Stats_5400_final_project

AUTHOR
Jayesh_Shubham

---

loading the data

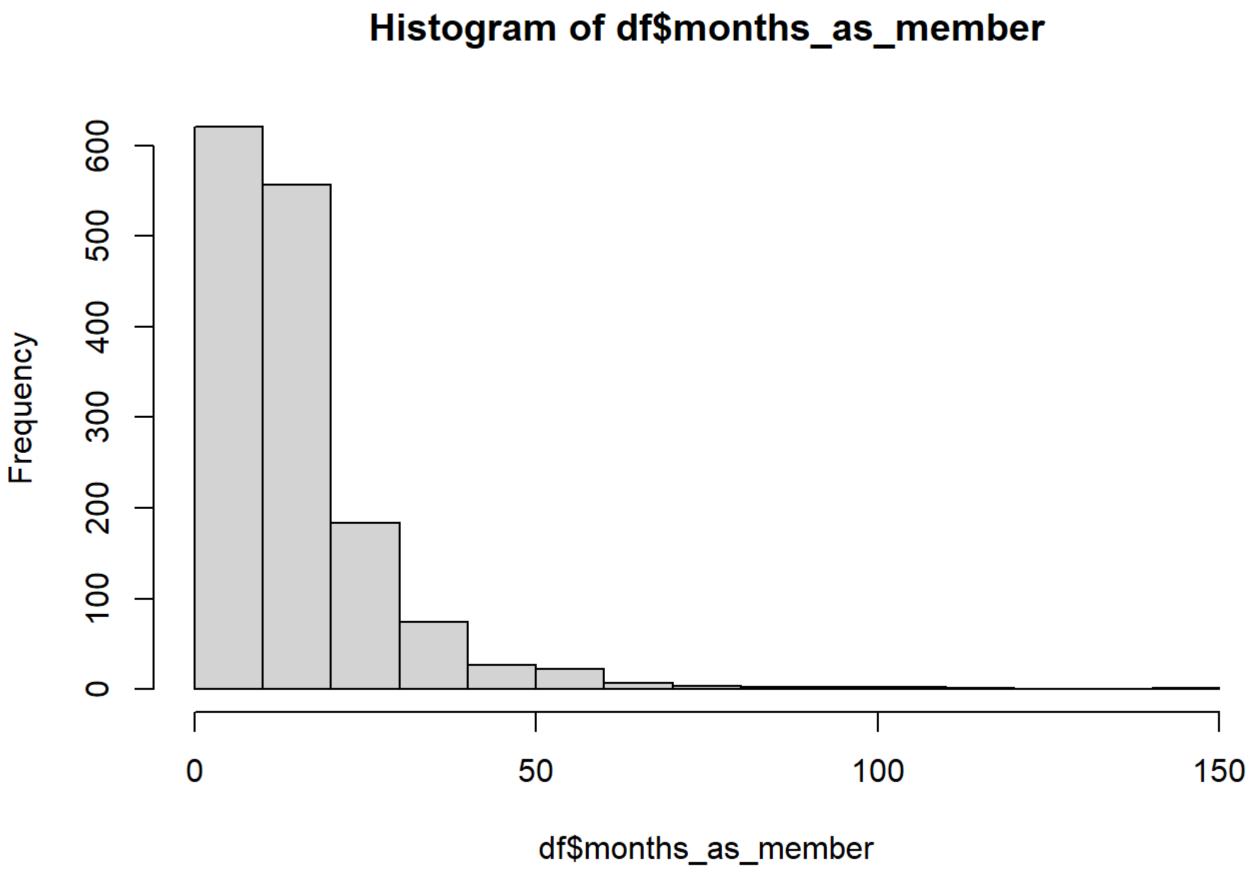
```r
df <- read.csv("C:/STAT 5405/Stats_final_project/fitness_class_2212.csv")
```

```r
str(df)
```

