Predictive Modeling to Optimize Class Capacity at GoalZone Fitness

Yan Naing Oo

2023-07-05

Load Packages

```r
library(readr)
library(ggplot2)
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```r
library(caret)
```

```
## Warning: package 'caret' was built under R version 4.3.1
```

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

```r
library(randomForest)
```

```
## Warning: package 'randomForest' was built under R version 4.3.1
```

```
## randomForest 4.7-1.1
```

```
## Type rfNews() to see new features/changes/bug fixes.
```

```
##
## Attaching package: 'randomForest'
## The following object is masked from 'package:dplyr':
##
##     combine

## The following object is masked from 'package:ggplot2':
##
##     margin
```

```
library(pROC)
```

```
## Warning: package 'pROC' was built under R version 4.3.1

## Type 'citation("pROC")' for a citation.

##
## Attaching package: 'pROC'

## The following objects are masked from 'package:stats':
##
##     cov, smooth, var
```

Load data from fitness_class_2212.csv file

```
df_fc <- read_csv("fitness_class_2212.csv", show_col_types = FALSE)

head(df_fc)
```

```
## # A tibble: 6 x 8
##   booking_id months_as_member weight days_before day_of_week time  category
##   <chr>                 <dbl>  <dbl> <chr>       <chr>       <chr> <chr>
## 1 0001                     17   79.6 8           Wed         PM    Strength
## 2 0002                     10   79.0 2           Mon         AM    HIIT
## 3 0003                     16   74.5 14          Sun         AM    Strength
## 4 0004                      5   86.1 10          Fri         AM    Cycling
## 5 0005                     15   69.3 8           Thu         AM    HIIT
## 6 0006                      7   93.3 2           Mon         AM    Cycling
## # i 1 more variable: attended <dbl>
```

Data Preprocessing

```
colSums(is.na(df_fc))
```

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

```r
# check the number of missing values
sum(is.na(df_fc$months_as_member)) # 0 missing data
```

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

```r
sum(is.na(df_fc$weight)) # 20 missing data
```

```
## [1] 20
```

```r
sum(is.na(df_fc$days_before)) # 0 missing data
```

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

```r
sum(is.na(df_fc$day_of_week)) # 0 missing data
```

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

```r
sum(is.na(df_fc$time)) # 0 missing data
```

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

```r
sum(is.na(df_fc$category)) # 0 missing data
```

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

```r
sum(is.na(df_fc$attended)) # 0 missing data
```

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

```r
# Replace missing values with the overall average weight
df_fc$weight[is.na(df_fc$weight)] <- mean(df_fc$weight, na.rm = TRUE)
sum(is.na(df_fc$weight)) # 0 missing data
```

