Multivariate Analysis Report

(MECD, Minor-CD, & MMAC, 1st Semester, 2023/2024)

2023-12-05

Professor: M. Rosário Oliveira

Group Members:

Antonio Villas Bôas	Darlan Nascimento	Emma Dennis-Knieriem	Leonardo Aikawa	Tunahan Güneş
ist1105429	ist1108076	ist1105545	ist1105545	ist1108108

Fitness Club Dataset for ML Classification

Introduction

Multivariate analysis is a pivotal statistical technique in the realm of data science, particularly when dealing with datasets encompassing multiple interrelated variables. It offers a sophisticated approach to understanding the nuanced relationships and patterns that emerge from complex data. This method is crucial for extracting meaningful insights from datasets where multiple factors interact in intricate ways, as is often the case in real-world scenarios.

Multivariate analysis includes a variety of techniques, each suited to different types of data and analysis objectives. Key methods such as logistic regression, decision trees, and cluster analysis are particularly relevant in scenarios where predicting outcomes or classifying data points are the main goals. These techniques allow for a comprehensive analysis, considering how various factors collectively influence outcomes [1].

Description of the problem under study

Our dataset was select from Kaggle. The dataset provided for this analysis originates from GoalZone, a well-known fitness club chain in Canada. GoalZone operates several fitness classes, with capacities of either 25 or 15 participants. A significant challenge they face is the discrepancy between class bookings and actual attendance. While some classes are consistently fully booked, the actual attendance rate often falls short of expectations. This scenario presents both a logistical challenge and an opportunity for optimization.

The primary challenge is predicting whether a member who has booked a class will actually attend. Accurately forecasting attendance allows GoalZone to optimize class capacities effectively, ensuring maximum utilization of resources and enhanced member satisfaction. This challenge involves analyzing various factors that might influence attendance, such as the time of the class, member demographics, previous attendance patterns, and class type.

Objectives

The analysis of the GoalZone dataset aims to achieve the following objectives:

Predictive Analysis: Develop a predictive model that can accurately forecast whether a member who has booked a class will attend. This involves identifying key variables that influence attendance and employing appropriate multivariate techniques. Optimization of Class Capacities: Use the insights gained from the predictive model to optimize class capacities, enabling GoalZone to allocate spaces more effectively and potentially increase the total number of spaces offered. Enhancing Member Experience: By improving the accuracy of attendance predictions, GoalZone can reduce overbooking and under-utilization issues, leading to a better overall experience for members. Data-Driven Decision Making: Provide GoalZone with data-driven strategies to manage class schedules and capacities, thereby improving operational efficiency.

You can also embed plots, for example:

```
fitdata <- read_csv("fitness_class_2212.csv", show_col_types = FALSE)
# str(fitdata)
summary(fitdata)
```

```
##   booking_id        months_as_member     weight       days_before
##  Length:1500        Min.   :  1.00   Min.   : 55.41   Length:1500
##  Class :character   1st Qu.:  8.00   1st Qu.: 73.49   Class :character
##  Mode  :character   Median : 12.00   Median : 80.76   Mode  :character
##                     Mean   : 15.63   Mean   : 82.61
##                     3rd Qu.: 19.00   3rd Qu.: 89.52
##                     Max.   :148.00   Max.   :170.52
##                                      NA's   :20
```

```
##  day_of_week            time             category           attended
##  Length:1500        Length:1500        Length:1500        Min.   :0.0000
##  Class :character   Class :character   Class :character   1st Qu.:0.0000
##  Mode  :character   Mode  :character   Mode  :character   Median :0.0000
##                                                           Mean   :0.3027
##                                                           3rd Qu.:1.0000
##                                                           Max.   :1.0000
##
```

```
sapply(fitdata, function(x) sum(is.na(x))) # check each column for NA values
```

```
##       booking_id months_as_member           weight      days_before
##                0                0               20                0
##      day_of_week             time         category         attended
##                0                0                0                0
```

