ST310 Final R Markdown

16923, 17915, 22220

4/6/2021

ST310 Machine Learning Project: Understanding and Predicting Stroke Occurrences in Imbalanced Data

Load the dataset

```r
data <- read.csv("healthcare-dataset-stroke-data.csv", header=TRUE)
data <- subset(data, select= -c(1)) #remove the id from the dataframe
```

Load the required packages

```r
library(tidyverse); library(ggplot2); library(GGally); library(corrplot); library(SmartEDA)
library(dplyr); library(DataExplorer); library(tibble); library(naniar); library(gridExtra);
library(caret); library(caTools); library(xgboost); library(broom); library(kernlab);
library(MASS); library(modelr); library(glmnet); library(selectiveInference); library(imbalance)
```

EDA———————————————————————————————

Inspect the data

```r
dim(data) # 5110 rows and 11 columns
```

```
## [1] 5110   11
```

```r
str(data)
```

```
## 'data.frame':    5110 obs. of  11 variables:
##  $ gender           : Factor w/ 3 levels "Female","Male",..: 2 1 2 1 1 2 2 1 1 1 ...
##  $ age              : num  67 61 80 49 79 81 74 69 59 78 ...
##  $ hypertension     : int  0 0 0 0 1 0 1 0 0 0 ...
##  $ heart_disease    : int  1 0 1 0 0 0 1 0 0 0 ...
##  $ ever_married     : Factor w/ 2 levels "No","Yes": 2 2 2 2 2 2 2 1 2 2 ...
##  $ work_type        : Factor w/ 5 levels "children","Govt_job",..: 4 5 4 4 5 4 4 4 4 4 ...
##  $ Residence_type   : Factor w/ 2 levels "Rural","Urban": 2 1 1 2 1 2 1 2 1 2 ...
##  $ avg_glucose_level: num  229 202 106 171 174 ...
##  $ bmi              : Factor w/ 419 levels "10.3","11.3",..: 240 419 199 218 114 164 148 102 419 116
##  $ smoking_status   : Factor w/ 4 levels "formerly smoked",..: 1 2 2 3 2 1 2 2 4 4 ...
##  $ stroke           : int  1 1 1 1 1 1 1 1 1 1 ...
```

```
summary(data)
```

```
##     gender          age         hypertension     heart_disease     ever_married
##  Female:2994   Min.   : 0.08   Min.   :0.00000   Min.   :0.00000   No :1757
##  Male  :2115   1st Qu.:25.00   1st Qu.:0.00000   1st Qu.:0.00000   Yes:3353
##  Other :   1   Median :45.00   Median :0.00000   Median :0.00000
##                Mean   :43.23   Mean   :0.09746   Mean   :0.05401
##                3rd Qu.:61.00   3rd Qu.:0.00000   3rd Qu.:0.00000
##                Max.   :82.00   Max.   :1.00000   Max.   :1.00000
##
##          work_type    Residence_type avg_glucose_level      bmi
##  children     : 687   Rural:2514     Min.   : 55.12    N/A    : 201
##  Govt_job     : 657   Urban:2596     1st Qu.: 77.25    28.7   :  41
##  Never_worked :  22                  Median : 91.89    28.4   :  38
##  Private      :2925                  Mean   :106.15    26.1   :  37
##  Self-employed: 819                  3rd Qu.:114.09    26.7   :  37
##                                      Max.   :271.74    27.6   :  37
##                                                        (Other):4719
##          smoking_status     stroke
##  formerly smoked: 885   Min.   :0.00000
##  never smoked   :1892   1st Qu.:0.00000
##  smokes         : 789   Median :0.00000
##  Unknown        :1544   Mean   :0.04873
##                         3rd Qu.:0.00000
##                         Max.   :1.00000
##
```

- Several categorical predictors are wrongly coded as numerical and vice versa.
- Gender has a single datapoint that falls into the level titled 'Other'
- bmi has 201 N/A values
- smoking_status has a category titled 'Unknown'
- Minimum age is 0.08

```
sum(data$age<1) ### 43 people are less than a year old
```

```
## [1] 43
```

Data preparation

```
# Remove the datapoint that falls under the level titled 'Other' for gender
data[data$gender=='Other',] # Iddentify the row corresponding to this datapoint
```

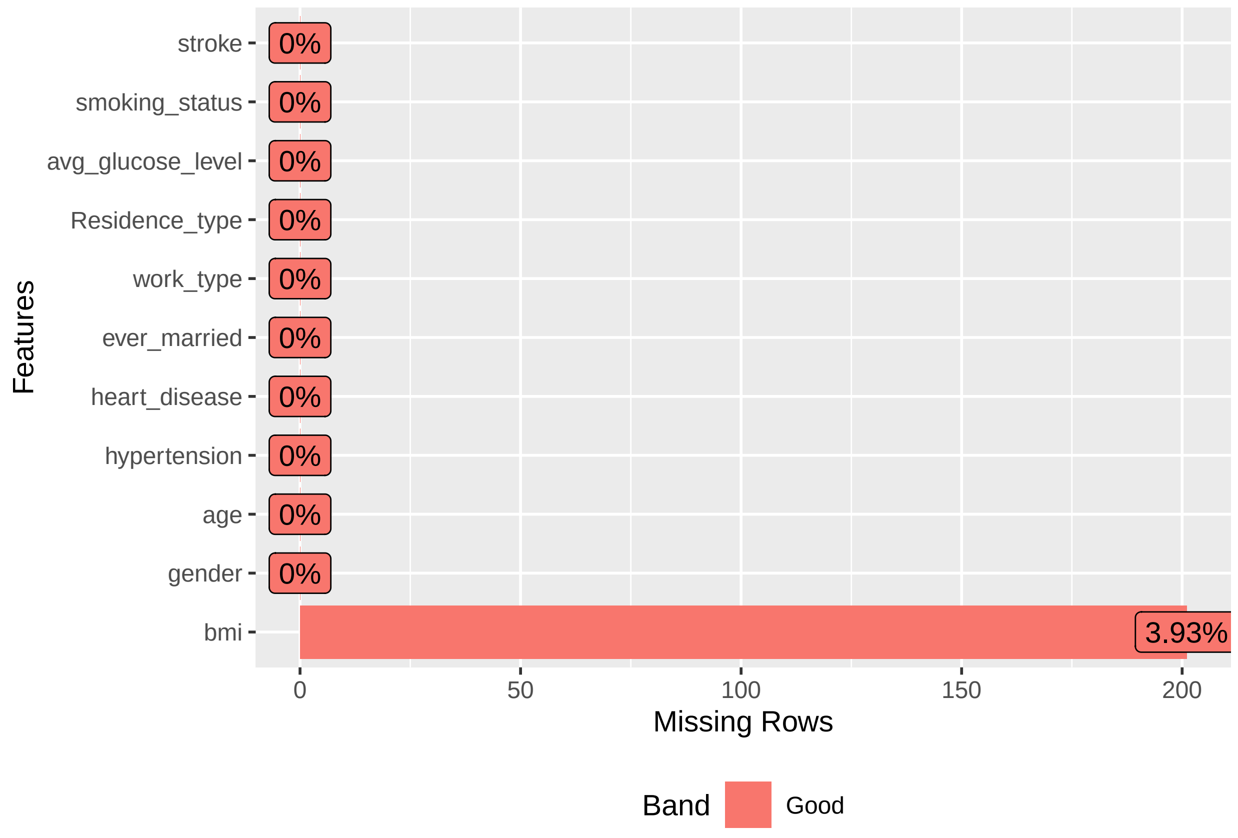
```
##      gender age hypertension heart_disease ever_married work_type
## 3117  Other  26            0             0           No   Private
##      Residence_type avg_glucose_level  bmi  smoking_status stroke
## 3117          Rural            143.33 22.4 formerly smoked      0
```

```r
data <- data[-3117,] # Remove the datapoint
data$gender <- fct_drop(data$gender) # Remove the level 'Other'

# Convert hypertension, heart_disease and stroke to categorical
data$hypertension <- as.factor(data$hypertension)
data$heart_disease <- as.factor(data$heart_disease)
data$stroke <- as.factor(data$stroke)

# Convert bmi too numeric
data$bmi <- as.character(data$bmi)
data$bmi <- as.numeric(data$bmi)

summary(data) # Changes have been made
```

