Stat 419 Group Project

Molly Behrends, Payton Bokowy, Freeman Chen, Marvin Lim

12/15/2021

Introduction

Dataset Information

The dataset for this project was obtained data about strokes from kaggle.com which was collected from the World Health Organization. The dataset contains information about each person's age, heart disease, hypertension, marital status, BMI, and their smoking status to list a few of the variables.

This dataset was chosen because strokes are the second leading cause of death in the world. It would be helpful to be able to know what features can possibly lead to a stroke, so we are aware and can reduce those factors.

Reads in the data

```
data = read.csv("healthcare-dataset-stroke-data.csv") # reads data file
#data = subset(data, select = -c(id)) # gets rid of the id

# changes the predictors to factors
data$gender = as.factor(data$gender)
data$hypertension = as.factor(data$hypertension)
data$heart_disease = as.factor(data$heart_disease)
data$ever_married = as.factor(data$ever_married)
data$work_type = as.factor(data$work_type)
data$Residence_type = as.factor(data$Residence_type)
data$smoking_status = as.factor(data$smoking_status)
data$stroke = as.factor(data$stroke)
data$bmi = as.numeric(data$bmi)
```

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

```
data = na.omit(data)
str(data)
```

```
## 'data.frame':    4908 obs. of  11 variables:
##  $ gender           : Factor w/ 2 levels "Female","Male": 2 2 1 1 2 2 1 1 1 1 ...
##  $ age              : num  67 80 49 79 81 74 69 78 81 61 ...
##  $ hypertension     : Factor w/ 2 levels "0","1": 1 1 1 2 1 2 1 1 2 1 ...
##  $ heart_disease    : Factor w/ 2 levels "0","1": 2 2 1 1 1 2 1 1 1 2 ...
```

```
##  $ ever_married     : Factor w/ 2 levels "No","Yes": 2 2 2 2 2 2 1 2 2 2 ...
##  $ work_type        : Factor w/ 5 levels "children","Govt_job",..: 4 4 4 5 4 4 4 4 4 2 ...
##  $ Residence_type   : Factor w/ 2 levels "Rural","Urban": 2 1 2 1 2 1 2 2 1 1 ...
##  $ 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   : Factor w/ 4 levels "formerly smoked",..: 1 2 3 2 1 2 2 4 2 3 ...
##  $ stroke           : Factor w/ 2 levels "0","1": 2 2 2 2 2 2 2 2 2 2 ...
##  - attr(*, "na.action")= 'omit' Named int [1:201] 2 9 14 20 28 30 44 47 51 52 ...
##   ..- attr(*, "names")= chr [1:201] "2" "9" "14" "20" ...
```

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

```
##
##          0          1
## 0.95741646 0.04258354
```

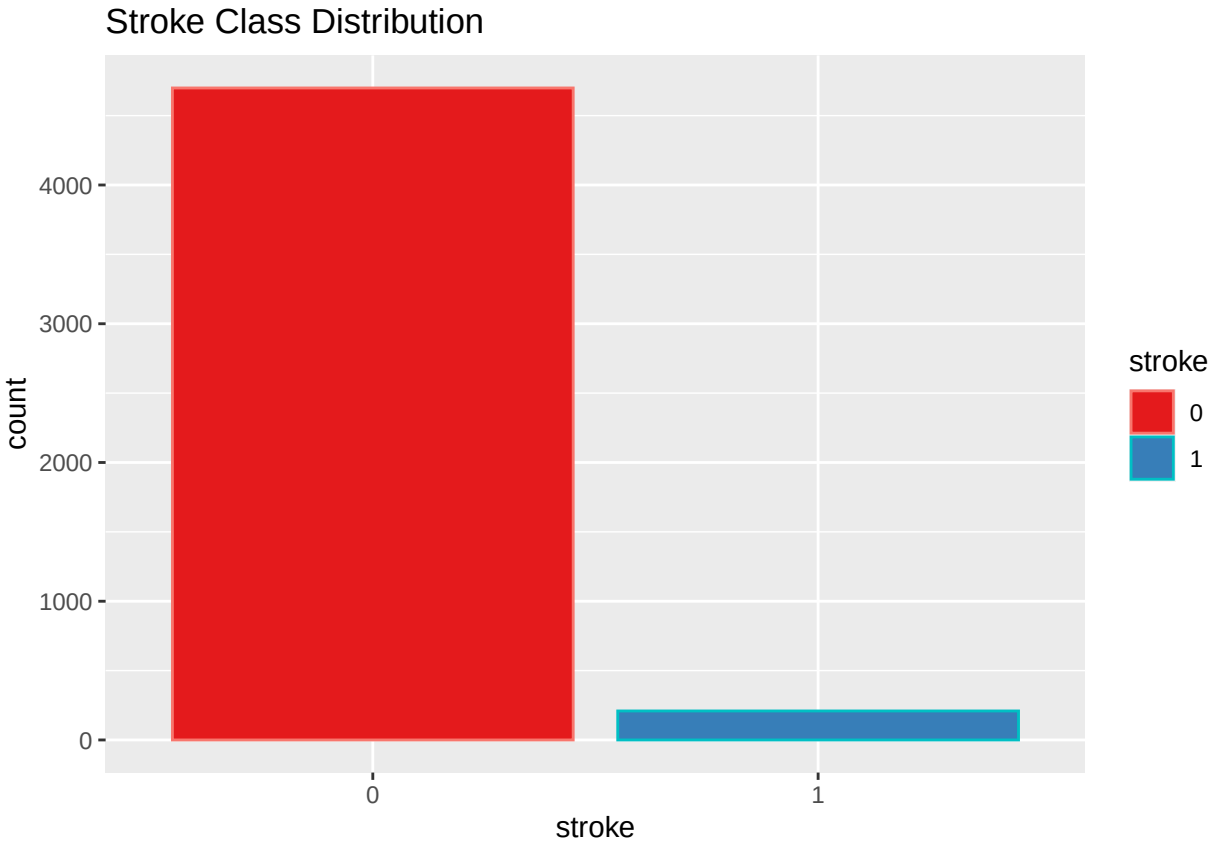
Splits the data into a testing and training set

```
set.seed(1)
n = dim(data)[1]

dt = sort(sample(n, n*.7))
training = data[dt,]
testing = data[-dt,]
```

Processing Problems

When the data was loaded in, we noticed the data was highly unbalanced. Only about 5% of the data results in strokes, while the other 95% results in no stroke. This resulted in the models being fitted on unbalanced data. In order to combat this issue, we created synthetic data.

```
ggplot(data=data, aes(x=stroke, color = stroke)) +
  geom_bar(aes(fill= stroke)) + ggtitle("Stroke Class Distribution") +
  scale_fill_brewer(palette = "Set1")
```



From the graph we can see that the response variable are higher unbalance, which will lead to a problem that even model predict every sample as likely non-stroke, it will still likely to have a good accuracy, so in this case we are more looking for the error rates from the model instead just look at the accuracy.

