Capstone Project - Estimating the Alignment of Comic Characters using Maschine Learning Algorithms and the Comic-Characters-Dataset

Nikola Ell (Lunikell)

2023-11-08

Contents

1 Introduction 3
1.1 Goal of the Project 3
1.2 The Comic-Characters-Dataset 3
1.3 Key Steps 3

2 Methods and Analysis 4
2.1 Data Preparation 4
2.2 Data Wrangling 5
2.2.1 General Design of the Dataframe 5
2.2.2 Transform Character Data to Factors 6
2.2.3 Correct False Entries 6
2.2.4 Extract RGB Information of Eye Color 7
2.2.5 Extract RGB Information of Hair Color 8
2.2.6 Create Optimized Dataframe 10
2.3 Partitioning of the Dataset 10
2.4 Data Exploration and Visualization 11
2.4.1 Number of NA Values 11
2.4.2 Eye and Hair Colors 12
2.4.3 Alignment vs. Sex of the Characters 15
2.4.4 Alignment vs. Eye and Hair Color Values 17
2.5 Association Analysis 19
2.5.1 Correlation 19
2.5.2 ANOVA 19
2.5.3 Cramer's V 19
2.5.4 Implemention of the Association Analysis 20
2.5.5 Results of Association Analysis 21
2.6 Modelling Approach 24
2.6.1 (1) Guessing the Outcome 24
2.6.2 (2) Estimation using Numeric Data: Loess Estimation 25
2.6.3 (3) Estimation using Numeric and Categorical Data: Decision Trees 27

3 Results 33

4 Conclusion 34

5 Bibliography 35

All files used in this project can by downloaded on the corresponding GitHub page:

https://github.com/lunikell/CapstoneComicData

1 Introduction

Comics about super heroes are a great source of entertainment for many people. DC and Marvel are the most important publishers for those stories and the characters and stories they invented are a reflection of the times in which they were published. The analysis of the main features of these characters might help to find patterns and reveal underlying principles.

This report describes data analysis and the development of machine learning methodologies using the Comic-Characters-Dataset. [1] The project is conducted as part of the capstone in the lecture series “Data Science” by HarvardX on the platform edx. [2]

All calculations will be done using the R programming language. [3]

1.1 Goal of the Project

This project aims at estimating the alignment of a comic character based on other information provided in the dataset. The approaches will be evaluated by calculating the overall accuracy. The goal is to identify, evaluate and compare different possible strategies using the Comic-Characters-Dataset.

1.2 The Comic-Characters-Dataset

The Comic-Characters-Dataset was generated as part of the article “Comic Books Are Still Made By Men, For Men And About Men” written by Walt Hickey and published on FiveThirtyEight. [4]

It contains over 23.000 Marvel and DC comic characters with a variety of specific information, eg. their name, alignment or visual appearance. It is based on the DC Comic Database [5] and the Marvel Database [6].

The dataset is available as a package in R [1].

One big challenge in the dataset is that many character entries do not contain full information. Handling all the resulting `NA` values is one major task in this work.

1.3 Key Steps

This report contains the following parts

- Introduction
- Methods and Analysis
  - Data Preparation: Download and processing of the dataset
  - Data Wrangling: Optimize the dataset for analysis and modelling by extracting new information
  - Partitioning: Create a train and a test dataset
  - Data Exploration and Visualization: Get to know key features of the dataset to find out relevant aspects for modelling approaches
  - Association Analysis: Determine the association of the variables among each other
  - Modelling Approach: Development and evaluation of the machine learning algorithms
- Results: Evaluation of the results and analysis of the model performance
- Conclusion: Summary and possible future work
- Bibliography

2 Methods and Analysis

This section describes the methods used to prepare the dataset for the analysis. Afterwards the dataset will be explored to gather information about its key features. An association analysis reveals the realtion of the variables among each other. Based on these insights a modeling approach will be discussed.

2.1 Data Preparation

The procedure to prepare the dataframe is described in the R documentation of the dataset. [1]

By executing this code separate datasets for Marvel and DC characters are downloaded from its github page. They are combined, then the `year` and `align` variables are optimized as described in the documentation.

```
library(readr)
library(dplyr)
library(tidyr)
library(lubridate)
library(janitor)
library(tidyverse)

# Get DC characters:
comic_characters_dc <-
    paste("https://github.com/fivethirtyeight/data/raw",
           "/master/comic-characters/dc-wikia-data.csv", sep="") %>%
    read_csv() %>%
    clean_names() %>%
    mutate(publisher = "DC")

# Get Marvel characters:
comic_characters_marvel <-
    paste("https://github.com/fivethirtyeight/data/raw",
          "/master/comic-characters/marvel-wikia-data.csv", sep="") %>%
    read_csv() %>%
    clean_names() %>%
    mutate(publisher = "Marvel")

# Merge two dataset and perform further data wrangling:
comic_characters <-
    comic_characters_dc %>%
    bind_rows(comic_characters_marvel) %>%
    separate(first_appearance, c("year2", "month"), ", ", remove = FALSE) %>%
    mutate(
        # If month was missing, set as January and day as 01:
        month = ifelse(is.na(month), "01", month),
        day = "01",
        # Note some years missing:
        date = ymd(paste(year, month, day, sep = "-")),
        align = factor(
        align,
        levels = c("Bad Characters", "Reformed Criminals",
                   "Netural Characters", "Good Characters"),
        ordered = TRUE)
        ) %>%
    select(publisher, everything(), -c(year2, day))
```

2.2 Data Wrangling

2.2.1 General Design of the Dataframe

The first five ratings in the given dataframe show key features of its general design: [7]

```
library(kableExtra)

comic_characters[1:5,1:5] %>%
    kbl(booktabs = T) %>%
    kable_styling(font_size=8.5, latex_options="hold_position")
```

publisher	page_id	name	urlslug	id
DC	1422	Batman (Bruce Wayne)	\/wiki\/Batman_(Bruce_Wayne)	Secret Identity
DC	23387	Superman (Clark Kent)	\/wiki\/Superman_(Clark_Kent)	Secret Identity
DC	1458	Green Lantern (Hal Jordan)	\/wiki\/Green_Lantern_(Hal_Jordan)	Secret Identity
DC	1659	James Gordon (New Earth)	\/wiki\/James_Gordon_(New_Earth)	Public Identity
DC	1576	Richard Grayson (New Earth)	\/wiki\/Richard_Grayson_(New_Earth)	Secret Identity

```
comic_characters[1:5,6:12] %>%
    kbl(booktabs = T) %>%
    kable_styling(font_size=8.5, latex_options="hold_position")
```

align	eye	hair	sex	gsm	alive	appearances
Good Characters	Blue Eyes	Black Hair	Male Characters	NA	Living Characters	3093
Good Characters	Blue Eyes	Black Hair	Male Characters	NA	Living Characters	2496
Good Characters	Brown Eyes	Brown Hair	Male Characters	NA	Living Characters	1565
Good Characters	Brown Eyes	White Hair	Male Characters	NA	Living Characters	1316
Good Characters	Blue Eyes	Black Hair	Male Characters	NA	Living Characters	1237

```
comic_characters[1:5,13:16] %>%
    kbl(booktabs = T) %>%
    kable_styling(font_size=8.5, latex_options="hold_position")
```

first_appearance	month	year	date
1939, May	May	1939	1939-05-01
1986, October	October	1986	1986-10-01
1959, October	October	1959	1959-10-01
1987, February	February	1987	1987-02-01
1940, April	April	1940	1940-04-01

The dataframe contains 16 columns: [1]

- **publisher:** company who published the comic including this character (DC or Marvel)
- **page_id:** unique Id to identify the character (in the wiki)
- **name:** name of the comic character and additional information in brackets (eg. the civil name)
- **urlslug:** url in the wiki
- **id:** information about the identity of the character (eg. secret or public)
- **align:** the alignment of the character (eg. good or bad)
- **eye:** eye color of the character
- **hair:** hair color of the character
- **sex:** sex of the character
- **gsm:** states if character is a gender or sexual minority
- **alive:** states if character is deceased or alive
- **appearances:** number of appearances in a comic book (as at Sep. 2, 2014)

- **first_appearance:** month and year of first appearance of the character in a comic book
- **month:** month of first appearance of the character in a comic book
- **year:** year of first appearance of the character in a comic book
- **date:** date of first appearance of the character in a comic book

In the following sections data wrangling is performed to extract addition information from the dataframe, which can be used for the estimating algorithms later.

2.2.2 Transform Character Data to Factors

Most data in the dataframe is stored as characters. To perform the analysis they are transformed to factor variables.

```
comic_characters2 <- comic_characters %>%
                     mutate(across(c(1,5,6,7,8,9,10,11,14),as.factor))
```

2.2.3 Correct False Entries

The levels of the `eye` variable in the dataframe contain some `Auburn Hair` entries

```
levels(comic_characters2$eye)
```

```
##  [1] "Amber Eyes"          "Auburn Hair"         "Black Eyeballs"
##  [4] "Black Eyes"          "Blue Eyes"           "Brown Eyes"
##  [7] "Compound Eyes"       "Gold Eyes"           "Green Eyes"
## [10] "Grey Eyes"           "Hazel Eyes"          "Magenta Eyes"
## [13] "Multiple Eyes"       "No Eyes"             "One Eye"
## [16] "Orange Eyes"         "Photocellular Eyes"  "Pink Eyes"
## [19] "Purple Eyes"         "Red Eyes"            "Silver Eyes"
## [22] "Variable Eyes"       "Violet Eyes"         "White Eyes"
## [25] "Yellow Eyeballs"     "Yellow Eyes"
```

The following entries are affected:

```
comic_characters2 %>% filter(!is.na(eye)) %>% filter(eye =="Auburn Hair")
```

```
## # A tibble: 7 x 16
##   publisher page_id name       urlslug id    align eye   hair  sex   gsm   alive
##   <fct>       <dbl> <chr>      <chr>   <fct> <ord> <fct> <fct> <fct> <fct> <fct>
## 1 DC          80676 Marcia Ki~ "\\/wi~ Publ~ Good~ Aubu~ <NA>  Fema~ <NA>  Livi~
## 2 DC         146812 Anthony A~ "\\/wi~ Publ~ <NA>  Aubu~ <NA>  Male~ <NA>  Livi~
## 3 DC         114487 Tawna (Ne~ "\\/wi~ <NA>  Good~ Aubu~ <NA>  Fema~ <NA>  Livi~
## 4 DC         192614 Razerkut ~ "\\/wi~ Secr~ Bad ~ Aubu~ <NA>  Fema~ <NA>  Livi~
## 5 DC         130938 Carla Dra~ "\\/wi~ Publ~ Bad ~ Aubu~ <NA>  Fema~ <NA>  Livi~
## 6 DC          71092 Madolyn C~ "\\/wi~ <NA>  <NA>  Aubu~ <NA>  Fema~ <NA>  Dece~
## 7 DC           4935 Sally Mil~ "\\/wi~ Publ~ <NA>  Aubu~ <NA>  Fema~ <NA>  Livi~
## # i 5 more variables: appearances <dbl>, first_appearance <chr>, month <fct>,
## #   year <dbl>, date <date>
```

They are all DC characters. A quick search on the DC-Comic-Database [5] (on which this dataset is based) shows, that all characters really have auburn hair and different eye colors. Thus these entries are false and should be corrected.

The following code corrects the hair information to `Auburn Hair`. The eye color is added based on pictures in the DC database.

```
comic_characters2[comic_characters2$page_id == 80676,  "hair"] <- "Auburn Hair"
comic_characters2[comic_characters2$page_id == 80676,  "eye"]  <- "Blue Eyes"
```

```
comic_characters2[comic_characters2$page_id == 146812, "hair"] <- "Auburn Hair"
comic_characters2[comic_characters2$page_id == 146812, "eye"]  <- "Brown Eyes"

comic_characters2[comic_characters2$page_id == 114487, "hair"] <- "Auburn Hair"
comic_characters2[comic_characters2$page_id == 114487, "eye"]  <- "Green Eyes"

comic_characters2[comic_characters2$page_id == 192614, "hair"] <- "Auburn Hair"
comic_characters2[comic_characters2$page_id == 192614, "eye"]  <- "Yellow Eyes"

comic_characters2[comic_characters2$page_id == 130938, "hair"] <- "Auburn Hair"
comic_characters2[comic_characters2$page_id == 130938, "eye"]  <- "Blue Eyes"

comic_characters2[comic_characters2$page_id ==  71092, "hair"] <- "Auburn Hair"
comic_characters2[comic_characters2$page_id ==  71092, "eye"]  <- "Brown Eyes"

comic_characters2[comic_characters2$page_id ==   4935, "hair"] <- "Auburn Hair"
comic_characters2[comic_characters2$page_id ==   4935, "eye"]  <-  NA
```

Then all levels that are not used in the dataframe are dropped:

```
comic_characters2 <- droplevels(comic_characters2)
```

Now the level `Auburn Hair` is not present in the `eye` variable anymore:

```
levels(comic_characters2$eye)
```

```
##  [1] "Amber Eyes"         "Black Eyeballs"     "Black Eyes"
##  [4] "Blue Eyes"          "Brown Eyes"         "Compound Eyes"
##  [7] "Gold Eyes"          "Green Eyes"         "Grey Eyes"
## [10] "Hazel Eyes"         "Magenta Eyes"       "Multiple Eyes"
## [13] "No Eyes"            "One Eye"            "Orange Eyes"
## [16] "Photocellular Eyes" "Pink Eyes"          "Purple Eyes"
## [19] "Red Eyes"           "Silver Eyes"        "Variable Eyes"
## [22] "Violet Eyes"        "White Eyes"         "Yellow Eyeballs"
## [25] "Yellow Eyes"
```

2.2.4 Extract RGB Information of Eye Color

The eye color is stored in a character string. The dataframe contains 26 different colors.

```
levels(comic_characters2$eye)
```

```
##  [1] "Amber Eyes"         "Black Eyeballs"     "Black Eyes"
##  [4] "Blue Eyes"          "Brown Eyes"         "Compound Eyes"
##  [7] "Gold Eyes"          "Green Eyes"         "Grey Eyes"
## [10] "Hazel Eyes"         "Magenta Eyes"       "Multiple Eyes"
## [13] "No Eyes"            "One Eye"            "Orange Eyes"
## [16] "Photocellular Eyes" "Pink Eyes"          "Purple Eyes"
## [19] "Red Eyes"           "Silver Eyes"        "Variable Eyes"
## [22] "Violet Eyes"        "White Eyes"         "Yellow Eyeballs"
## [25] "Yellow Eyes"
```

To obtain information about the relation between these colors (eg. which are more reddish or blueish) the color names are transformed into its HEX and RGB color values. The colors are based on the list of color shades on wikipedia [8] and the website color-name.com [9].

