**Predicting Heart Failure using Machine Learning**

(binary classification problems)

IST 687 – Introduction to Data Science

Adrian Torres

Paige Madison

Frank Williams

**Table of Contents**

Introduction .........................................................................................

Business Questions ..............................................................................

Data Cleanse/Munge/Preparation...........................................................

Descriptive Statistics & Visualizations ....................................................

Data Modeling & Visualizations...............................................................

Summary................................................................................................

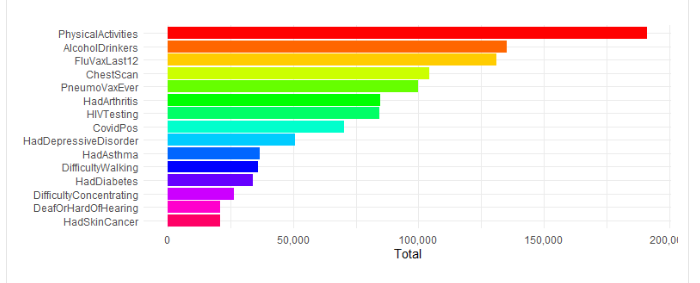
Appendix – R Code...................................................................................

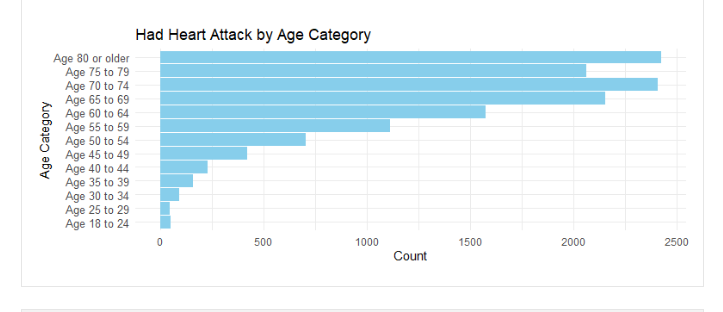
**Introduction**

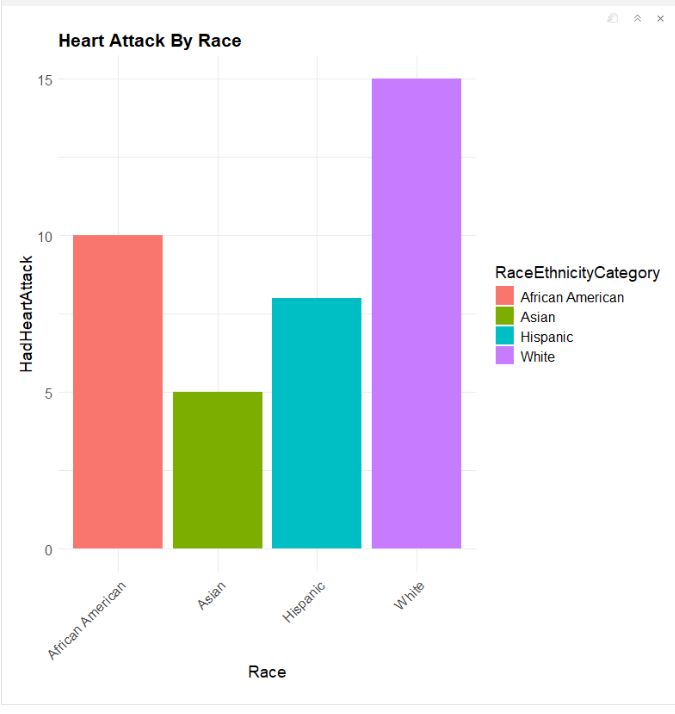
**Business Questions**

**Data Cleanse/Munge/Preparation**

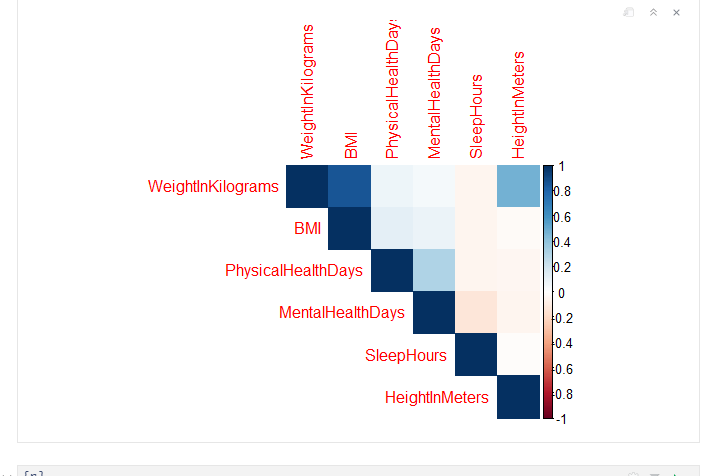
**Descriptive Statistics & Visualizations**