Synthetic Data

after research we found that 'rose' library might gave us a better chance to make the dataset look a bit 'balance', rose functions is design to deal with binary classification problems in the presence of unbalanced classes by generating synthetic balanced samples(https://cran.r-project.org/web/packages/ROSE/ROSE.pdf)

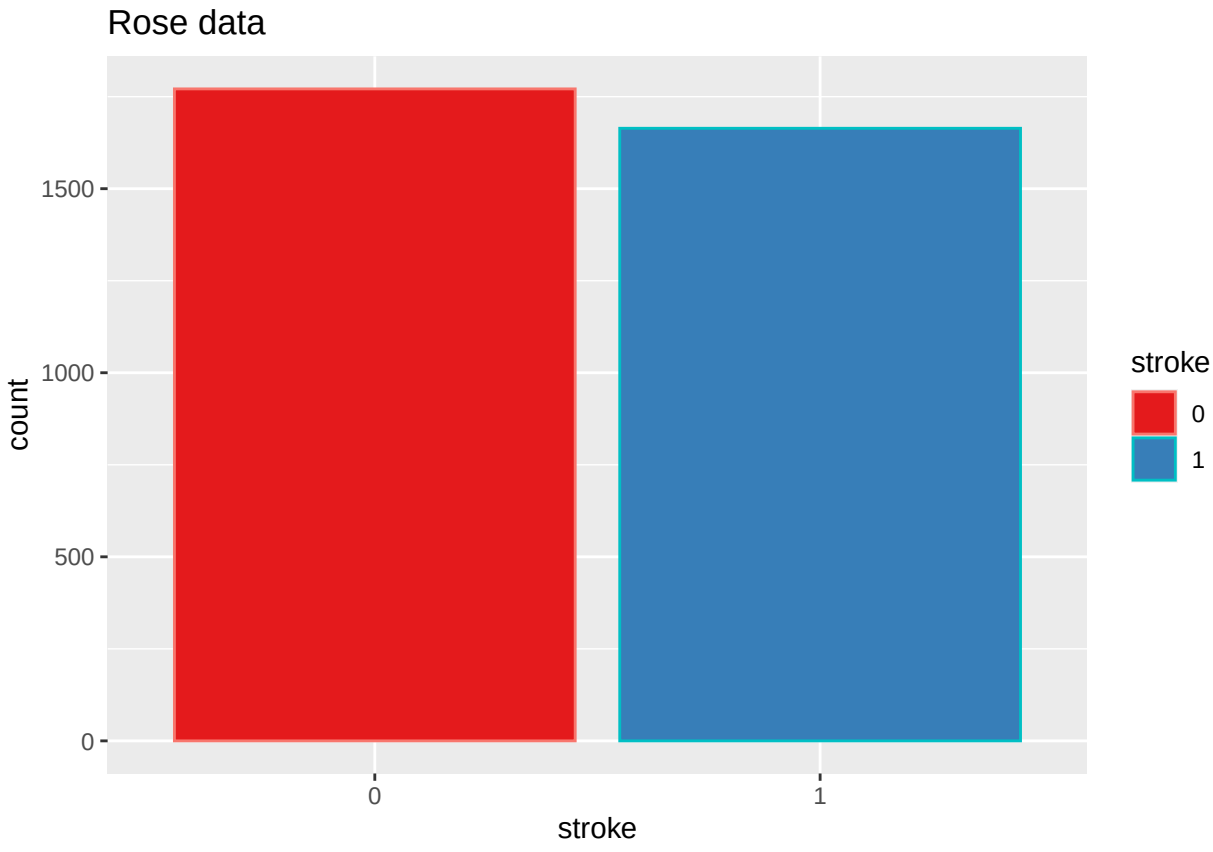
```
set.seed(1)

syn_train = ROSE(stroke~.,data=training)$data
table(syn_train$stroke)
```

```
##
##    0    1
## 1771 1664
```

```
syn_test = ROSE(stroke~.,data=testing)$data
table(syn_test$stroke)
```

```
##
##   0   1
## 766 707
```

```r
num_train = syn_train
num_test = syn_test

num_train$gender = as.numeric(as.factor(num_train$gender))
num_train$ever_married = as.numeric(as.factor(num_train$ever_married))
num_train$work_type = as.numeric(as.factor(num_train$work_type))
num_train$Residence_type = as.numeric(as.factor(num_train$Residence_type))
num_train$smoking_status = as.numeric(as.factor(num_train$smoking_status))

num_test$gender = as.numeric(as.factor(num_test$gender))
num_test$ever_married = as.numeric(as.factor(num_test$ever_married))
num_test$work_type = as.numeric(as.factor(num_test$work_type))
num_test$Residence_type = as.numeric(as.factor(num_test$Residence_type))
num_test$smoking_status = as.numeric(as.factor(num_test$smoking_status))

ggplot(data=syn_train, aes(x=stroke, color = stroke)) +
  geom_bar(aes(fill= stroke)) + ggtitle("Rose data") +
  scale_fill_brewer(palette = "Set1")
```

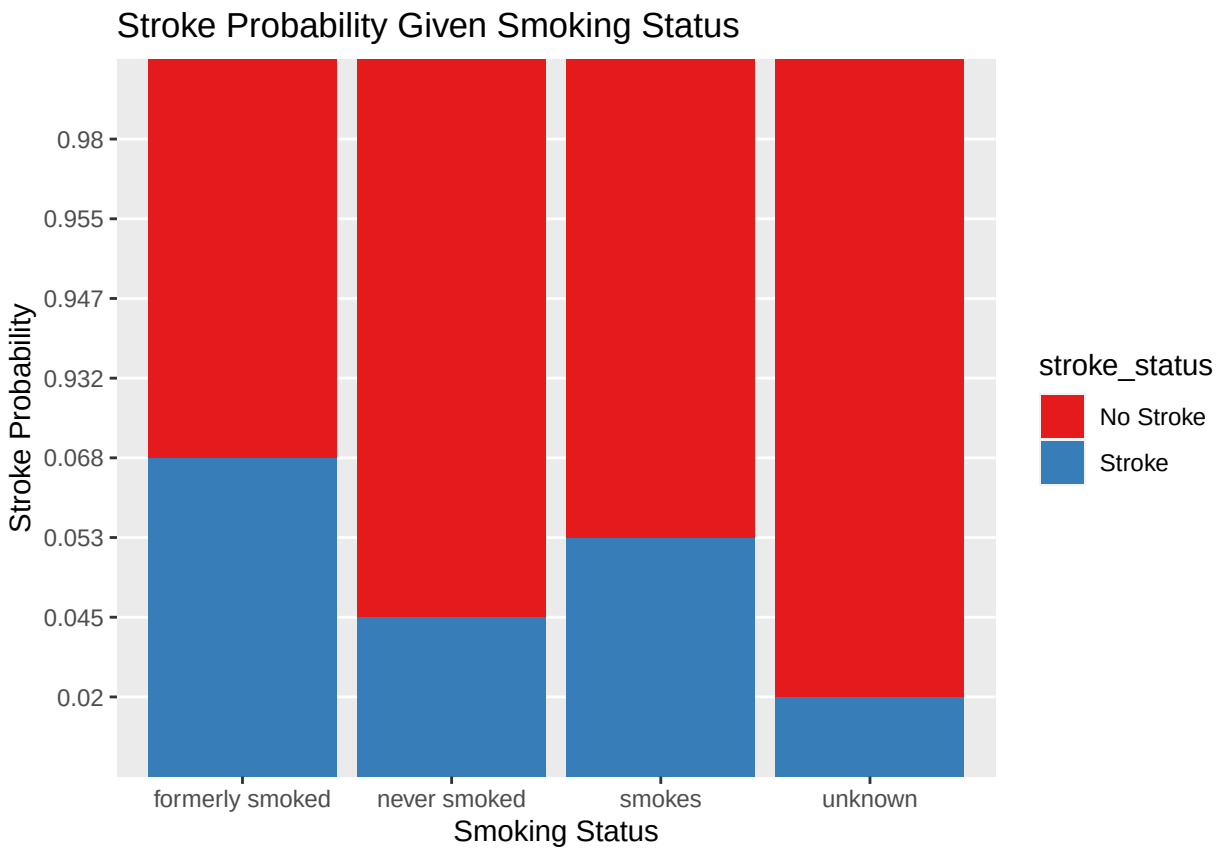


Research Questions

Will smoking status of an individual increase the chance of having a stroke?

