# Assignment 2: Regression Analysis Using R

Written by Daanish Ahmed

DATA 630 9041

Semester Summer 2017

Professor Edward Herranz

UMUC

June 19, 2017

## Introduction

The logistic regression model is a powerful method for determining the probabilities of certain outcomes within a dataset. This model estimates the values of the coefficients for each independent variable, and it generates a function of best fit that can be used to estimate the value of the dependent variable. Unlike linear regression, logistic regression is used to predict variables with categorical outputs, and it requires the use of a logarithmic function rather than a strictly linear function (Herranz, 2017). This model is extremely useful for determining whether a certain observation will produce an output falling under one of several categories. For instance, we can use this model on a dataset consisting of smoking behavior to determine the likelihood of an individual developing lung cancer. In my analysis, I will use R software to apply this model to a dataset containing heart attack information (Hosmer, Lemeshow, & May, 2008). I will focus on predicting a patient's vital status at their most recent follow-up session, indicating whether the patient died from the heart attack. Since this variable only has two outcomes (alive or dead), I will be using binary logistic regression to create my model. After building the model, I will evaluate its accuracy and attempt to improve it by creating a minimal adequate model. Finally, I will compare it to a Naïve Bayes classification model on the same dataset to see which model produces the best results. I expect that my findings will offer a realistic prediction on a patient's likelihood of survival based on the variables included in the model.

## Data Exploration

I will begin my analysis by describing my exploration of the original dataset. This dataset contains 22 variables and 500 observations (see Figure 1 in Appendix B). Among these variables

are the patient's age, gender, initial heart rate, body mass index, history of cardiovascular disease, length of hospital stay, the patient's discharge status from the hospital, and their vital status at the last follow-up session (Hosmer et al., 2008). As stated before, I will use the patient's vital status during their final follow-up as the dependent variable in my model. Three of these variables are listed as factors by default, but these variables are listed in date format. One of the variables, body mass index, is a numeric type. The remaining variables are all integers, but many of them represent categorical values of 0 and 1. From here, I generated a set of descriptive statistics to gain further insight on the data (see Figure 2 in Appendix B). Please note that these statistics represent the variables after the preprocessing process—some of the variables have either been removed or converted into factors. These numbers indicate that 60% of the patients are male, the median patient age is 72, and about 43% of patients die from their heart attack. The statistics also reveal that there are no missing values in any of the variables. These findings serve as a good starting point towards understanding the data before creating the regression model.

## Preprocessing

Before building the model, it is crucial to perform preprocessing on the dataset to ensure that my code functions properly on the data. The first step is to remove any unique identifiers (IDs) from the dataset because they are unnecessary for the analysis. The first variable is an ID, and thus I removed it. Afterwards, I proceeded to handle any missing values that appeared within the dataset. But since no missing values existed in any of the variables (see Figure 3 in Appendix B), this step was unnecessary. Next, I converted the dependent variable "fstat" into a factor. Because we are using the "glm" function in R, we need to convert the dependent variable into a factor so that the function will perform logistic regression on the data.

After this step, I removed additional variables that were either unnecessary or problematic to the model. Several variables were either in date format or referred to a specific year, and thus they were likely unneeded for the analysis. Another variable that I removed was "dstat," which referred to the patient's vital status when they were discharged from the hospital. This variable was directly correlated with my dependent variable "fstat," and thus it would have caused problems in the model if it were not removed. Additionally, I removed two other independent variables: the length of the hospital stay and the total length of the follow-up period. These two variables were correlated with each other, and therefore I removed them to prevent problems of multicollinearity (Herranz, 2017). The last step was to convert eight other numeric variables into factors. These variables—including gender, atrial fibrillation, and congestive heart complications—had numeric values of 0 and 1 which represented categorical values. Converting these variables into factors is not a requirement for creating the model, but I chose to perform these steps to make the results easier to understand. This completes the preprocessing process, and the data is now ready to be implemented with the logistic regression model.

## Logistic Regression Model

I will now describe the steps that I took to build the logistic regression model for my dataset. First, I divided the dataset into a training set containing 70% of the data and a test set with the remaining 30% of data. The training data is used to create the classification tree model, while the test data is needed to evaluate the model's accuracy ("Linear Regression," 2017). The reason why I split the dataset is to prevent overfitting, in which too many independent variables are used in the model. Many patterns that are found in an overfit dataset are caused by chance and do not indicate an actual relationship between the variables (James, Witten, Hastie, & Tibshirani, 2013).

But by splitting the data into two subsets, we limit the amount of data processed at once and therefore reduce the likelihood of overfitting. After this step, I built the logistic regression model on the training data using the "glm" function in R. For the input values, I used "fstat" as the dependent variable and included all remaining columns as independent variables. The reason that I include all remaining variables is because I already removed the problematic variables during preprocessing. If any of these variables are not significant, they will be removed later in the analysis when I create the minimal adequate model.