First a new column named `eye_HEX` is created and initialized with NA-values.

```
comic_characters2$eye_HEX <- NA
```

Then this column is filled with the HEX values based on the entry in the `eye` column. Some levels of the `eye`variable don't describe a color, but the form of eyes of this character, eg. `Compound Eyes` or `No Eyes`. The HEX value for these entries is set to `NA`, as there is no color information available.

```
comic_characters2$eye_HEX[comic_characters2$eye == "Amber Eyes"]         <- "#FFBF00"
comic_characters2$eye_HEX[comic_characters2$eye == "Black Eyeballs"]     <- "#000000"
comic_characters2$eye_HEX[comic_characters2$eye == "Black Eyes"]         <- "#000000"
comic_characters2$eye_HEX[comic_characters2$eye == "Blue Eyes"]          <- "#0000FF"
comic_characters2$eye_HEX[comic_characters2$eye == "Brown Eyes"]         <- "#964B00"
comic_characters2$eye_HEX[comic_characters2$eye == "Compound Eyes"]      <-  NA
comic_characters2$eye_HEX[comic_characters2$eye == "Gold Eyes"]          <- "#FFD700"
comic_characters2$eye_HEX[comic_characters2$eye == "Green Eyes"]         <- "#00FF00"
comic_characters2$eye_HEX[comic_characters2$eye == "Grey Eyes"]          <- "#808080"
comic_characters2$eye_HEX[comic_characters2$eye == "Hazel Eyes"]         <- "#C9C789"
comic_characters2$eye_HEX[comic_characters2$eye == "Magenta Eyes"]       <- "#FF00FF"
comic_characters2$eye_HEX[comic_characters2$eye == "Multiple Eyes"]      <-  NA
comic_characters2$eye_HEX[comic_characters2$eye == "No Eyes"]            <-  NA
comic_characters2$eye_HEX[comic_characters2$eye == "One Eye"]            <-  NA
comic_characters2$eye_HEX[comic_characters2$eye == "Orange Eyes"]        <- "#FF8000"
comic_characters2$eye_HEX[comic_characters2$eye == "Photocellular Eyes"] <-  NA
comic_characters2$eye_HEX[comic_characters2$eye == "Pink Eyes"]          <- "#FFC0CB"
comic_characters2$eye_HEX[comic_characters2$eye == "Purple Eyes"]        <- "#800080"
comic_characters2$eye_HEX[comic_characters2$eye == "Red Eyes"]           <- "#FF0000"
comic_characters2$eye_HEX[comic_characters2$eye == "Silver Eyes"]        <- "#C0C0C0"
comic_characters2$eye_HEX[comic_characters2$eye == "Variable Eyes"]      <-  NA
comic_characters2$eye_HEX[comic_characters2$eye == "Violet Eyes"]        <- "#8000FF"
comic_characters2$eye_HEX[comic_characters2$eye == "White Eyes"]         <- "#FFFFFF"
comic_characters2$eye_HEX[comic_characters2$eye == "Yellow Eyeballs"]    <- "#FFFF00"
comic_characters2$eye_HEX[comic_characters2$eye == "Yellow Eyes"]        <- "#FFFF00"
```

Afterwards the HEX values are converted to RGB and stored in three new, separate columns `eye_R`, `eye_G` and `eye_B`.

```
comic_characters2$eye_R <- col2rgb(comic_characters2$eye_HEX)[1,]
comic_characters2$eye_G <- col2rgb(comic_characters2$eye_HEX)[2,]
comic_characters2$eye_B <- col2rgb(comic_characters2$eye_HEX)[3,]
```

The `col2rgb` command interprets `NA` values as black (255-255-255). They are set to the median value of the column manually.

```
comic_characters2$eye_R[is.na(comic_characters2$eye_HEX)] <-
                       median(comic_characters2$eye_R, na.rm = TRUE)
comic_characters2$eye_G[is.na(comic_characters2$eye_HEX)] <-
                       median(comic_characters2$eye_G, na.rm = TRUE)
comic_characters2$eye_B[is.na(comic_characters2$eye_HEX)] <-
                       median(comic_characters2$eye_B, na.rm = TRUE)
```

2.2.5 Extract RGB Information of Hair Color

The same operations are performed to extract the RGB color information from the hair column.

There are the following levels in the `hair` variable:

```
levels(comic_characters2$hair)
```

```
##  [1] "Auburn Hair"           "Bald"                  "Black Hair"           
##  [4] "Blond Hair"            "Blue Hair"             "Bronze Hair"          
##  [7] "Brown Hair"            "Dyed Hair"             "Gold Hair"            
## [10] "Green Hair"            "Grey Hair"             "Light Brown Hair"     
## [13] "Magenta Hair"          "No Hair"               "Orange Hair"          
## [16] "Orange-brown Hair"     "Pink Hair"             "Platinum Blond Hair"  
## [19] "Purple Hair"           "Red Hair"              "Reddish Blond Hair"   
## [22] "Reddish Brown Hair"    "Silver Hair"           "Strawberry Blond Hair"
## [25] "Variable Hair"         "Violet Hair"           "White Hair"           
## [28] "Yellow Hair"
```

Then the HEX color values are added in the column `hair_HEX`

```
comic_characters2$hair_HEX <- NA
comic_characters2$hair_HEX[comic_characters2$hair == "Auburn Hair"]           <- "#A52A2A"
comic_characters2$hair_HEX[comic_characters2$hair == "Bold"]                  <-  NA
comic_characters2$hair_HEX[comic_characters2$hair == "Black Hair"]            <- "#000000"
comic_characters2$hair_HEX[comic_characters2$hair == "Blond Hair"]            <- "#F5DEB3"
comic_characters2$hair_HEX[comic_characters2$hair == "Blue Hair"]             <- "#0000FF"
comic_characters2$hair_HEX[comic_characters2$hair == "Bronze Hair"]           <- "#CD7F32"
comic_characters2$hair_HEX[comic_characters2$hair == "Brown Hair"]            <- "#964B00"
comic_characters2$hair_HEX[comic_characters2$hair == "Gold Hair"]             <- "#FFD700"
comic_characters2$hair_HEX[comic_characters2$hair == "Green Hair"]            <- "#00FF00"
comic_characters2$hair_HEX[comic_characters2$hair == "Grey Hair"]             <- "#808080"
comic_characters2$hair_HEX[comic_characters2$hair == "Light Brown Hair"]      <- "#BD9A7A"
comic_characters2$hair_HEX[comic_characters2$hair == "Magenta Hair"]          <- "#FF00FF"
comic_characters2$hair_HEX[comic_characters2$hair == "No Hair"]               <-  NA
comic_characters2$hair_HEX[comic_characters2$hair == "Orange Hair"]           <- "#FF8000"
comic_characters2$hair_HEX[comic_characters2$hair == "Orange-brown Hair"]     <- "#C36F29"
comic_characters2$hair_HEX[comic_characters2$hair == "Pink Hair"]             <- "#FFC0CB"
comic_characters2$hair_HEX[comic_characters2$hair == "Platinum Blond Hair"]   <- "#F3ECDE"
comic_characters2$hair_HEX[comic_characters2$hair == "Purple Hair"]           <- "#800080"
comic_characters2$hair_HEX[comic_characters2$hair == "Red Hair"]              <- "#FF0000"
comic_characters2$hair_HEX[comic_characters2$hair == "Reddish Blond Hair"]    <- "#FF7F50"
comic_characters2$hair_HEX[comic_characters2$hair == "Reddish Brown Hair"]    <- "#A52A2A"
comic_characters2$hair_HEX[comic_characters2$hair == "Silver Hair"]           <- "#C0C0C0"
comic_characters2$hair_HEX[comic_characters2$hair == "Strawberry Blond Hair"] <- "#FFE5B4"
comic_characters2$hair_HEX[comic_characters2$hair == "Variable Hair"]         <-  NA
comic_characters2$hair_HEX[comic_characters2$hair == "Violet Hair"]           <- "#8000FF"
comic_characters2$hair_HEX[comic_characters2$hair == "White Hair"]            <- "#FFFFFF"
comic_characters2$hair_HEX[comic_characters2$hair == "Yellow Hair"]           <- "#FFFF00"
```

Afterwards the values are converted to RGB and the `NA` values corrected:

```
comic_characters2$hair_R <- col2rgb(comic_characters2$hair_HEX)[1,]
comic_characters2$hair_G <- col2rgb(comic_characters2$hair_HEX)[2,]
comic_characters2$hair_B <- col2rgb(comic_characters2$hair_HEX)[3,]

comic_characters2$hair_R[is.na(comic_characters2$hair_HEX)] <-
                        median(comic_characters2$hair_R, na.rm = TRUE)
comic_characters2$hair_G[is.na(comic_characters2$hair_HEX)] <-
                        median(comic_characters2$hair_G, na.rm = TRUE)
comic_characters2$hair_B[is.na(comic_characters2$hair_HEX)] <-
                        median(comic_characters2$hair_B, na.rm = TRUE)
```

2.2.6 Create Optimized Dataframe

Finally the columns which are not needed in the further examination are removed using the `select` command from the `tidyverse` package. [10]

```
comic_characters2 <- comic_characters2 %>%
  subset(select = -c(urlslug, first_appearance,date,eye_HEX,hair_HEX))
```

The goal in this report is to estimate the alignment of the characters. But the `align` variable contains a high amount of `NA` values. Their share can be calculated using the following code:

```
share_NA_in_align <- sum(is.na(comic_characters2$align)) / nrow(comic_characters2) * 100
```

It results to 26.6 %. Those `NA` values do not contain any information and cannot be estimated. They are removed from the dataframe.

```
comic_characters2 <- comic_characters2 %>%
  filter(!is.na(align))
```

The optimized `comic_characters2` dataframe contains the following columns:

- **`publisher`:** company who published the comic including this character (DC or Marvel)
- **`page_id`:** unique Id to identify the character (in the wiki)
- **`name`:** name of the comic character and additional information in brackets (eg. the civil name)
- **`id`:** information about the identity of the character (eg. secret or public)
- **`align`:** the alignment of the character (eg. good or bad)
- **`eye`:** eye color of the character
- **`hair`:** hair color of the character
- **`sex`:** sex of the character
- **`gsm`:** states if character is a gender or sexual minority
- **`alive`:** states if character is deceased or alive
- **`appearances`:** number of appearances in a comic book (as at Sep. 2, 2014)
- **`month`:** month of first appearance of the character in a comic book
- **`year`:** year of first appearance of the character in a comic book
- **`eye_R`:** RGB value for red (R) of eye color
- **`eye_G`:** RGB value for red (G) of eye color
- **`eye_B`:** RGB value for red (B) of eye color
- **`hair_R`:** RGB value for red (R) of hair color
- **`hair_G`:** RGB value for red (G) of hair color
- **`hair_B`:** RGB value for red (B) of hair color

2.3 Partitioning of the Dataset

The dataframe is partitioned into a `train` and a `test` dataset using the `caret` package. [11]

```
library(caret)

set.seed(1)
test_index <- createDataPartition(comic_characters2$align, times = 1,
                                  p = 0.1, list = FALSE)
test  <- comic_characters2[test_index,]      # 10% of data
train <- comic_characters2[-test_index,]     # 90% of data
```

The `train` set includes with 90% and 15376 entries the majority of the data and is used to train the algorithm. To evaluate the methods the rest of the data in the `test` set with 1710 entries will be used. This partitioning makes sure that there is enough data for testing while only a minimum of the data is lost for training.

All exploration will be done using the `train` dataset.

2.4 Data Exploration and Visualization

The plots for the visualization are inspired by the R-Graph-Gallery [12] and created using the **ggplot2** package. [13]

2.4.1 Number of NA Values

Most columns in the dataframe contain lots of NA values. Figure 1 shows a radar chart of the NA values. It is computed using the **fmbs** package. [14]

```
library(fmsb)
```

The columns with the RGB color information, which were derived from the color names, include zero NA values as They were removed manually and set to the median values (see chapters 2.2.4 and 2.2.5). That is why they are excluded from the following visualization.