This research question can be answered by analyzing the data that consists of individual's smoking and stroke status. We can answer this question by looking at the probabilities of each smoking status group and the stroke classification.

```
n = dim(data)[1]
past_smoked_data = data[data$smoking_status == "formerly smoked", ]
past_smoked_prob = dim(past_smoked_data[past_smoked_data$stroke == 1, ])[1]/
  dim(past_smoked_data)[1]
past_smoked_no_stroke = dim(past_smoked_data[past_smoked_data$stroke == 0,
                                             ])[1]/ dim(past_smoked_data)[1]

never_smoked_data = data[data$smoking_status == "never smoked", ]
never_smoked_prob = dim(never_smoked_data[never_smoked_data$stroke == 1, ])[1]/
  dim(never_smoked_data)[1]
never_smoked_no_stroke = dim(never_smoked_data[never_smoked_data$stroke == 0,
                                               ])[1]/ dim(never_smoked_data)[1]

smokes_data = data[data$smoking_status == "smokes", ]
smokes_prob = dim(smokes_data[smokes_data$stroke == 1, ])[1]/
  dim(smokes_data)[1]
smokes_no_stroke = dim(smokes_data[smokes_data$stroke == 0, ])[1]/
  dim(smokes_data)[1]

u_smoke_data = data[data$smoking_status == "Unknown", ]
u_smoke_prob = dim(u_smoke_data[u_smoke_data$stroke == 1, ])[1]/
  dim(u_smoke_data)[1]
u_smoke_no_stroke = dim(u_smoke_data[u_smoke_data$stroke == 0, ])[1]/
  dim(u_smoke_data)[1]

smoking_status = c("formerly smoked", "never smoked", "smokes", "unknown",
                   "formerly smoked", "never smoked", "smokes", "unknown")
stroke_status = c("Stroke", "Stroke", "Stroke", "Stroke", "No Stroke",
                  "No Stroke", "No Stroke", "No Stroke")
smoking_prob = c(round(past_smoked_prob, 3), round(never_smoked_prob, 3),
                 round(smokes_prob, 3), round(u_smoke_prob, 3),
                 round(past_smoked_no_stroke, 3),
                 round(never_smoked_no_stroke, 3), round(smokes_no_stroke, 3),
                 round(u_smoke_no_stroke, 3))

smoking_data = as.data.frame(cbind(smoking_status, stroke_status,
                                   smoking_prob))

ggplot(smoking_data, aes(fill=stroke_status, y=smoking_prob,
                         x=smoking_status)) +
  geom_bar(position = position_stack(reverse = TRUE), stat="identity") +
  xlab("Smoking Status") + ylab("Stroke Probability") +
  ggtitle("Stroke Probability Given Smoking Status") +
  scale_fill_brewer(palette = "Set1")
```



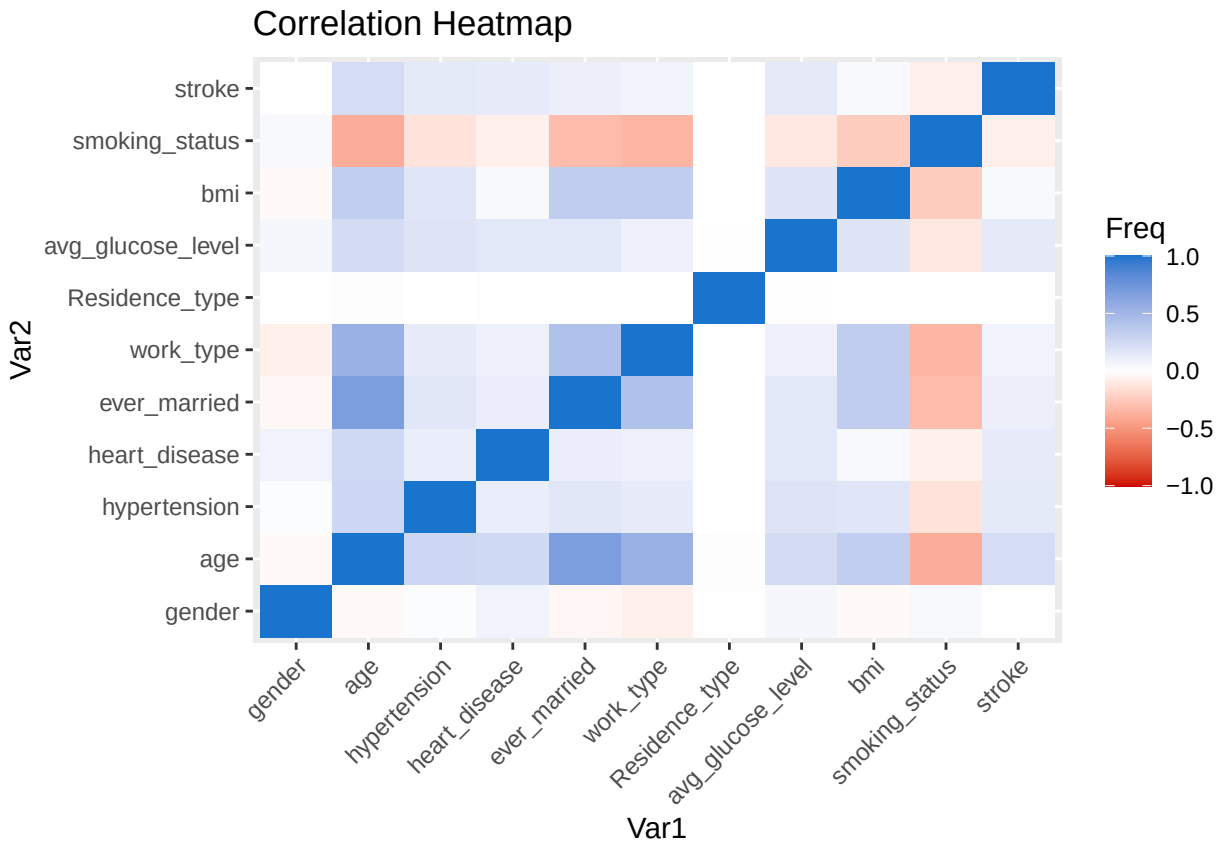
What variables best classify if a patient is likely to have a stroke?

The answer to this question would be helpful to know, when a patient comes in with the highly correlated health issues, a doctor can be on the look out for a possible stroke. If a doctor and a patient know they are at high risk for a stroke, this can help prevent a stroke by getting the right treatment beforehand. To answer this question, we will look at the correlation between all the variables and the outcome of having a stroke.

Correlation Visualization

```
cor_data = data
cor_data$gender = as.numeric(cor_data$gender)
cor_data$hypertension = as.numeric(cor_data$hypertension)
cor_data$heart_disease = as.numeric(cor_data$heart_disease)
cor_data$ever_married = as.numeric(cor_data$ever_married)
cor_data$work_type = as.numeric(cor_data$work_type)
cor_data$Residence_type = as.numeric(cor_data$Residence_type)
cor_data$smoking_status = as.numeric(cor_data$smoking_status)
cor_data$stroke = as.numeric(cor_data$stroke)
cors = cor(cor_data)
cor_df = data.frame(as.table(cors))

ggplot(data=cor_df, mapping=aes(Var1, Var2))+
  geom_tile(mapping = aes(fill = Freq)) +
  scale_fill_gradient2(low = "red3", high = "dodgerblue3", mid = "white",
```

```
  midpoint = 0, limit = c(-1,1)) + ggtitle("Correlation Heatmap") +
scale_x_discrete(guide = guide_axis(angle=45))
```