After building the initial model, I generated a set of descriptive statistics from the training data which includes predictions for the intercept values and coefficients (see Figure 4 in Appendix B). The intercept and coefficients are needed for the logistic regression equation to predict the value of the dependent variable. This command also provides information on the residuals, which are the difference between the predicted values and the actual values ("Logistic Regression," 2017). In this model, the median residual is -0.376, meaning that the median predictions are roughly 38% smaller than the actual values. Furthermore, the descriptive statistics provide the p-values for each variable and whether that variable is significant. If a variable has a p-value that is greater than 0.05, then it is not significant and can be removed from the model ("Logistic Regression," 2017). Next, I generated confusion matrices for the training and test datasets (see Figures 5 and 6 in Appendix B). These matrices can be used to determine the classification accuracy of our model. Based on these figures, we find that the model's accuracy on the training data is 74.4%, while its accuracy on the test data is 81.1%. The last step to building my initial model is to create a residuals plot that shows the predictions compared to residuals (see Figure 7 in Appendix B). These results will be described in greater detail in the following section.

## Initial Results

By analyzing these findings, we can see whether my model is a good fit to the dataset. I will first examine the residuals to see their implications towards the model's accuracy (see Figure 4 in Appendix B). The figure indicates that the minimum value is -2.32, the maximum is 2.51, and the median is roughly -0.38. Based on these numbers, there seems to be room for improvement regarding the model's accuracy. Additionally, the p-values suggest that only three of the variables are significant—age, initial heart rate, and congestive heart complications. Due to this, the model can most likely be improved by creating a minimal adequate model that removes the insignificant variables. Also, examining the residuals plot reveals a normally-shaped prediction curve (see Figure 7 in Appendix B), which may imply that the model fits the data reasonably well. However, an ideal model would have a curve that is shaped closer to a straight horizontal line (Charpentier, 2013), meaning that the current model is not quite a perfect fit.

But perhaps the strongest way to measure the model's accuracy is to analyze the confusion matrices for the training and test datasets (see Figures 5 and 6 in Appendix B). The reason why confusion matrices are so useful is because they show the exact percentage of how accurately the model predicts the values in the current dataset. In these matrices, the predicted class appears as the rows while the actual class appears as the columns. The numbers that appear diagonally from top left to bottom right are the number of correctly-predicted values, and dividing the sum of these numbers by the total number of instances will produce the model's classification accuracy ("Logistic Regression," 2017). As mentioned earlier, I calculated the accuracies of the training and test sets as 74.4% and 81.1% respectively. Interestingly, the model is more accurate on the test set than on the training set. It is possible that using a larger sample of test data (that is still smaller than the training data) may yield a result closer to the training accuracy. But since the test

results are more important than the training results, we will focus on the 81.1% accuracy of the test data. These results seem to suggest that the model is somewhat accurate. However, if a model can only accurately predict four out of five heart attack deaths, then it is not a reliable model and will therefore need significant improvement to be useful for preventing future deaths.

## Minimal Adequate Model

To improve my model's accuracy, I created a minimal adequate model that filters out all independent variables that are not significant. To determine which variables are significant, I used a step function that removes insignificant variables one at a time until only the significant ones remain. This function works by attempting to minimize the Akaike Information Criterion (AIC). It will drop variables to lower the AIC and will continue this process until the AIC has reached its minimum level ("Linear Regression," 2017). When the process was finished, the remaining variables were age, initial heart rate, initial diastolic blood pressure, cardiogenic shock, congestive heart complications, and complete heart block. These variables do indeed seem very important to the model. For instance, high diastolic blood pressure is one of the largest contributors to premature death across the world ("Hypertension," n.d.). Likewise, cardiogenic shock results in a heart that is too weak to pump blood throughout the body, and it is usually fatal if not treated immediately ("What is Cardiogenic Shock," 2011). Next, I created a new model using the "glm" function—but this time I used only these six variables as parameters instead of every independent variable. I then generated a set of descriptive statistics that includes recalculated values for the coefficients, intercept, residuals, and p-values (see Figure 8 in Appendix B). After this, I computed the confusion matrices for the minimal adequate model on both training and test data (see Figures 9 and 10 in Appendix B). These figures reveal that the model's accuracy for the training and test

sets are 74.1% and 81.1% respectively. Finally, I created a residuals plot for the minimal adequate model for comparison with the original plot (see Figure 11 in Appendix B).

I will now compare the results of this new model to those of the original regression model. First, I will examine the descriptive statistics—particularly the residuals and the p-values (see Figure 8 in Appendix B). Interestingly, the residuals do not differ drastically from those of the original model. The minimum, maximum, and median deviance residuals are only slightly lower than their original values. However, the p-values suggest that four out of the six remaining variables are significant. This implies that the overall model is a better fit for the data because it excludes the variables of lesser significance. But when looking at the confusion matrices for the minimal adequate model, we see that the classification accuracies for the training and test sets are 74.1% and 81.1%. These are almost identical to the accuracies for the original model (74.4% and 81.1%). Furthermore, we see that the residual plot for the minimal adequate model (see Figure 11 in Appendix B) is also nearly identical to that of the first model. These findings seem to raise doubts as to whether my new model is more accurate. The original model was only somewhat accurate, but the minimal adequate model seems to have the same accuracy even though it removed all insignificant variables. One way to improve the model would be to change the variables that were excluded in the beginning, since some of them may have been important. However, it is also possible that looking at a different type of model would produce more accurate results.