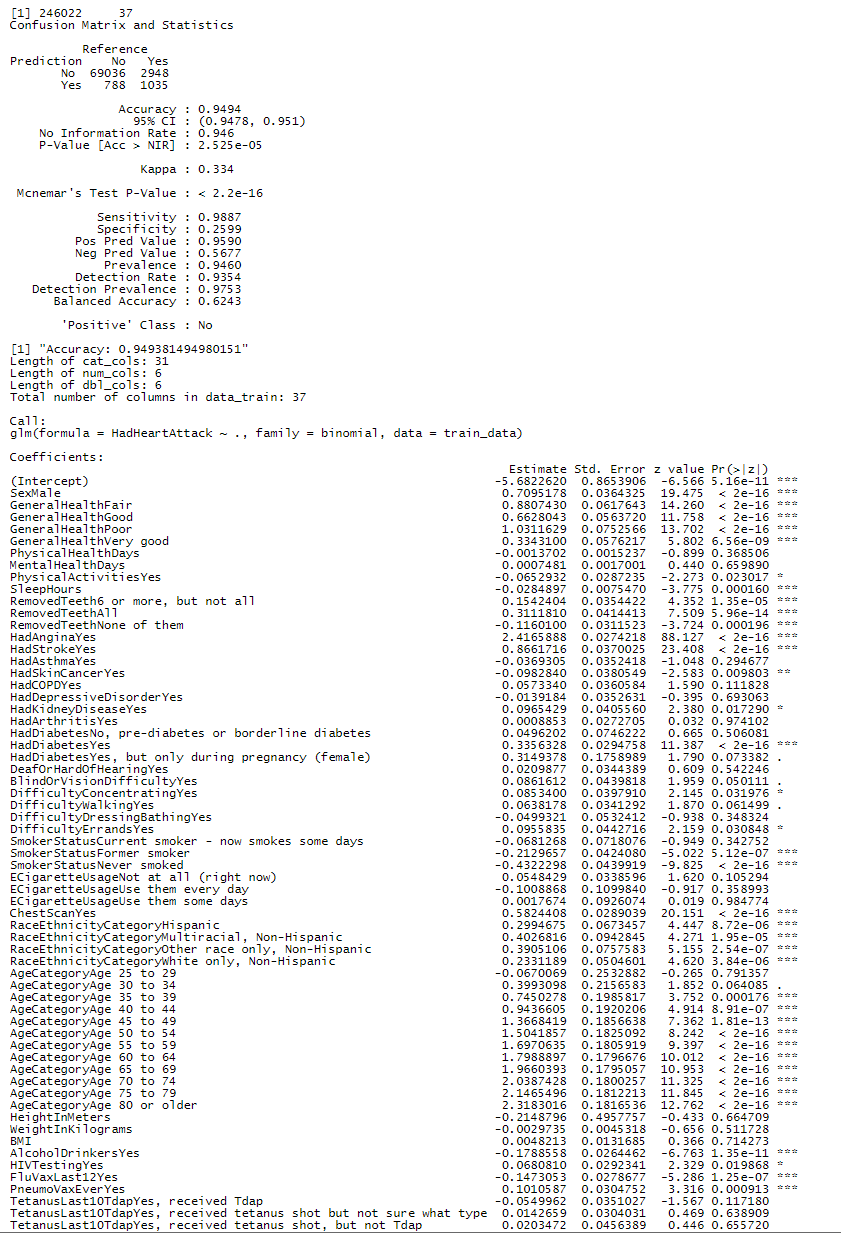


**Correlation Statistics -**



Data Modeling & Visualizations

Regression Model -



Summary

Machine Learning Algorithms Used and Explanation

A screenshot of a computer program

Description automatically generated

First, relevant columns were selected and excluded, and data types of each column were verified. Specifically, the column “State” was excluded as it didn’t appear to be relevant to predicting heart attacks.

A screenshot of a computer program

Description automatically generated

After, a data subset was created, which converted categorical variables to usable factors, and Nas were removed from that subset. Next, the data was split using 70% of the data to train and 30% to test. Then, a logistic regression model was fitted using the training data, which produced the output below:

A screenshot of a computer

Description automatically generated

Figure 1: Logistic Regression Output

# Referenced Equations

The model output for the logistic regression showed the following:

* There were 69,036 true negatives (TN) or correctly predicted “No” values.
* There were 1,035 true positives (TP) or correctly predicted "Yes" values.
* There were 788 false positives (FP) or incorrectly predicted "Yes" values.
* There were 2,948 false negatives (FN) or incorrectly predicted "No" values.

