PSTAT 131 Final Project: Stroke Prediction - Binary Classification

Sharanya Sharma & Yuval Madne

2023-12-05

```
# Load in Data
stroke_data <- read.csv("data/stroke_data.csv")
head(stroke_data)
```

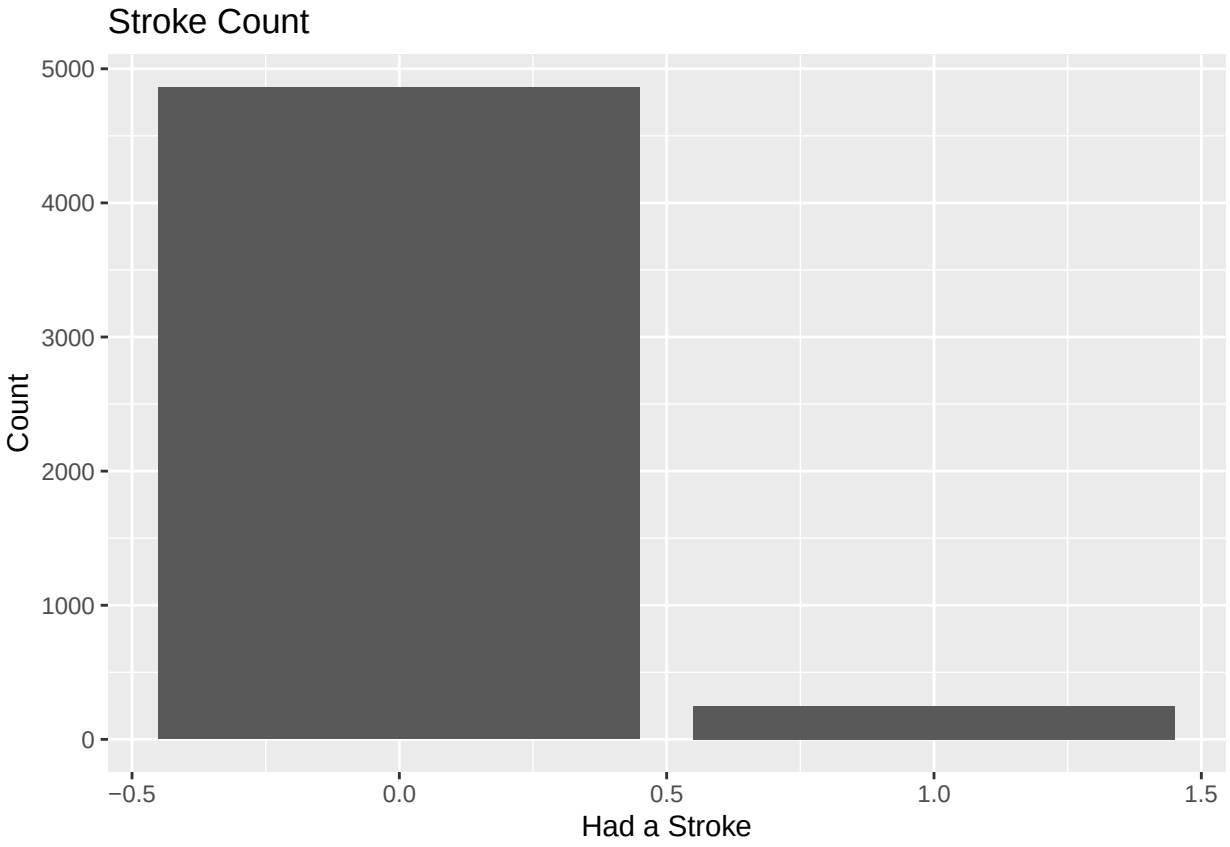
```
##      id gender age hypertension heart_disease ever_married     work_type
## 1  9046   Male  67            0             1          Yes       Private
## 2 51676 Female  61            0             0          Yes Self-employed
## 3 31112   Male  80            0             1          Yes       Private
## 4 60182 Female  49            0             0          Yes       Private
## 5  1665 Female  79            1             0          Yes Self-employed
## 6 56669   Male  81            0             0          Yes       Private
##   Residence_type avg_glucose_level  bmi  smoking_status stroke
## 1          Urban            228.69 36.6 formerly smoked      1
## 2          Rural            202.21  N/A    never smoked      1
## 3          Rural            105.92 32.5    never smoked      1
## 4          Urban            171.23 34.4          smokes      1
## 5          Rural            174.12   24    never smoked      1
## 6          Urban            186.21   29 formerly smoked      1
```

```
# Load in libraries
library(glmnet)
library(dplyr)
library(tidymodels)
library(tidyverse)