## Naïve Bayes Classification

In this section, I will briefly construct a Naïve Bayes classification model using my dataset and compare its accuracy to that of my previous model. Since my focus in this assignment is on

logistic regression, my description of the Bayes model will not be as detailed and is mostly intended as a comparison to the original model. Since the Naïve Bayes method in R requires all variables to be factors ("Naïve Bayes Classification," 2017), I discretized every remaining numeric variable. Next, I once again split the dataset into a training set and test set consisting of 70% and 30% of the data respectively. I then generated the model using "fstat" as the dependent variable and all remaining variables as predictors. The model output displays the probabilities for the dependent variable in addition to those for all independent variables (see Figure 12 in Appendix B). We see that there is a 44% chance that the victim will die from their heart attack, which is consistent with my original findings (see Figure 2 in Appendix B). From here, I created confusion matrices for the training and test datasets, revealing that the classification accuracies for training and test data are 76.7% and 80.4% respectively (see Figures 13 and 14 in Appendix B). Finally, I generated a mosaic plot that illustrates the accuracy of the predictions (see Figure 15 in Appendix B). For this plot, the blue areas show when the prediction is equal to the actual value, while red areas indicate all misclassified instances.

Once again, the confusion matrices are the strongest way to evaluate the accuracies of the models. Surprisingly, the classification accuracies I obtained from the Naïve Bayes model are very similar to those from the logistic regression model. The test data accuracy from the Bayes model is 80.4%, which only slightly differs from the logistic regression model's accuracy of 81.1%. In fact, the accuracy for this model is slightly lower than that of the original model. This seems to suggest that both models fit the data roughly the same, though the logistic regression model may be a slightly better fit. When examining the mosaic plot for the model (see Figure 15 in Appendix B), we see that most of the predictions are accurate. However, the noticeable amounts of red space suggest that the model needs to be much more accurate to be reliable for medical

purposes. Altogether, the Bayes model did not produce a more accurate prediction as I had expected. Thus, the most logical way to improve my model's accuracy would be to recreate it while keeping some of the variables that had been removed in the beginning.

## Conclusion

In this study, I created a logistic regression model to predict a patient's likelihood of survival from a heart attack. Though my initial model was accurate 80% of the time, there was still the need for a more accurate model. Thus, I created a minimal adequate model by removing all variables that were not significant to the analysis. And while this new model was more streamlined, its accuracy was nearly identical to that of the first model. One possible method of improvement would be to use a different type of model and see if that model is a better fit to the data. Thus, I created a Naïve Bayes model for my dataset and compared the results to my original model. But once again, I found that the accuracy of the two models was nearly identical. In fact, the logistic regression model appears to be slightly more accurate. Because of this, I believe that the main limitation to my model was the choice of variables used to create each model. It is likely that some of the variables I removed during preprocessing may have been critical for the analysis, or that I did not remove enough variables from the dataset. Therefore, I would recommend recreating this study while experimenting with different variables for future research. I believe that my model serves as a good foundation for predicting heart attack deaths, but I think that it would need to have an accuracy of at least 95% to 99% to be viable for medical use. If this model can be refined to reach an accuracy of that level, then it can be effectively used to predict the likelihood of a patient dying from a heart attack. This may eventually allow researchers to work towards preventing these deaths in the future.

## References


Charpentier, A. (2013, August 23). Residuals from a logistic regression. Retrieved June 22, 2017, from http://freakonometrics.hypotheses.org/8210

Herranz, E. (2017, February 17). *DATA 630 Regression Herranz* [Video file]. Retrieved June 12, 2017, from https://www.youtube.com/watch?v=ObZGJEukhvk&feature=youtu.be

Hosmer, D. W., Lemeshow, S., & May, S. (2008). Description of the variables in the WHAS500 dataset. Retrieved June 17, 2017, from http://www.utstat.toronto.edu/burkett/sta442f15/whas500.txt

Hypertension. (n.d.). Retrieved June 23, 2017, from http://www.world-heart-federation.org/cardiovascular-health/cardiovascular-disease-risk-factors/hypertension/

James, G., Witten, D., Hastie, T., & Tibshirani, R. (2013). *An introduction to statistical learning: with applications in R*. Retrieved June 14, 2017.

Linear Regression Using R – Prediction Exercise (2017). Retrieved June 12, 2017.

Logistic Regression Using R – Classification via Logistic Regression (2017). Retrieved June 14, 2017.

Naïve Bayes Classification Using R – Optional Exercise (2017). Retrieved June 14, 2017.

What Is Cardiogenic Shock? (2011, July 1). Retrieved June 23, 2017, from https://www.nhlbi.nih.gov/health/health-topics/topics/shock

