Fragrance Recommendation System Project

Ahmad Abu Obaid

2021-12-21

1. Introduction

This capstone project (Choose Your Own!) is a part of the course HarvardX:PH125.9x Data Science: Capstone, in the HarvardX Professional Certificate Data Science Program, to presents my skills in "Data Science" and demonstrates the capacities of R and its associated statistical routines in the management and analysis of Big Data.

The aim of this project is create a Fragrance Recommendation System to train a machine learning algorithm using the inputs in one subset from Kaggle which is " Fragrances - Notes and User Rating dataset " to predict Fragrances ratings (from 0 to 5 stars) in the validation set, the column labeled rating_score is the variable to be predicted.

Within the data there are relationships which are understandable, several methods has been used with a root mean square error (RMSE), and results have been compared to get maximum possible accuracy in this prediction.

2. Setup the Environment

First, I set up the working environment by installing essential packages and libraries.

```
##Installing Packages

# Note: this process could take a couple of minutes
if(!require(tidyverse)) install.packages("tidyverse", repos = "http://cran.us.r-project.org")
if(!require(caret)) install.packages("caret", repos = "http://cran.us.r-project.org")
if(!require(data.table)) install.packages("data.table", repos = "http://cran.us.r-project.org")
if(!require(lubridate)) install.packages("lubridate", repos = "http://cran.us.r-project.org")
if(!require(stringr)) install.packages("stringr", repos = "http://cran.us.r-project.org")
if(!require(ggplot2)) install.packages("ggplot2", repos = "http://cran.us.r-project.org")
if(!require(knitr)) install.packages("knitr", repos = "http://cran.us.r-project.org")
if(!require(kableExtra)) install.packages("kableExtra", repos = "http://cran.us.r-project.org")
if(!require(scales)) install.packages("scales", repos = "http://cran.us.r-project.org")

## Loading library
library(tidyverse)
library(caret)
library(data.table)
library(lubridate)
library(stringr)
library(ggplot2)
```

```
library(knitr)
library(kableExtra)
library(scales)
```

3. Dataset

I chose this dataset because I am interested in Perfumes, i have a lot of them on my shelves.

Information of the Fragrances - Notes and User Rating dataset can be found on Kaggle: - https://www.kaggle.com/sagikeren88/fragrances-and-perfumes

The dataset can be downloaded directly from my Github repository: - https://github.com/ahmadabuobaid/HarvardX-PH125.9x-Data-Science-Capstone/raw/main/archive.zip