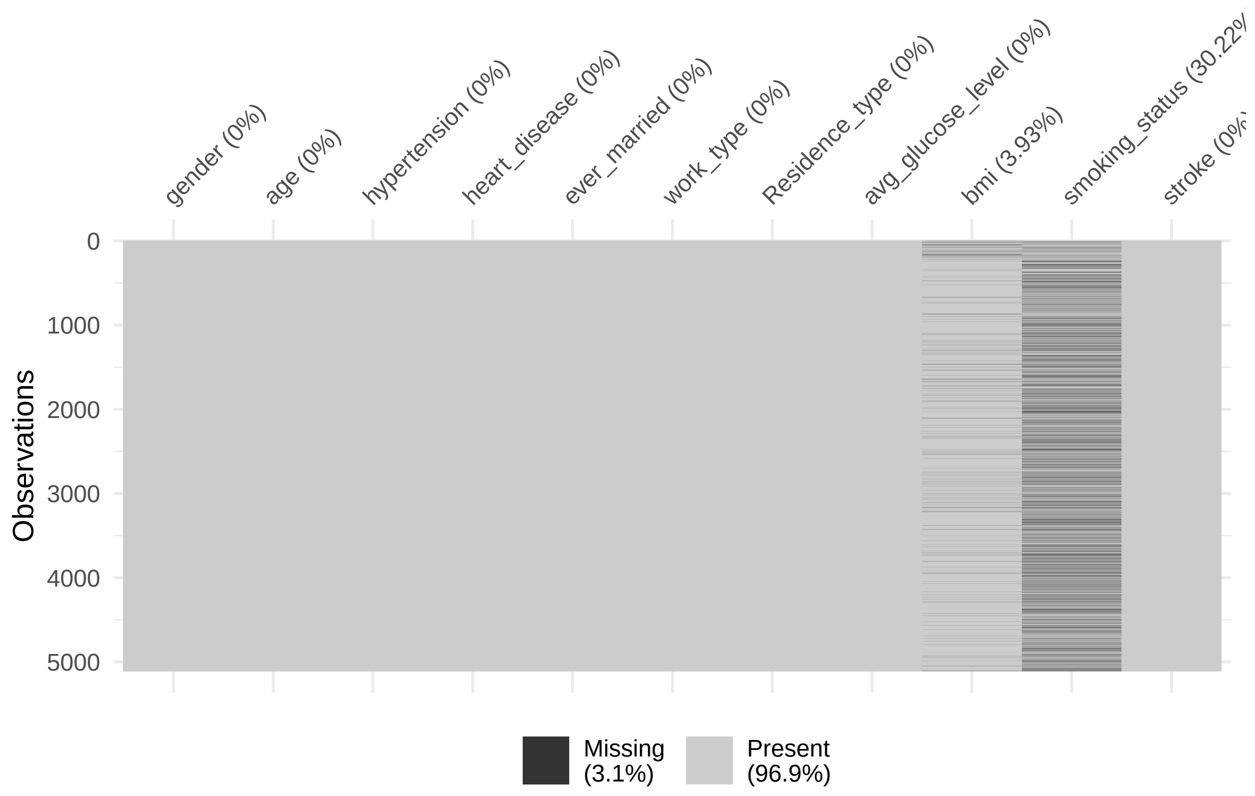
```
##     gender          age         hypertension heart_disease ever_married
##  Female:2994   Min.   : 0.08   0:4611       0:4833        No :1756
##  Male  :2115   1st Qu.:25.00   1: 498       1: 276        Yes:3353
##                Median :45.00
##                Mean   :43.23
##                3rd Qu.:61.00
##                Max.   :82.00
##
##          work_type    Residence_type avg_glucose_level      bmi
##  children     : 687   Rural:2513     Min.   : 55.12    Min.   :10.30
##  Govt_job     : 657   Urban:2596     1st Qu.: 77.24    1st Qu.:23.50
##  Never_worked :  22                  Median : 91.88    Median :28.10
##  Private      :2924                  Mean   :106.14    Mean   :28.89
##  Self-employed: 819                  3rd Qu.:114.09    3rd Qu.:33.10
##                                      Max.   :271.74    Max.   :97.60
##                                                        NA's   :201
##          smoking_status stroke
##  formerly smoked: 884   0:4860
##  never smoked   :1892   1: 249
##  smokes         : 789
##  Unknown        :1544
##
##
##
```

Iddentifying N/A values

```r
# Plot the amount of missing values for each of the parameters in the dataset
plot_missing(data) # 4% of BMI is missing
```



```
# Replace the "N/A" in bmi & "Unknown" in Smoking status using the naniar package
data_clean <- replace_with_na(data = data, replace = list(bmi = c("N/A"), smoking_status = c("Unknown")

vis_miss(data_clean) #plot where missing values for each of the parameter in the dataset are + percent
```



```
#30% of the data is missing for smoking and 4% is missing for bmi

sapply(data_clean, function(x) sum(is.na(x)))
```

```
##            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              1544                 0
```

```
#201 missing values for BMI and 1544 missing values for smoking_status

summary(data_clean) # The changes have been made
```

```
##     gender          age        hypertension heart_disease ever_married
##  Female:2994   Min.   : 0.08   0:4611       0:4833        No :1756    
##  Male  :2115   1st Qu.:25.00   1: 498       1: 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.   : 55.12    Min.   :10.30  
##  Govt_job     : 657   Urban:2596     1st Qu.: 77.24    1st Qu.:23.50
##  Never_worked :  22                  Median : 91.88    Median :28.10
##  Private      :2924                  Mean   :106.14    Mean   :28.89
##  Self-employed: 819                  3rd Qu.:114.09    3rd Qu.:33.10
##                                      Max.   :271.74    Max.   :97.60
##                                                        NA's   :201
##         smoking_status stroke
##  formerly smoked: 884   0:4860
##  never smoked   :1892   1: 249
##  smokes         : 789
##  Unknown        :   0
##  NA's           :1544
##
##
```

We have 2 possible approaches that we can take when dealing with the N/A values: 1. Remove the rows containing N/A values
2. Impute the bmi/smoking_status values with the most common value (mode)

```
data_remove <- data_clean[complete.cases(data_clean), ]
data_remove$smoking_status <- fct_drop(data_remove$smoking_status) # Remove 'Unknown' as a level of the

dim(data_remove) #3425 observations
```

Method 1 - Remove the rows containing N/A values

```
## [1] 3425   11
```

```
summary(data_remove)
```

```
##    gender          age        hypertension heart_disease ever_married
##  Female:2086   Min.   :10.00   0:3017       0:3219        No : 826
##  Male  :1339   1st Qu.:34.00   1: 408       1: 206        Yes:2599
##                Median :50.00
##                Mean   :48.65
##                3rd Qu.:63.00
##                Max.   :82.00
##         work_type    Residence_type avg_glucose_level      bmi
##  children     :  68   Rural:1680     Min.   : 55.12    Min.   :11.50
##  Govt_job     : 514   Urban:1745     1st Qu.: 77.23    1st Qu.:25.30
##  Never_worked :  14                  Median : 92.35    Median :29.10
##  Private      :2200                  Mean   :108.31    Mean   :30.29
##  Self-employed: 629                  3rd Qu.:116.20    3rd Qu.:34.10
##                                      Max.   :271.74    Max.   :92.00
##         smoking_status stroke
##  formerly smoked: 836   0:3245
##  never smoked   :1852   1: 180
##  smokes         : 737
##
##
##
```

```
data_impute <- impute_median_at(data_clean, .vars=c("bmi")) #impute bmi at median value using the nania
data_impute <- fill(data_clean, smoking_status) #fills missing values using the previous entry, assumpt
dim(data_impute) #5110
```

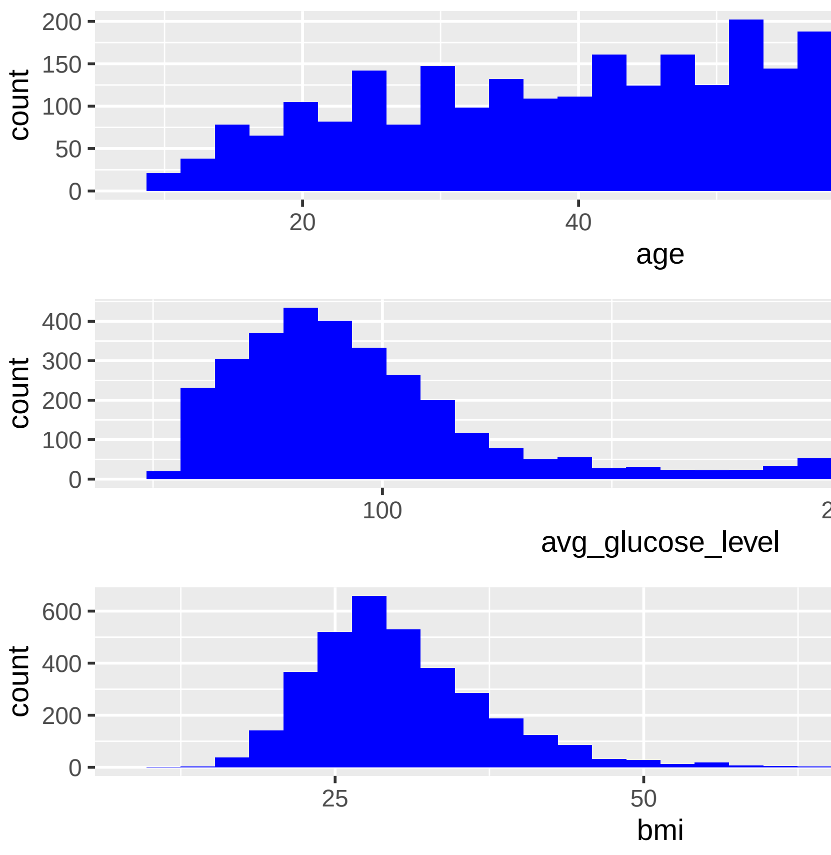
Method 2 - Impue values for the N/A values

```
## [1] 5109   11
```

We decided to remove the rows that contained N/A values (method 1) for a number of reasons explained in the report