A dataframe data_NA is extracted from train. It summarizes the number of NA values.

```
data_NA <- train %>%
           subset(select = -c(eye_R,eye_G,eye_B,hair_R,hair_G,hair_B)) %>%
           summarise(across(everything(), ~ sum(is.na(.))))
```

The values are converted to percent of the whole train dataframe.

```
data_NA <- data_NA/nrow(train)*100
```

The results can be seen in this table:

```
data_NA %>% pivot_longer(everything()) %>%
            arrange(desc(value)) %>%
            kbl(booktabs = T, linesep = "") %>%
            kable_styling(font_size=8.5, latex_options="hold_position")
```

name	value
gsm	99.3171176
eye	56.2955255
hair	27.1331946
id	22.5806452
appearances	5.7817378
sex	3.5509886
year	3.5054631
alive	0.0130073
publisher	0.0000000
page_id	0.0000000
name	0.0000000
align	0.0000000
month	0.0000000

To scale the plot additional columns have to be added which contain the minimum an maximum values for the plot axis.

```
data_NA <- rbind(rep(100,13) , rep(0,13) , data_NA)
```

Then the plot is created:

```
radarchart(data_NA, axistype=1 ,
           pcol=rgb(0,0.2,0.7,0.9),
           pfcol=rgb(0,0.2,0.7,0.2),
           plwd=4,
           cglcol="grey",
           cglty=1,
           axislabcol="grey",
           caxislabels=seq(0,100,25),
           cglwd=0.8,
           vlcex=0.6
           )
```

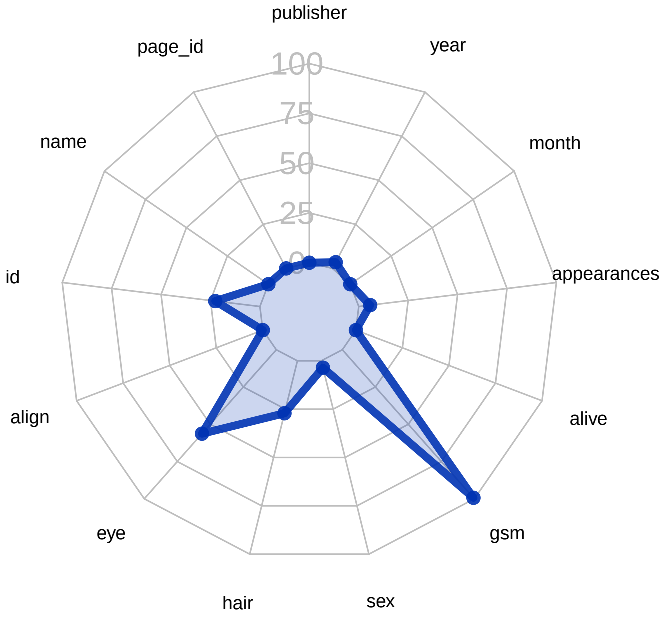

Figure 1: NA values in the train dataframe by variable in percent

The number of `NA` values differs over the variables. In the `gsm` column nearly all values (99,3%) are `NA`. That is why this variable is not suited for machine learning as only very few elements contain any information at all.

The `eye` column contains about 56% and the `hair` column about 27% `NA` values. The high number of `NA` values makes these variables difficult for any analysis.

The `id` variable contains about 22% `NA` values, similar to `hair`. The variables `month`, `align`, `name`, `page_id` and `publisher` contain no `NA` values at all, `alive`, `year`, `sex` and `appearances` only very few (<6%). They are well suited for further analysis.

2.4.2 Eye and Hair Colors

To visualize the eye and hair color information the data is extracted and the number of elements of each level are counted.

```
data_eye  <- train %>% group_by(eye)  %>% summarize(count = n()) %>% arrange(desc(count))
data_hair <- train %>% group_by(hair) %>% summarize(count = n()) %>% arrange(desc(count))
```

The results are shown in these tables:

```
t1 <- data_eye
t2 <- data_hair
knitr::kable(list(t1, t2),booktabs = T, linesep = "") %>%
        kable_styling(font_size=8.5, latex_options="hold_position")
```

Then the treemaps are plotted using the **`treemap`** package. [15]

eye	count
NA	8656
Blue Eyes	2125
Brown Eyes	1875
Black Eyes	629
Green Eyes	623
Red Eyes	522
White Eyes	348
Yellow Eyes	224
Grey Eyes	83
Hazel Eyes	68
Photocellular Eyes	42
Purple Eyes	30
Variable Eyes	27
Orange Eyes	26
Pink Eyes	19
Violet Eyes	18
Gold Eyes	17
One Eye	13
Silver Eyes	9
Amber Eyes	5
Yellow Eyeballs	5
Multiple Eyes	4
No Eyes	3
Black Eyeballs	2
Magenta Eyes	2
Compound Eyes	1

hair	count
NA	4172
Black Hair	3664
Brown Hair	2326
Blond Hair	1525
No Hair	763
White Hair	760
Red Hair	698
Bald	565
Grey Hair	437
Green Hair	108
Blue Hair	62
Auburn Hair	54
Strawberry Blond Hair	54
Purple Hair	53
Orange Hair	34
Pink Hair	25
Variable Hair	18
Yellow Hair	13
Silver Hair	12
Gold Hair	10
Reddish Blond Hair	5
Light Brown Hair	4
Magenta Hair	3
Orange-brown Hair	3
Reddish Brown Hair	3
Violet Hair	3
Bronze Hair	1
Platinum Blond Hair	1

```r
library(treemap)

treemap(data_eye,
        index="eye",
        vSize="count",
        type="index"
       )
```

```r
treemap(data_hair,
        index="hair",
        vSize="count",
        type="index"
       )
```

Figures 2 and 3 show the results for the `eye` and `hair` columns. There are 26 different `eye` values and 28 different `hair` values. The tables above already show that the upper third contains the biggest part of the characters in the dataset. The other values are present in only a little amount of elements.

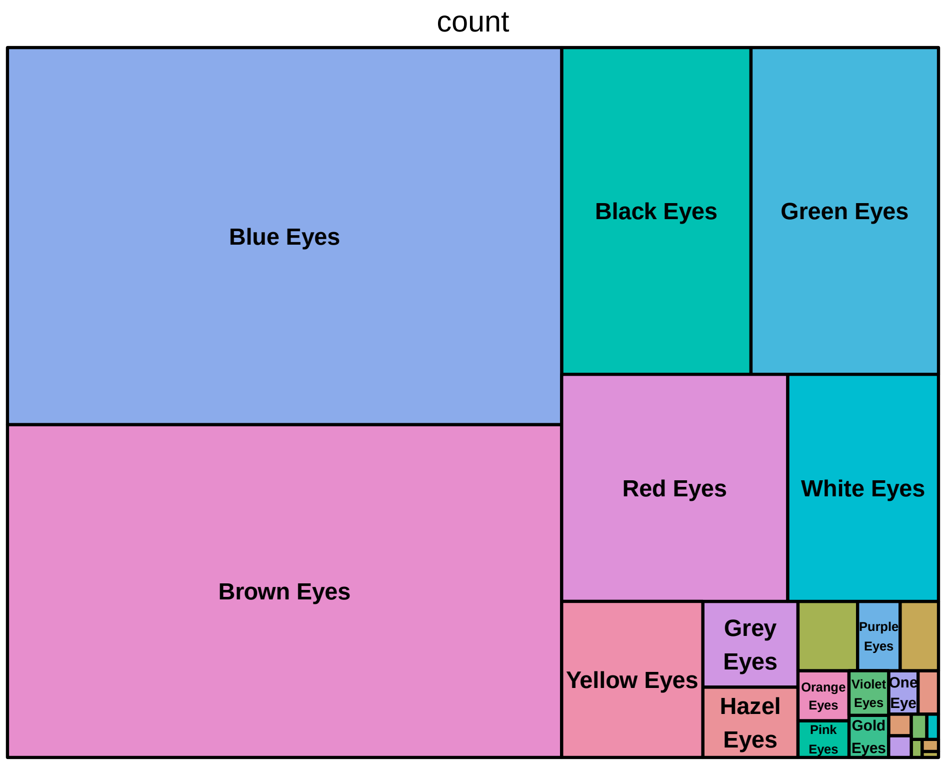

Figure 2: Treemap of eye colors (omitting the NA values)

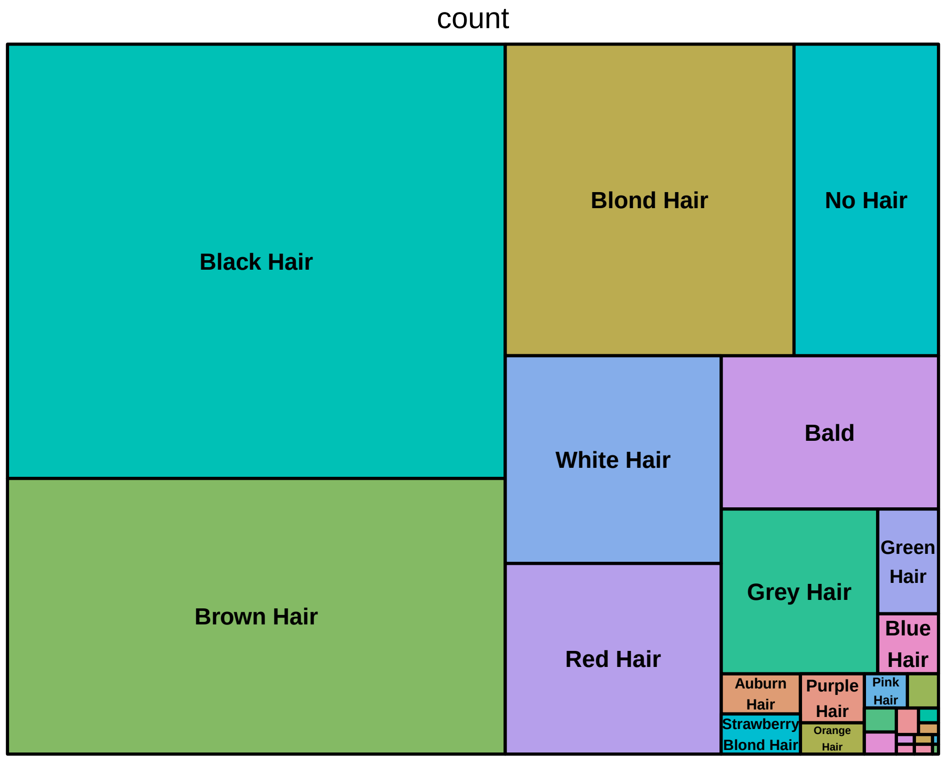

Figure 3: Treemap of hair colors (omitting the NA values)

2.4.3 Alignment vs. Sex of the Characters

In a next step the alignment of the characters depending on their sex is evaluated. The results can be seen in this table:

```
train %>% group_by(align,sex) %>%
          summarize(count = n()) %>%
          kbl(booktabs = T, linesep = "") %>%
          kable_styling(font_size=8.5, latex_options="hold_position")
```

align	sex	count
Bad Characters	Agender Characters	20
Bad Characters	Female Characters	1425
Bad Characters	Genderless Characters	10
Bad Characters	Male Characters	6792
Bad Characters	Transgender Characters	1
Bad Characters	NA	405
Reformed Criminals	Female Characters	1
Reformed Criminals	Male Characters	1
Good Characters	Agender Characters	9
Good Characters	Female Characters	2246
Good Characters	Genderfluid Characters	1
Good Characters	Genderless Characters	5
Good Characters	Male Characters	4319
Good Characters	NA	141

Figure 4 and 5 show barplots of this data. To reduce the number of graphs displayed the `Reformed Criminals` are left out. As can be seen in the table above this affects only two characters. Therefore the statistical power of this class is very low anyway.

```
train %>% filter(align!="Reformed Criminals") %>%
          ggplot(aes(x=sex, fill=align)) +
          geom_bar() +
          theme(legend.position = "bottom") +
          scale_x_discrete(labels = function(sex) str_wrap(sex, width = 10))
```

In figure 4 the absolute numbers of characters in each sex are shown. It becomes clear, that by far most of the characters are male. Female characters are in second place. Other genders are considerably more seldom.

```
train %>% filter(align!="Reformed Criminals") %>%
          ggplot(aes(x=sex, fill=align)) +
          geom_bar(position="fill") +
          theme(legend.position = "bottom") +
          scale_x_discrete(labels = function(sex) str_wrap(sex, width = 10))
```

Figure 5 shows the same data, but the bars are all scaled to 1. Now the distribution of the alignment in each sex can be evaluated. Note that as there is only one transgender and one genderfluid character in the datset those bars are uniform.

In nearly all sexes considerably more bad than good aligned characters can be found. Only female characters and the one genderfluid character are mostly aligned good. That shows that the comic characters in this dataset show a clear sexist bias. This has been been stated before in the artice “Comic Books Are Still Made By Men, For Men And About Men”published on FiveThirtyEight. [4]

Nevertheless the sex information is a good candidate for machine learning concepts.