I use the following code to download the dataset.

```
# Downloading the Dataset
# Note: this process could take a couple of minutes
db <- tempfile()
download.file("https://github.com/ahmadabuobaid/HarvardX-PH125.9x-Data-Science-Capstone/raw/main/archive
unzip(db)

# Loading the Dataset
perfume_tb <- read.csv("perfume.csv")

# here is the structure of Dataset
str(perfume_tb)
```

```
## 'data.frame':    51212 obs. of  86 variables:
##  $ brand                  : chr  "The-Spirit-of-Dubai" "Ajmal" "Al-Jazeera-Perfumes" "Art-of-Scent-
##  $ title                  : chr  "Aamal The Spirit of Dubai for women and men" "Aatifa Ajmal for wo
##  $ date                   : int  2017 2014 0 2010 0 2015 0 2016 2015 2015 ...
##  $ accords                : chr  "woody,earthy,animalic,amber,musky,balsamic" "fresh spicy,woody,mu
##  $ rating_score           : num  5 4.2 0 0 0 4.17 4.75 3.83 4.1 3.71 ...
##  $ votes                  : int  3 10 0 1 2 10 4 14 22 18 ...
##  $ longevity_poor         : int  0 1 0 0 0 0 1 0 0 0 ...
##  $ longevity_weak         : int  0 0 0 0 0 0 0 1 2 0 ...
##  $ longevity_moderate     : int  0 0 0 0 0 1 0 1 1 2 ...
##  $ longevity_long         : int  0 0 0 0 0 2 2 3 4 1 ...
##  $ longevity_very_long    : int  3 5 0 0 2 1 0 0 1 0 ...
##  $ sillage_soft           : int  0 0 0 0 0 2 0 1 0 0 ...
##  $ sillage_moderate       : int  1 0 0 0 2 2 1 4 7 4 ...
##  $ sillage_heavy          : int  0 3 0 0 0 1 3 3 2 2 ...
##  $ sillage_enormous       : int  3 3 0 0 0 4 1 0 1 1 ...
##  $ clslove                : int  100 100 1 1 1 100 100 66 100 83 ...
##  $ clslike                : int  1 80 1 1 1 25 33 100 54 100 ...
##  $ clsdislike             : int  1 20 1 1 1 25 1 33 27 50 ...
##  $ clswinter              : int  50 100 1 1 100 25 100 66 54 50 ...
##  $ clsspring              : int  50 60 1 100 100 50 1 83 18 33 ...
##  $ clssummer              : int  50 60 1 100 100 50 1 66 18 33 ...
##  $ clsautumn              : int  50 60 1 100 100 50 100 66 72 1 ...
##  $ clscold                : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ clshot                 : int  0 0 0 0 0 0 0 0 0 0 ...
```

```
##  $ clsday                  : int  50 80 1 100 100 25 100 83 36 100 ...
##  $ clsnight                : int  50 80 1 100 100 75 1 50 72 33 ...
##  $ clslove_female25under   : int  1 1 1 1 1 25 1 1 1 1 ...
##  $ clslove_male25under     : int  1 1 1 1 1 25 1 16 1 1 ...
##  $ clslove_female25older   : int  1 60 1 1 1 25 33 1 72 83 ...
##  $ clslove_male25older     : int  100 40 1 1 1 25 66 50 27 1 ...
##  $ clslike_female25under   : int  1 1 1 1 1 1 1 16 1 16 ...
##  $ clslike_male25under     : int  1 1 1 1 1 1 33 1 9 1 ...
##  $ clslike_female25older   : int  1 80 1 1 1 25 1 66 18 50 ...
##  $ clslike_male25older     : int  1 1 1 1 1 1 1 16 27 33 ...
##  $ clsdislike_female25under: int  1 1 1 1 1 1 1 1 9 16 ...
##  $ clsdislike_male25under  : int  1 1 1 1 1 1 1 16 1 1 ...
##  $ clsdislike_female25older: int  1 20 1 1 1 25 1 1 18 1 ...
##  $ clsdislike_male25older  : int  1 1 1 1 1 1 1 16 1 50 ...
##  $ clswinter_female25under : int  1 1 1 1 1 1 1 16 9 1 ...
##  $ clswinter_male25under   : int  100 1 1 1 100 75 100 16 1 16 ...
##  $ clswinter_female25older : int  1 80 1 1 1 1 33 16 27 16 ...
##  $ clswinter_male25older   : int  1 20 1 1 1 25 66 16 18 33 ...
##  $ clsspring_female25under : int  1 1 1 1 1 1 1 16 1 1 ...
##  $ clsspring_male25under   : int  100 1 1 1 100 75 100 16 1 16 ...
##  $ clsspring_female25older : int  1 40 1 1 1 1 1 16 9 33 ...
##  $ clsspring_male25older   : int  1 20 1 100 1 50 1 33 9 1 ...
##  $ clssummer_female25under : int  1 1 1 1 1 1 1 16 1 1 ...
##  $ clssummer_male25under   : int  100 1 1 1 100 75 100 16 1 16 ...
##  $ clssummer_female25older : int  1 60 1 1 1 1 1 16 9 33 ...
##  $ clssummer_male25older   : int  1 1 1 100 1 50 1 16 9 1 ...
##  $ clsautumn_female25under : int  1 1 1 1 1 1 1 1 16 9 1 ...
##  $ clsautumn_male25under   : int  100 1 1 1 100 75 100 16 1 16 ...
##  $ clsautumn_female25older : int  1 60 1 1 1 1 33 16 36 1 ...
##  $ clsautumn_male25older   : int  1 1 1 100 1 50 66 16 27 1 ...
##  $ clsday_female25under    : int  1 1 1 1 1 1 1 16 1 16 ...
##  $ clsday_male25under      : int  100 1 1 1 100 75 100 16 1 16 ...
##  $ clsday_female25older    : int  1 60 1 1 1 1 33 16 27 50 ...
##  $ clsday_male25older      : int  1 20 1 100 1 25 66 33 9 33 ...
##  $ clsnight_female25under  : int  1 1 1 1 1 25 1 1 9 1 ...
##  $ clsnight_male25under    : int  100 1 1 1 100 75 100 16 1 16 ...
##  $ clsnight_female25older  : int  1 60 1 1 1 1 1 1 27 33 ...
##  $ clsnight_male25older    : int  1 20 1 100 1 50 1 33 36 1 ...
##  $ Ihaveit                 : num  3 8 0 0 0 10 2 4 19 10 ...
##  $ Ihadit                  : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ Iwantit                 : num  0 0 3 1 2 0 0 0 0 0 ...
##  $ notes_1                 : chr  "Top1Bulgarian Rose" "Top1Rose" "Top1Rose" "Top1Green Tea" ...
##  $ notes_2                 : chr  "Top2Bergamot" "Top2Cumin" "Top2Sandalwood" "Top2White Flowers" ..
##  $ notes_3                 : chr  "Top3Fruits" "Middle0Amber" "Top3Apple" "Top3Ozonic notes" ...
##  $ notes_4                 : chr  "Top4Agarwood (Oud)" "Middle1Woody Notes" "Top4Agarwood (Oud)" "Mi
##  $ notes_5                 : chr  "Middle0Sandalwood" "Base0Musk" "Middle0nan" "Base0nan" ...
##  $ notes_6                 : chr  "Middle1Agarwood (Oud)" "Base1Amber" "Base0nan" "" ...
##  $ notes_7                 : chr  "Middle2Cypriol Oil or Nagarmotha" "" "" "" ...
##  $ notes_8                 : chr  "Middle3Benzoin" "" "" "" ...
##  $ notes_9                 : chr  "Base0Amber" "" "" "" ...
##  $ notes_10                : chr  "Base1Castoreum" "" "" "" ...
##  $ notes_11                : chr  "Base2Civet" "" "" "" ...
##  $ notes_12                : chr  "Base3Moss" "" "" "" ...
##  $ notes_13                : chr  "Base4Agarwood (Oud)" "" "" "" ...
```

```
##  $ notes_14                : chr  "Base5Indian Oud" "" "" "" ...
##  $ notes_15                : chr  "" "" "" "" ...
##  $ notes_16                : chr  "" "" "" "" ...
##  $ notes_17                : chr  "" "" "" "" ...
##  $ notes_18                : chr  "" "" "" "" ...
##  $ notes_19                : chr  "" "" "" "" ...
##  $ notes_20                : chr  "" "" "" "" ...
##  $ gender                  : chr  "women" "women" "women" "women" ...
```

4. Data Cleaning

Dataset have 86 Columns, we want make a subset from it with only 11 Columns to works on it in this project, and cleaning these columns to be sure there is no NA or non-consistent data.

```
# selecting the specific useful columns for analysis
trimmed_perfume_tb <- perfume_tb %>% select(brand,title,date,accords,rating_score,votes,Ihaveit,Ihadit,

# replace empty cells and spaces cell with NA
trimmed_perfume_tb[trimmed_perfume_tb == "" | trimmed_perfume_tb == " "] <- NA

# check null values in dataset
map_df(trimmed_perfume_tb, ~sum(is.na(.x)))
```

```
## # A tibble: 1 x 10
##   brand title  date accords rating_score votes Ihaveit Ihadit Iwantit gender
##   <int> <int> <int>   <int>        <int> <int>   <int>  <int>   <int>  <int>
## 1     0     0     0     944            0     0       0      0       0  10479
```

```
# Number of records in the dataset
count(trimmed_perfume_tb)
```

```
##       n
## 1 51212
```

```
# Dataset have NA values and it is not clean and some of data is not consistent.
trimmed_perfume_tb$accords[is.na(trimmed_perfume_tb$accords)] <- "NotDefined"
trimmed_perfume_tb$gender[is.na(trimmed_perfume_tb$gender)] <- "NotDefined"
```

There are 51212 records in the original downloaded dataset.