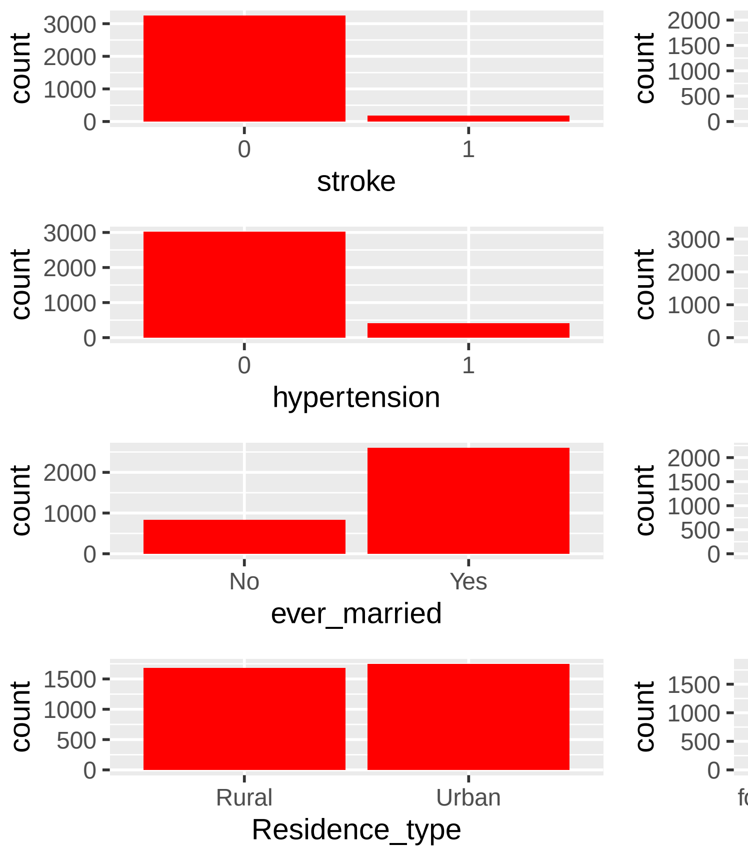
Conducting a univariate analysis of the variables

```
age <- ggplot(data_remove, aes(x=age)) + geom_histogram(fill = "blue")
#approx normally distributed
avg_glucose_level <- ggplot(data_remove, aes(x=avg_glucose_level)) + geom_histogram(fill = "blue")
#bi-modal
bmi <- ggplot(data_remove, aes(x=bmi)) + geom_histogram(fill = "blue")
#approx positively skewed

grid.arrange( age, avg_glucose_level, bmi, ncol=1)
```



Plot histograms for the continuous variables

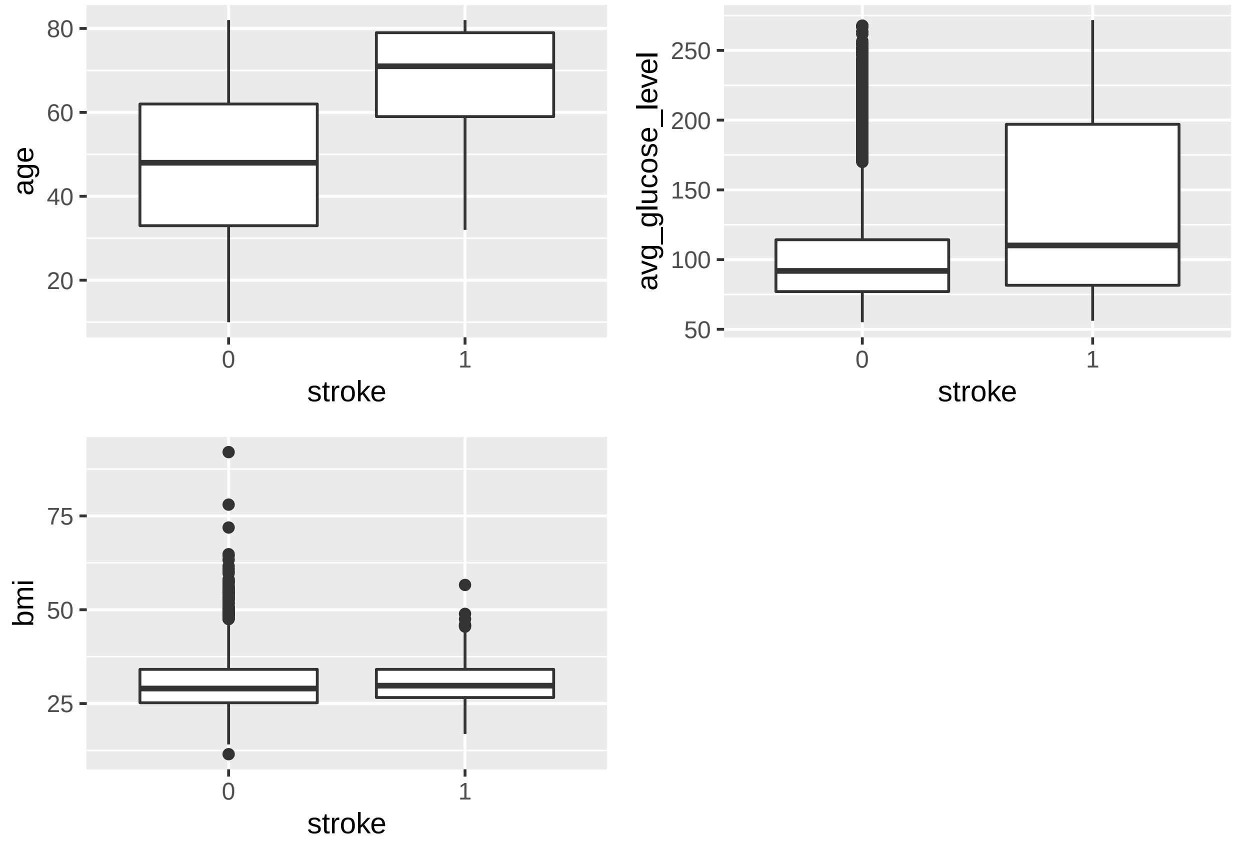
```
stroke <- ggplot(data_remove, aes(x=stroke)) + geom_bar(stat='count', fill = "red")
gender <- ggplot(data_remove, aes(x=gender)) + geom_bar(stat='count', fill = "red")
hypertension <- ggplot(data_remove, aes(x=hypertension)) + geom_bar(stat='count', fill = "red")
heart_disease <- ggplot(data_remove, aes(x=heart_disease)) + geom_bar(stat='count', fill = "red")
ever_married <- ggplot(data_remove, aes(x=ever_married)) + geom_bar(stat='count', fill = "red")
work_type <- ggplot(data_remove, aes(x=work_type)) + geom_bar(stat='count', fill = "red")
Residence_type <- ggplot(data_remove, aes(x=Residence_type)) + geom_bar(stat='count', fill = "red")
smoking_status <- ggplot(data_remove, aes(x=smoking_status)) + geom_bar(stat='count', fill = "red")

grid.arrange(stroke, gender, hypertension, heart_disease, ever_married, work_type, Residence_type, smok
```



Plot bar charts to observe the categorical variables

```
age_boxplot <- ggplot(data = data_remove, aes(stroke, age)) +
  geom_boxplot()
avg_glucose_level_boxplot <- ggplot(data = data_remove, aes(stroke, avg_glucose_level)) +
  geom_boxplot()
bmi_boxplot <- ggplot(data = data_remove, aes(stroke, bmi)) +
  geom_boxplot()
grid.arrange( age_boxplot, avg_glucose_level_boxplot, bmi_boxplot, nrow=2)
```

Plot boxplots to determine the relationship between stroke and the continuous predictors