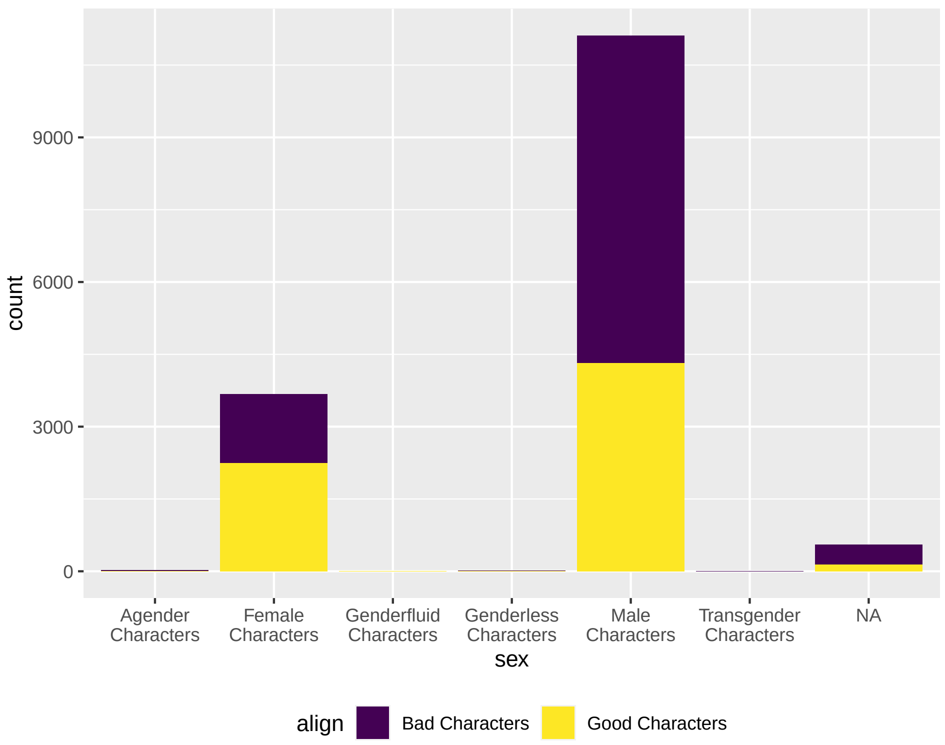

Figure 4: Barplot of sex of characters: The colors displays the proportion of alignment in this sex (omitting "Reformed Criminals" and the NA values)

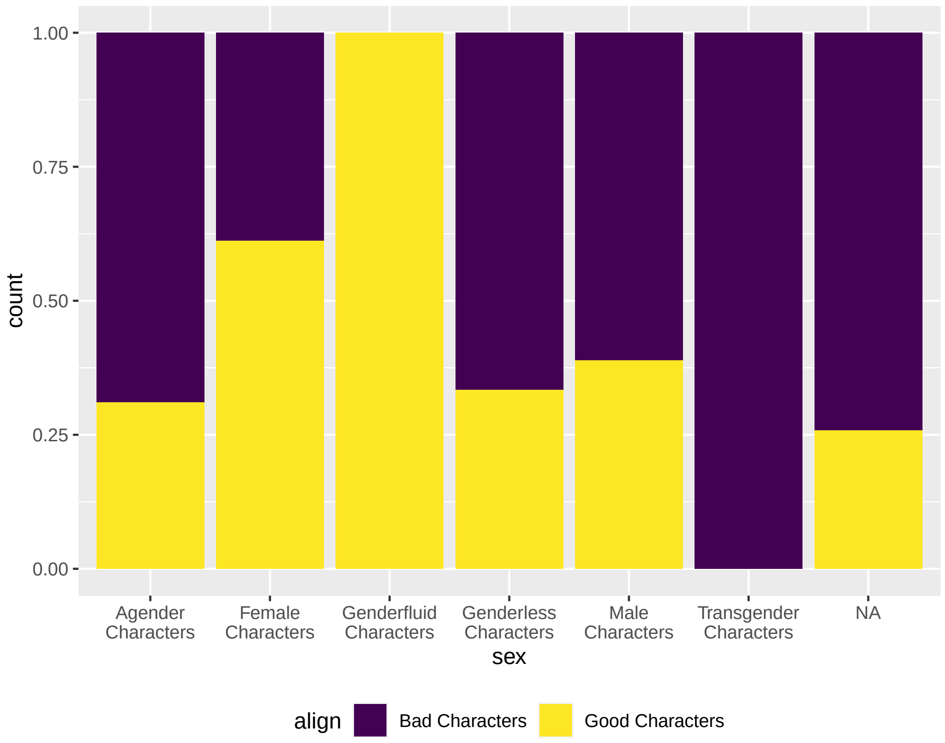

Figure 5: Stacked barplot of sex of characters: All bars are scaled to 1, the colors displays the proportion of alignment in this sex (omitting "Reformed Criminals" and the NA values)

2.4.4 Alignment vs. Eye and Hair Color Values

The relation between alignment and hair and eye colors is discussed in this chapter. The level `Reformed Criminals` is left out again as described before.

Figure 6 shows a violin plots of the distribution of the RGB values of the eye color over the alignment. The same plot for the hair color can be found in figure 7.

The RGB values respectively show similar distributions for good and bad characters Nevertheless differences can be seen. Especially in the eye color (figure 6) the middle part of the distribution of blue and green is increased for good characters. The hair color plot (figure 7) shows only little difference between good and bad characters.

```
train %>% filter(align!="Reformed Criminals") %>%
          select(align, eye_R, eye_G, eye_B) %>%
          pivot_longer(.,
                       cols = c(eye_R,eye_G,eye_B),
                       names_to = "Color",
                       values_to = "RGB_value") %>%
          ggplot( aes(x=align, y=RGB_value, fill=Color)) +
          geom_violin() +
          scale_fill_manual(values=c("blue", "green", "red")) +
          theme(legend.position = "bottom")
```

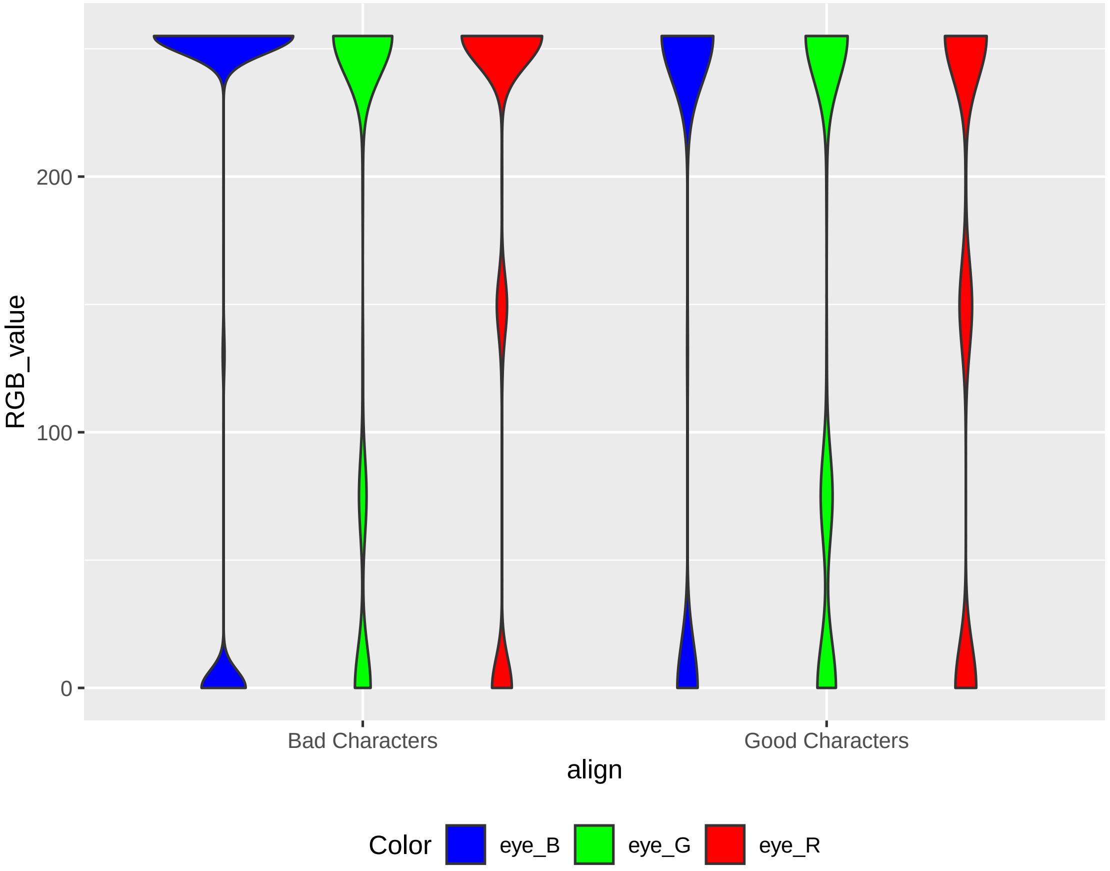

Figure 6: Violin plot of eye color RGB values over character alignment (without level "Reformed Criminals")

```
train %>% filter(align!="Reformed Criminals") %>%
          select(align, hair_R, hair_G, hair_B) %>%
          pivot_longer(.,
                       cols = c(hair_R,hair_G,hair_B),
                       names_to = "Color",
                       values_to = "RGB_value") %>%
          ggplot( aes(x=align, y=RGB_value, fill=Color)) +
          geom_violin() +
          scale_fill_manual(values=c("blue", "green", "red")) +
          theme(legend.position = "bottom")
```

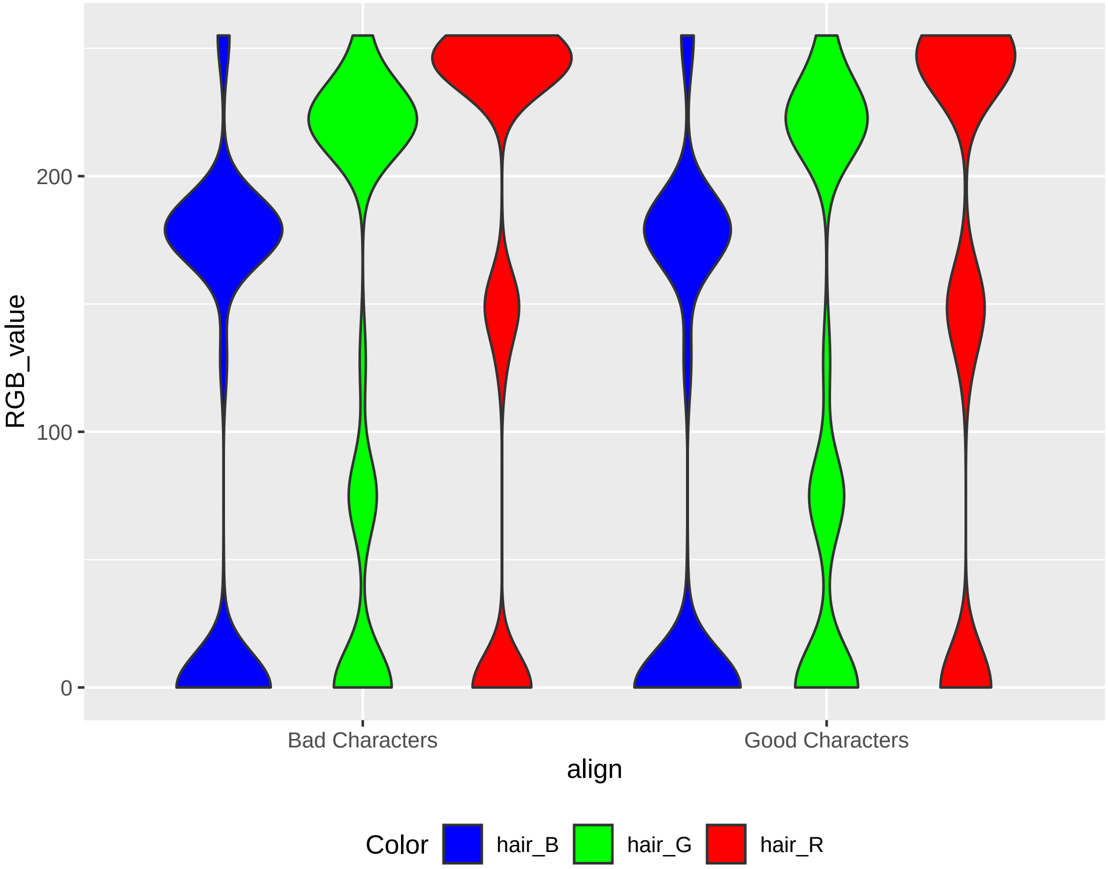

Figure 7: Violin plot of hair color RGB values over character alignment (without level "Reformed Criminals")

2.5 Association Analysis

The relationship between variables can be detected by doing an association analysis. The results can be used to determine which variables are well suited to use in machine learning algorithms.

It is important to distinguish two general types: numeric and categorical data. [16] Quantitative (numerical) data contains any discrete or continuous values, which describe some feature, eg. the year the comic was published or the number of appearances of a character. Non numeric elements are called qualitative (categorical) data. This kind of data includes any named elements or categories eg. the hair color of a character. In some cases these categories can be ordered like the alignment from good to bad.

Most of the 13 variables in the `train` dataframe are categorical (factors or character variables). Only three are numerical.

The method to analyse the association differs depending on the type of data:

- Both variables are numeric: **Correlation**
- One variable is numeric, one categorical: **ANOVA**
- Both variables are categorical: **Cramer's V**

2.5.1 Correlation

The correlation coefficient can be used to determine the relationship between two numeric variables. [17]

In R it can be calculated using the `cor` method.

```
cor(variable1,variable2)
```

The correlation coefficient is always a value between -1 and +1 with 0 meaning no correlation at all and -1 respectively 1 meaning full linear correlation.

2.5.2 ANOVA

ANOVA (analysis of variance) can be used to analyse the relationship between a nominal and a numeric variable.

The basic idea is to model the relationship between the variables and then analyse how much of the variance can be explained by the model. This can be done using a linear model and the `lm` command in R. [18] Then the coefficient of determination R^2 of the model is a measurement of association. [19]

```
r_squared = summary(lm(variable1 ~ variable2)$r.squared
```

In this code `variable1` is numeric, `variable2` is categorical

An R^2 value of 1 indicates that the model fits the data perfectly and a value of 0 that it does not fit at all. [17]

2.5.3 Cramer's V

Cramer's V is one method to determine the association of two categorical variables. It is the square root of the chi-square value χ divided by the sample size n_{sample}. [20] [17]

$$V = \frac{\chi^2}{\chi^2_{max}} = \frac{\chi^2}{n_{sample} - m_{cat}} \qquad (1)$$

m_{cat} is the smaller number of categories of both variables minus 1:

$$m_{cat} = min(n_{cat} - 1) \qquad (2)$$

The resulting value is between 0 and 1, with 1 meaning one variable is totally determined by the other.

Compared to correlation analysis the resulting values are much smaller, especially if the variables contain many categories. According to [21] for four and more categories a value larger than 0.29 already shows large association.