library(class)
library(FNN)
library(MASS)
library(readxl)
library(ISLR)
library(tree)
library(maptree)
library(randomForest)
library(gbm)
library(ROCR)
library(boot)
library(gbm)
library(xgboost)
```

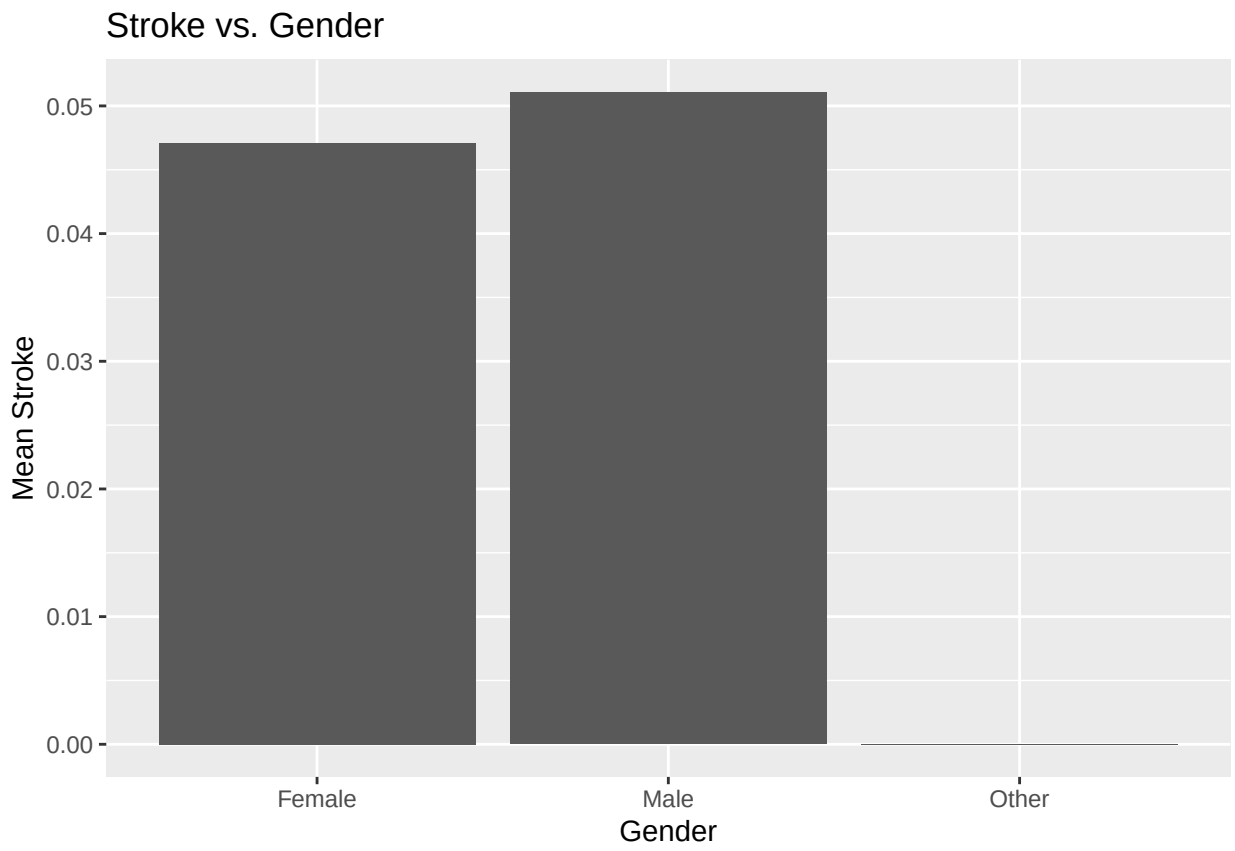
This dataset was acquired from kaggle by fedesoriano, with the intention to educate others about the various factors that can potentially cause a stroke. As the World Health Organization claims strokes to be the 2nd leading cause of death in the world with a global death rate of 11%, prevention has gained increasing

popularity. This data set considers 10 factors that can contribute to whether a person has a stroke or not. These factors include gender, age, hypertension, heart health (diseased or not), marriage status, work-type, residence-type, average glucose level, BMI, and smoking history. Using these factors, various statistical learning algorithms can be applied to predict the likelihood of a stroke. Since stroke is a binary outcome, where people can either have a stroke or not have a stroke, binary classification will be used. Methods to implement binary classification include k-NN, Ridge, Lasso, Trees/Random Forests, Boosting, LDA, and SVM. While each of these methods have their own benefits, we will compare the results of their predictions to see the overall trend of prediction.

