Capstone CYO Project: HarvardX PH125.9x

Authored By: Brad Rossen

Project Duration: May 21st, 2021 - May 30th, 2021

Project Background

For this project, we are tasked with applying machine learning techniques that go beyond standard linear regression to a publicly available dataset. A successful version of this project includes; - an introduction/overview/executive summary section that describes the dataset and variables, and summarizes the goal of the project and key steps that were performed - a methods/analysis section that explains the process and techniques used, including data cleaning, data exploration and visualization, insights gained, and your modeling approaches (you must use at least two different models or algorithms) - a results section that presents the modeling results and discusses the model performance - a conclusion section that gives a brief summary of the report, its potential impact, its limitations, and future work

Project Introduction

For this project, we chose a publicly available dataset from Kaggle ("https://www.kaggle.com/fedesoriano/stroke-prediction-dataset") that pertains to determining predictors of stroke victims. According to the Mayo Clinic, 'a stroke occurs when the blood supply to part of your brain is interrupted or reduced, preventing brain tissue from getting oxygen and nutrients. Brain cells begin to die in minutes'. MayoClinic Link According to the CDC, stroke is a leading cause of death among Americans and roughly ever 4 minutes someone in the USA (United States of America) dies from a stroke. CDC Link This dataset is used to predict whether a patient is likely to get stroke based on the input parameters like gender, age, various diseases, and smoking status. Each row in the data provides relevant information about the patient. As this issue is so prominent around the world and affects so many families every year, I was quite interested to explore this particular dataset. Obesity is thought of as the biggest threat facing the modernized world and this issue goes hand in hand with an increase on stroke cases.

Installing Required Databases & Setting Digits

```
if (!require(caret)) install.packages('caret')
library(caret)
if (!require(class)) install.packages('class')
library(class)
if (!require(corrplot)) install.packages('corrplot')
library(corrplot)
if (!require(data.table)) install.packages('data.table')
library(data.table)
if (!require(dplyr)) install.packages('dplyr')
library(dplyr)
if (!require(e1071)) install.packages('e1071')
library(e1071)
```

```
if (!require(ggplot2)) install.packages('ggplot2')
library(ggplot2)
if (!require(httr)) install.packages('httr')
library(httr)
if (!require(knitr)) install.packages('knitr')
library(knitr)
if (!require(lubridate)) install.packages('lubridate')
library(lubridate)
if (!require(randomForest)) install.packages('randomForest')
library(randomForest)
if (!require(readxl)) install.packages('readxl')
library(readxl)
if (!require(rpart)) install.packages('rpart')
library(rpart)
if (!require(ROSE)) install.packages('ROSE')
library(ROSE)
if (!require(tidyverse)) install.packages('tidyverse')
library(tidyverse)

options(digits = 4)
```

Project Initiation

To load out dataset, we use the following line of code. The dataset has been loaded into a GitHub account and is retrieved as follows:

```
strokedata <- read.csv("https://raw.githubusercontent.com/RossenBradley/stroke-prediction/main/strokeda
```

Methods & Analysis Section

For this project, we will develop and use 2 distinct models to predict the likelihood of a patient suffering a stroke. In the following section we will successfully complete the following tasks: - cleaning the dataset - exploring the data through summary tables and graphics - discuss insights as they are unearthed through data exploration techniques - clearly define the modeling approaches - work through each of the modeling approaches in a systematic way. #### Dataset Information This dataset contains 12 distinct columns of information which will be shown below using some code but are explained here as follows:

```
class(strokedata)
```

```
## [1] "data.frame"
```

We run the class() function to determine the 'class' of our dataset. As we can see, this dataset falls into the data.frame class which is what we want to work with in this project Next, we run the glimpse() function to obtain some important information regarding the breakdown of our dataset and the respective columns

```
glimpse(strokedata)
```

```
## Rows: 5,110
## Columns: 12
## $ id                <int> 9046, 51676, 31112, 60182, 1665, 56669, 53882, 10434~
```

```
## $ gender            <chr> "Male", "Female", "Male", "Female", "Female", "Male"~
## $ age               <dbl> 67, 61, 80, 49, 79, 81, 74, 69, 59, 78, 81, 61, 54, ~
## $ hypertension      <int> 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 1~
## $ heart_disease     <int> 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 1, 1, 0, 1, 0~
## $ ever_married      <chr> "Yes", "Yes", "Yes", "Yes", "Yes", "Yes", "Yes", "No~
## $ work_type         <chr> "Private", "Self-employed", "Private", "Private", "S~
## $ Residence_type    <chr> "Urban", "Rural", "Rural", "Urban", "Rural", "Urban"~
## $ avg_glucose_level <dbl> 228.69, 202.21, 105.92, 171.23, 174.12, 186.21, 70.0~
## $ bmi               <chr> "36.6", "N/A", "32.5", "34.4", "24", "29", "27.4", "~
## $ smoking_status    <chr> "formerly smoked", "never smoked", "never smoked", "~
## $ stroke            <int> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1~
```

Using the function above, we get a breakdown of the dataset's input (and target) variables and we see our 12 distinct column headers. The explanation of these 12 variables is as follows: 1. id: unique person number identifier 2. gender: "Male", "Female" or "Other" 3. age: age of the patient. ranges from 0 - 82. 4. hypertension: 0 if the patient doesn't have hypertension, 1 if the patient has hypertension 5. heart_disease: 0 if the patient doesn't have any heart diseases, 1 if the patient has a heart disease 6. ever_married: "No" or "Yes" 7. work_type: "children", "Govt_jov", "Never_worked", "Private" or "Self-employed" 8. Residence_type: "Rural" or "Urban" 9. avg_glucose_level: average glucose level in blood. ranges from 55 - 272 10. bmi: body mass index. ranges from 10 - 98 11. smoking_status: "formerly smoked", "never smoked", "smokes" or "Unknown"* 12. stroke: 1 if the patient had a stroke or 0 if not

Next, we run the summary() function as our last function to display key information regarding the makeup of the dataset. Personally speaking, I like to run these three functions right away when initializing a new project as it gives a lot of information and can help dictate the best way to proceed when starting the data transformations and data cleaning.

```
summary(strokedata)
```

```
##        id            gender                age          hypertension
##  Min.   :   67   Length:5110        Min.   : 0.08   Min.   :0.0000
##  1st Qu.:17741   Class :character   1st Qu.:25.00   1st Qu.:0.0000
##  Median :36932   Mode  :character   Median :45.00   Median :0.0000
##  Mean   :36518                      Mean   :43.23   Mean   :0.0975
##  3rd Qu.:54682                      3rd Qu.:61.00   3rd Qu.:0.0000
##  Max.   :72940                      Max.   :82.00   Max.   :1.0000
##  heart_disease   ever_married        work_type         Residence_type
##  Min.   :0.000   Length:5110        Length:5110        Length:5110
##  1st Qu.:0.000   Class :character   Class :character   Class :character
##  Median :0.000   Mode  :character   Mode  :character   Mode  :character
##  Mean   :0.054
##  3rd Qu.:0.000
##  Max.   :1.000
##  avg_glucose_level     bmi            smoking_status         stroke
##  Min.   : 55.1     Length:5110        Length:5110        Min.   :0.0000
##  1st Qu.: 77.2     Class :character   Class :character   1st Qu.:0.0000
##  Median : 91.9     Mode  :character   Mode  :character   Median :0.0000
##  Mean   :106.2                                           Mean   :0.0487
##  3rd Qu.:114.1                                           3rd Qu.:0.0000
##  Max.   :271.7                                           Max.   :1.0000
```

Data Exploration

In this section of the report, the dataset will be explored and key takeaways will be presented from that exploration. Check the number of Null values in the dataset. With no Null values in the data, we don't need to clean the data to account for this potential issue.

```
sum(is.na(strokedata))
```

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

Checking our dataset participants by gender

```
table(strokedata$gender)
```

```
##
## Female   Male  Other
##   2994   2115      1
```

From the information above, we can see that one person in the dataset chose "Other" as the gender. We need to clean this dataset and remove this row of data as this is obviously an outlier and removing this row only trims our dataset by 0.02%.

```
strokedata<- strokedata[strokedata$gender !="Other", ]
table(strokedata$gender)
```

```
##
## Female   Male
##   2994   2115
```