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

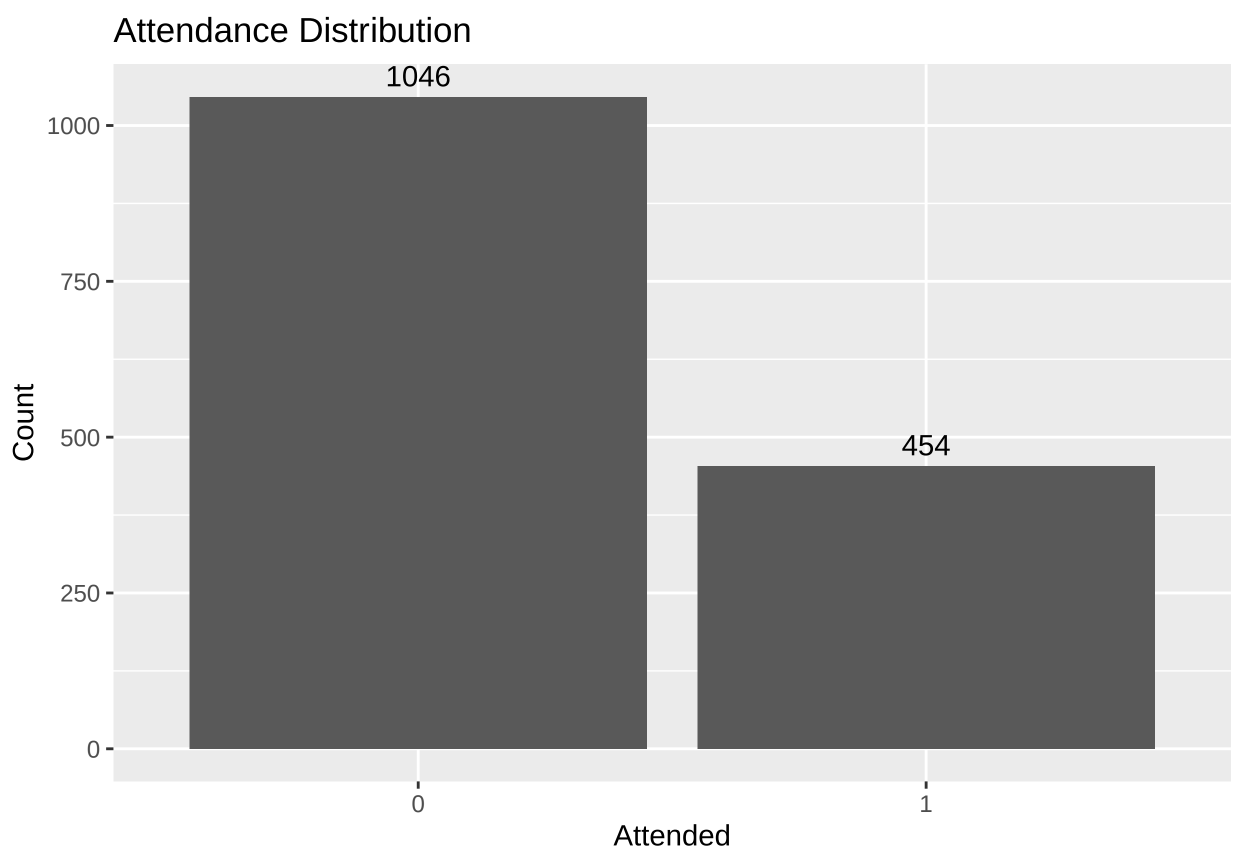
```r
# Remove 'days' from the values and convert the column into numeric
df_fc$days_before <- as.numeric(gsub(" days", "", df_fc$days_before))

# Replace hyphens with "unknown" in the 'category' column
df_fc$category <- gsub("-", "unknown", df_fc$category)

# Create a mapping dictionary
day_mapping <- c("Mon" = "Mon",
                 "Monday" = "Mon",
                 "Tue" = "Tue",
                 "Wed" = "Wed",
                 "Wednesday" = "Wed",
                 "Thu" = "Thu",
                 "Fri" = "Fri",
                 "Fri." = "Fri",
                 "Sat" = "Sat",
                 "Sun" = "Sun")

# Use the mapping dictionary to replace the values in the 'day_of_week' column
df_fc$day_of_week <- day_mapping[as.character(df_fc$day_of_week)]


head(df_fc ,10)
```

```
## # A tibble: 10 x 8
##    booking_id months_as_member weight days_before day_of_week time  category
##    <chr>                 <dbl>  <dbl>       <dbl> <chr>       <chr> <chr>
##  1 0001                     17   79.6           8 Wed         PM    Strength
##  2 0002                     10   79.0           2 Mon         AM    HIIT
##  3 0003                     16   74.5          14 Sun         AM    Strength
##  4 0004                      5   86.1          10 Fri         AM    Cycling
##  5 0005                     15   69.3           8 Thu         AM    HIIT
##  6 0006                      7   93.3           2 Mon         AM    Cycling
##  7 0007                     11   88.6           6 Wed         PM    HIIT
##  8 0008                      9   89.5          10 Fri         AM    HIIT
##  9 0009                     23   71.1          10 Fri         AM    HIIT
## 10 0010                      7   81.2          10 Fri         AM    HIIT
## # i 1 more variable: attended <dbl>
```

Descriptive Analysis

```
# T2
# Create bar plot
ggplot(df_fc, aes(x = factor(attended))) +
  geom_bar() +
  geom_text(stat = "count", aes(label = ..count..), vjust = -0.5) +
  labs(x = "Attended", y = "Count", title = "Attendance Distribution")
```