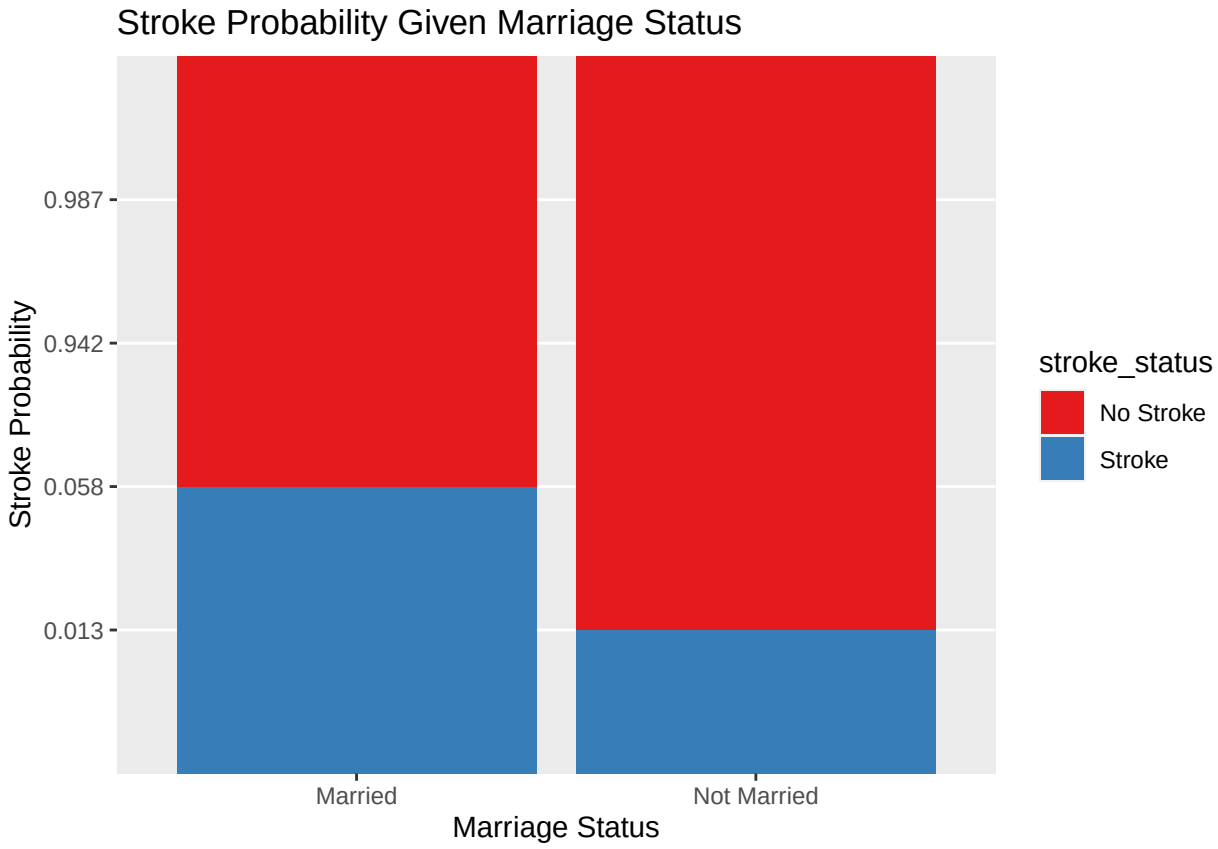


Does marriage status affect the likelihood of having a stroke?

This research question can be answered by analyzing the data that consists of individual's marriage and stroke status. We can answer this question by looking at the probabilities of each marital status group and the stroke classification.

```
married_data = data[data$ever_married == "Yes", ]
married_stroke_prob = dim(married_data[married_data$stroke == 1, ])[1]/ dim(married_data)[1]
married_no_stroke = dim(married_data[married_data$stroke == 0, ])[1]/ dim(married_data)[1]

not_married_data = data[data$ever_married == "No", ]
not_married_stroke_prob = dim(not_married_data[not_married_data$stroke == 1,
                                               ])[1]/ dim(not_married_data)[1]
not_married_no_stroke = dim(not_married_data[not_married_data$stroke == 0,
                                             ])[1]/ dim(not_married_data)[1]

marriage_status = c("Married", "Not Married", "Married", "Not Married")
stroke_status = c("Stroke", "Stroke", "No Stroke", "No Stroke")
marriage_prob = c(round(married_stroke_prob, 3),
                  round(not_married_stroke_prob, 3),
                  round(married_no_stroke, 3),
                  round(not_married_no_stroke, 3))
```

```r
marriage_data = as.data.frame(cbind(marriage_status, stroke_status,
                                    marriage_prob))

ggplot(marriage_data, aes(fill=stroke_status, y=marriage_prob,
                          x=marriage_status)) +
  geom_bar(position = position_stack(reverse = TRUE), stat="identity") +
  xlab("Marriage Status") + ylab("Stroke Probability") +
  ggtitle("Stroke Probability Given Marriage Status") +
  scale_fill_brewer(palette = "Set1")
```



Do people with heart disease have a higher risk of having a stroke?

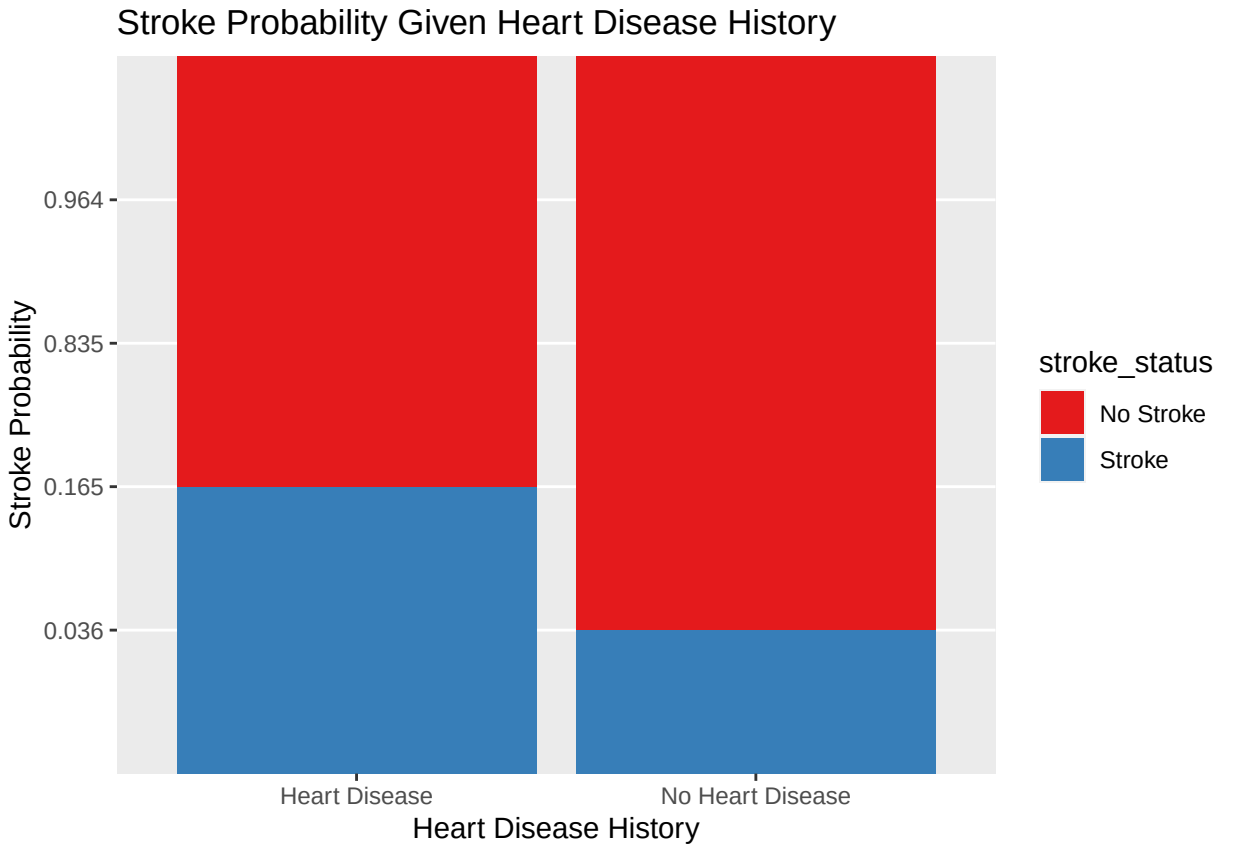
This research question can be answered by analyzing the data that consists of individual's heart disease and stroke status. We can answer this question by looking at the probabilities of each person's heart disease status and the stroke classification.

```r
no_heart_data = data[data$heart_disease == 0, ]
no_heart_prob = dim(no_heart_data[no_heart_data$stroke == 1, ])[1]/ dim(no_heart_data)[1]

no_heart_no_stroke = dim(no_heart_data[no_heart_data$stroke == 0, ])[1]/ dim(no_heart_data)[1]

heart_data = data[data$heart_disease == 1, ]
heart_stroke_prob = dim(heart_data[heart_data$stroke == 1, ])[1]/ dim(heart_data)[1]

heart_no_stroke_prob = dim(heart_data[heart_data$stroke == 0, ])[1]/ dim(heart_data)[1]
```

```
heart_status = c("No Heart Disease", "Heart Disease", "No Heart Disease", "Heart Disease")
stroke_status = c("Stroke", "Stroke", "No Stroke", "No Stroke")

heart_prob = c(round(no_heart_prob, 3), round(heart_stroke_prob, 3),
               round(no_heart_no_stroke, 3), round(heart_no_stroke_prob, 3))

heart_data = as.data.frame(cbind(heart_status, stroke_status, heart_prob))
heart_data
```

```
##       heart_status stroke_status heart_prob
## 1 No Heart Disease        Stroke      0.036
## 2    Heart Disease        Stroke      0.165
## 3 No Heart Disease     No Stroke      0.964
## 4    Heart Disease     No Stroke      0.835
```

```
ggplot(heart_data, aes(fill=stroke_status, y=heart_prob, x=heart_status)) +
    geom_bar(position = position_stack(reverse = TRUE), stat="identity") +
  xlab("Heart Disease History") + ylab("Stroke Probability") +
  ggtitle("Stroke Probability Given Heart Disease History") +
  scale_fill_brewer(palette = "Set1")
```



What classification model is best at predicting having stroke class?