## Appendix A

R Script for Assignment 2

```
# DATA 630 Assignment 2

# Written by Daanish Ahmed

# Semester Summer 2017

# June 17, 2017

# Professor Edward Herranz


# This R script creates a logistic regression model using a dataset containing

# information on heart attack victims.  The purpose of this assignment is to

# generate a model to estimate the probability of a variable having a certain

# outcome.  I am focusing on predicting the variable fstat, which represents

# a patient's vital status at the time of his or her last follow-up session

# (indicating whether they are alive or dead).  This script consists of several

# components, including opening and initializing the data, exploration and

# preprocessing, building and analyzing the logistic regression model, and

# creating a minimal adequate model that excludes insignificant variables.


# This section of code covers opening the dataset and initializing the packages

# that are used in this script.


# Sets the working directory for this assignment.  Please change this directory
```

```
# to whichever directory you are using, and make sure that all files are placed
# in that location.
setwd("~/Class Documents/2016-17 Summer/DATA 630/R/Assignment 2")

# We do not need to install or load any packages in this section of the script.
# The logistic regression function is located in the stats package, which by
# default is loaded automatically into R.

# Opens the CSV file "whas500.csv".
heart_atk <- read.csv(file="whas500.csv", head=TRUE, sep=",")

# End of opening the dataset.


# This section of code covers data preprocessing.  It includes exploration of
# the original dataset, removing variables, and dealing with missing values.

# Previews the heart attack dataset.
View(heart_atk)

# Shows the descriptive statistics for all variables in the dataset.
summary(heart_atk)
```

```
# Displays the structure of the data.  This is necessary to see if there are
# any unique identifiers (IDs) that can be removed.  Such variables are not
# useful for the analysis and should be removed.
str(heart_atk)

# The first variable is an ID, and we remove it.
heart_atk <- heart_atk[, -1]

# Verifies that the ID variable has been removed.
str(heart_atk)

# This function checks to see how many missing values are in each variable.
apply(heart_atk, 2, function(heart_atk) sum(is.na(heart_atk)))

# Since there are no missing values in any of the variables, we do not need to
# replace any values.

# Since we are using logistic regression, it is important to convert the
# dependent variable fstat into a factor so that the glm function will perform
# logistic regression.
heart_atk$fstat <- as.factor(heart_atk$fstat)

# We remove several variables.  Variables 15, 16, and 17 are dates, while
```

```
# variable 14 refers to the year, all of which are unneeded.  # Variable 19,
# dstat, is directly correlated with the dependent variable fstat, thus we
# remove it.  Variables 18 and 20 are independent variables that are correlated
# with each other, thus we remove them as well.
heart_atk <- heart_atk[, -c(14, 15, 16, 17, 18, 19, 20)]

# Verifies that the unwanted variables have been removed.
View(heart_atk)

# Several variables have numeric values 0 and 1 that represent categorical
# values.  We convert these into factors.  This step is not required to
# generate the model, but it is useful for analysis.
heart_atk$gender <- as.factor(heart_atk$gender)
heart_atk$cvd <- as.factor(heart_atk$cvd)
heart_atk$afb <- as.factor(heart_atk$afb)
heart_atk$sho <- as.factor(heart_atk$sho)
heart_atk$chf <- as.factor(heart_atk$chf)
heart_atk$av3 <- as.factor(heart_atk$av3)
heart_atk$miord <- as.factor(heart_atk$miord)
heart_atk$mitype <- as.factor(heart_atk$mitype)

# Verifies that the variables have been converted to factors.
summary(heart_atk)
```

```
# End of data preprocessing.



# This section of code covers the creation of the logistic regression model.
# It includes dividing the data into training and test datasets, creating
# and analyzing the model, and plotting the model.

# Generates a random seed to allow us to reproduce the results.
set.seed(1234)

# The following code splits the heart attack dataset into a training set
# consisting of 70% of the data and a test set containing 30% of the data.
ind <- sample(2, nrow(heart_atk), replace = TRUE, prob = c(0.7, 0.3))
train.data <- heart_atk [ind == 1, ]
test.data <- heart_atk [ind == 2, ]

# Generates the logistic regression model on the training data.
model <- glm(fstat ~ ., family = binomial, data = train.data)

# Displays important information including the summary statistics, the
# values of the intercept and the coefficients, and the p-values.
summary(model)
```

```
# Builds the confusion matrix for the training data.
table(round(predict(model, train.data, type="response")), train.data$fstat)

# Builds the confusion matrix for the test data.
mypredictions <- round(predict(model, test.data, type="response"))
table(mypredictions, test.data$fstat)

# Creates the residuals plot for the data.
plot(predict(model), residuals(model), col=c("blue"))
lines(lowess(predict(model), residuals(model)), col=c("black"), lwd=2)
abline(h=0, col="grey")

# End of creating the logistic regression model.


# This section of code covers the creation of the minimal adequate model.
# This model will remove all variables that are not significant, leaving
# only significant variables.

# This command shows a step model that removes insignificant variables
# one at a time until only significant variables remain.  This step is
# needed to obtain the significant variables to generate the actual
# minimal adequate model.
```

```
summary(step(model))

# Creates the minimal adequate model using the variables found at the
# final iteration of the step command.
mamodel <- glm(fstat ~ age + hr + diasbp + sho + chf + av3,
            family=binomial, train.data)

# Shows the summary statistics of the minimal adequate model.
summary(mamodel)

# Builds the confusion matrix for the training data using the minimal
# adequate model.
table(round(predict(mamodel, train.data, type="response")), train.data$fstat)

# Builds the confusion matrix for the test data.
mypredictions <- round(predict(mamodel, test.data, type="response"))
table(mypredictions, test.data$fstat)

# Creates the residuals plot for the minimal adequate model.
plot(predict(mamodel), residuals(mamodel), col=c("blue"))
lines(lowess(predict(mamodel), residuals(mamodel)), col=c("black"), lwd=2)
abline(h=0, col="grey")
```

```
# End of creating minimal adequate model.

# The next two sections of code cover the implementation of the Naive Bayes
# classification model.  This model is generated for the purpose of comparing
# it to my original model and seeing which one is more accurate.

# The following code covers the preprocessing of the dataset to prepare it for
# implementation with the Naive Bayes method.

# The following packages are required for implementing the Naive Bayes model
# and discretization.  If you have not installed these packages yet, please
# remove the two # symbols below.
# install.packages("arules")
# install.packages("e1071")

# Loads the arules and e1071 packages into the system.
library("arules")
library("e1071")

# We need to discretize all of the variables that are not already factors.  This
# is required before running the Naive Bayes method.  Since most of the variables
# have already been converted, we only need to convert a few.
heart_atk$age <- discretize(heart_atk$age, "frequency", categories=6)
```

```
heart_atk$hr <- discretize(heart_atk$hr, "frequency", categories=6)
heart_atk$sysbp <- discretize(heart_atk$sysbp, "frequency", categories=6)
heart_atk$diasbp <- discretize(heart_atk$diasbp, "frequency", categories=6)
heart_atk$bmi <- discretize(heart_atk$bmi, "frequency", categories=6)

# Verifies that all variables are factors.
str(heart_atk)

# End of Naive Bayes preprocessing section.



# The following code covers the creation of the Naive Bayes model.  It involves
# splitting the data into training and test sets, creating and analyzing the
# model, and plotting the results.

# Generates a random seed to allow us to reproduce the results.
set.seed(1234)

# We once again split the data into a training set containing 70% of the data
# and a test set containing 30% of the data.
ind <- sample(2, nrow(heart_atk), replace = TRUE, prob = c(0.7, 0.3))
train.data <- heart_atk[ind == 1, ]
test.data <- heart_atk[ind == 2, ]
```

```
# Generates the Naive Bayes classification model on the training data.
model <- naiveBayes(fstat ~ ., train.data)

# Displays the model output, including probabilities for each variable.
print(model)

# Builds the confusion matrix for the training data.
table(predict(model, train.data), train.data$fstat)

# Builds the confusion matrix for the test data.
table(predict(model, test.data), test.data$fstat)

# Generates a mosaic plot showing the predicted values of fstat compared to
# the actual values.  The blue color shows when the prediction is equal to
# the actual value.  Red indicates all misclassified instances.
mosaicplot(table(predict(model, test.data), test.data$fstat), shade=TRUE,
        main="Predicted vs. Actual FSTAT")

# End of creating Naive Bayes classification model.

# End of script.
```