As the first fragrance and personal care product company was in America and founded in 1752, date column is not have a valid values. So we need to remove any rows that have date < 1752 or date > 2021, which is mean that we will drop 12457 rows.

```
# removing any rows that have date < 1752 or date > 2021.
trimmed_perfume_tb$date[trimmed_perfume_tb$date < 1752 | trimmed_perfume_tb$date > 2021] <- NA

# of rows that have NA value
sum(is.na(trimmed_perfume_tb$date))
```

```
## [1] 12457
```

```r
# Remove NA rows
trimmed_perfume_tb <- trimmed_perfume_tb %>% drop_na()

# Number of records in the trimmed dataset
count(trimmed_perfume_tb)
```

```
##       n
## 1 38755
```

After we drop NA rows the dataset now have 38755 rows.

Now we want to show how much unique values we have in each column.

```r
# after we drop NA rows the dataset now have 38755 rows
# now we want to show how much unique values we have in each column
unique_count <- trimmed_perfume_tb %>% summarize(n_brand= n_distinct(brand), n_title=n_distinct(title),

unique_count
```

```
##   n_brand n_title n_date n_accords n_rating_score n_votes n_Ihaveit n_Ihadit
## 1    3203   38671    217     35859            304    1566      1345       16
##   n_Iwantit n_gender
## 1        40        4
```

After investigate duplicates columns, we found the same fragrance title with different date and accords and voting, so we can deal with it as a new (different) fragrance, to do so we need to add a unique id for each fragrance row.

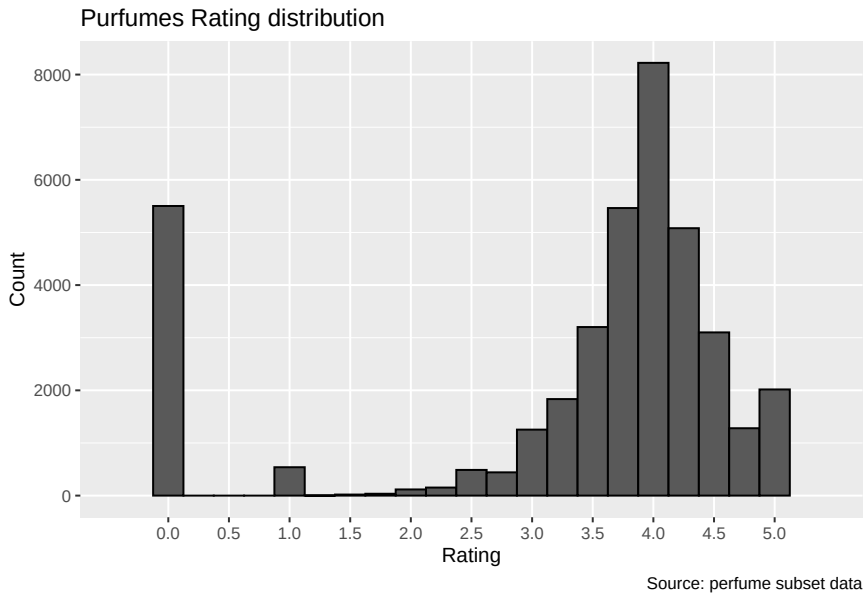
```r
# to make the fragrance unique we need to number the rows
trimmed_perfume_tb <- trimmed_perfume_tb %>% mutate(id = row_number())

# to re allocate id column to be first column
trimmed_perfume_tb <- trimmed_perfume_tb %>% relocate(id, .before = brand)
```

5. Exploratory Methods and Analysis

5.1 Data Analysis

To see the overall rating distribution, we do that by plotting histogram as follows

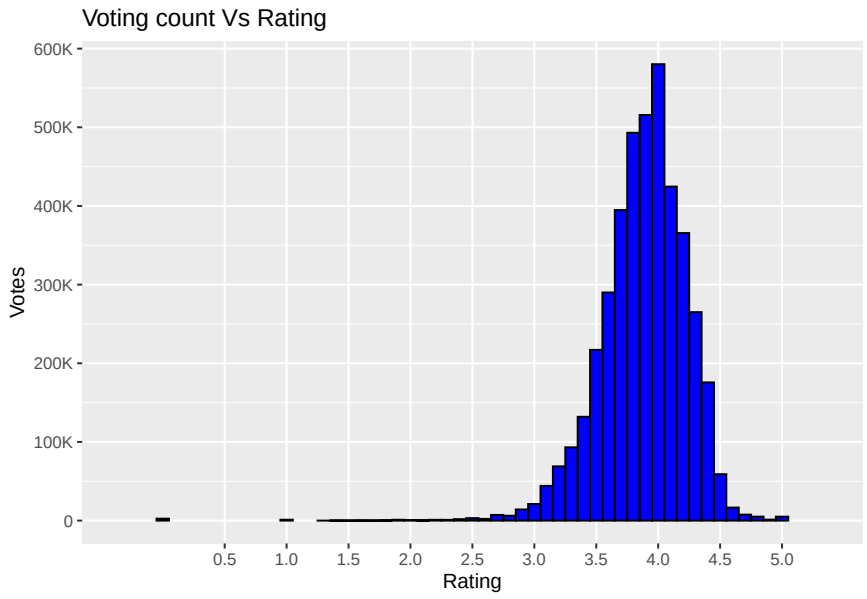
```r
# trimmed_perfume_tb subset data Ratings distribution
trimmed_perfume_tb %>%
  ggplot(aes(rating_score)) +
  geom_histogram(binwidth = 0.25, color = ("black")) +
  scale_x_discrete(limits = c(seq(0,5,0.5))) +
  labs(x = "Rating", y = "Count", caption = "Source: perfume subset data") +
  ggtitle("Purfumes Rating distribution")
```



