AD699_Assignment.3

2023-10-30

R Markdown

This is an R Markdown document. Markdown is a simple formatting syntax for authoring HTML, PDF, and MS Word documents. For more details on using R Markdown see http://rmarkdown.rstudio.com (http://rmarkdown.rstudio.com).

When you click the **Knit** button a document will be generated that includes both content as well as the output of any embedded R code chunks within the document. You can embed an R code chunk like this:

Main Topic: Classification

● K-NearestNeighbors

```
songs <- read.csv("C://Users/maxma/Documents/AD 699/AD 699 assignment 3/spot23.csv")
```

a. What song did you pick?

I picked the song “Am I Dreaming” by Metro Boomin & A$AP Rocky, Roisee.

b. This song is from my favorite movie that came out this year. The movie name is Spiderman: Across the spiderverse.

c.

danceability: 0.6 energy: 0.53 speechiness: 0.04 acousticness: 0.04 liveness: 0.21 valence: 0.13

2.

```
row_index <- 172
my_song <- songs[row_index, ]
```

3.

```
spotify <- read.csv("C://Users/maxma/Documents/AD 699/AD 699 assignment 3/spotify.csv")
str(spotify)
```

```
## 'data.frame':    2017 obs. of  17 variables:
##  $ X               : int  0 1 2 3 4 5 6 7 8 9 ...
##  $ acousticness    : num  0.0102 0.199 0.0344 0.604 0.18 0.00479 0.0145 0.0202 0.0481 0.00208 ...
##  $ danceability    : num  0.833 0.743 0.838 0.494 0.678 0.804 0.739 0.266 0.603 0.836 ...
##  $ duration_ms     : int  204600 326933 185707 199413 392893 251333 241400 349667 202853 226840 ...
##  $ energy          : num  0.434 0.359 0.412 0.338 0.561 0.56 0.472 0.348 0.944 0.603 ...
##  $ instrumentalness: num  2.19e-02 6.11e-03 2.34e-04 5.10e-01 5.12e-01 0.00 7.27e-06 6.64e-01 0.00 0.00 ...
##  $ key             : int  2 1 2 5 5 8 1 10 11 7 ...
##  $ liveness        : num  0.165 0.137 0.159 0.0922 0.439 0.164 0.207 0.16 0.342 0.571 ...
##  $ loudness        : num  -8.79 -10.4 -7.15 -15.24 -11.65 ...
##  $ mode            : int  1 1 1 1 0 1 1 0 0 1 ...
##  $ speechiness     : num  0.431 0.0794 0.289 0.0261 0.0694 0.185 0.156 0.0371 0.347 0.237 ...
##  $ tempo           : num  150.1 160.1 75 86.5 174 ...
##  $ time_signature  : num  4 4 4 4 4 4 4 4 4 4 ...
##  $ valence         : num  0.286 0.588 0.173 0.23 0.904 0.264 0.308 0.393 0.398 0.386 ...
##  $ target          : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ song_title      : chr  "Mask Off" "Redbone" "Xanny Family" "Master Of None" ...
##  $ artist          : chr  "Future" "Childish Gambino" "Future" "Beach House" ...
```

a.

Target is a numeric variable.

```
spotify$target <- factor(spotify$target)
```

b.

```
unique(spotify$target)
```

```
## [1] 1 0
## Levels: 0 1
```

```
table(spotify$target)
```

```
## 
##    0    1 
##  997 1020
```

George liked 1020 songs and does not like 997 songs.

4.

```
any(is.na(spotify))
```

```
## [1] FALSE
```

The dataset does not have any NA values.

5.a

```
library(tidyverse)
```

```
## Warning: package 'tidyverse' was built under R version 4.2.3
```

```
## Warning: package 'ggplot2' was built under R version 4.2.3
```

```
## Warning: package 'tibble' was built under R version 4.2.3
```

```
## Warning: package 'tidyr' was built under R version 4.2.3
```

```
## Warning: package 'readr' was built under R version 4.2.3
```

```
## Warning: package 'purrr' was built under R version 4.2.3
```

```
## Warning: package 'dplyr' was built under R version 4.2.3
```

```
## Warning: package 'forcats' was built under R version 4.2.3
```

```
## Warning: package 'lubridate' was built under R version 4.2.3
```

```
## ── Attaching core tidyverse packages ──────────────────────── tidyverse 2.0.0 ──
## ✓ dplyr     1.1.3     ✓ readr     2.1.4
## ✓ forcats   1.0.0     ✓ stringr   1.5.0
## ✓ ggplot2   3.4.2     ✓ tibble    3.2.1
## ✓ lubridate 1.9.2     ✓ tidyr     1.3.0
## ✓ purrr     1.0.2
## ── Conflicts ────────────────────────────────────────── tidyverse_conflicts() ──
## ✕ dplyr::filter() masks stats::filter()
## ✕ dplyr::lag()    masks stats::lag()
## ℹ Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
```

```
columns_to_convert <- c("danceability_.", "energy_.", "speechiness_.", "valence_.", "acousticness_.", "liveness_.")
my_song <- my_song %>%
  mutate(across(all_of(columns_to_convert), ~./100))
```

5.b

```
my_song <- my_song %>%
  rename(
    danceability = danceability_.,
    energy = energy_.,
    speechiness = speechiness_.,
    valence = valence_.,
    acousticness = acousticness_.,
    liveness = liveness_.
  )
```

6.

```
set.seed(1626)
train.index <- sample(c(1:nrow(spotify)), nrow(spotify)*0.6)
train.df <- spotify[train.index, ]
valid.df <- spotify[-train.index, ]
```

7.a

```
library(dplyr)
G_liked <- filter(train.df, target=="1")
G_notliked <- filter(train.df, target=="0")
```

```
t.test(G_liked$danceability, G_notliked$danceability)
```

```
##
##  Welch Two Sample t-test
##
## data:  G_liked$danceability and G_notliked$danceability
## t = 7.1632, df = 1202.3, p-value = 1.371e-12
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  0.04760652 0.08352156
## sample estimates:
## mean of x mean of y
## 0.6486601 0.5830960
```

```
t.test(G_liked$energy, G_notliked$energy)
```

```
##
##  Welch Two Sample t-test
##
## data:  G_liked$energy and G_notliked$energy
## t = 2.453, df = 1070.2, p-value = 0.01433
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  0.005967017 0.053678562
## sample estimates:
## mean of x mean of y
## 0.6911175 0.6612947
```

```
t.test(G_liked$speechiness, G_notliked$speechiness)
```

```
##
##  Welch Two Sample t-test
##
## data:  G_liked$speechiness and G_notliked$speechiness
## t = 6.6869, df = 1077.8, p-value = 3.651e-11
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  0.02468074 0.04518021
## sample estimates:
##  mean of x  mean of y
## 0.11229670 0.07736623
```

```
t.test(G_liked$valence, G_notliked$valence)
```

```
##
##  Welch Two Sample t-test
##
## data:  G_liked$valence and G_notliked$valence
## t = 4.5589, df = 1207.6, p-value = 5.666e-06
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  0.03614244 0.09075202
## sample estimates:
## mean of x mean of y
## 0.5248355 0.4613882
```

```
t.test(G_liked$acousticness, G_notliked$acousticness)
```

```
##
##  Welch Two Sample t-test
##
## data:  G_liked$acousticness and G_notliked$acousticness
## t = -5.6977, df = 1067.9, p-value = 1.57e-08
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -0.11341091 -0.05530715
## sample estimates:
## mean of x mean of y
## 0.1466622 0.2310213
```

```
t.test(G_liked$liveness, G_notliked$liveness)
```

```
##
##  Welch Two Sample t-test
##
## data:  G_liked$liveness and G_notliked$liveness
## t = 1.6024, df = 1189.7, p-value = 0.1093
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -0.003220761  0.031931353
## sample estimates:
## mean of x mean of y
## 0.1982505 0.1838952
```