## Appendix B

Relevant R Output Images

```
> str(heart_atk)
'data.frame':   500 obs. of  22 variables:
 $ id        : int  1 2 3 4 5 6 7 8 9 10 ...
 $ age       : int  83 49 70 70 70 70 57 55 88 54 ...
 $ gender    : int  0 0 1 0 0 0 0 0 1 0 ...
 $ hr        : int  89 84 83 65 63 76 73 91 63 104 ...
 $ sysbp     : int  152 120 147 123 135 83 191 147 209 166 ...
 $ diasbp    : int  78 60 88 76 85 54 116 95 100 106 ...
 $ bmi       : num  25.5 24 22.1 26.6 24.4 ...
 $ cvd       : int  1 1 0 1 1 1 1 1 1 1 ...
 $ afb       : int  1 0 0 0 0 0 0 0 0 0 ...
 $ sho       : int  0 0 0 0 0 0 0 0 0 0 ...
 $ chf       : int  0 0 0 1 0 0 0 0 1 0 ...
 $ av3       : int  0 0 0 0 0 0 1 0 0 0 ...
 $ miord     : int  1 0 0 0 0 0 0 0 0 0 ...
 $ mitype    : int  0 1 1 1 1 0 1 1 0 0 ...
 $ year      : int  1 1 1 1 1 1 1 1 1 1 ...
 $ admitdate: Factor w/ 394 levels "1/1/1997","1/10/1999",..: 6 13 1 153 168 170 169 3 141 8 ...
 $ disdate   : Factor w/ 394 levels "1/1/1999","1/11/1999",..: 7 16 25 173 198 182 185 5 24 12 ...
 $ fdate     : Factor w/ 203 levels "1/1/2001","1/10/1998",..: 73 73 73 55 73 94 73 82 175 73 ...
 $ los       : int  5 5 5 10 6 1 5 4 4 5 ...
 $ dstat     : int  0 0 0 0 0 1 0 0 0 0 ...
 $ lenfol    : int  2178 2172 2190 297 2131 1 2122 1496 920 2175 ...
 $ fstat     : int  0 0 0 1 0 1 0 1 1 0 ...
>
```

*Figure 1.* Initial Data Structure of Heart Attack Dataset.

```
> summary(heart_atk)
      age         gender          hr             sysbp           diasbp            bmi         cvd      afb      sho
 Min.   : 30.00   0:300   Min.   : 35.00   Min.   : 57.0   Min.   :  6.00   Min.   :13.05   0:125   0:422   0:478
 1st Qu.: 59.00   1:200   1st Qu.: 69.00   1st Qu.:123.0   1st Qu.: 63.00   1st Qu.:23.22   1:375   1: 78   1: 22
 Median : 72.00           Median : 85.00   Median :141.5   Median : 79.00   Median :25.95
 Mean   : 69.85           Mean   : 87.02   Mean   :144.7   Mean   : 78.27   Mean   :26.61
 3rd Qu.: 82.00           3rd Qu.:100.25   3rd Qu.:164.0   3rd Qu.: 91.25   3rd Qu.:29.39
 Max.   :104.00           Max.   :186.00   Max.   :244.0   Max.   :198.00   Max.   :44.84
 chf      av3     miord   mitype  fstat
 0:345   0:489   0:329   0:347   0:285
 1:155   1: 11   1:171   1:153   1:215
```