We can now see that the outlier gender of 'Other' has been taken out of our dataset. After filtering out our outlier gender variable, we can better understand the breakdown of our dataset by gender, ever married, residence_type etc.

```
prop.table(table(strokedata$gender))
```

```
##
## Female   Male
##  0.586  0.414
```

```
prop.table(table(strokedata$ever_married))
```

```
##
##     No    Yes
## 0.3437 0.6563
```

```
prop.table(table(strokedata$Residence_type))
```

```
##
##  Rural  Urban
## 0.4919 0.5081
```

We will continue to explore our dataset as follows by this time focusing on the breakdown of stroke victims by gender classification:

```
stroke <- strokedata %>% filter(stroke==1)
nostroke <- strokedata %>% filter(stroke==0)
tableinput <- ifelse(strokedata$stroke==0,"No_Stroke","Stroke")
table(strokedata$gender, tableinput)
```

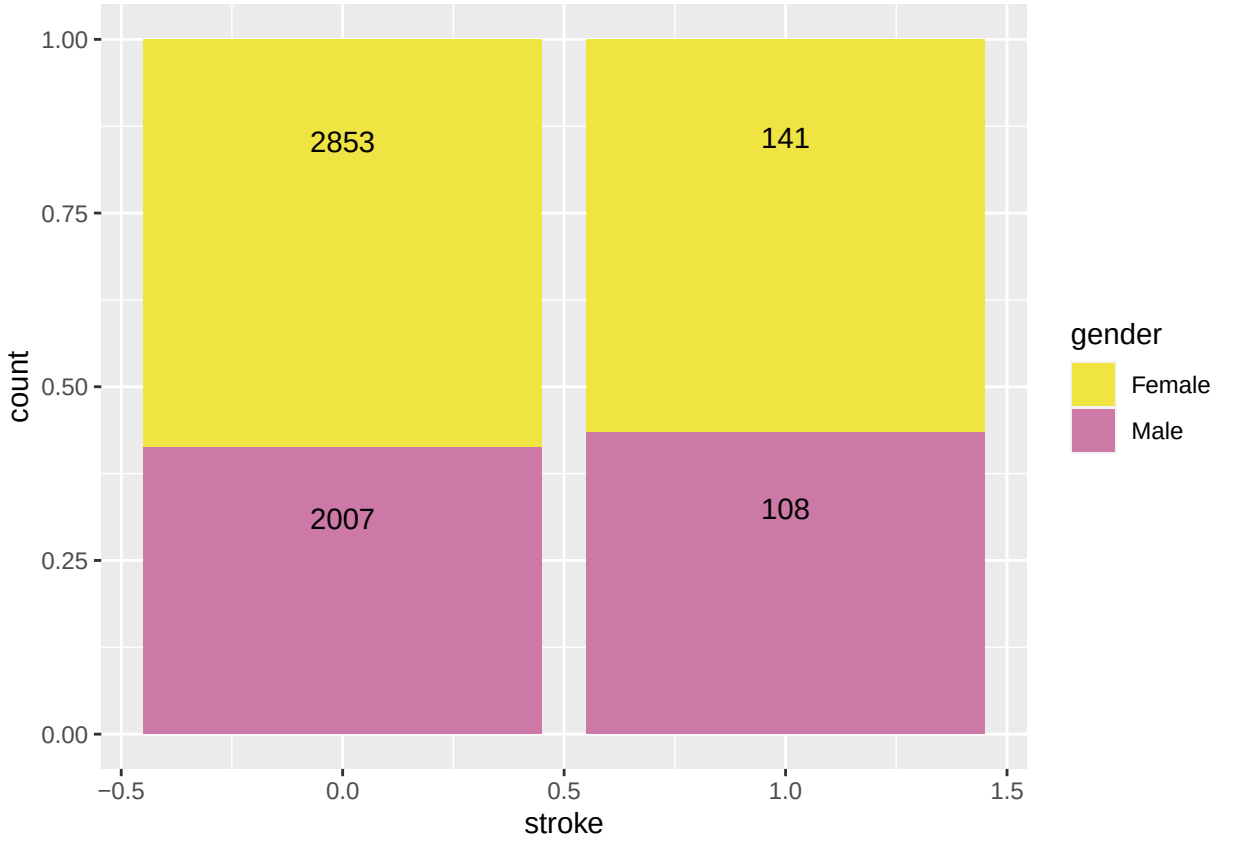
```
##         tableinput
##          No_Stroke Stroke
##   Female      2853    141
##   Male        2007    108
```

```
prop.table(table(strokedata$gender, tableinput))
```

```
##         tableinput
##          No_Stroke  Stroke
##   Female   0.55843 0.02760
##   Male     0.39284 0.02114
```

We see the results both in their raw form (using the table() function) and as proportions of the dataset using the prop.table() function.The table shows that we have 249 patients that were diagnosed with stroke which represents 4.9% of all people included in the dataset. This is made up of 141 Females and 108 Males. We can also report these values in the form of a bar plot, which is done below:

```
plot(ggplot(strokedata, aes(x = stroke, fill = gender)) +
       geom_bar(position = "fill")+ stat_count(geom = "text", aes(label =
       stat(count)), position = position_fill(vjust = 0.75), color = "black")
       + scale_fill_manual(values = c("#F0E442", "#CC79A7")))
```



We are then going to explore the relationship/breakdown of stroke cases by married status. The first line of code tells us the raw totals of how many patients make up each distinction. For example, we have 220 married people who developed a stroke. Conversely, using the second line of code, we can see that approx 4.3% of all members of our dataset were married and suffered a stroke.

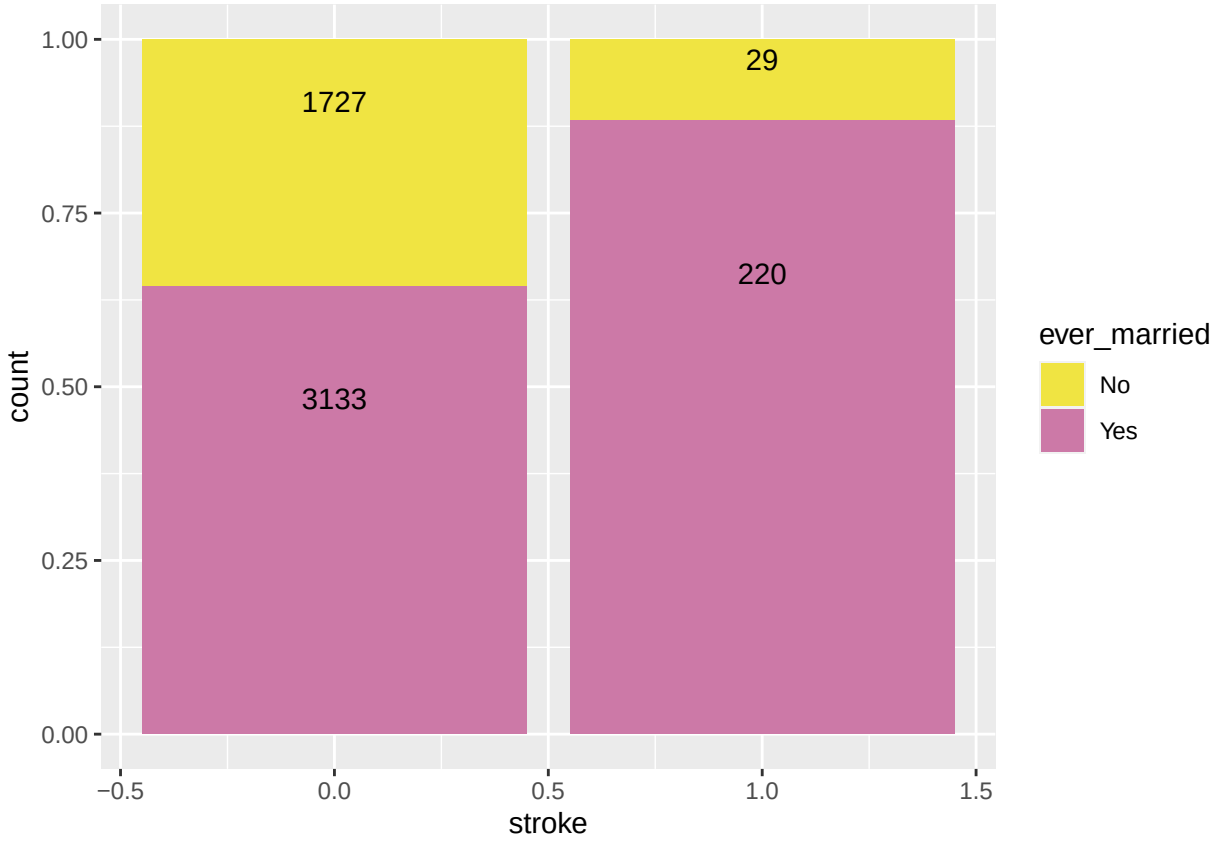
```
table(strokedata$ever_married, tableinput)
```

```
##      tableinput
##       No_Stroke Stroke
##   No       1727     29
##   Yes      3133    220
```

```
prop.table(table(strokedata$ever_married, tableinput))
```

