CENG 4515 DATA SCIENCE AND ANALYTICS

Final Project

Ayşe Ceren Çiçek & Gizem Kurnaz

01/13/2021

TABLE OF CONTENTS

1) Dataset

Dataset: http://www2.informatik.uni-freiburg.de/~cziegler/BX/

The dataset contains three csv files which are ratings, users and books.

Ratings.csv: Contains the book rating information. Ratings (Book-Rating) are either explicit, expressed on a scale from 1-10 (higher values denoting higher appreciation), or implicit, expressed by 0. This file contains 999,999 rows and 3 columns:

- User.ID: User identification ID
- ISBN: ISBN Id for the book
- Book.Rating: Rating for each book given by user

Users.csv: Contains the users. User IDs (User-ID) have been anonymized and map to integers. Demographic data is provided (Location, Age) if available. Otherwise, these fields contain NULL-values. This file contains 278,858 rows and 3 columns:

- User.ID: Unique identification ID for the user
- Location: Location of the user
- Age: Age of the user

Books.csv: Books are identified by their respective ISBN. Invalid ISBNs have already been removed from the dataset. Moreover, some content-based information is given (Book-Title, Book-Author, Year-Of-Publication, Publisher), obtained from Amazon Web Services. In case of several authors, only the first is provided. URLs linking to cover images are also given, appearing in three different flavours (Image-URL-S, Image-URL-M, Image-URL-L), i.e., small, medium, large. These URLs point to the Amazon web site. This file contains 271,379 rows and 8 columns:

- ISBN: ISBN ID for each book
- Book.Title: Title of the book
- Book.Author: Author name
- Year.Of.Publication: Publication year
- Publisher: Name of the publisher
- Image.URL.S: Small sized image URL of the book
- Image.URL.M: Medium sized image URL of the book
- Image.URL.L: Large sized image URL of the book

Importing libraries

```
library(funModeling)
library(tidyverse)
library(dplyr)
library(data.table)
library(stringr)
library(ggplot2)
library(Hmisc)
library(missForest)
library(mice)
library(ROSE)
library(cluster)
library(ClusterR)
library(caTools)
library(caret)
library(knitr)
library(Amelia)
library(rpart)
library(ggcorrplot)
library(factoextra)
library(class)
```

Loading dataset

We will load 3 data files first.

```
ratings <- fread("BookDataset/ratings.csv", sep = ";")
head(ratings, n=5)
```

```
##    User-ID       ISBN Book-Rating
## 1:  276725 034545104X           0
## 2:  276726  155061224           5
## 3:  276727  446520802           0
## 4:  276729 052165615X           3
## 5:  276729  521795028           6
```

```
users <- fread("BookDataset/users.csv", sep = ";")
head(users, n=5)
```

```
##    User-ID                            Location  Age
## 1:       1                  nyc, new york, usa NULL
## 2:       2           stockton, california, usa   18
## 3:       3     moscow, yukon territory, russia NULL
## 4:       4           porto, v.n.gaia, portugal   17
## 5:       5 farnborough, hants, united kingdom NULL
```

```
books <- fread("BookDataset/books.csv", sep = ";")
head(books, n=5)
```

```
##          ISBN
## 1: 0195153448
## 2: 0002005018
## 3: 0060973129
## 4: 0374157065
## 5: 0393045218
##                                                                                Book-Titl
## 1:                                                                          Classical Mythology
## 2:                                                                                 Clara Callan
## 3:                                                                        Decision in Normandy
## 4: Flu: The Story of the Great Influenza Pandemic of 1918 and the Search for the Virus That Caused It
## 5:                                                                        The Mummies of Urumchi
##             Book-Author Year-Of-Publication                 Publisher
## 1:   Mark P. O. Morford                2002    Oxford University Press
## 2: Richard Bruce Wright                2001      HarperFlamingo Canada
## 3:         Carlo D'Este                1991            HarperPerennial
## 4:     Gina Bari Kolata                1999       Farrar Straus Giroux
## 5:      E. J. W. Barber                1999 W. W. Norton &amp;amp; Company
##                                                Image-URL-S
## 1: http://images.amazon.com/images/P/0195153448.01.THUMBZZZ.jpg
## 2: http://images.amazon.com/images/P/0002005018.01.THUMBZZZ.jpg
## 3: http://images.amazon.com/images/P/0060973129.01.THUMBZZZ.jpg
## 4: http://images.amazon.com/images/P/0374157065.01.THUMBZZZ.jpg
## 5: http://images.amazon.com/images/P/0393045218.01.THUMBZZZ.jpg
##                                                Image-URL-M
## 1: http://images.amazon.com/images/P/0195153448.01.MZZZZZZZ.jpg
## 2: http://images.amazon.com/images/P/0002005018.01.MZZZZZZZ.jpg
## 3: http://images.amazon.com/images/P/0060973129.01.MZZZZZZZ.jpg
## 4: http://images.amazon.com/images/P/0374157065.01.MZZZZZZZ.jpg
## 5: http://images.amazon.com/images/P/0393045218.01.MZZZZZZZ.jpg
##                                                Image-URL-L
## 1: http://images.amazon.com/images/P/0195153448.01.LZZZZZZZ.jpg
## 2: http://images.amazon.com/images/P/0002005018.01.LZZZZZZZ.jpg
```

```
## 3: http://images.amazon.com/images/P/0060973129.01.LZZZZZZZ.jpg
## 4: http://images.amazon.com/images/P/0374157065.01.LZZZZZZZ.jpg
## 5: http://images.amazon.com/images/P/0393045218.01.LZZZZZZZ.jpg
```

2) Exploratory data analysis

Let's check the unique number of books & users.

```
n_distinct(ratings$`User-ID`)
```

```
## [1] 91407
```

```
n_distinct(books$ISBN)
```

```
## [1] 271379
```

Merging dataframes

We will first merge ratings and users dataframes based on User-ID column.

```
dataset = merge(ratings, users, by.x = "User-ID", by.y = "User-ID")
```

Now we will retrieve country information from the Location column which contains state, city, country information.

```
dataset$Country <- sub('.*,\\s*','', dataset$Location)
dataset <- dataset[(which(nchar(dataset$Country) >= 2)),]
head(dataset, n=5)
```

```
##    User-ID      ISBN Book-Rating                  Location  Age Country
## 1:       2 195153448           0 stockton, california, usa   18     usa
## 2:       7  34542252           0       washington, dc, usa NULL     usa
## 3:       8   2005018           5  timmins, ontario, canada NULL  canada
## 4:       8  60973129           0  timmins, ontario, canada NULL  canada
## 5:       8 374157065           0  timmins, ontario, canada NULL  canada
```

We will merge the last dataframe with our new one based on ISBN.

```
dataset <- merge(dataset, books, by.x = "ISBN", by.y = "ISBN")
head(dataset, n=5)
```

```
##          ISBN User-ID Book-Rating                             Location  Age
## 1: 000104687X   23902           6     london, england, united kingdom NULL
## 2: 000104799X   28204           7     south ohio, nova scotia, canada   61
## 3: 000104799X  166596           8 greenwood, british columbia, canada   57
## 4: 000123207X  198711           0       little canada, minnesota, usa   62
## 5: 000160418X   10067           7            watton, norfolk, england   61
##           Country                                          Book-Title
```

```
## 1: united kingdom T.S. Eliot Reading \\"The Wasteland\\" and Other Poems
## 2:         canada                                                Monk’s-hood
## 3:         canada                                                Monk’s-hood
## 4:            usa                             Paddington’s Birthday Party
## 5:        england                          The Clue in the Crumbling Wall
##      Book-Author Year-Of-Publication               Publisher
## 1:    T.S. Eliot                1993 HarperCollins Publishers
## 2:  Ellis Peters                1994 HarperCollins Publishers
## 3:  Ellis Peters                1994 HarperCollins Publishers
## 4:  Michael Bond                1942 HarperCollins Publishers
## 5: Carolyn Keene                1984 HarperCollins Publishers
##                                                    Image-URL-S
## 1: http://images.amazon.com/images/P/000104687X.01.THUMBZZZ.jpg
## 2: http://images.amazon.com/images/P/000104799X.01.THUMBZZZ.jpg
## 3: http://images.amazon.com/images/P/000104799X.01.THUMBZZZ.jpg
## 4: http://images.amazon.com/images/P/000123207X.01.THUMBZZZ.jpg
## 5: http://images.amazon.com/images/P/000160418X.01.THUMBZZZ.jpg
##                                                    Image-URL-M
## 1: http://images.amazon.com/images/P/000104687X.01.MZZZZZZZ.jpg
## 2: http://images.amazon.com/images/P/000104799X.01.MZZZZZZZ.jpg
## 3: http://images.amazon.com/images/P/000104799X.01.MZZZZZZZ.jpg
## 4: http://images.amazon.com/images/P/000123207X.01.MZZZZZZZ.jpg
## 5: http://images.amazon.com/images/P/000160418X.01.MZZZZZZZ.jpg
##                                                    Image-URL-L
## 1: http://images.amazon.com/images/P/000104687X.01.LZZZZZZZ.jpg
## 2: http://images.amazon.com/images/P/000104799X.01.LZZZZZZZ.jpg
## 3: http://images.amazon.com/images/P/000104799X.01.LZZZZZZZ.jpg
## 4: http://images.amazon.com/images/P/000123207X.01.LZZZZZZZ.jpg
## 5: http://images.amazon.com/images/P/000160418X.01.LZZZZZZZ.jpg
```

Changing column names

We have 13 columns and we will change name of the some of them.

```
colnames(dataset)
```

```
##  [1] "ISBN"                "User-ID"             "Book-Rating"
##  [4] "Location"            "Age"                 "Country"
##  [7] "Book-Title"          "Book-Author"         "Year-Of-Publication"
## [10] "Publisher"           "Image-URL-S"         "Image-URL-M"
## [13] "Image-URL-L"
```

```
colnames(dataset)[which(colnames(dataset) %in%  c("User-ID", "Book-Rating", "Book-Title", "Book-Author"
colnames(dataset)
```

```
##  [1] "ISBN"                "User.ID"             "Book.Rating"
##  [4] "Location"            "Age"                 "Country"
##  [7] "Book.Title"          "Book.Author"         "Year.Of.Publication"
## [10] "Publisher"           "Image.URL.S"         "Image.URL.M"
## [13] "Image.URL.L"
```

Now we can check the classes of those columns.

```
sapply(dataset, class)
```

```
##                ISBN             User.ID         Book.Rating            Location
##         "character"           "integer"           "integer"         "character"
##                 Age             Country          Book.Title         Book.Author
##         "character"         "character"         "character"         "character"
## Year.Of.Publication           Publisher         Image.URL.S         Image.URL.M
##           "integer"         "character"         "character"         "character"
##         Image.URL.L
##         "character"
```

Except for the Age column, it seems like others' have appropriate classes. We will turn Age column to numeric.

```
dataset <- transform(dataset, Age = as.numeric(Age))
head(dataset, n=5)
```

```
##          ISBN User.ID Book.Rating                            Location Age
## 1: 000104687X   23902           6     london, england, united kingdom  NA
## 2: 000104799X   28204           7     south ohio, nova scotia, canada  61
## 3: 000104799X  166596           8 greenwood, british columbia, canada  57
## 4: 000123207X  198711           0       little canada, minnesota, usa  62
## 5: 000160418X   10067           7            watton, norfolk, england  61
##           Country                                              Book.Title
## 1: united kingdom T.S. Eliot Reading \\"The Wasteland\\" and Other Poems
## 2:         canada                                              Monk's-hood
## 3:         canada                                              Monk's-hood
## 4:            usa                                Paddington's Birthday Party
## 5:        england                             The Clue in the Crumbling Wall
##      Book.Author Year.Of.Publication               Publisher
## 1:    T.S. Eliot                1993 HarperCollins Publishers
## 2:  Ellis Peters                1994 HarperCollins Publishers
## 3:  Ellis Peters                1994 HarperCollins Publishers
## 4:  Michael Bond                1942 HarperCollins Publishers
## 5: Carolyn Keene                1984 HarperCollins Publishers
##                                                    Image.URL.S
## 1: http://images.amazon.com/images/P/000104687X.01.THUMBZZZ.jpg
## 2: http://images.amazon.com/images/P/000104799X.01.THUMBZZZ.jpg
## 3: http://images.amazon.com/images/P/000104799X.01.THUMBZZZ.jpg
## 4: http://images.amazon.com/images/P/000123207X.01.THUMBZZZ.jpg
## 5: http://images.amazon.com/images/P/000160418X.01.THUMBZZZ.jpg
##                                                    Image.URL.M
## 1: http://images.amazon.com/images/P/000104687X.01.MZZZZZZZ.jpg
## 2: http://images.amazon.com/images/P/000104799X.01.MZZZZZZZ.jpg
## 3: http://images.amazon.com/images/P/000104799X.01.MZZZZZZZ.jpg
## 4: http://images.amazon.com/images/P/000123207X.01.MZZZZZZZ.jpg
## 5: http://images.amazon.com/images/P/000160418X.01.MZZZZZZZ.jpg
##                                                    Image.URL.L
## 1: http://images.amazon.com/images/P/000104687X.01.LZZZZZZZ.jpg
## 2: http://images.amazon.com/images/P/000104799X.01.LZZZZZZZ.jpg
## 3: http://images.amazon.com/images/P/000104799X.01.LZZZZZZZ.jpg
## 4: http://images.amazon.com/images/P/000123207X.01.LZZZZZZZ.jpg
## 5: http://images.amazon.com/images/P/000160418X.01.LZZZZZZZ.jpg
```

Check for any duplications

```
sum(duplicated(dataset))
```

```
## [1] 0
```

Also, we will turn ID columns into factors to make easier usage in the future.

```
dataset$User.ID <- as.factor(dataset$User.ID)
dataset$ISBN <- as.factor(dataset$ISBN)
summary(dataset)
```

```
##          ISBN           User.ID        Book.Rating       Location
##  044023722X:   551   11676  :  2899   Min.   : 0.000   Length:167460
##  067976402X:   518   98391  :  1064   1st Qu.: 0.000   Class :character
##  059035342X:   475   198711 :   872   Median : 0.000   Mode  :character
##  044021145X:   450   153662 :   859   Mean   : 3.126
##  1400034779:   357   35859  :   771   3rd Qu.: 7.000
##  044022165X:   333   278418 :   588   Max.   :10.000
##  (Other)   :164776   (Other):160407
##       Age           Country           Book.Title        Book.Author
##  Min.   :  0.00   Length:167460      Length:167460      Length:167460
##  1st Qu.: 28.00   Class :character   Class :character   Class :character
##  Median : 34.00   Mode  :character   Mode  :character   Mode  :character
##  Mean   : 36.33
##  3rd Qu.: 44.00
##  Max.   :244.00
##  NA's   :40398
##  Year.Of.Publication  Publisher         Image.URL.S        Image.URL.M
##  Min.   :   0        Length:167460      Length:167460      Length:167460
##  1st Qu.:1994        Class :character   Class :character   Class :character
##  Median :1998        Mode  :character   Mode  :character   Mode  :character
##  Mean   :1944
##  3rd Qu.:2001
##  Max.   :2037
##
##  Image.URL.L
##  Length:167460
##  Class :character
##  Mode  :character
##
##
##
##
```

Number of ratings by users

```
rating.count.users <- dataset %>% count(User.ID)
head(rating.count.users, n=5)
```

```
##    User.ID n
## 1:       8 7
## 2:      10 1
## 3:      12 1
## 4:      22 4
## 5:      32 2
```

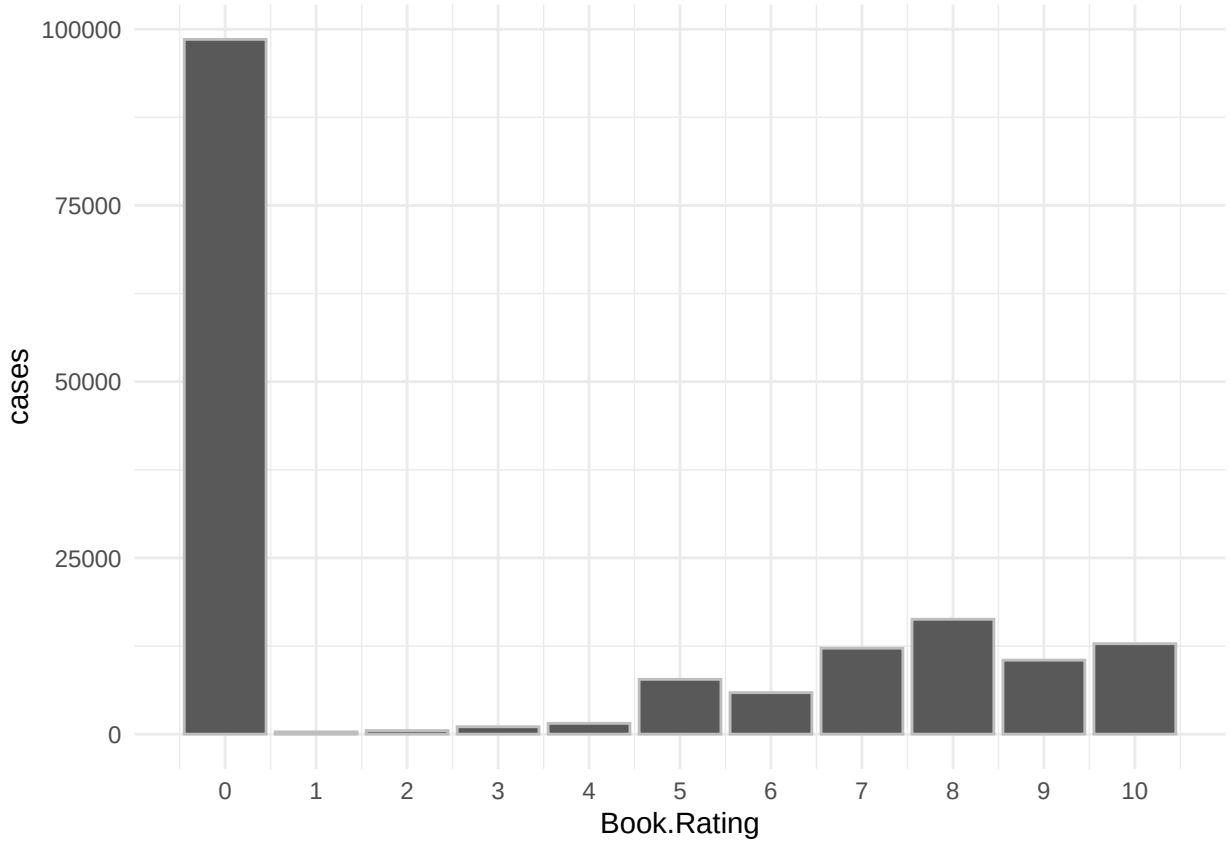
From the table below, we can see that the maximum number of rating by a user is 2899 and the mean is about 5.196.

```
summary(rating.count.users)
```

```
##     User.ID            n
##  8      :    1   Min.   :   1.000
##  10     :    1   1st Qu.:   1.000
##  12     :    1   Median :   1.000
##  22     :    1   Mean   :   5.196
##  32     :    1   3rd Qu.:   3.000
##  36     :    1   Max.   :2899.000
##  (Other):32222
```

Let's visualize number of ratings.

```
dataset %>%
  group_by(Book.Rating) %>%
  summarize(cases = n()) %>%
  ggplot(aes(Book.Rating, cases)) + geom_col(color="gray") +
  theme_minimal() + scale_x_continuous(breaks = 0:10)
```



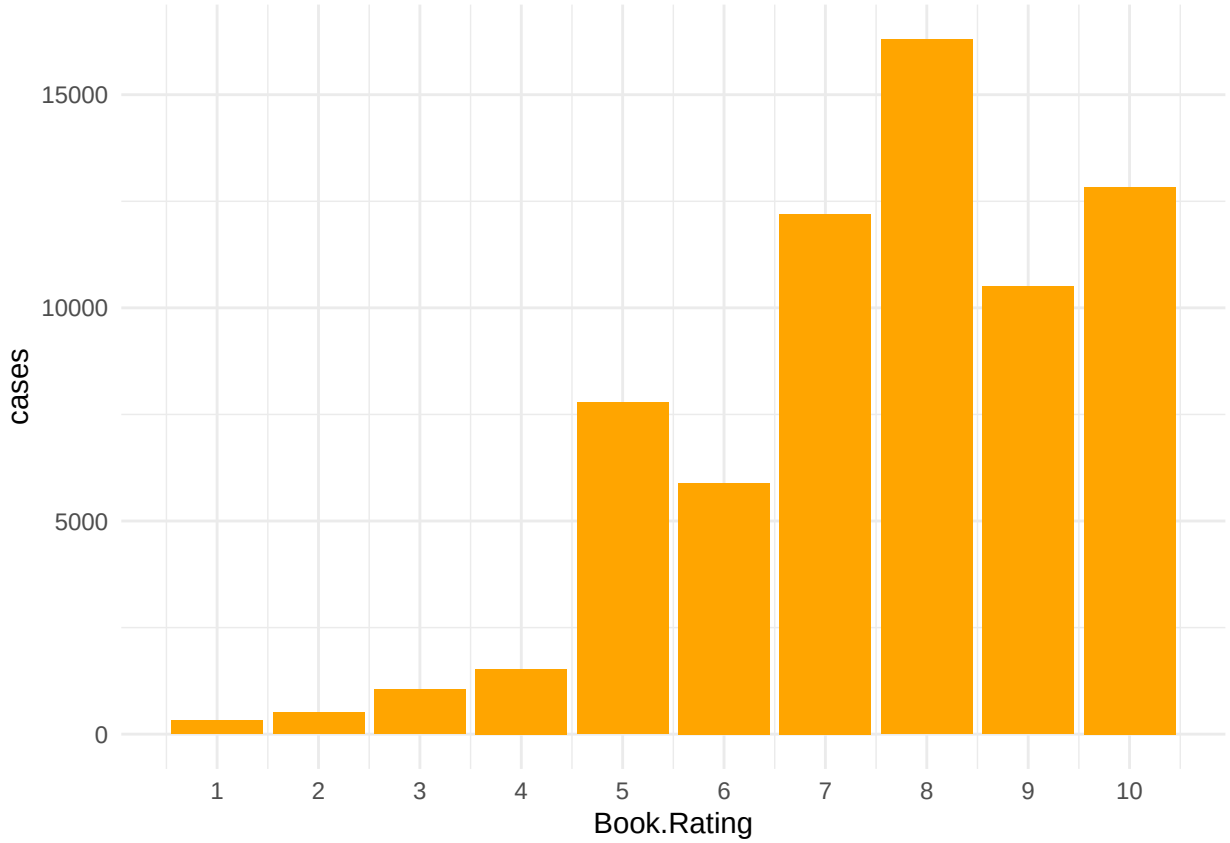
There are a lot of zero values. It might indicate the absence of rating. So we will remove those rows.

```
dataset <- dataset[dataset$Book.Rating!= 0, ]
```

3) Visualization techniques

So the rating change as seen below. 8 is the most used voting value.

```
dataset %>%
  group_by(Book.Rating) %>%
  summarize(cases = n()) %>%
  ggplot(aes(Book.Rating, cases)) + geom_col(fill="orange") +
  theme_minimal() + scale_x_continuous(breaks = 0:10)
```



We do not have a categorical variable but we will create one based on average votes for each book and classify them as yes if its average rating is greater than the overall mean.