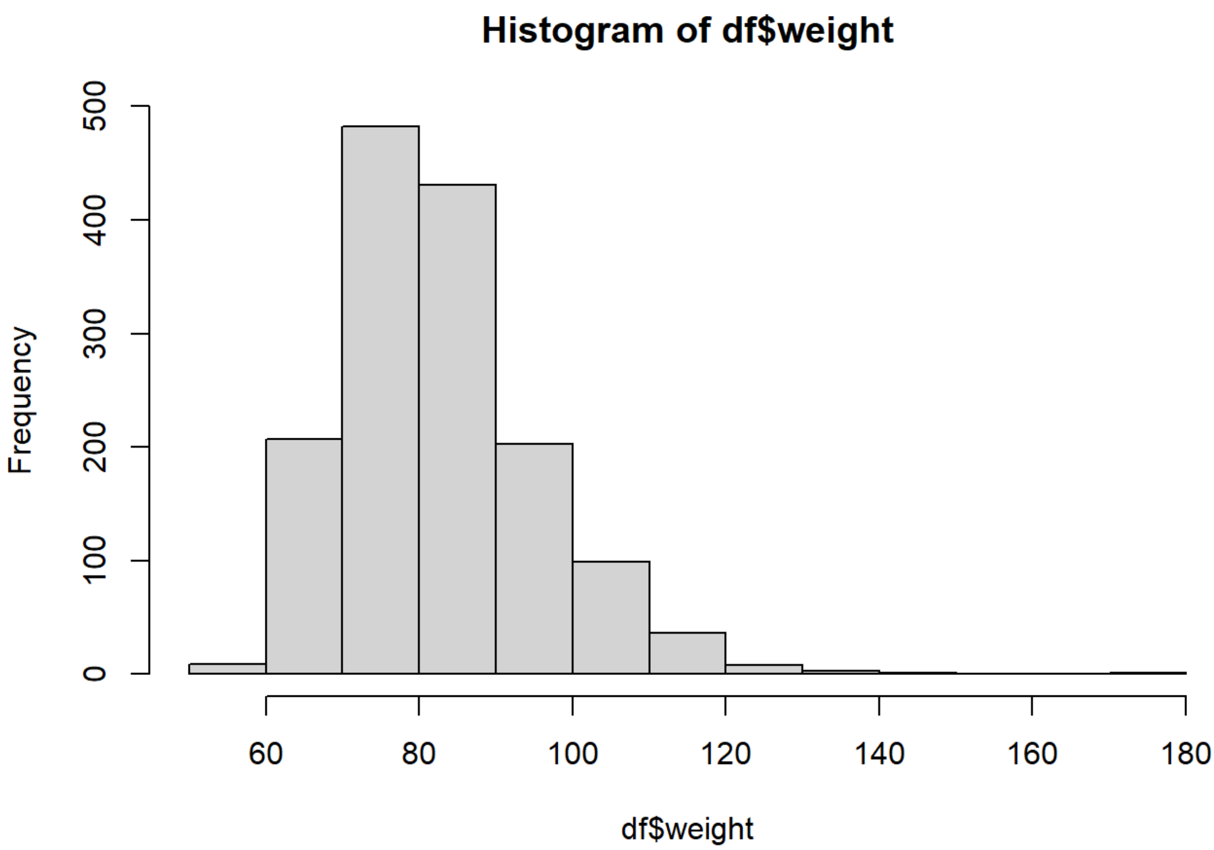
```
'data.frame':	1500 obs. of  8 variables:
 $ booking_id      : int  1 2 3 4 5 6 7 8 9 10 ...
 $ months_as_member: int  17 10 16 5 15 7 11 9 23 7 ...
 $ weight          : num  79.6 79 74.5 86.1 69.3 ...
 $ day_before      : int  8 2 14 10 8 2 6 10 10 10 ...
 $ day_of_week     : chr  "Wed" "Mon" "Sun" "Fri" ...
 $ time            : chr  "PM" "AM" "AM" "AM" ...
 $ category        : chr  "Strength" "HIIT" "Strength" "Cycling" ...
 $ attended        : int  0 0 0 0 0 0 0 0 1 0 ...
```

