(x), ncol=length(w0))
  c_pred <- vector(mode="numeric", length = nrow(d))

  # discriminant scores
  for (i in 1:nrow(w)) {
```

```
    d[,i] <- w[i, ] %*% x + w0[i]
  }

  # making the predictions according to highest d-score
  for (n in 1:nrow(d)) {
    c_pred[n]  <- levels(target)[which.max(d[n,])]

  }

  results <- list("discriminant scores" = d, "predicted" = c_pred)
  return(results)
}

d <- discriminant(features, m1)
misclassed <- which(target != d$predicted)

# plotting the results
par(mar = c(4.1, 4.1, 2.5, 2))
plot(
  data$Sepal.Width,
  data$Sepal.Length,
  col = as.factor(d$predicted),
  pch = 20,
  main = "Predicted Classes",
  xlab = "Sepal Width",
  ylab = "Sepal Length",
  cex.main = 0.9,
  cex.lab = 0.8,
  cex.axis = 0.8
)
# adding circles to display correct class for misclassified points
points(
  data$Sepal.Width[misclassed],
  data$Sepal.Length[misclassed],
  pch = 1,
  cex = 1.2,
  col = as.factor(data$Species[misclassed])
)

# using lda() to train and predict
m2 <- lda(Species~.,data = data[,-c(3:4)])
m2_pred <- predict(m2)

# getting and plotting error rates and confusion matrices
m1_conf <- table("True" = target, "Predicted" = d$predicted)
m1_err <- 1-(sum(diag(m1_conf))/sum(m1_conf))

m2_conf <- table("True" = target, "Predicted" = m2_pred$class)
m2_err <- 1-(sum(diag(m2_conf))/sum(m2_conf))

message("Confusion matrix for own LDA implementation")
m1_conf
message("Error rate for own LDA implementation")
```

```r
m1_err

message("Confusion matrix for lda()")
m2_conf
message("Error rate for lda()")
m2_err


# function to generate data
gen <- function(mu, s, n) {
  df <- matrix(nrow = n, ncol = ncol(mu) + 1)

  for (i in 1:nrow(mu)) {
    k <-
      seq(i * n / nrow(mu) - (n / nrow(mu) - 1), i * n / nrow(mu), by = 1)

    # if the covariance matrix is a matrix it means the pooled covariance matrix
    if (is.matrix(s)) {
      df[k, 1:2] <- rmvnorm(n = n / nrow(mu),
                            mean = mu[i,],
                            sigma = s)
    } else { # else it is a list of separate covariance matrices
      df[k, 1:2] <- rmvnorm(n = n / nrow(mu),
                            mean = mu[i,],
                            sigma = s[[i]])
    }
    df[k, 3] <- i # bringing along the class it was generated from
  }
  df
}

# calling the function for separate covariance matrices
set.seed(12345)
new_features1 <- gen(mu, s, nrow(features))

# calling the function for pooled covariance matrix
set.seed(12345)
new_features2 <- gen(mu, S, nrow(features))

# plot parameter settings and the plotting
# main parameter for each plot should give a clue of what it shows
par(mfrow = c(2, 2), mar = c(2.4, 2.1, 3, 1), cex.main = 0.9,
  cex.lab = 0.8,
  cex.axis = 0.8)

plot(
  new_features1[, 2],
  new_features1[, 1],
  col = as.factor(new_features1[, 3]),
  pch = 20,
  xlim = c(2, 4.5),
  ylim = c(4, 8.5),
  xlab = NA,
```

```
  ylab = NA,
  main = "Generated w. Non-equal Covariance"
)

plot(
  new_features2[, 2],
  new_features2[, 1],
  col = as.factor(new_features2[, 3]),
  pch = 20,
  xlim = c(2, 4.5),
  ylim = c(4, 8.5),
  xlab = NA,
  ylab = NA,
  main = "Generated w. Equal Covariance"
)

plot(
  data$Sepal.Width,
  data$Sepal.Length,
  col = data$Species,
  xlim = c(2, 4.5),
  ylim = c(4, 8.5),
  pch = 20,
  main = "Original Data"
)

# lda() method from MASS
plot(
  data$Sepal.Width,
  data$Sepal.Length,
  col = m2_pred$class,
  xlim = c(2, 4.5),
  ylim = c(4, 8.5),
  pch = 20,
  main = "Predicted Classes",
  xlab = "Sepal Width",
  ylab = "Sepal Length",
)