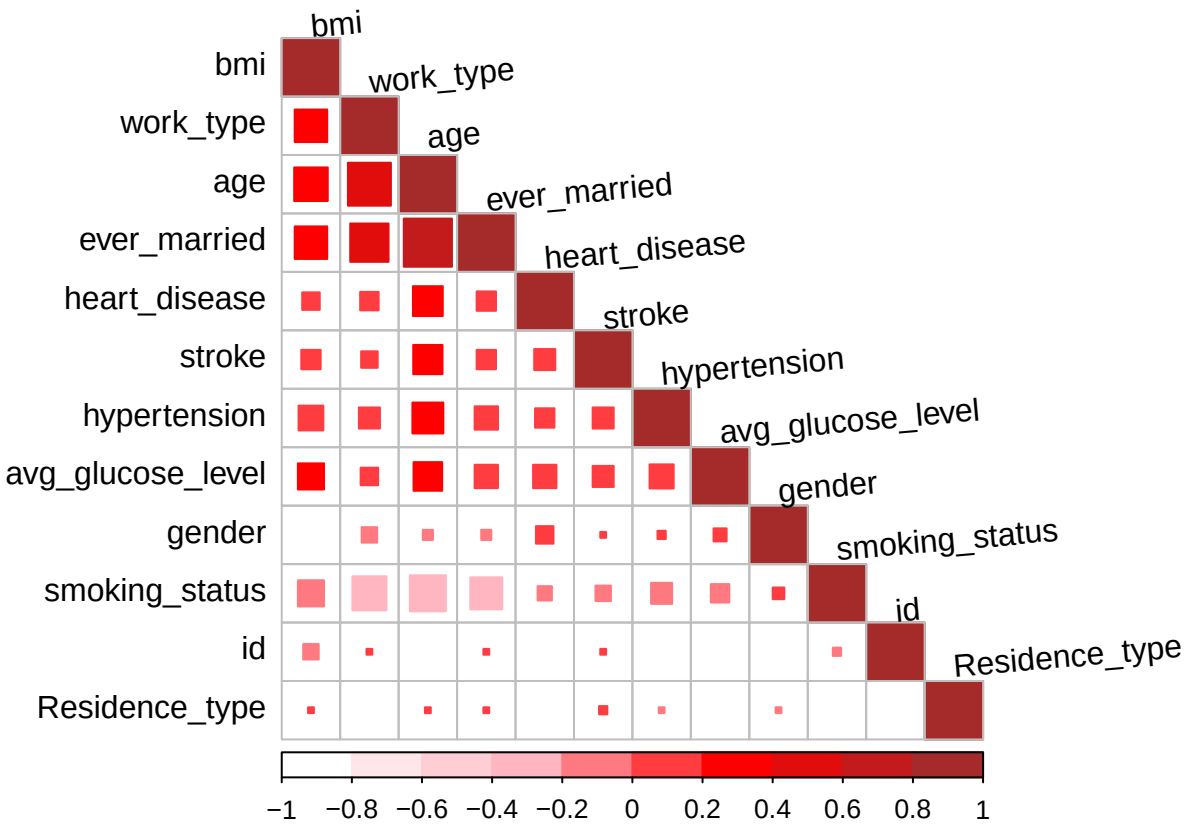
```
##      tableinput
##       No_Stroke   Stroke
##   No   0.338031 0.005676
##   Yes  0.613232 0.043061
```

```
plot(ggplot(strokedata, aes(x = stroke, fill = ever_married)) +
       geom_bar(position = "fill") + stat_count(geom = "text", aes(label =
       stat(count)), position = position_fill(vjust = 0.75), color = "black") +
       scale_fill_manual(values = c("#F0E442", "#CC79A7")))
```



Next, we are going to explore the correlation of the variables in the dataset. This is done as follows:

```
corrplot(round(cor(data.frame(data.matrix(strokedata))),2), method = "square",
         col= colorRampPalette(c("white","lightpink", "red","brown"))(10),
         type = "lower", order = "hclust", tl.col = "black", tl.srt = 5)
```



Based on this correlation matrix, we can develop some questions of the dataset, which we can answer visually through additional data exploration. That is why the correlation piece is important for us as it helps us very easily see and understand the correlation between different dataset variables, and allows us to pick out which relationships to explore in further detail. For example, through the correlation visual and through the following code, we can clearly see a relationship with age and stroke likelihood simply by plotting the mean:

```
mean(strokedata$age)
```

```
## [1] 43.23
```

```
mean(stroke$age)
```

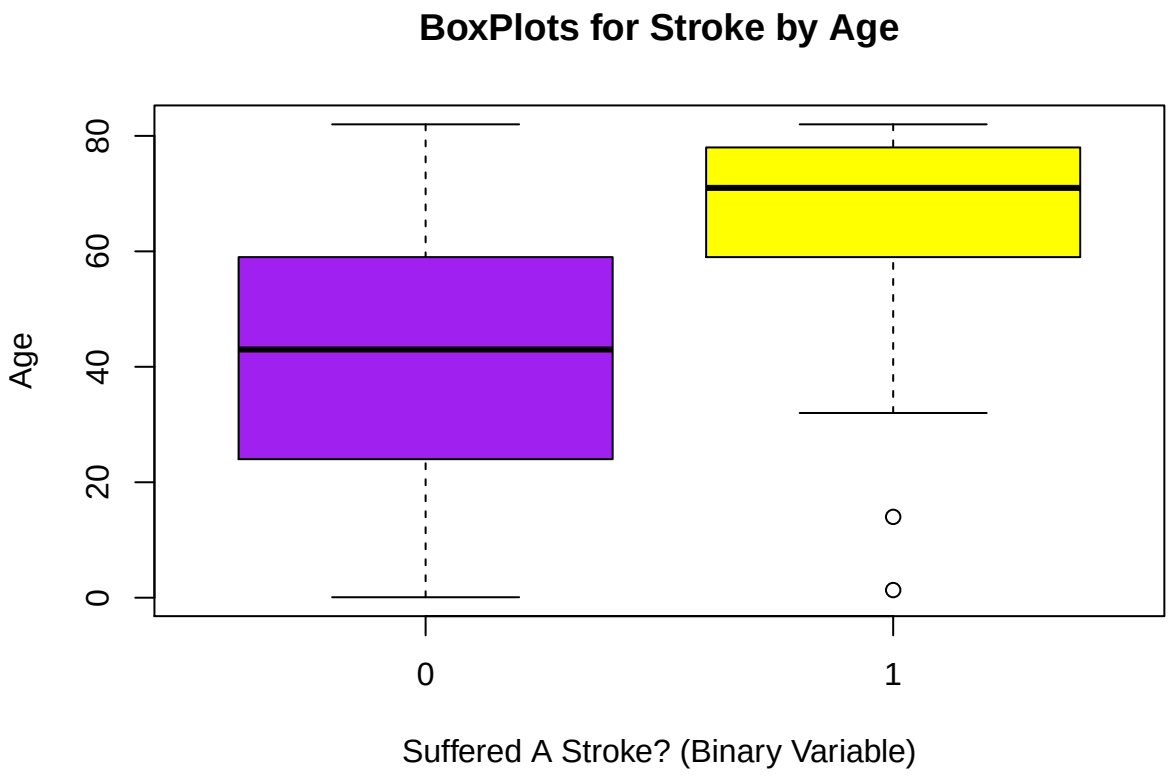
```
## [1] 67.73
```

```
mean(nostroke$age)
```

```
## [1] 41.97
```