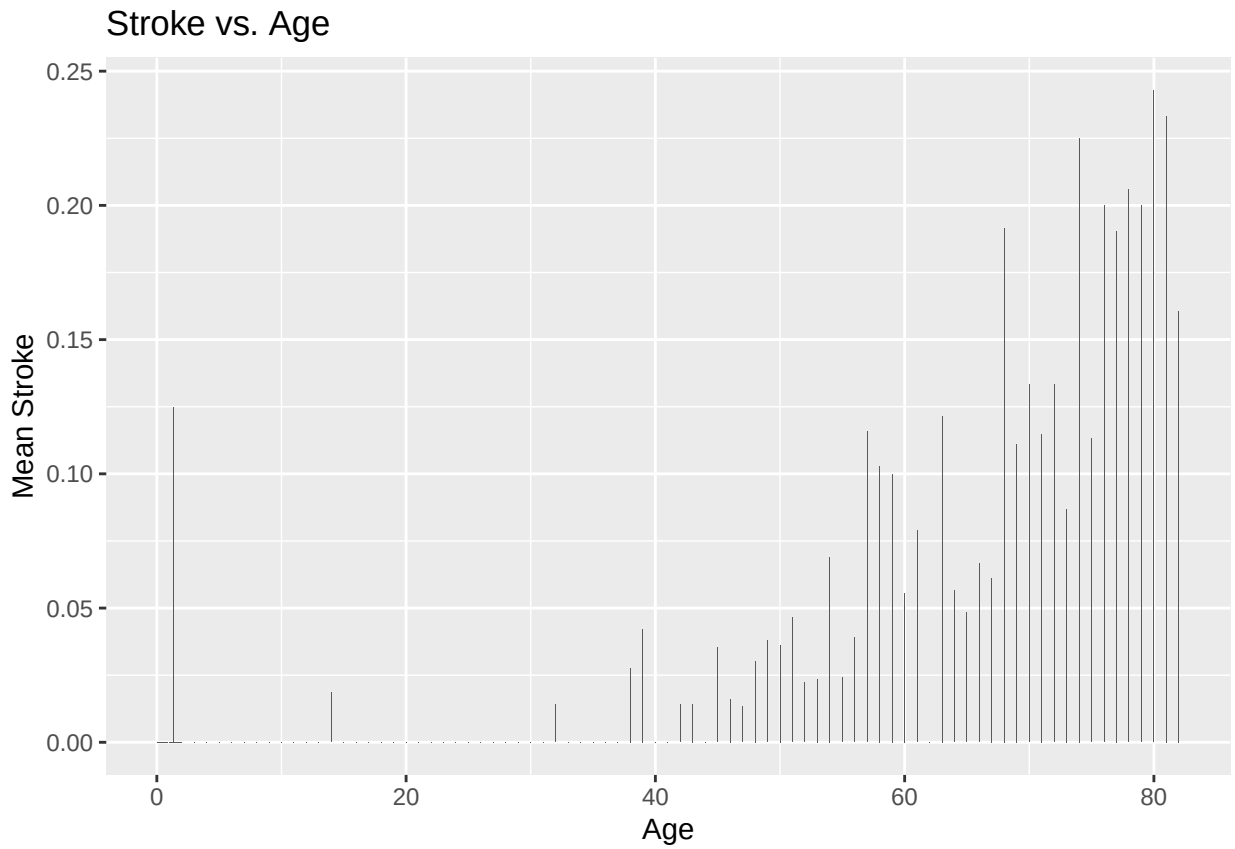
```
ggplot(stroke_data, aes(x=stroke))+ geom_bar(stat = "count")+ labs(title = "Stroke Count", x = "Had a S
```



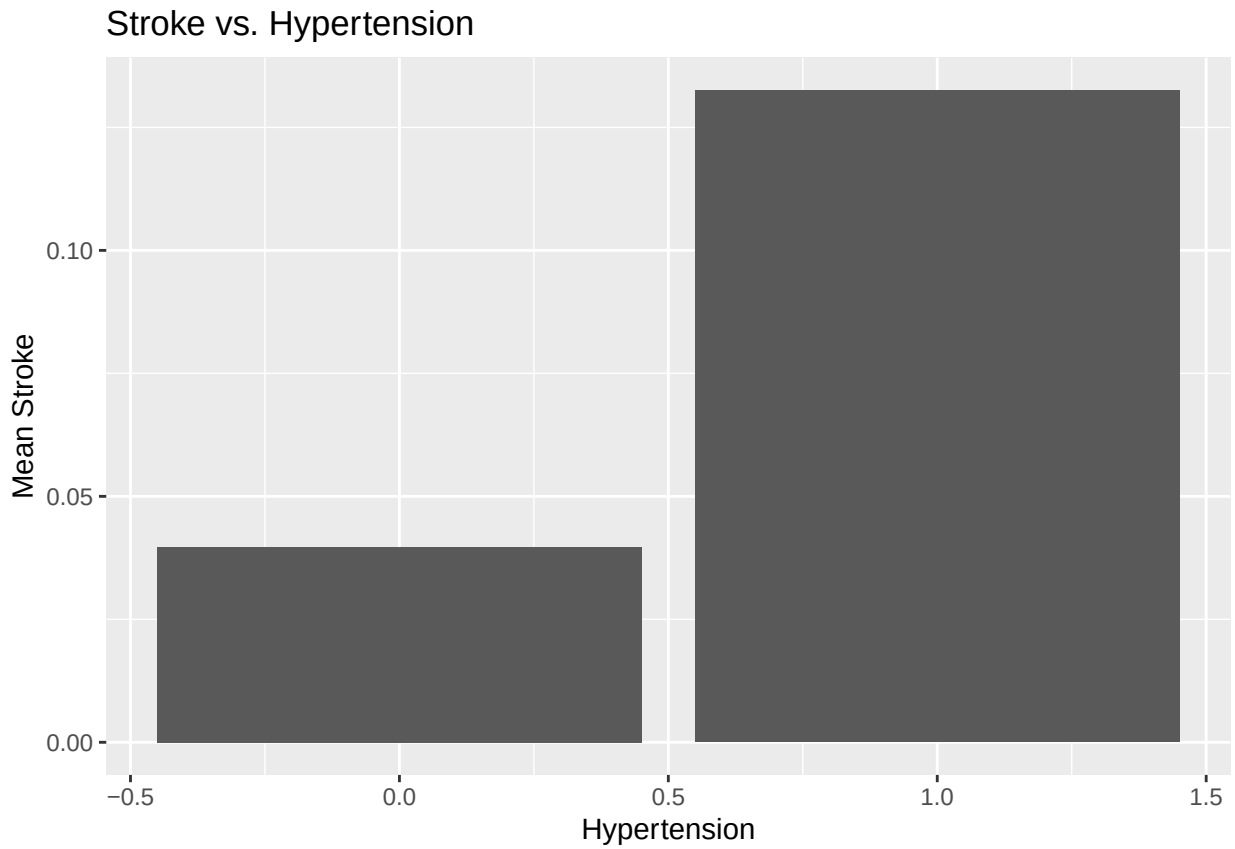

```