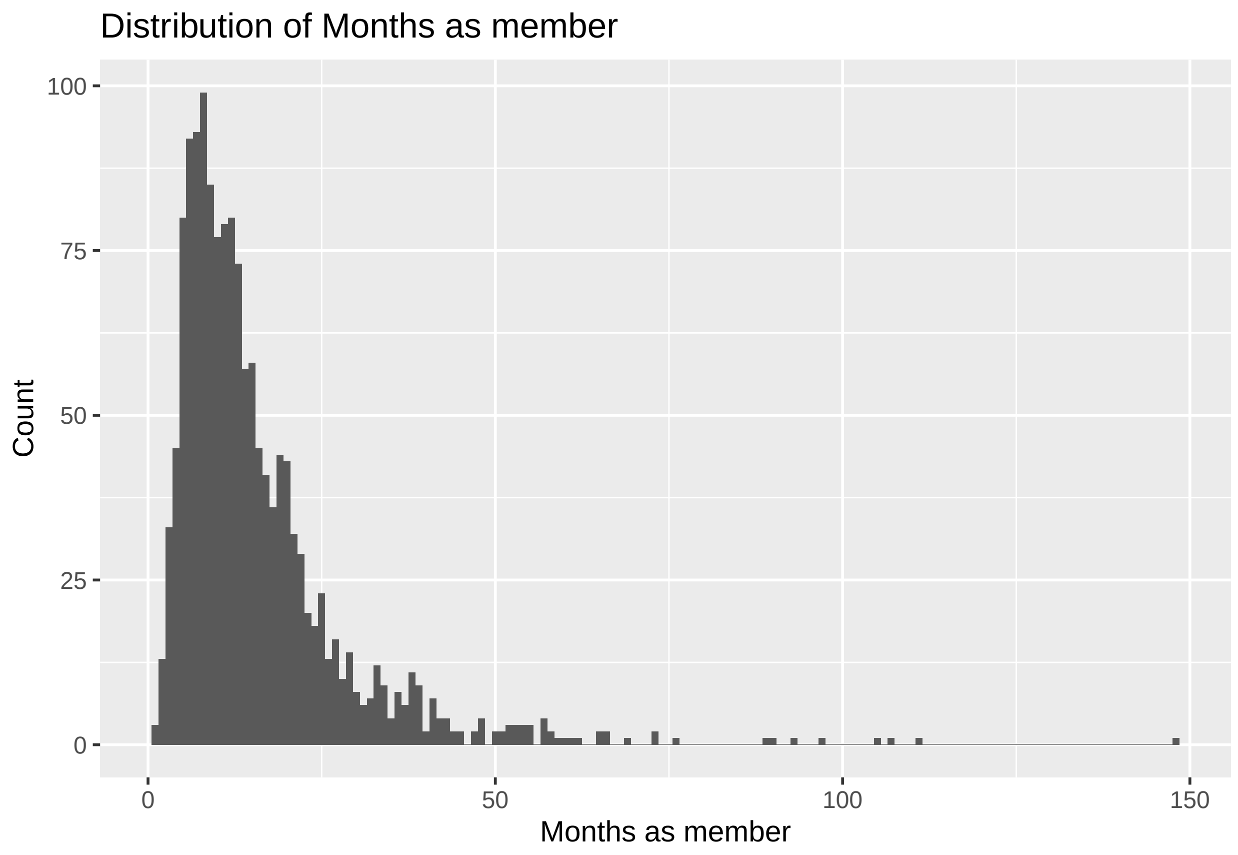
```
## Warning: The dot-dot notation (‘..count..‘) was deprecated in ggplot2 3.4.0.
## i Please use ‘after_stat(count)‘ instead.
## This warning is displayed once every 8 hours.
## Call ‘lifecycle::last_lifecycle_warnings()‘ to see where this warning was
## generated.
```



"0" indicates that a member did not attend the class that they booked. "1" indicates that a member attended the class that they booked.

The bar plot appears that more people book classes but don't attend (Attended = 0) compared to those who book and do attend (Attended = 1).