In R it can be calculated using the **rcompanion** package [21] and the command `cramerV`.

```
library(rcompanion)
```

```
cramerV(variable1,variable2)
```

2.5.4 Implemention of the Association Analysis

The implementation of the association analysis is based on code proposed in [22]. It got enhanced and adapted to the tasks given in this report. The method aims at automating the calculation of the association values of all possible combinations of variables in the dataframe.

First two functions are implemented to evaluate if a variable is nominal or numeric:

```
is_nominal_value <- function(val){
    is.factor(val) || is.character(val)
    }

is_numeric_value <- function(val){
    is.double(val) || is.integer(val)
    }
```

The variables `name` and `page_id` in the dataframe contain only unique information. They are not suited for association analysis. That is why they are removed from the `train` dataframe and the new subset `train_assoc` is created:

```
train_assoc <- train %>% subset(select = -c(page_id,name))
```

The `train_assoc` dataframe contains of 11 variables. All possible combinations of these variables are calculated:

```
combi <-crossing(1:ncol(train_assoc),1:ncol(train_assoc))
```

Than the column names of these variables are extracted and stored in `col_names`:

```
col_names <- names(train_assoc)
```

A function `calc_association` is built to calculate the association values of all elements in the dataframe `data` depending on the combination of data types using the methods described above:

```
calc_association <- function(pos1,pos2){
    xdata <- pull(train_assoc,pos1)
    xname <- col_names[pos1]

    ydata <- pull(train_assoc,pos2)
    yname <- col_names[pos2]

    # if both vectors are nominal calculate cramersV
    if(is_nominal_value(xdata) && is_nominal_value(ydata)){
        cv <-  cramerV(as.character(xdata), as.character(ydata))
        data.frame(xname, yname, assoc_value=cv, type="cramersV")
        }
    # if both vectors are numeric, calculate correlation
    else if(is_numeric_value(xdata) && is_numeric_value(ydata)){
        corr <- cor(xdata, ydata,use="complete.obs")
        data.frame(xname, yname, assoc_value=corr, type="correlation")
        }
    # if first vector is numeric and second nominal calculate ANOVA
    else if(is_numeric_value(xdata) && is_nominal_value(ydata)){
        r_squared = summary(lm(xdata ~ ydata))$r.squared
        data.frame(xname, yname, assoc_value=r_squared, type="anova")
        }
    # if first vector is nominal and second numeric calculate ANOVA
    else if(is_nominal_value(xdata) && is_numeric_value(ydata)){
        r_squared = summary(lm(ydata ~ xdata))$r.squared
        data.frame(xname, yname, assoc_value=r_squared, type="anova")
        }
    else{
        print("not defined")
        }
    }
```

Finally the association values of all combinations are calculated:

```
library(fBasics)
association_matrix <- map2_df(rowVec(combi[,1]), rowVec(combi[,2]), calc_association)
```

The **fBasics** package [23] is needed to create row-vectors from numeric vectors.

2.5.5 Results of Association Analysis

The results of the association analysis are plotted separately for each association type in tile diagrams. They are shown and discussed in this section. The machine learning models in chapter 2.6 aim at estimating the alignment of the characters. Thus a special interested will be taken in the association of the `align` variable with others in the dataset.

Correlation

Figure 8 shows the results of the correlation analysis. The RGB values of eye and hair color respectively are highly correlated among each other.

Year of introduction and the number of appearances of a comic character are only very little correlated with all other variables.

```
association_matrix %>%
  filter(type=="correlation") %>%
  ggplot(aes(xname,yname,fill=assoc_value))+
  geom_tile()+
  geom_text(size=3,aes(xname,yname,label=round(assoc_value,2)))+
  scale_fill_gradient(low="red", high="yellow")+
  theme(legend.position = "bottom",
        legend.text = element_text(angle = 45, vjust = 1, hjust=1),
        legend.text.align = 1,
        axis.text.x = element_text(angle = 45, vjust = 1, hjust=1)
        )
```

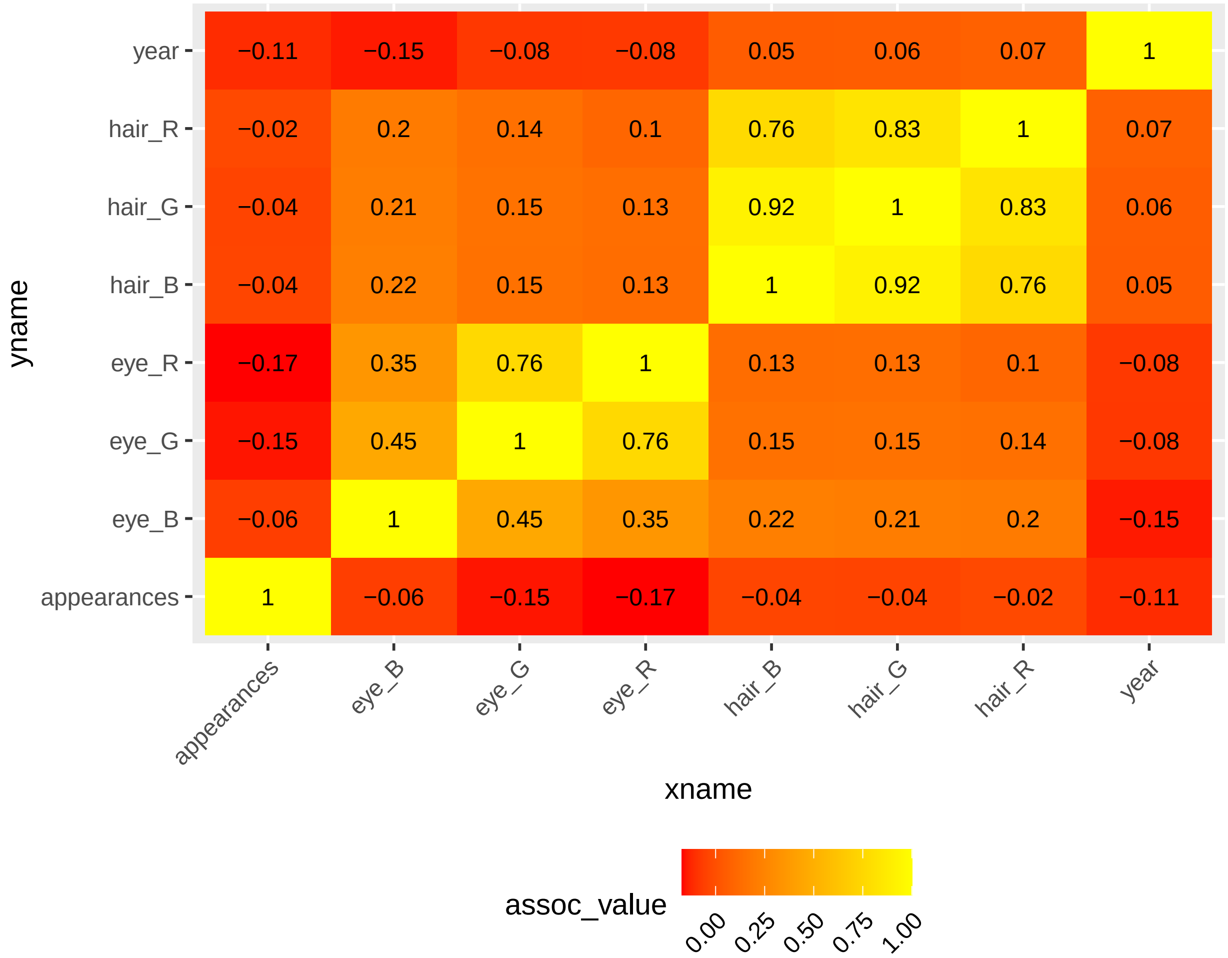

Figure 8: Results of correlation analysis of numeric variables in the train dataframe

ANOVA

The results of the ANOVA analysis can be found in figure 9. The categorical name of the hair and eye color respectively and their RGB values are perfectly correlated (value 1). The alignment (variable `align` which we will try to estimate later) shows only little association with the RGB color values and even less with all other variables.

```
association_matrix %>%
  filter(type=="anova") %>%
  ggplot(aes(xname,yname,fill=assoc_value))+
  geom_tile()+
  geom_text(size=3,aes(xname,yname,label=round(assoc_value,2)))+
  scale_fill_gradient(low="red", high="yellow")+
  theme(legend.position = "bottom",
        legend.text = element_text(angle = 45, vjust = 1, hjust=1),
        legend.text.align = 1,
        axis.text.x = element_text(angle = 45, vjust = 1, hjust=1)
        )
```

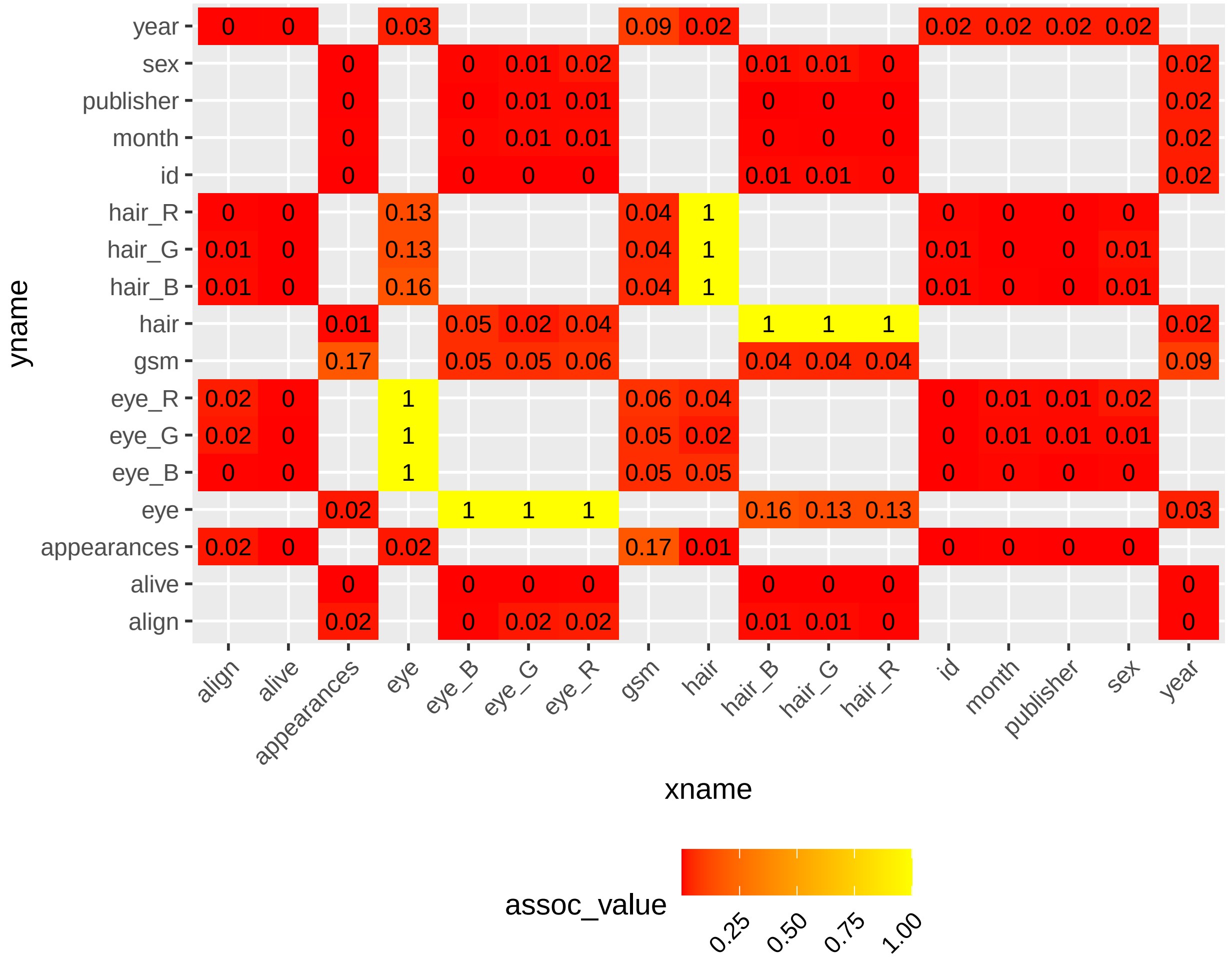

Figure 9: Results of ANOVA analysis of variables in the train dataframe

Cramer's V

Figure 10 shows the results of the Cramer's V calculations. The month of first publication and the publisher are highly associated (values close to 1). That is probably due to the individual publication cycle of the two publishers. As described before in Cramer's V lower values might already show a relevant association. That in mind an association between the alignment of the characters and the hair and eye color names seems to be relevant as well as the variables `sex` and `id`. The variables `alive` and `publisher` seem to be less associated. Nevertheless it will be worth to try these variables later.

```r
association_matrix %>%
  filter(type=="cramersV") %>%
  ggplot(aes(xname,yname,fill=assoc_value))+
  geom_tile()+
  geom_text(size=3,aes(xname,yname,label=round(assoc_value,2)))+
  scale_fill_gradient(low="red", high="yellow")+
  theme(legend.position = "bottom",
        legend.text = element_text(angle = 45, vjust = 1, hjust=1),
        legend.text.align = 1,
        axis.text.x = element_text(angle = 45, vjust = 1, hjust=1)
        )
```