ggplot(stroke_data, aes(x=gender,y=stroke))+ geom_bar(stat = "summary", fun = "mean", position = "dodge
```



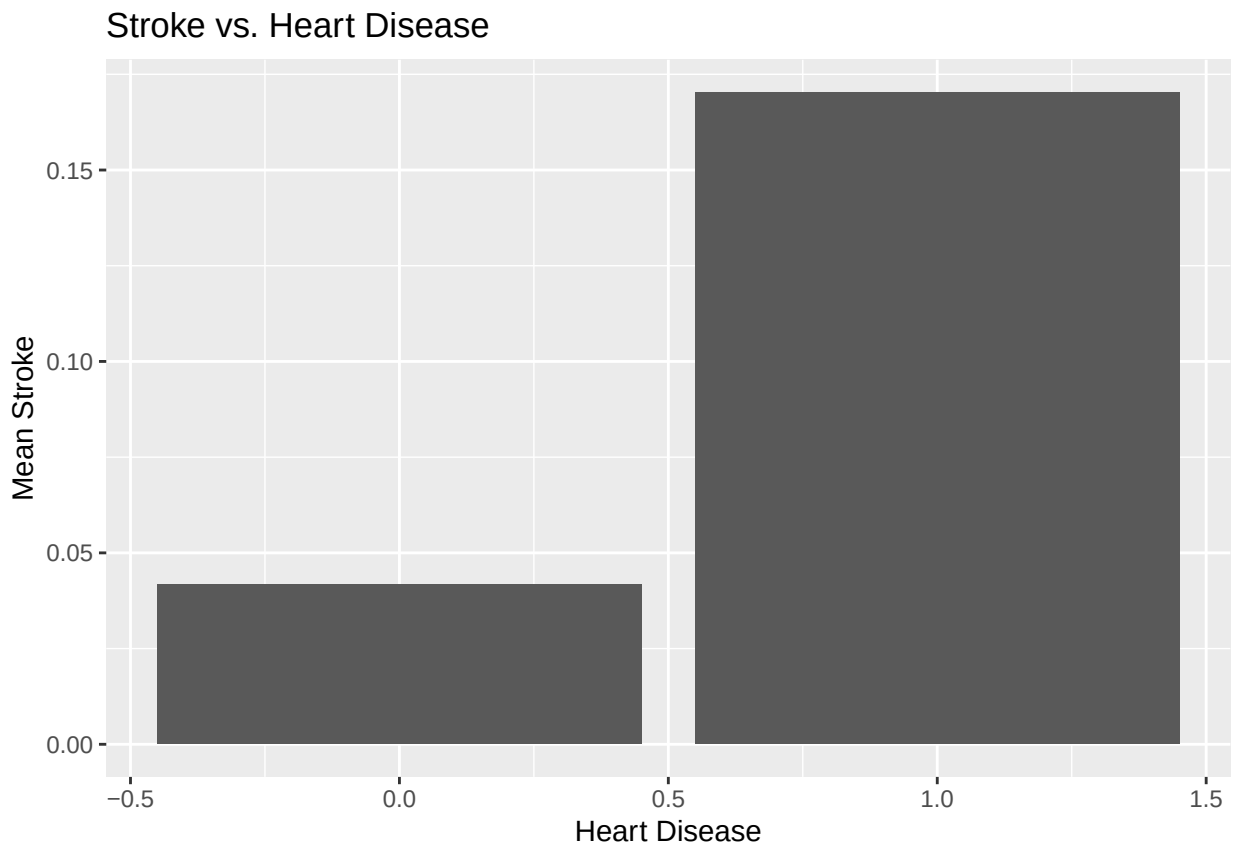

```
ggplot(stroke_data, aes(x=age,y=stroke))+