Now we will get average rating for each book.

```
books.rating.mean <- aggregate(Book.Rating ~ ISBN, dataset, mean)
head(books.rating.mean, n=5)
```

```
##         ISBN Book.Rating
## 1 000104687X         6.0
## 2 000104799X         7.5
## 3 000160418X         7.0
## 4 000215871X         7.0
## 5 000222674X         9.0
```

The highest rate a book has is 10. The mean value is 7.529.

```
summary(books.rating.mean)
```

```
##          ISBN        Book.Rating
##  000104687X:    1   Min.   : 1.000
##  000104799X:    1   1st Qu.: 6.500
##  000160418X:    1   Median : 8.000
##  000215871X:    1   Mean   : 7.529
##  000222674X:    1   3rd Qu.: 9.000
##  000223257X:    1   Max.   :10.000
##  (Other)   :37009
```

Now we will get the books which has more average rating than the mean and classify them as above the mean or not.

```
nrow(books.rating.mean)
```

```
## [1] 37015
```

```
sum(books.rating.mean$Book.Rating > 7.529)
```

```
## [1] 20055
```

Let order it in descending order.

```
book.rating.mean <-  books.rating.mean[order(-books.rating.mean$Book.Rating),]
```

Get the books which has higher rating more than the mean and classify by yes or no.

```
books.rating.mean <- books.rating.mean[books.rating.mean$Book.Rating > 7.692,]
```

```
dataset$Rating.Count.Above.Mean <- ifelse(dataset$ISBN %in% books.rating.mean$ISBN, "Yes", "No")
```

36151 books has count more than mean and 32748 are below the mean.

```
nrow(dataset[dataset$Rating.Count.Above.Mean == "Yes",])
```

```
## [1] 36151
```

```
nrow(dataset[dataset$Rating.Count.Above.Mean == "No",])
```

```
## [1] 32748
```

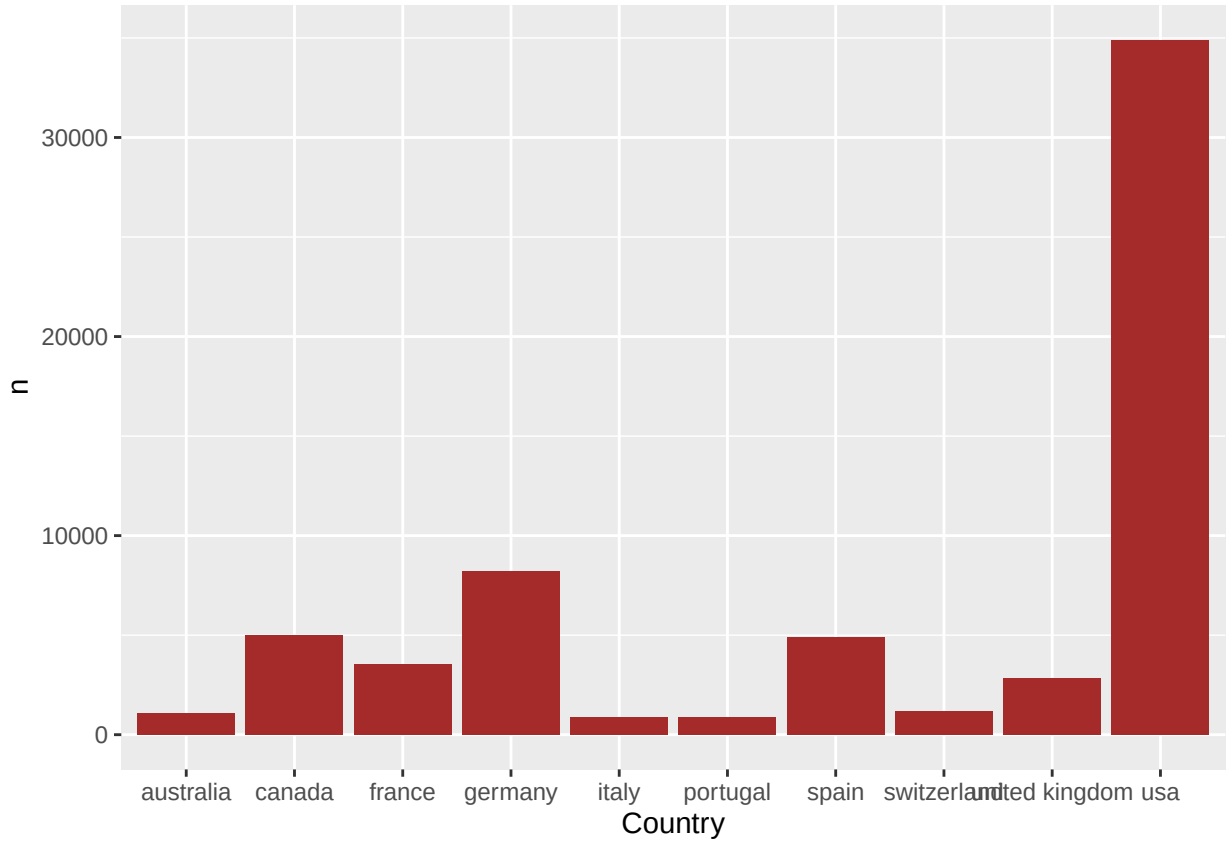
Top 10 countries

Top 10 countries that are users from.

```
countries <- dataset %>% count(Country)
countries <- countries[!(countries$Country=="n/a")]
countries <- countries[order(-n)][1:10]
head(countries, n=10)
```

```
##            Country     n
##  1:            usa 34903
##  2:        germany  8198
##  3:         canada  5002
##  4:          spain  4902
##  5:         france  3555
##  6: united kingdom  2820
##  7:    switzerland  1186
##  8:      australia  1101
##  9:       portugal   892
## 10:          italy   860
```

```
countries %>%
ggplot(aes(Country, n)) +
  geom_col(fill="brown")
```



The users are mostly from the USA. The second country is Germany and Canada is the third one.

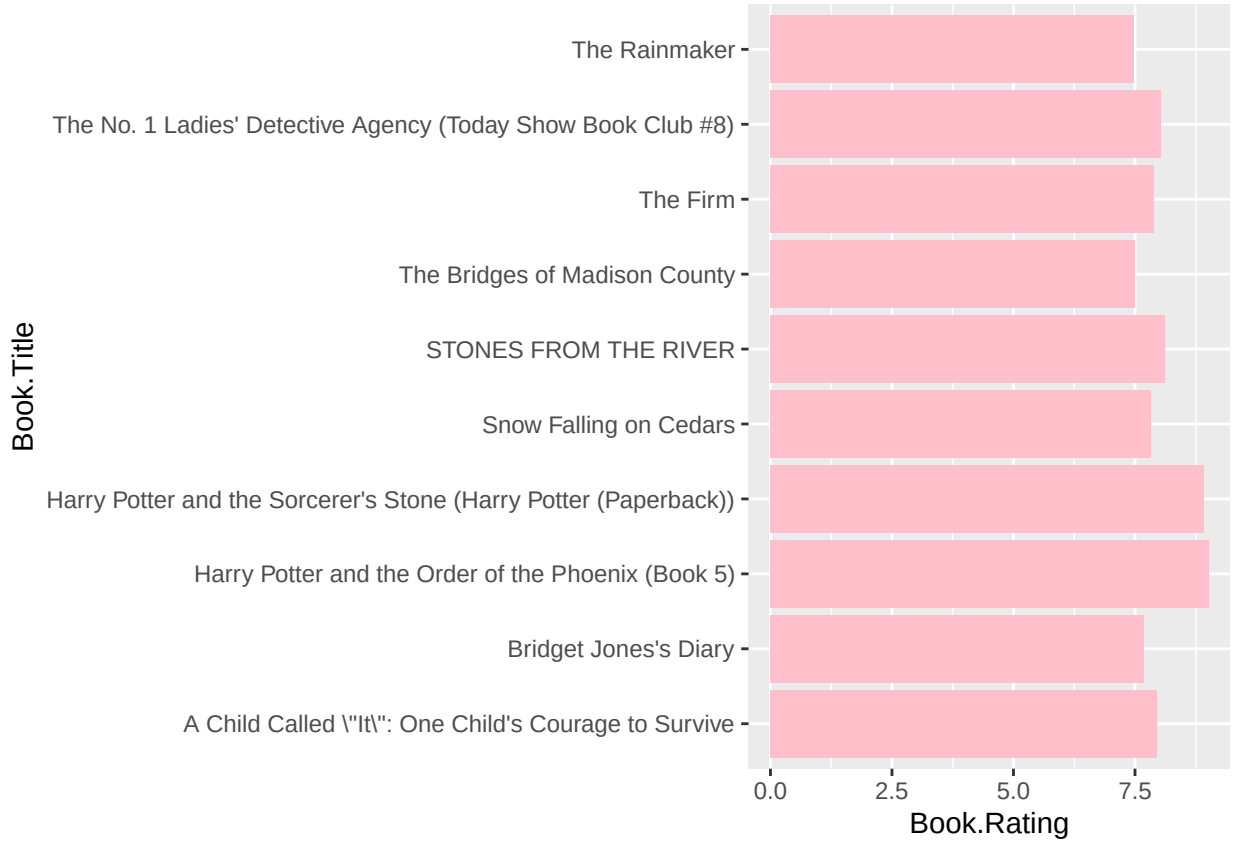
Top 10 highest rated books which has more than 100 votes

```
ratings.book <- dataset %>% group_by(ISBN) %>% filter(n()>100)
ratings.mean <- setorder(setDT(ratings.book)[, .(Book.Rating = mean(Book.Rating)), by = Book.Title], -B
ratings.mean
```

```
##                                                    Book.Title
##  1:              Harry Potter and the Order of the Phoenix (Book 5)
##  2: Harry Potter and the Sorcerer's Stone (Harry Potter (Paperback))
##  3:                                          STONES FROM THE RIVER
##  4:     The No. 1 Ladies' Detective Agency (Today Show Book Club #8)
##  5:          A Child Called \\"It\\": One Child's Courage to Survive
##  6:                                                      The Firm
##  7:                                        Snow Falling on Cedars
##  8:                                         Bridget Jones's Diary
##  9:                                The Bridges of Madison County
## 10:                                                 The Rainmaker
```

```
##     Book.Rating
##  1:    9.011765
##  2:    8.922780
##  3:    8.112150
##  4:    8.026490
##  5:    7.957265
##  6:    7.883978
##  7:    7.833333
##  8:    7.681818
##  9:    7.500000
## 10:    7.474576
```

```
ratings.mean %>%
ggplot(aes(Book.Rating, Book.Title)) +
  geom_col(fill='pink')
```



Harry Potter and the Order of the Phoenix (Book 5) is the one that has the highest rating in books which has more votes than 100. And Harry Potter and the Sorcerer's Stone (Harry Potter (Paperback)) is the second one with an 8.92 rates.

Exploration of year of publication

Minimum value of year is 0 which means we do not have the year information of that book. We will replace those 0 values with NA.

```
summary(dataset$Year.Of.Publication)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##       0    1994    1999    1938    2001    2037
```

```
dataset$Year.Of.Publication[dataset$Year.Of.Publication == 0] <- NA
```

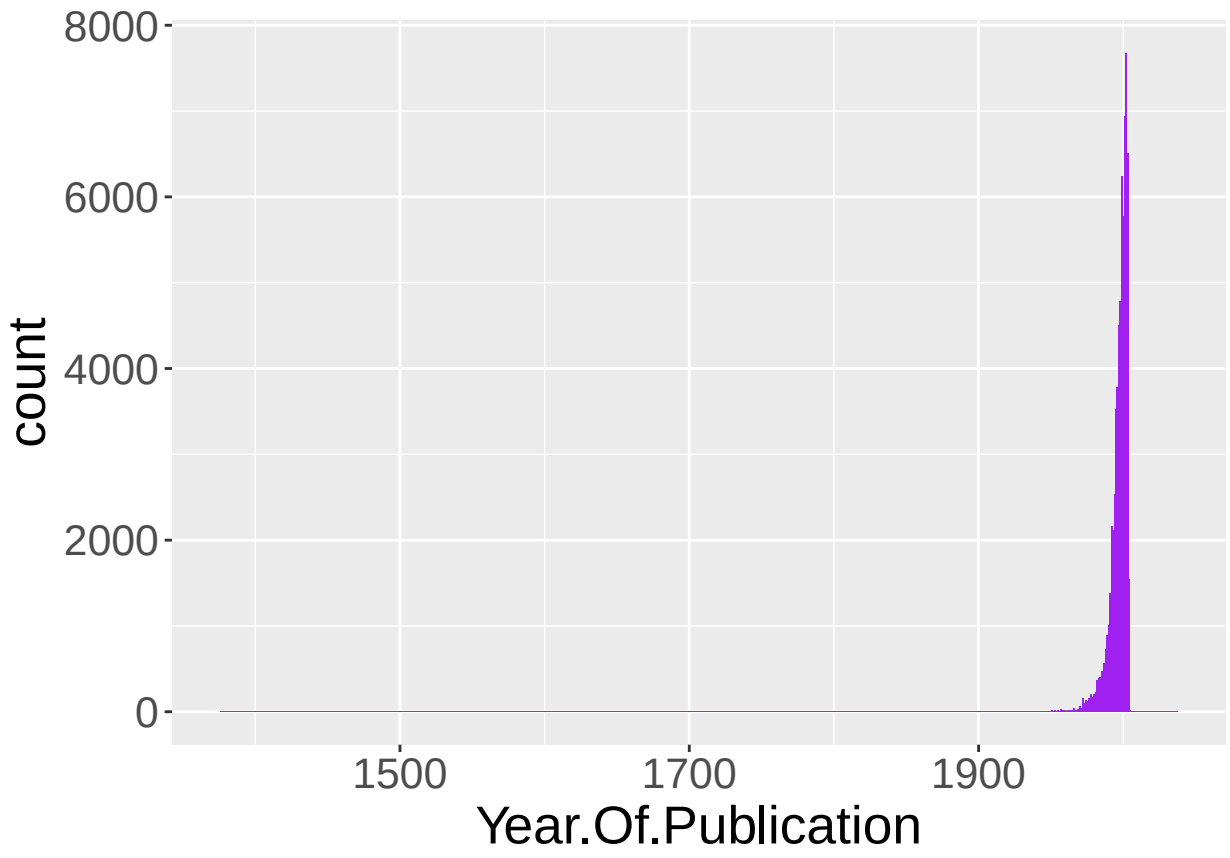
```
summary(dataset$Year.Of.Publication)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
##    1376    1995    1999    1997    2001    2037    2024
```

The graph shows the year of publication data of books.

```
year_hist <- dataset %>%
    ggplot(aes(Year.Of.Publication)) +
    geom_histogram(binwidth=1, fill='purple') +
    theme(text = element_text(size = 20))

year_hist
```

```
## Warning: Removed 2024 rows containing non-finite values (stat_bin).
```



Exploration of Authors

Check how many unique author values we have

```
length(dataset$Book.Author)
```

```
## [1] 68899
```

```
n_distinct(dataset$Book.Author)
```

```
## [1] 21926
```

There are 21,926 unique authors.