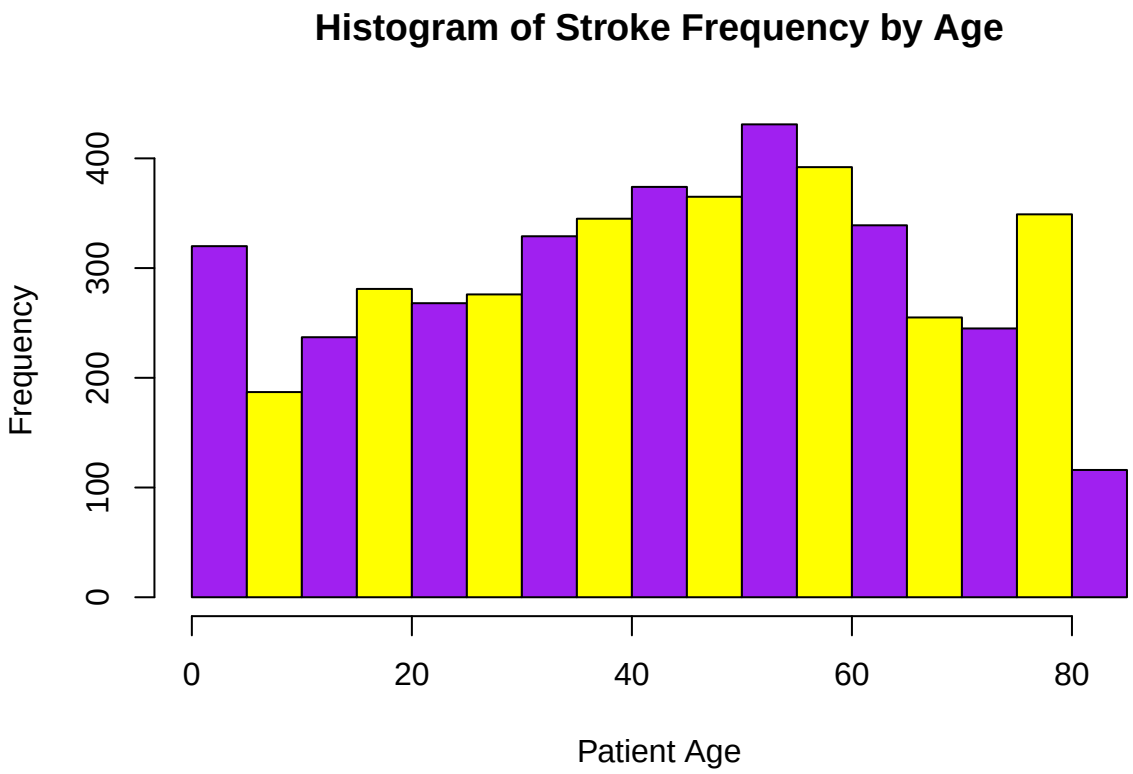
One key question revolves around the relationship between age and 'Stroke Status - (Stroke or No Stroke)

```
boxplot(age~stroke, data=strokedata, main="BoxPlots for Stroke by Age",
        xlab="Suffered A Stroke? (Binary Variable)", ylab="Age",
        col=(c("purple","yellow")))
```



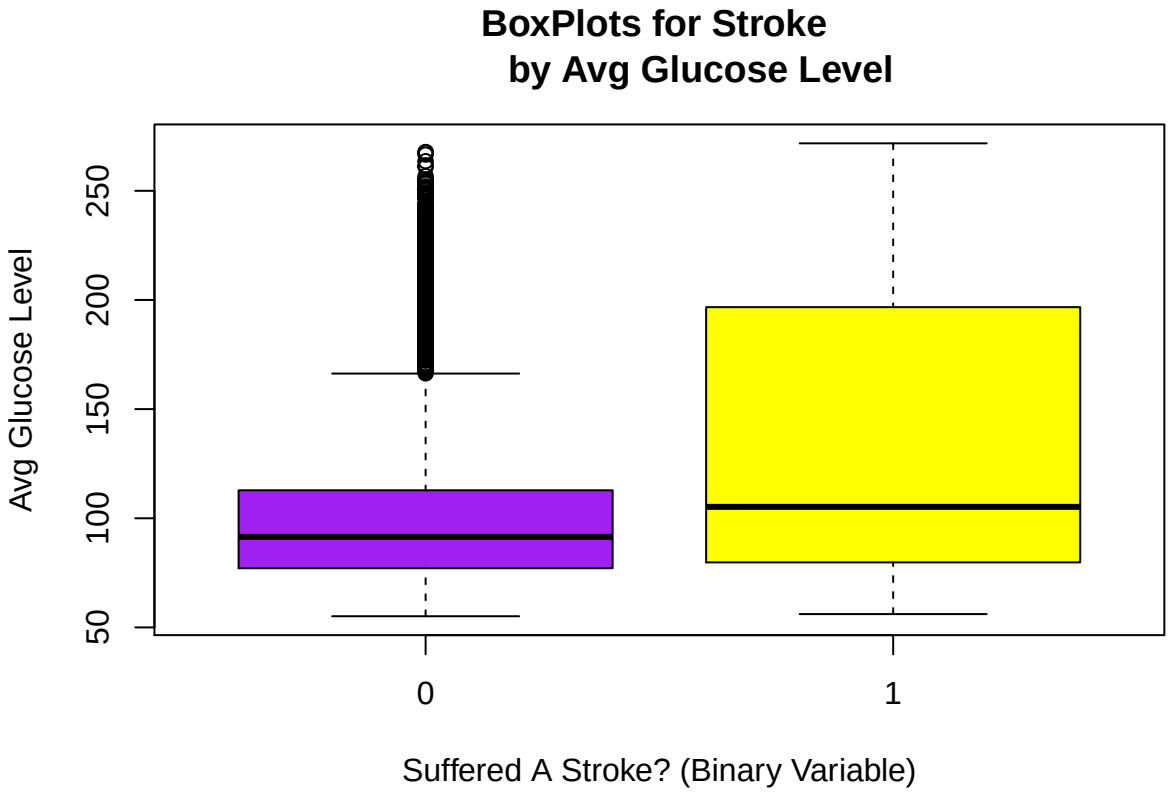
From the boxplot results above, we confirm the results from our correlation matrix showing a positive correlation between age and stroke likelihood. This means that the older the patient, the higher risk they inherently have to suffer a stroke.This can be shown in a different way, depending on reader preference, using a histogram as follows:

```
hist(strokedata$age, main="Histogram of Stroke Frequency by Age", xlab="Patient Age", ylab="Frequency",
```



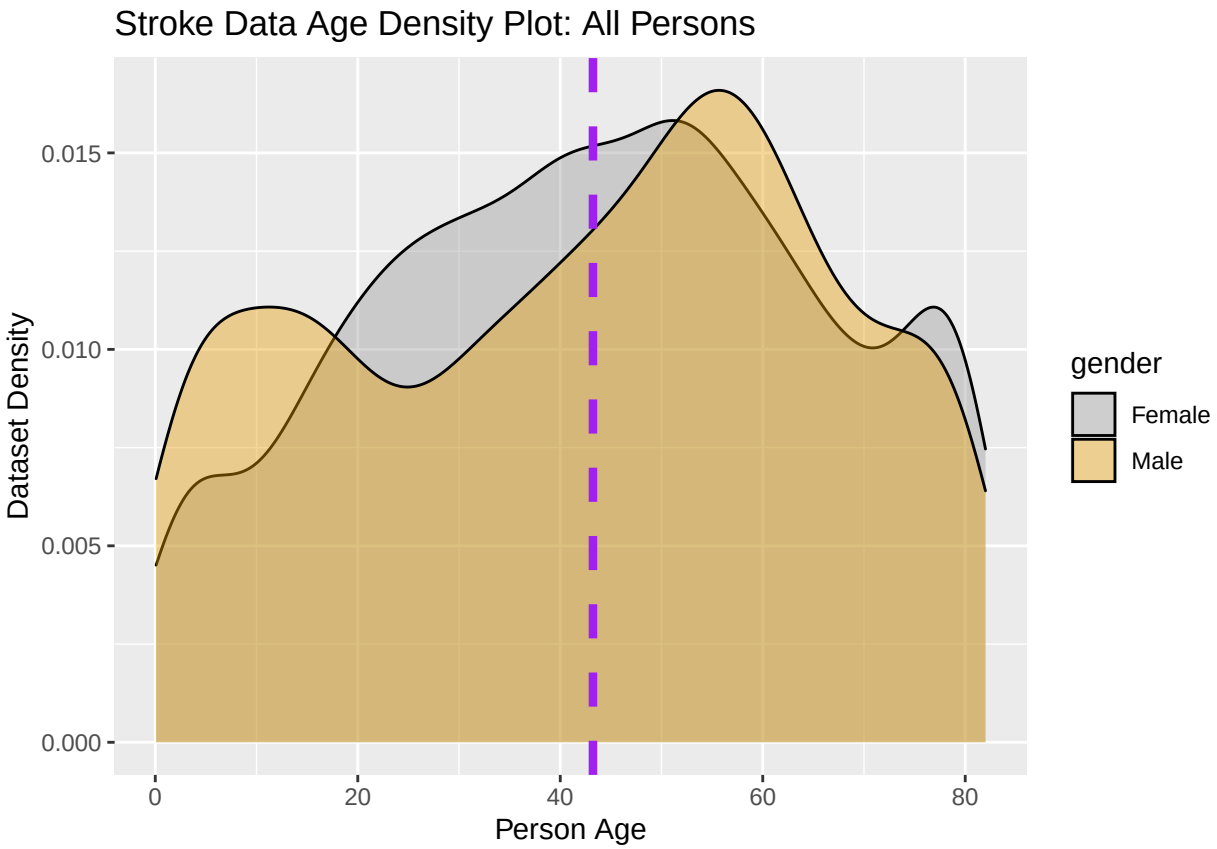
We can run these boxplots on a number of different variable combinations such as age versus average glucose level which is done as follows:

```
boxplot(avg_glucose_level~stroke, data=strokedata, main="BoxPlots for Stroke
        by Avg Glucose Level", xlab="Suffered A Stroke? (Binary Variable)",
        ylab="Avg Glucose Level",col=(c("purple","yellow")))
```