# training model using multinom() from nnet
m3 <- multinom(Species~.,data = data[,-c(3:4)])
m3_pred <- predict(m3)

# calculating errors
m3_conf <- table("True" = target, "Predicted" = m3_pred)
m3_err <- 1-(sum(diag(m3_conf))/sum(m3_conf))
# outputting error rates
message("Confusion matrix for multinom()")
m3_conf
message("Error rate for multinom()")
m3_err
```

```r
misclassed <- which(target != m3_pred)

par(mar = c(4.1, 4.1, 2.5, 2))
# plotting the predictions from multinom()
plot(
  data$Sepal.Width,
  data$Sepal.Length,
  col = m3_pred,
  pch = 20,
  cex.main = 0.9,
  cex.lab = 0.8,
  cex.axis = 0.8,
  main = "Predicted Classes using multinom()",
  xlab = "Sepal Width",
  ylab = "Sepal Length"
)

# with circles for misclassified observations
points(
  data$Sepal.Width[misclassed],
  data$Sepal.Length[misclassed],
  pch = 1,
  cex = 1.2,
  col = as.factor(data$Species[misclassed])
)

# Cleanup
rm(list=ls())

###################### END OF ASSIGNMENT 1 ######################
library("tree")

library("e1071")# Naive Bayes

library("rpart")

library("ggplot2")
data = read.csv2("bank-full.csv",header = TRUE,stringsAsFactors = TRUE)

# Data Insights
##  Input variables: 16
##  Total Records: 45,211
##  Records Labeled as yes: 5289 (11%)
##  Records Labeled as No: 39922 (89%)

# Removing "Duration Variable"
drop = c("duration")

data <- data[,!(names(data) %in% drop)]

n <- dim(data)[1]

ncol <- dim(data)[2]
```

```r
##  Selecting Training data (40% of total)
set.seed(12345)

id <- sample (1:n, floor(n * 0.4))

train <- data[id, ]

train_target <- train[,ncol]

##  Selecting the data to validate (30%)

id1 <- setdiff(1:n, id)

set.seed(12345)

id2 <- sample(id1, floor(n * 0.30))

valid <- data[id2, ]

valid_target <- valid[,ncol]

##  Selecting the data to Test (30%)

id3 <- setdiff(id1, id2)

test <- data[id3, ]

test_target <- test[,ncol]

ptime <- proc.time()
fit = tree (y~., data = train)
ptime_fit1 <- proc.time()-ptime

#plot(fit)
#text(fit, pretty = 0)
meanDev_1 <- summary(fit)$dev/summary(fit)$df


yfit = predict(fit, newdata = train, type = "class") # Fit the model on Train
cat("Confusion Matrix: Train \n")# Confusion Matrix: Train
tConfMat <- table("true"= train$y, "predicted" = yfit)
tConfMat
train_error_1<-(sum(tConfMat) - sum(diag(tConfMat)))/sum(tConfMat)
cat(" Misclassifincation Error in train: ",train_error_1,"\n\n")


yfit = predict(fit, newdata = valid, type = "class")# Fit the model on valid
cat("Confusion Matrix: Valid \n")# Confusion Matrix: valid
vConfMat <- table("true" = valid$y, "predicted" = yfit)
vConfMat
valid_error_1<-(sum(vConfMat) - sum(diag(vConfMat)))/sum(vConfMat)
cat(" Misclassifincation Error in Valid: ",valid_error_1,"\n")
```

```
ptime <- proc.time()
fit = tree (y~., data = train,control = tree.control(nobs = nrow(train),minsize = 7000))
ptime_fit2 <- proc.time()-ptime


#(fit)
#text(fit, pretty = 0)
meanDev_2 <- summary(fit)$dev/summary(fit)$df


yfit = predict(fit, newdata = train, type = "class") # Fit the model on Train
cat("Confusion Matrix: Train \n")# Confusion Matrix: Train
tConfMat <- table("true"= train$y, "predicted" = yfit)
tConfMat
train_error_2<-(sum(tConfMat) - sum(diag(tConfMat)))/sum(tConfMat)
cat(" Misclassifincation Error in train: ",train_error_2,"\n\n")


yfit = predict(fit, newdata = valid, type = "class")# Fit the model on valid
cat("Confusion Matrix: Valid \n")# Confusion Matrix: valid
vConfMat <- table("true" = valid$y, "predicted" = yfit)
vConfMat
valid_error_2<-(sum(vConfMat) - sum(diag(vConfMat)))/sum(vConfMat)
cat(" Misclassifincation Error in Valid: ",valid_error_2,"\n")
ptime <- proc.time()
fit = tree (y~., data = train, mindev = 0.0005)
ptime_fit3 <- proc.time()-ptime