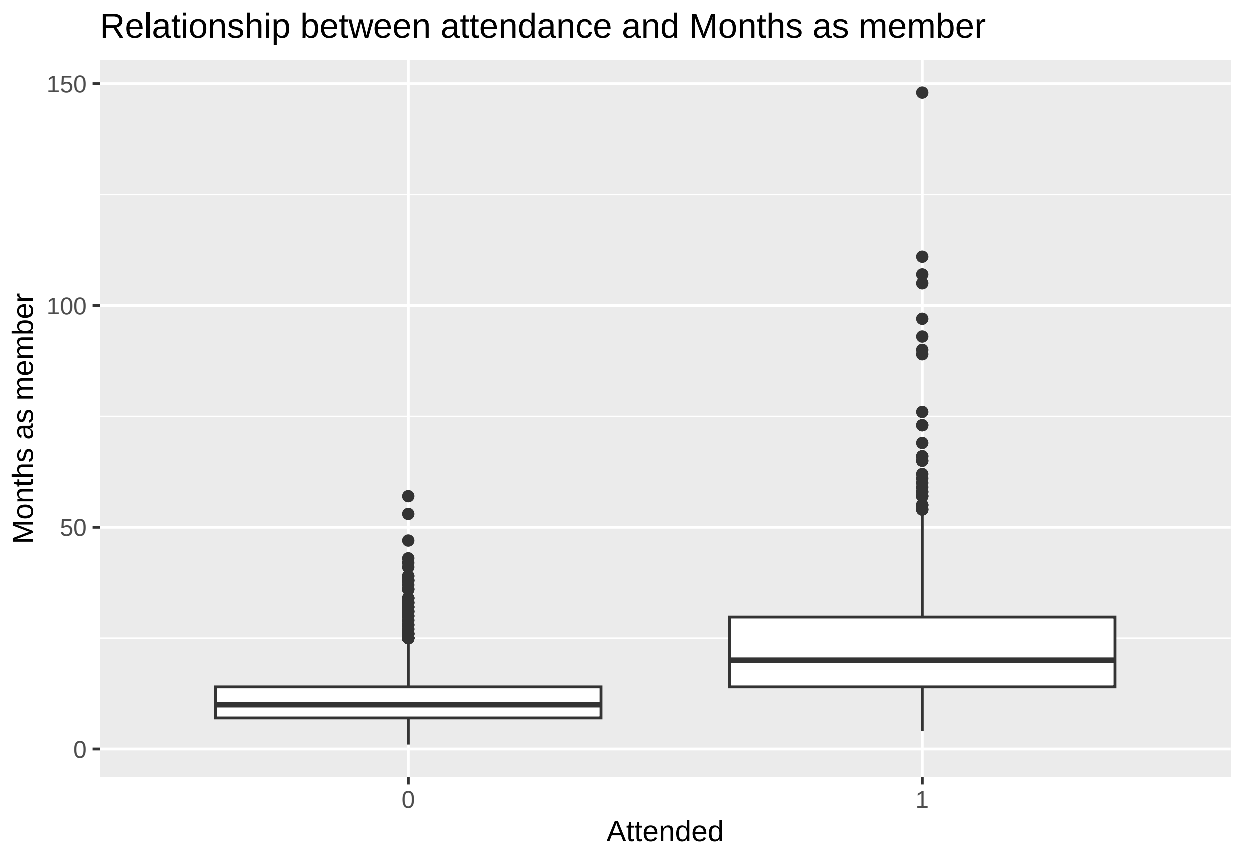
```r
# T3
# Create the histogram
ggplot(df_fc, aes(x=months_as_member)) + geom_histogram(binwidth=1) +
  labs(x = "Months as member", y = "Count", title = "Distribution of Months as member")
```



it appears that the histogram for the "Months as member" variable has a unimodal distribution with the mode (highest frequency) at around the 20 months.

This suggests that the majority of the members have been a part of the fitness club for approximately 20 months, after which the number of members begins to decline.

```r
# T4
# Create the box plot
ggplot(df_fc, aes(x=factor(attended), y=months_as_member)) + geom_boxplot() +
  labs(x = "Attended", y = "Months as member", title = "Relationship between attendance and Months as me
```



The outliers for Attended(0) range from 25 to 50 months, while those for Attended(1) range from 50 to 150 months.

This suggests that members who have been a part of the club for a longer time (over 50 months) are more likely to attend their booked classes.