danceability: t = 7.1632 p-value = 1.371e-12 The t-test for danceability has the smallest p-value, indicating the most significant difference between the groups. speechiness: t = 6.6869 p-value = 3.651e-11

acousticness: t = -5.6977 p-value = 1.57e-08

valence: t = 4.5589 p-value = 5.666e-06

energy: t = 2.453 p-value = 0.01433 This t test for energy has a somewhat small p-value. liveness: t = 1.6024 p-value = 0.1093 This t test value is not that significant. If we make the significant threshold for this p value or the alpha value to be 0.5, the liveness p value is bigger. Thus there are not a lot of significant difference between the variables tested.

7.b

```
my_song <- subset(my_song, select = -liveness)
```

7.c It may make sense to remove variables with very similar values for both outcome classes in a k-nearest neighbors (k-NN) model because these variables are less informative for distinguishing between classes, potentially leading to noise in the model’s predictions and increased computational complexity without adding meaningful discriminatory power.

8.

```
library(caret)
```

```
## Warning: package 'caret' was built under R version 4.2.3
```

```
## Loading required package: lattice
```

```
##
## Attaching package: 'caret'
```

```
## The following object is masked from 'package:purrr':
##
##     lift
```

```
# Specify the columns to normalize
columns_to_normalize <- c("acousticness", "danceability", "energy", "speechiness", "valence")

# Normalize train.df
norm_values <- preProcess(train.df[, columns_to_normalize], method = c("center", "scale"))
train.norm.df <- as.data.frame(predict(norm_values, train.df[, columns_to_normalize]))

# Normalize valid.df
valid.norm.df <- as.data.frame(predict(norm_values, valid.df[, columns_to_normalize]))

# Normalize spotify
spotify.norm.df <- as.data.frame(predict(norm_values, spotify[, columns_to_normalize]))

# Normalize my_song
my_song.norm <- as.data.frame(predict(norm_values, my_song[, columns_to_normalize]))
```

9.

```
library(FNN)
```

```
## Warning: package 'FNN' was built under R version 4.2.3
```

```
my_song.norm <- my_song.norm[, 1:2]

 nn <- knn(train = train.norm.df[, 1:2], test = my_song.norm,
 cl = train.df[, 15], k = 7)
 row.names(train.df)[attr(nn, "nn.index")]
```

```
## [1] "1448" "798"  "9"    "466"  "1864" "975"  "581"
```

```
nn
```

```
## [1] 1
## attr(,"nn.index")
##      [,1] [,2] [,3] [,4] [,5] [,6] [,7]
## [1,]  538  411  599  983  966   68  223
## attr(,"nn.dist")
##            [,1]       [,2]       [,3]       [,4]       [,5]       [,6]
## [1,] 0.02400641 0.03093992 0.03614287 0.04500627 0.04590258 0.04681017
##            [,7]
## [1,] 0.05055079
## Levels: 1
```

George will like the song as the outcome is 1.

```
values_to_filter <- c("1448", "798", "9", "466", "1864", "975", "581")

filtered_data <- subset(spotify, X %in% values_to_filter,
                        select = c("song_title", "artist","target"))

print(filtered_data)
```

```
##            song_title                 artist target
## 10     Digital Animal            Honey Claws      1
## 467       Close to Me Teams vs. Star Slinger      1
## 582      Harlem Shake                 Baauer      1
## 799  Whatcha Gonna Do         Koopsta Knicca      1
## 976           I Got U            Duke Dumont      1
## 1449          Twinbow                Slushii      0
## 1865        Don't Say               Lullanas      0
```

Here are the seven nearest songs for my song. The “target” variable contains two distinct classes I predicted. They are 0 (meaning George does not like my song and 1 meaning George likes my song)

10.

```r
accuracy.df <- data.frame(k = seq(1, 50, 1), accuracy = rep(0, 50))

for(i in 1:50) {
knn.pred <- knn(train.norm.df[, 1:2], valid.norm.df[, 1:2],
cl = train.df[, 15], k = i)
accuracy.df[i, 2] <- confusionMatrix(knn.pred, valid.df[, 15])$overall[1]
}

View(accuracy.df)
```

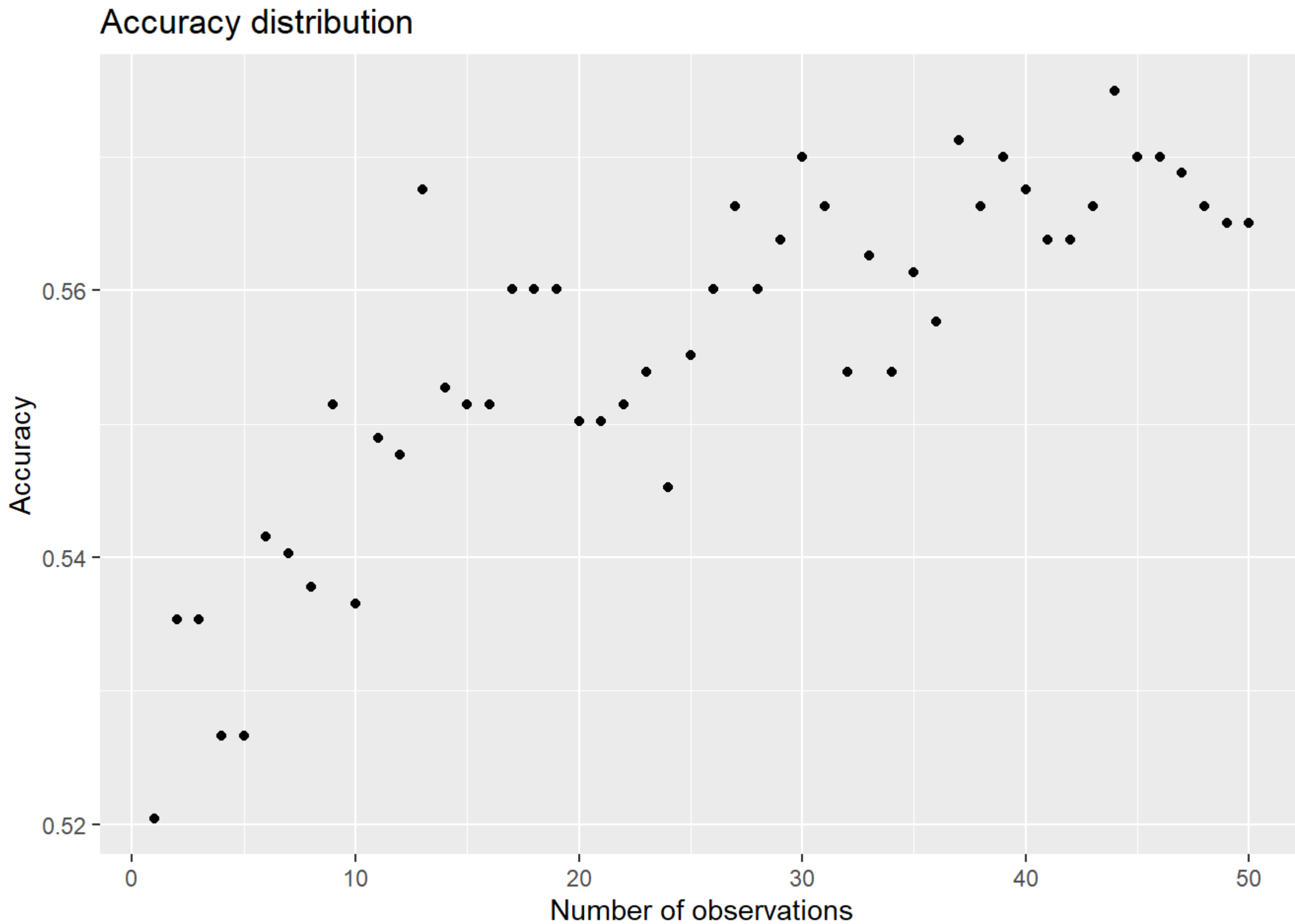
```r
max(accuracy.df$accuracy)
```

```
## [1] 0.574969
```