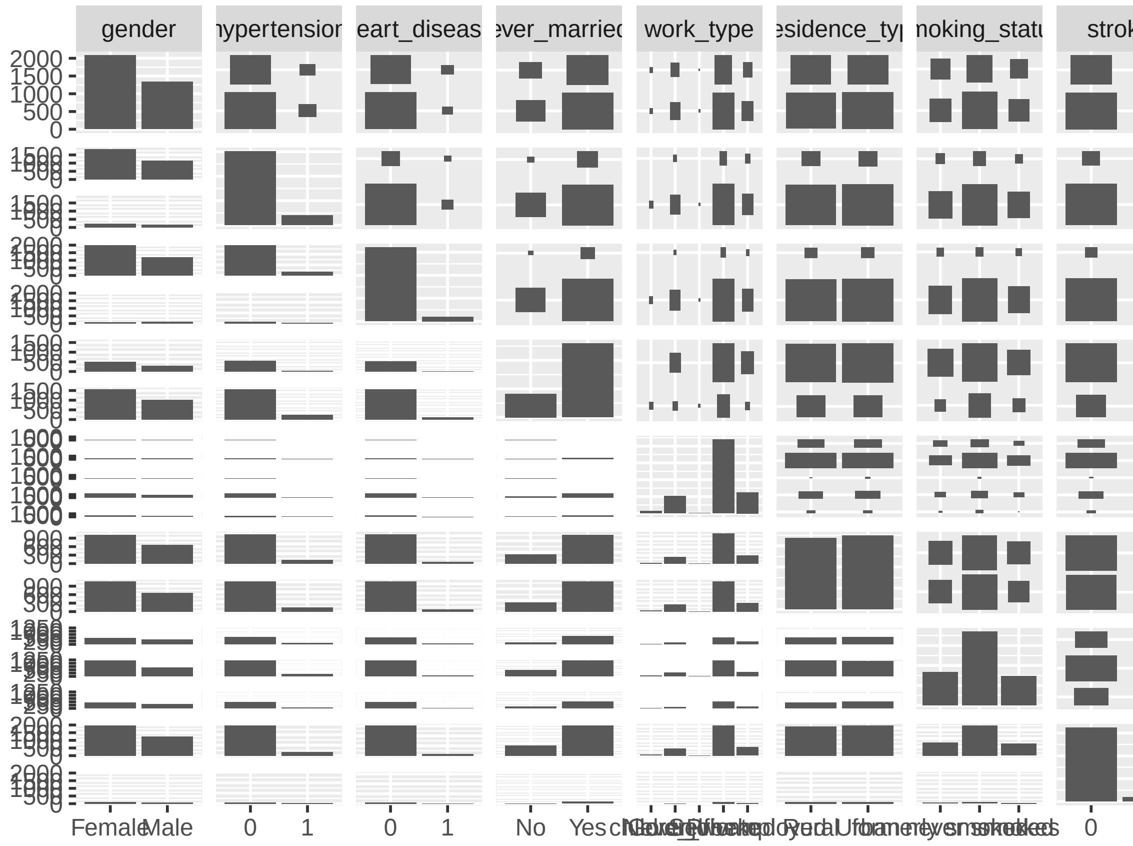


Observe the correlation among variables

Create subsets of the continuous & discrete variables

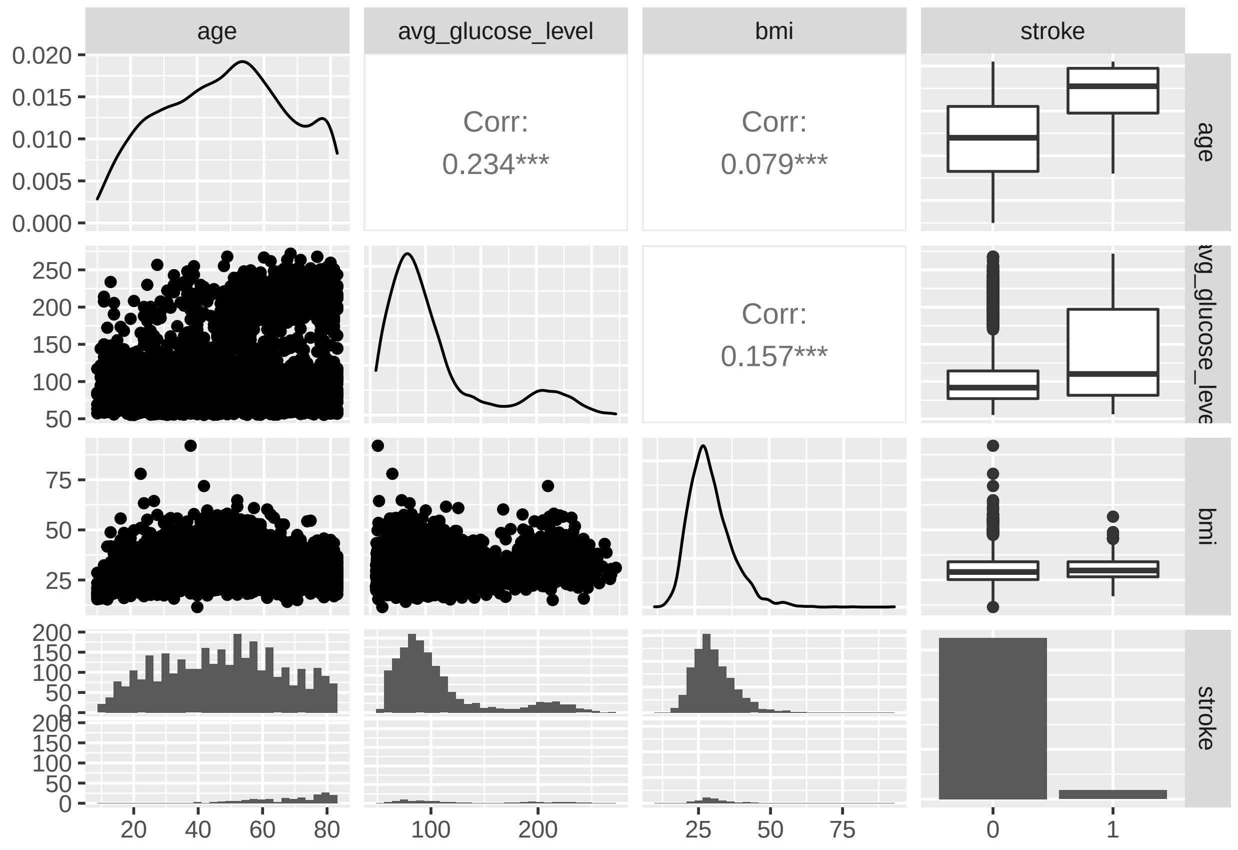
```
contVars <- subset(data_remove, select=c(age, avg_glucose_level,bmi, stroke))
discVars <- subset(data_remove, select=c(gender, hypertension, heart_disease, ever_married, work_type, 
```

```


disccorr <- ggpairs(discVars)
disccorr
```



Pairwise correlations

```
contcorr <- ggpairs(contVars)
contcorr
```



Observe the distribution of people who had a stroke———————————-

```
table(data_remove$stroke) #180 total people with a stroke, 3246 with no stroke
```

```
##
##    0    1
## 3245  180
```

```
3245/(180+3245) # Percentage of people who had no stroke
```

```
## [1] 0.9474453
```

- There is a massive class imbalance
- This can be rectified using several methods such as undersampling, oversampling etc.

Dummy model: If we build a model to continuously predict that an individual does not have a stroke, the misclassification rate would be 5% (1-0.9474606)

There are several ways of dealing with the imbalance in the dataset. We will focus on 2 of them 1. Undersampling - Downsampling the larger class 2. Oversampling - Oversampling the minority class

We will develop our models under both methods

1. Using undersampling

Create a balanced dataset with the same number of observations in both classes (in Stroke) using undersampling

```r
stroke_No <- data_remove %>%
  filter(stroke == 0) %>%
  sample_n(size = 180)

stroke_Yes <- data_remove %>%
  filter(stroke == 1)

data_under <- rbind(stroke_No, stroke_Yes)
summary(data_under)
```

```
##     gender         age        hypertension heart_disease ever_married
##  Female:216   Min.   :12.00   0:286        0:317         No : 64     
##  Male  :144   1st Qu.:43.00   1: 74        1: 43         Yes:296     
##               Median :59.00                                          
##               Mean   :56.78                                          
##               3rd Qu.:73.25                                          
##               Max.   :82.00                                          
##          work_type   Residence_type avg_glucose_level      bmi       
##  children     :  4   Rural:172      Min.   : 55.46    Min.   :16.90  
##  Govt_job     : 43   Urban:188      1st Qu.: 79.08    1st Qu.:25.50  
##  Never_worked :  1                  Median : 97.66    Median :29.40  
##  Private      :226                  Mean   :120.81    Mean   :30.23  
##  Self-employed: 86                  3rd Qu.:168.29    3rd Qu.:34.10  
##                                     Max.   :271.74    Max.   :56.60  
##          smoking_status stroke 
##  formerly smoked:105    0:180  
##  never smoked   :189    1:180  
##  smokes         : 66           
##                                
##                                
## 
```

Create the training and testing data for the dataset obtained using undersampling

```r
set.seed(4)

data_under$id <- 1:nrow(data_under) # Create an id
training_under <- data_under %>% sample_frac(.7)
testing_under  <- anti_join(data_under, training_under, by = 'id')
training_under <- training_under %>% dplyr::select(-id)
testing_under <- testing_under %>% dplyr::select(-id)
beta_testing_under <- testing_under %>% dplyr::select(-stroke)

#summary(training_under)
#dim(training_under)
#summary(testing_under)
#dim(testing_under)
```