Data Exploration and Cleaning

We noticed that some columns are not in the ideal format to work with our classification models. The column `days_before` should be a discrete (integer) number of days before the class the member registered, but in `fitdata` we got `class(fitdata$days_before)` as character. Here are the steps we used to convert to integer:

```
unique(fitdata$days_before)
```

```
##  [1] "8"       "2"       "14"      "10"      "6"       "4"       "9"
##  [8] "12"      "5"       "3"       "7"       "13"      "12 days" "20"
## [15] "1"       "15"      "6 days"  "11"      "13 days" "3 days"  "16"
## [22] "1 days"  "7 days"  "8 days"  "10 days" "14 days" "17"      "5 days"
## [29] "2 days"  "4 days"  "29"
```

```
fitdata$days_before <- fitdata$days_before %>%
     strsplit(split = " ") %>% # split some strings that had more than the number of days
     sapply(function(x) x[1]) %>% # gets the 1st element of the split strings
     as.integer() # converts to integer

class(fitdata$days_before)
```

```
## [1] "integer"
```

For `day_of_week`, we needed to standardize to uniquely 7 values corresponding to the days of the week.

```
unique(fitdata$day_of_week)
```

```
##  [1] "Wed"       "Mon"       "Sun"       "Fri"       "Thu"       "Wednesday"
##  [7] "Fri."      "Tue"       "Sat"       "Monday"
```

```r
fitdata$day_of_week <- fitdata$day_of_week %>%
     tolower() %>% # converts to lower case
     substr(1, 3) %>% # gets only the 3 first elements of each string
     factor() # Converts to factor to be used as categorical variable

table(fitdata$day_of_week)
```

```
##
## fri mon sat sun thu tue wed
## 305 228 202 213 241 195 116
```

`fitdata$time` is a ordinal variable which indicates if the booked fitness class is in the morning (AM) or afternoon (PM). This is also categorical so we encoded it to numerical representation ("AM" / "PM" to 0 / 1).

```r
table(fitdata$time)
```

```
##
##   AM   PM
## 1141  359
```

```r
fitdata$time <- ifelse(fitdata$time == "AM", 0, 1) %>%
     factor()

table(fitdata$time)
```

```
##
##    0    1
## 1141  359
```

`fitdata$category` correspond to the category of the fitness class. In this dataset only 6 categories are present and one of them is "-".

```r
table(fitdata$category)
```

```
##
##        -     Aqua  Cycling     HIIT Strength     Yoga
##       13       76      376      667      233      135
```

```r
fitdata <- fitdata %>%
     mutate(category = na_if(category, "-")) # Replace "-" with NA

fitdata$category <- factor(fitdata$category)
table(fitdata$category)
```

```
##
##     Aqua  Cycling     HIIT Strength     Yoga
##       76      376      667      233      135
```

Dealing with missing values is crucial in data analysis to ensure the integrity and validity of the results. Missing data can introduce bias, reduce the statistical power, and lead to invalid conclusions. By appropriately addressing these gaps, whether through imputation, removal, or analysis modifications, we can enhance the robustness of our findings and make more accurate inferences from the data. This process is essential for maintaining the quality and reliability of statistical analysis in any research or data-driven decision-making [2].

```
get_na <- function(df) { # get indices of NA values in each column
     na_rows <- lapply(df, function(x) which(is.na(x))) %>%
          unlist() %>% unique() # get the unique indices of all columns
     return(na_rows)
}
na_rows <- get_na(fitdata)
na_ratio <- length(na_rows) / dim(fitdata)[1]

fitdata <- fitdata[-na_rows, ]
```

For `na_ratio` = 0.022 the removal of missing values (NAs) is acceptable since their proportion is very small, as it minimally affects the dataset's overall integrity and distribution.

Estimation and validation methods

1. K-means clustering

The K-means clustering algorithm, crucial in high-dimensional data analysis, is detailed in classics like John Hartigan's "Clustering Algorithms." This iterative, algorithmic method identifies centroids of pre-specified clusters in multi-dimensional spaces, a task challenging in higher dimensions. K-means partitions data into clusters, requiring an initial guess of cluster numbers. Adjusting this number in subsequent runs helps in fine-tuning the clustering outcome, essential for analyzing complex data structures. [3]

As an unsupervised learning algorithm, K-means does not work with a target variable (in our case, `fitdata$attended`), but it can segment data into clusters, which can then potentially be analyzed to infer patterns that might be relevant for prediction.