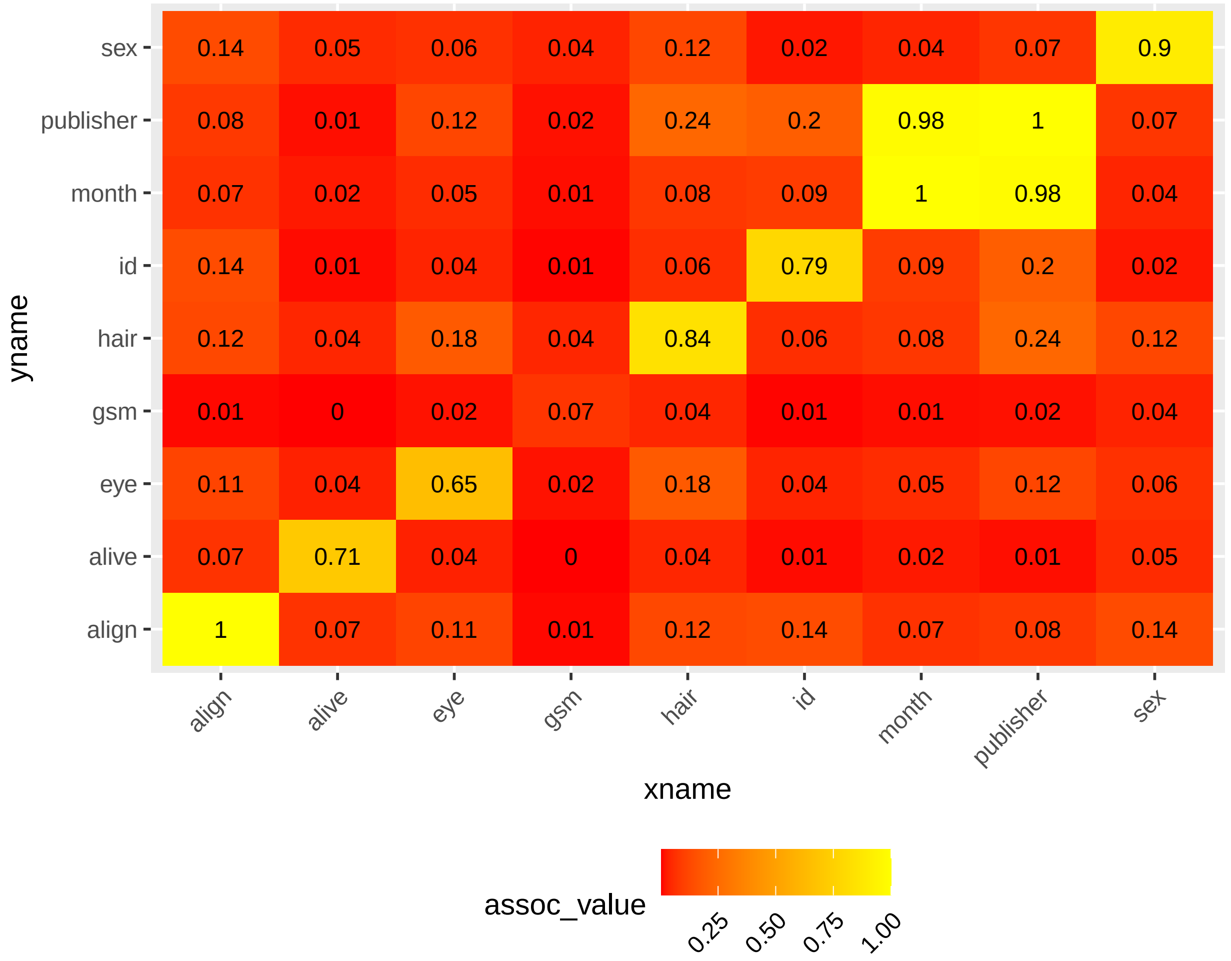

Figure 10: Results of Cramers V analysis of categorical variables in the train dataframe

2.6 Modelling Approach

In this chapter different models will be created using the `train` dataset. They will then be evaluated using the `test` dataset.

The approaches are evaluated by calculating their overall accuracy.

Besides this the time needed to carry out the fit and predict operations is analysed. The start and end time are set at the relevant parts of the code. Afterwards the time difference Δt is calculated. This code shows the method in principle:

```
timeStart <- Sys.time() # get start time of calculation
  # put here the fit and predict operations
timeEnd <- Sys.time() # get end time of calculation

timeDiff <- timeEnd - timeStart
```

2.6.1 (1) Guessing the Outcome

The most simple and baseline approach is to simply guess the outcome.

As there are only very few `Reformed Criminals` in the data, only the other alignments `Good Characters` and `Bad Characters` are used as possible guesses.

```r
set.seed(1)

timeStart <- Sys.time() # get start time of calculation

    y_hat1 <- sample(c("Bad Characters", "Good Characters"),
                     length(test_index), replace = TRUE) %>%
              ordered(levels = levels(test$align))

timeEnd   <- Sys.time() # get end time of calculation
timeDiff1 <- timeEnd - timeStart
```

Then the overall accuracy acc_1 in the test set using this approach is calculated:

```r
acc1 <- mean(y_hat1 == test$align)
```

It results to $acc_1 = 0.488$. Fit and prediction took $\Delta t_1 = 0.00248$ seconds. This is the baseline for other approaches discussed in the next chapters.

2.6.2 (2) Estimation using Numeric Data: Loess Estimation

The next approach to estimate the alignment of the characters is to use the numeric RGB values for eye and hair color. This will be done using local regression and the `gamLoess` method. [17] In R this can be achieved with the **`gam`** [24] package. Additionally the packages **`splines`** [25] and **`foreach`** [26] are needed.

```r
library(gam)
library(splines)
library(foreach)
```

The `gamloess` algorithm cannot deal with `NA` values in the data. Thus only the cleaned RGB values can be used as all `NA` values have been set manually to the median value. Other variables are not tested in this model.

2.6.2.1 (2a) Estimation using Hair and Eye RGB Values

The calculated RGB values are used to calculate a fit with `gamLoess`. Afterwards the prediction $\hat{y}_{2a}$ is calculated. The time is measured using the method described above.

```r
timeStart <- Sys.time() # get start time of calculation

train_control <- trainControl(verboseIter = FALSE)

    # Calculate Fit
    fit2a <- train(align ~ hair_R + hair_G + hair_B + eye_R + eye_G + eye_B,
                   data = train,
                   method = "gamLoess",
                   verbose = FALSE)
    # Prediction
    y_hat2a <- predict(fit2a, newdata=test) %>% factor(levels = levels(test$align))

timeEnd <- Sys.time() # get end time of calculation
timeDiff2a <- timeEnd - timeStart
```

These calculations took $\Delta t_{2a} = 26.25349$ seconds.

The overall accuracy is calculated using this code.

```r
acc2a <- mean(y_hat2a == test$align)
```

The accuracy acc_{2a} is 0.464. This is worse than guessing the outcome in model 1. Figure 11 shows the values of the alignment variable `align` in the prediction `y_hat2a` and the `test` set. The model estimated exclusively `Bad Characters` and `Reformed Criminals` but no `Good Characters`. In the `train` set as well as in the `test` set nearly no `Reformed Criminals` can be found. This error is probably due to the transformation of the estimated numeric values to ordered levels as `Reformed Criminals` is the mid level.

```
results_mod2a  <- data.frame(y_hat2a,test$align) %>% pivot_longer(everything())
results_mod2a %>% ggplot(aes(x=name,fill=value)) +
                  geom_bar() +
                  theme(legend.position = "bottom")
```

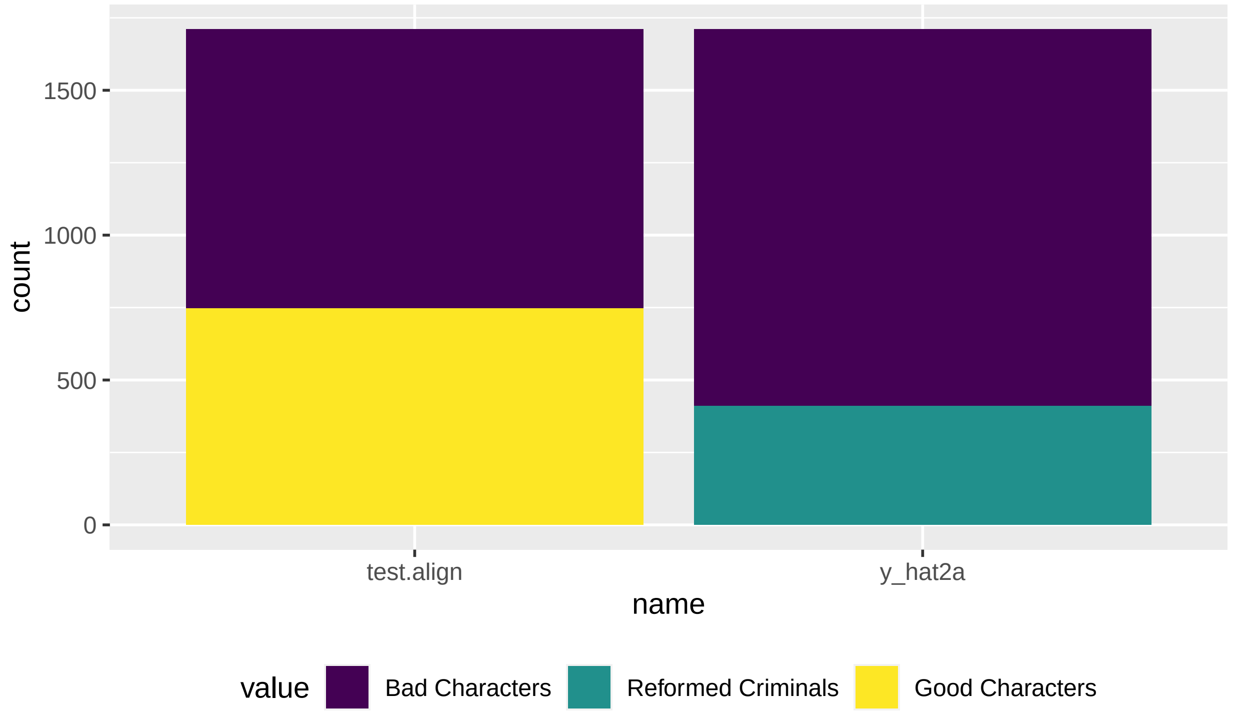

Figure 11: Distribution of alignments in prediction and the test dataset

2.6.2.2 (2b) Model without Level `Reformed Criminals`

To improve the estimation the level `Reformed Criminals` is removed from the training set.

A new training dataset `train2b` is created:

```
train2b <- train %>% filter(align!="Reformed Criminals")
train2b <- droplevels(train2b)
```

Then the fit and prediction operations are performed as described above:

```
timeStart <- Sys.time() # get start time of calculation

    # Calculate Fit
    fit2b <- train(align ~ hair_R + hair_G + hair_B + eye_R + eye_G + eye_B,
                   data = train2b,
                   method = "gamLoess")
    # Prediction
    y_hat2b <- predict(fit2b, newdata=test) %>% factor(levels = levels(test$align))
```

```
timeEnd <- Sys.time() # get end time of calculation
timeDiff2b <- timeEnd - timeStart
```

The calculations took $\Delta t_{2b} = 23.76295$ seconds

Then the accuracy of the improved model is calculated:

```
acc2b <- mean(y_hat2b == test$align)
```

It is $acc_{2b} = 0.605$ now, which is much better than before. This can be seen using figure 12 as well. Now no `Reformed Criminals` are estimated anymore.

```
results_mod2b  <- data.frame(y_hat2b,test$align) %>% pivot_longer(everything())
results_mod2b %>% ggplot(aes(x=name,fill=value)) +
                  geom_bar() +
                  theme(legend.position = "bottom")
```

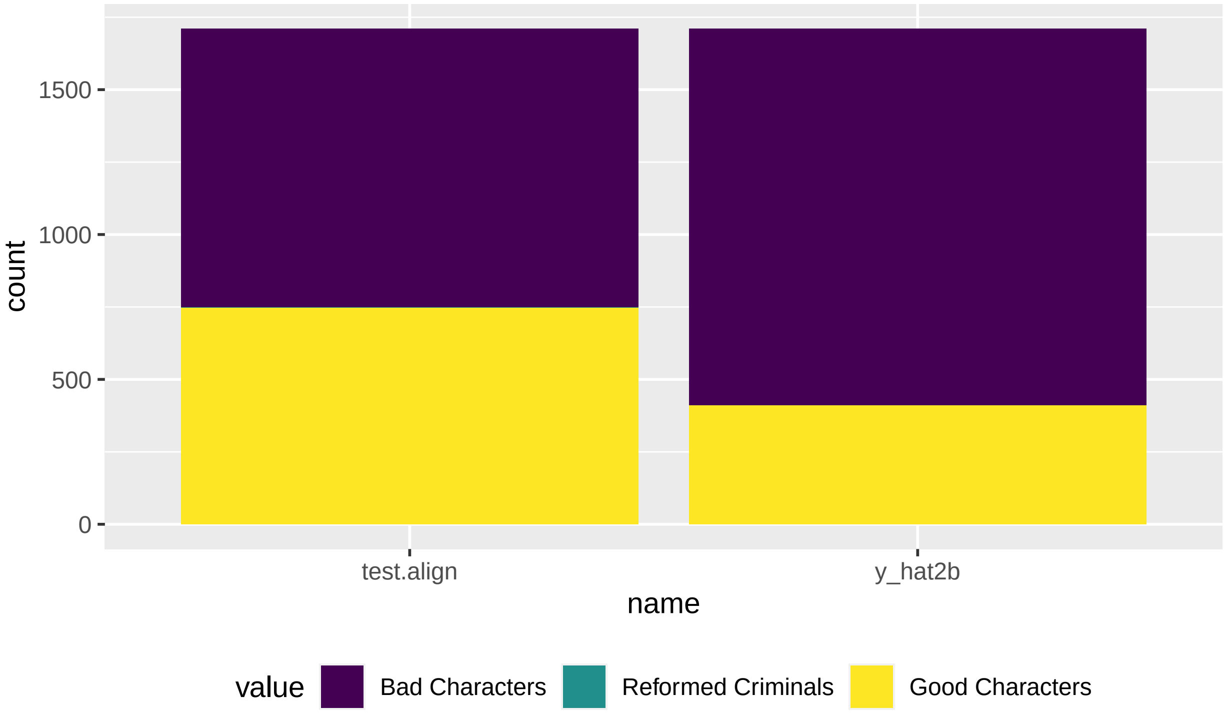

Figure 12: Distribution of alignments in estimation and test dataset in the new model 2b without "Reformed Criminals"

2.6.3 (3) Estimation using Numeric and Categorical Data: Decision Trees

As the given dataset includes mostly categorical data decision trees are an interesting approach as described in [17]. They can be calculated using the **`rpart`** package. [27] [17]

```
library(rpart)
```

In this chapter different approaches using decision trees are discussed.