2. Using oversampling

Create a balanced dataset with the same number of observations in both classes using oversampling

```
data_over <- data_remove

# Compute the imbalance ratio of stroke
imbalanceRatio(as.data.frame(data_over), classAttr = "stroke")
```

```
## [1] 0.05546995
```

```
# Name the levels of stroke
data_over$stroke <- as.factor(ifelse(data_over$stroke == 0, "no", "yes"))

# Put variables as correct format
data_over$gender <- as.factor(data_over$gender)
data_over$hypertension <- as.factor(data_over$hypertension)
data_over$heart_disease <- as.factor(data_over$heart_disease)
data_over$ever_married <- as.factor(data_over$ever_married)
data_over$work_type <- as.factor(data_over$work_type)
data_over$Residence_type <- as.factor(data_over$Residence_type)
data_over$smoking_status <- as.factor(data_over$smoking_status)
data_over <- as.data.frame(lapply(data_over, as.numeric))


data_over <- oversample(as.data.frame(data_over), classAttr = "stroke", ratio = 1, method = "MWMOTE")

data_over$stroke <- as.factor(data_over$stroke)

table(data_over$stroke)
```

```
##
##    1    2
## 3245 3245
```

```
#Bot as is no shown, there are now equally as many stroke cases as non-stroke cases
summary(data_over)
```

```
##      gender           age         hypertension   heart_disease
##  Min.   :1.000   Min.   :10.00   Min.   :1.000   Min.   :1.000
##  1st Qu.:1.000   1st Qu.:47.00   1st Qu.:1.000   1st Qu.:1.000
##  Median :1.072   Median :63.24   Median :1.000   Median :1.000
##  Mean   :1.408   Mean   :58.96   Mean   :1.208   Mean   :1.144
##  3rd Qu.:2.000   3rd Qu.:75.00   3rd Qu.:1.292   3rd Qu.:1.000
##  Max.   :2.000   Max.   :82.00   Max.   :2.000   Max.   :2.000
##   ever_married     work_type     Residence_type  avg_glucose_level
##  Min.   :1.000   Min.   :1.000   Min.   :1.000   Min.   : 55.12
##  1st Qu.:2.000   1st Qu.:4.000   1st Qu.:1.000   1st Qu.: 80.88
##  Median :2.000   Median :4.000   Median :1.547   Median : 99.72
##  Mean   :1.827   Mean   :3.928   Mean   :1.512   Mean   :124.18
##  3rd Qu.:2.000   3rd Qu.:4.340   3rd Qu.:2.000   3rd Qu.:180.95
##  Max.   :2.000   Max.   :5.000   Max.   :2.000   Max.   :271.74
##       bmi         smoking_status  stroke
##  Min.   :11.50   Min.   :1.000   1:3245
##  1st Qu.:26.13   1st Qu.:1.478   2:3245
##  Median :29.30   Median :2.000
##  Mean   :30.21   Mean   :1.944
##  3rd Qu.:33.23   3rd Qu.:2.032
##  Max.   :92.00   Max.   :3.000
```

Create the training and testing data for the dataset obtained using oversampling

```
data_over$id <- 1:nrow(data_over)
training_over <- data_over %>% sample_frac(.7)
testing_over  <- anti_join(data_over, training_over, by = 'id')
training_over <- training_over %>% dplyr::select(-id)
testing_over <- testing_over %>% dplyr::select(-id)
beta_testing_over <- testing_over %>% dplyr::select(-stroke)

dim(training_over) #4543 observations
```

```
## [1] 4543   11
```

```
dim(testing_over) #1947 observations
```

```
## [1] 1947   11
```

Models developed using undersampling——————————

Model 1: Logistic regression

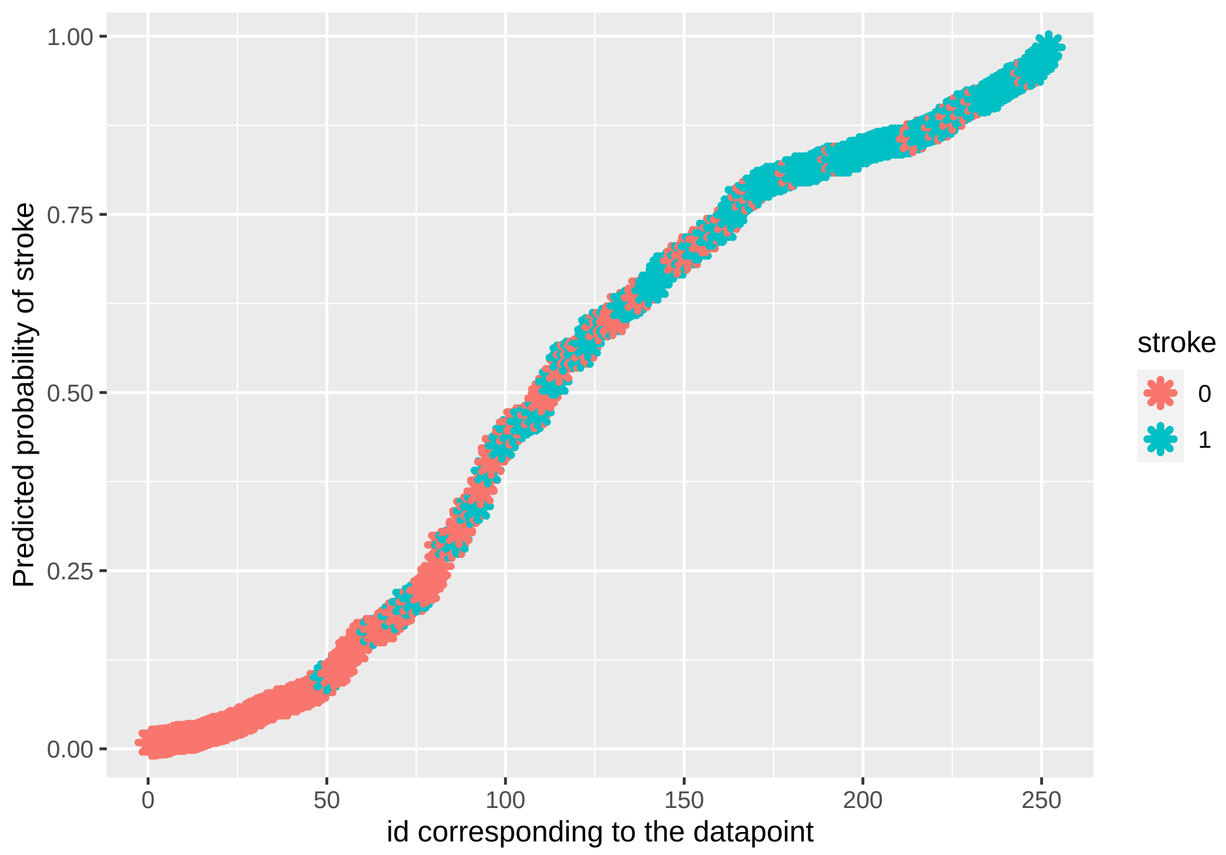
stepAIC in the MASS package package was used to obtain the model that contains the most contributive predictors by minimising AIC

```
# Build a glm with all the predictors
glm_all_under <- glm(stroke~., family=binomial(link = "logit"), data=training_under)

# Build a glm with only the most contributive predictors
glm_under <- glm_all_under %>% stepAIC(trace = FALSE)
# trace = FALSE allows the function to provide only the model with the lowest AIC and none of the inter
summary(glm_under)
```

```
##
## Call:
## glm(formula = stroke ~ age + hypertension + smoking_status, family = binomial(link = "logit"),
##     data = training_under)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.4330  -0.6003   0.2654   0.6639   2.1441
##
## Coefficients:
```

```
##                             Estimate Std. Error z value Pr(>|z|)
## (Intercept)                 -5.70361    0.81788  -6.974 3.09e-12 ***
## age                          0.09392    0.01224   7.674 1.67e-14 ***
## hypertension1                0.85917    0.42102   2.041  0.04129 *
## smoking_statusnever smoked -0.15206    0.37382  -0.407  0.68418
## smoking_statussmokes         1.37822    0.51234   2.690  0.00714 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 349.20  on 251  degrees of freedom
## Residual deviance: 221.71  on 247  degrees of freedom
## AIC: 231.71
##
## Number of Fisher Scoring iterations: 5
```

```
#Create a visual representation of the logistic model
predicted.data <- data.frame(
  probability.of.stroke = glm_under$fitted.values,
  stroke=training_under$stroke)

predicted.data <- predicted.data[
  order(predicted.data$probability.of.stroke, decreasing=FALSE),]
predicted.data$rank <- 1:nrow(predicted.data)

ggplot(data=predicted.data, aes(x=rank, y=probability.of.stroke)) +
  geom_point(aes(color=stroke), alpha=1, shape=8, size = 2, stroke = 2) +
  xlab("id corresponding to the datapoint") +
  ylab("Predicted probability of stroke")
```



```
# Predict the testing data using the glm
pred_glm_under = predict(glm_under, testing_under, type="response")
pred_glm_under <- ifelse((pred_glm_under>=0.3), 1, 0) # This threshold value was one of the values that

# Create a confusion matrix to assess the performance of the model
conf_matrix_glm_under <- table(pred_glm_under, testing_under$stroke, deparse.level = 2)
conf_matrix_glm_under
```

```
##                 testing_under$stroke
## pred_glm_under  0  1
##               0 29  4
##               1 28 47
```

Comparing glm__all__under and glm__under

```
rbind(glance(glm_all_under), glance(glm_under))
```

```
## # A tibble: 2 x 8
##   null.deviance df.null logLik   AIC   BIC deviance df.residual  nobs
##           <dbl>   <int>  <dbl> <dbl> <dbl>    <dbl>       <int> <int>
## 1          349.     251  -109.  247.  300.     217.         237   252
## 2          349.     251  -111.  232.  249.     222.         247   252
```