4 star is the most frequently and 0.5 star is the less frequently.

The below plotting histogram show the voting distribution.

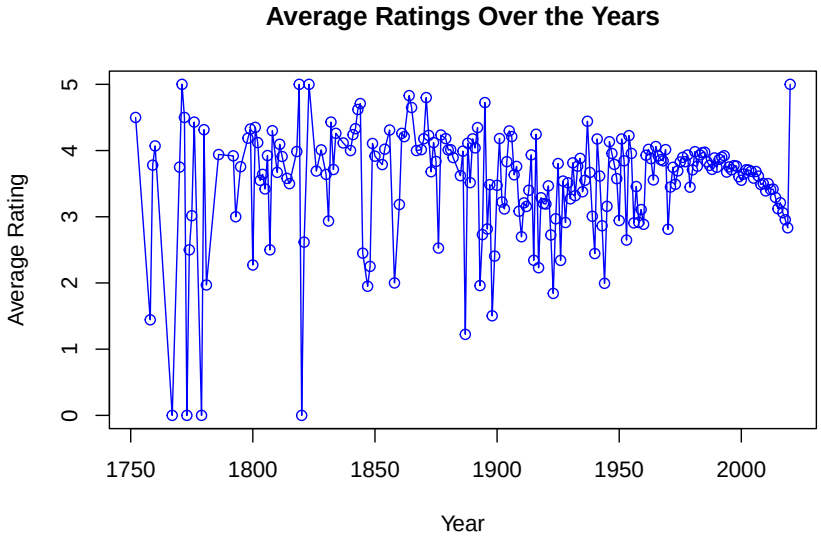
```
# Voting distributions
rat <- group_by(trimmed_perfume_tb, round(rating_score,digits = 1))
purf_rat <- summarise(rat,  count=sum(votes))
rat_score<- arrange(purf_rat, desc(0))
names(rat_score)[names(rat_score)=="round(rating_score, digits = 1)"] <- "Rating"
ggplot(rat_score, aes(x = Rating, y = count))+
  geom_bar(stat = 'identity', width = 0.1, col = "black", fill = "blue") +
  labs(x = "Rating", y = "Votes", title = "Voting count Vs Rating")+
  scale_y_continuous(breaks = c(0,100000,200000,300000,400000,500000,600000),labels = c("0","100K", "20
  scale_x_discrete(limits = c(seq(0.5,5,0.5)))
```



As rating distribution before, 4 star is the most frequently and 0.5 star is the less frequently.

Now we visualize average of ratings over the years.

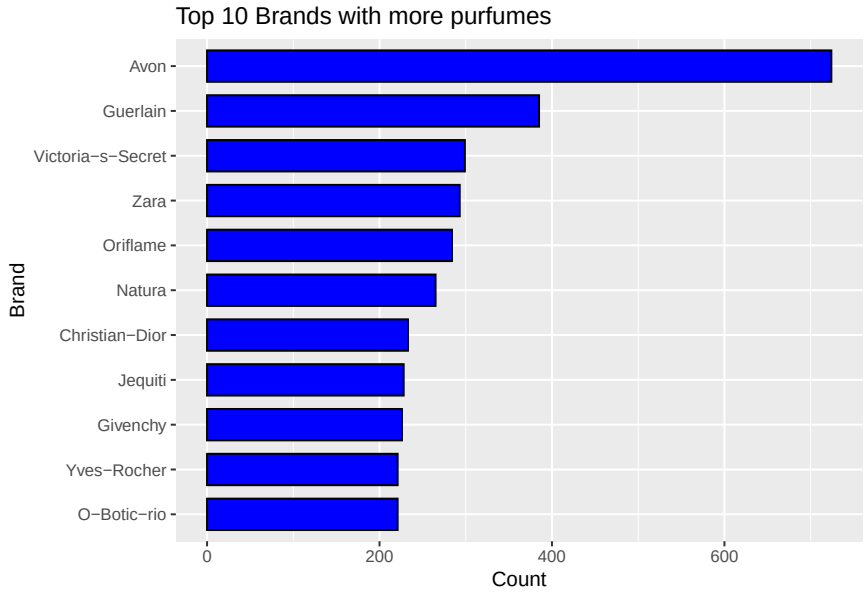
```
# Average of Ratings Over the Years
rate_year_avg <- aggregate(trimmed_perfume_tb$rating_score, by=list(Year=trimmed_perfume_tb$date),FUN=m

#Rename column x in rate_year_avg
names(rate_year_avg)[names(rate_year_avg)=="x"] <- "Average_Rating"
plot(rate_year_avg,type="o",col = "blue", xlab = "Year", ylab = "Average Rating",main = "Average Rating
```