The K value of 44 has the highest accuracy.

11.

```r
library(ggplot2)
ggplot(accuracy.df, aes(x = k, y = accuracy)) +
  geom_point() +
  labs(x = "Number of observations", y = "Accuracy", title = "Accuracy distribution")
```



12.

```
my_song.norm <- my_song.norm[, 1:2]

 nn2 <- knn(train = train.norm.df[, 2:3], test = my_song.norm,
 cl = train.df[, 15], k = 44)
 row.names(train.df)[attr(nn2, "nn.index")]
```

```
##  [1] "1031" "1272" "449"  "1780" "103"  "1855" "1679" "989"  "1774" "131"
## [11] "1675" "430"  "1242" "1099" "843"  "1290" "1245" "1897" "1040" "1845"
## [21] "110"  "1424" "783"  "813"  "1411" "1256" "452"  "1493" "418"  "56"
## [31] "1253" "165"  "1284" "267"  "65"   "835"  "2002" "177"  "1050" "1059"
## [41] "1586" "296"  "440"  "896"
```

```
nn2
```

```
## [1] 0
## attr(,"nn.index")
##      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10] [,11] [,12] [,13] [,14]
## [1,]  565  120  761  230  479 1050  357    7   88  1189   435  1029   103   970
##      [,15] [,16] [,17] [,18] [,19] [,20] [,21] [,22] [,23] [,24] [,25] [,26]
## [1,]   130  1171   702   188   572   734   554  1002   126   287   539   419
##      [,27] [,28] [,29] [,30] [,31] [,32] [,33] [,34] [,35] [,36] [,37] [,38]
## [1,]   629   669  1041    93  1107   186   673   169   920   152   513   462
##      [,39] [,40] [,41] [,42] [,43] [,44]
## [1,]   268   914   805   222   414   380
## attr(,"nn.dist")
##            [,1]       [,2]      [,3]       [,4]       [,5]       [,6]      [,7]
## [1,] 0.01322616 0.01322616 0.0492189 0.05915036 0.05990102 0.09198024 0.1066327
##          [,8]      [,9]     [,10]     [,11]     [,12]     [,13]     [,14]
## [1,] 0.124444 0.1365059 0.1578499 0.1711442 0.1711851 0.1733729 0.1758548
##          [,15]    [,16]     [,17]     [,18]     [,19]     [,20]     [,21]
## [1,] 0.1864693 0.192038 0.1989024 0.2085761 0.2129864 0.2296923 0.2445257
##          [,22]    [,23]     [,24]     [,25]     [,26]     [,27]    [,28]
## [1,] 0.2452434 0.251113 0.2607364 0.2620592 0.2738096 0.2774213 0.280901
##         [,29]     [,30]     [,31]     [,32]     [,33]     [,34]     [,35]
## [1,] 0.283596 0.2859832 0.2916621 0.3057882 0.3105504 0.3141821 0.3228606
##          [,36]    [,37]     [,38]     [,39]     [,40]     [,41]    [,42]
## [1,] 0.3260178 0.326114 0.3277187 0.3283484 0.3305773 0.3461485 0.349492
##          [,43]     [,44]
## [1,] 0.3497411 0.3535365
## Levels: 0
```

```r
values_to_filter2 <- c( "1031","1272","449","1780","103","1855","1679","989","1774","131",  "1675","430", "1242", "1099", "8
43",  "1290", "1245", "1897", "1040", "1845", "110","1424", "783","813","1411", "1256", "452","1493", "418","56", "1253", "1
65","1284","267", "65", "835", "2002", "177", "1050","1059", "1586","296","440","896" )



filtered_data2 <- subset(spotify, X %in% values_to_filter2,
                        select = c("song_title", "artist","target"))

table(filtered_data2$target)
```

```
##
##  0  1
## 24 20
```

The results are wildly different. First of all, I chose a larger number to do the accuracy test and this time George did not like my song. The outcome class here again was 0 and 1. From that we can see that George liked 20 songs and did not like 24 songs.

13. Using numeric attributes to predict whether someone will like a song can have limitations. It assumes a linear relationship between numeric features and the likelihood of liking a song, which may not capture more complex patterns in music preferences. The binary target variable (0 or 1) oversimplifies the notion of liking, as musical preferences can be highly nuanced and multifaceted. Also, k-NN tends to memorize the training data rather than generalize from it. This can result in overfitting if the training dataset is noisy.

Naive Bayes: 1.

```r
fitness_zone <- read.csv("C://Users/maxma/Documents/AD 699/AD 699 assignment 3/fitness_zone.csv")
```

```r
sapply(fitness_zone, class)
```

```
##       booking_id months_as_member           weight      days_before
##        "integer"        "integer"        "numeric"      "character"
##      day_of_week             time         category         attended
##      "character"      "character"      "character"        "integer"
```

2.a

```
missing_summary <- summary(is.na(fitness_zone))
print(missing_summary)
```

```
##  booking_id      months_as_member   weight        days_before
##  Mode :logical   Mode :logical    Mode :logical   Mode :logical
##  FALSE:1500      FALSE:1500       FALSE:1480      FALSE:1500
##                                   TRUE :20
##  day_of_week        time          category        attended
##  Mode :logical   Mode :logical   Mode :logical   Mode :logical
##  FALSE:1500      FALSE:1500      FALSE:1500      FALSE:1500
##
```

The variable weight has 20 missing values. The other variables do not have any missing vallues.

3.

```
fitness_zone$days_before <- as.factor(fitness_zone$days_before)
fitness_zone$day_of_week <- as.factor(fitness_zone$day_of_week)
fitness_zone$time <- as.factor(fitness_zone$time)
fitness_zone$category <- as.factor(fitness_zone$category)
```

4.

```
table(fitness_zone$attended)
```

```
##
##    0    1
## 1046  454
```

a. The response variables are 0 and 1. 0 means not attended and 1 means attended. There are more 0s than there are 1s.Thus not attending is more prevalent.

b.

```
fitness_zone$attended <- as.factor(fitness_zone$attended)
```

5.

While unique ID columns such as booking_id are essential for data management and record identification purposes, they are not suitable as predictors for predictive modeling tasks. It usually does not contain meaningful information related to the target variable or the underlying patterns in the data.