To answer this question, we implemented logistic regression, linear SVM, LDA, and KNN. The answer to this question is helpful in knowing what model to use when predicting if a patient is likely to have a stroke

or not based on their health background.

Logistic Regression with Unbalanced Data

```
# performs logistic regression on the stroke data
full_logistic = glm(stroke~., data = training, family = "binomial")
summary(full_logistic)
```

```
##
## Call:
## glm(formula = stroke ~ ., family = "binomial", data = training)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.1853  -0.2798  -0.1350  -0.0701   3.3995
##
## Coefficients:
##                               Estimate Std. Error z value Pr(>|z|)
## (Intercept)                 -6.924004   1.096037  -6.317 2.66e-10 ***
## genderMale                  -0.217819   0.192400  -1.132 0.257586
## age                          0.080518   0.008055   9.996  < 2e-16 ***
## hypertension1                0.503586   0.212268   2.372 0.017672 *
## heart_disease1               0.290192   0.265386   1.093 0.274185
## ever_marriedYes             -0.224076   0.304718  -0.735 0.462122
## work_typeGovt_job           -1.219773   1.172320  -1.040 0.298118
## work_typeNever_worked      -10.268746 400.708503  -0.026 0.979555
## work_typePrivate            -1.324364   1.158821  -1.143 0.253099
## work_typeSelf-employed      -1.710292   1.184446  -1.444 0.148751
## Residence_typeUrban         -0.036733   0.183037  -0.201 0.840943
## avg_glucose_level            0.005245   0.001586   3.307 0.000943 ***
## bmi                          0.004218   0.014271   0.296 0.767556
## smoking_statusnever smoked  -0.114802   0.228160  -0.503 0.614849
## smoking_statussmokes         0.248684   0.283563   0.877 0.380487
## smoking_statusUnknown       -0.412705   0.316603  -1.304 0.192389
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1182.86  on 3434  degrees of freedom
## Residual deviance:  912.51  on 3419  degrees of freedom
## AIC: 944.51
##
## Number of Fisher Scoring iterations: 14
```

```
# predicts on the testing data
probs = predict(full_logistic, type = "response", testing)

n = dim(testing)[1]
t = 0.5
pred.label = c()
```

```
pred.label = rep(1, n)
pred.label[probs>t] = 0

table(predicted = pred.label, truth = testing$stroke) # confusion matrix
```

```
##          truth
## predicted    0    1
##         0    0    2
##         1 1406   65
```

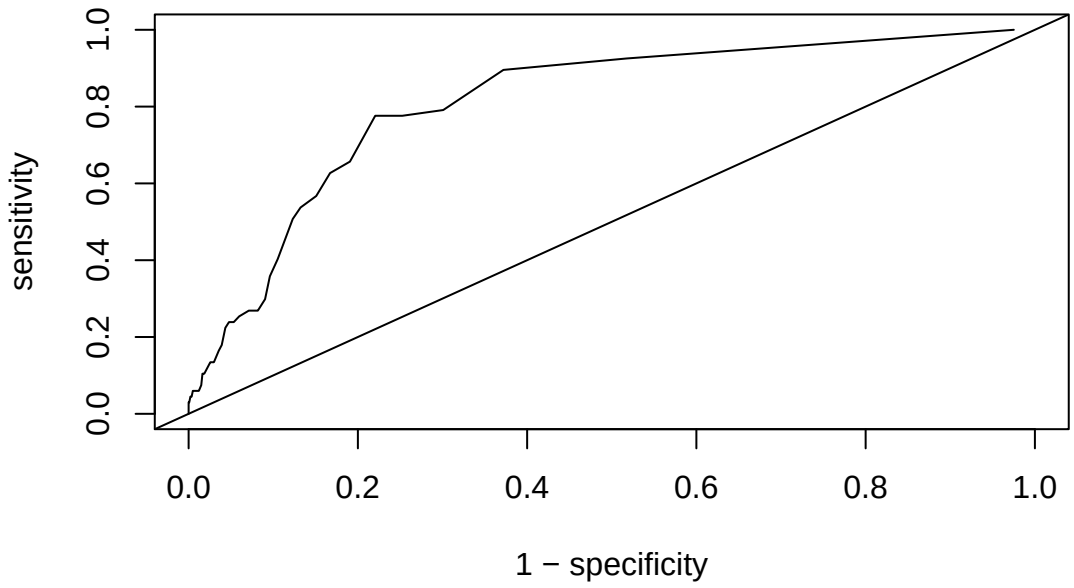
```
log1_error = (1406 + 2)/ n
log1_error
```

```
## [1] 0.9558724
```

```
log1_acc = (0+65) /n
log1_acc
```

```
## [1] 0.04412763
```

```
# plots the ROC Curve
tseq = seq(0.001, 0.999, length.out = 100)
sensitivity = c(); specificity = c()

for (j in 1:length(tseq)){
t = tseq[j]

pred.label[probs>t] = 1
pred.label[probs < t] = 0

p.ind = which(testing$stroke == 1)
sensitivity[j] = mean(pred.label[p.ind] == testing$stroke[p.ind])

n.ind = which(testing$stroke == 0)
specificity[j] = mean(pred.label[n.ind] == testing$stroke[n.ind])
}

plot(1 - specificity, sensitivity, type = "l", xlim = c(0, 1), ylim = c(0, 1))
abline(a = 0, b = 1)
```



Logistic Regression with Synthetic Data

```
full_logistic = glm(stroke~., data = syn_train, family = "binomial")

probs = predict(full_logistic, type = "response", syn_test)
n = dim(syn_test)[1]
t = 0.5
pred.label = c()
pred.label = rep(0, n)
pred.label[probs>t] = 1
table(predicted = pred.label, truth = syn_test$stroke)
```

```
##          truth
## predicted   0   1
##         0 557 178
##         1 209 529
```

```
log_TPM = (205 + 165) / n
log_TPM
```