Univariarte EDA and cleaning

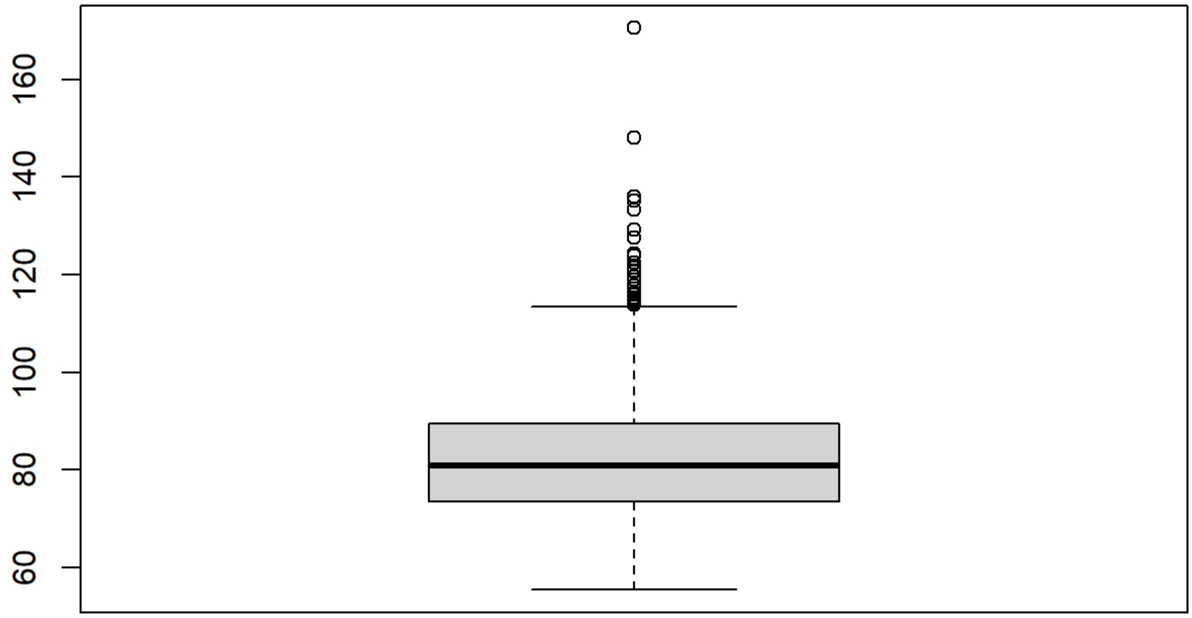
```r
hist(df$months_as_member)
```




```r
hist(df$weight)
```



```r
boxplot(df$weight)
```




```r
table(df$day_before)
```

```

  1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17  20  29 
 10 200  32 157  31  73  39 195  24 299  26 181  26 175  24   3   3   1   1 
```

```r
table(df$day_of_week)
```

```

      Fri      Fri.       Mon    Monday       Sat       Sun       Thu       Tue 
      279        26       218        10       202       213       241       195 
      Wed Wednesday 
       81        35 
```

```r
# Function to combine categories for days of the week
combine_days <- function(x) {
  # Define a mapping of full day names to their abbreviations
  day_map <- c("Monday" = "Mon", "Tuesday" = "Tue", "Wednesday" = "Wed",
               "Thursday" = "Thu", "Fri." = "Fri", "Saturday" = "Sat",
               "Sunday" = "Sun")

  # Remove periods from abbreviations
 # x <- gsub("\\.", "", x)
```

```
  # If the day is a full name, replace it with the abbreviation
  if (x %in% names(day_map)) {
    return(day_map[x])
  } else {
    return(x)
  }
}

# Apply the function to your column
df$day_of_week <- sapply(df$day_of_week, combine_days)

# Checking the modified data
table(df$day_of_week)
```

```
Fri Mon Sat Sun Thu Tue Wed
305 228 202 213 241 195 116
```

```
table(df$time)
```

```
  AM   PM
1141  359
```

```
table(df$category)
```

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

```
df$category[df$category == '-'] <- 'Others'
table(df$category)
```

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

```
table(df$attended)
```

```
   0    1
1046  454
```

Build a basic logit classifier and see how it performs. Check the train and test accuracy also.