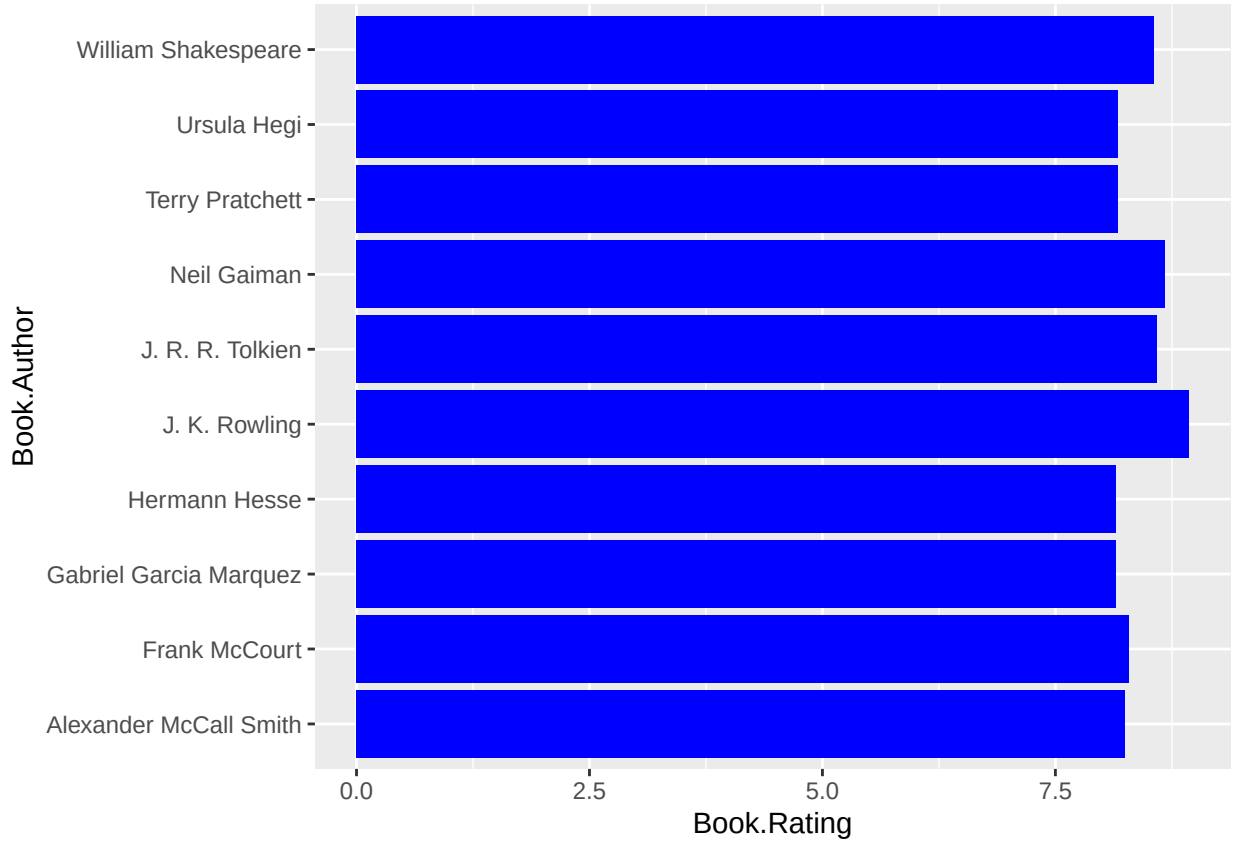
Top 10 rated authors

Get the authors which has been voted more than 100 times. Calculate rating means for each one and rate them in descending order.

```
author.high.count <-  dataset %>% group_by(Book.Author) %>% filter(n()>100)
author.high.count.mean <- setorder(setDT(author.high.count)[, .(Book.Rating = mean(Book.Rating)), by = B
author.high.count.mean
```

```
##                Book.Author Book.Rating
##  1:          J. K. Rowling    8.931459
##  2:            Neil Gaiman    8.679688
##  3:       J. R. R. Tolkien    8.594203
##  4:    William Shakespeare    8.556391
##  5:          Frank McCourt    8.292035
##  6: Alexander McCall Smith    8.243243
##  7:            Ursula Hegi    8.173913
##  8:        Terry Pratchett    8.170455
##  9:          Hermann Hesse    8.154412
## 10: Gabriel Garcia Marquez    8.151786
```

```
author.high.count.mean %>%
ggplot(aes(Book.Rating, Book.Author)) +
  geom_col(fill='blue')
```



J. K. Rowling has the highest rate with 8.93. Neil Gaiman and J. R. R. Tolkien are the second and the third ones.

Z.a) Imbalanced data set

We will look at how much data we have by class.

```
table(dataset$Rating.Count.Above.Mean)
```

```
##
##    No   Yes
## 32748 36151
```

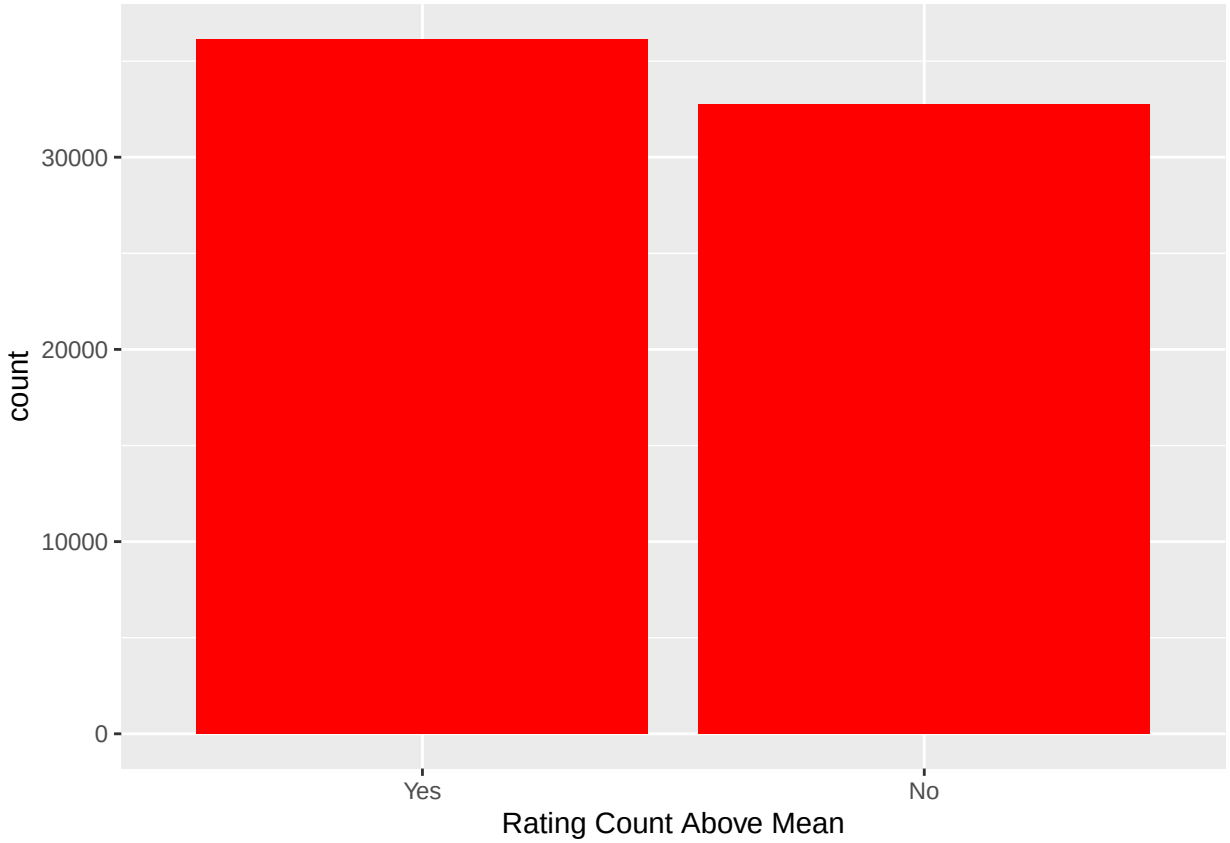
It seems that the Yes class has more data.

```
prop.table(table(dataset$Rating.Count.Above.Mean))
```

```
##
##        No       Yes
## 0.4753044 0.5246956
```

Visualization of how many books has greater rating than the mean

```
ggplot(dataset, aes(x=reorder(Rating.Count.Above.Mean, Rating.Count.Above.Mean, function(x)-length(x))))
geom_bar(fill='red') +  labs(x='Rating Count Above Mean')
```



We will store the old dataset as imbalanced dataset.

```
imbalanced.dataset <- dataset
missing.dataset <- dataset
```

Now we can apply oversampling to make them equal size.

```
n_legit <- 36151
new_frac_legit <- 0.68
new_n_total <- n_legit/new_frac_legit
```

```
oversampling_result <- ovun.sample(Rating.Count.Above.Mean ~ ., data = dataset, method = "over",
                                   N = new_n_total, seed = 2018)
dataset <- oversampling_result$data
row.names(dataset) <- NULL
table(dataset$Rating.Count.Above.Mean)
```

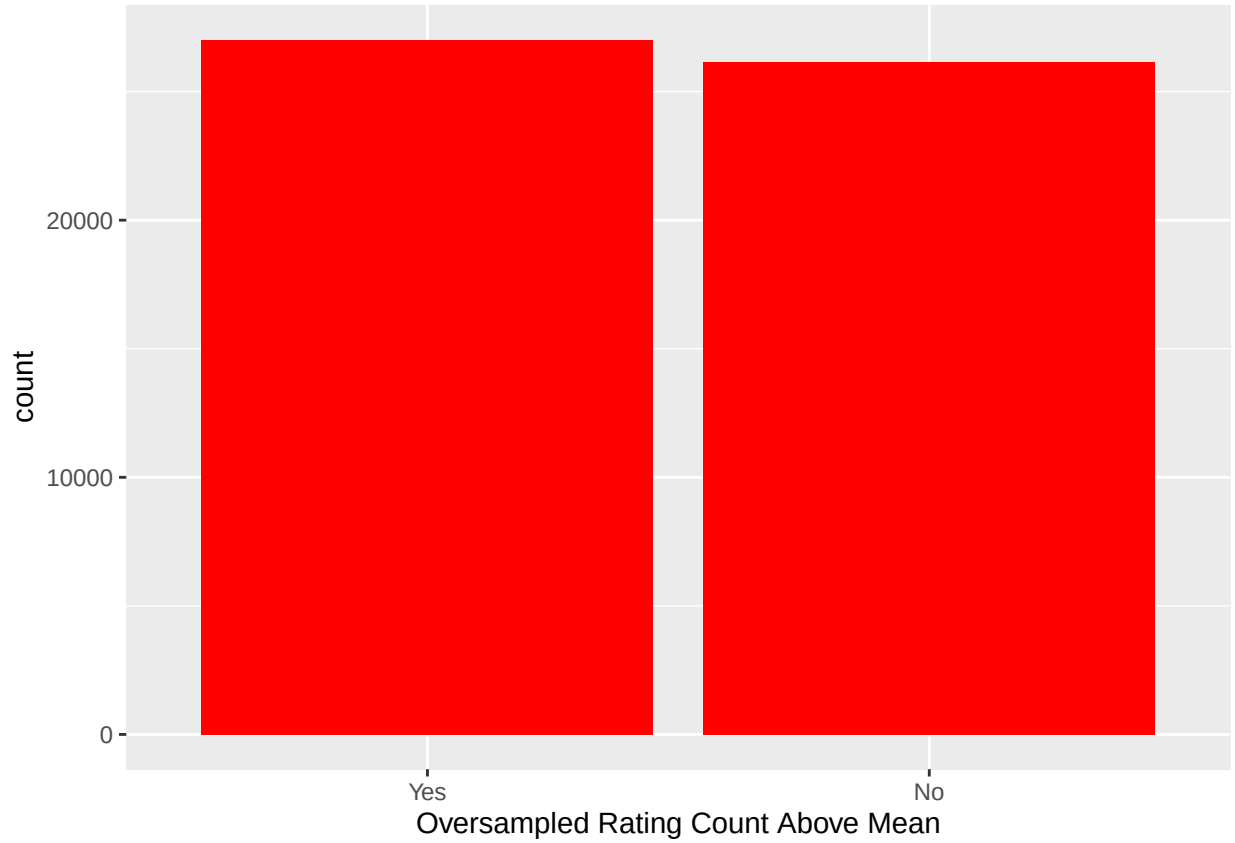
```
##
##    No   Yes
## 26142 27021
```

Now our classes have almost the same number of data.

```
prop.table(table(dataset$Rating.Count.Above.Mean))
```

```
##
##       No      Yes
## 0.491733 0.508267
```

```
ggplot(dataset, aes(x=reorder(Rating.Count.Above.Mean, Rating.Count.Above.Mean, function(x)-length(x))))
geom_bar(fill='red') +  labs(x='Oversampled Rating Count Above Mean')
```



Y.a) Missing data imputation

Check for NA values

```
sum(is.na(dataset))
```

```
## [1] 0
```

```
sum(is.na(imbalanced.dataset))
```

```
## [1] 19626
```

```
names(which(colSums(is.na(dataset)) > 0))
```

```
## character(0)
```

```
names(which(colSums(is.na(imbalanced.dataset)) > 0))
```

```
## [1] "Age"                 "Year.Of.Publication"
```

We have missing values and which are in the Age and Year of Publication columns. We will use mean imputation (or mean substitution) that replaces missing values of a certain variable by the mean of non-missing cases of that variable.

```
dataset$Age <- impute(dataset$Age, mean)
dataset$Year.Of.Publication <- impute(dataset$Year.Of.Publication, mean)

imbalanced.dataset$Age <- impute(imbalanced.dataset$Age, mean)
imbalanced.dataset$Year.Of.Publication <- impute(imbalanced.dataset$Year.Of.Publication, mean)
```

Now, let's check again for NA values

```
sum(is.na(dataset))
```

```
## [1] 0
```

```
sum(is.na(imbalanced.dataset))
```

```
## [1] 0
```

We have fixed missing data.

Subset data

The dataset contains 53,163 values after data preprocessing. Because we had some performance issues with our machines while fitting models, we had to take a subset of the dataset with 500 rows.

```
set.seed(12345)
dataset <- dataset[sample(1:nrow(dataset),500),]
imbalanced.dataset <- imbalanced.dataset[sample(1:nrow(dataset),500),]
missing.dataset <- missing.dataset[sample(1:nrow(dataset),500),]
row.names(dataset) <- NULL
row.names(imbalanced.dataset) <- NULL
row.names(missing.dataset) <- NULL
nrow(dataset)
```

```
## [1] 500
```

4) Multicollinearity

We replaced Rating.Count.Above.Mean values Yes with 2 and No with 1.

```
dataset$Rating.Count.Above.Mean <- ifelse(dataset$Rating.Count.Above.Mean == "No", 1, 2)
```

We selected certain columns of the data to calculate the correlation.

```
data <-dataset[, c('Book.Rating',"Age","Year.Of.Publication","Rating.Count.Above.Mean")]
```

```
sapply(data, class)
```

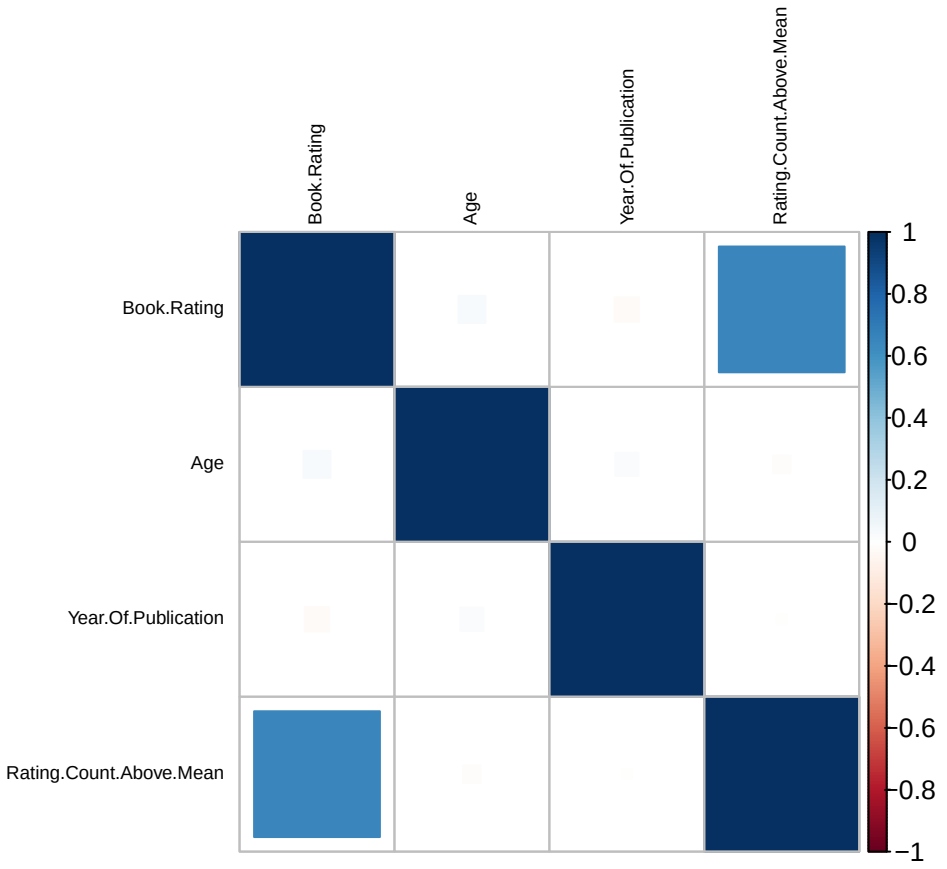
```
##             Book.Rating                     Age     Year.Of.Publication
##               "integer"               "numeric"                 "integer"
## Rating.Count.Above.Mean
##               "numeric"
```

We deleted NA values.

```
data <- data[!is.na(data$Age),]
data <- data[!is.na(data$Year.Of.Publication),]
data <- data[!is.na(data$Book.Rating),]
data <- data[!is.na(data$Rating.Count.Above.Mean),]
```

We drew a correlation map to see the correlation between our columns.

```
correlations <- cor(data)
corrplot::corrplot(correlations,method = "square",tl.cex = 0.6, tl.col = "black")
```



Multicollinearity occurs when features are highly correlated with one or more of the other features in the dataset.

As you see above some of the features in the dataset are highly correlated with each other. So, there exists multicollinearity. We can effectively eliminate multicollinearity between features using PCA.

We will apply PCA (5) after Logistic Regression and we will also apply PCA to Logistic Regression.

6) Logistic Regression

Logistic Regression is a classification model which is used to understand the relationship between the dependent variable and one or more independent variables by estimating probabilities using a logistic regression equation.

- The dependent variable should be binary like yes or no.
- It can help you predict the likelihood of an event happening or a choice being made.

Linear Regression outputs continuous value, and it has a straight line to map the input variables to the dependent variables. The output can be any of an infinite number of possibilities. On the other hand, Logistic Regression uses a logistic function to map the input variables to categorical dependent variables. In contrast to Linear Regression, Logistic Regression outputs a probability between 0 and 1.

6.a) Functions and arguments

We have a factor variable which is Rating.Count.Above.Mean (our dependent variable) but R assumes it is numeric. We modified it to factor.

```
data$Rating.Count.Above.Mean <- as.factor(data$Rating.Count.Above.Mean)
```

We split our data into train and test data.

```
sample<- createDataPartition(y= data$Rating.Count.Above.Mean,p=0.8,list = FALSE)

train_data <- data[sample,]
test_data <- data[-sample,]
```

We created our model.

```
logistic_model <- glm(Rating.Count.Above.Mean~.,data = train_data,family = "binomial")
```

We made predictions.

```
prob <- predict(logistic_model,newdata=test_data,type="response")
pred <- ifelse(prob > 0.5, 2, 1)
```

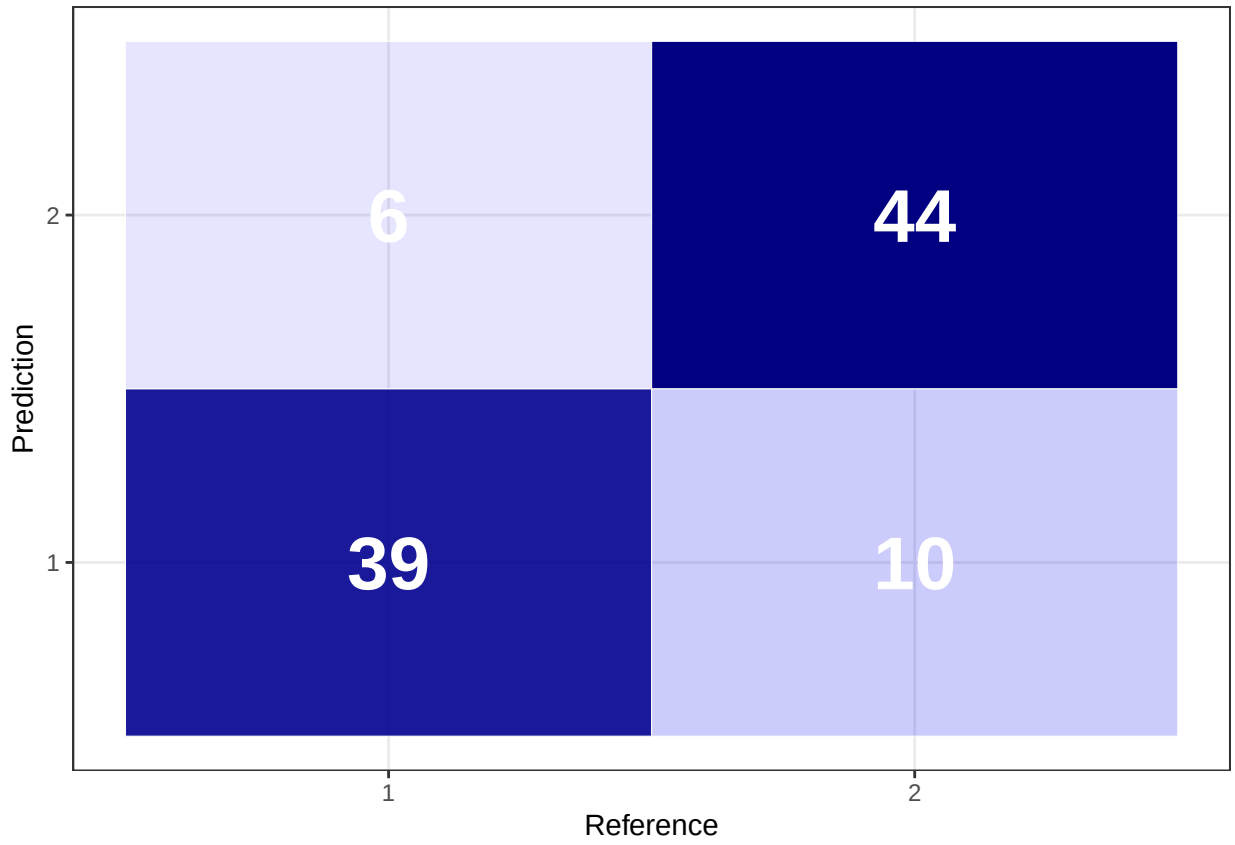
We generated a confusion matrix.

```r
conf.matrix <- confusionMatrix(test_data$Rating.Count.Above.Mean,as.factor(pred))
conf.matrix
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  1  2
##          1 39 10
##          2  6 44
##
##                Accuracy : 0.8384
##                  95% CI : (0.7509, 0.9047)
##     No Information Rate : 0.5455
##     P-Value [Acc > NIR] : 6.848e-10
##
##                   Kappa : 0.6765
##
##  Mcnemar's Test P-Value : 0.4533
##
##             Sensitivity : 0.8667
##             Specificity : 0.8148
##          Pos Pred Value : 0.7959
##          Neg Pred Value : 0.8800
##              Prevalence : 0.4545
##          Detection Rate : 0.3939
##    Detection Prevalence : 0.4949
##       Balanced Accuracy : 0.8407
##
##        'Positive' Class : 1
##
```

6.b) Visualization

```r
# Heatmap visualization of confusion matrix
table <- data.frame(conf.matrix$table)
plotTable <- table %>%
  group_by(Prediction) %>%
  mutate(prop = Freq/sum(Freq))
ggplot(data =  plotTable, mapping = aes(x = Reference, y = Prediction, alpha = prop)) +
  geom_tile(aes(fill = Freq), colour = "white") +
  geom_text(aes(label = Freq), vjust = .5, fontface  = "bold", alpha = 1, color="white", size=10) +
  scale_fill_gradient(low = "blue", high = "navyblue") +
  theme_bw() + theme(legend.position = "none")
```



6.c)

As you see above model accuracy is 83%. The model classified 39 + 44 datapoint correctly. We can also increase this ratio by applying PCA(Principal Component Analysis) to our dataset.