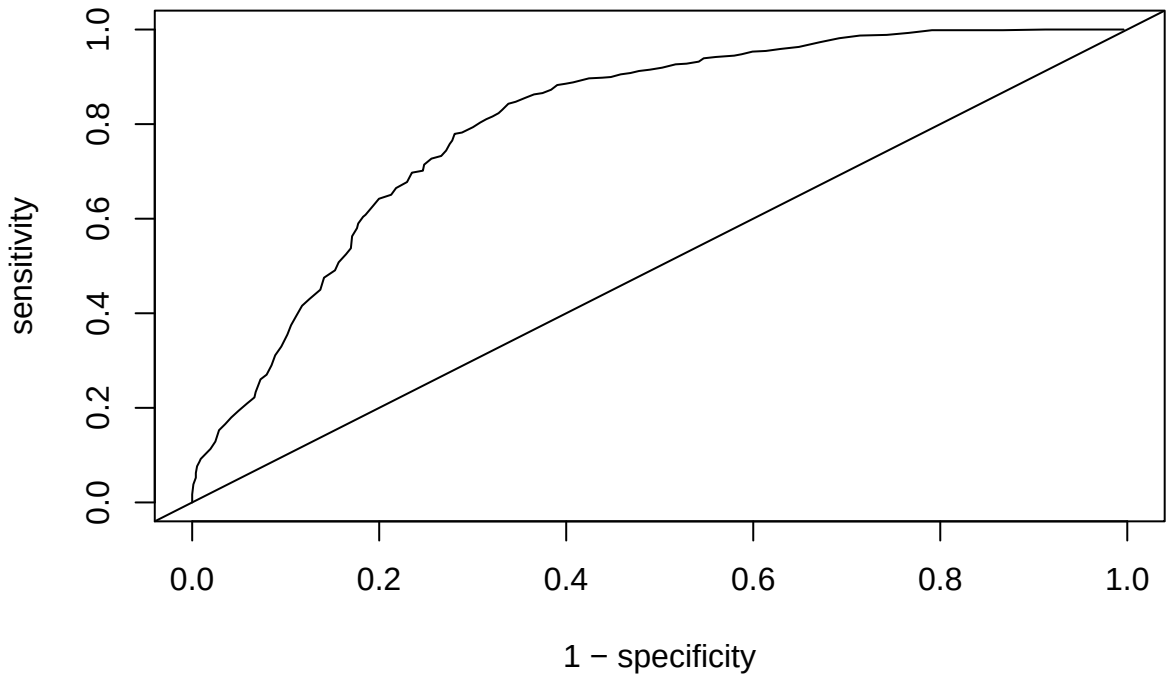
```
## [1] 0.2511881
```

```
log_acc = (562 + 541) / n
log_acc
```

```
## [1] 0.7488119
```

```
# plots the ROC Curve
tseq = seq(0.001, 0.999, length.out = 100)
sensitivity = c(); specificity = c()
```

```
for (j in 1:length(tseq)){
t = tseq[j]

pred.label[probs>t] = 1
pred.label[probs < t] = 0

p.ind = which(syn_test$stroke == 1)
sensitivity[j] = mean(pred.label[p.ind] == syn_test$stroke[p.ind])

n.ind = which(syn_test$stroke == 0)
specificity[j] = mean(pred.label[n.ind] == syn_test$stroke[n.ind])
}

plot(1 - specificity, sensitivity, type = "l", xlim = c(0, 1), ylim = c(0, 1))
abline(a = 0, b = 1)
```



Linear SVM

```
# performs linear support vector machine
# svm.lin = tune(svm, stroke~., data = syn_train, kernel = "linear",
# ranges = list(cost = c(0.1, 1, 10, 1000)))

# svm.best = svm.lin$best.model

# best model: cost = 1
library(ROCR)

# predicts on the testing set
svm.fit = svm(stroke~., data = num_train, kernel = "linear",
```

```
              cost = 1, probability=TRUE)

pred=predict(svm.fit,num_test[, !names(num_test) %in% c("stroke")], probability=TRUE)

table(prediction = pred, truth = num_test$stroke)  # confusion matrix
```

```
##           truth
## prediction   0   1
##          0 565 180
##          1 201 527
```

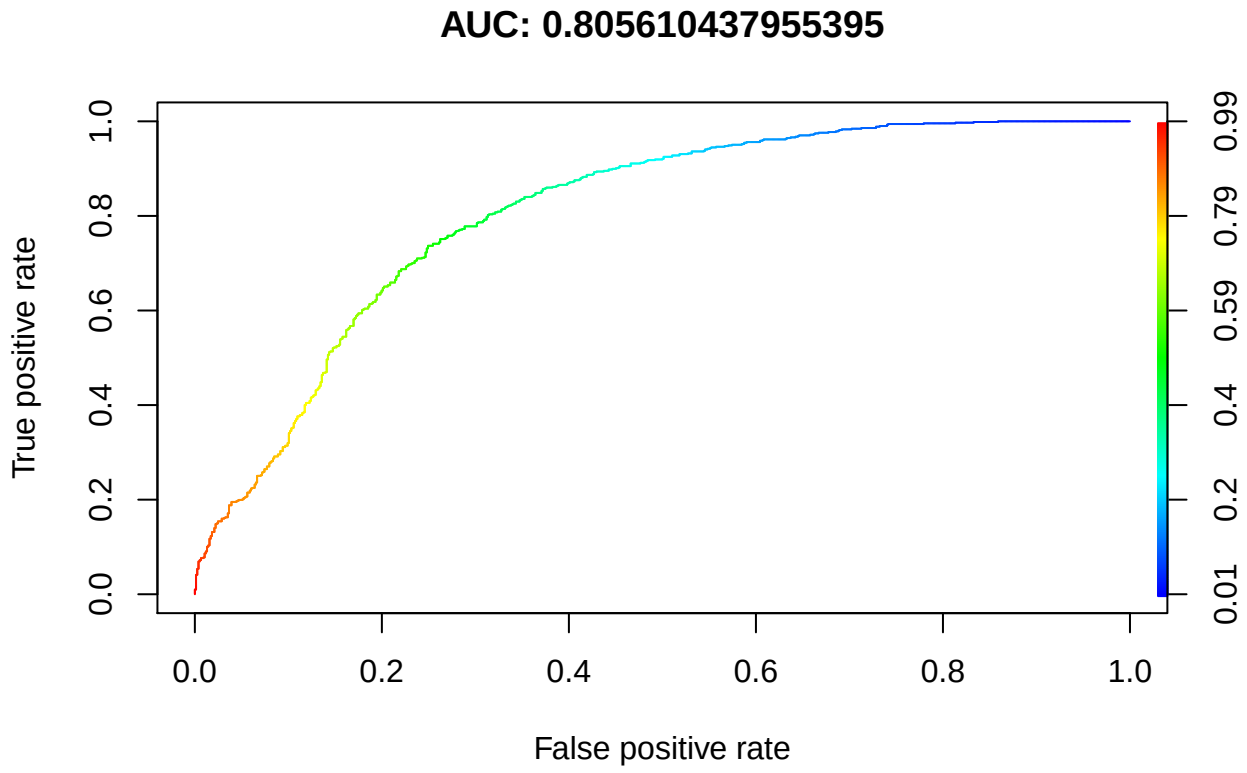
```
svm_error = (201 + 181) / dim(num_test)[1]
svm_error
```

```
## [1] 0.2593347
```

```
svm_acc = (565 + 526) / dim(num_test)[1]
svm_acc
```

```
## [1] 0.7406653
```

```
pred.prob = attr(pred, "probabilities")
pred.to.roc = pred.prob[, 2]

predct.rocr = prediction(as.numeric(pred.to.roc), num_test$stroke )
perf.rocr = performance(predct.rocr, measure = "auc", x.measure = "cutoff")
perf.tpr.rocr = performance(predct.rocr, "tpr","fpr")
plot(perf.tpr.rocr, colorize=T,main=paste("AUC:",(perf.rocr@y.values)))
```



Linear Discriminant Analysis

```
set.seed(123)
lda.fit = lda(stroke~., data = num_train, probability=TRUE) # performs LDA classification

lda.pred = predict(lda.fit, num_test)  # predicts on the test set

table(prediction = lda.pred$class, truth = num_test$stroke)  # confusion matrix
```

```
##           truth
## prediction   0   1
##          0 547 165
##          1 219 542
```