2.6.3.1 (3a) Hair and Eye Information as categorical variables

First the data is fitted on the `train` dataset. Then this fit is used to predict the alignments in the `test` set

```
set.seed(1)

timeStart <- Sys.time() # get start time of calculation

  # Calculate Fit
  fit3a <- rpart(align ~ hair + eye,
                 data = train,
                 method = "class",
                 cp = 0.01)
  # Prediction
  y_hat3a <- predict(fit3a, newdata=test,type = "vector")

timeEnd <- Sys.time() # get end time of calculation
timeDiff3a <- timeEnd - timeStart
```

The fit and prediction took $\Delta t_{3a} = 55.17967$ seconds.

The resulting decision tree can be plotted using the **rpart.plot** package. [28]

```
library(rpart.plot)
prp(fit3a, type=2, extra=0)
```

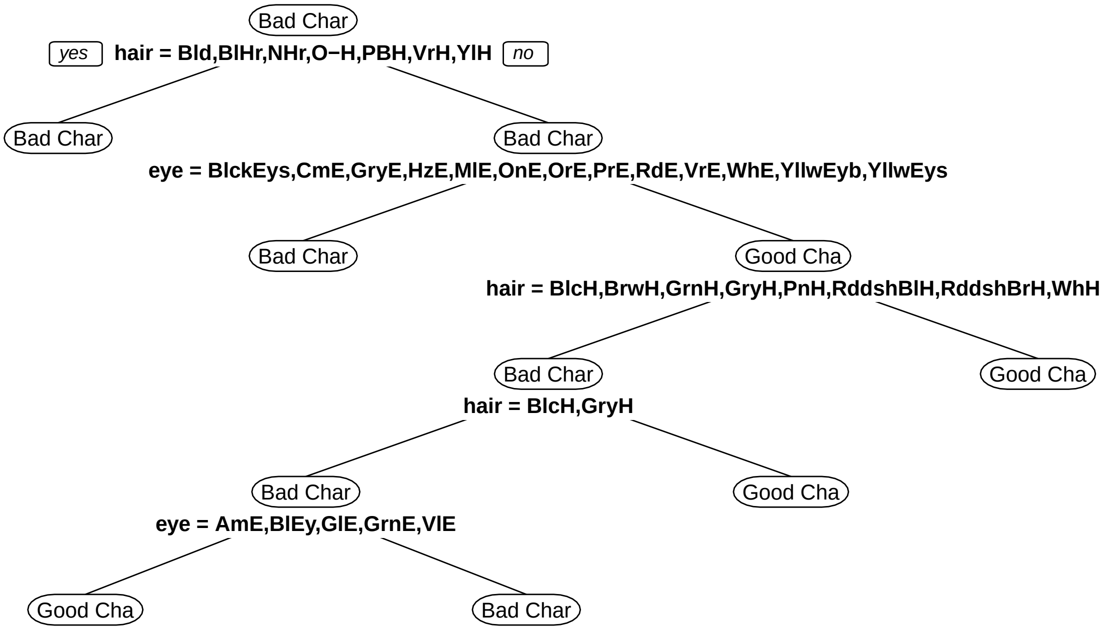
Bad Char
yes
hair = Bld,BlHr,NHr,O−H,PBH,VrH,YlH
no
Bad Char
Bad Char
eye = BlckEys,CmE,GryE,HzE,MlE,OnE,OrE,PrE,RdE,VrE,WhE,YllwEyb,YllwEys
Bad Char
Good Cha
hair = BlcH,BrwH,GrnH,GryH,PnH,RddshBlH,RddshBrH,WhH
Bad Char
Good Cha
hair = BlcH,GryH
Bad Char
Good Cha
eye = AmE,BlEy,GlE,GrnE,VlE
Good Cha
Bad Char

Figure 13: Calculated decision tree to predict the alignment of a character using the categorical eye and hair color information, the rpart package and the train dataframe

Figure 13 shows the resulting decision tree. In the plot the names of the eye and hair colors are abbreviated in a strange way and some letters are missed out. Nevertheless the original names can still be guessed. Then the overall accuracy is calculated:

```
acc3a <- mean(y_hat3a == as.numeric(test$align))
```

acc_{3a} is 0.598.

2.6.3.2 (3b) Hair and Eye Information as numeric RGB values

Now the eye and hair color information will be included using the numeric RGB values.

First the fit and predict operations are carried out as described before.

```
set.seed(1)

timeStart <- Sys.time() # get start time of calculation

  # Calculate Fit
  fit3b <- rpart(align ~ hair_R + hair_G + hair_B + eye_R + eye_G + eye_B,
                 data = train,
                 method = "class",
                 cp = 0.01)

  # Prediction
  y_hat3b <- predict(fit3b, newdata=test,type = "vector")

timeEnd <- Sys.time() # get end time of calculation
timeDiff3b <- timeEnd - timeStart
```

Afterwards the overall accuracy is calculated:

```
acc3b <- mean(y_hat3b == as.numeric(test$align))
```

The accuracy acc_{3b} is 0.604. This is only little better than model 3a with the categorical information. At the same time the calculations are performed much faster with $\Delta t_{3b} = 0.39533$ seconds.

The calculated decision tree can be seen in figure 14.

```
prp(fit3b, type=2, extra=0)
```

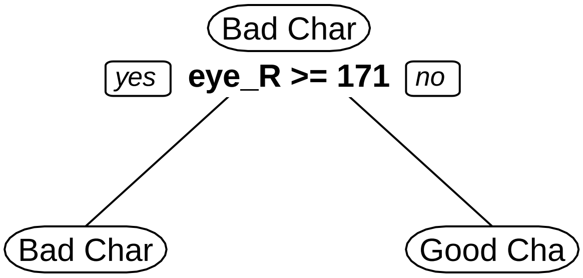

Figure 14: Calculated decision tree to predict the alignment of a character using the numeric RGB eye and hair color information, the rpart package and the train dataframe

The decisions are based on only one numeric value being higher or lower than a specific limit. The resulting tree is very simple. This can be changed by optimizing the parameter c_p as performed in model 3d later.

2.6.3.3 (3c) Adding other information from the dataframe

Now the variables `sex`, `alive`, `appearances`, `year`, `publisher` and `id` are added to the model to increase its performance. All these variables showed some association in the Cramer's V analysis in chapter 2.5.5.

Especially `sex` should include valuable information as the analysis in chapter 2.4.3 showed a clear relation of sex and alignment.

The calculations are performed as described above

```
set.seed(1)

timeStart <- Sys.time() # get start time of calculation
```

```
    # Calculate Fit
    fit3c <- rpart(align ~ hair_R + hair_G + hair_B + eye_R + eye_G + eye_B +
                  sex + alive + appearances + year + publisher + id,
                  data = train, method = "class", cp = 0.01)

    # Prediction
    y_hat3c <- predict(fit3c, newdata=test,type = "vector")

timeEnd <- Sys.time() # get end time of calculation
timeDiff3c <- timeEnd - timeStart

acc3c <- mean(y_hat3c == as.numeric(test$align))
```

The overall accuracy acc_{3c} is 0.664. The calculation took $\Delta t_{3c} = 0.63697$ seconds.

Afterwards the resulting decision tree is plotted:

```
prp(fit3c, type=2, extra=0)
```

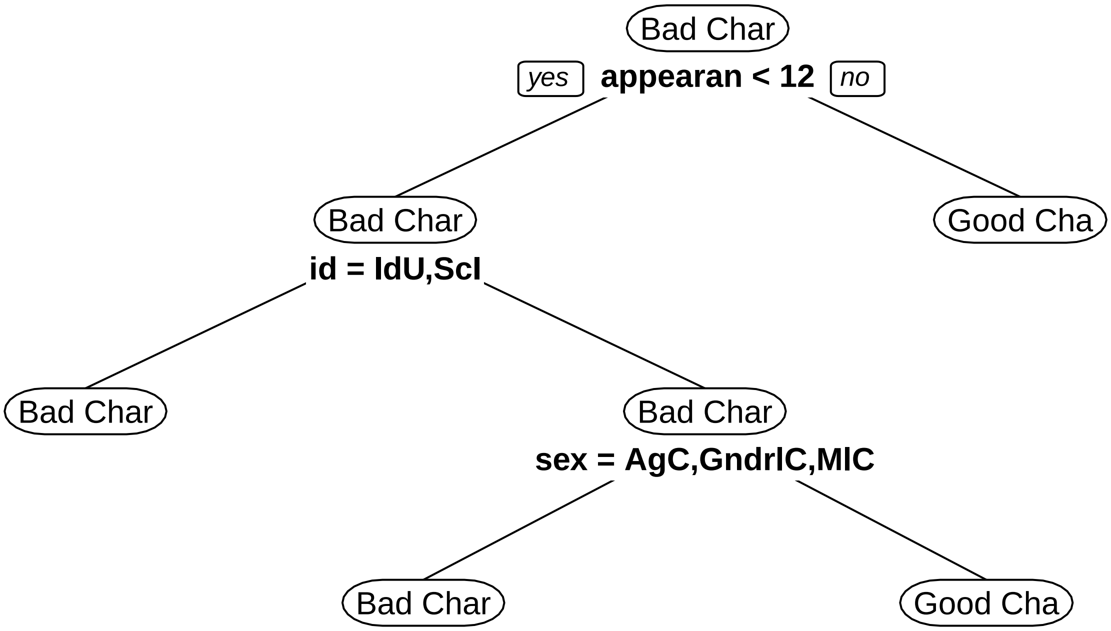

Figure 15: Calculated decision tree for model 3c

The plot shows that only little information is used, the tree has not many branches. This can be improved by optimizing the parameter c_p in the next model.

2.6.3.4 (3d) Model with optimized c_p value

The value of c_p has to be selected. This can be achieved by optimizing the given equation to reach a maximal accuracy.

Optimizing c_p parameter

The optimization is done using only the `train` dataset. It is parted again to get the new `train_from_train` and `test_from_train` datasets:

```
set.seed(1)
# create new data partitions for optimization with train dataset
```

```
test_index <- createDataPartition(train$align, times = 1, p = 0.2, list = FALSE)
test_from_train  <- train[test_index,]     # 10% of data
train_from_train <- train[-test_index,]    # 90% of data
```

Using these new datasets c_p is optimized. First vector of possible values is created:

```
cp_values =seq(0,0.015, len = 25)
```

Afterwards the accuracy for each c_p value in the vector is computed:

```
acc_values <- sapply(cp_values, function(cp_val){
  # Fit
  fit_opt <- rpart(align ~ hair_R + hair_G + hair_B + eye_R + eye_G + eye_B +
                           sex + alive + appearances + year + publisher + id,
                   data = train_from_train,
                   method = "class",
                   cp = cp_val)
  # Prediction
  y_hat_opt <- predict(fit_opt, newdata=test_from_train,type = "vector")
  # Overall Accuracy
  mean(y_hat_opt == as.numeric(test_from_train$align))
})
```

The results are given in figure 16.

```
results_cp <- data.frame(cp_values,acc_values)
results_cp %>% ggplot(aes(x=cp_values, y=acc_values)) + geom_point()
```

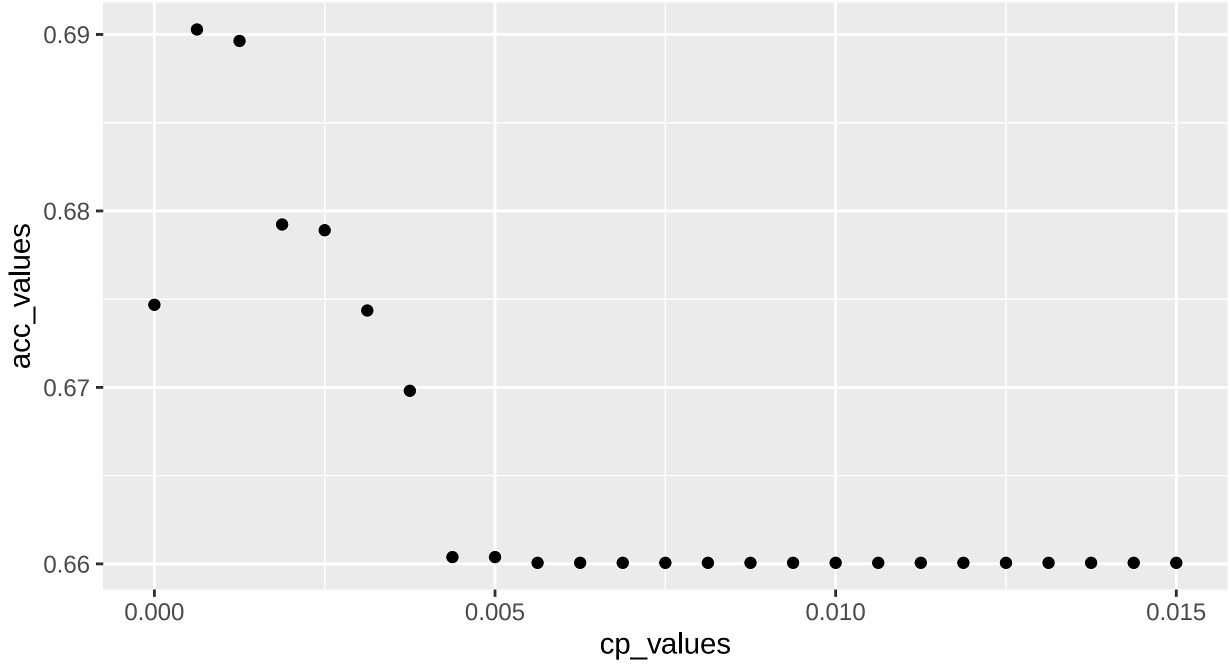

Figure 16: Results of the optimization of c_p

The optimal c_p is the value for maximal accuracy:

```
cp_opt <- cp_values[which.max(acc_values)]
```

It is 6.25×10^{-4}.