6.

```
num_bins <- 5

fitness_zone$months_as_member_binned <- cut(fitness_zone$months_as_member,
                                            breaks = quantile(fitness_zone$months_as_member,
                                                              probs = seq(0, 1, length.out = num_bins + 1)),
                                            labels = c("very_new", "new", "loyal", "silver", "gold_member"),
                                            include.lowest = TRUE)

fitness_zone$weight_binned <- cut(fitness_zone$weight,
                                  breaks = quantile(fitness_zone$weight,
                                                    probs = seq(0, 1, length.out = num_bins + 1),
                                                    na.rm = TRUE),
                                  labels = c("very_light", "light", "moderate", "moderately_heavy", "heavy"),
                                  include.lowest = TRUE)
```

a.

```
table(fitness_zone$months_as_member_binned)
```

```
##
##    very_new         new       loyal      silver gold_member
##         359         261         289         299         292
```

```
table(fitness_zone$weight_binned)
```

```
##
##       very_light            light         moderate moderately_heavy
##              296              296              296              296
##            heavy
##              296
```

b. In equal width binning, you divide the range of the numeric variable into a fixed number of equally spaced bins.In equal frequency binning, you divide the data into a fixed number of bins such that each bin contains approximately the same number of observations.

If the data has a skewed distribution (e.g., positively or negatively skewed), equal width binning may result in some bins containing very few data points, making those bins less informative. Equal frequency binning ensures that each bin has a roughly equal number of data points, even in the presence of skewness.

When the data contain outliers, equal width binning can be sensitive to these extreme values, resulting in bins that are heavily influenced by outliers. Equal frequency binning is more robust to outliers because it focuses on the distribution of data points rather than their specific values.

c.

```
library(forcats)

fitness_zone$weight_binned <- fct_explicit_na(fitness_zone$weight_binned, "NA")
```

```
## Warning: `fct_explicit_na()` was deprecated in forcats 1.0.0.
## ℹ Please use `fct_na_value_to_level()` instead.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```

```
weight_table <- table(fitness_zone$weight_binned)

weight_table
```

```
##
##       very_light            light         moderate moderately_heavy
##              296              296              296              296
##            heavy               NA
##              296               20
```

i. Sometimes, the presence or absence of missing values (NAs) itself can be informative. It can indicate a specific pattern or behavior in the data that may be relevant to the analysis. NAs can also reflect the quality of the data. Variables with a high proportion of missing values may be less reliable or may need special treatment during analysis.

ii.

```
level_table <- table(fitness_zone$weight_binned, useNA = "ifany")