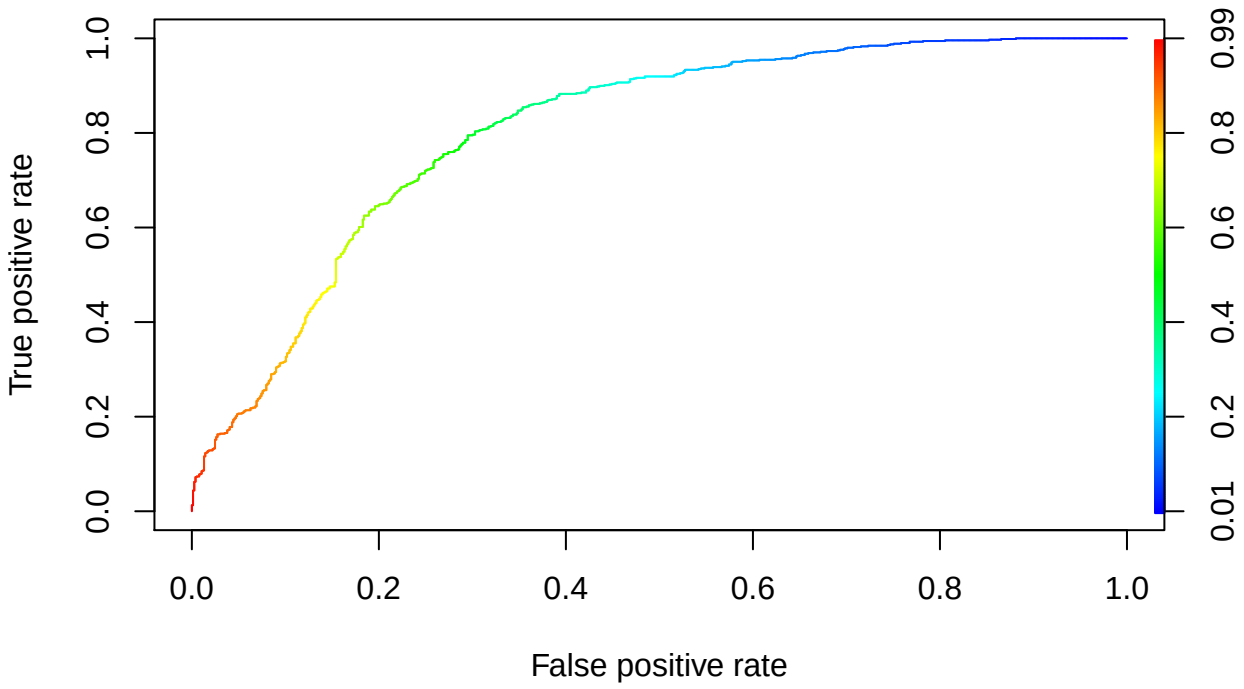
```
lda_error = (219 + 165) / dim(syn_test)[1]
lda_error
```

```
## [1] 0.2606925
```

```
lda_acc = (547 + 542) / dim(syn_test)[1]
lda_acc
```

```
## [1] 0.7393075
```

```
pred = predict(lda.fit,num_test[, !names(num_test) %in% c("stroke")], probability=TRUE)

pred.prob = pred$posterior

pred.to.roc = pred.prob[, 2]

predct.rocr = prediction(as.numeric(pred.to.roc), num_test$stroke )
perf.rocr = performance(predct.rocr, measure = "auc", x.measure = "cutoff")
perf.tpr.rocr = performance(predct.rocr, "tpr","fpr")
plot(perf.tpr.rocr, colorize=T,main=paste("AUC:",(perf.rocr@y.values)))
```



KNN

```
set.seed(1)

knn5 = knn(num_train[, -11], num_test[, -11], syn_train$stroke, k=5)
table(syn_test$stroke, knn5)
```

```
##    knn5
##       0   1
##   0 530 236
##   1 180 527
```

```
TPM = (171+229) / dim(syn_test)[1]
TPM
```

```
## [1] 0.2715547
```

```
knn5_acc = (538+535) / dim(syn_test)[1]
knn5_acc
```

```
## [1] 0.7284453
```

```
knn10 = knn(num_train[, -11], num_test[, -11], syn_train$stroke, k=10)
table(syn_test$stroke, knn10)
```

```
##    knn10
##       0   1
##   0 522 244
##   1 169 538
```

```
TPM = (165+239) / dim(syn_test)[1]
TPM
```

```
## [1] 0.2742702
```

```
knn10_acc = (519+543) / dim(syn_test)[1]
knn10_acc
```

```
## [1] 0.7209776
```

```
knn50 = knn(num_train[, -11], num_test[, -11], syn_train$stroke, k=50)
table(syn_test$stroke, knn50)
```

```
##    knn50
##       0   1
##   0 533 233
##   1 148 559
```

```
TPM = (139+231) / dim(syn_test)[1]
TPM
```

```
## [1] 0.2511881
```

```
knn50_acc = (538+565) / dim(syn_test)[1]
knn50_acc
```

```
## [1] 0.7488119
```

PCA & kmean

```
library(devtools)
```

```
## Loading required package: usethis
```

```
install_github("vqv/ggbiplot")
```

```
## Skipping install of 'ggbiplot' from a github remote, the SHA1 (7325e880) has not changed since last i
##   Use `force = TRUE` to force installation
```

```
require(tidyverse)
```

```
## Loading required package: tidyverse
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.1 --

## v tibble  3.1.4     v dplyr   1.0.7
## v tidyr   1.1.3     v stringr 1.4.0
## v readr   2.0.1     v forcats 0.5.1
## v purrr   0.3.4

## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## x dplyr::select() masks MASS::select()
```

```
require(ggbiplot)
```

```
## Loading required package: ggbiplot

## Loading required package: plyr

## ------------------------------------------------------------------------------

## You have loaded plyr after dplyr - this is likely to cause problems.
## If you need functions from both plyr and dplyr, please load plyr first, then dplyr:
## library(plyr); library(dplyr)

## ------------------------------------------------------------------------------

##
## Attaching package: 'plyr'

## The following objects are masked from 'package:dplyr':
##
##     arrange, count, desc, failwith, id, mutate, rename, summarise,
##     summarize

## The following object is masked from 'package:purrr':
##
##     compact

## Loading required package: scales

##
## Attaching package: 'scales'

## The following object is masked from 'package:purrr':
##
##     discard

## The following object is masked from 'package:readr':
##
##     col_factor

## Loading required package: grid
```

```
require(ggthemes)
set.seed(123)
data = num_train[, !names(syn_train) %in% c("stroke")]
str(data)
```

```
## 'data.frame':    3435 obs. of  10 variables:
##  $ gender           : num  2 2 1 1 2 2 1 2 2 2 ...
##  $ age              : num  0.515 44.085 36.951 28.701 49.877 ...
##  $ hypertension     : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 2 ...
##  $ heart_disease    : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...
##  $ ever_married     : num  1 2 2 1 2 1 1 2 2 2 ...
##  $ work_type        : num  1 5 4 1 5 4 1 5 5 4 ...
##  $ Residence_type   : num  1 2 2 1 1 2 2 1 1 2 ...
##  $ avg_glucose_level: num  77.7 121.1 99.7 111.2 88.1 ...
##  $ bmi              : num  17.8 25.3 31.5 25 34.3 ...
##  $ smoking_status   : num  4 3 4 4 2 1 2 2 3 2 ...
```

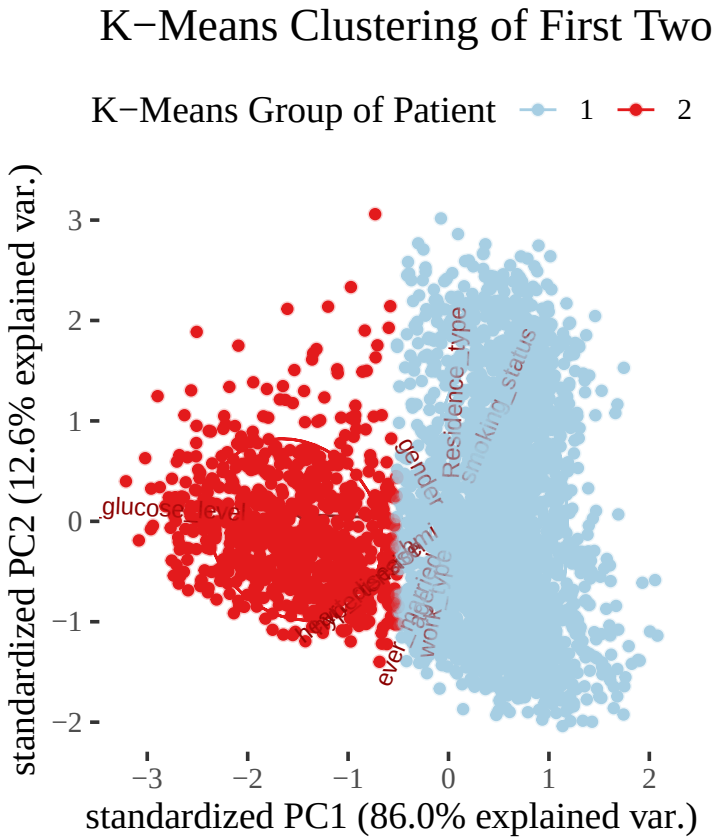
```
data$hypertension = as.numeric(data$hypertension)
data$heart_disease = as.numeric(data$heart_disease)
df.pca = prcomp(data)
pca_2 = df.pca$x %>%
    as.tibble %>%
    select(PC1, PC2)
```

```
## Warning: `as.tibble()` was deprecated in tibble 2.0.0.
## Please use `as_tibble()` instead.
## The signature and semantics have changed, see `?as_tibble`.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_warnings()` to see where this warning was generated.
```

```
pca_kmeans = pca_2 %>%
    kmeans(centers = 2)
pca_kmeans$betweenss/pca_kmeans$totss ##accuracy
```