5) PCA

5.a) Functions and arguments

We converted our columns to numeric to apply PCA.

```
train_data$Book.Rating <- as.numeric(train_data$Book.Rating)
train_data$Year.Of.Publication <- as.numeric(train_data$Year.Of.Publication)
train_data$Rating.Count.Above.Mean <- as.numeric(train_data$Rating.Count.Above.Mean)
test_data$Book.Rating <- as.numeric(test_data$Book.Rating)
test_data$Year.Of.Publication <- as.numeric(test_data$Year.Of.Publication)
test_data$Rating.Count.Above.Mean <- as.numeric(test_data$Rating.Count.Above.Mean)
```

```
sapply(train_data,class)
```

```
##             Book.Rating                     Age     Year.Of.Publication
##               "numeric"               "numeric"                "numeric"
```

```
## Rating.Count.Above.Mean
##               "numeric"
```

```
levels(train_data$Rating.Count.Above.Mean)
```

```
## NULL
```

Apply PCA

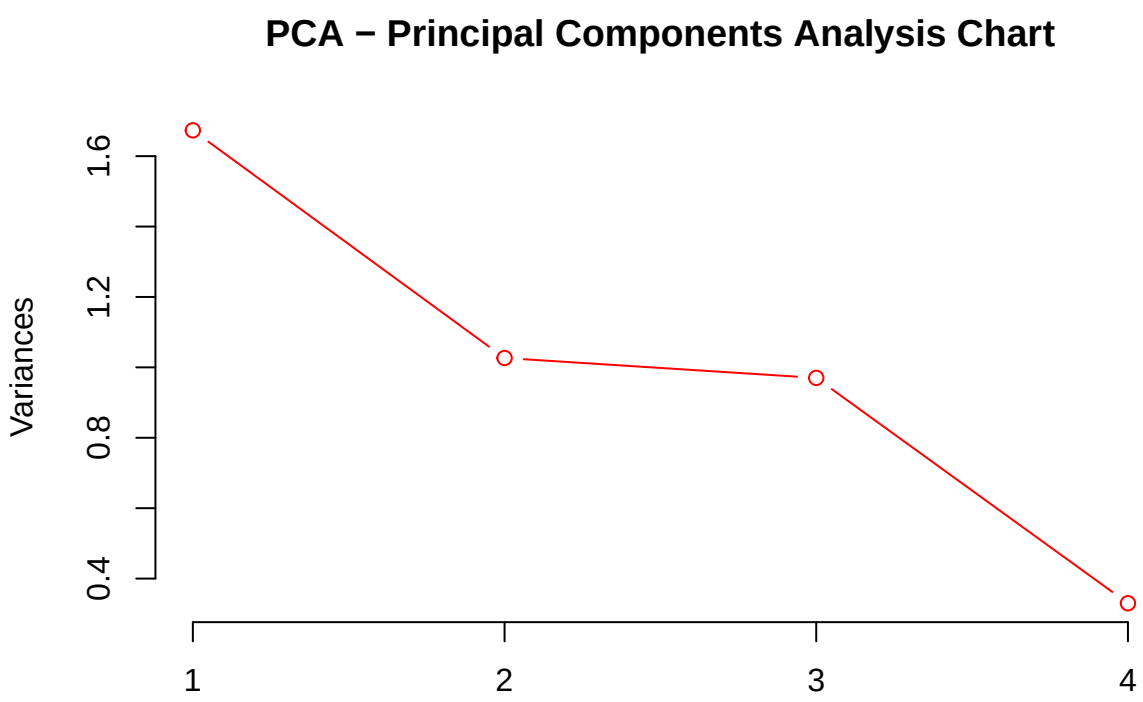
We applied Pca in our dataset.

```
pca <- prcomp(train_data, center = TRUE, scale = TRUE)
pca_test<-prcomp(test_data,center = TRUE, scale=TRUE)
pca
```

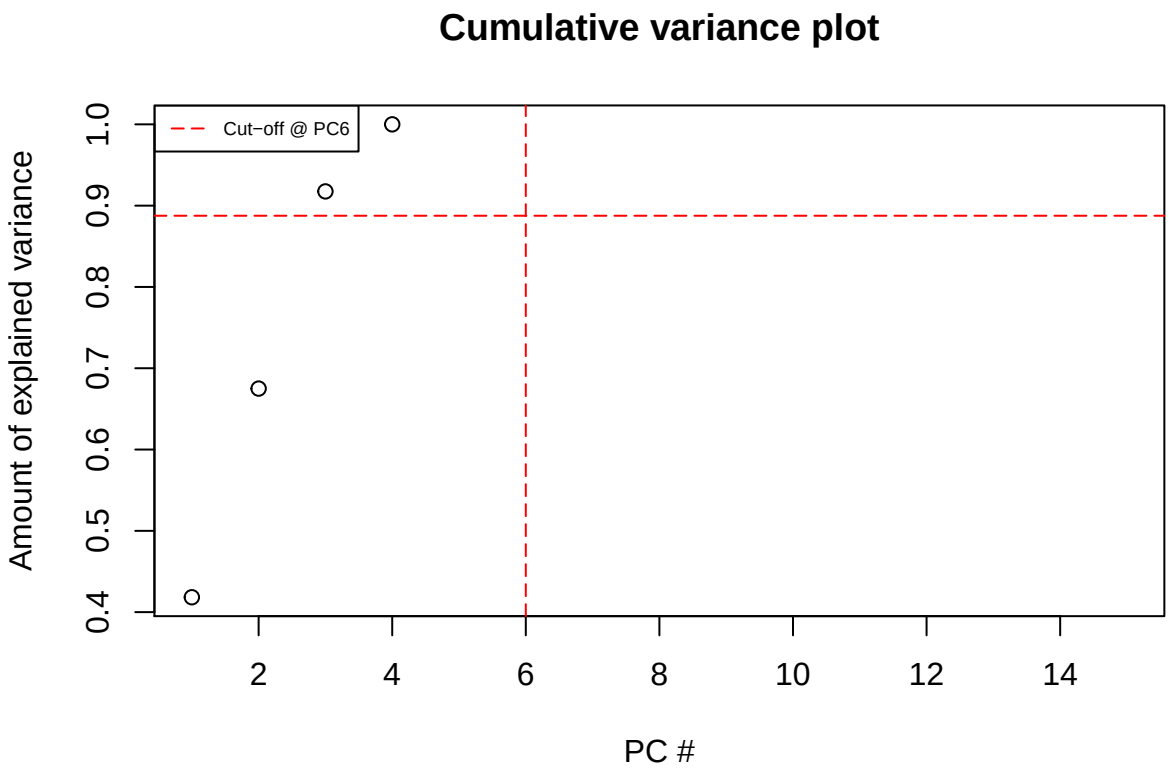
```
## Standard deviations (1, .., p=4):
## [1] 1.2935581 1.0132289 0.9849450 0.5744195
##
## Rotation (n x k) = (4 x 4):
##                                  PC1         PC2         PC3         PC4
## Book.Rating             -0.701836995 -0.06626641 -0.09090051  0.70339938
## Age                     -0.001287941  0.73046730 -0.68263312 -0.02068551
## Year.Of.Publication     -0.094992430  0.67919490  0.72506120  0.06290457
## Rating.Count.Above.Mean -0.705974229 -0.02684369 -0.00594752 -0.70770363
```

5.b) Visualization

```
plot(pca, type='l', main="PCA - Principal Components Analysis Chart", col="red")
```




```
cumpro <- cumsum(pca$sdev^2 / sum(pca$sdev^2))
plot(cumpro[0:15], xlab = "PC #", ylab = "Amount of explained variance", main = "Cumulative variance pl
abline(v = 6, col="red", lty=5)
abline(h = 0.88759, col="red", lty=5)
legend("topleft", legend=c("Cut-off @ PC6"),
       col=c("red"), lty=5, cex=0.6)
```



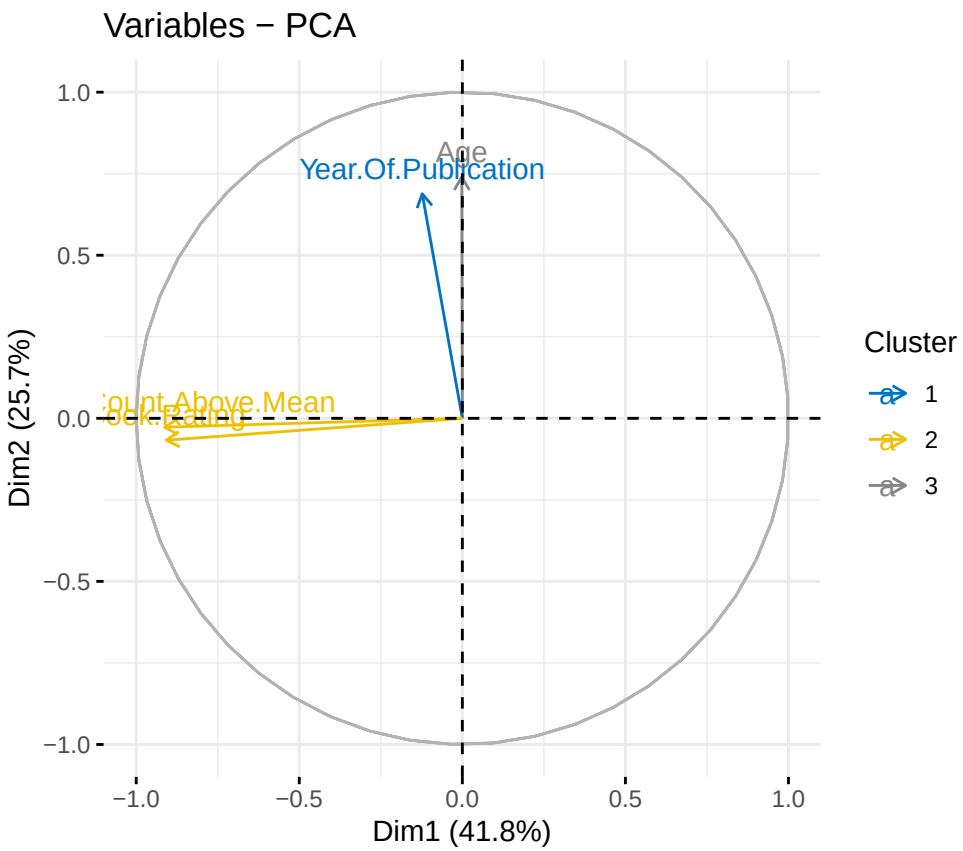
Get eigenvalues

```
explained.variance <- pca$sdev^2
explained.variance
```

```
## [1] 1.6732926 1.0266329 0.9701167 0.3299578
```

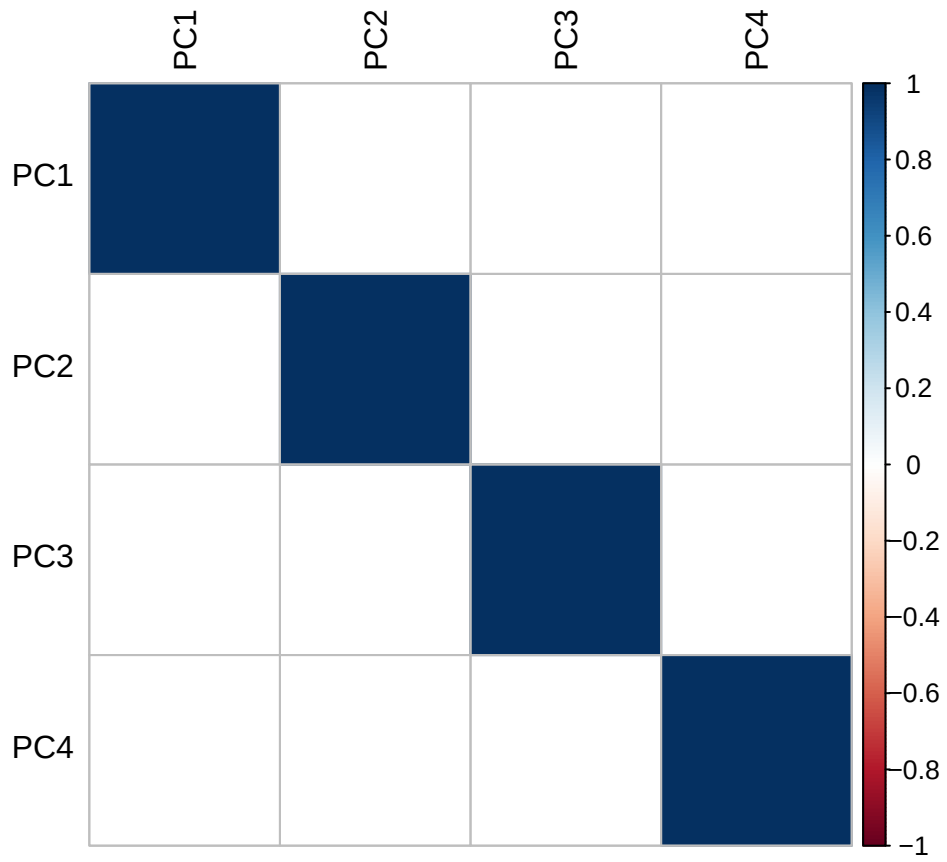
Plot PCA

```
pca.var <- get_pca_var(pca)

kmean <- kmeans(pca.var$coord, centers = 3, nstart=25)
group <- as.factor(kmean$cluster)

fviz_pca_var(pca, col.var=group, palette='jco', legend.title='Cluster')
```



After applying it to PCA, we drew a correlation map again.

```
correlations <- cor(pca$x[,c(0:4)])

corrplot::corrplot(correlations,method = "square", tl.col = "black")
```



5.c) Final comments

We cannot see any correlation between components. This is because of PCA has transformed the set of correlated variables in the original dataset into a set of uncorrelated variables.

6.X) Building a logistic regression model on the transformed data

After applying PCA, we applied logistic regression to our data again to compare the results and see the effect of PCA.

```
train_data$Rating.Count.Above.Mean <- as.factor(train_data$Rating.Count.Above.Mean)
```

```
set.seed(42)

data_pca <- data.frame(Rating.Count.Above.Mean=train_data[,"Rating.Count.Above.Mean"],pca$x[,0:4])
head(data_pca)
```

```
##   Rating.Count.Above.Mean        PC1        PC2        PC3         PC4
## 1                       2 -1.5530413 -0.3875536  0.1883351  0.17552978
## 2                       1  2.4112556 -0.1977540  1.5990839 -0.97518884
## 4                       2 -0.7653505 -0.1324779 -0.4604700 -0.61445885
## 5                       1  0.9273708 -0.2580050  1.1382906  0.51122765
```

```
## 6                       2 -1.4623745  0.5331674 -2.0100576  0.06863622
## 7                       2 -1.5672241  0.1787382 -0.1497357  0.17103918
```

```
set.seed(42)
model_pca <- glm(Rating.Count.Above.Mean ~ .,data= data_pca,family = binomial)
```

```
## Warning: glm.fit: algorithm did not converge
```

```
test_data_pca <- predict(pca,newdata = test_data)
```

```
prob <- predict(model_pca , newdata = data.frame(test_data_pca[,0:4]),type = "response")

pred <- factor(ifelse(prob>0.5,2,1))

levels(as.factor(pred))
```

```
## [1] "1" "2"
```

```
levels(test_data$Rating.Count.Above.Mean)
```

```
## NULL
```

```
confusionMatrix(as.factor(test_data$Rating.Count.Above.Mean),as.factor(pred))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  1  2
##          1 49  0
##          2  0 50
##
##                Accuracy : 1
##                  95% CI : (0.9634, 1)
##     No Information Rate : 0.5051
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 1
##
##  Mcnemar's Test P-Value : NA
##
##             Sensitivity : 1.0000
##             Specificity : 1.0000
##          Pos Pred Value : 1.0000
##          Neg Pred Value : 1.0000
##              Prevalence : 0.4949
##          Detection Rate : 0.4949
##    Detection Prevalence : 0.4949
##       Balanced Accuracy : 1.0000
##
##        'Positive' Class : 1
##
```

As you see above our accuracy become 100%.The test accuracy has increased by 17%. Both false positives and false negatives have also been reduced. The reason behind the performance increase in this model is is PCA has effectively eliminated the multicollinearity.

7) Clustering

7.a) Why K-means and Hierarchical Clustering

We will cluster users based on their age and book ratings.

We decided to use K-means Clustering and Hierarchical Clustering algorithms. The main reason is that we learned them in the lesson and they are the most popular ones. Besides those reasons there are some other reasons:

Advantages of K-Means:

- Guarantees convergence
- Can warm-start the positions of centroids
- Easily adapts to new examples
- Generalizes to clusters of different shapes and sizes, such as elliptical clusters

Advantages of Hierarchical Clustering:

- It is a powerful technique that allows building tree structures from data similarities
- It lets us see how different sub-clusters relate to each other, and how far apart data points are

7.1) K-Means Clustering Algorithm Application

K-means clustering aims to partition data into k clusters in a way that data points in the same cluster are similar and data points in the different clusters are farther apart. It's an unsupervised machine learning algorithm. It computes the centroids and iterates until it finds optimal centroid. It assumes that the number of clusters are already known.

The k-means clustering works as follows:

- Choose the k number of clusters
- Select k random points, the centroids (they don't have to be part of the dataset)
- Assign each point to the closest centroid
- Compute and replace the new centroid of each cluster
- Reassign each data point to the new closest centroid. If any reassignment happens, go back to previous step.

7.1.a) Functions and arguments

We will use age and book rating to cluster users.

```r
X <- dataset[sample,] %>% select("User.ID", "Age", "Book.Rating")
head(X, n=5)
```

```
##   User.ID Age Book.Rating
## 1   95226  30          10
## 2  220248  20           3
## 4  152562  38           8
## 5   17183  23           7
## 6   23725  57          10
```

Determining Optimal Number of Clusters The number of clusters that we choose for a given dataset cannot be random. Each cluster is formed by calculating and comparing the distances of data points within a cluster to its centroid. An ideal way to figure out the right number of clusters would be to calculate the Within-Cluster-Sum-of-Squares (WCSS).

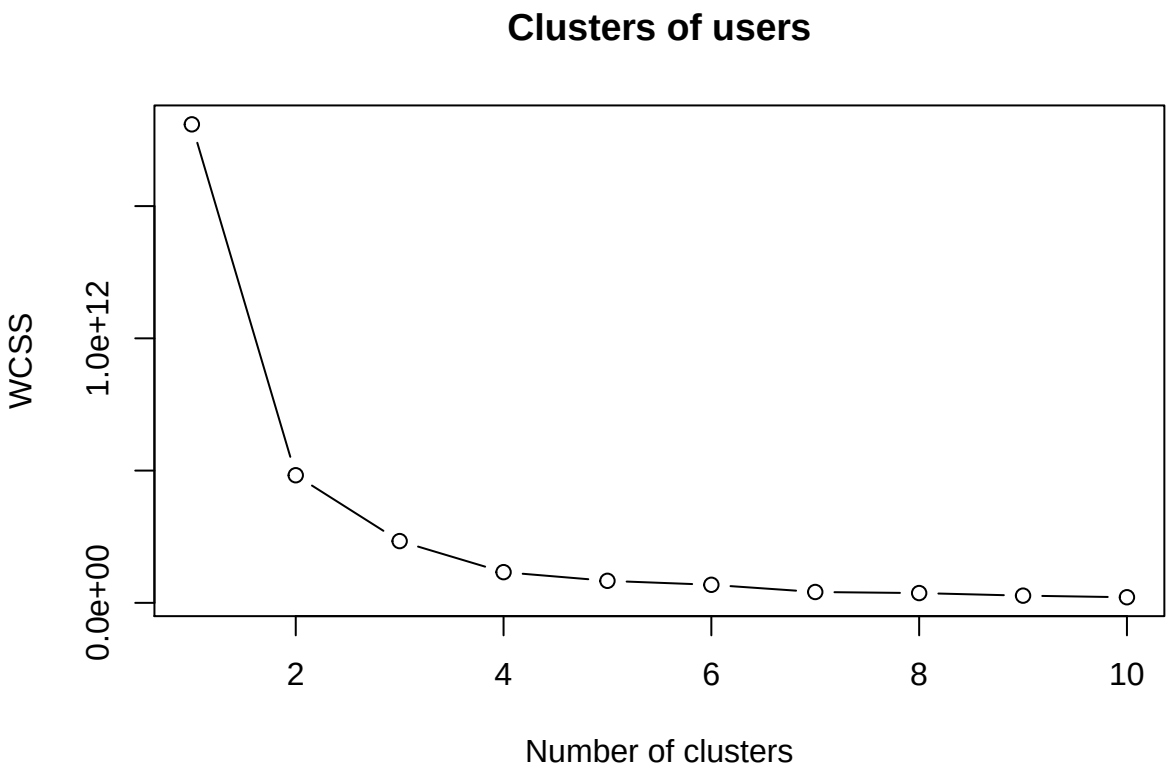
WCSS is the sum of squares of the distances of each data point in all clusters to their respective centroids.