We can also analyze our dataset age distribution by using a density plot as follows; note that the vertical purple line represents the mean for that version of the dataset.

```
strokedata %>% ggplot(aes(x = age, fill = gender)) + geom_density(alpha = 0.4) +       geom_vline(aes(x
  size = 1.5) + labs(title = "Stroke Data Age Density Plot: All Persons", x =
  "Person Age", y = "Dataset Density") +scale_fill_manual(values=c("#999999",
  "#E69F00", "#56B4E9"))
```



And we can run the same plot for only persons in our dataset who have suffered a stroke as follows and compare it to the plot above:

```
stroke %>% ggplot(aes(x = age, fill = gender)) + geom_density(alpha = 0.4) + geom_vline(aes(xintercept=r
    size = 1.5) + labs(title = "Age Density Plot: Suffered Stroke", x = "Person
    Age", y = "Dataset Density") +scale_fill_manual(values=c("#999999",
    "#E69F00", "#56B4E9"))
```

In conclusion, for this portion of the data exploration, we have gone into depth to explore and show the relationships between the variables within our dataset. This is useful information as we push forward towards building out a model that will allow us to predict whether a person in our dataset will suffer a stroke or not suffer a stroke. Again, there are a large number of connections between these variables that can be explored (stroke likelihood vs martial status, stroke likeilihood versus presence of hypertension etc) but the above diagrams show how we make these connections to gather useful insights into the makeup of the dataset. Our correlation matrix does a good job to highlight these connections between variables. ### Further Data Cleaning In this section, we will begin developing and discussing modeling techniques to help us predict whether or not a person in out dataset has suffered a stroke or not suffered a stroke. Prior to this, there are some quick data cleanups that we have to complete: 1. Remove variable ID - We do not need this variable and it is simply going to be in the way so we might as well remove it from our strokedata(), stroke() and nostroke() datasets which is done as follows:

```
stroke <- stroke[2:12]
nostroke <- nostroke[2:12]
strokedata <- strokedata[2:12]
```

2. Converting Binary Variables over to Yes/No

```
strokedata$hypertension<- factor(strokedata$hypertension, levels = c(0,1),
    labels = c("No", "Yes"))
strokedata$heart_disease<- factor(strokedata$heart_disease, levels = c(0,1),
    labels = c("No", "Yes"))
strokedata$stroke<- factor(strokedata$stroke, levels = c(0,1),
    labels = c("No", "Yes"))
```

3. Converting Variables over to Factor/Numeric Variables from Character Variables

```
strokedata$gender<-as.factor(strokedata$gender)
strokedata$ever_married<-as.factor(strokedata$ever_married)
strokedata$work_type<-as.factor(strokedata$work_type)
strokedata$Residence_type<-as.factor(strokedata$Residence_type)
strokedata$bmi<-as.numeric(strokedata$bmi)
```

```
## Warning: NAs introduced by coercion
```

```
strokedata$smoking_status<-as.factor(strokedata$smoking_status)
```

When we run the summary() function again, we see this populates a more elaborate and detailed description of our dataset compared to the results from our first summary() run at the beginning portion of the project.

```
summary(strokedata)
```

```
##     gender          age        hypertension heart_disease ever_married
##  Female:2994   Min.   : 0.08   No :4611     No :4833      No :1756
##  Male  :2115   1st Qu.:25.00   Yes: 498     Yes: 276      Yes:3353
##                Median :45.00
##                Mean   :43.23
##                3rd Qu.:61.00
##                Max.   :82.00
##
##          work_type    Residence_type avg_glucose_level      bmi
##  children     : 687   Rural:2513     Min.   : 55.1     Min.   :10.3
##  Govt_job     : 657   Urban:2596     1st Qu.: 77.2     1st Qu.:23.5
##  Never_worked :  22                  Median : 91.9     Median :28.1
##  Private      :2924                  Mean   :106.1     Mean   :28.9
##  Self-employed: 819                  3rd Qu.:114.1     3rd Qu.:33.1
##                                      Max.   :271.7     Max.   :97.6
##                                                        NA's   :201
##          smoking_status stroke
##  formerly smoked: 884   No :4860
##  never smoked   :1892   Yes: 249
##  smokes         : 789
##  Unknown        :1544
##
##
##
```

From the above results, we have two important issues to clean up before splitting our data into train/test data and beginning the build of our respective models. Both of these issues for us pertain to the 'BMI' variable. Firstly, our attention immediately turns to the 'BMI' input variable where we now have a reported 201 NA variables. This finding is confirmed, as we like to do, with the following line of code:

```
colSums(is.na(strokedata))
```

```
##            gender               age      hypertension     heart_disease
##                 0                 0                 0                 0
##      ever_married         work_type    Residence_type avg_glucose_level
##                 0                 0                 0                 0
##               bmi    smoking_status            stroke
##               201                 0                 0
```

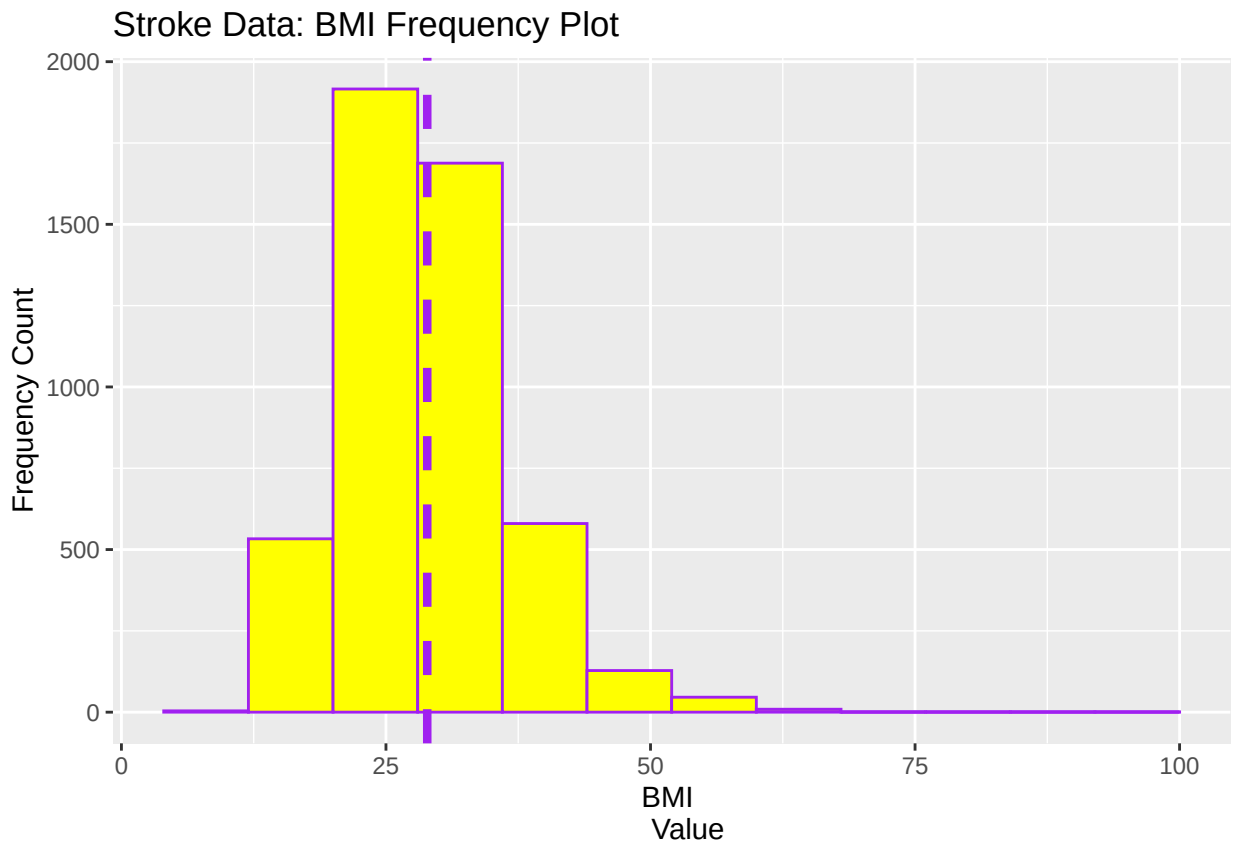
We run the na.omit() function to get rid of the NAs in the BMI column. This takes away the 201 NA values we have detected. We then re-run the previous function to confirm these rows have been taken out. This is confirmed and now our strokedata is down from 5109 rows to 4908 (a decrease of 3.9%).

```
full_strokedata <- strokedata
strokedata <- na.omit(strokedata)
colSums(is.na(strokedata))
```

```
##            gender               age      hypertension     heart_disease
##                 0                 0                 0                 0
##      ever_married         work_type    Residence_type avg_glucose_level
##                 0                 0                 0                 0
##               bmi    smoking_status            stroke
##                 0                 0                 0
```