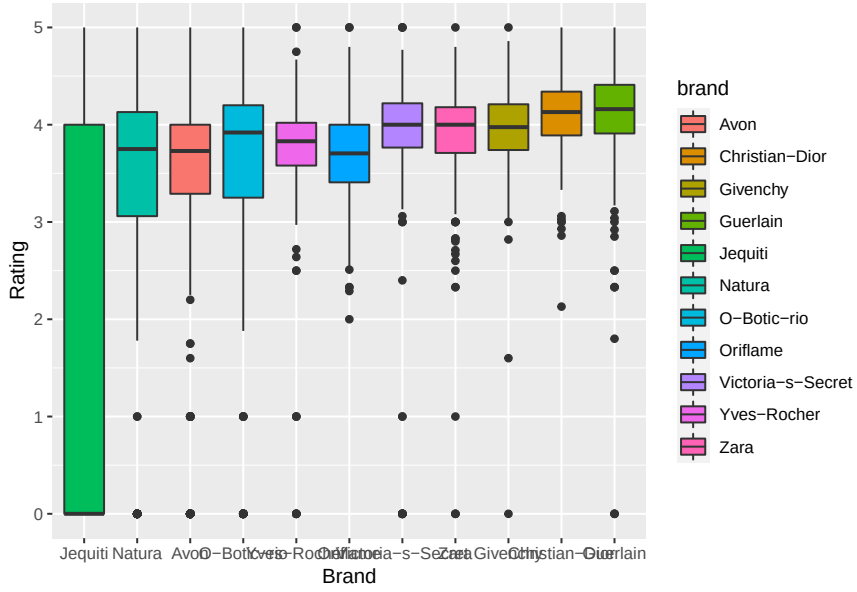
Note that the distribution is wavering, as previews plot shown that from the 1945 year and above perfume average ratings is above 2 star.

Next, now we want to discover the top 10 brands which have more perfumes, and see the relation between these brands and their ratings.

```
# number of perfumes per brand in perfume_set subset data
# Checking which brand have more perfumes and taking the top 10
select_brand <- trimmed_perfume_tb %>% group_by(brand) %>% summarize(count=n()) %>% arrange(desc(count))
ggplot(select_brand, aes(x = reorder(brand, count), y = count))+
  geom_bar(stat = 'identity', width = 0.7, col = "black", fill = "blue") +
  labs(x = "Brand", y = "Count", title = "Top 10 Brands with more purfumes")+
  coord_flip()
```



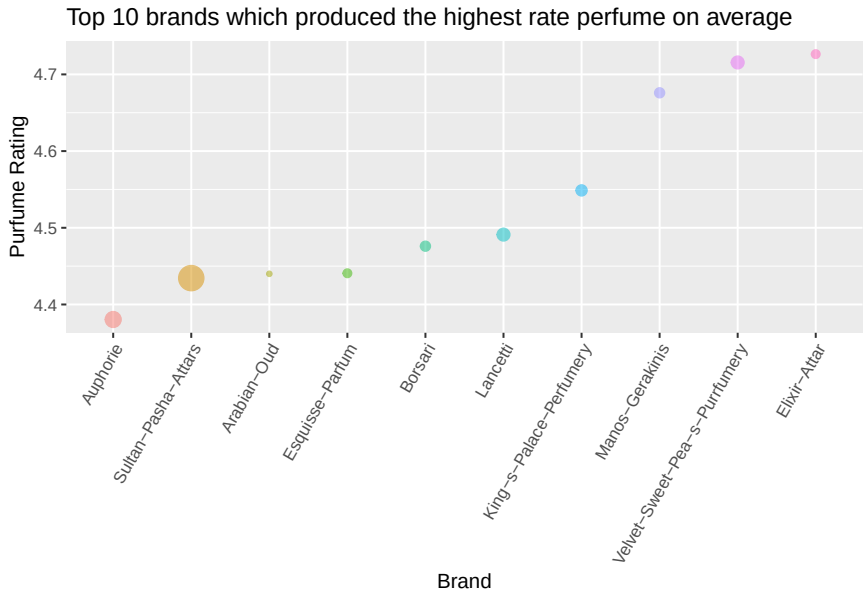

```
# With a boxplot we're going to see the relation between brand and their ratings for the brands which h
select_top_brand = as.character(select_brand$brand)
score_select_brand <-  trimmed_perfume_tb %>% filter(brand %in% select_top_brand) %>% select(brand, rati
ggplot(score_select_brand,aes(x = reorder(brand,rating_score), y = rating_score)) +
  geom_boxplot(aes(fill = brand)) +
  labs(x='Brand',y='Rating')
```



At the following plot, we want to show brand produces the highest rating perfume on average which is perfume count > perfume mean count for the same brand.

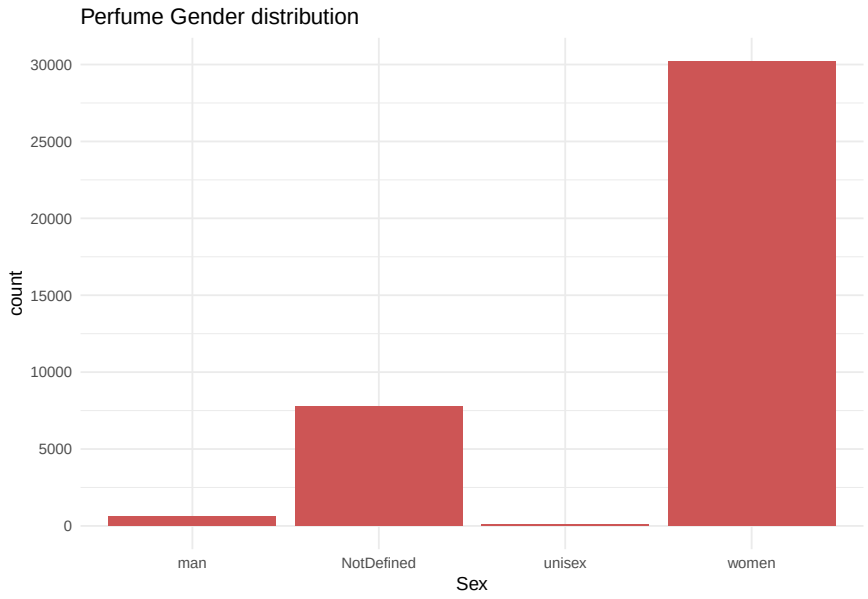
```
# we cant include rate for brands have 1 or small count of perfumes.
# we will take only the top 10
bra <- group_by(trimmed_perfume_tb, brand)
```

```
purf <- summarise(bra,  count=n(),
                  rate1= mean(rating_score)) %>% filter(count > mean(count) )
bra_score<- arrange(purf, desc(rate1)) %>% top_n(10)

ggplot(bra_score,aes(x=reorder(brand,rate1), y=rate1)) +geom_point(aes(size=count, colour=factor(rate1))
  theme(axis.text.x = element_text(angle = 60, hjust = 1) , legend.position="none") +
  labs(x="Brand", y="Purfume Rating",title="Top 10 brands which produced the highest rate perfume on ave
```