Data Transformation

Encoding categorical variables

```
# Ensure the attended variable is a factor
df_fc$attended <- as.factor(df_fc$attended)

# For nominal variables
df_fc$booking_id <- as.factor(df_fc$booking_id)
df_fc$category <- as.factor(df_fc$category)

# For ordinal variables
df_fc$day_of_week <- ordered(df_fc$day_of_week, levels = c('Mon', 'Tue', 'Wed', 'Thu', 'Fri', 'Sat', 'Su
df_fc$time <- ordered(df_fc$time, levels = c('AM', 'PM'))
```

Split train and test set

```r
# Set seed for reproducibility
set.seed(123)

# Create index for the split
train_index <- createDataPartition(df_fc$attended, p = 0.7, list = FALSE)

# training and test sets
train_set <- df_fc[train_index,] # 70%
test_set <- df_fc[-train_index,] # 30%
```

Model Fitting

```r
# Exclude the 'booking_id' as it's an identifier and not a predictor
train_set$booking_id <- NULL
test_set$booking_id <- NULL
```

```r
# Fit the model
lr_model <- glm(attended ~ ., data = train_set, family = binomial)

# Predict on the test set
pred_lr <- predict(lr_model, newdata = test_set, type = "response")

# Convert probabilities to binary prediction
pred_lr_binary <- ifelse(pred_lr > 0.5, 1, 0)
```

Fit Logistic Regression for (Baseline) Model

```r
# Fit the model
rf_model <- randomForest(attended ~ ., data = train_set, ntree = 100)

# Predict on the test set
pred_rf <- predict(rf_model, newdata = test_set)
```

Fit Random Forest for (Comparison) Model Compared two models: Logistic Regression (baseline model) and Random Forest (comparison model). Here's an explanation of why these models were chosen

`Logistic Regression (Baseline Model):`

- Chosen for its interpretability and simplicity.
- Models the relationship between predictors and the probability of the binary outcome.
- Assumes a linear relationship between predictors and log-odds.

- Provides coefficients indicating the impact of predictors on the outcome.

`Random Forest (Comparison Model):`

- its ability to handle complex relationships and interactions.
- An ensemble method that combines multiple decision trees.
- Captures non-linear patterns and complex feature interactions.
- Robust against overfitting, performs well with default settings.
- Generates feature importance measures.