print(level_table)
```

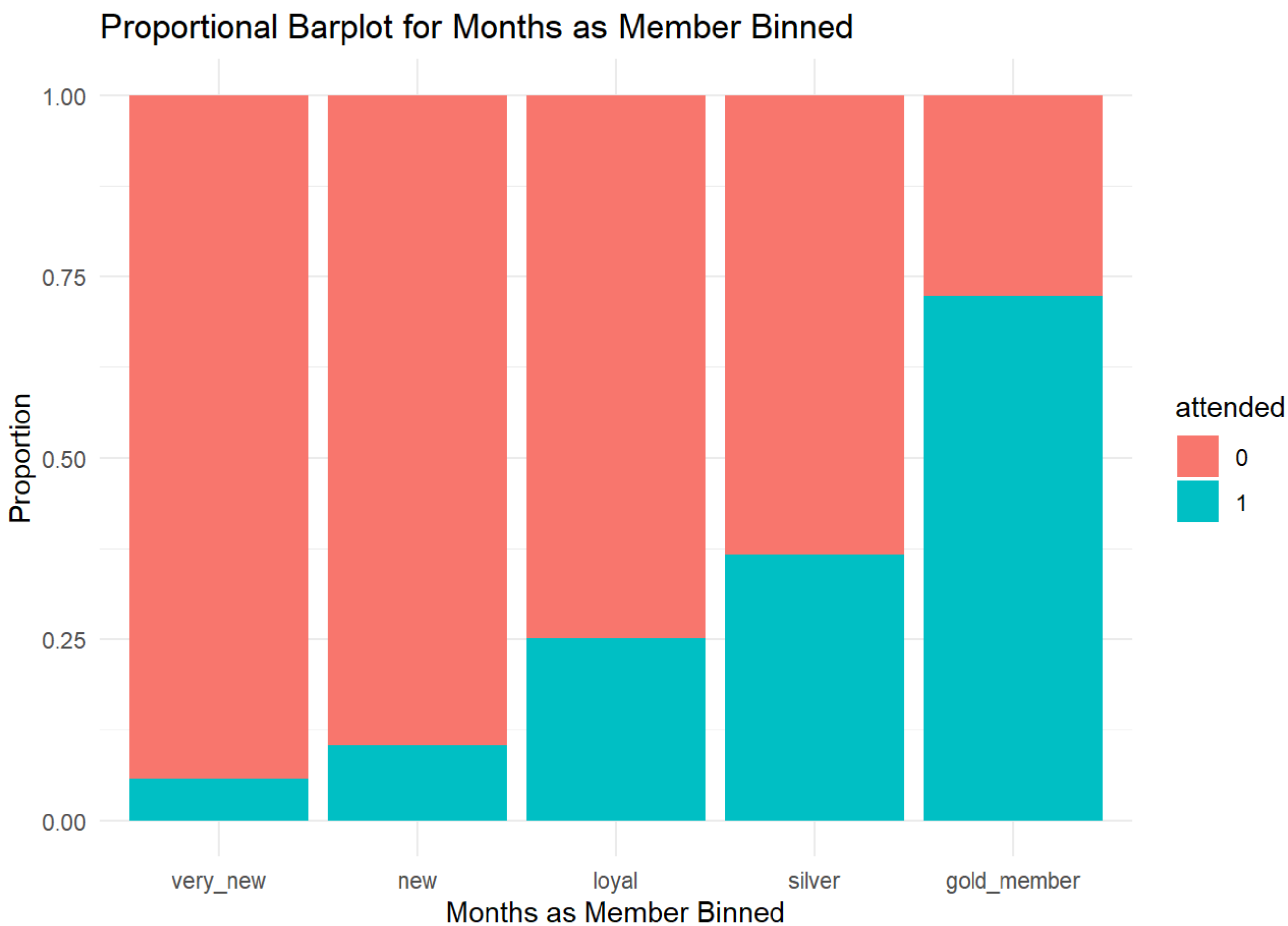
```
##
##       very_light            light         moderate moderately_heavy
##              296              296              296              296
##            heavy               NA
##              296               20
```

7.

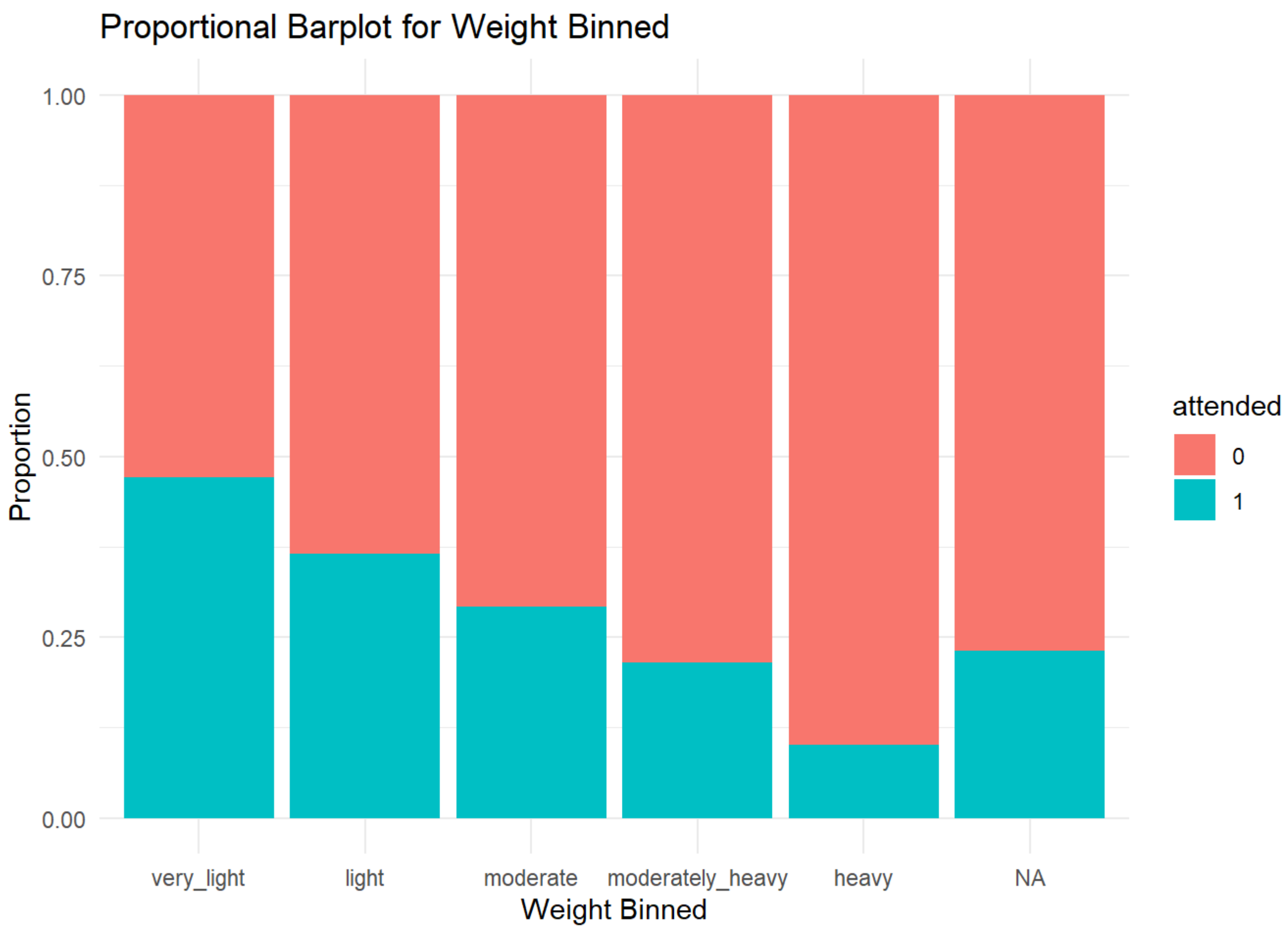
```
set.seed(1626)
train.index <- sample(c(1:nrow(fitness_zone)), nrow(fitness_zone)*0.6)
train_df <- fitness_zone[train.index, ]
valid_df <- fitness_zone[-train.index, ]
```

8.

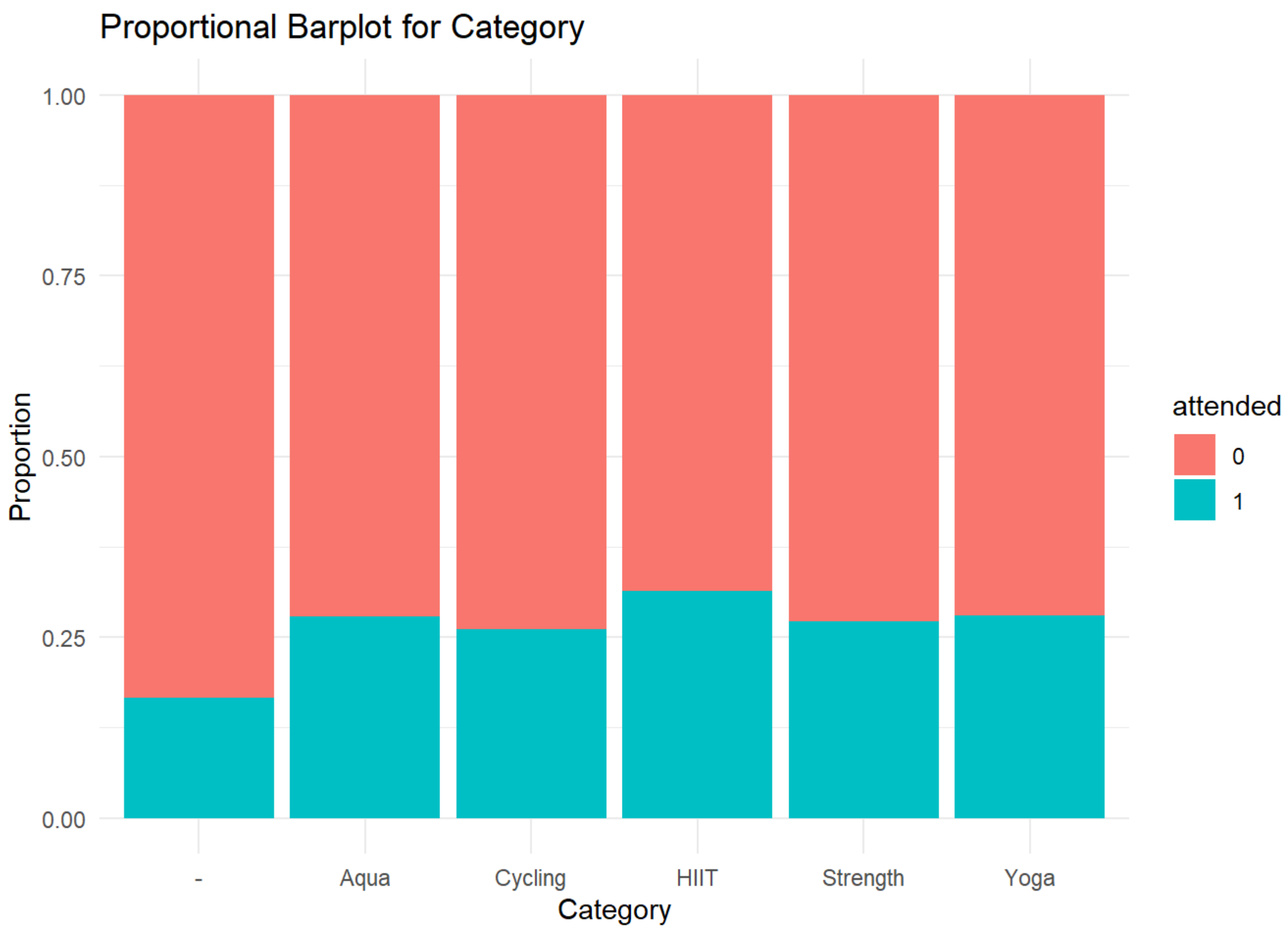
```
library(ggplot2)