Now we visualize perfume gender distribution.

```
# Perfume Gender distribution
ggplot(trimmed_perfume_tb, aes(x=gender,fill=rating_score))+geom_bar(fill="indianred3")+ labs(x="Sex",t
  scale_y_continuous(breaks = c(0,5000,10000,15000,20000,25000,30000,35000)) +
  theme_minimal(base_size=10)
```



In accords column, there are multiple values for the same fragrance at the same row, we want to investigate this column, and see how accords can affect ratings

```
# as we see, accords column have multiple values, one fragrance have multiple accords, we want to inves
Acc <- trimmed_perfume_tb %>%
  separate_rows(accords, sep = "\\,") %>%
  group_by(accords) %>%
  summarize(PerfumeCount=n_distinct(id))
Acc
```

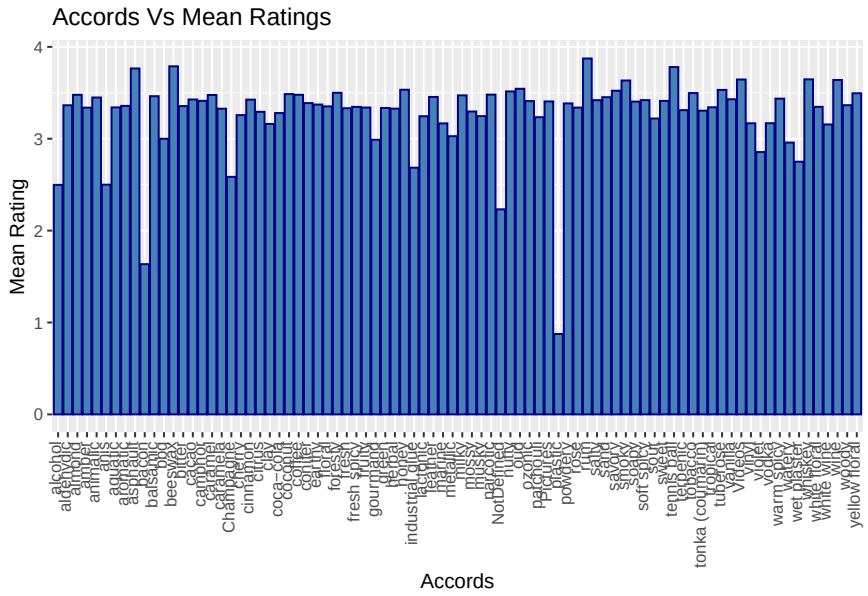
```
## # A tibble: 84 x 2
##    accords   PerfumeCount
##    <chr>            <int>
##  1 alcohol              9
##  2 aldehydic          614
##  3 almond             566
##  4 amber             5355
##  5 animalic          2520
##  6 anis                 2
##  7 aquatic           1718
##  8 aromatic         12551
##  9 asphault             4
## 10 bacon                2
## # ... with 74 more rows
```

```
Acc2 <- trimmed_perfume_tb %>%
  separate_rows(accords, sep = "\\,") %>%
  group_by(accords) %>%
  summarize(MeanRating=mean(rating_score))
Acc2
```