Two unsupervised learning techniques were used to compare the results: k-means clustering and agglomerative nesting (Hierarchical Clustering). Although is not directly correlated, we used 2 centroids to try to see if each model would classify the fitness club attendance into 0 or 1 and then compare with `fitdata$attended`.

The metric used to compare models was a simple proportion of matching attendance and cluster assignments given by:

$$\frac{\sum(\text{fitdata.attended} == \text{prediction})}{\text{length(fitdataattended)}}$$

For the agglomerative nesting (agnes) we tried an algorithm that tested every combination of `metrics` and `methods` such:

```
metrics <- c('euclidean', 'manhattan')
methods <- c('single', 'complete', 'average')
```

The best AgNes combination was: {metric: manhattan, method: complete}. And the accuracy results are shown below:

Model	Accuracy
K-means	0.49488
AgNes	0.72392

The image below shows plots of the actual fitness club attendance, the attendance prediction of the K-means models, and the attendance of the best AgNes model, respectively. For more details about the algorithm used to create the models and plots, please check the `kmeans.R` file.

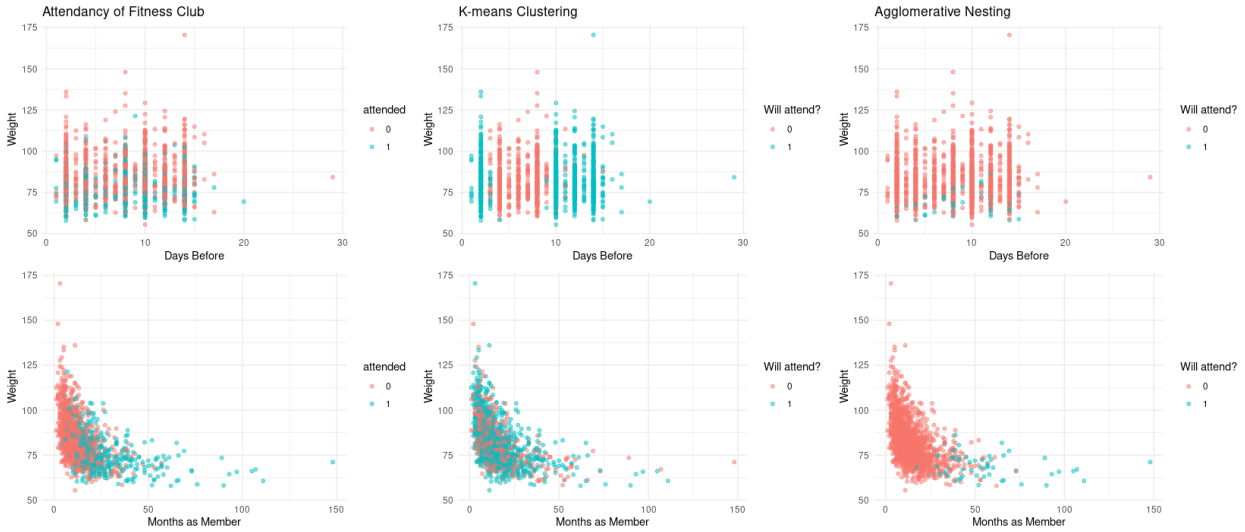


Figure 1: Attendance Comparison

According to the plots above, there doesn't appear to be a clear boundary separating the attended and not-attended groups in either plot, suggesting that these features alone may not strongly predict attendance.