ggplot(train_df, aes(x = months_as_member_binned, fill = attended)) +
  geom_bar(position = "fill") +
  labs(title = "Proportional Barplot for Months as Member Binned") +
  xlab("Months as Member Binned") +
  ylab("Proportion") +
  theme_minimal()
```



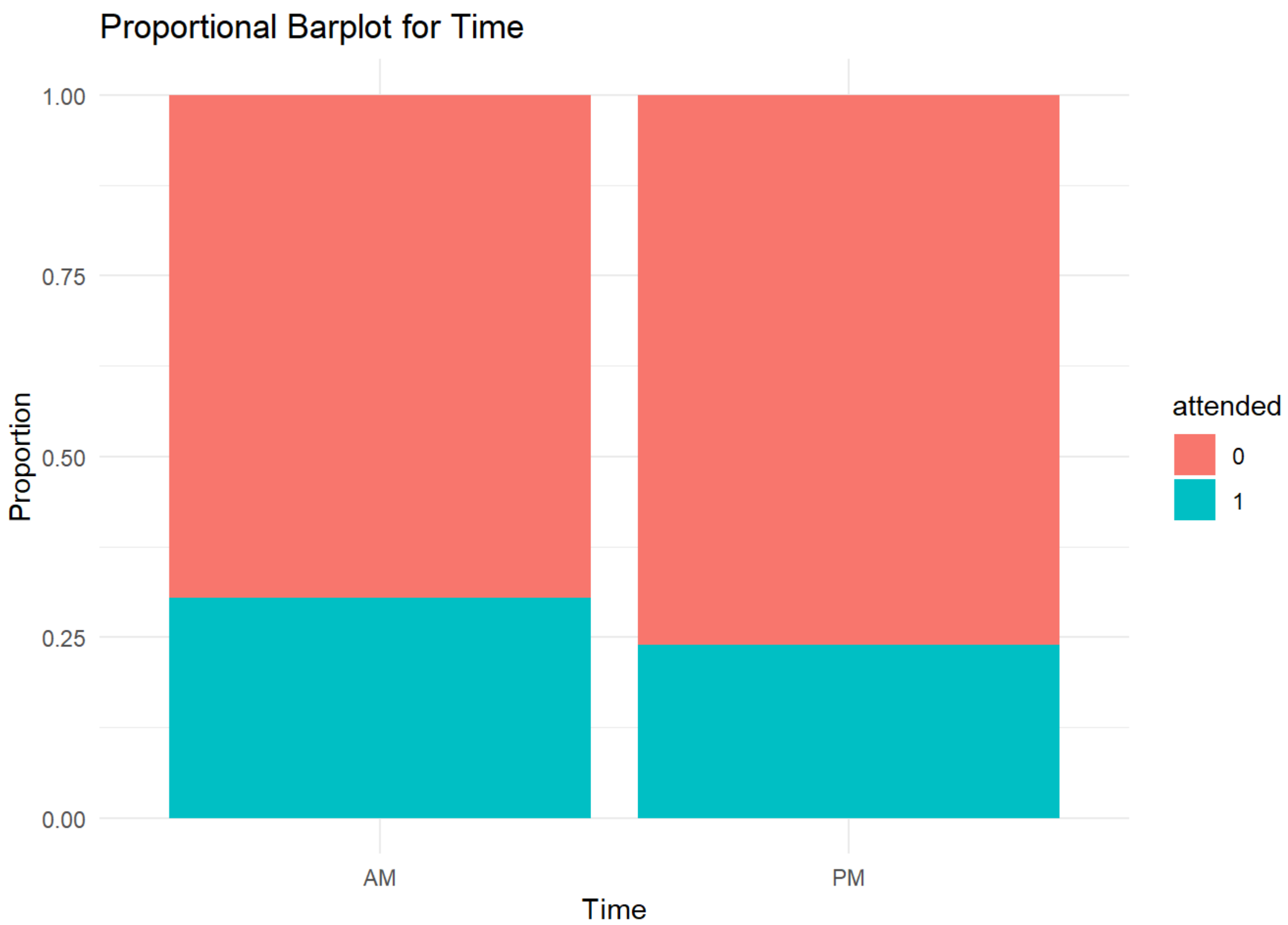

```
ggplot(train_df, aes(x = weight_binned, fill = attended)) +
  geom_bar(position = "fill") +
  labs(title = "Proportional Barplot for Weight Binned") +
  xlab("Weight Binned") +
  ylab("Proportion") +
  theme_minimal()
```



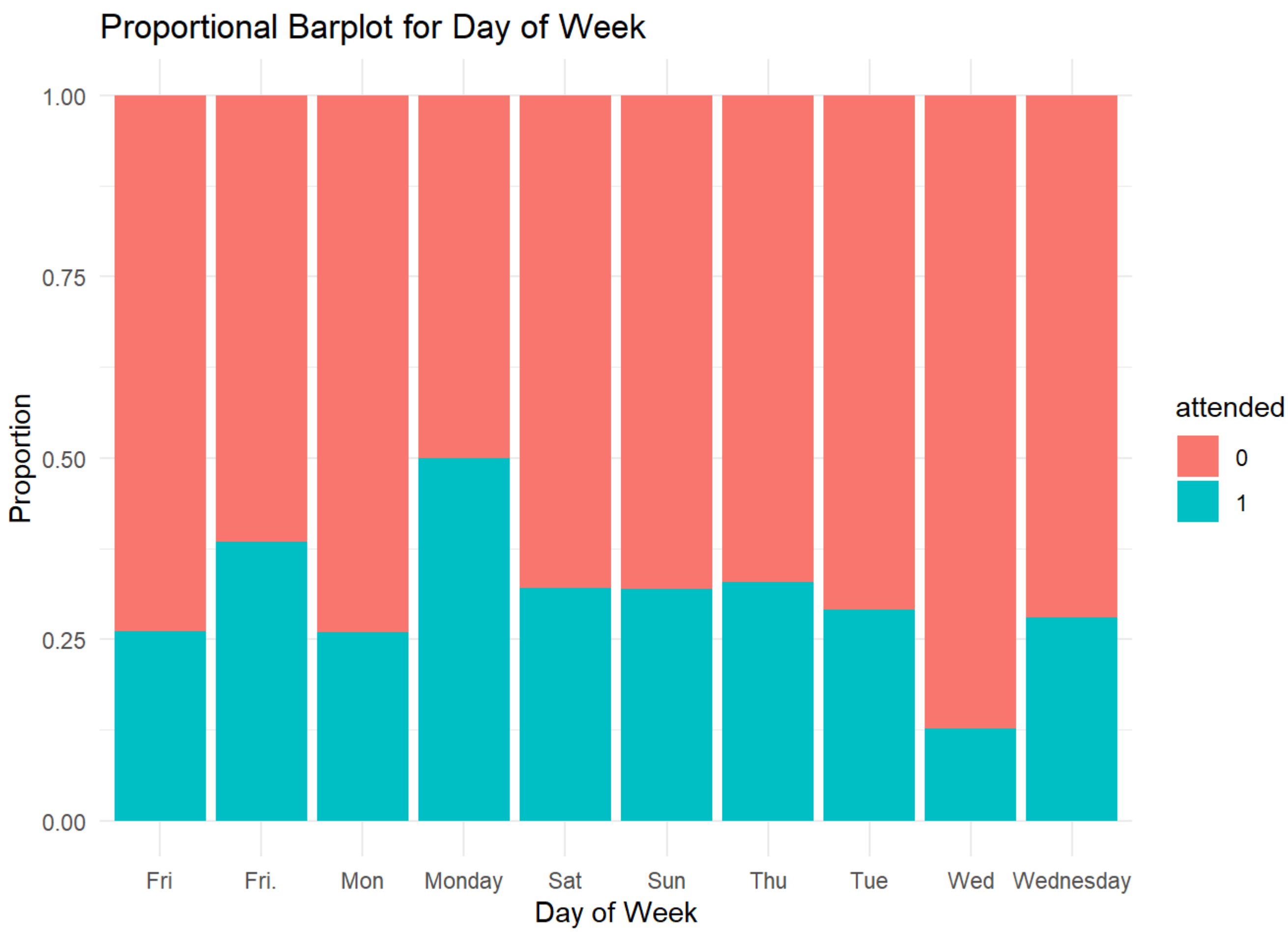