Performance Evaluation

Compare two model performance

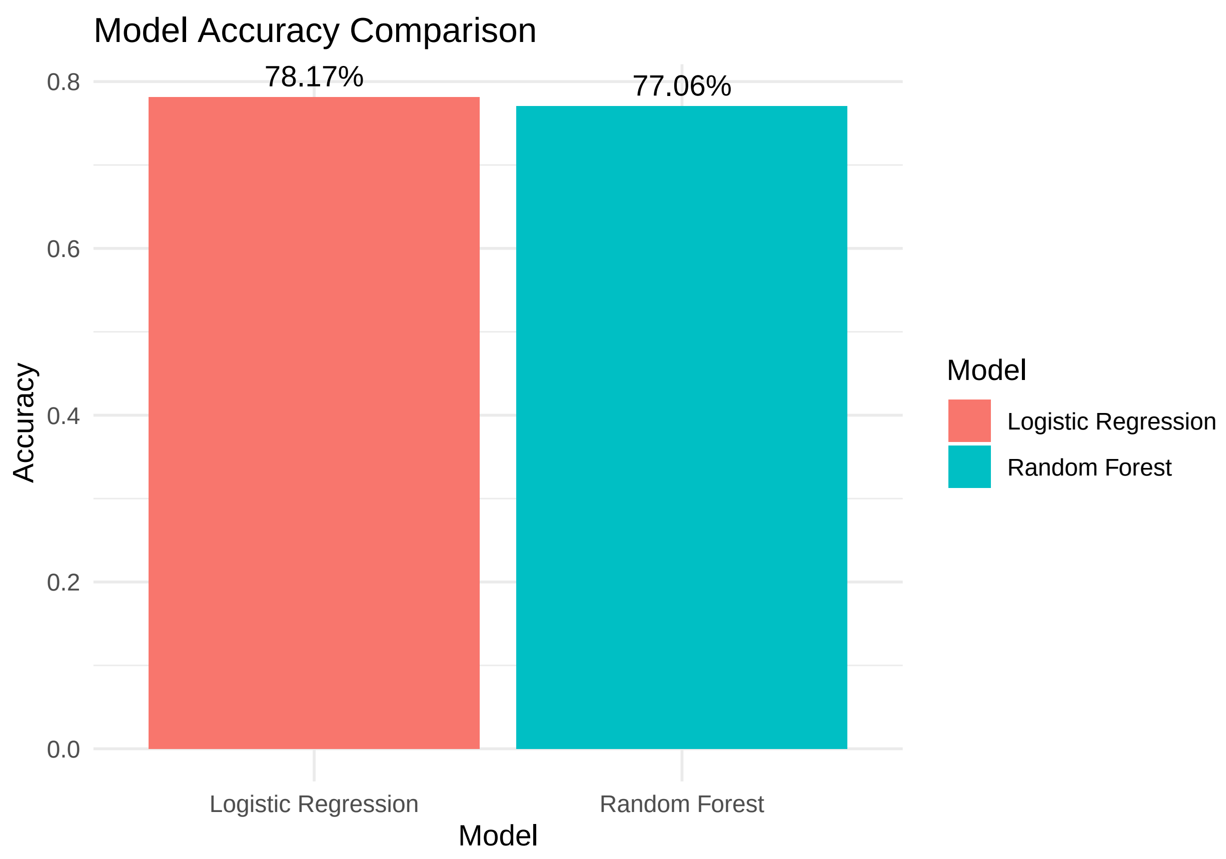
```r
# Compare actual vs. predicted for both models
lr_perf <- table(test_set$attended, pred_lr_binary)
rf_perf <- table(test_set$attended, pred_rf)

# Calculate accuracy for both models
lr_accuracy <- sum(diag(lr_perf)) / sum(lr_perf)
rf_accuracy <- sum(diag(rf_perf)) / sum(rf_perf)

# Print the performance metrics
cat("Logistic Regression Accuracy:", lr_accuracy, "\n")
```

```
## Logistic Regression Accuracy: 0.7817372
```

```r
cat("Random Forest Accuracy:", rf_accuracy, "\n")
```

```
## Random Forest Accuracy: 0.7706013
```

```r
# Create a data frame for performance evaluation
perf_data <- data.frame(Model = c("Logistic Regression", "Random Forest"),
                        Accuracy = c(lr_accuracy, rf_accuracy))

# Create the bar plot
ggplot(perf_data, aes(x = Model, y = Accuracy, fill = Model, label = paste0(round(Accuracy * 100, 2), "
  geom_col(position = "identity") +
  geom_text(position = position_dodge(width = 0.9), vjust = -0.5) +
  labs(x = "Model", y = "Accuracy", title = "Model Accuracy Comparison") +
  theme_minimal()
```



Model Evaluation Metrics

```r
# Convert variables to numeric
test_set$attended <- as.numeric(as.character(test_set$attended))
pred_lr <- as.numeric(as.character(pred_lr))
# Convert variables to numeric
test_set$attended <- as.numeric(as.character(test_set$attended))
pred_rf <- as.numeric(as.character(pred_rf))


# Calculate RMSE for logistic regression
rmse_lr <- sqrt(mean((test_set$attended - pred_lr)^2))
# Calculate RMSE for random forest
rmse_rf <- sqrt(mean((test_set$attended - pred_rf)^2))
# Create a data frame for performance evaluation
perf_data <- data.frame(Model = c("Logistic Regression", "Random Forest"),
                        Accuracy = c(lr_accuracy, rf_accuracy),
                        RMSE = c(rmse_lr, rmse_rf))

# Print the performance metrics
print(perf_data)
```

```
##                 Model  Accuracy      RMSE
## 1 Logistic Regression 0.7817372 0.3864716
## 2       Random Forest 0.7706013 0.4789558
```

the logistic regression model performs better than the random forest model in predicting attendance for fitness classes. The best model performance is based on the higher accuracy rate and lower RMSE value achieved by the logistic regression model.