*Figure 2.* Descriptive Statistics of Variables After Preprocessing.

```
> apply(heart_atk, 2, function(heart_atk) sum(is.na(heart_atk)))
      age    gender        hr     sysbp    diasbp       bmi       cvd       afb       sho       chf       av3     miord
        0         0         0         0         0         0         0         0         0         0         0         0
   mitype      year admitdate   disdate     fdate       los     dstat    lenfol     fstat
        0         0         0         0         0         0         0         0         0
```

*Figure 3.* Number of Missing Values for all Variables.

```
> summary(model)

Call:
glm(formula = fstat ~ ., family = binomial, data = train.data)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-2.3235  -0.7842  -0.3761   0.8352   2.5088

Coefficients:
             Estimate Std. Error z value Pr(>|z|)
(Intercept) -5.553603   1.511595  -3.674 0.000239 ***
age          0.068879   0.012338   5.582 2.37e-08 ***
gender1      0.178078   0.279259   0.638 0.523683
hr           0.014786   0.006029   2.453 0.014186 *
sysbp       -0.002160   0.005667  -0.381 0.703070
diasbp      -0.007066   0.009443  -0.748 0.454314
bmi         -0.016682   0.027001  -0.618 0.536698
cvd1        -0.025377   0.338454  -0.075 0.940231
afb1         0.004813   0.349924   0.014 0.989026
sho1         1.187106   0.797112   1.489 0.136419
chf1         0.968580   0.290358   3.336 0.000851 ***
av31         1.392688   0.952948   1.461 0.143891
miord1       0.118876   0.289996   0.410 0.681862
mitype1     -0.197360   0.326416  -0.605 0.545426
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 483.42  on 351  degrees of freedom
Residual deviance: 362.90  on 338  degrees of freedom
AIC: 390.9

Number of Fisher Scoring iterations: 4
```

*Figure 4.* Descriptive Statistics of Logistic Regression Model on Training Data.

```
> table(round(predict(model, train.data, type="response")), train.data$fstat)

     0   1
  0 155  49
  1  41 107
>
```

*Figure 5.* Confusion Matrix for Logistic Regression Model on Training Data.

```
> mypredictions <- round(predict(model, test.data, type="response"))
> table(mypredictions, test.data$fstat)

mypredictions  0  1
            0 75 14
            1 14 45
> |
```

*Figure 6.* Confusion Matrix for Logistic Regression Model on Test Data.