Secondly, the summary function brings our attention to the BMI variable again through the reported values for: minimum, 1st quartile, mean, 3rd quartile and maximum. This maximum value is huge compared to the reported values for mean and 3rd quartile so we need to look into this to account for outlier issues in our dataset for this column. We can plot the BMI distribution for our dataset as follows:

```
ggplot(strokedata, aes(x=bmi)) + geom_histogram(color="purple", fill="yellow",
    binwidth = 8) + geom_vline(aes(xintercept=28.9), color="purple", linetype=
    "dashed", size=1.5)+ labs(title="Stroke Data: BMI Frequency Plot", x="BMI
    Value", y = "Frequency Count")
```



According to data published on the WHO (World Health Organization) website, BMI values range from

10 to 50. 15 would be 'Extreme Thinness' and 45 would be 'Extreme Obesity'. From this dataset, we can see that out maximum BMI is 97.6 which is an astronomical figure. We need to account for these extreme values in the BMI column after accounting for the NA issues with BMI column. Running the following line of code, we find that there are 35 BMI inputs that are greater than 54.9999. We allowed some freedom over the 45 maximum denoted by the WHO to account for those grossly obese people whose BMI are slightly above the expected maximum of 50. But this helps us account for the extremely unexpected variables like our pre-existing maximum of 97.6. The following line of code tells us that we have 35 values that are above 54.99999.

```
sum(strokedata$bmi > 54.99999)
```

```
## [1] 35
```

What we will do here to take away these 35 outlier variables is to first convert them to NA variables and then to remove those variables using that same function we did previously.

```
strokedata$bmi[strokedata$bmi > 54.99999] <- NA
strokedata <- na.omit(strokedata)
```

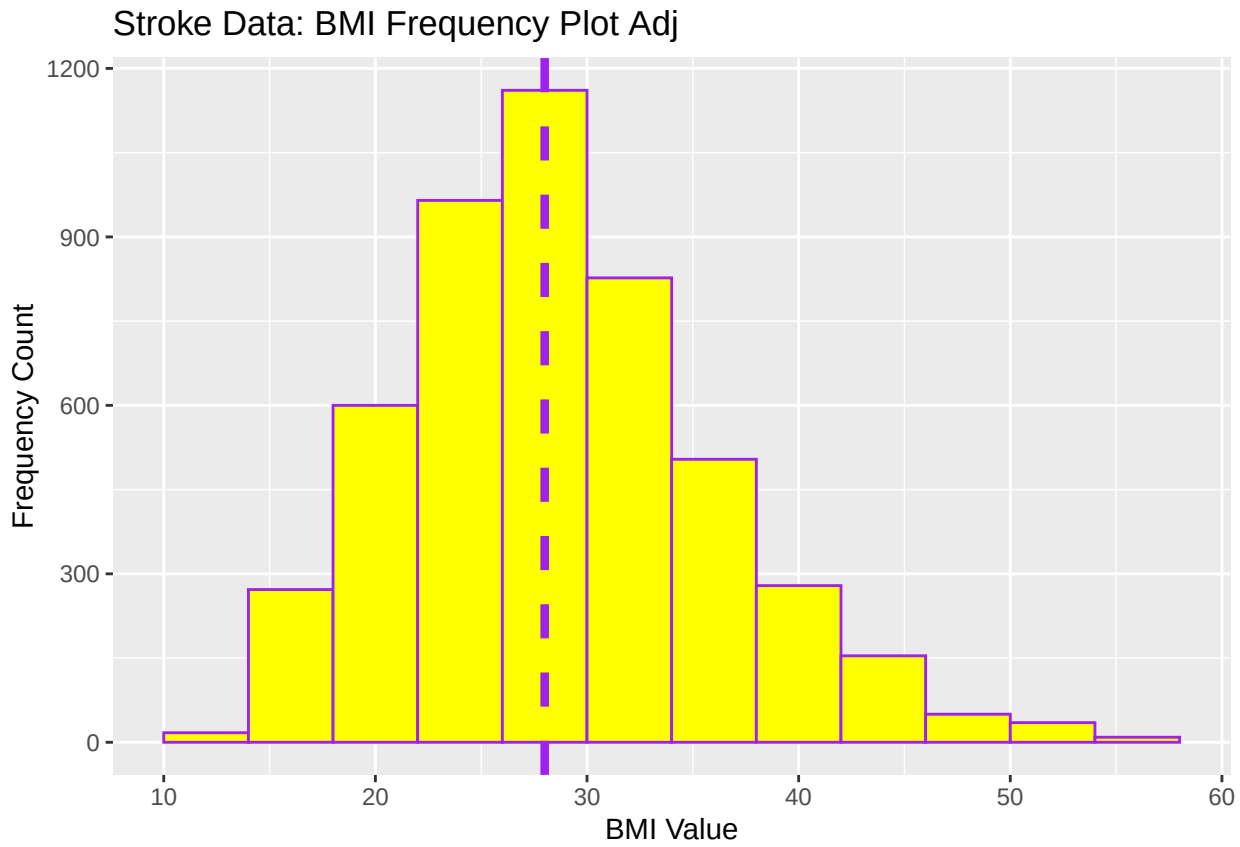
Using the glimpse() function, we see that our mutated dataset (removing NAs in the BMI and extreme outliers in the BMI column) consists of 4,873 rows. After completing this data cleaning, we are still left with 95.4% of our original dataset which consisted of 5,109 rows of data. This dataset should now allow us to make more meaningful conclusions through the following modeling.

```
nrow(strokedata)/nrow(full_strokedata)
```

```
## [1] 0.9538
```

Finally, we can re run our BMI plot to compare with the one we sat previously.

```
ggplot(strokedata, aes(x=bmi)) + geom_histogram(color="purple", fill="yellow",
    binwidth = 4) + geom_vline(aes(xintercept=28), color="purple", linetype=
    "dashed", size=1.5) + labs(title="Stroke Data: BMI Frequency Plot Adj",
    x="BMI Value", y = "Frequency Count")
```