Model 2: Linear model to predict stroke implemented using gradient descent

Preprocess the data

```
x_var_train_for_lm_under  <- training_under %>%
  dplyr::select(age, bmi)

# Create a design matrix for the training data
X_train_for_lm_under <- model.matrix(~. -1, data = x_var_train_for_lm_under) # -1 removes the intercept

# Convert stroke to numeric
train_under_2 <- training_under
train_under_2$stroke <- as.numeric(train_under_2$stroke)
Y_train_for_lm_under <- train_under_2 %>% pull(stroke)

x_var_test_for_lm_under  <- testing_under %>%
  dplyr::select(age, bmi)

# Create a design matrix for the testing data
X_test_for_lm_under <- model.matrix(~. -1, data = x_var_test_for_lm_under)

# Convert stroke to numeric
test_under_2 <- testing_under
test_under_2$stroke <- as.numeric(test_under_2$stroke)

Y_test_for_lm_under <- test_under_2 %>% pull(stroke)

dim(X_train_for_lm_under) #252
```

```
## [1] 252   2
```

```
dim(X_test_for_lm_under) #108
```

```
## [1] 108   2
```

```
# Define the least squares function
least_squares_gradient <- function(x, y, beta) {
  -2 * t(x)  %*% (y - x  %*% beta)
}
# t(x) is the transpose of x

# Define the loss function
least_squares_loss <- function(x, y, beta) {
  sum((y - x %*% beta)^2)
}

# Initialize coefficients
gamma = 0.000001 # this is the step size
p = 2 # This is the number of predictors

beta0 <- rep(0, p) # This is the vector of all 0s
previous_loss <- least_squares_loss(X_train_for_lm_under, Y_train_for_lm_under, beta0) # Loss function
```

```r
grad0 <- least_squares_gradient(X_train_for_lm_under, Y_train_for_lm_under, beta0) # Initialise the gra
beta1 <- beta0 - gamma * grad0
next_loss <- least_squares_loss(X_train_for_lm_under, Y_train_for_lm_under, beta1)
previous_beta <- beta1
steps <- 1

while (abs(previous_loss - next_loss) > 0.00001) {
  gradn <- least_squares_gradient(X_train_for_lm_under, Y_train_for_lm_under, previous_beta)
  # Refine update by allowing step size to change at each iteration. Make step size a sequence of numbe
  next_beta <- previous_beta - (0.99)^steps * gradn / sqrt(sum(gradn^2))
  # We rescale the gradient i.e. use sqrt(sum(gradn^2)) to prevent the algorithm from diverging
  steps <- steps + 1
  previous_beta <- next_beta
  previous_loss <- next_loss
  next_loss <- least_squares_loss(X_train_for_lm_under, Y_train_for_lm_under, next_beta)
}

# Predict the expected values for the testing data
pred_lm_grad_desc_under = X_test_for_lm_under %*% previous_beta
pred_lm_grad_desc_under = round(as.integer(pred_lm_grad_desc_under))
pred_lm_grad_desc_under = as.factor(pred_lm_grad_desc_under)

# Create a confusion matrix to assess the performance of the model on the testing data
table(pred_lm_grad_desc_under, testing_under$stroke, deparse.level = 2)
```

```
##                          testing_under$stroke
## pred_lm_grad_desc_under  0  1
##                        0  9  0
##                        1 48 48
##                        2  0  3
```

```r
# Variable that is rounded to 2, is actually a 1, as 0 and 1 are the only outcomes
```

Model 3: Lasso (least absolute shrinkage and selection operator) regression

Preprocess this data to use for lasso

```r
x_var_train_for_lasso_under  <- training_under %>%
  dplyr::select(-stroke)

# Create a design matrix for the training data
X_train_for_lasso_under <- model.matrix(~. -1, data = x_var_train_for_lasso_under) # -1 removes the int

Y_train_for_lasso_under <- training_under %>% pull(stroke)

x_var_test_for_lasso_under  <- testing_under %>%
  dplyr::select(-stroke)

# Create a design matrix for the testing data
X_test_for_lasso_under <- model.matrix(~. -1, data = x_var_test_for_lasso_under) #

Y_test_for_lasso_under <- testing_under %>% pull(stroke)
```

```
dim(X_train_for_lasso_under) #252
```

```
## [1] 252  15
```

```
dim(X_test_for_lasso_under) #108
```

```
## [1] 108  15
```

The model was created using lambda.min as the best lambda

```
#Perform 10 fold cross validation using the misclassification rate to find the best lambda
lasso_cv_under = cv.glmnet(X_train_for_lasso_under,
                  Y_train_for_lasso_under,
                  family = "binomial",
                  type.measure = "class") # type.measure = "class" allows us to use the misclassificati
#plot(lasso_cv_under)

# Fit the lasso model on the training data
lasso_under <- glmnet(X_train_for_lasso_under,
                         Y_train_for_lasso_under,
                         alpha = 1,
                         family = "binomial",
                         lambda = lasso_cv_under$lambda.min)

# Predict the testing data using the glm
pred_lasso_under <- lasso_under %>% predict(newx = X_test_for_lasso_under)
pred_lasso_under <- ifelse(pred_lasso_under >= 0.3, 1, 0)

# Create a confusion matrix to assess the performance of the model
table(pred_lasso_under, Y_test_for_lasso_under, deparse.level = 2)
```

```
##                  Y_test_for_lasso_under
## pred_lasso_under  0  1
##                 0 42 14
##                 1 15 37
```

Model 4: kernel method

```
# Build the model using training data
kvsm_under <- ksvm(stroke ~ age + hypertension + heart_disease + avg_glucose_level + smoking_status , k
kvsm_under
```

```
## Support Vector Machine object of class "ksvm"
##
## SV type: C-svc  (classification)
##  parameter : cost C = 1
##
## Gaussian Radial Basis kernel function.
##  Hyperparameter : sigma =  0.483051818661611
```

```
##
## Number of Support Vectors : 146
##
## Objective Function Value : -111.6252
## Training error : 0.178571
```

```
# Predict the testing data using the model built
pred_kvsm_under <- predict(kvsm_under, type = 'response', newdata = testing_under)

# Create a confusion matrix to assess the performance of the model
conf_matrix_kvsm_under<-table(Predicted=pred_kvsm_under,Reference=testing_under[,11])
confusionMatrix(conf_matrix_kvsm_under, stroke = 1)
```

```
## Confusion Matrix and Statistics
##
##          Reference
## Predicted  0  1
##         0 41  8
##         1 16 43
##
##                Accuracy : 0.7778
##                  95% CI : (0.6876, 0.8521)
##     No Information Rate : 0.5278
##     P-Value [Acc > NIR] : 6.833e-08
##
##                   Kappa : 0.5578
##
##  Mcnemar's Test P-Value : 0.153
##
##             Sensitivity : 0.7193
##             Specificity : 0.8431
##          Pos Pred Value : 0.8367
##          Neg Pred Value : 0.7288
##              Prevalence : 0.5278
##          Detection Rate : 0.3796
##    Detection Prevalence : 0.4537
##       Balanced Accuracy : 0.7812
##
##        'Positive' Class : 0
##
```

Model 5: random forest building

```
# Set the parameters for the train function
rftunegrid <- data.frame(
  .mtry=c(2,3,4,5,6), .splitrule="gini", .min.node.size=5
)
rfcontrol <- trainControl(
  method="oob", number=5, verboseIter=TRUE
)
```

```r
# Build the model
randomforest_under <- train(
  stroke~., training_under, method="ranger", tuneLength=3, tuneGrid= rftunegrid, trControl=rfcontrol
)
```

```
## + : mtry=2, splitrule=gini, min.node.size=5
## - : mtry=2, splitrule=gini, min.node.size=5
## + : mtry=3, splitrule=gini, min.node.size=5
## - : mtry=3, splitrule=gini, min.node.size=5
## + : mtry=4, splitrule=gini, min.node.size=5
## - : mtry=4, splitrule=gini, min.node.size=5
## + : mtry=5, splitrule=gini, min.node.size=5
## - : mtry=5, splitrule=gini, min.node.size=5
## + : mtry=6, splitrule=gini, min.node.size=5
## - : mtry=6, splitrule=gini, min.node.size=5
## Aggregating results
## Selecting tuning parameters
## Fitting mtry = 3, splitrule = gini, min.node.size = 5 on full training set
```

```r
randomforest_under
```

```
## Random Forest
##
## 252 samples
##  10 predictor
##   2 classes: '0', '1'
##
## No pre-processing
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa
##   2     0.7420635  0.4815799
##   3     0.7500000  0.4979127
##   4     0.7420635  0.4821700
##   5     0.7500000  0.4982935
##   6     0.7500000  0.4979127
##
## Tuning parameter 'splitrule' was held constant at a value of gini
##
## Tuning parameter 'min.node.size' was held constant at a value of 5
## Accuracy was used to select the optimal model using the largest value.
## The final values used for the model were mtry = 3, splitrule = gini
##  and min.node.size = 5.
```

```r
# Predict the testing data using the model built
randomforest_under_prediction <- predict(randomforest_under, newdata=beta_testing_under)

# Create a confusion matrix to assess the performance of the model on the testing data
confusionMatrix(randomforest_under_prediction, factor(testing_under[["stroke"]]), positive = "1")
```

```
## Confusion Matrix and Statistics
##
```

```
##           Reference
## Prediction  0  1
##          0 40  8
##          1 17 43
##
##                Accuracy : 0.7685
##                  95% CI : (0.6775, 0.8443)
##     No Information Rate : 0.5278
##     P-Value [Acc > NIR] : 2.104e-07
##
##                   Kappa : 0.5399
##
##  Mcnemar's Test P-Value : 0.1096
##
##             Sensitivity : 0.8431
##             Specificity : 0.7018
##          Pos Pred Value : 0.7167
##          Neg Pred Value : 0.8333
##              Prevalence : 0.4722
##          Detection Rate : 0.3981
##    Detection Prevalence : 0.5556
##       Balanced Accuracy : 0.7724
##
##        'Positive' Class : 1
##
```