We can use some techniques to determine optimal number of clusters. **Elbow method** is one of them. The method consists of plotting the explained variation as a function of the number of clusters, and picking the elbow of the curve as the number of clusters to use.

```r
X <- X[!is.na(X$Age), ]
```

Elbow method We are going to use the Elbow Method to decide the optimal number of clusters.

```r
set.seed(6)
wcss <- vector()
for (i in 1:10) wcss[i] <-  sum(kmeans(X, i)$withinss)
plot(1:10, wcss, type = "b", main = paste("Clusters of users"), xlab = "Number of clusters", ylab = "WC
```



As seen on the plot, the optimal number of clusters seems as 3.

Apply K-Means We will split our data into 3 clusters. The nstart parameter attempts multiple initial configurations and reports on the best one.

```
set.seed(29)
kmeans.model <- kmeans(X, 3, iter.max = 300, nstart = 10)
kmeans.model
```

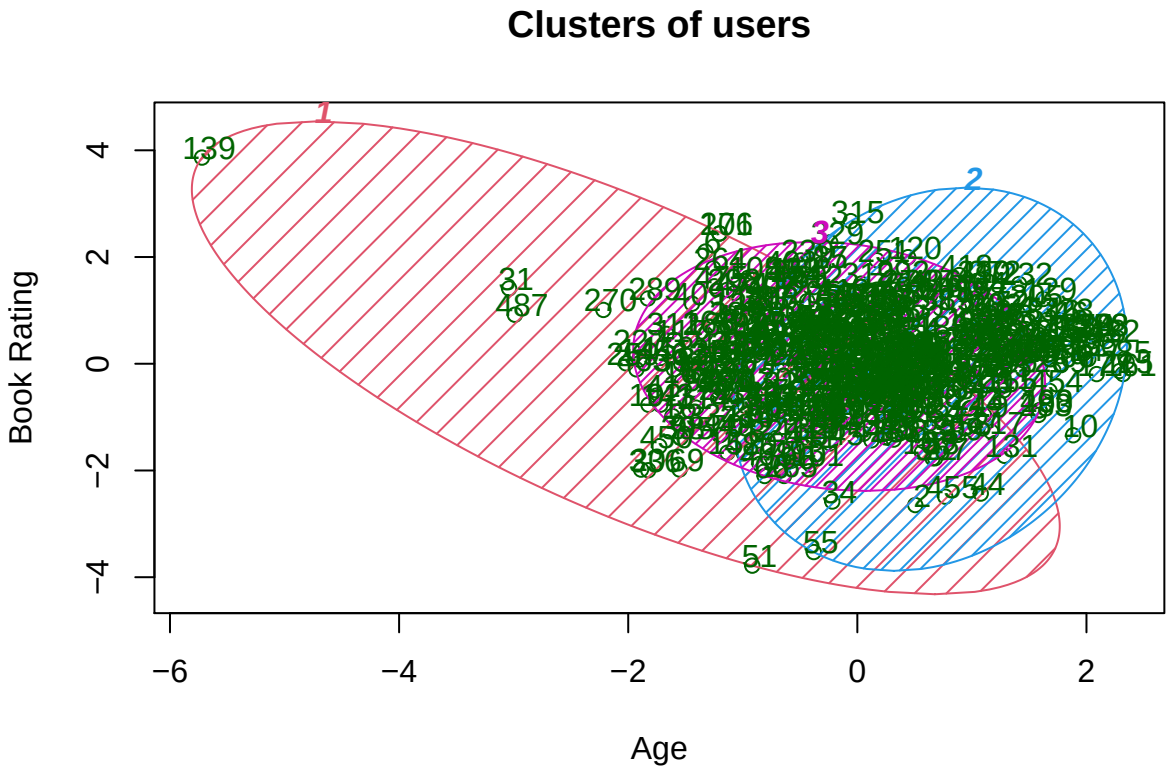
```
## K-means clustering with 3 clusters of sizes 120, 125, 156
##
## Cluster means:
##     User.ID      Age Book.Rating
## 1  38448.51 36.20000    7.658333
## 2 198820.33 33.92000    7.792000
## 3 115744.62 35.27564    7.679487
##
## Clustering vector:
##   1   2   4   5   6   7   8  10  11  13  14  15  16  17  19  20  21  22  23  24
##   3   2   3   1   1   3   1   2   2   1   3   3   3   1   3   1   2   2   3   2
##  25  27  28  30  31  32  33  34  35  36  38  40  41  42  44  45  46  47  48  49
##   1   1   3   1   1   2   3   2   3   3   1   3   1   3   2   3   2   3   1   2
##  50  51  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67  69  70  72
##   2   1   2   2   2   2   3   1   3   3   2   1   1   2   2   3   3   1   1   2
##  73  74  76  77  78  79  80  82  83  84  85  86  87  90  91  92  97  98  99 101
```

```
##   3   1   1   3   2   1   2   1   2   3   2   3   3   3   2   2   2   1   2   3
## 102 103 104 105 106 107 108 109 110 112 113 114 116 117 119 120 121 122 123 125
##   1   2   3   2   1   1   1   2   3   3   2   1   3   1   2   2   1   2   2   1
## 126 127 128 129 130 131 132 134 135 137 138 139 140 142 143 144 145 146 147 148
##   1   3   3   2   3   2   3   3   1   3   1   1   3   2   1   3   3   2   2   2
## 149 150 151 152 153 154 155 156 157 158 159 160 161 162 163 165 167 168 169 170
##   3   3   3   1   2   3   2   1   2   1   3   2   2   1   1   3   2   1   1   1
## 172 173 174 175 176 177 178 179 180 181 182 183 184 185 186 188 189 190 191 192
##   3   1   1   2   3   2   3   1   1   3   1   2   3   1   1   2   2   1   1   1
## 193 194 195 196 197 198 199 200 201 202 203 204 205 206 207 208 210 211 213 214
##   3   3   2   2   3   3   2   1   3   2   2   2   2   3   3   1   2   2   3   2
## 215 217 218 219 220 221 222 224 225 226 227 228 229 230 232 234 235 236 237 238
##   2   2   3   3   3   3   1   3   3   3   3   2   3   3   2   1   3   3   2   2
## 239 240 241 243 246 249 250 251 252 253 254 256 258 259 260 262 263 264 265 266
##   3   3   1   3   2   3   3   3   3   2   3   3   3   1   3   3   3   3   3   3
## 267 268 269 270 271 272 273 274 275 276 277 279 280 281 282 283 284 286 287 289
##   1   3   2   1   1   2   1   3   1   2   2   1   3   1   1   2   1   3   1   1
## 290 292 293 294 295 296 297 298 299 300 302 303 304 305 307 309 310 311 312 313
##   1   3   3   3   3   2   1   1   2   2   3   1   2   3   1   3   2   3   1   3
## 314 315 317 318 319 320 321 322 323 325 326 328 329 330 331 333 334 335 336 338
##   3   2   3   2   3   2   1   1   1   2   3   2   3   2   2   3   1   3   1   1
## 343 344 345 346 348 349 350 352 353 354 355 356 357 358 360 362 363 364 366 367
##   3   1   3   2   3   2   2   1   2   2   2   3   2   1   1   1   1   1   2   2
## 368 370 371 372 374 375 376 377 378 380 383 384 385 386 387 388 389 390 391 392
##   3   1   3   1   1   2   3   1   1   1   2   3   1   2   1   3   3   3   1   3
## 393 394 395 396 397 398 399 401 403 404 405 407 408 409 410 411 412 413 414 415
##   3   3   1   2   3   3   3   3   1   2   3   2   3   3   3   3   2   1   3   2
## 418 420 421 422 424 425 426 427 428 429 430 431 432 433 434 435 436 438 439 440
##   2   1   3   3   1   2   3   2   3   2   2   3   2   1   2   2   3   2   2   3
## 441 442 443 445 447 448 450 451 452 454 455 457 458 459 460 462 463 465 466 467
##   1   3   2   2   2   1   1   1   3   2   2   1   2   3   3   1   1   3   1   3
## 468 470 471 473 474 475 476 478 479 480 481 483 487 488 489 491 492 493 494 495
##   3   3   3   3   1   3   3   1   3   1   1   3   1   1   2   2   2   3   3   2
## 497
##   3
##
## Within cluster sum of squares by cluster:
## [1] 60473689440 81909040655 90069345292
##  (between_SS / total_SS =  87.2 %)
##
## Available components:
##
## [1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss"
## [6] "betweenss"    "size"         "iter"         "ifault"
```

7.1.b) Visualization

With clusplot function we can draw a 2 dimensional clustering plot with our clusters.

```
clusplot(X, clus = kmeans.model$cluster, lines = 0, shade = TRUE, color = TRUE, labels = 2, plotchar = F
```



7.1.c)

We can see the clusters above. Age and book ratings explain 68.34% of the point variability. There are some far points like 139 and 1 which have very low age and very high book ratings. We assumed that they can be outliers. Users might not enter the correct age and might give very few votes with a high rating value.

7.2) Hierarchical Clustering Algorithm Application

Hierarchical clustering is **an algorithm that groups similar objects into groups called clusters**. It is an alternative approach to k-means clustering for identifying groups. The endpoint is a set of clusters, where each cluster is distinct from the other cluster, and the objects within each cluster are broadly similar to each other.

- It is an unsupervised machine learning algorithm.
- The hierarchical clustering does not require us to pre-specify the number of clusters to be generated as is required by the k-means approach.
- It has a tree-based representation called **dendrogram** which is a diagram that shows the hierarchical relationship between objects.

Hierarchical clustering can be divided into two main types: **agglomerative** and **divisive**.

A) Agglomerative Clustering

It is a bottom-up approach. In the beginning, each object is initially considered as a single-element cluster. At each step of the algorithm, the two clusters that are the most similar are combined into a new bigger cluster. This procedure is iterated until all points are member of just one single big cluster.

This process has a **O(N^3) time complexity** and a **O(N^2) memory complexity** that makes it *not tractable for large datasets.*

How it works:

- Make each data point a single-point cluster
- Take the two closest *data points* and make them one cluster
- Take the two closest *clusters* and make them one cluster
- Repeat the previous step until there is only one cluster

B) Divisive Clustering

It is a top-down approach. It begins with the root, in which all objects are included in a single cluster. At each step of iteration, the most heterogeneous cluster is divided into two. The process is iterated until all objects are in their own cluster.

This process requires at each iteration to search for the best split, implying a **O(2N) time complexity** that has to be tackled with some heuristics. *Divisive hierarchical clustering is good at identifying large clusters.*

7.2.a) Functions and arguments

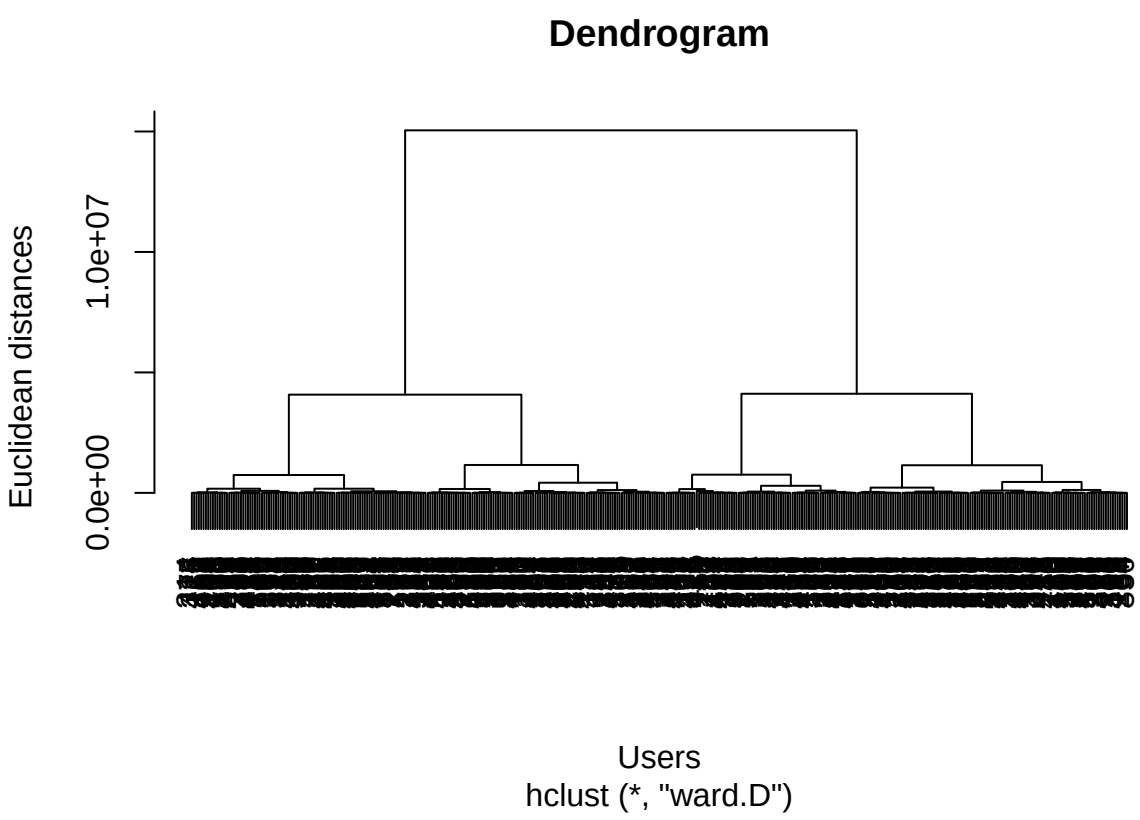
We will use the same data frame that we have been created in the previous model.

```
head(X, n=5)
```

```
##   User.ID Age Book.Rating
## 1   95226  30          10
## 2  220248  20           3
## 4  152562  38           8
## 5   17183  23           7
## 6   23725  57          10
```

Dendrogram A dendrogram is a tree-like chart that shows the sequences of merges or splits of clusters. We will use it to find the optimal number of clusters.

```
dendrogram <- hclust(dist(X, method = 'euclidean'), method = 'ward.D')
plot(dendrogram, main = 'Dendrogram', xlab = 'Users', ylab = 'Euclidean distances')
```



The larger gap cut generates 2 clusters so we can say optimal number of clusters is 2.

```
hc <- hclust(dist(X, method = 'euclidean'), method = 'ward.D')
```

Apply hierarchical clustering Cutree method cuts a dendrogram tree into several groups by specifying the desired number of clusters k(s), or cut height(s).

```
y_hc <- cutree(hc, 2)
y_hc
```

```
##   1   2   4   5   6   7   8  10  11  13  14  15  16  17  19  20  21  22  23  24
##   1   2   2   1   1   1   1   2   2   1   1   1   1   1   2   1   2   2   2   2
##  25  27  28  30  31  32  33  34  35  36  38  40  41  42  44  45  46  47  48  49
##   1   1   2   1   1   2   1   2   1   2   1   1   1   1   2   2   2   2   1   2
##  50  51  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67  69  70  72
##   2   1   2   2   2   2   1   1   2   1   2   1   1   2   2   1   2   1   1   2
##  73  74  76  77  78  79  80  82  83  84  85  86  87  90  91  92  97  98  99 101
##   2   1   1   1   2   1   2   1   2   2   2   1   1   2   2   2   2   1   2   1
## 102 103 104 105 106 107 108 109 110 112 113 114 116 117 119 120 121 122 123 125
##   1   2   2   2   1   1   1   2   2   1   2   1   1   1   2   2   1   2   2   1
## 126 127 128 129 130 131 132 134 135 137 138 139 140 142 143 144 145 146 147 148
##   1   2   1   2   2   2   2   1   1   2   1   1   2   2   1   1   1   2   2   2
## 149 150 151 152 153 154 155 156 157 158 159 160 161 162 163 165 167 168 169 170
```

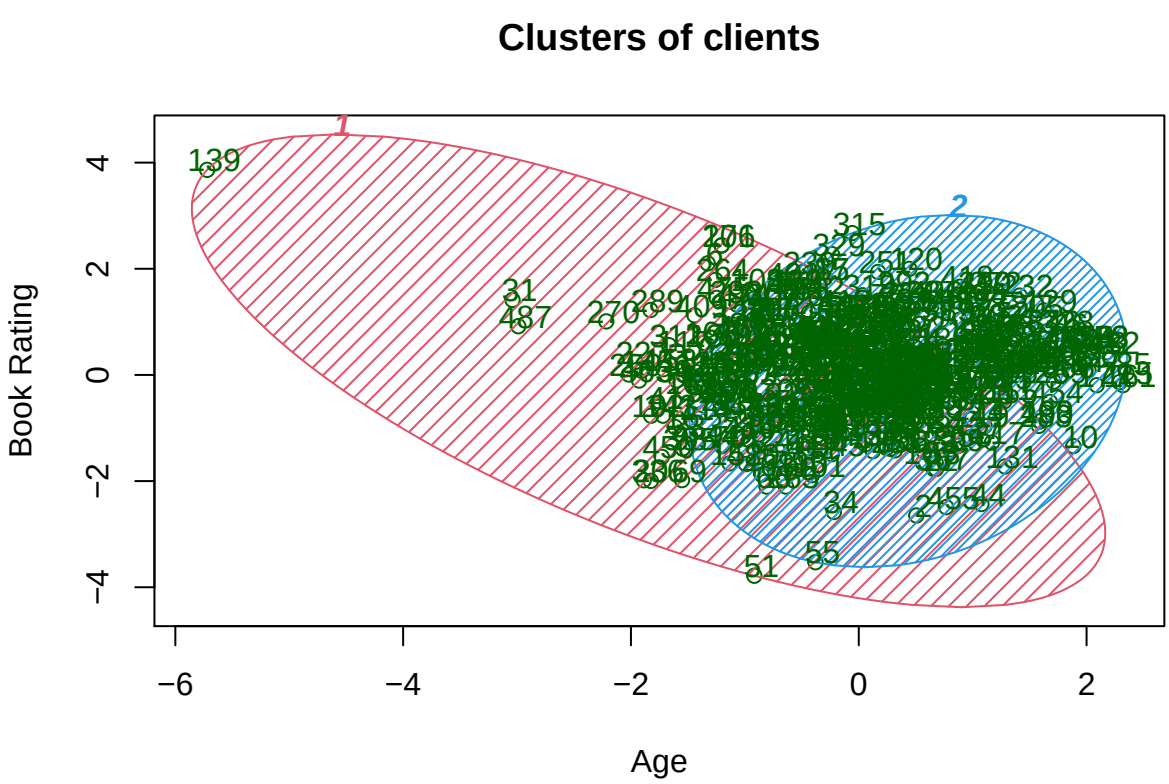
```
##   1   1   1   1   2   1   2   1   2   1   1   2   2   1   1   1   2   1   1   1
## 172 173 174 175 176 177 178 179 180 181 182 183 184 185 186 188 189 190 191 192
##   2   1   1   2   2   2   2   1   1   2   1   2   1   1   1   2   2   1   1   1
## 193 194 195 196 197 198 199 200 201 202 203 204 205 206 207 208 210 211 213 214
##   2   1   2   2   2   1   2   1   1   2   2   2   2   1   2   1   2   2   1   2
## 215 217 218 219 220 221 222 224 225 226 227 228 229 230 232 234 235 236 237 238
##   2   2   1   2   2   1   1   1   1   1   1   2   2   1   2   1   1   2   2   2
## 239 240 241 243 246 249 250 251 252 253 254 256 258 259 260 262 263 264 265 266
##   1   2   1   1   2   2   2   2   1   2   1   2   2   1   2   2   1   1   1   1
## 267 268 269 270 271 272 273 274 275 276 277 279 280 281 282 283 284 286 287 289
##   1   2   2   1   1   2   1   2   1   2   2   1   1   1   1   2   1   1   1   1
## 290 292 293 294 295 296 297 298 299 300 302 303 304 305 307 309 310 311 312 313
##   1   2   1   2   2   2   1   1   2   2   1   1   2   1   1   1   2   2   1   1
## 314 315 317 318 319 320 321 322 323 325 326 328 329 330 331 333 334 335 336 338
##   1   2   1   2   2   2   1   1   1   2   2   2   2   2   2   2   1   2   1   1
## 343 344 345 346 348 349 350 352 353 354 355 356 357 358 360 362 363 364 366 367
##   2   1   2   2   2   2   2   1   2   2   2   1   2   1   1   1   1   1   2   2
## 368 370 371 372 374 375 376 377 378 380 383 384 385 386 387 388 389 390 391 392
##   2   1   2   1   1   2   1   1   1   1   2   2   1   2   1   1   2   1   1   2
## 393 394 395 396 397 398 399 401 403 404 405 407 408 409 410 411 412 413 414 415
##   1   2   1   2   2   2   2   1   1   2   2   2   2   1   1   1   2   1   1   2
## 418 420 421 422 424 425 426 427 428 429 430 431 432 433 434 435 436 438 439 440
##   2   1   2   2   1   2   1   2   2   2   2   1   2   1   2   2   2   2   2   2
## 441 442 443 445 447 448 450 451 452 454 455 457 458 459 460 462 463 465 466 467
##   1   2   2   2   2   1   1   1   2   2   2   1   2   1   1   1   1   1   1   1
## 468 470 471 473 474 475 476 478 479 480 481 483 487 488 489 491 492 493 494 495
##   1   1   1   2   1   1   1   1   1   1   1   1   1   1   2   2   2   2   1   2
## 497
##   2
```

We can see the clusters above.