The higher accuracy indicates that the logistic regression model makes more correct predictions, while the lower RMSE suggests that its predictions are closer to the actual values. Overall, the logistic regression model provides better performance in predicting attendance compared to the random forest model.

```
# For logistic regression
pred_prob_lr <- predict(lr_model, test_set, type = "response")
# For random forest
pred_prob_rf <- predict(rf_model, test_set, type = "prob")[, "1"]

roc_lr <- roc(test_set$attended, pred_prob_lr)
```

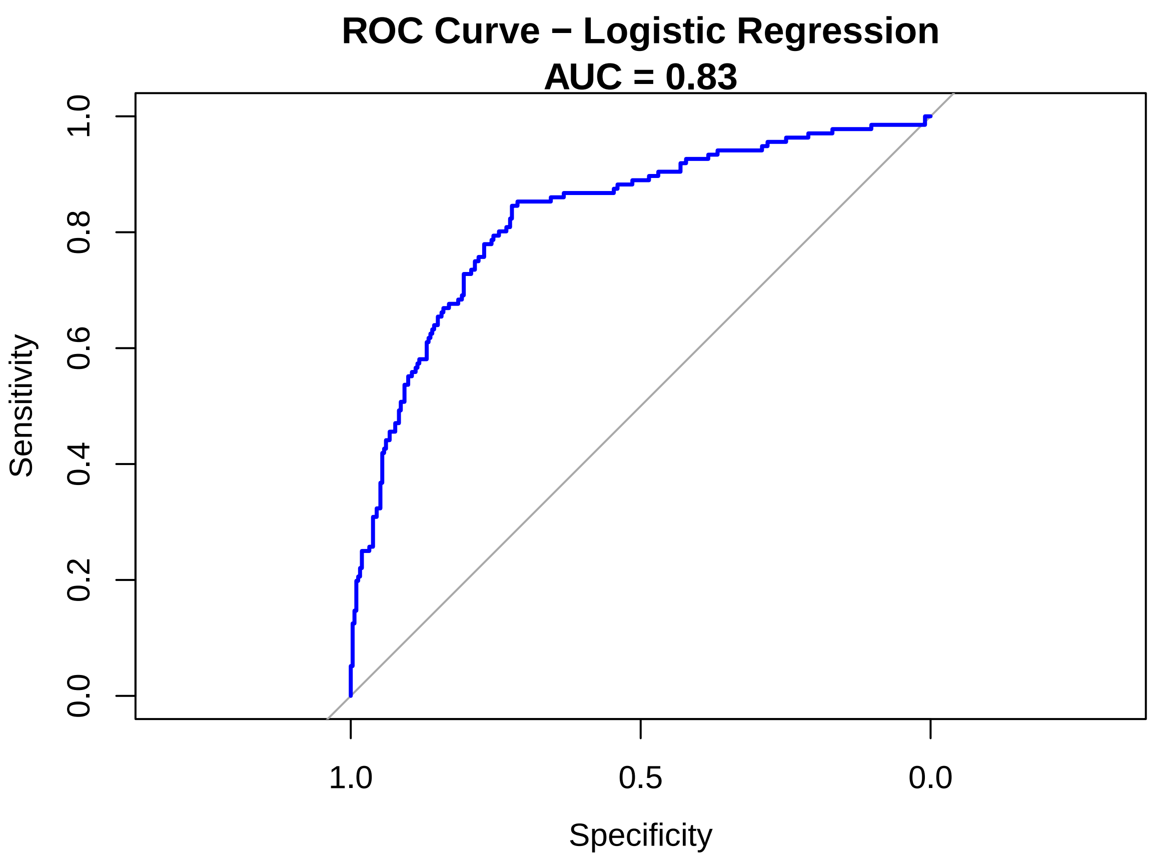
```
## Setting levels: control = 0, case = 1
```

```
## Setting direction: controls < cases
```