Model 6: extreme gradient boosting tree

```
# Set the parameters for the train function
xgbgrid <- expand.grid(
  nrounds = 3500, max_depth = 7, eta = 0.01, gamma = 0.01,
  colsample_bytree = 0.75, min_child_weight = 0, subsample = 0.5
)

xgbcontrol <- trainControl(
  method = "cv", number = 5
)

# Build the model
xgb_under <- train(
  stroke ~ ., training_under, method = "xgbTree", tuneLength = 3, tuneGrid = xgbgrid, trControl = xgbco
)

xgb_under
```

```
## eXtreme Gradient Boosting
##
## 252 samples
##  10 predictor
##   2 classes: '0', '1'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 201, 202, 201, 202, 202
## Resampling results:
##
##   Accuracy   Kappa
##   0.7301176  0.4584377
##
## Tuning parameter 'nrounds' was held constant at a value of 3500
##
## Tuning parameter 'min_child_weight' was held constant at a value of 0
##
## Tuning parameter 'subsample' was held constant at a value of 0.5
```

```
# Predict the testing data using the model built
xbg_pred <- predict(xgb_under, newdata = beta_testing_under)

# Create a confusion matrix to assess the performance of the model on the testing data
confusionMatrix(xbg_pred, factor(testing_under[["stroke"]]), positive = "1")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 38  7
##          1 19 44
##
##                Accuracy : 0.7593
##                  95% CI : (0.6675, 0.8363)
##     No Information Rate : 0.5278
##     P-Value [Acc > NIR] : 6.161e-07
##
##                   Kappa : 0.5229
##
##  Mcnemar's Test P-Value : 0.03098
##
##             Sensitivity : 0.8627
##             Specificity : 0.6667
##          Pos Pred Value : 0.6984
##          Neg Pred Value : 0.8444
##              Prevalence : 0.4722
##          Detection Rate : 0.4074
##    Detection Prevalence : 0.5833
##       Balanced Accuracy : 0.7647
##
##        'Positive' Class : 1
##
```

Models developed using oversampling

Model 1: Logistic regression

```r
glm_all_over <- glm(stroke~., family=binomial(link = "logit"), data=training_over)
glm_over <- glm_all_over %>% stepAIC(trace = FALSE)
summary(glm_over)
```

```
##
## Call:
## glm(formula = stroke ~ gender + age + hypertension + heart_disease +
##     ever_married + work_type + avg_glucose_level + smoking_status,
##     family = binomial(link = "logit"), data = training_over)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.7563  -0.6286  -0.1230   0.7218   2.5002
##
## Coefficients:
##                    Estimate Std. Error z value Pr(>|z|)
## (Intercept)       -8.2111915  0.4315030 -19.029  < 2e-16 ***
## gender            -0.1903558  0.0897963  -2.120  0.03402 *
## age                0.0935111  0.0033173  28.189  < 2e-16 ***
## hypertension       0.5992638  0.1088342   5.506 3.67e-08 ***
## heart_disease      0.7698440  0.1366198   5.635 1.75e-08 ***
## ever_married      -0.3236938  0.1485721  -2.179  0.02935 *
## work_type          0.1309077  0.0471366   2.777  0.00548 **
## avg_glucose_level  0.0066687  0.0007319   9.111  < 2e-16 ***
## smoking_status     0.1918944  0.0610377   3.144  0.00167 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 6296.6  on 4542  degrees of freedom
## Residual deviance: 4076.0  on 4534  degrees of freedom
## AIC: 4094
##
## Number of Fisher Scoring iterations: 5
```

```r
# create a confusion matrix
pred_glm_over = predict(glm_over, testing_over, type="response")
pred_glm_over <- ifelse((pred_glm_over>=0.3), 1, 0) # This value was preset
conf_matrix_over <- table(pred_glm_over, testing_over$stroke, deparse.level = 2)
conf_matrix_over
```

```
##              testing_over$stroke
## pred_glm_over   1   2
##             0 621  63
##             1 314 949
```

Model 2: Linear regression to predict using gradient descent

Preprocess the data

```r
x_var_train_for_lm_over  <- training_over %>%
  dplyr::select(age, bmi)

# Create a design matrix for the training data
X_train_for_lm_over <- model.matrix(~. -1, data = x_var_train_for_lm_over) # -1 removes the intercept f

train_over_2 <- training_over
train_over_2$stroke <- as.numeric(train_over_2$stroke)

Y_train_for_lm_over <- train_over_2 %>% pull(stroke)

x_var_test_for_lm_over  <- testing_over %>%
  dplyr::select(age, bmi)

# Create a design matrix for the testing data
X_test_for_lm_over <- model.matrix(~. -1, data = x_var_test_for_lm_over) #

test_over_2 <- training_over
test_over_2$stroke <- as.numeric(test_over_2$stroke)

Y_test_for_lm_over <- test_over_2 %>% pull(stroke)

dim(X_train_for_lm_over) #4543
```

```
## [1] 4543    2
```

```r
dim(X_test_for_lm_over) #1947
```

```
## [1] 1947    2
```

```r
# Initialize coefficients
gamma = 0.000001 # this is the step size
p = 2 # This is the number of predictors

beta0 <- rep(0, p) # This is the vector of all 0s
previous_loss <- least_squares_loss(X_train_for_lm_over, Y_train_for_lm_over, beta0) # Loss function at
grad0 <- least_squares_gradient(X_train_for_lm_over, Y_train_for_lm_over, beta0) # Initialise gradient
beta1 <- beta0 - gamma * grad0
next_loss <- least_squares_loss(X_train_for_lm_over, Y_train_for_lm_over, beta1)
previous_beta <- beta1
steps <- 1

while (abs(previous_loss - next_loss) > 0.00001) {
  gradn <- least_squares_gradient(X_train_for_lm_over, Y_train_for_lm_over, previous_beta)
  # Refine update by allowing step size to change at each iteration. Make step size a sequence of numbe
  next_beta <- previous_beta - (0.99)^steps * gradn / sqrt(sum(gradn^2))
  # We rescale the gradient i.e. use sqrt(sum(gradn^2)) to prevent the algorithm from diverging
  steps <- steps + 1
  previous_beta <- next_beta
  previous_loss <- next_loss
  next_loss <- least_squares_loss(X_train_for_lm_over, Y_train_for_lm_over, next_beta)
}
```

```r
# Use the model the parameter estimates obtained using gradient descent to predict the testing data
pred_lm_grad_desc_over = X_test_for_lm_over %*% previous_beta
pred_lm_grad_desc_over = round(as.integer(pred_lm_grad_desc_over))
pred_lm_grad_desc_over = as.factor(pred_lm_grad_desc_over + 1)

# Create a confusion matrix to see how well the testing data is predicted
table(pred_lm_grad_desc_over, testing_over$stroke, deparse.level = 2)
```

```
##                       testing_over$stroke
## pred_lm_grad_desc_over    1    2
##                      1  264    3
##                      2  671 1008
##                      3    0    1
```