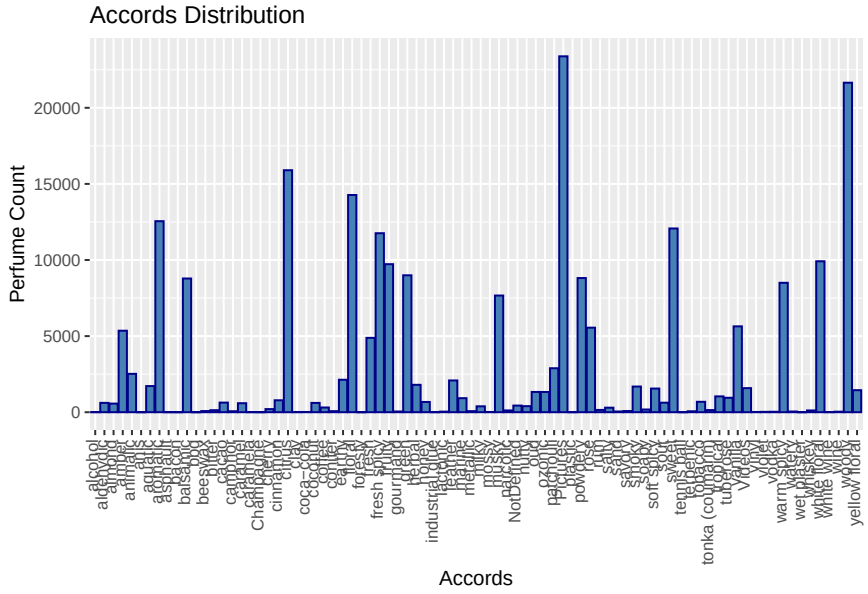
```
## # A tibble: 84 x 2
##    accords   MeanRating
##    <chr>          <dbl>
##  1 alcohol         2.50
##  2 aldehydic       3.36
##  3 almond          3.48
##  4 amber           3.34
##  5 animalic        3.45
##  6 anis            2.5
##  7 aquatic         3.34
##  8 aromatic        3.36
##  9 asphault        3.76
## 10 bacon           1.64
## # ... with 74 more rows
```

```
# As we're interested how accords can affect ratings,
# let us make a visualization of ratings per accords:
# we will see that the ratings per accords is not evenly distributed
plot_Acc2 <- Acc2 %>% ggplot(aes(x = accords, y=MeanRating)) +
  geom_bar(stat="identity", colour = "darkblue", fill = "steelblue") +
  theme(axis.text.x = element_text(angle = 90, vjust=0.25, hjust = 1)) +
  labs(title = "Accords Vs Mean Ratings",
```

```
       x = "Accords", y = "Mean Rating")
plot_Acc2
```




```
# let us make a visualization of perfumes per accords:
# we will see that the perfumes per accords is not evenly distributed
plot_Acc <- Acc %>% ggplot(aes(x = accords, y=PerfumeCount)) +
  geom_bar(stat="identity", colour = "darkblue", fill = "steelblue") +
  theme(axis.text.x = element_text(angle = 90, vjust=0.25, hjust = 1)) +
  labs(title = "Accords Distribution",
       x = "Accords", y = "Perfume Count")
plot_Acc
```



As we see, the ratings per accords is not evenly distributed.

5.2 Methods and Modeling

5.2.1 RMSE Function

We start with defining the RMSE Function that calculate the root mean square error.

```
# RMSE Function to calculate (root mean square error)
RMSE <- function(true_ratings, predicted_ratings){
  sqrt(mean((true_ratings - predicted_ratings)^2, na.rm = TRUE))
}
```

Lower RMSE is the better result.

5.2.2 Training and Testing Sets

We need to modifying accords column by separate accords information using separate_rows function for the dataset that we use, which will caused an amplification of the total number of rows.

```
# modifying accords column
# we separate accords information using separate_rows function for all out dataset that we use:
trimmed_perfume_tb_final <- trimmed_perfume_tb %>% separate_rows(accords, sep = "\\,")

#This caused an amplification of the total number of rows:
bf_train_set <- data.frame(Dataset = "trimmed_perfume_tb", Rows = length(trimmed_perfume_tb$rating_score
af_train_set <- data.frame(Dataset = "trimmed_perfume_tb_final", Rows = length(trimmed_perfume_tb_final$
union_set <- rbind(bf_train_set,af_train_set)
union_set
```

```
##                    Dataset   Rows
## 1       trimmed_perfume_tb  38755
## 2 trimmed_perfume_tb_final 229892
```

```
rm(union_set,bf_train_set,af_train_set)
```

Here we split the downloaded dataset into 2 subsets, " trimmed_perfume_tb subset " which is a training subset to train the algorithm, and " validation subset " which is a subset to test the perfume ratings.

```
# Validation set will be 10% of trimmed_perfume_tb_final data
set.seed(3, sample.kind="Rounding") # if using R 3.5 or earlier, use `set.seed(1)`
test_index1 <- createDataPartition(y = trimmed_perfume_tb_final$rating_score, times = 1, p = 0.1, list =
perfume_set1 <- trimmed_perfume_tb_final[-test_index1,]
temp1 <- trimmed_perfume_tb_final[test_index1,]
# Make sure Id in validation subset are also in perfume_set subset
validation_set1 <- temp1 %>% semi_join(perfume_set1, by = "id")

# Add rows removed from validation set back into perfume_set set
removed <- anti_join(temp1, validation_set1)
perfume_set1 <- rbind(perfume_set1, removed)

# Remove temporary files to tidy environment
rm(db,trimmed_perfume_tb_final,test_index,temp1,removed)
```

There are 206977 records in perfume_set1 training set and 22915 in validation test sets. and there are no missing values in any column.

5.2.3 Modeling

The developing and selection of the statistical models is as much a crucial aspect of machine learning as the preceding efforts at predictor development.

The modeling approach starts with a regression on the mean to establish the starting point RMSE.

The goal of this project is to achieve as low an RMSE as possible.

Model 1

In this model we predicts the same rating for all perfumes, Average perfumes rating (Calculate the overall average rating across all perfumes in perfume_set1 subset data).

```
# Model1: Average perfumes rating
# Calculate the overall average rating across all perfumes in perfume_set_final subset data)
mu <- mean(perfume_set1$rating_score)

# Calculate RMSE between each rating included in validation subset and the overall Average perfume rati
Model1_rmse <- RMSE(validation_set1$rating_score, mu)
Model1_rmse
```

```
## [1] 1.457942
```

Here we store the result of RMSE for Model1.

```
rmse_results <- data_frame(method = "Model1", RMSE = Model1_rmse)
```

Model 2

To improve the previous model we taking into account the brand effect in this model.

```
# Model2: taking into account the brand effect
# Calculate b_i (Estimate brand effect b_i)
brand_avgs <- perfume_set1 %>%
  group_by(brand) %>%
  summarise(b_i = mean(rating_score - mu))

# Predict ratings adjusting for brand effects
predicted_b_i <- mu + validation_set1 %>%
  left_join(brand_avgs, by = "brand") %>%
  pull(b_i)
# Calculate RMSE based on brand effects model
Model2_rmse <- RMSE(predicted_b_i, validation_set1$rating_score)
Model2_rmse
```

```
## [1] 1.066408
```

As shown RMSE result, the error has dropped.