7.2.b) Visualize the clusters

With clusplot function we can draw a 2 dimensional clustering plot with our clusters.

```
clusplot(X, clus = y_hc, lines = 0, shade = TRUE, color = TRUE, labels = 2, plotchar = FALSE, span = TRU
         main = paste("Clusters of clients"), xlab = "Age", ylab = "Book Rating")
```



7.2.c)

With hierarchical clustering, we have almost the same clusters. But we divided the data into 2 clusters here. The blue cluster has the same data as the blue and pink clusters of k-means. There are also extreme points which are 139 and 1.

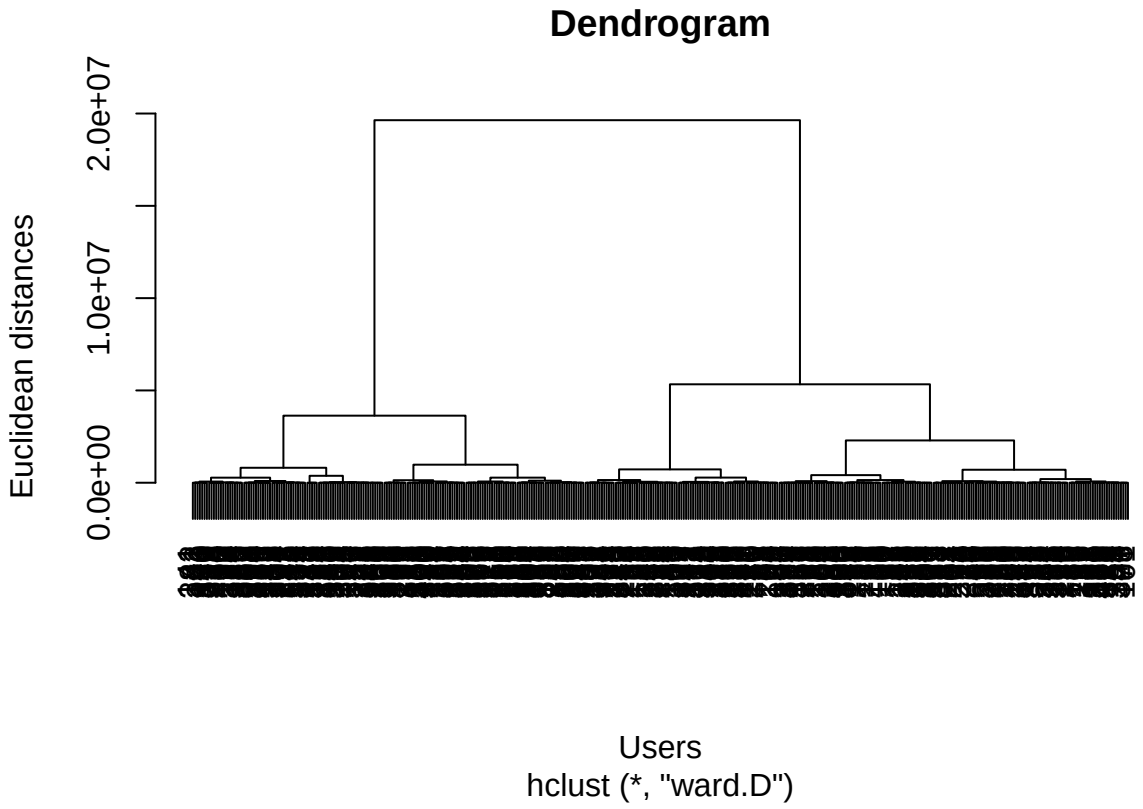
Y.b) Missing Data with Hierarchical Clustering

```
X <- missing.dataset[sample,] %>% select("User.ID", "Age", "Book.Rating")
head(X, n=5)
```

```
##    User.ID Age Book.Rating
## 1:   85560  NA           8
## 2:   98547  43           7
## 3:   70415  57          10
## 4:   92215  23           7
## 5:   79441  43           7
```

Dendrogram A dendrogram is a tree-like chart that shows the sequences of merges or splits of clusters. We will use it to find the optimal number of clusters.

```
dendrogram <- hclust(dist(X, method = 'euclidean'), method = 'ward.D')
plot(dendrogram, main = 'Dendrogram', xlab = 'Users', ylab = 'Euclidean distances')
```



The larger gap cut generates 2 clusters so we can say optimal number of clusters is 2.

```
hc <- hclust(dist(X, method = 'euclidean'), method = 'ward.D')
```

Apply hierarchical clustering Cutree method cuts a dendrogram tree into several groups by specifying the desired number of clusters k(s), or cut height(s).

```
y_hc <- cutree(hc, 2)
y_hc
```

```
##   [1] 1 1 1 1 1 2 1 2 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 2 1 1 1 1 2 1 2 1 2 2 1
##  [38] 2 1 2 1 1 2 1 1 2 1 2 1 2 1 1 1 1 2 2 1 1 1 1 1 1 1 1 2 2 1 1 1 1 1 1 2
##  [75] 1 1 1 1 2 2 2 1 2 2 1 1 1 1 2 1 1 2 1 2 1 1 1 1 2 1 1 1 1 2 1 1 1 1 1 2 1
## [112] 1 2 2 2 1 1 2 2 1 2 1 2 2 2 2 1 1 1 2 2 2 1 1 1 2 2 1 1 1 2 1 2 2 1 1 1 1
## [149] 1 2 1 1 2 1 2 1 1 2 2 1 2 1 1 1 1 2 1 2 1 2 1 2 2 1 1 2 1 1 2 2 1 1 2 1 1
## [186] 1 2 1 1 2 2 1 1 2 2 2 1 2 1 2 1 2 1 1 1 2 1 2 2 2 2 2 1 1 1 1 1 1 1 1 2 2
## [223] 1 1 1 1 1 2 1 1 2 2 1 2 1 1 1 1 2 1 2 1 2 2 1 1 1 1 2 1 1 2 2 1 1 2 2 1 2
## [260] 2 2 2 1 2 1 1 1 2 1 1 2 1 1 2 1 1 1 2 1 1 1 2 1 1 2 2 1 1 1 2 2 2 2 2 1 1
## [297] 1 1 2 1 1 2 2 2 2 2 2 2 2 1 1 2 2 1 1 1 2 2 1 1 1 2 2 2 1 2 1 1 2 1 2 2 2
```

```
## [334] 2 1 2 1 2 1 2 2 2 2 1 2 2 1 1 1 1 2 2 2 1 2 1 1 2 2 2 1 2 1 1 2 1 2 1 2 2
## [371] 1 1 1 2 2 2 1 2 2 1 2 2 2 1 2 2 1 2 1 2 2 1 2 1 1 1 1 1 2 2 1
```

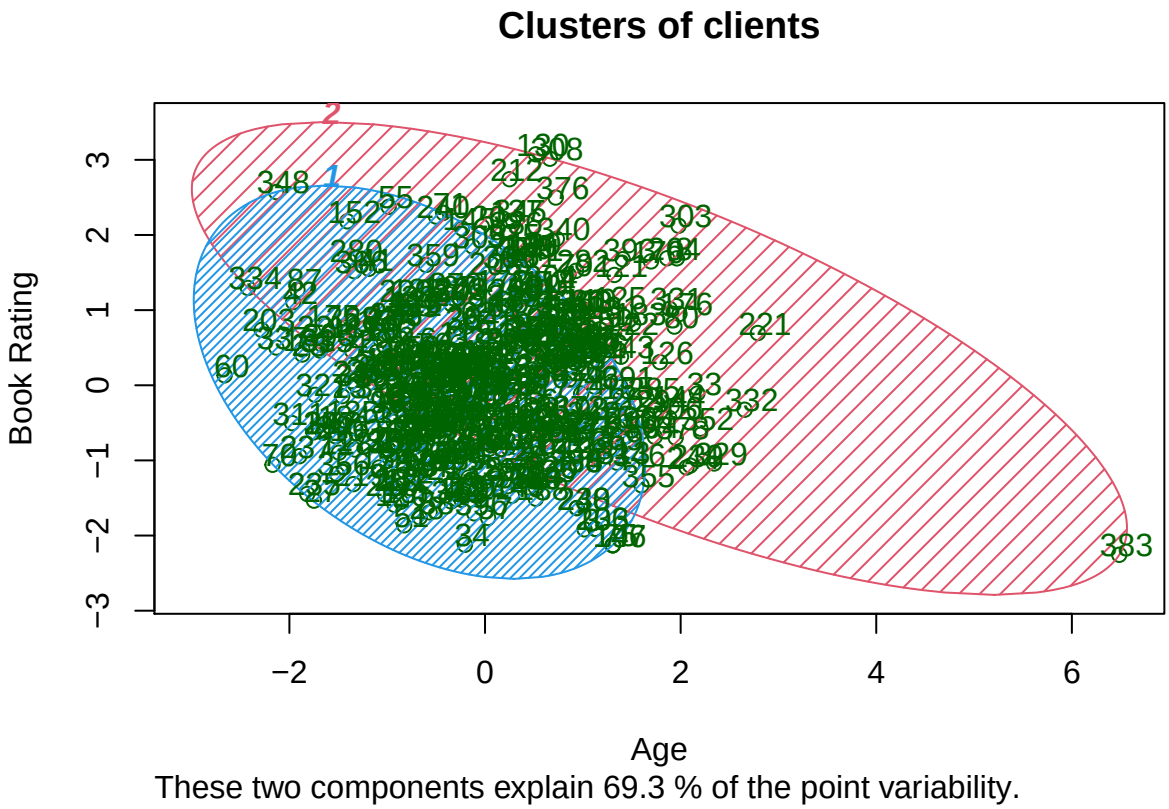
We can see the clusters above.

Visualize the clusters

With clusplot function we can draw a 2 dimensional clustering plot with our clusters.

```
clusplot(X, clus = y_hc, lines = 0, shade = TRUE, color = TRUE, labels = 2, plotchar = FALSE, span = TRU
         main = paste("Clusters of clients"), xlab = "Age", ylab = "Book Rating")
```

```
## Missing values were displaced by the median of the corresponding variable(s)
```



We tried to cluster data with missing values but algorithm displaced them by the median of the corresponding variable(s).

7.b)

Those two clustering algorithms have almost the same results. Except that we divided the data into 2 clusters in hierarchical, which was 3 on kmeans. They both explain the variability on the same percentage.

8) Classification

8.a) Why Decision Tree and KNN

We will classify books as their average rating will be grater than the mean of overall ratings or not. We will classify them based on Book.Rating and Year.Of.Publication.

The reason we use Decision Tree is that the decision trees' outputs are easy to read and interpret without requiring and less data cleaning is required. Also, we needed high accuracy in the classification we will apply, so we decided that KNN is one of the best choices.

8.1) Decision Tree Algorithm Application

Decision Tree is a supervised learning technique that can be used for both classification and regression problems, but mostly it is preferred for solving classification problems. - It is a tree-structured classifier - Internal nodes represent the features of a dataset - Branches represent the decision rules - Each leaf node represents the outcome

8.1.a) Functions and arguments

```
Z <- dataset[sample,] %>% select("Book.Rating", "Year.Of.Publication", "Rating.Count.Above.Mean")
head(Z, n=5)
```

```
##   Book.Rating Year.Of.Publication Rating.Count.Above.Mean
## 1          10                1997                       2
## 2           3                2002                       1
## 4           8                1994                       2
## 5           7                2001                       1
## 6          10                1990                       2
```

Turn the target feature to factor

```
Z$Rating.Count.Factor <- factor(Z$Rating.Count.Above.Mean, levels = c(1, 2))
```

```
Z <- Z[!is.na(Z$Year.Of.Publication), ]
Z <- select(Z,-c(Rating.Count.Above.Mean))
head(Z, n=5)
```

```
##   Book.Rating Year.Of.Publication Rating.Count.Factor
## 1          10                1997                   2
## 2           3                2002                   1
## 4           8                1994                   2
## 5           7                2001                   1
## 6          10                1990                   2
```

Split data into Train and Test set Rating.Count.Factor column is our dependent variable.

```
set.seed(123)
splitted <- sample.split(Z$Rating.Count.Factor, SplitRatio = 0.75)
train_Set <- subset(Z, splitted == TRUE)
test_Set <- subset(Z, splitted == FALSE)
```

Feature Scaling Feature scaling is a method used to normalize the range of independent variables or features of data.

We will scale all the features except our dependent variable, Rating.Count.Factor.

```
train_y = train_Set[,3]
test_y = test_Set[,3]

row.names(train_Set) <- NULL
row.names(test_Set) <- NULL

# Scaled test and train set
trainSet = data.frame(scale(train_Set[,-3]))
trainSet$Rating.Count.Factor = train_y

testSet = data.frame(scale(test_Set[,-3]))
testSet$Rating.Count.Factor = test_y
```

```
model.decision <- rpart(formula = Rating.Count.Factor ~ ., data = trainSet)
model.decision
```

Apply Decision Tree

```
## n= 301
##
## node), split, n, loss, yval, (yprob)
##       * denotes terminal node
##
## 1) root 301 148 2 (0.4916944 0.5083056)
##   2) Book.Rating< -0.0815365 126  14 1 (0.8888889 0.1111111) *
##   3) Book.Rating>=-0.0815365 175  36 2 (0.2057143 0.7942857) *
```

Prediction Probability prediction show us predicted probabilities that the book will be classified as above the mean or not.

```
probability.prediction <- predict(model.decision, newdata = testSet[-3,], type = 'class')
probability.prediction
```

```
##   1   2   4   5   6   7   8   9  10  11  12  13  14  15  16  17  18  19  20  21
##   2   1   2   2   2   2   1   1   1   2   2   2   1   2   2   2   1   1   1   1
##  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41
##   1   1   2   1   1   1   1   2   1   2   1   2   1   2   2   2   2   1   2   1
##  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61
```

```
##   2   2   1   2   1   1   2   2   1   2   1   1   2   1   2   2   2   2   2   2
##  62  63  64  65  66  67  68  69  70  71  72  73  74  75  76  77  78  79  80  81
##   1   2   2   2   2   2   2   2   1   1   2   1   2   2   2   1   2   1   2   1
##  82  83  84  85  86  87  88  89  90  91  92  93  94  95  96  97  98  99 100
##   2   1   2   1   2   1   1   1   1   2   2   1   2   2   1   1   2   1   1
## Levels: 1 2
```

```
levels(as.factor(probability.prediction))
```

Confusion Matrix

```
## [1] "1" "2"
```

```
levels(test_Set$Rating.Count.Factor)
```

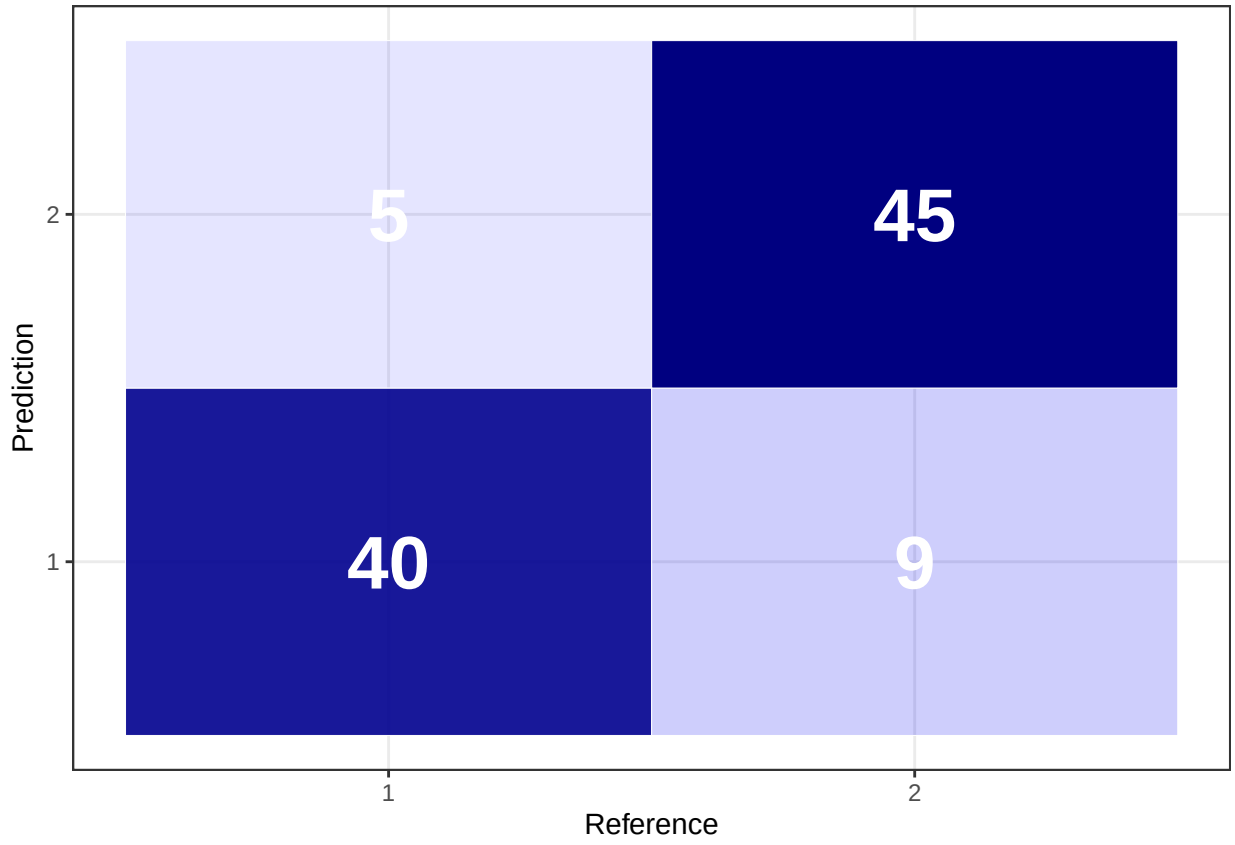
```
## [1] "1" "2"
```

```
conf.matrix <- confusionMatrix(as.factor(testSet[2:100, 3]),as.factor(probability.prediction))
decision.accuracy.balanced <- conf.matrix$overall['Accuracy']
conf.matrix
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  1  2
##          1 40  9
##          2  5 45
##
##                Accuracy : 0.8586
##                  95% CI : (0.7741, 0.9205)
##     No Information Rate : 0.5455
##     P-Value [Acc > NIR] : 3.182e-11
##
##                   Kappa : 0.7169
##
##  Mcnemar's Test P-Value : 0.4227
##
##             Sensitivity : 0.8889
##             Specificity : 0.8333
##          Pos Pred Value : 0.8163
##          Neg Pred Value : 0.9000
##              Prevalence : 0.4545
##          Detection Rate : 0.4040
##    Detection Prevalence : 0.4949
##       Balanced Accuracy : 0.8611
##
##        'Positive' Class : 1
##
```