Splitting The Data

To train and test our supervised models, we applied a 80-20 split for our data. And, all the models we applied are then trained and tested on the same sets such that:

```
library(caret)
```

```
## Loading required package: ggplot2
```

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

```
library(rpart)

set.seed(42) #Setting the seed for reproducibility

# Specify the proportion of the test data
test_proportion <- 0.2

# Use createDataPartition to create indices for the training set randomly
train_indices <- createDataPartition(fitdata$attended,p = 1 - test_proportion, list = FALSE)
```

```r
#Creating Stratified Training And Test Datasets
train_data <- fitdata[train_indices, ]
test_data <- fitdata[-train_indices, ]
```

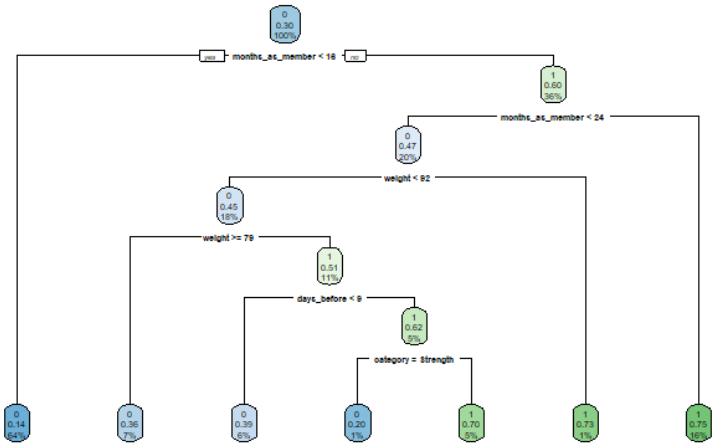
2. Logistic Regression

3. Decision Tree

The Decision Tree is a supervised classification model known for its simplicity and interpretability. It handles categorical variables well and is not affected much by irrelevant features. However, it can overfit and is sensitive to training data variability. For our categorical-heavy data, a decision tree could be advantageous. To avoid overfitting, we can limit tree growth by setting constraints or prune the tree after full growth, though care must be taken to prevent underfitting with too much restriction.

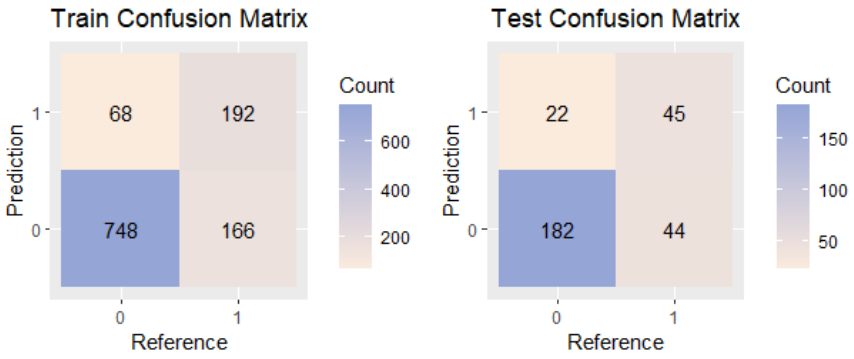
In this project, both of the approaches are applied and tested with a large number of different parameters. At their best settings of both approaches by looking at the test data performance, the resulting trees were identical. Therefore, we kept the model with the first approach with minsplit=20, and minbucket=7 for simplicity:

```r
dt_model <- rpart(attended ~ ., data = train_data, method = "class", minsplit=20, minbucket=7)
```



As it can be seen by the tree our target variable is too dependent on the “months_before” and “weight” features. So that, majority of our data can be classified by just splitting the “months_before” feature.

The following results and confusion matrices are gathered by this tree:



Data	Accuracy	Balanced Accuracy	F-1 Score
Train	0.8007	0.7265	0.8647399
Test	0.7747	0.6989	0.8465116

Although the accuracies are relatively better, having significantly lower balanced accuracies is worrisome. This means that the model is not effectively dealing the class imbalance. Even with experiements with different settings, and models such as SVMs, Neural Networks we were not able to get significantly better results. Therefore, we can conclude that the classes in our data are not easily seperable, and in order to have better classification, it is required to have more features that can explain our classes better.

Discussion of the results and interpretation of the findings

Conclusions

References

[1] R. A. Johnson and D. W. Wichern, *Applied multivariate statistical analysis*, 6th ed. New Jersey: Prentice-Hall, 2007.

[2] R. J. A. Little and D. B. Rubin, *Statistical analysis with missing data*. John Wiley & Sons, 2002.

[3] R. D. Peng, *Exploratory data analysis with r*. lulu.com, 2016. Available: https://bookdown.org/rdpeng/exdata/