Here we store the result of RMSE for Model2.

```
rmse_results <- bind_rows(rmse_results,
                          data_frame(method="Model2",
                                     RMSE = Model2_rmse ))
```

Model 3

To improve the previous model we taking into account the brand effect and votes effect in this model.

```
# Model3: taking into account the brand and votes effect
# Calculate b_u (Estimate votes effect b_u)
votes_avgs <- perfume_set1 %>%
  left_join(brand_avgs, by = "brand") %>%
  group_by(votes) %>%
  summarise(b_u = mean(rating_score - mu - b_i))
# Predict ratings adjusting for brand and votes effects
predicted_b_u <- validation_set1 %>%
  left_join(brand_avgs, by="brand") %>%
  left_join(votes_avgs, by="votes") %>%
  mutate(pred =  mu + b_i + b_u) %>%
  pull(pred)
# Calculate RMSE based on votes effects model
Model3_rmse <- RMSE(predicted_b_u, validation_set1$rating_score)
Model3_rmse
```

```
## [1] 0.9427715
```

A further improvement in the RMSE is achieved by adding the votes effect.

Now we store the result of RMSE for model3.

```
rmse_results <- bind_rows(rmse_results,
                          data_frame(method="Model3",
                                     RMSE = Model3_rmse ))
```

Until now we need to do more improvement.

Model 4

To improve the RMSE in this model we taking into account the brand effect and votes effect and accords effect.

```
# Model4: Using Model3 taking into account the accords effect
# Calculate b_g (Estimate accords effect b_g)
accords_avgs <- perfume_set1 %>%
  left_join(brand_avgs, by="brand") %>%
  left_join(votes_avgs, by="votes") %>%
  group_by(accords) %>%
  summarise(b_g = mean(rating_score - mu - b_i - b_u))
# Predict ratings adjusting for brand, votes and accords effects
predicted_b_g <- validation_set1 %>%
  left_join(brand_avgs, by="brand") %>%
```

```
  left_join(votes_avgs, by="votes") %>%
  left_join(accords_avgs, by = "accords") %>%
  mutate(pred = mu + b_i + b_u + b_g) %>%
  pull(pred)
# Calculate RMSE based on accords effects model
Model4_rmse <- RMSE(predicted_b_g, validation_set1$rating_score)
Model4_rmse
```

```
## [1] 0.942053
```

Adding accords effects into the model provide an Unnoticeable improvement in the accuracy of the algorithm.

Now we store the result of RMSE for model4.

```
rmse_results <- bind_rows(rmse_results,
                          data_frame(method="Model4",
                                     RMSE = Model4_rmse ))
```

Regularizing Model

We regularizing Model4 (which is taking into account the brand effect and votes effect and accords effect) in order to constrain the variability of effect sizes.

```
# Regularized model: Using Model4 (taking into account the the brand and votes and accords effect)
lambdas <- seq(0, 10, 0.25)

rmses <- sapply(lambdas, function(l){

  mu <- mean(perfume_set1$rating_score)

  b_i <- perfume_set1 %>%
    group_by(brand) %>%
    summarize(b_i = sum(rating_score - mu)/(n()+l))

  b_u <- perfume_set1 %>%
    left_join(b_i, by="brand") %>%
    group_by(votes) %>%
    summarize(b_u = sum(rating_score - b_i - mu)/(n()+l))

  b_g <- perfume_set1 %>%
    left_join(brand_avgs, by = "brand") %>%
    left_join(votes_avgs, by = "votes") %>%
    group_by(accords) %>%
    summarise(b_g = sum(rating_score - mu - b_i - b_u)/(n()+l))

  predicted_ratings <-
    validation_set1 %>%
    left_join(b_i, by = "brand") %>%
    left_join(b_u, by = "votes") %>%
    left_join(b_g, by = "accords") %>%
    mutate(pred = mu + b_i + b_u + b_g) %>%
    pull(pred)
```

```
  return(RMSE(predicted_ratings, validation_set1$rating_score))
})
```

The optimal lambda for the model is:

```
# The optimal lambda
lambda <- lambdas[which.min(rmses)]
lambda
```

```
## [1] 10
```

The new RMSE result will be:

```
min(rmses)
```

```
## [1] 0.9132027
```

Now we store the result of RMSE for Regularizing Model.

```
rmse_results <- bind_rows(rmse_results,
                          data_frame(method="Regularizing Model",
                                     RMSE = min(rmses)))
```

6. Result

The RMSE values of all the represented models are the following:

```
rmse_results
```

```
## # A tibble: 5 x 2
##   method              RMSE
##   <chr>              <dbl>
## 1 Model1             1.46
## 2 Model2             1.07
## 3 Model3             0.943
## 4 Model4             0.942
## 5 Regularizing Model 0.913
```

We therefore found the lowest value of RMSE that is 0.9132027.

7. Discussion

So we can say that the final model for our project is the optimal model for this recommendation system.

8. Conclusion

We confirm that a recommendation system with machine learning algorithm to predict perfume ratings with the selected dataset was built in this project.

The model including the effect of brand and votes and accords is the optimal model to use for this recommendation system.

9. References

1. Rafael A. Irizarry (2021), Introduction to Data Science: https://rafalab.github.io/dsbook/
2. Model Fitting and Recommendation Systems Overview: https://learning.edx.org/course/course-v1:HarvardX+PH125.8x+2T2021/block-v1:HarvardX+PH125.8x+2T2021+type@sequential+block@568d02a4412440489621921b87e7536f/block-v1:HarvardX+PH125.8x+2T2021+type@vertical+block@d7b4b0783dd443d1bd14504fe2333c24

10. Technical Annex

- This paper was written in R Markdown and converted into a PDF.
- For conversion process, latex is required to be installed in your system (https://www.latex-project.org/get/).