Modeling Development In this section of the report, we are going to go through a number of different models that will be explored with regards to their validity in predicting our target variable, stroke. The first thing we are going to do is create our test and train datasets by splitting our cleaned strokedata dataset. The train-test split procedure is used to estimate the performance of machine learning algorithms when they are used to make predictions on data not used to train the model. We split our data into train and test splits to prevent your model from overfitting and to accurately evaluate your model. 70/30 and 80/20 splits are common in the data science world and allow us more freedom than if we chose a 90/10 split. 90/10 splits are more commonly reserved for very large datasets where the 10% still captures a high number of rows/information and the variance of the overall dataset. For our purposes, because of the size of the dataset, we are going with the 70/30 split for training and testing which is also the most common data partition in the data science world. This will give us 3412 rows of data in our training set and 1461 in our testing set.

```
set.seed(123)
split<- sample(nrow(strokedata), nrow(strokedata)*0.7)
strokedata_train <- strokedata[split,]
strokedata_test <- strokedata[-split,]
```

Next, we are going to create some proposition tables on our original dataset, the training dataset and the testing dataset to see the differences in splits for the target variable, which is 'stroke'.

```
prop.table(table(strokedata_train$stroke))
```

```
##
##      No     Yes
## 0.95749 0.04251
```

```
prop.table(table(strokedata_test$stroke))
```

```
##
##      No     Yes
## 0.95691 0.04309
```

```
prop.table(table(strokedata$stroke))
```

```
##
##      No     Yes
## 0.95732 0.04268
```

Since these proportions are very similar, we can trust that there won't be an influence from our target variable when using the training set and applying that to the test set. ### Modeling Results This section of the report will present the findings from the chosen modeling techniques and discuss the overall performance of both approaches. Now that we have thoroughly explored the datset, it is time to work through developing a model that will allow us to predict our binary variable 'Stroke' where 1 represents the person suffering a stroke and 0 represents the person not suffering a stroke. One thing that needs to be noted is that we will report upon and discuss models that did not succeed in predicting stroke for the purposes of this academic exercise. In practice, the successful model would simply be discussed and presented but the steps undertaken during this project are important for full grades so all attempts to satistify this problem will be shown.

1. Logistic Regression At this point of the exercise, we are able to run our first model which is a generalized linear regression model. This one one of the most basic regression models that can be run in R. For this project, we are asked to go beyond this type of model (which will be done) but this is the initial starting point.

```
lin_model <- glm(formula = stroke ~ gender + age + hypertension + heart_disease
      + ever_married + work_type + Residence_type + avg_glucose_level + bmi +
        smoking_status, family = "binomial", data = strokedata_train)
summary(lin_model)
```

```
##
## Call:
## glm(formula = stroke ~ gender + age + hypertension + heart_disease +
##     ever_married + work_type + Residence_type + avg_glucose_level +
##     bmi + smoking_status, family = "binomial", data = strokedata_train)
##
## Deviance Residuals:
##    Min      1Q  Median      3Q     Max
## -1.118  -0.295  -0.154  -0.078   3.403
##
## Coefficients:
##                             Estimate Std. Error z value Pr(>|z|)
## (Intercept)                -6.94525    1.09715   -6.33  2.4e-10 ***
## genderMale                 -0.10797    0.18605   -0.58    0.562
## age                         0.07025    0.00753    9.33  < 2e-16 ***
## hypertensionYes             0.45323    0.21351    2.12    0.034 *
## heart_diseaseYes            0.39156    0.25190    1.55    0.120
## ever_marriedYes            -0.00131    0.30737    0.00    0.997
## work_typeGovt_job          -1.25433    1.16311   -1.08    0.281
```

```
## work_typeNever_worked      -10.23061  372.88628   -0.03    0.978
## work_typePrivate            -0.94133    1.14044   -0.83    0.409
## work_typeSelf-employed      -1.34868    1.16612   -1.16    0.247
## Residence_typeUrban         -0.02385    0.18078   -0.13    0.895
## avg_glucose_level            0.00504    0.00153    3.29    0.001 ***
## bmi                          0.01203    0.01442    0.83    0.404
## smoking_statusnever smoked  -0.20125    0.22131   -0.91    0.363
## smoking_statussmokes         0.22573    0.27075    0.83    0.404
## smoking_statusUnknown       -0.48958    0.30335   -1.61    0.107
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1199.57  on 3410  degrees of freedom
## Residual deviance:  950.89  on 3395  degrees of freedom
## AIC: 982.9
##
## Number of Fisher Scoring iterations: 14
```

From the results achieved in the summary() function, we see that we have a wide range of statistical significance on our model. In order to improve this, we will apply the stepwise method to our regression. Stepwise regression is a method of fitting regression models in which the choice of predictive variables is carried out by an automatic procedure. In each step, a variable is considered for addition to or subtraction from the set of explanatory variables based on some pre-specified criterion

```
lin_stepwise <- step(object = glm(stroke ~ ., family = "binomial", data =
    strokedata_train), scope = list(lower = glm(stroke ~ 1, family = "binomial",
    data = strokedata_train), upper = glm(stroke ~ ., family = "binomial",
    data = strokedata_train)), direction = "both", trace = F)
summary(lin_stepwise)
```

```
##
## Call:
## glm(formula = stroke ~ age + hypertension + heart_disease + avg_glucose_level,
##     family = "binomial", data = strokedata_train)
##
## Deviance Residuals:
##    Min      1Q  Median      3Q     Max
## -1.093  -0.295  -0.165  -0.083   3.548
##
## Coefficients:
##                   Estimate Std. Error z value Pr(>|z|)
## (Intercept)       -7.50159    0.44849  -16.73  < 2e-16 ***
## age                0.06412    0.00646    9.92  < 2e-16 ***
## hypertensionYes    0.49785    0.21063    2.36  0.01810 *
## heart_diseaseYes   0.41592    0.24783    1.68  0.09330 .
## avg_glucose_level  0.00540    0.00150    3.61  0.00031 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
```

```
##     Null deviance: 1199.57  on 3410  degrees of freedom
## Residual deviance:  961.95  on 3406  degrees of freedom
## AIC: 971.9
##
## Number of Fisher Scoring iterations: 7
```

Armed with the results from our stepwise regression, we can apply these modeling results to our test set to understand the predictive power of this model in determine whether or not a person in our dataset has suffered a stroke or not suffered a stroke. As is common in the data science space, we will first create the model set up and then evaluate it through a confusion matrix. In order to set the model up and evaluate it against the test dataset, the following steps must be taken:

```
strokedata$test$prediction <- predict(lin_stepwise, type = "response", newdata
    = strokedata_test)
strokedata_pred <- predict(lin_stepwise, type = "response", newdata =
    strokedata_test)
result_pred <- ifelse(strokedata_pred >= 0.5, "Yes", "No")
strokedata_test$prediction <- result_pred
glm_conf_mat <- print(confusionMatrix(as.factor(result_pred), reference =
    strokedata_test$stroke, positive = "Yes"))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   No  Yes
##        No  1399   63
##        Yes    0    0
##
##                Accuracy : 0.957
##                  95% CI : (0.945, 0.967)
##     No Information Rate : 0.957
##     P-Value [Acc > NIR] : 0.533
##
##                   Kappa : 0
##
##  Mcnemar's Test P-Value : 5.66e-15
##
##             Sensitivity : 0.0000
##             Specificity : 1.0000
##          Pos Pred Value :    NaN
##          Neg Pred Value : 0.9569
##              Prevalence : 0.0431
##          Detection Rate : 0.0000
##    Detection Prevalence : 0.0000
##       Balanced Accuracy : 0.5000
##
##        'Positive' Class : Yes
##
```

Through the confusion matrix of our first model, glm_conf_mat, we get to see some useful information that will help us through further developing a legitimate prediction model:

1. The overall accuracy of this model is 0.956, which on the surface looks excellent.

2. The sensitivity of the model is 0.00 and the specificity of the model is 1.00 which tells us that the model is excellent at predicting persons who do not suffer a stroke but a total miss for predicting persons in the dataset who have suffered a stroke. This model is basically ignoring the stroke victim class of our dataset because of the small proportion of the dataset that this group represents. Since so many persons in the dataset have not suffered a stroke, we have an issue with modeling/machine learning techniques basically predicting 'no stroke' for each person in the dataset which keeps overall accuracy very high but gives a 0% prediction ability for the key group we really want to be able to predict. In order to have a model that is effective through both target classes (stroke and no stroke), we need a model that can better handle the sample size imbalances in our predictor variable.