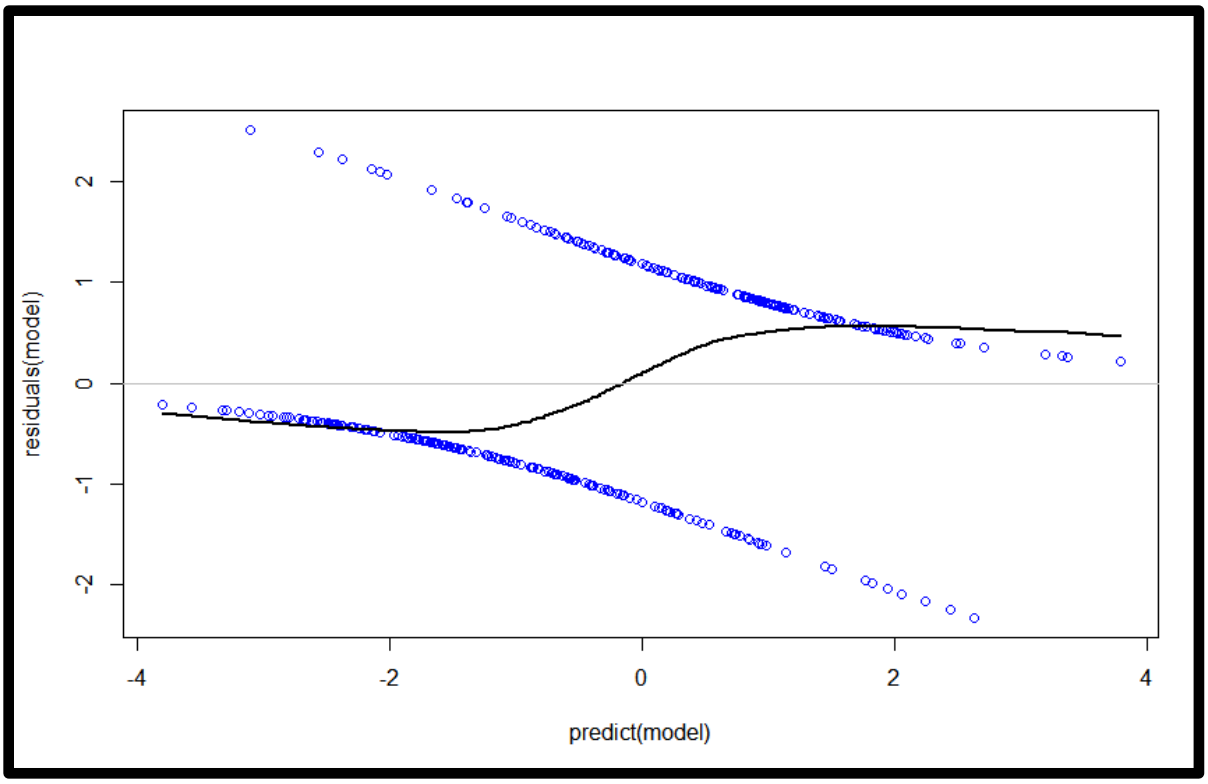


*Figure 7.* Residuals Plot for Logistic Regression Model.

```
> summary(mamodel)

Call:
glm(formula = fstat ~ age + hr + diasbp + sho + chf + av3, family = binomial,
    data = train.data)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-2.2806  -0.8173  -0.3724   0.8531   2.5960

Coefficients:
             Estimate Std. Error z value Pr(>|z|)
(Intercept) -6.472178   1.066750  -6.067  1.3e-09 ***
age          0.073988   0.011115   6.657  2.8e-11 ***
hr           0.016443   0.005719   2.875 0.004037 **
diasbp      -0.010932   0.006591  -1.659 0.097192 .
sho1         1.194531   0.770382   1.551 0.121005
chf1         0.962399   0.282778   3.403 0.000666 ***
av31         1.388309   0.952785   1.457 0.145087
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 483.42  on 351  degrees of freedom
Residual deviance: 364.47  on 345  degrees of freedom
AIC: 378.47

Number of Fisher Scoring iterations: 4
```

*Figure 8.* Descriptive Statistics of Minimal Adequate Model on Training Data.

```
> table(round(predict(mamodel, train.data, type="response")), train.data$fstat)

     0   1
  0 154  49
  1  42 107
>
```

*Figure 9.* Confusion Matrix for Minimal Adequate Model on Training Data.

```
> mypredictions <- round(predict(mamodel, test.data, type="response"))
> table(mypredictions, test.data$fstat)

mypredictions  0  1
            0 74 13
            1 15 46
>
```

*Figure 10.* Confusion Matrix for Minimal Adequate Model on Test Data.