Model 3: Lasso (least absolute shrinkage and selection operator) regression

Preprocess this data to use for lasso

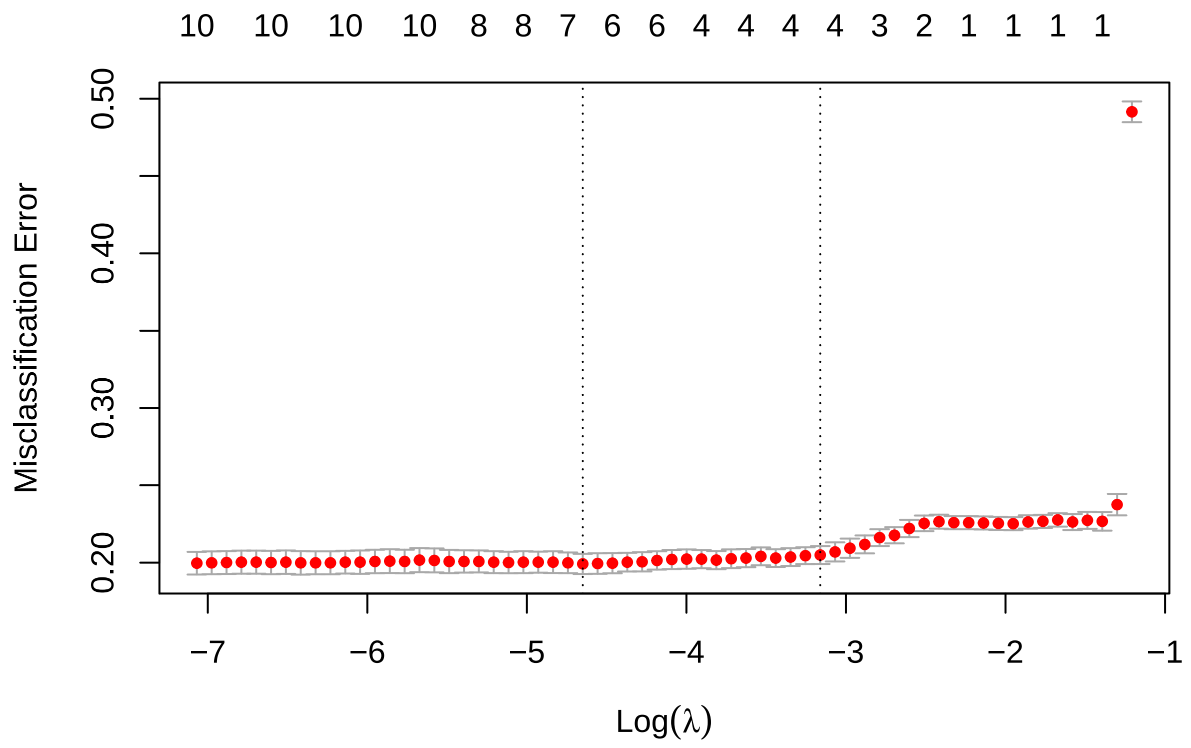
```r
x_var_train_for_lasso_over  <- training_over %>%
  dplyr::select(-stroke)
# Create the design matrix for the training data
X_train_for_lasso_over <- model.matrix(~. -1, data = x_var_train_for_lasso_over)

Y_train_for_lasso_over <- training_over %>% pull(stroke)

x_var_test_for_lasso_over  <- testing_over %>%
  dplyr::select(-stroke)
# Create the design matrix for the testing data
X_test_for_lasso_over <- model.matrix(~. -1, data = x_var_test_for_lasso_over)

Y_test_for_lasso_over <- testing_over %>% pull(stroke)

dim(X_train_for_lasso_over)
```

```
## [1] 4543   10
```

```r
dim(X_test_for_lasso_over)
```

```
## [1] 1947   10
```

```r
#Perform 10 fold cross validation using the misclassification rate to find the best lambda
lasso_cv_over = cv.glmnet(X_train_for_lasso_over,
                  Y_train_for_lasso_over,
                  family = "binomial",
                  type.measure = "class") # type.measure = "class" allows us to use the misclassificati
plot(lasso_cv_over)
```



```
# Fit the lasso model on the training data
lasso_over <- glmnet(X_train_for_lasso_over,
                     Y_train_for_lasso_over,
                     alpha = 1,
                     family = "binomial",
                     lambda = lasso_cv_over$lambda.min)

coef(lasso_over)  # Observe what coefficients are included in the model
```

```
## 11 x 1 sparse Matrix of class "dgCMatrix"
##                             s0
## (Intercept)       -7.586969947
## gender            -0.006107622
## age                0.085735590
## hypertension       0.507606465
## heart_disease      0.593544989
## ever_married       .
## work_type          0.057024201
## Residence_type     .
## avg_glucose_level  0.005307559
## bmi                .
## smoking_status     0.072539170
```

```
# Making predictions based off of testing data
pred_lasso_over <- lasso_over %>% predict(newx = X_test_for_lasso_over)
pred_lasso_over <- ifelse(pred_lasso_over >= 0.3, 1, 0)
table(pred_lasso_over, Y_test_for_lasso_over, deparse.level = 2)
```

```
##                 Y_test_for_lasso_over
## pred_lasso_over   1   2
##               0 775 201
##               1 160 811
```

Model 4: Kernel Methods

```
# Find the value of gamma that minimizes the number of false negatives
try <- -100:100 # List of possible values for gamma
false_neg <- data.frame()
error <- data.frame()
for (gam in c(-100:100)){
  # Create the model using training data
  kvsm_over <- ksvm(stroke ~ ., kernal = 'rbfdot', data = training_over, gamma = gam)
  pred_kvsm_over <- predict(kvsm_over, type = 'response', newdata = training_over)
  pred_kvsm_over <- round(as.numeric(pred_kvsm_over))
  pred_kvsm_over <- as.factor(pred_kvsm_over)
  conf_mat <- as.factor(training_over[,11])
  # Substract accuracy from the false positives
  ans = (confusionMatrix(pred_kvsm_over, conf_mat)[[2]][3]) - confusionMatrix(pred_kvsm_over, conf_mat)
  # As accuracy < 1, this will give us the value with the highest accuracy out of those with the joint
  false_neg <- rbind(false_neg, ans)
}

try[which(false_neg == min(false_neg))] # Choose one of these values as gamma
```

```
## [1] -100  -91   38   47
```

```
kvsm_over <- ksvm(stroke ~ ., kernal = 'rbfdot', data = training_over, gamma = 10)
pred_kvsm_over <- predict(kvsm_over, type = 'response', newdata = testing_over)
pred_kvsm_over <- round(as.numeric(pred_kvsm_over))
pred_kvsm_over <- as.factor(pred_kvsm_over)
conf_mat <- as.factor(testing_over[,11])
confusionMatrix(pred_kvsm_over, conf_mat)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   1   2
##          1 836 105
##          2  99 907
##
##                Accuracy : 0.8952
##                  95% CI : (0.8808, 0.9085)
##     No Information Rate : 0.5198
##     P-Value [Acc > NIR] : <2e-16
##
```

```
##                   Kappa : 0.7902
##
##  Mcnemar's Test P-Value : 0.7263
##
##             Sensitivity : 0.8941
##             Specificity : 0.8962
##          Pos Pred Value : 0.8884
##          Neg Pred Value : 0.9016
##              Prevalence : 0.4802
##          Detection Rate : 0.4294
##    Detection Prevalence : 0.4833
##       Balanced Accuracy : 0.8952
##
##        'Positive' Class : 1
##
```

Model 5: Random Forest

```
# Build the model using the training data
randomforest_over <- train(
  stroke~., training_over, method="ranger", tuneLength=3, tuneGrid= rftunegrid, trControl=rfcontrol
)
```

```
## + : mtry=2, splitrule=gini, min.node.size=5
## - : mtry=2, splitrule=gini, min.node.size=5
## + : mtry=3, splitrule=gini, min.node.size=5
## - : mtry=3, splitrule=gini, min.node.size=5
## + : mtry=4, splitrule=gini, min.node.size=5
## - : mtry=4, splitrule=gini, min.node.size=5
## + : mtry=5, splitrule=gini, min.node.size=5
## - : mtry=5, splitrule=gini, min.node.size=5
## + : mtry=6, splitrule=gini, min.node.size=5
## - : mtry=6, splitrule=gini, min.node.size=5
## Aggregating results
## Selecting tuning parameters
## Fitting mtry = 5, splitrule = gini, min.node.size = 5 on full training set
```

```
randomforest_over
```

```
## Random Forest
##
## 4543 samples
##   10 predictor
##    2 classes: '1', '2'
##
## No pre-processing
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa
##   2     0.9511336  0.9021617
##   3     0.9575171  0.9149570
##   4     0.9610390  0.9220165
##   5     0.9630200  0.9259846
##   6     0.9603786  0.9207001
##
## Tuning parameter 'splitrule' was held constant at a value of gini
##
## Tuning parameter 'min.node.size' was held constant at a value of 5
## Accuracy was used to select the optimal model using the largest value.
## The final values used for the model were mtry = 5, splitrule = gini
##  and min.node.size = 5.
```

```
# Making predictions based off of testing data
pred_randomforest_over<- predict(randomforest_over, newdata=beta_testing_over)

# Create a confusion matrix
confusionMatrix(pred_randomforest_over, factor(testing_over[["stroke"]]), positive = "1")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   1   2
##          1 917  44
##          2  18 968
##
##                Accuracy : 0.9682
##                  95% CI : (0.9594, 0.9755)
##     No Information Rate : 0.5198
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9363
##
##  Mcnemar's Test P-Value : 0.001498
##
##             Sensitivity : 0.9807
##             Specificity : 0.9565
##          Pos Pred Value : 0.9542
##          Neg Pred Value : 0.9817
##              Prevalence : 0.4802
##          Detection Rate : 0.4710
##    Detection Prevalence : 0.4936
##       Balanced Accuracy : 0.9686
##
##        'Positive' Class : 1
##
```

Model 6: Extreme gradient boosting tree

```
# Build the model using the training data
xgb_over <- train(
  stroke ~ ., training_over, method = "xgbTree", tuneLength = 3, tuneGrid = xgbgrid, trControl = xgbcon
)
```

```r
xgb_over
```

```
## eXtreme Gradient Boosting
##
## 4543 samples
##   10 predictor
##    2 classes: '1', '2'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 3635, 3634, 3634, 3635, 3634
## Resampling results:
##
##   Accuracy   Kappa
##   0.9621395  0.9242213
##
## Tuning parameter 'nrounds' was held constant at a value of 3500
##
## Tuning parameter 'min_child_weight' was held constant at a value of 0
##
## Tuning parameter 'subsample' was held constant at a value of 0.5
```

```r
# Making predictions based off of testing data
pred_xgb_over <- predict(xgb_over, newdata = beta_testing_over)

# Create a confusion matrix
confusionMatrix(pred_xgb_over, factor(testing_over[["stroke"]]), positive = "1")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   1   2
##          1 921  47
##          2  14 965
##
##                Accuracy : 0.9687
##                  95% CI : (0.9599, 0.976)
##     No Information Rate : 0.5198
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9373
##
##  Mcnemar's Test P-Value : 4.182e-05
##
##             Sensitivity : 0.9850
##             Specificity : 0.9536
##          Pos Pred Value : 0.9514
##          Neg Pred Value : 0.9857
##              Prevalence : 0.4802
##          Detection Rate : 0.4730
##    Detection Prevalence : 0.4972
##       Balanced Accuracy : 0.9693
```

```
##
##        'Positive' Class : 1
##
```