3. RMSE Modeling Root Mean Squared Error, which measures the model prediction error. It corresponds to the average difference between the observed known values of the outcome and the predicted value by the model. RMSE is computed as RMSE = mean((observed - predicted)^2) %>% sqrt() . The lower the RMSE, the better the model. Our first step is to convert the dataset to a new dataframe and to switch required variables over to dummy variables.

```
strokedata_dummy <- dummyVars("~stroke + gender + age + hypertension +
    heart_disease + ever_married + work_type + Residence_type +
    avg_glucose_level + bmi + smoking_status", data = strokedata)
strokedata_dummy <- data.frame(predict(strokedata_dummy, newdata = strokedata))
str(strokedata_dummy)
```

```
## 'data.frame':    4873 obs. of  24 variables:
##  $ stroke.No                     : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ stroke.Yes                    : num  1 1 1 1 1 1 1 1 1 1 ...
##  $ gender.Female                 : num  0 0 1 1 0 0 1 1 1 1 ...
##  $ gender.Male                   : num  1 1 0 0 1 1 0 0 0 0 ...
##  $ age                           : num  67 80 49 79 81 74 69 78 81 61 ...
##  $ hypertension.No               : num  1 1 1 0 1 0 1 1 0 1 ...
##  $ hypertension.Yes              : num  0 0 0 1 0 1 0 0 1 0 ...
##  $ heart_disease.No              : num  0 0 1 1 1 0 1 1 1 0 ...
##  $ heart_disease.Yes             : num  1 1 0 0 0 1 0 0 0 1 ...
##  $ ever_married.No               : num  0 0 0 0 0 0 1 0 0 0 ...
##  $ ever_married.Yes              : num  1 1 1 1 1 1 0 1 1 1 ...
##  $ work_type.children            : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ work_type.Govt_job            : num  0 0 0 0 0 0 0 0 0 1 ...
##  $ work_type.Never_worked        : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ work_type.Private             : num  1 1 1 0 1 1 1 1 1 0 ...
##  $ work_type.Self.employed       : num  0 0 0 1 0 0 0 0 0 0 ...
##  $ Residence_type.Rural          : num  0 1 0 1 0 1 0 0 1 1 ...
##  $ Residence_type.Urban          : num  1 0 1 0 1 0 1 1 0 0 ...
##  $ avg_glucose_level             : num  229 106 171 174 186 ...
##  $ bmi                           : num  36.6 32.5 34.4 24 29 27.4 22.8 24.2 29.7 36.8 ...
##  $ smoking_status.formerly.smoked: num  1 0 0 0 1 0 0 0 0 0 ...
##  $ smoking_status.never.smoked   : num  0 1 0 1 0 1 1 0 1 0 ...
##  $ smoking_status.smokes         : num  0 0 1 0 0 0 0 0 0 1 ...
##  $ smoking_status.Unknown        : num  0 0 0 0 0 0 0 1 0 0 ...
```

Now, our step is to remove double counted data categories (such as ever_married No and ever_married Yes counting the same variable which is ever_married)

```
strokedata_dummy$stroke.No <- NULL
strokedata_dummy$gender.Female <- NULL
```

```
strokedata_dummy$hypertension.No <- NULL
strokedata_dummy$heart_disease.No <- NULL
strokedata_dummy$ever_married.No <- NULL
strokedata_dummy$Residence_type.Rural <- NULL
```

After creating the dummy dataset, we can create new test and train datasets for the purposes of us breaking down the model.

```
index<- createDataPartition(strokedata_dummy$stroke, p = 0.7, list = FALSE)
train_d<-strokedata_dummy[index,]
test_d<-strokedata_dummy[-index,]
```

Calculate the mean of the test data and calculate the RMSE

```
mu <- print(mean(train_d$stroke))
```

```
## [1] 0.0422
```

```
model_a <- print(RMSE(test_d$stroke, mu))
```

```
## [1] 0.2047
```

Our RMSE is very low, which on the surface appears to be an excellent thing, this model is suffering from an imbalance between the proportions of our target variable who have suffered a stroke and those who have not. In order to properly run a prediction model, we need to account for this issue. A good way of thinking about this is that 4.3% of our persons have suffered a stroke and if we simply created a model which predicted 'No Stroke' for each person in the 4873 strokedata set, this model would be 95.73% accurate in overall terms but we would be 0% effective at predicting any of our 208 stroke persons.

3. ROSE Modeling

From the above modeling attempts, it is clear that we are going to get nowhere unless we can deal with the sampling disparity in our target variable. With only 4.3% of the persons having a stroke, the modeling and machine learning model/algorithms we have used to date are not properly assessing this particular group of persons which is making us unable to properly predict them. The overall accuracy and fit of the models are strong because they are being biases influenced by the disproportionate amount of non stroke persons. ROSE stands for Random Oversampling Examples and this package provides functions to deal with binary classification problems in the presence of imbalanced classes. Synthetic balanced samples are generated according to ROSE. A imbalance in a binary target variable is precisely what we have here in this dataset. To deal with the noted imbalances in our target variable, we will take a more involved attempt to establish a modeling technique which maintains a quality accuracy but accounts for the distribution difference. The library we will use for this is the ROSE library. We will also run this ROSE library in conjunction with another library, randomForest. RandomForest is based on generating a large number of decision trees, each constructed using a different subset of your training set. These subsets are usually selected by sampling at random and with replacement from the original data set. The decision trees are then used to identify a classification consensus by selecting the most common output (mode). Working together, these packages will allow us to artificially boost the number of 'stroke' cases in our target set so that the model is properly accounting for this subgroup of persons, while generating enough decision trees/outcomes that we can obtain reliable and concise results which we will then display in a confusion matrix.

```
strokedata_rose <- ROSE(stroke~., data = strokedata_train, N = 9000, seed =
    1000)$data
table(strokedata_rose$stroke)
```

```
##
##   No  Yes
## 4524 4476
```

```
rfrose <- randomForest(stroke~., data = strokedata_rose)
rose_conf_mat <- print(confusionMatrix(predict(rfrose, strokedata_test),
    strokedata_test$stroke, positive = "Yes"))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   No  Yes
##        No  1154   35
##        Yes  245   28
##
##                Accuracy : 0.808
##                  95% CI : (0.787, 0.828)
##     No Information Rate : 0.957
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.104
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.4444
##             Specificity : 0.8249
##          Pos Pred Value : 0.1026
##          Neg Pred Value : 0.9706
##              Prevalence : 0.0431
##          Detection Rate : 0.0192
##    Detection Prevalence : 0.1867
##       Balanced Accuracy : 0.6347
##
##        'Positive' Class : Yes
##
```

The results from this confusion matrix are much much more promising as we are now better able to predict stroke victims after better accounting for our sampling biases while still maintaining an overall accuracy of about 81%. This is our best model to date for prediction purposes. While the accuracy is lower than we have seen previously (which isn't a 'real' accuracy measure because of the issues predicting stroke cases), we do a much better job at predicting both cases rather than just the no stroke victims. This ROSE library was very important for our purposes of better evening out the imbalance of results for our stroke target variable between stroke victims and non stroke victims. These synthetically created data points forced our models to properly account for the stroke cases in the data rather than ignore them in the pursuit of a higher overall accuracy.

Conclusions & Project Extensions

From the exploratory data analysis portion of this project, we can make a number of important conclusions: - The risk of having a stroke is directly correlated to a person's age. This means that as a person ages, their likelihood of having a stroke increases. From the modeling portion of our analysis, we can make some important conclusions. - From our logistic regression work, we concluded that (for this dataset) that 'smoking status' did not have a statistically significant affect upon stroke prediction. For this project, the final model using the ROSE library allowed us to achieve the best results since it best allowed us to account for how we had only a small percentage of persons in our dataset (4.3%) who actually suffered a stroke. It did not take long before it was realized that very limited useful intelligence can be taken from this work without accounting for the count disparity between persons in our dataset with a stroke and without a stroke (heavily favouring those without a stroke). By implementing the ROSE library, this allowed us to create synthetic variables to even out our imbalances and thus gain a model that can both predict no stroke person and stroke persons, whereas before it was overfitted to the no stroke persons. Fortunately, we were able to manipulate the data and bring in some more sophisticated modeling packages that allowed us to generate a model that effectively predicts both persons with stroke and without stroke. In terms of project extensions there are really only two things that come to mind with regards to this dataset: 1. Perhaps during the data collection portion (creating the dataset as a whole) we could try to both sample more persons (increase sample size) and to seek out more people who have suffered a stroke to take part in order to give more people to the stroke set and even out our imbalances in the tatget variable. 2. We did find and implement the ROSE package in order to help account for some of our imbalance issues, but further packages and approaches could be explored to find a algorithm that account for target variable imbalances even better than the ROSE package.