#plot(fit)
meanDev_3 <- summary(fit)$dev/summary(fit)$df

yfit = predict(fit, newdata = train, type = "class") # Fit the model on Train
cat("Confusion Matrix: Train \n")# Confusion Matrix: Train
tConfMat <- table("true"= train$y, "predicted" = yfit)
tConfMat
train_error_3 <- (sum(tConfMat) - sum(diag(tConfMat)))/sum(tConfMat)
cat(" Misclassifincation Error in train: ",train_error_3,"\n\n")


yfit = predict(fit, newdata = valid, type = "class")# Fit the model on valid
cat("Confusion Matrix: Valid \n")# Confusion Matrix: valid
vConfMat <- table("true" = valid$y, "predicted" = yfit)
vConfMat
valid_error_3 <- (sum(vConfMat) - sum(diag(vConfMat)))/sum(vConfMat)
cat(" Misclassifincation Error in Valid: ",valid_error_3,"\n")
library(knitr)
kable(data.frame("Tree Setting" = c("Default", "minsize=7000", "mindev=0.0005"),
      "Residual mean deviance" = c(meanDev_1, meanDev_2, meanDev_3),
      "Train Error Rate" = c(train_error_1,train_error_2,train_error_3),
      "Valid Error Rate" = c(valid_error_1,valid_error_2,valid_error_3),
      "Execution Time" = c(ptime_fit1[[3]],ptime_fit2[[3]],ptime_fit3[[3]])),
      caption = "Model Observations"
```

```
)

## Holdout Method
fit = tree (y~., data = train, mindev = 0.0005)

trainScore <- rep(0,50)# Initializing Training Score

validScore <- rep(0,50)# Initializing Validation Score

misclass<-rep(0,50)# Initializing misclassification error

for (i in 2:50){

  prunedTree <- prune.tree(fit, best = i)

  predtr <-predict(prunedTree, newdata = train, type = "tree")

  pred = predict(prunedTree, newdata = valid, type = "tree")

  # Calculation of deviance
  trainScore[i] <- deviance(predtr)

  validScore[i] <- deviance(pred)

  # Calulation of misclassification error

  valid_pred <- predict(prunedTree, newdata = valid, type = "class")
  vConfMat <- table("true" = valid$y, "predicted" = valid_pred)
  misclass[i] <- (sum(vConfMat)-sum(diag(vConfMat)))/sum(vConfMat)

}


terminalNode <- which(validScore==min(validScore[-1]))# optimal Terminal node

plot(2:50, trainScore[2:50], type = "b", pch=19, col = "red", ylim=c(8000,12000),
     main ="Deviance: Train Score vs Validation Score",
     ylab = "Deviance", xlab = "Number of Leaves")
points(2:50, validScore[2:50], type = "b",pch=18, col = "blue")
abline(v = terminalNode, lty = "dashed", col = "gray")
abline(h = min(validScore[-1]), lty = "dashed", col = "gray")

  text(
    paste("Optimal Number Of leaves = ", terminalNode),
    x = terminalNode,
    y =  min(validScore[-1]) + 1000,
    pos = 4,
    cex = 0.7
  )
legend(40, 11000, legend=c("TrainScore", "ValidScore"),
       col=c("red", "blue"), lty=1:2, cex=0.8)

ptime <- proc.time()
```

```
finalTree = prune.tree(fit, best = terminalNode)# Pruning the tree to optimal nodes
ptime_fit4 <- proc.time()-ptime

cat("Optimal Number Of Leaves: ",terminalNode,"\n")
cat("Most Contributing Variables: ",as.character(summary(finalTree)[[3]]),"\n")


yfit = predict(finalTree, newdata = test, type="class")# Fit the Test to pruned tree
cat("Confusion Matrix: Test \n")# Confusion Matrix: Test
prunedConfMat <- table("true" = test$y, "predicted" = yfit)
prunedConfMat
pruned_error <- (sum(prunedConfMat)- sum(diag(prunedConfMat)))/sum(prunedConfMat)
cat(" Misclassifincation Error in Test: ",pruned_error,"\n")

#summary(finalTree)
plot(finalTree,type=c("uniform"), title = "Optimal Tree")
text(finalTree,pretty = 1)
L = matrix(c(0,5,1,0), ncol=2)# Loss Matrix
# Using RPART pACKAGE