Once you get the stable model, try feature selection using the stepwise function.

```
#df$attended <- as.factor(ifelse(df$attended=="yes",1,0))
```

```
df$day_of_week<- as.factor(df$day_of_week)
df$time <- as.factor(df$time)
df$category<- as.factor(df$category)
str(df)
```

```
'data.frame':   1500 obs. of  8 variables:
 $ booking_id      : int  1 2 3 4 5 6 7 8 9 10 ...
 $ months_as_member: int  17 10 16 5 15 7 11 9 23 7 ...
 $ weight          : num  79.6 79 74.5 86.1 69.3 ...
 $ day_before      : int  8 2 14 10 8 2 6 10 10 10 ...
 $ day_of_week     : Factor w/ 7 levels "Fri","Mon","Sat",..: 7 2 4 1 5 2 7 1 1 1 ...
 $ time            : Factor w/ 2 levels "AM","PM": 2 1 1 1 1 1 2 1 1 1 ...
 $ category        : Factor w/ 6 levels "Aqua","Cycling",..: 5 3 5 2 3 2 3 3 3 3 ...
 $ attended        : int  0 0 0 0 0 0 0 0 1 0 ...
```

```
#col_class <- sapply(1:ncol(df), function(x) class(df[,x]))
#col_id <- which(col_class == "character")
#for(i in 1:length(col_id)){
#  df[,col_id[i]] <- as.factor(df[,col_id[i]])
#}#
```

```
str(df)
```


```
'data.frame':   1500 obs. of  8 variables:
 $ booking_id      : int  1 2 3 4 5 6 7 8 9 10 ...
 $ months_as_member: int  17 10 16 5 15 7 11 9 23 7 ...
 $ weight          : num  79.6 79 74.5 86.1 69.3 ...
 $ day_before      : int  8 2 14 10 8 2 6 10 10 10 ...
 $ day_of_week     : Factor w/ 7 levels "Fri","Mon","Sat",..: 7 2 4 1 5 2 7 1 1 1 ...
 $ time            : Factor w/ 2 levels "AM","PM": 2 1 1 1 1 1 2 1 1 1 ...
 $ category        : Factor w/ 6 levels "Aqua","Cycling",..: 5 3 5 2 3 2 3 3 3 3 ...
 $ attended        : int  0 0 0 0 0 0 0 0 1 0 ...
```


Logit Model Fitting

Training and Test Data

```
set.seed(123457)
train.prop <- 0.80
strats <- df$attended
rr <- split(1:length(strats), strats)
idx <- sort(as.numeric(unlist(sapply(rr,
        function(x) sample(x, length(x)*train.prop)))))
df.train <- df[idx, ]
df.test <- df[-idx, ]
```

We can see whether the proportions of the two levels of the response ***y*** are the same in the train, test, and the entire data.

Fitting the full binary logit model

We use the *glm()* function to fit a binary regression with a logit link to the training data in *df.train*. The response is ***attended*** and the model includes all the predictors; we can call it the *full model*.

```
full.logit <- glm(attended  ~ . ,data = df.train,
                  family = binomial(link = "logit"))
summary(full.logit)
```

```
Call:
glm(formula = attended ~ ., family = binomial(link = "logit"),
    data = df.train)

Coefficients:
                   Estimate Std. Error z value Pr(>|z|)
(Intercept)       1.230e+00  1.616e+00   0.761  0.44661
booking_id        7.636e-05  1.750e-04   0.436  0.66255
months_as_member  1.208e-01  1.015e-02  11.902  < 2e-16 ***
weight           -1.528e-02  7.661e-03  -1.995  0.04603 *
day_before       -2.588e-01  1.423e-01  -1.819  0.06887 .
day_of_weekMon   -2.141e+00  1.166e+00  -1.837  0.06628 .
day_of_weekSat    5.341e-01  3.877e-01   1.378  0.16831
day_of_weekSun    1.224e+00  6.398e-01   1.913  0.05571 .
day_of_weekThu   -4.798e-01  3.848e-01  -1.247  0.21235
day_of_weekTue   -1.630e+00  9.116e-01  -1.788  0.07379 .
day_of_weekWed   -1.860e+00  6.869e-01  -2.708  0.00676 **
timePM           -1.122e-01  2.029e-01  -0.553  0.58023
categoryCycling  -2.617e-01  3.528e-01  -0.742  0.45828
categoryHIIT     -7.852e-02  3.374e-01  -0.233  0.81598
categoryOthers   -7.817e-01  1.206e+00  -0.648  0.51693
categoryStrength -4.592e-01  3.772e-01  -1.217  0.22349
categoryYoga     -3.498e-02  4.003e-01  -0.087  0.93036
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 1450.5  on 1182  degrees of freedom
Residual deviance: 1081.4  on 1166  degrees of freedom
  (16 observations deleted due to missingness)
AIC: 1115.4

Number of Fisher Scoring iterations: 5
```

The output shows which coefficients are significant for explaining the incidence of ***yes*** to subscribing to a deposit.