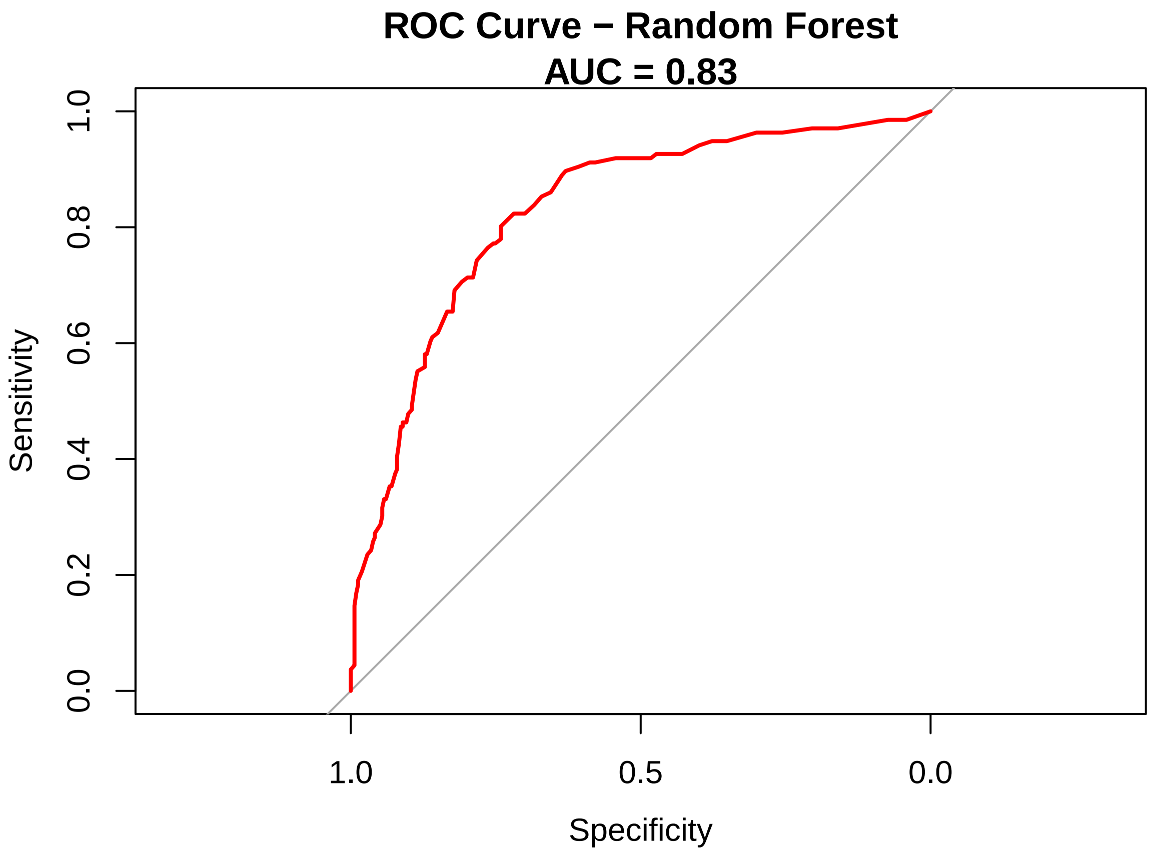
```
roc_rf <- roc(test_set$attended, pred_prob_rf)
```

```
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
```

```
auc_lr <- auc(roc_lr)
auc_rf <- auc(roc_rf)

# Plot the ROC curve
plot(roc_lr, main = paste("ROC Curve - Logistic Regression\nAUC =", round(auc_lr, 2)),col = "blue")
```



```
plot(roc_rf, main = paste("ROC Curve - Random Forest\nAUC =", round(auc_rf, 2)), col = "red")
```



In general, an AUC value of 0.8 or higher is considered to be indicative of a good model performance. Therefore, both the Random Forest and Logistic Regression models can be considered reliable in predicting attendance based on the fitness class dataset.

Business Implications

According to the prediction outcomes, there are some business implications:

- **Increased revenue:** By predicting which members are most likely to attend a class, GoalZone can allocate more space to those members. This will lead to more people attending classes, which will increase revenue.
- **Improved customer experience:** By ensuring that members can get a spot in the classes they want, GoalZone can improve the customer experience. This will help to reduce frustration and keep members happy.
- **Better allocation of resources:** By predicting which classes are likely to be most popular, GoalZone can ensure they have enough staff and equipment to meet demand. This will help improve the customer experience and avoid turning people away.
- **Reduced no-shows:** By predicting which members are most likely to attend a class, GoalZone can avoid having to offer refunds to members who do not attend. This will help to reduce costs and improve the bottom line.

By use of a machine learning algorithm to predict whether or not a member will attend a class can have many positive business implications for GoalZone. By increasing revenue, improving the customer experience, and better allocating resources, GoalZone can use machine learning to improve its business.