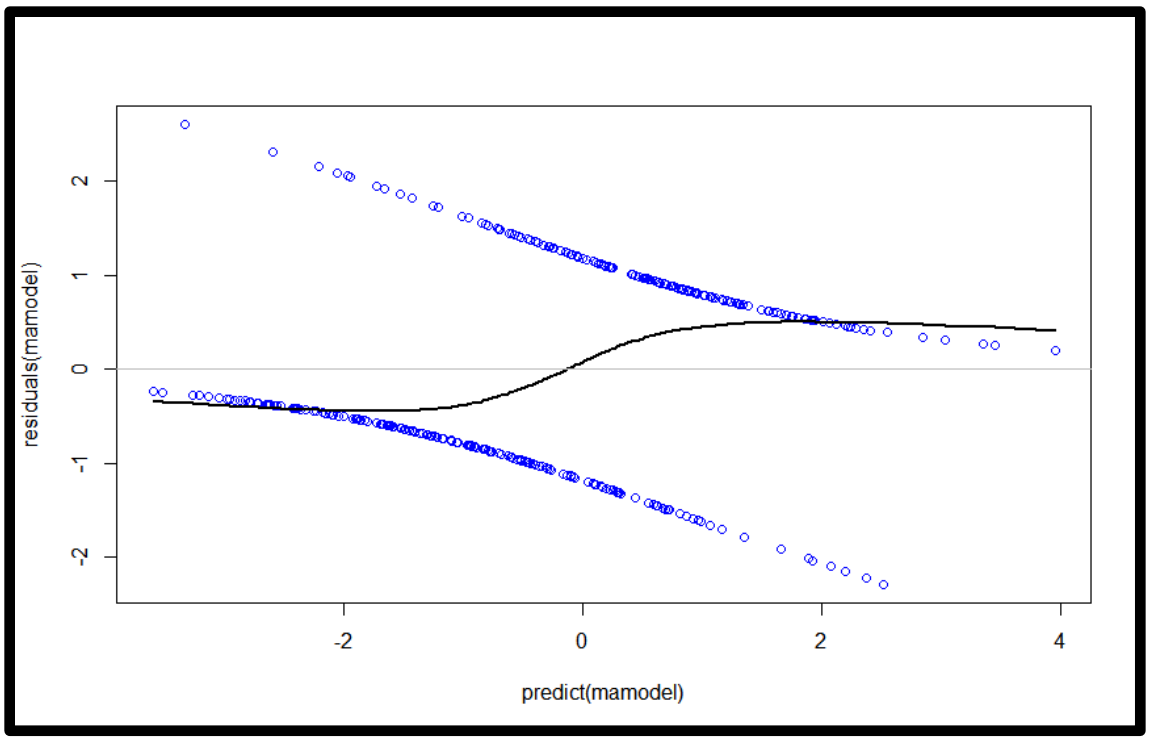


*Figure 11.* Residuals Plot for Minimal Adequate Model.

```
> print(model)

Naive Bayes Classifier for Discrete Predictors

Call:
naiveBayes.default(x = X, y = Y, laplace = laplace)

A-priori probabilities:
Y
        0         1
0.5568182 0.4431818

Conditional probabilities:
   age
Y      [30, 55)   [55, 65)   [65, 73)   [73, 80)   [80, 85)   [85,104]
  0 0.24489796 0.27551020 0.19387755 0.10204082 0.12755102 0.05612245
  1 0.05769231 0.05769231 0.14102564 0.26923077 0.19230769 0.28205128

   gender
Y           0         1
  0 0.6683673 0.3316327
  1 0.5128205 0.4871795

   hr
Y    [ 35, 65) [ 65, 75) [ 75, 86) [ 86, 97) [ 97,111) [111,186]
  0 0.2142857 0.1938776 0.1938776 0.1428571 0.1428571 0.1122449
  1 0.1089744 0.1089744 0.1346154 0.1987179 0.2435897 0.2051282

   sysbp
Y    [ 57,116) [116,131) [131,142) [142,158) [158,175) [175,244]
  0 0.1377551 0.1785714 0.1377551 0.2040816 0.1734694 0.1683673
  1 0.2179487 0.2051282 0.1538462 0.1153846 0.1282051 0.1794872

   diasbp
Y    [  6, 60)  [ 60, 71)  [ 71, 80)  [ 80, 88)  [ 88,100)  [100,198]
  0 0.11734694 0.17346939 0.13265306 0.22959184 0.16326531 0.18367347
  1 0.23717949 0.20512821 0.19871795 0.09615385 0.12179487 0.14102564
```

*Figure 12.* Probabilities of Variables Using Naïve Bayes Classification Model.

```
> table(predict(model, train.data), train.data$fstat)

     0   1
  0 155  41
  1  41 115
```

*Figure 13.* Confusion Matrix for Naïve Bayes Model on Training Data.

```
> table(predict(model, test.data), test.data$fstat)

    0  1
  0 76 16
  1 13 43
```

*Figure 14.* Confusion Matrix for Naïve Bayes Model on Test Data.

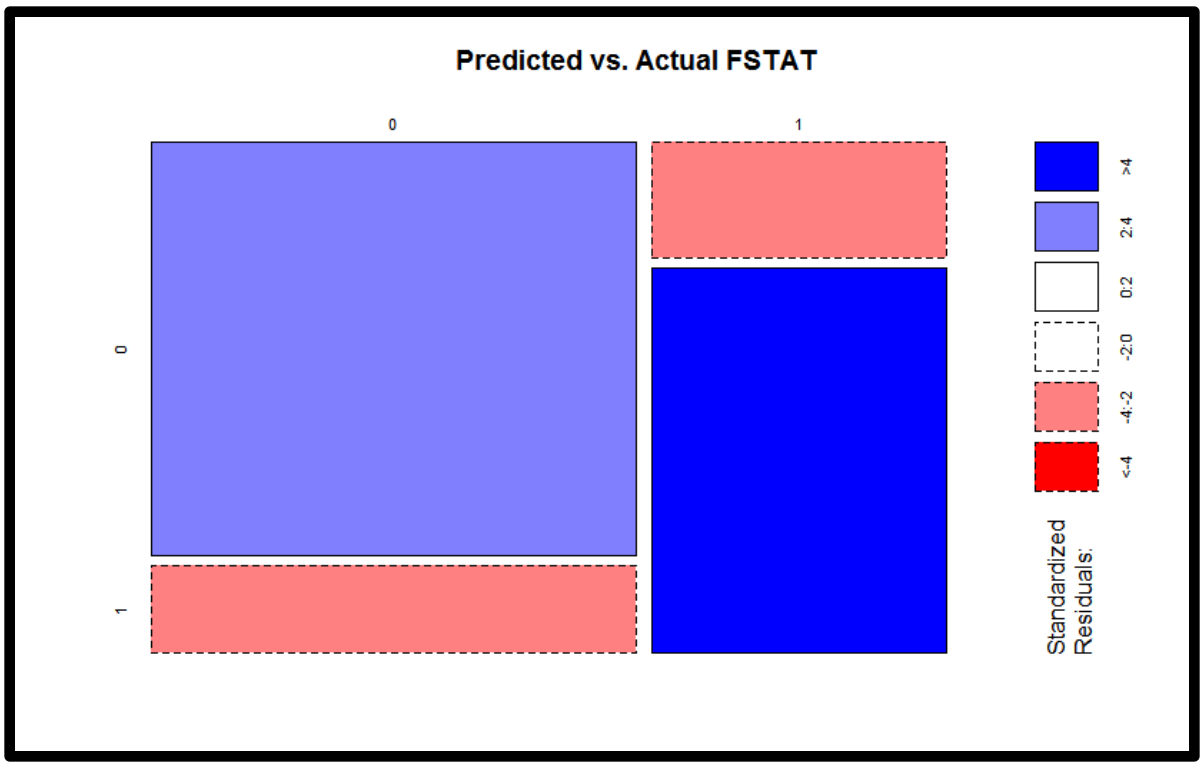


*Figure 15.* Naïve Bayes Mosaic Plot for Predicted vs. Actual "FSTAT" Values