#ptime <- proc.time()
#fit = rpart(formula = y~., data = train, parms = list(loss=L))
#ptime_fit5 <- proc.time()-ptime
#
#yfit = predict(fit, newdata = test, type="class")
#
## Confusion Matrix: Test
#tConfMat <- table("true" = test$y, "predicted" = yfit)
##tConfMat
#lossM_error<-(sum(tConfMat)- sum(diag(tConfMat)))/sum(tConfMat)
#cat(" Misclassifincation Error in Test with Loss Matrix: ",lossM_error,"\n")

# Using formula
yfit_new = predict(finalTree, newdata = test, type="vector")

vect <- cbind("No"= yfit_new[,1]*1, "yes" = yfit_new[,2]*5) # Assigning the weight as per loss matrix


new_class <- factor(ifelse(vect[,1]>vect[,2],"no","yes"),levels=c("no","yes"))

# Confusion Matrix: Test
tConfMat <- table("true" = test$y, "predicted" = new_class)
#tConfMat
lossM_error<-(sum(tConfMat)- sum(diag(tConfMat)))/sum(tConfMat)
cat(" Misclassifincation Error in Test with Loss Matrix: ",lossM_error,"\n")


cat("Optimal Tree Confusion Matrix \n")
prunedConfMat

cat("5-1 Loss Matrix Tree Confusion Matrix \n")
tConfMat
```

```r
# Naive Bayes fitting
cat("Classification Using Naive Bayes \n")
naiveFit <- naiveBayes(y ~ ., data = train)
nPred <- predict(naiveFit, newdata = train) # Fitting training data
nTrainConfMat <- table("true"=train$y, "predicted"=nPred) # Confusion Matrix
cat("Confusion Matrix: Train \n")
print(nTrainConfMat)
cat(" Misclassifincation Error in Train: ",
    1- sum(diag(nTrainConfMat))/sum(nTrainConfMat),"\n")

nPred <- predict(naiveFit, newdata = test) # Fitting test data
nTrainConfMat <- table("true" = test$y, "predicted"=nPred) # Confusion Matrix
cat("Confusion Matrix: Test \n")
print(nTrainConfMat)
cat(" Misclassifincation Error in Test: ",
    1- sum(diag(nTrainConfMat))/sum(nTrainConfMat),"\n")

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

pi <- seq(0.05,0.95,0.05)# initializing pi

treeTpr <- c() # optimal tree True Positive Rate
treeFpr <- c() # Optimal tree False Positive Rate

nTpr <- c()#  Naive Bayes True Positive Rate
nFpr <- c()# Naive Bayes False Positive Rate

## Getting optimal tree predictions
treePred  <- as.data.frame(predict(finalTree,newdata = test, type="vector"))

#table(test$y)

## Naive Bayes Prediction
nPred <- predict(naiveFit, newdata = test, type = "raw")

for(i in 1:length(pi)){

  treePrinciple <- factor(ifelse(treePred[,2]>pi[i],"yes","no"), levels=c("no","yes"))
  treeConfMatrix <- table("true" = test$y, "predicted" = treePrinciple)

  treeTpr[i] <- treeConfMatrix[2,2]/sum(treeConfMatrix[2,])
  treeFpr[i] <- treeConfMatrix[1,2]/sum(treeConfMatrix[1,])

  nPrinciple <- factor(ifelse(nPred[,2]>pi[i],"yes","no"), levels=c("no","yes"))
  nConfMatrix <- table("true"=test$y,"predicted"=nPrinciple)

  nTpr[i] <- nConfMatrix[2,2]/sum(nConfMatrix[2,])
  nFpr[i] <- nConfMatrix[1,2]/sum(nConfMatrix[1,])

}
```

```
## ROC Plot

ggplot()+
  geom_line(aes(x=seq(0,1,by=0.5),y=seq(0,1,by=0.5)), linetype="dashed",color="grey")+
  geom_line(aes(x = treeFpr, y= treeTpr, color = "Optimal Tree"),linetype="dashed")+
  geom_point(aes(x = treeFpr, y= treeTpr,color = "Optimal Tree"))+
  geom_line(aes(x = nFpr, y= nTpr, color = "Naive Bayes"),linetype="dashed")+
  geom_point(aes(x = nFpr, y= nTpr, color = "Naive Bayes"))+
  ylab("TPR") + xlab("FPR") + ggtitle("ROC Curve for Naive Bayes and Optimal Tree")+
  theme(plot.title = element_text(hjust = 0.5))
#3.1
library("dplyr")
library("ggplot2")