The accuracy is 85%. We have 5 + 9 incorrect classifications.

```
# Heatmap visualization of confusion matrix
table <- data.frame(conf.matrix$table)
plotTable <- table %>%
  group_by(Prediction) %>%
  mutate(prop = Freq/sum(Freq))
ggplot(data =  plotTable, mapping = aes(x = Reference, y = Prediction, alpha = prop)) +
  geom_tile(aes(fill = Freq), colour = "white") +
  geom_text(aes(label = Freq), vjust = .5, fontface  = "bold", alpha = 1, color="white", size=10) +
  scale_fill_gradient(low = "blue", high = "navyblue") +
  theme_bw() + theme(legend.position = "none")
```

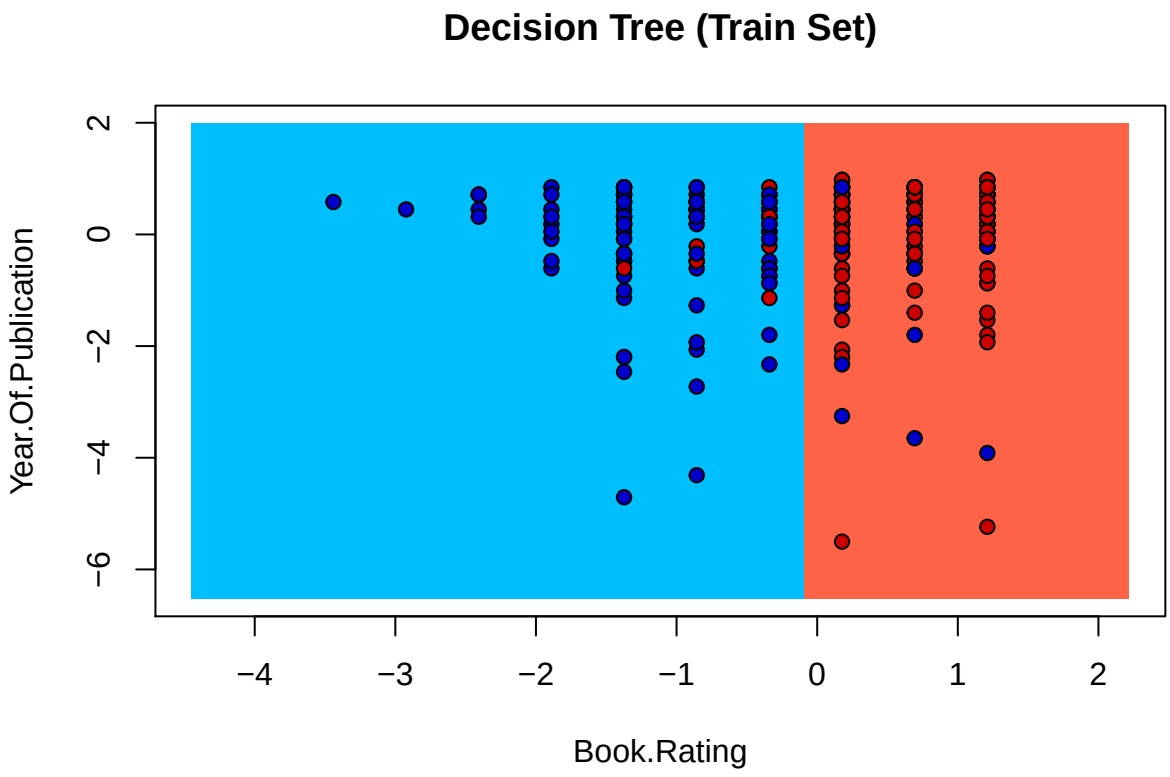


8.1.b) Visualization

Visualize Train Set Results

```
set <- trainSet
X1 <- seq(min(set[, 1]) - 1, max(set[, 1]) + 1, by = 0.01)
X2 <- seq(min(set[, 2]) - 1, max(set[, 2]) + 1, by = 0.01)
grid_set <- expand.grid(X1, X2)
colnames(grid_set) <- c('Book.Rating', 'Year.Of.Publication')
y_grid <- predict(model.decision, newdata = grid_set, type = 'class')
```

```
plot(set[, -3], main = 'Decision Tree (Train Set)',
     xlab = 'Book.Rating', ylab = 'Year.Of.Publication',
     xlim = range(X1), ylim = range(X2))
contour(X1, X2, matrix(as.numeric(y_grid), length(X1), length(X2)), add = TRUE)
points(grid_set, pch = '.', col = ifelse(y_grid == 1, 'deepskyblue', 'tomato'))
points(set, pch = 21, bg = ifelse(set[, 3] == 1, 'blue3', 'red3'))
```



Visualize Test Set Results

```
set <- testSet
X1 <- seq(min(set[, 1]) - 1, max(set[, 1]) + 1, by = 0.01)
X2 <- seq(min(set[, 2]) - 1, max(set[, 2]) + 1, by = 0.01)
grid_set <- expand.grid(X1, X2)
colnames(grid_set) <- c('Book.Rating', 'Year.Of.Publication')
y_grid <- predict(model.decision, newdata = grid_set, type = 'class')
plot(set[, -3], main = 'Decision Tree (Test Set)',
     xlab = 'Book.Rating', ylab = 'Year.Of.Publication',
     xlim = range(X1), ylim = range(X2))
contour(X1, X2, matrix(as.numeric(y_grid), length(X1), length(X2)), add = TRUE)
points(grid_set, pch = '.', col = ifelse(y_grid == 1, 'deepskyblue', 'tomato'))
points(set, pch = 21, bg = ifelse(set[, 3] == 1, 'blue3', 'red3'))
```

8.1.c)

As we see above the books that are classified as above the mean are mostly on the red side of the plot. Most of the books have a higher year of publication. The model classified true most of the data.

Z.b) Decision tree with imbalanced data

```
imbalanced.dataset$Rating.Count.Above.Mean <- ifelse(imbalanced.dataset$Rating.Count.Above.Mean == "No"
```

```
K <- imbalanced.dataset[sample,] %>% select("Book.Rating", "Year.Of.Publication", "Rating.Count.Above.M
row.names(K) <- NULL
head(K, n=5)
```

```
##    Book.Rating Year.Of.Publication Rating.Count.Above.Mean
## 1:           8            2002.000                       2
## 2:           8            2002.000                       2
## 3:           6            1992.000                       1
## 4:           9            1997.022                       2
## 5:           6            1997.022                       1
```

Turn the target feature to factor

```
K$Rating.Count.Factor <- factor(K$Rating.Count.Above.Mean, levels = c(1, 2))
```

```
K <- K[!is.na(K$Year.Of.Publication), ]
K <- select(K,-c(Rating.Count.Above.Mean))
head(K, n=5)
```

```
##    Book.Rating Year.Of.Publication Rating.Count.Factor
## 1:           8            2002.000                   2
## 2:           8            2002.000                   2
## 3:           6            1992.000                   1
## 4:           9            1997.022                   2
## 5:           6            1997.022                   1
```

Split data into Train and Test set

Rating.Count.Factor column is our dependent variable.

```
set.seed(123)
splitted <- sample.split(K$Rating.Count.Factor, SplitRatio = 0.75)
train_Set <- subset(K, splitted == TRUE)
test_Set <- subset(K, splitted == FALSE)
```

Feature Scaling

Feature scaling is a method used to normalize the range of independent variables or features of data.

We will scale all the features except our dependent variable, Rating.Count.Factor.

```
train_y = train_Set[,3]
test_y = test_Set[,3]

row.names(train_Set) <- NULL
row.names(test_Set) <- NULL

# Scaled test and train set
trainSet = data.frame(scale(train_Set[,-3]))
trainSet[,3] = train_y

testSet = data.frame(scale(test_Set[,-3]))
testSet[,3] = test_y
```

Apply Decision Tree

```
model.decision <- rpart(formula = Rating.Count.Factor ~ ., data = trainSet)
model.decision
```

```
## n= 301
##
## node), split, n, loss, yval, (yprob)
```

```
##       * denotes terminal node
##
##   1) root 301 129 2 (0.42857143 0.57142857)
##     2) Book.Rating< -0.04629781 130  33 1 (0.74615385 0.25384615)
##       4) Year.Of.Publication< -0.2416527 47   2 1 (0.95744681 0.04255319) *
##       5) Year.Of.Publication>=-0.2416527 83  31 1 (0.62650602 0.37349398)
##        10) Year.Of.Publication>=-0.1101464 64  15 1 (0.76562500 0.23437500)
##          20) Year.Of.Publication>=0.941904 21   0 1 (1.00000000 0.00000000) *
##          21) Year.Of.Publication< 0.941904 43  15 1 (0.65116279 0.34883721)
##            42) Year.Of.Publication< 0.8103977 33   8 1 (0.75757576 0.24242424)
##              84) Year.Of.Publication< 0.2200519 8   0 1 (1.00000000 0.00000000) *
##              85) Year.Of.Publication>=0.2200519 25   8 1 (0.68000000 0.32000000)
##               170) Year.Of.Publication>=0.3515582 16   2 1 (0.87500000 0.12500000) *
##               171) Year.Of.Publication< 0.3515582 9   3 2 (0.33333333 0.66666667) *
##            43) Year.Of.Publication>=0.8103977 10   3 2 (0.30000000 0.70000000) *
##        11) Year.Of.Publication< -0.1101464 19   3 2 (0.15789474 0.84210526) *
##     3) Book.Rating>=-0.04629781 171  32 2 (0.18713450 0.81286550)
##       6) Year.Of.Publication< -0.2416527 43  17 2 (0.39534884 0.60465116)
##        12) Year.Of.Publication>=-0.5046652 16   3 1 (0.81250000 0.18750000) *
##        13) Year.Of.Publication< -0.5046652 27   4 2 (0.14814815 0.85185185) *
##       7) Year.Of.Publication>=-0.2416527 128  15 2 (0.11718750 0.88281250) *
```

Prediction

Probability prediction show us predicted classes of a book if its average rating is above the mean or not.

```
probability.prediction <- predict(model.decision, newdata = testSet[-3,], type = 'class')
probability.prediction
```

```
##   1   2   4   5   6   7   8   9  10  11  12  13  14  15  16  17  18  19  20  21
##   2   2   2   1   2   1   2   1   2   1   2   2   2   2   2   2   2   2   2   1
##  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41
##   2   1   1   2   2   2   2   1   2   1   2   2   2   1   2   2   2   1   2   2
##  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61
##   2   1   2   1   2   2   1   2   1   2   2   2   2   2   2   2   2   2   1   2
##  62  63  64  65  66  67  68  69  70  71  72  73  74  75  76  77  78  79  80  81
##   1   2   2   2   2   2   2   2   2   2   2   2   1   2   2   1   2   1   1   2
##  82  83  84  85  86  87  88  89  90  91  92  93  94  95  96  97  98  99 100
##   2   2   2   1   2   1   1   2   2   2   1   2   2   1   1   2   1   2   2
## Levels: 1 2
```

Confusion Matrix

```
levels(as.factor(probability.prediction))
```

```
## [1] "1" "2"
```

```
levels(test_Set$Rating.Count.Factor)
```

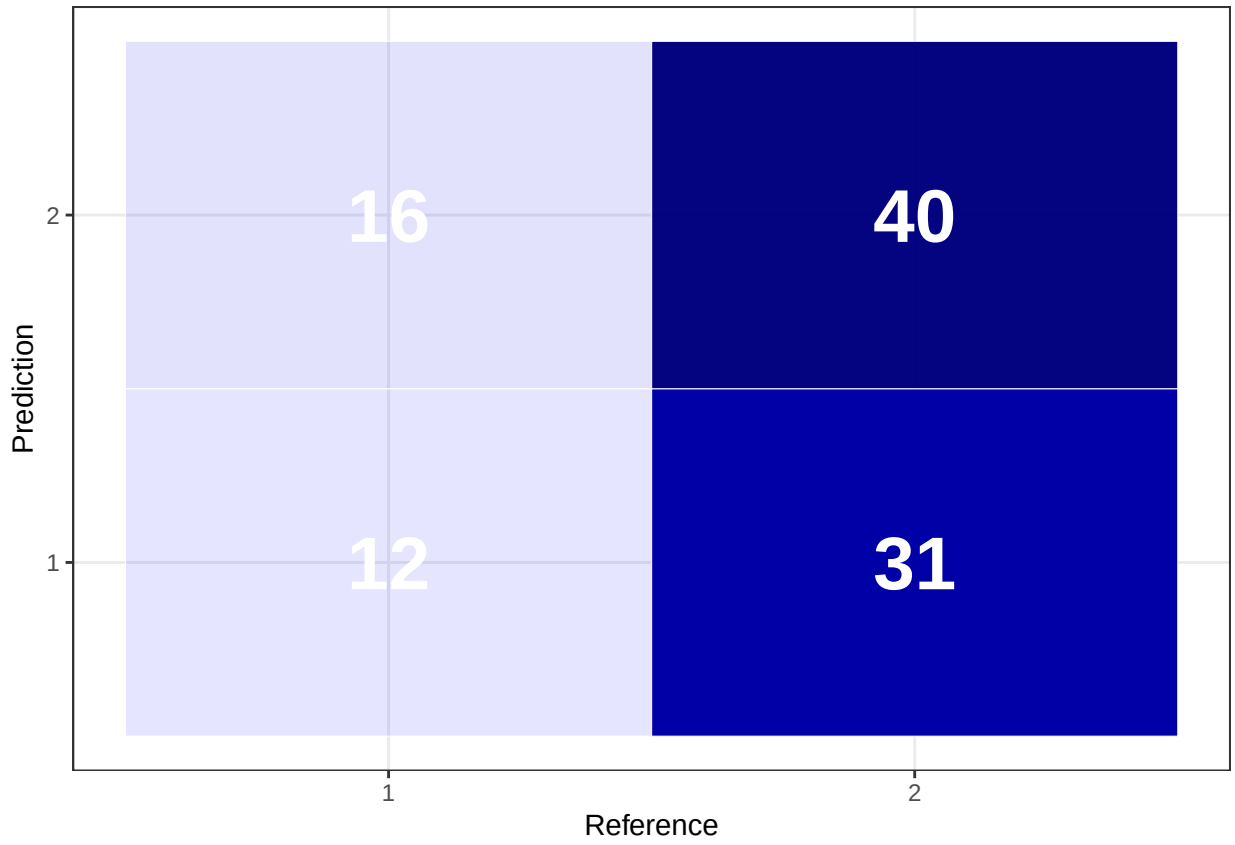
```
## [1] "1" "2"
```

```
conf.matrix <- confusionMatrix(as.factor(testSet[2:100, 3]),as.factor(probability.prediction))
decision.accuracy.imbalanced <- conf.matrix$overall['Accuracy']
conf.matrix
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  1  2
##          1 12 31
##          2 16 40
##
##                Accuracy : 0.5253
##                  95% CI : (0.4224, 0.6266)
##     No Information Rate : 0.7172
##     P-Value [Acc > NIR] : 0.99998
##
##                   Kappa : -0.0069
##
##  Mcnemar's Test P-Value : 0.04114
##
##             Sensitivity : 0.4286
##             Specificity : 0.5634
##          Pos Pred Value : 0.2791
##          Neg Pred Value : 0.7143
##              Prevalence : 0.2828
##          Detection Rate : 0.1212
##    Detection Prevalence : 0.4343
##       Balanced Accuracy : 0.4960
##
##        'Positive' Class : 1
##
```

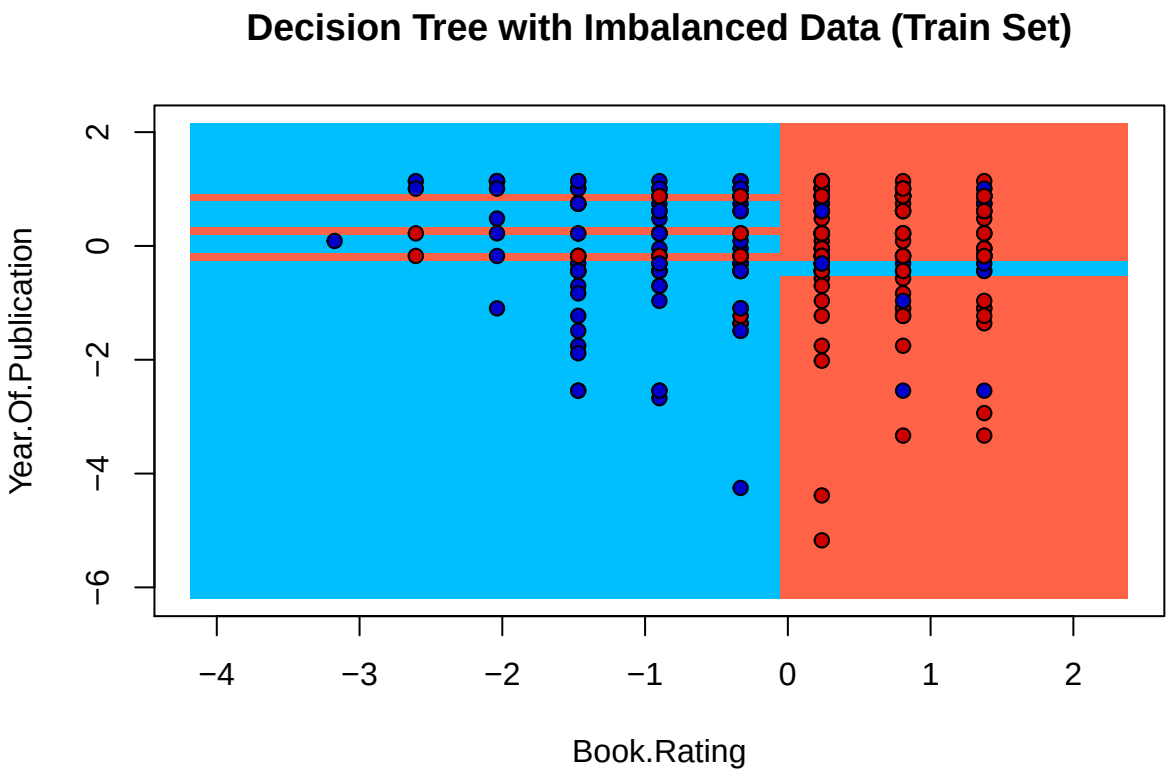
The accurary is 52%. We have 16 + 31 incorrect classifications.

```
# Heatmap visualization of confusion matrix
table <- data.frame(conf.matrix$table)
plotTable <- table %>%
  group_by(Prediction) %>%
  mutate(prop = Freq/sum(Freq))
ggplot(data =  plotTable, mapping = aes(x = Reference, y = Prediction, alpha = prop)) +
  geom_tile(aes(fill = Freq), colour = "white") +
  geom_text(aes(label = Freq), vjust = .5, fontface  = "bold", alpha = 1, color="white", size=10) +
  scale_fill_gradient(low = "blue", high = "navyblue") +
  theme_bw() + theme(legend.position = "none")
```



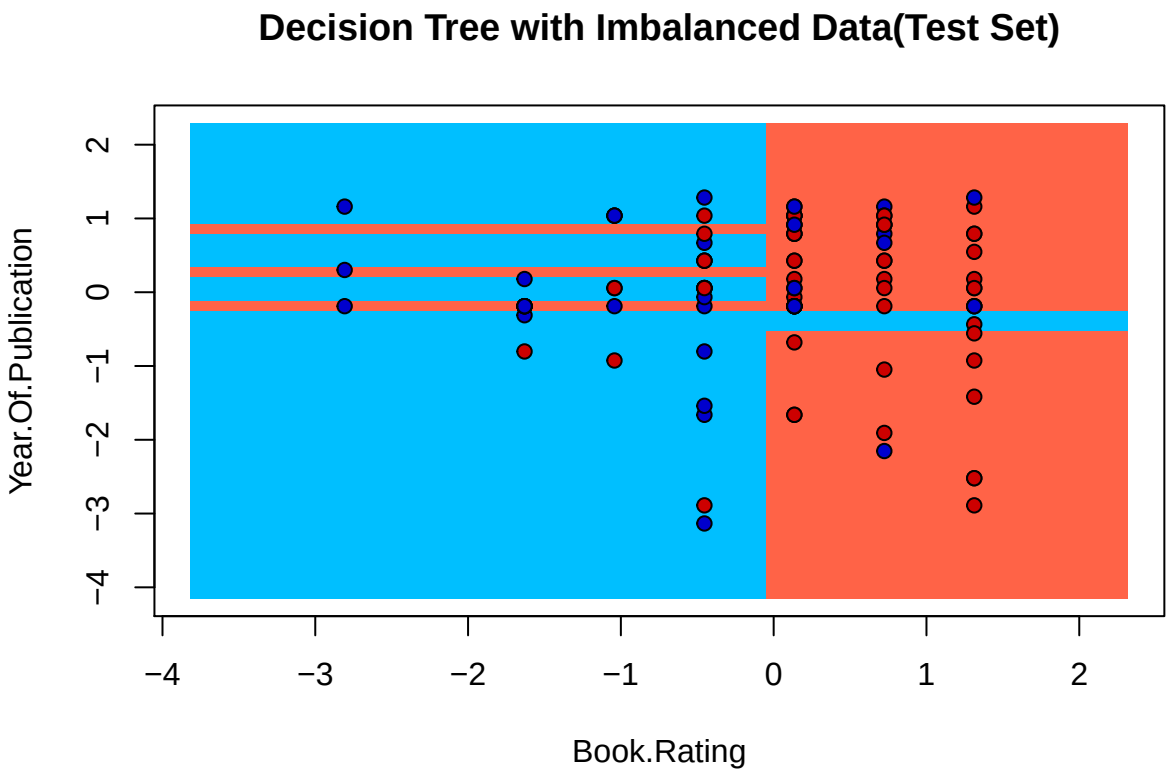
Visualize Train Set Results

```
set <- trainSet
X1 <- seq(min(set[, 1]) - 1, max(set[, 1]) + 1, by = 0.01)
X2 <- seq(min(set[, 2]) - 1, max(set[, 2]) + 1, by = 0.01)
grid_set <- expand.grid(X1, X2)
colnames(grid_set) <- c('Book.Rating', 'Year.Of.Publication')
y_grid <- predict(model.decision, newdata = grid_set, type = 'class')
plot(set[, -3], main = 'Decision Tree with Imbalanced Data (Train Set)',
     xlab = 'Book.Rating', ylab = 'Year.Of.Publication',
     xlim = range(X1), ylim = range(X2))
contour(X1, X2, matrix(as.numeric(y_grid), length(X1), length(X2)), add = TRUE)
points(grid_set, pch = '.', col = ifelse(y_grid == 1, 'deepskyblue', 'tomato'))
points(set, pch = 21, bg = ifelse(set[, 3] == 1, 'blue3', 'red3'))
```