The **Accuracy metric** was 0.9494 or 94.94%, for our logistic regression model, which meant that it correctly predicted heart attacks 94.94% of the time. The **Balanced Accuracy** metric had a value of 62.43%, which suggested that even with an imbalanced data set, it still managed to score above a score of 0.5, which would be the equivalent of guessing. Additionally, because Balanced Accuracy considers both the sensitivity (i.e., the true positive rate) and the specificity (i.e., the true negative rate), it is a realistic metric for accuracy or the model’s ability to predict correctly.

The **No Information Rate** (**NIR**), or the proportion of predicted values that belong to the majority class, was 94.6%, which meant of the predicted outcomes 94.6% was the majority class (i.e., the “No” class). The **P-Value [Acc>NIR]** has a value of suggests that the model’s No Information Rate was significantly different from the Accuracy value. The p-value that compares both the Accuracy and No Information Rate, assesses if the model’s performance would be significantly better than would be expected by predicting the “No” response or the majority class in most cases. Since the p-value was significantly lower than 0.05, it suggests that the observed difference between the Accuracy and NIR, is not due to chance or randomly guessing, which makes the Accuracy Metric meaningful.

**Cohen’s Kappa value** considers the possibility of agreement between the actual and predicted classes happening by chance and has a scale from -1 to 1. Our model had a Cohen’s Kappa value of 0.334, which suggested fair agreement between the actual and predicted classes.

The **Mcnemar’s Test’s p-value** had a value of which suggested that there were significant differences between the observed discordant pairs (i.e., the instances where the predicted values and actual values are different or the FP +FN) and expected discordant pairs (i.e., the expected complete agreement between the predicted and actual values or the FP + FN), the latter value is considered under the assumption that our model and a hypothetical copy of our model (i.e., the copy of the model where there is no difference between the actual and predicted values) performed equally. In summary, it means that the disagreement between actual and predicted values, is not by chance alone, and there is possibly systematic bias due to the imbalanced dataset.

The **Detection Rate** or the proportion of true positive instances (i.e., the number of “No” values), which were detected by the model, has a rate of 0.9354, which indicated that the model detected 93.54% of positive instances.

The **Prevalence**, or the measure of the positive class in the dataset, or the amounts of “No” in our data set, was 0.946, which indicates that 94.6% of our instances belonged to the positive class. The **Detection Prevalence** is the predicted positive cases by the model and has a value of 0.9753 or 97.53%.

We also ran a decision tree algorithm to see if there was an alternate model that could produce similar results.

A screenshot of a computer

Description automatically generated

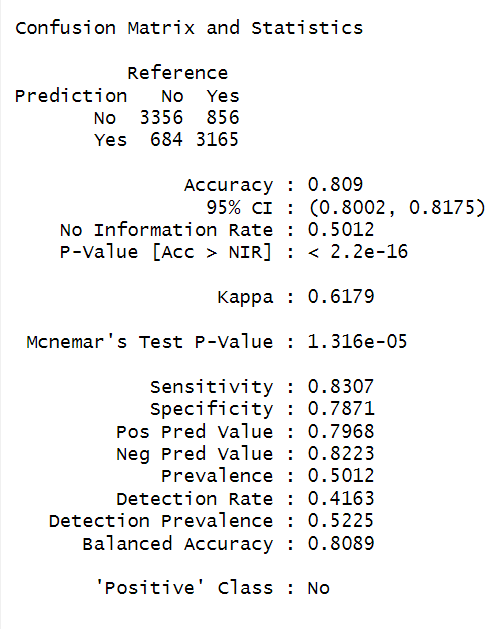
Overall, the model produced a similar output with a slightly lower accuracy score, so we kept the logistic regression model. We also tried a **5-fold cross-validation** to assess the performance of the model and got the following output:

A screenshot of a computer screen

Description automatically generated

As can be seen by the accuracy metric, using **k-folds** did not increase the accuracy score, but did increase the time it took to fit the model using the subset version of our data.

Lastly, we tried to **under sample** the “No” response, by identifying the major and minority class, calculating the numbers of samples in each class, **undersampling** the “No” class (i.e., the “No” response), combining the undersampled majority class with the minority class, shuffling and splitting the newly created dataset (i.e., the dataset with the undersampled “No” response). Afterwards, the model was fitted, and its accuracy and confusion matrix were printed. The following is the output from **undersampled “No”** **model**:



As seen in the output, the accuracy metric dropped to 80.9% and all other metrics were also affected. In summary, although the accuracy metric dropped, undersampling the “No” response reflected a more realistic representation of the Accuracy metric as evidenced by the agreement between the Accuracy and Balanced Accuracy for the undersampled version of the model.

Appendix – R Code