```
ggplot(train_df, aes(x = category, fill = attended)) +
  geom_bar(position = "fill") +
  labs(title = "Proportional Barplot for Category") +
  xlab("Category") +
  ylab("Proportion") +
  theme_minimal()
```



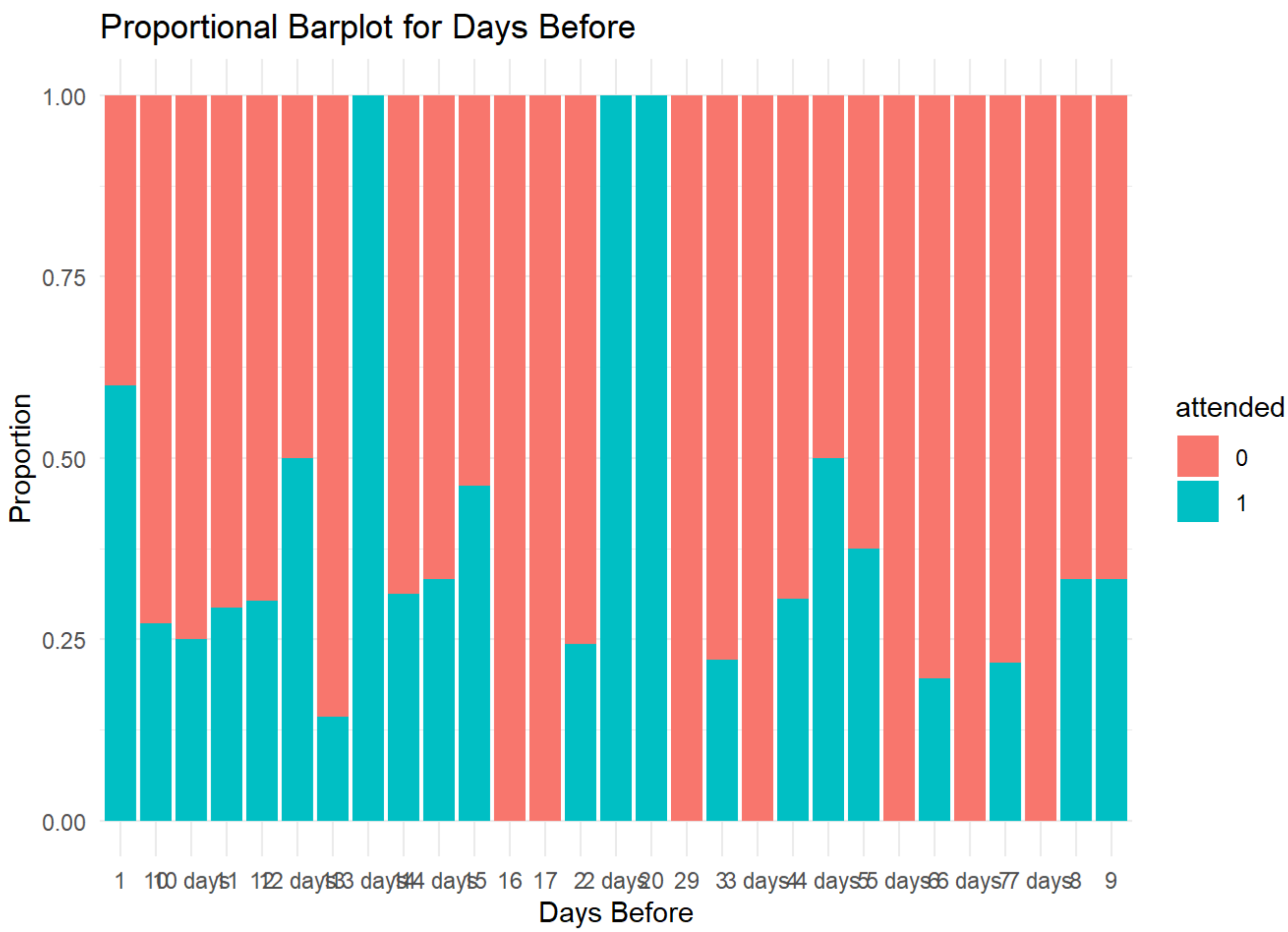

```
ggplot(train_df, aes(x = time, fill = attended)) +
  geom_bar(position = "fill") +
  labs(title = "Proportional Barplot for Time") +
  xlab("Time") +
  ylab("Proportion") +
  theme_minimal()
```




```
ggplot(train_df, aes(x = day_of_week, fill = attended)) +
  geom_bar(position = "fill") +
  labs(title = "Proportional Barplot for Day of Week") +
  xlab("Day of Week") +
  ylab("Proportion") +
  theme_minimal()
```




```
ggplot(train_df, aes(x = days_before, fill = attended)) +
  geom_bar(position = "fill") +
  labs(title = "Proportional Barplot for Days Before") +
  xlab("Days Before") +
  ylab("Proportion") +
  theme_minimal()
```



a.

The variable “time” might not have much predictive power in a naive bayes model.Naive Bayes is a probabilistic machine learning algorithm and the time variable classes (AM and PM) seem to have similar probabilities.

```
train_df <- subset(train_df, select = -time)
valid_df <- subset(valid_df, select = -time)
```

9.

```
library(e1071)
```

```
## Warning: package 'e1071' was built under R version 4.2.3
```

```
fitness.nb <- naiveBayes(attended ~ months_as_member+weight+days_before + category, data = train_df)
fitness.nb
```

```
##
## Naive Bayes Classifier for Discrete Predictors
##
## Call:
## naiveBayes.default(x = X, y = Y, laplace = laplace)
##
## A-priori probabilities:
## Y
##         0         1
## 0.7122222 0.2877778
##
## Conditional probabilities:
##    months_as_member
## Y       [,1]      [,2]
##   0 11.39626  7.081595
##   1 25.51351 17.883853
##
##    weight
## Y       [,1]      [,2]
##   0 84.97540 12.914348
##   1 76.97984  9.537778
##
##    days_before
## Y             1      1 days          10     10 days          11          12
##   0 0.003120125 0.000000000 0.188767551 0.004680187 0.018720749 0.121684867
##   1 0.011583012 0.000000000 0.173745174 0.003861004 0.019305019 0.131274131
##    days_before
## Y       12 days          13     13 days          14     14 days          15
##   0 0.001560062 0.018720749 0.000000000 0.117004680 0.003120125 0.010920437
##   1 0.003861004 0.007722008 0.003861004 0.131274131 0.003861004 0.023166023
##    days_before
## Y            16          17           2      2 days          20          29
##   0 0.004680187 0.003120125 0.131045242 0.000000000 0.000000000 0.001560062
##   1 0.000000000 0.000000000 0.104247104 0.003861004 0.003861004 0.000000000
##    days_before
## Y             3      3 days           4      4 days           5      5 days
##   0 0.021840874 0.001560062 0.102964119 0.001560062 0.015600624 0.001560062
##   1 0.015444015 0.000000000 0.111969112 0.003861004 0.023166023 0.000000000
##    days_before
```

```
## Y            6     6 days          7     7 days          8     8 days
##   0 0.057722309 0.004680187 0.028081123 0.001560062 0.118564743 0.000000000
##   1 0.034749035 0.000000000 0.019305019 0.000000000 0.146718147 0.000000000
##    days_before
## Y            9
##   0 0.015600624
##   1 0.019305019
##
##    category
## Y            -        Aqua     Cycling        HIIT    Strength        Yoga
##   0 0.015600624 0.048361934 0.243369735 0.441497660 0.159126365 0.092043682
##   1 0.007722008 0.046332046 0.212355212 0.498069498 0.146718147 0.088803089
```

10.