The output also shows the *null deviance* of 1450.5 on 1182 d.f and residual deviance 1081.4 on 1166 d.f.

The larger the difference between the null deviance and residual deviance, better the model fit. The AIC is 1115.4.

Fitting the null model

The null model is a model with the intercept only, as shown below.

```
null.logit <- glm(attended~1, data = df.train,
                  family = binomial(link = "logit"))
summary(null.logit)
```

```
Call:
glm(formula = attended ~ 1, family = binomial(link = "logit"),
    data = df.train)

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept) -0.83423    0.06286  -13.27   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 1470.4  on 1198  degrees of freedom
Residual deviance: 1470.4  on 1198  degrees of freedom
AIC: 1472.4

Number of Fisher Scoring iterations: 4
```

Variable selection

The following steps show how we can use both backward and forward selection to choose predictors for explaining the response **attended**, using the option *direction* ="*both*" in the *step()* function.

```
df <- na.omit(df)  # Where 'df' is your data frame
df$weight[is.na(df$weight)] <- mean(df$weight, na.rm = TRUE)
```

```
null.logit <- glm(attended ~ 1, data = df, family = "binomial")
full.logit <- glm(attended ~ ., data = df, family = "binomial")
```

```
both.logit <- step(null.logit, list(lower=formula(null.logit),
                                    upper=formula(full.logit)),
                   direction="both",trace=0, data = df.train)
formula(both.logit)
```

```
attended ~ months_as_member + time
```

```
summary(both.logit)
```

```
Call:
glm(formula = attended ~ months_as_member + time, family = "binomial",
    data = df)

Coefficients:
```

```
                 Estimate Std. Error z value Pr(>|z|)
(Intercept)      -2.81605    0.14945 -18.843   <2e-16 ***
months_as_member  0.12681    0.00818  15.503   <2e-16 ***
timePM           -0.27614    0.16229  -1.702   0.0888 .
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 1816.6  on 1479  degrees of freedom
Residual deviance: 1398.9  on 1477  degrees of freedom
AIC: 1404.9

Number of Fisher Scoring iterations: 5
```

The residual deviances from all three models are shown below

```
null.logit$deviance
```

```
[1] 1816.551
```

```
full.logit$deviance
```

```
[1] 1380.491
```

```
both.logit$deviance
```

```
[1] 1398.868
```

The full model is preferred for the training data set.

Assess test data accuracy

We assess how well the models full.logit and both.logit fit the response from the test data. Use the *predict()* function to predict the test data under both fitted models

```
pred.both <- predict(both.logit, newdata = df.test, type="response")
pred.full <- predict(full.logit, newdata = df.test, type="response")
```

We compute and compare the confusion matrices using the code below.

```
(table.both <- table(pred.both > 0.5, df.test$attended))
```

```
         0   1
  FALSE 198  60
  TRUE   12  31
```

```
(table.full <- table(pred.full > 0.5, df.test$attended))
```

```
         0   1
```

```
  FALSE 192  61
  TRUE   14  30
```

We can then compute prediction accuracy for the test data as percentages.

```r
(accuracy.both <- round((sum(diag(table.both))/sum(table.both))*100,2))
```

```
[1] 76.08
```

```r
(accuracy.full <- round((sum(diag(table.full))/sum(table.full))*100,2))
```

```
[1] 74.75
```

Both models have almost about the same accuracy for predicting the test data.

```r
library(pROC)
```

```
Warning: package 'pROC' was built under R version 4.3.2

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


Attaching package: 'pROC'

The following objects are masked from 'package:stats':

    cov, smooth, var
```

```r
roc.both <- roc(df.test$attended, pred.both, levels=c(1,0))
```

```
Setting direction: controls > cases
```

```r
auc(df.test$attended, pred.both)
```

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

Setting direction: controls < cases

Area under the curve: 0.7847
```

We also get the AUC for "full.logit":

```r
roc.full <- roc(df.test$attended, pred.full, levels=c(1,0))
```

```
Setting direction: controls > cases
```

```r
auc(df.test$attended, pred.full)
```

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

Setting direction: controls < cases
```

```
Area under the curve: 0.7838
```