Recalculate model using optimal c_p

Now model 3c is recalculated using the optimized c_p value.

```
set.seed(1)

timeStart <- Sys.time() # get start time of calculation

    # Calculate Fit
    fit3d <- rpart(align ~ hair_R + hair_G + hair_B + eye_R + eye_G + eye_B +
                           sex + alive + appearances + year + publisher + id,
                   data = train,
                   method = "class", cp = cp_opt)

    # Prediction
    y_hat3d <- predict(fit3d, newdata=test,type = "vector")

timeEnd <- Sys.time() # get end time of calculation
timeDiff3d <- timeEnd - timeStart

# Overall Accuracy
acc3d <- mean(y_hat3d == as.numeric(test$align))
```

The overall accuracy acc_{3d} is 0.702. It got increased by 0.038 compared to the calculation without optimization. The calculation took $\Delta t_{3d} = 1.03282$ seconds, which is only 0.39585 seconds slower than before.

The resulting decision tree is shown in figure 17

```
prp(fit3d, type=2, extra=0)
```

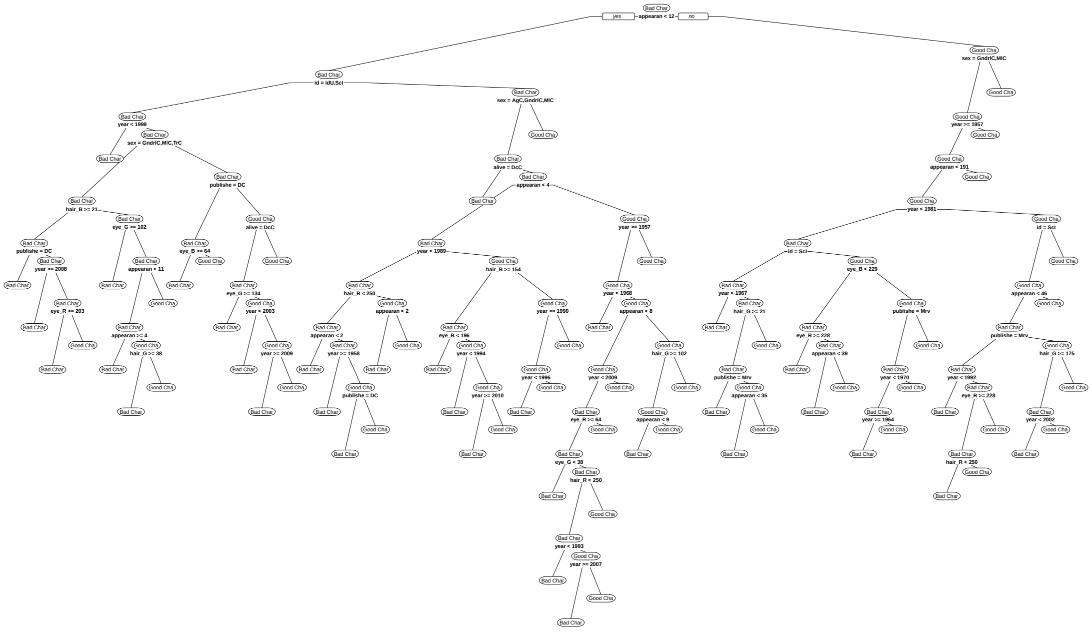

Figure 17: Calculated decision tree for model 4 with optimized c_p

The decision tree got much more complex and more information in the data got used. This leads to better results while increasing the computation time.

3 Results

This chapter presents the results of the different machine learning concepts which were developed in chapter 2.6. For each model the overall accuracy acc on the `test` set as well as the computation time Δt are shown in table 1.

The first approach only guesses the outcome as a baseline estimate (model1). It has an accuracy acc_1 of 0.488. The prediction is computed in $\Delta t_1 = 0.00248$ seconds and very fast.

Then a local regression (loess) estimation was performed. In model 2a many levels are estimated wrong as `Reformed Criminals` and the resulting accuracy acc_{2a} of 0.464 is even below the one achieved by guessing the outcome. At the same time it is a lot slower with $\Delta t_{2a} = 26.25349$ seconds. A much better accuracy of $acc_{2b} = 0.605$ can be reached with a similar loess estimation but after the level `Reformed Criminals` has been removed from the `train` dataset in model 2b. This approach is a little slower with $\Delta t_{2b} = 23.76295$ seconds.

To include categorical variables the next approach is performed using decision trees. In model 3a the eye and hair color names are included as categorical data. This leads to an accuracy acc_{3a} of 0.598 which is much better than guessing. The computation takes $\Delta t_{3a} = 55.17967$ seconds. The next model uses the RGB values of eye and hair color instead of the color names. The computation time is reduced noteworthy by 54.78434 seconds to $\Delta t_{3b} = 0.39533$ seconds. The accuracy improves little to $acc_{3b} = 0.604$. Better results can be achieved in model 3c by adding other (categorical) information. This results in an accuracy acc_{3c} of 0.664. The computation time is $\Delta t_{3c} = 0.63697$ seconds and a little higher than in model 3b but still a lot faster than using categorical hair and eye color names. The best results can be achieved using this approach and an optimized parameter c_p. Then the accuracy acc_{3d} is 0.702 and the computation time Δt_{3d} is 1.03282 seconds.

Nevertheless all reached accuracies are quite low. This is probably due to the nature of the dataset. It includes mostly categorical data, which can't be used in many modelling approaches. At the same time important variables include more than 20% (and up to 99%) `NA` values. This incomplete information in the dataset reduces the prediction quality.

Table 1: Results of different modelling approaches

No.	Model	Used Data	Accuracy	Time in Seconds
1	Guessing the Outcome	none	0.488	0.00248
2a	Loess-Estimation: RGB color values	RGB colors (eye and hair)	0.464	26.25349
2b	Loess-Estimation: Without "Reformed Criminals"	RGB colors (eye and hair)	0.605	23.76295
3a	Decision Tree: Categorical color values	Categorical color names (eye and hair)	0.598	55.17967
3b	Decision Tree: RGB color values	RGB colors (eye and hair)	0.604	0.39533
3c	Decision Tree: Other information added	RGB colors (eye and hair), sex, alive, appearances, year, publisher,id	0.664	0.63697
3d	Decision Tree: Optimized c_p	RGB colors (eye and hair), sex, alive, appearances, year, publisher,id	0.702	1.03282

4 Conclusion

In this report the Comic-Characters-Dataset was analysed and machine learning algorithms were used to estimate the alignment of a character using other available information.

First the dataset was optimized so that the hair and eye color are available as numerical RGB values. Data exploration shows that for many characters not all information is included. This results in a lot of `NA` values in the dataframe.

An association analysis is performed to analyse which variables are well suited for machine learning. It becomes clear that many variables are relevant candidates.

Three different machine learning concepts are tested in seven models to estimate the alignment of the characters. First the outcome is guessed, than a local regression method is tested (loess) using the RGB color information. The model has problems dealing with the `NA` values in the dataset. Thus only the cleaned RGB values can be used and no other variables.

Finally decision trees are computed to use categorical as well as numerical inputs. It can be shown that a significant reduction of computation time can be achieved, if the colors are included as RGB values. Best results can be achieved using a decision tree with RGB colors (eye and hair), `sex`, `alive`, `appearances`, `year`, `publisher` and `id` as input variables. Nevertheless only an overall accuracy of 0.702 can be reached

This quite low value can be explained by the difficult nature of the dataset. It contains mostly categorical data which is more difficult to deal with. At the same time important variables contain more than 20% `NA` values. This makes every approach very difficult.

Nevertheless the analysis in this work shows clearly that it is possible to predict if a character is “good” or “bad” solely based on optical appearance (like hair and eye color) or the gender. These attributes are not objectively logically connected. It becomes clear that comic books have e clear bias in terms of alignment of a character

In future work it would be interesting to test imputation methods to reduce the number of `NA` values in the dataset. This could lead to better results. At the same time other methods to train machine learning models should be tested.

This work focused on the bias of comic characters regarding their alignment. It would be interesting to analyse other possible biases in comic characters as well, eg. their chance to survive.

5 Bibliography

[1] "comic_characters function (RDocumentation)." https://www.rdocumentation.org/packages/fivethirtyeight/versions/0.5.0/topics/comic_characters.

[2] "Hardvard University, Data Science: Capstone." https://www.edx.org/course/data-science-capstone.

[3] R Core Team, *R: A language and environment for statistical computing.* Vienna, Austria: R Foundation for Statistical Computing, 2021. Available: https://www.R-project.org/

[4] W. Hickey, "Comic books are still made by men, for men and about men," 2014, Available: https://fivethirtyeight.com/features/women-in-comic-books/

[5] Fandom Wiki, "DC Comics Database." https://dc.fandom.com/wiki/DC_Comics_Database.

[6] Fandom Wiki, "Marvel Database." https://marvel.fandom.com/wiki/Marvel_Database.

[7] H. Zhu, *kableExtra: Construct complex table with 'kable' and pipe syntax.* 2021. Available: https://CRAN.R-project.org/package=kableExtra

[8] Wikipedia, "List of colors by shade." https://en.wikipedia.org/wiki/List_of_colors_by_shade.

[9] color_name, "Find Your Color." https://www.color-name.com/.

[10] H. Wickham *et al.*, "Welcome to the tidyverse," *Journal of Open Source Software*, vol. 4, no. 43, p. 1686, 2019, doi: 10.21105/joss.01686.

[11] Kuhn and Max, "Building predictive models in r using the caret package," *Journal of Statistical Software*, vol. 28, no. 5, pp. 1–26, 2008, doi: 10.18637/jss.v028.i05.

[12] Yan Holtz, "The R Graph Gallery." https://r-graph-gallery.com/.

[13] H. Wickham, *ggplot2: Elegant graphics for data analysis.* Springer-Verlag New York, 2016. Available: http://ggplot2.org

[14] M. Nakazawa, *Fmsb: Functions for medical statistics book with some demographic data.* 2023. Available: https://CRAN.R-project.org/package=fmsb

[15] M. Tennekes, *Treemap: Treemap visualization.* 2023. Available: https://CRAN.R-project.org/package=treemap

[16] The University of Adelaide - Study Online, "Types of Data in Statistics: Numerical vs Categorical Data." https://online.adelaide.edu.au/blog/types-of-data.

[17] R. A. Irizarry, *Introduction to data science: Data analysis and prediction algorithms with r.* CRC Press, 2019. Available: http://ggplot2.org

[18] Stackoverflow, "Correlation between a nominal (IV) and a continuous (DV) variable." https://stats.stackexchange.com/questions/119835/correlation-between-a-nominal-iv-and-a-continuous-dv-variable/124618#124618.

[19] D. Navarro, *ANOVA as a linear model.* LibreTexts STATISTICS, 2022. Available: https://stats.libretexts.org/Bookshelves/Applied_Statistics/Learning_Statistics_with_R_-_A_tutorial_for_Psychology_Students_and_other_Beginners_(Navarro)/16%3A_Factorial_ANOVA/16.06%3A_ANOVA_As_a_Linear_Model

[20] A. C. Acock and G. R. Stavig, "A Measure of Association for Nonparametric Statistics," *Social Forces*, vol. 57, no. 4, pp. 1381–1386, Jun. 1979, doi: 10.1093/sf/57.4.1381.

[21] S. S. Mangiafico, *Summary and analysis of extension program evaluation in r, version 1.20.05.* revised 2023. Available: rcompanion.org/documents/RHandbookProgramEvaluation.pdf

[22] Stackoverflow, "Plot the equivalent of correlation matrix for factors (categorical data)? And mixed types?" https://stackoverflow.com/questions/52554336/plot-the-equivalent-of-correlation-matrix-for-factors-categorical-data-and-mi.

[23] D. Wuertz, T. Setz, and Y. Chalabi, *fBasics: Rmetrics - markets and basic statistics*. 2023. Available: https://CRAN.R-project.org/package=fBasics

[24] T. Hastie, *Gam: Generalized additive models*. 2023. Available: https://CRAN.R-project.org/package=gam

[25] R Core Team, *R: A language and environment for statistical computing*. Vienna, Austria: R Foundation for Statistical Computing, 2023. Available: https://www.R-project.org/

[26] Microsoft and S. Weston, *Foreach: Provides foreach looping construct*. 2022. Available: https://CRAN.R-project.org/package=foreach

[27] T. Therneau and B. Atkinson, *Rpart: Recursive partitioning and regression trees*. 2022. Available: https://CRAN.R-project.org/package=rpart

[28] S. Milborrow, *Rpart.plot: Plot 'rpart' models: An enhanced version of 'plot.rpart'*. 2022. Available: https://CRAN.R-project.org/package=rpart.plot