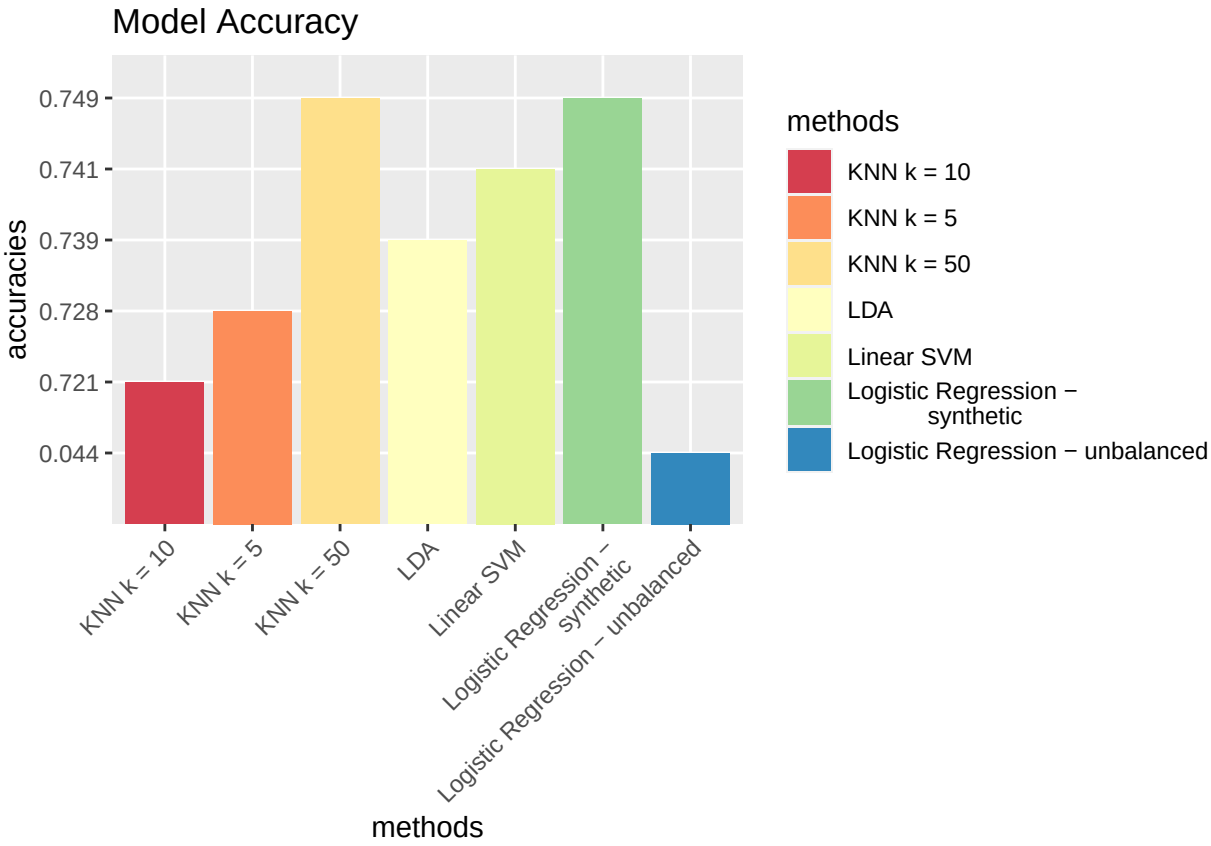
```
## [1] 0.6656663
```

```
ggbiplot(df.pca, groups = factor(pca_kmeans$cluster),
         ellipse = TRUE) +
    theme_tufte(base_size = 14) +
    geom_point(aes(col = factor(pca_kmeans$cluster)),
               size = 2, alpha = 0.2) +
    theme(legend.position = 'top') +
    scale_color_manual(name = 'K-Means Group of Patient',
                       values = c('#a6cee3', '#e31a1c')) +
    ggtitle('K-Means Clustering of First Two Principal Components')
```



Results

```
accuracies = c(round(log1_acc, 3), round(log_acc, 3), round(svm_acc, 3),
               round(lda_acc, 3), round(knn5_acc, 3), round(knn10_acc, 3),
               round(knn50_acc, 3))
methods = c("Logistic Regression - unbalanced", "Logistic Regression -
            synthetic", "Linear SVM", "LDA", "KNN k = 5", "KNN k = 10",
            "KNN k = 50")

accuracy = as.data.frame(cbind(accuracies, methods))

ggplot(data=accuracy, aes(x=methods, y=accuracies)) +
  geom_bar(stat="identity", aes(fill = methods)) +
  scale_x_discrete(guide = guide_axis(angle=45))+ ggtitle("Model Accuracy") +
  scale_fill_brewer(palette = "Spectral")
```



Methods

Rose Data Method- Out of the whole dataset only 4.25% result in a stroke, assigned to the class number "1". As we can see from the data, most of the observations are a 0, meaning no stroke. This also means that the data is highly unbalanced. Logistic Regression will be negatively affected because of the unbalanced data. At first we ran the logistic model without synthetic data, and then with synthetic data. The synthetic data improved the performance of Logistic Regression but did not affect Linear SVM, LDA, or KNN.

Logistic Regression- Similar to Linear Regression, unlike Linear Regression, the response variables can be categorical, continuous data. To predict the group membership, Linear Regression uses the Log Odds ratio. Probabilities uses an interative maximum likelihood method.

Linear SVM- A classification and regression model. It can solve linear and non-linear probelms by creating a line or a hyperplane which then seperates the data into classes.

LDA- Linear classifier that uses Bayesian statistics

KNN- Supervised learning model that is used for both classification and regression. It uses feature similarity to predict the cluster that the new point will be grouped into

PCA + k-Means- Show unsupervised learning model performance

Conclusion

We implemented Logistic Regression, Linear Support Vector Machine, Linear Discriminant Analysis, K-Nearest Neighbors, PCA and k-Means Clustering to find the best predictors to determine is someone is at a higher risk of having a stroke. Logistic Regression with synthetic data had the highest rate of error out of all the models we tested. The Logistic Regression model had a 25.11% error rate. The next model we

implemented was Linear Support Vector Machine with the linear kernel cost=1. This model had a 25.9% error rate. Linear Discriminant Analysis had a 26% error rate. Lastly, we implemented K-Nearest Neighbors on the data. This method gave us a 25% error rate similar to the others. The best models to predict a person's likelihood of having a stroke are K-Nearest Neighbors and Logistic Regression using synthetic data.

Additional Analyses

After discussion, to further this research project it would have been interesting to test more models and compare the accuracy of those classification models to the classification modesl that we tested. We could have implemented decision trees and random forest models.

We could have implemented Dimensionality reduction to reduce the number of prediction features. It would be beneficial if we tried different methods on the dataset such as PCA and k-Means clustering.

To continue exploration with the stroke dataset, it would be beneficial to gather real world data that is balanced between having a stroke and not having a stoke.