theme_set(theme_bw())
theme_update(plot.title = element_text(size=16,face="bold",hjust = 0.5),
             axis.text = element_text(size=10,colour="black"),
             axis.title = element_text(size=15),
             legend.title = element_text(size=15))
#3.1
com <- read.csv("communities.csv")
com_scaled<- com %>% select(-ViolentCrimesPerPop) %>% scale() %>% data.frame()
e <- eigen(cov(com_scaled))
M <- which(cumsum(e$values/sum(e$values))>=0.95)[1]
#3.1
ggplot() + geom_point(aes(x=1:100, y=cumsum(e$values/sum(e$values))), color="orange") +
geom_hline(yintercept=0.95,color="red",linetype="dashed") +
labs(title="Proportion of variation", x="Number of PC", y="Proportion of variation")

# What is the proportion of variation explained by each of the first two principal components?
(e$values[1] + e$values[2])/sum(e$values) # ~42%
#3.2
#res <- prcomp(com_scaled) # obs! need to use princomp not prcomp!
#U <- res$rotation
res <- princomp(com_scaled)
U <- res$loadings
colnames(res$scores) <- paste("PC",1:100,sep="") # renaming column for quick fix

#3.2
ggplot() + geom_point(aes(y=(U[,1]),x=1:100), color="orange")  + labs(title="PC1 vs Features", x="Featu


knitr::kable( sort(abs(U[,1]),decreasing = TRUE )[1:5])

#3.2
PCA_data <- data.frame(res$scores,ViolentCrimesPerPop=com$ViolentCrimesPerPop)
ggplot(PCA_data) + geom_point(aes(x=PC1,y=PC2,color=cut_interval(ViolentCrimesPerPop,3)), alpha=0.5) + l
#3.2
model <- lm(ViolentCrimesPerPop~poly(PC1,degree=2),PCA_data)

pred <- predict(model)
ggplot(PCA_data) + geom_point(aes(x=PC1,y=ViolentCrimesPerPop),alpha=0.7, color="orange") +
geom_point(aes(x=PC1,y=pred),alpha=0.5,color="dark red") +
```

```
labs(title="ViolentCrimesPerPop vs PC1 \n with fitted prediction")
#3.4
library("boot")
data2 <- PCA_data[order(PCA_data$PC1),]
fit <- lm(ViolentCrimesPerPop~poly(PC1,degree=2),data2)
rng = function(data,model){
n <- nrow(data)
data1<- data.frame(ViolentCrimesPerPop=data$ViolentCrimesPerPop,PC1=data$PC1)
data1$ViolentCrimesPerPop <- rnorm(n,predict(model, newdata=data1),sd(model$residuals))
return(data1)}


f1=function(data1){
res=lm(ViolentCrimesPerPop~poly(PC1,degree=2),data1)
ViolentCrimesPerPopP=predict(res,newdata=data2)  ## ... global variable?
return(ViolentCrimesPerPopP)}
resy <- boot(data2, statistic=f1, R=2000, mle = fit,ran.gen=rng  , sim="parametric")
es <- envelope(resy)


# 3.4
ggplot(data2) + geom_point(aes(x=PC1, y=ViolentCrimesPerPop), color="orange") +
                geom_line(aes(x=PC1, y= es$point[1,]),color="blue") +
                geom_line(aes(x=PC1, y= es$point[2,]),color="blue") +
                geom_line(aes(x=PC1, y= predict(fit)), color="dark red") + labs(title="ViolentCrimesPerP
# 3.4

f2=function(data1){
res=lm(ViolentCrimesPerPop~poly(PC1,degree=2),data1)
ViolentCrimesPerPopP=predict(res,newdata=data2)
n=length(data2$ViolentCrimesPerPop)
ViolentCrimesPerPopP=rnorm(n,ViolentCrimesPerPopP, sd(fit$residuals))
return(ViolentCrimesPerPopP)
}

res=boot(data2, statistic=f2, R=10000, mle=  fit,ran.gen=rng, sim="parametric")
es <- envelope(res)
# 3.4
ggplot(data2) + geom_point(aes(x=PC1, y=ViolentCrimesPerPop), color="orange") +
                geom_line(aes(x=PC1, y= es$point[1,]),color="blue") +
                geom_line(aes(x=PC1, y= es$point[2,]),color="blue") +
                geom_line(aes(x=PC1, y= predict(fit)), color="dark red") + ylim(c(0,1)) + labs(title="Vi
```