```
library(caret)

pred.class <- predict(fitness.nb, newdata = train_df)
confusionMatrix(pred.class, train_df$attended)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 607 146
##          1  34 113
##
##                Accuracy : 0.8
##                  95% CI : (0.7723, 0.8257)
##     No Information Rate : 0.7122
##     P-Value [Acc > NIR] : 1.084e-09
##
##                   Kappa : 0.4399
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.9470
##             Specificity : 0.4363
##          Pos Pred Value : 0.8061
##          Neg Pred Value : 0.7687
##              Prevalence : 0.7122
##          Detection Rate : 0.6744
##    Detection Prevalence : 0.8367
##       Balanced Accuracy : 0.6916
##
##        'Positive' Class : 0
##
```

```
pred.class <- predict(fitness.nb, newdata = valid_df)
confusionMatrix(pred.class, valid_df$attended)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 376 113
##          1  29  82
##
##                Accuracy : 0.7633
##                  95% CI : (0.7272, 0.7968)
##     No Information Rate : 0.675
##     P-Value [Acc > NIR] : 1.307e-06
##
##                   Kappa : 0.3928
##
##  Mcnemar's Test P-Value : 3.279e-12
##
##             Sensitivity : 0.9284
##             Specificity : 0.4205
##          Pos Pred Value : 0.7689
##          Neg Pred Value : 0.7387
##              Prevalence : 0.6750
##          Detection Rate : 0.6267
##    Detection Prevalence : 0.8150
##       Balanced Accuracy : 0.6745
##
##        'Positive' Class : 0
##
```

11. The naive rule in classification is a very simple and baseline approach to classification. It doesn't involve any sophisticated modeling or analysis. Instead, it classifies all records into the most frequent class in the training dataset, regardless of the input features or predictors. The naive rule assumes that the most common class in the training set will also be the most common class for any new data points.

```
table(train_df$attended)
```

```
##
##   0   1
## 641 259
```

Because there are about twice as many unattended classes than attended, with the naive rule I would classify all of them as unattended.

a.

Model Accuracy (training data) = 0.8 Naive Rule Accuracy = 0.7122

Percentage Difference = [(0.8 - 0.7122) / 0.7122] * 100% Percentage Difference = (0.0734 / 0.7122) * 100% Percentage Difference ≈ 12.32%

So, the model's accuracy against training data is approximately 12.32% higher than the naive rule accuracy.

Model Accuracy (validation data) = 0.7633 Naive Rule Accuracy = 0.6750

Percentage Difference = [(0.7633 - 0.6750) / 0.6750] * 100% Percentage Difference = (0.0917 / 0.6750) * 100% Percentage Difference ≈ 13.08%

So, the model's accuracy against validation data is approximately 13.08% higher than the naive rule accuracy.

12.

```
pred.prob <- predict(fitness.nb, newdata = valid_df, type = "raw")
pred.class <- predict(fitness.nb, newdata = valid_df)


df <- data.frame(actual = valid_df$attended, predicted = pred.class, pred.prob)

subset_records <- df[pred.class == 1, ][1:100, ]

missed_actual <- sum(subset_records$actual == 1)

accuracy_subset <- sum(subset_records$actual == subset_records$predicted) / nrow(subset_records)

overall_accuracy <- sum(df$actual == df$predicted) / nrow(df)


missed_actual
```

```
## [1] 74
```

```
accuracy_subset
```

```
## [1] 0.74
```

```
overall_accuracy
```

```
## [1] 0.7633333
```

a. Among the 100 records, 74 people actually missed their class. The accuracy for this subset is 0.74 and the overall accuracy is 0.7633333. The accuracy of the subset is slightly lower.

b. This information can be used to proactively engage with these members and potentially reduce unattendance rates. Fitness Zone can reach out to these members with personalized messages, offers, or incentives to encourage their continued attendance. The gym can offer support and resources to address any specific concerns or challenges that these members may be facing. Implement retention strategies such as reward programs, social engagement events, or goal-setting sessions to keep members motivated and committed to their fitness goals.

13.

The record I picked is - booking_id 1111 months_as_member 18
weight 68.84
days_before 10 day_of_week Fri
category HIIT
months_as_member_binned silver weight_binned very_light attended 0

a. The person did not attend the class they booked.

b.

```
new_record <- data.frame(
  booking_id = 1111,
  months_as_member = 18,
  weight = 68.84,
  days_before = 10,
  day_of_week = "Fri",
  category = "HIIT",
  months_as_member_binned = "silver",
  weight_binned = "very_light"
)

predicted_attendance <- predict(fitness.nb, newdata = new_record, type = "class")
```

```
## Warning in predict.naiveBayes(fitness.nb, newdata = new_record, type =
## "class"): Type mismatch between training and new data for variable
## 'days_before'. Did you use factors with numeric labels for training, and
## numeric values for new data?
```

```
predicted_attendance
```

```
## [1] 0
## Levels: 0 1
```

The model predicted that the person will not attend the class. The prediction is correct.

c.

```
predicted_probabilities <- predict(fitness.nb, newdata = new_record, type = "raw")
```

```
## Warning in predict.naiveBayes(fitness.nb, newdata = new_record, type = "raw"):
## Type mismatch between training and new data for variable 'days_before'. Did you
## use factors with numeric labels for training, and numeric values for new data?
```

```
probability_of_attendance <- predicted_probabilities[, "1"]

probability_of_attendance
```

```
##         1
## 0.3438775
```

The probability that my person will attend the class is 0.3438775 . Which is almost 35%.

d. P(Y = 0 | X) = P(Y = 0) * P(months_as_member = 18 | Y = 0) * P(weight = 68.84 | Y = 0) * P(days_before = 10 | Y = 0) * P(category = “HIIT” | Y = 0)

P(Y = 0 | X)= 0.7122222* 11.39626 * 84.97540* 0.188767551 * 0.441497660

= 57.22

P(Y = 1 | X) = P(Y = 1) * P(months_as_member = 18 | Y = 1) * P(weight = 68.84 | Y = 1) * P(days_before = 10 | Y = 1) * P(category = “HIIT” | Y = 1)

$P(Y = 1 \mid X) = 0.2877778 * 25.51351 * 76.97984 * 0.173745174 * 0.498069498 = 48.86$

$P(\text{Attendance} = 1 \mid X) = P(Y = 1 \mid X) / [P(Y = 0 \mid X) + P(Y = 1 \mid X)]$

$= 48.86 / (57.22 + 48.86)$

$= 0.34$