Visualize Test Set Results

```
set <- testSet
X1 <- seq(min(set[, 1]) - 1, max(set[, 1]) + 1, by = 0.01)
X2 <- seq(min(set[, 2]) - 1, max(set[, 2]) + 1, by = 0.01)
grid_set <- expand.grid(X1, X2)
colnames(grid_set) <- c('Book.Rating', 'Year.Of.Publication')
y_grid <- predict(model.decision, newdata = grid_set, type = 'class')
plot(set[, -3], main = 'Decision Tree with Imbalanced Data(Test Set)',
     xlab = 'Book.Rating', ylab = 'Year.Of.Publication',
     xlim = range(X1), ylim = range(X2))
contour(X1, X2, matrix(as.numeric(y_grid), length(X1), length(X2)), add = TRUE)
points(grid_set, pch = '.', col = ifelse(y_grid == 1, 'deepskyblue', 'tomato'))
points(set, pch = 21, bg = ifelse(set[, 3] == 1, 'blue3', 'red3'))
```



Compare balanced data accuracy vs. imbalanced data accuracy

```
print("Accuracy of decision tree with balanced data")
```

```
## [1] "Accuracy of decision tree with balanced data"
```

```
decision.accuracy.balanced
```

```
##  Accuracy
## 0.8585859
```

```
print("Accuracy of decision tree with imbalanced data")
```

```
## [1] "Accuracy of decision tree with imbalanced data"
```

```
decision.accuracy.imbalanced
```

```
##  Accuracy
## 0.5252525
```

As seen above balanced data has higher accuracy than the imbalanced data.

8.2) K-Nearest Neighbors (K-NN) Algorithm Application

8.2.a) Functions and Arguments

```
K <- dataset[sample,] %>% select("Book.Rating", "Year.Of.Publication", "Rating.Count.Above.Mean")
head(K, n=5)
```

```
##   Book.Rating Year.Of.Publication Rating.Count.Above.Mean
## 1          10                1997                       2
## 2           3                2002                       1
## 4           8                1994                       2
## 5           7                2001                       1
## 6          10                1990                       2
```

Turn the target feature to factor

```
K$Rating.Count.Factor <- factor(K$Rating.Count.Above.Mean, levels = c(1, 2))
```

```
K <- K[!is.na(K$Year.Of.Publication), ]
K <- select(K,-c(Rating.Count.Above.Mean))
head(K, n=5)
```

```
##   Book.Rating Year.Of.Publication Rating.Count.Factor
## 1          10                1997                   2
## 2           3                2002                   1
## 4           8                1994                   2
## 5           7                2001                   1
## 6          10                1990                   2
```

Split data into Train and Test set Rating.Count.Factor column is our dependent variable.

```
set.seed(123)
splitted <- sample.split(K$Rating.Count.Factor, SplitRatio = 0.75)
train_Set <- subset(K, splitted == TRUE)
test_Set <- subset(K, splitted == FALSE)
```

Feature Scaling

Feature scaling is a method used to normalize the range of independent variables or features of data.

We will scale all the features except our dependent variable, Rating.Count.Factor.

```
train_y <- train_Set[,3]
test_y <- test_Set[,3]

# Scaled test and train set
trainSet <- data.frame(scale(train_Set[,-3]))
trainSet[,3] <- train_y

testSet <- data.frame(scale(test_Set[,-3]))
testSet[,3] <- test_y
```

```
y_pred <- knn(train = trainSet[, -3], test = testSet[, -3], cl = trainSet[, 3], k = 5, prob = TRUE)
y_pred
```

Apply KNN

```
##   [1] 2 1 2 2 2 2 1 1 1 1 2 2 2 1 2 2 2 1 1 2 1 1 1 2 1 1 1 1 2 1 2 1 2 1 2 2 1
##  [38] 2 1 2 1 2 1 1 2 1 1 2 2 1 2 1 1 1 1 2 2 2 2 2 2 1 2 2 2 2 2 2 2 2 1 2 1 2
##  [75] 2 2 2 2 1 2 1 2 1 2 1 2 2 1 1 1 2 2 2 2 2 1 1 2 1 1
## attr(,"prob")
##   [1] 0.8333333 0.7000000 1.0000000 0.8333333 1.0000000 1.0000000 0.6000000
##   [8] 0.8000000 0.8000000 1.0000000 1.0000000 0.8333333 1.0000000 0.8000000
##  [15] 0.8571429 1.0000000 0.6000000 1.0000000 0.7500000 0.8000000 1.0000000
##  [22] 1.0000000 0.7500000 0.8888889 1.0000000 0.6666667 0.7500000 1.0000000
##  [29] 1.0000000 0.7500000 0.8000000 0.8571429 0.8888889 1.0000000 0.8000000
##  [36] 0.8888889 0.8000000 0.6000000 0.8333333 0.9090909 0.8571429 0.8000000
##  [43] 0.6666667 0.8000000 1.0000000 1.0000000 1.0000000 0.8333333 0.7000000
##  [50] 0.8333333 1.0000000 0.7500000 0.7500000 0.6666667 1.0000000 0.7777778
##  [57] 0.7000000 0.8000000 0.6000000 0.8750000 0.7142857 0.8333333 0.5714286
##  [64] 0.5714286 0.8888889 1.0000000 0.6000000 1.0000000 0.8000000 0.8000000
##  [71] 0.8000000 0.9000000 0.8000000 0.7777778 1.0000000 0.6000000 0.8000000
##  [78] 0.8888889 0.7000000 1.0000000 1.0000000 0.5000000 1.0000000 0.8888889
##  [85] 1.0000000 0.8571429 0.8000000 0.8571429 0.8000000 1.0000000 1.0000000
##  [92] 0.8000000 0.8000000 0.8000000 1.0000000 1.0000000 0.8000000 0.5000000
##  [99] 0.8333333 0.8333333
## Levels: 1 2
```

```
conf.matrix <- confusionMatrix(as.factor(testSet[, 3]),as.factor(y_pred))
knn.accuracy<- conf.matrix$overall['Accuracy']
conf.matrix
```

Confusion Matrix

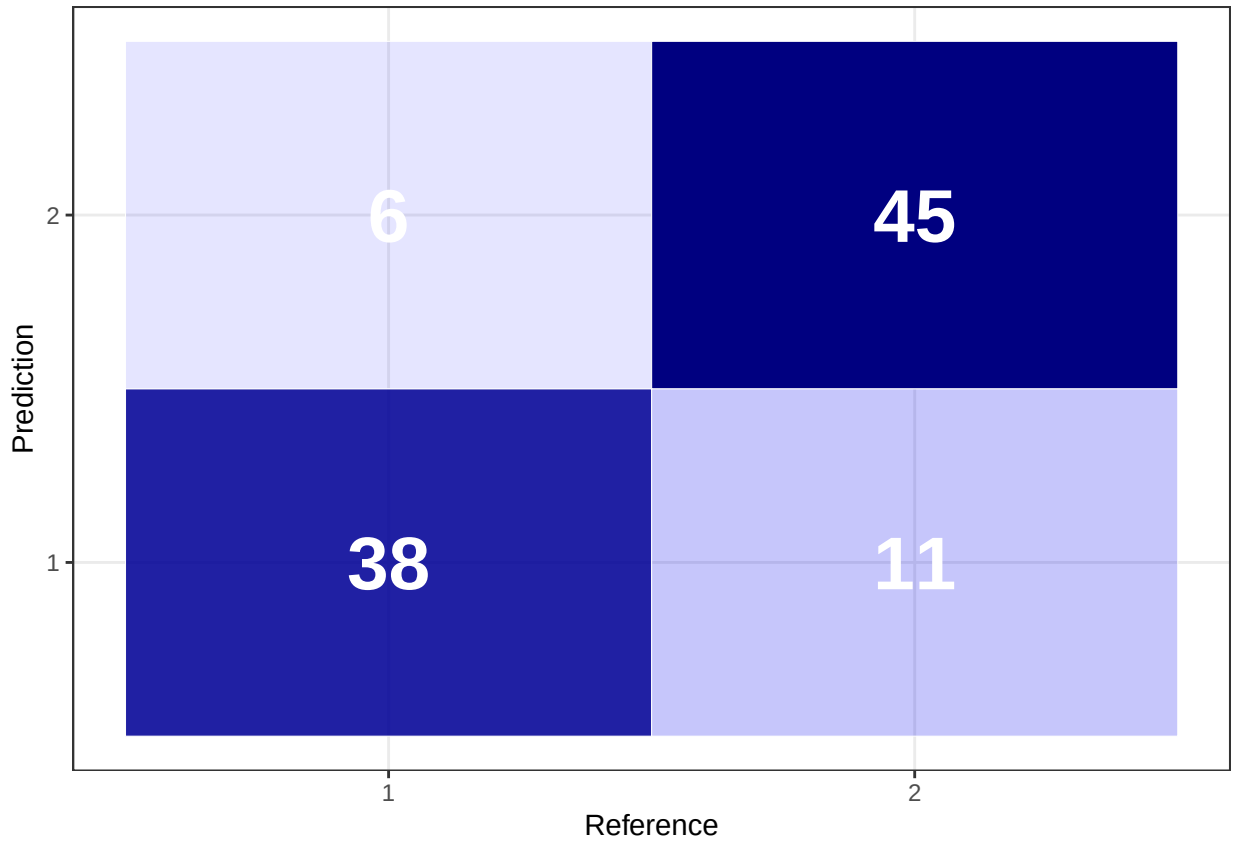
```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  1  2
##          1 38 11
##          2  6 45
##
##                Accuracy : 0.83
##                  95% CI : (0.7418, 0.8977)
##     No Information Rate : 0.56
##     P-Value [Acc > NIR] : 9.707e-09
##
##                   Kappa : 0.6592
##
##  Mcnemar's Test P-Value : 0.332
##
```

```
##             Sensitivity : 0.8636
##             Specificity : 0.8036
##          Pos Pred Value : 0.7755
##          Neg Pred Value : 0.8824
##              Prevalence : 0.4400
##          Detection Rate : 0.3800
##    Detection Prevalence : 0.4900
##       Balanced Accuracy : 0.8336
##
##        'Positive' Class : 1
##
```

The accurary is 83%. We have 11 + 6 incorrect classifications.

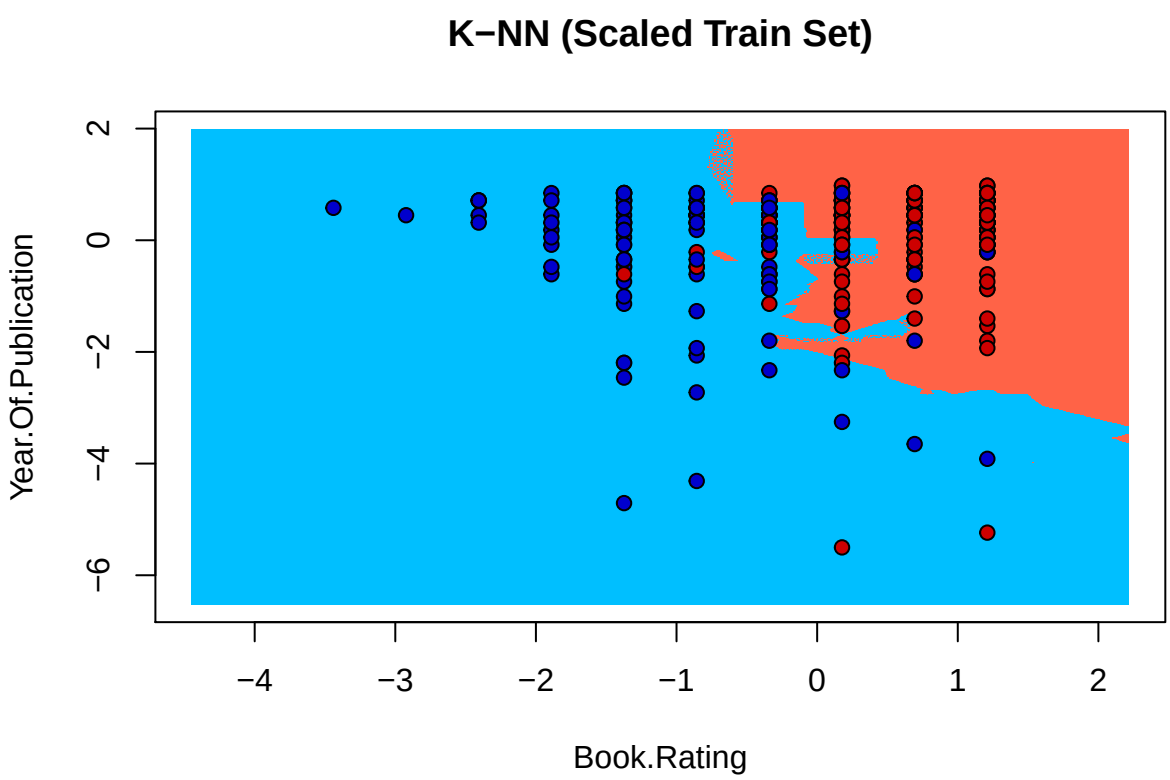
8.2.b) Visualization

```
# Heatmap visualization of confusion matrix
table <- data.frame(conf.matrix$table)
plotTable <- table %>%
  group_by(Prediction) %>%
  mutate(prop = Freq/sum(Freq))
ggplot(data =  plotTable, mapping = aes(x = Reference, y = Prediction, alpha = prop)) +
  geom_tile(aes(fill = Freq), colour = "white") +
  geom_text(aes(label = Freq), vjust = .5, fontface  = "bold", alpha = 1, color="white", size=10) +
  scale_fill_gradient(low = "blue", high = "navyblue") +
  theme_bw() + theme(legend.position = "none")
```



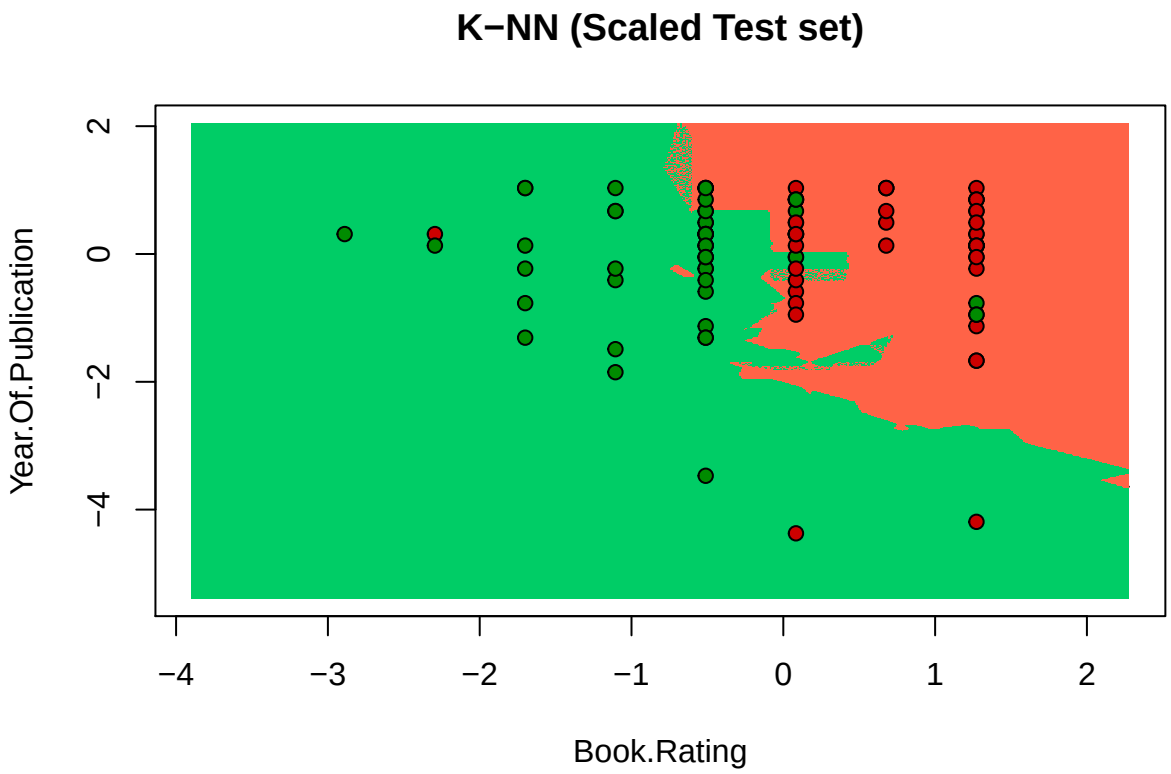
Visualize Results of Scaled Data

```
set <- trainSet
X1 <- seq(min(set[, 1]) - 1, max(set[, 1]) + 1, by = 0.01)
X2 <- seq(min(set[, 2]) - 1, max(set[, 2]) + 1, by = 0.01)
grid_set <- expand.grid(X1, X2)
colnames(grid_set) <- c('Book.Rating', 'Year.Of.Publication')
y_grid <- knn(train = trainSet[, -3], test = grid_set, cl = trainSet[, 3], k = 5)
plot(set[, -3], main = 'K-NN (Scaled Train Set)',
     xlab = 'Book.Rating', ylab = 'Year.Of.Publication',
     xlim = range(X1), ylim = range(X2))
contour(X1, X2, matrix(as.numeric(y_grid), length(X1), length(X2)), add = TRUE)
points(grid_set, pch = '.', col = ifelse(y_grid == 1, 'deepskyblue', 'tomato'))
points(set, pch = 21, bg = ifelse(set[, 3] == 1, 'blue3', 'red3'))
```



Train Set Results

```
set = testSet
X1 = seq(min(set[, 1]) - 1, max(set[, 1]) + 1, by = 0.01)
X2 = seq(min(set[, 2]) - 1, max(set[, 2]) + 1, by = 0.01)
grid_set = expand.grid(X1, X2)
colnames(grid_set) = c('Book.Rating', 'Year.Of.Publication')
y_grid = knn(train = trainSet[, -3], test = grid_set, cl = trainSet[, 3], k = 5)
plot(set[, -3],
     main = 'K-NN (Scaled Test set)',
     xlab = 'Book.Rating', ylab = 'Year.Of.Publication',
     xlim = range(X1), ylim = range(X2))
contour(X1, X2, matrix(as.numeric(y_grid), length(X1), length(X2)), add = TRUE)
points(grid_set, pch = '.', col = ifelse(y_grid == 1, 'springgreen3', 'tomato'))
points(set, pch = 21, bg = ifelse(set[, 3] == 1, 'green4', 'red3'))
```



Test Set Results

8.2.c) KNN is also good at classifying classes.

b) Compare decision tree model accuracy vs. knn model accuracy

```
print("Accuracy of decision tree")
```

```
## [1] "Accuracy of decision tree"
```

```
decision.accuracy.balanced
```

```
##  Accuracy
## 0.8585859
```

```
print("Accuracy of knn")
```

```
## [1] "Accuracy of knn"
```

```
knn.accuracy
```

```
## Accuracy
##     0.83
```

As seen above Decision tree has higher accuracy than KNN.