The area under the curve (AUC) is similar for the test data under the two model fits

The *confusionMatrix()* function in the package *caret* is also useful for looking at several criteria for assessing the predictions, including the ones we showed above. We show the code below

```
library(caret)
```

```
Loading required package: ggplot2

Loading required package: lattice
```

```
b <- ifelse(pred.both > 0.5,1,0)
cm.both <- confusionMatrix(reference=as.factor(df.test$attended),
            data=as.factor(b), mode="everything")
f <- ifelse(pred.full > 0.5,1,0)
cm.full <- confusionMatrix(reference=as.factor(df.test$attended),
            data=as.factor(f), mode="everything")
```

Assess train data accuracy

Using code similar to what we showed for the test data, we can also predict the train data and assess accuracy under both models

```
## Predict train data using both.logit and full.logit
pred.tr.both <- predict(both.logit, newdata = df.train, type="response")
pred.tr.full <- predict(full.logit, newdata = df.train, type="response")
# Accuracy of both.logit and full.logit
# Confusion matrix
(table.tr.both <- table(pred.tr.both > 0.5, df.train$attended))
```

```
         0   1
  FALSE 781 208
  TRUE   55 155
```

```
(table.tr.full <- table(pred.tr.full > 0.5, df.train$attended))
```

```
         0   1
  FALSE 760 195
  TRUE   65 163
```

```
# Accuracy
(accuracy.tr.both <- round((sum(diag(table.tr.both))/sum(table.tr.both))*100,2))
```

```
[1] 78.07
```

```
(accuracy.tr.full <- round((sum(diag(table.tr.full))/sum(table.tr.full))*100,2))
```

```
[1] 78.02
```

```
# AUC
roc.tr.both <- roc(df.train$attended, pred.tr.both, levels=c(1,0))
```

```
Setting direction: controls > cases
```

```
auc(df.train$attended, pred.tr.both)
```

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

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

```
Area under the curve: 0.8327
```

```
roc.tr.full <- roc(df.train$attended, pred.tr.full, levels=c(1,0))
```

```
Setting direction: controls > cases
```

```
auc(df.train$attended, pred.tr.full)
```

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

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

```
Area under the curve: 0.8382
```

We see that the two models give similar performance.

The accuracy on both.logit and full.logit are 78.07% and 78.02 respectively, and their respective AUC's are 0.8327 and 0.8382.

Backward elimination for variable selection

Instead of stepwise selection of predictors, the code for variable selection using only backward elimination is shown below.

```
backwards <- step(full.logit, trace = 0)  #suppress details of each iteration
# backwards <- step(full.logit)  # to show all details
formula(backwards)
```

```
attended ~ months_as_member + time
```

```
summary(backwards)
```

```

Call:
glm(formula = attended ~ months_as_member + time, family = "binomial",
    data = df)

Coefficients:
```

```
                 Estimate Std. Error z value Pr(>|z|)    
(Intercept)      -2.81605    0.14945 -18.843   <2e-16 ***
months_as_member  0.12681    0.00818  15.503   <2e-16 ***
timePM           -0.27614    0.16229  -1.702   0.0888 .  
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 1816.6  on 1479  degrees of freedom
Residual deviance: 1398.9  on 1477  degrees of freedom
AIC: 1404.9

Number of Fisher Scoring iterations: 5
```

Forward selection

The code for variable selection using only forward selection is shown below.

```
forwards = step(null.logit, trace=0,
      scope=list(lower=formula(null.logit),
                 upper=formula(full.logit)), direction="forward")
formula(forwards)
```

```
attended ~ months_as_member + time
```

```
summary(forwards)
```


```
Call:
glm(formula = attended ~ months_as_member + time, family = "binomial",
    data = df)

Coefficients:
                 Estimate Std. Error z value Pr(>|z|)    
(Intercept)      -2.81605    0.14945 -18.843   <2e-16 ***
months_as_member  0.12681    0.00818  15.503   <2e-16 ***
timePM           -0.27614    0.16229  -1.702   0.0888 .  
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 1816.6  on 1479  degrees of freedom
Residual deviance: 1398.9  on 1477  degrees of freedom
AIC: 1404.9

Number of Fisher Scoring iterations: 5
```