  geom_bar(stat = "summary", fun = "mean", position = "dodge")+
  labs(title = "Stroke vs. Age",
       x = "Age",
       y = "Mean Stroke")
```



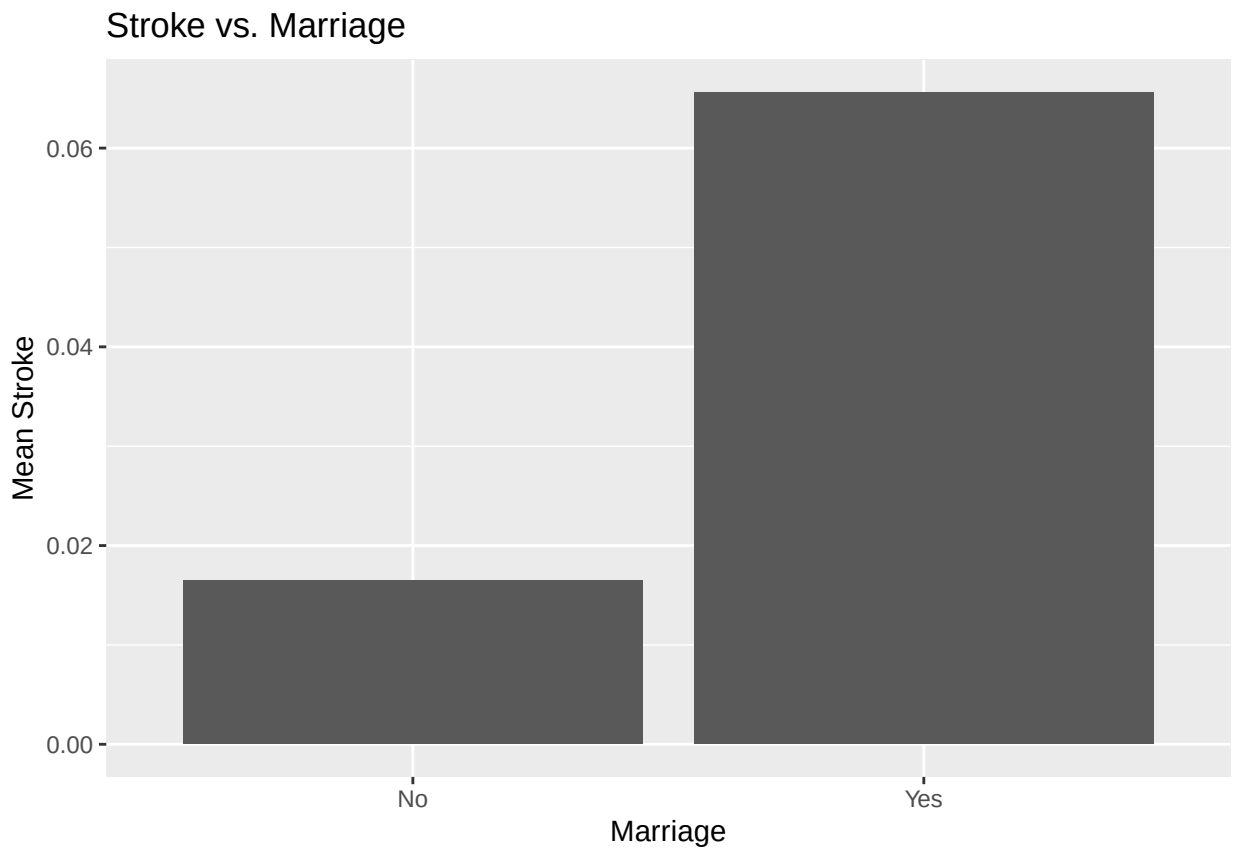

```
ggplot(stroke_data, aes(x=hypertension,y=stroke))+
  geom_bar(stat = "summary", fun = "mean", position = "dodge")+
  labs(title = "Stroke vs. Hypertension",
       x = "Hypertension",
       y = "Mean Stroke")
```



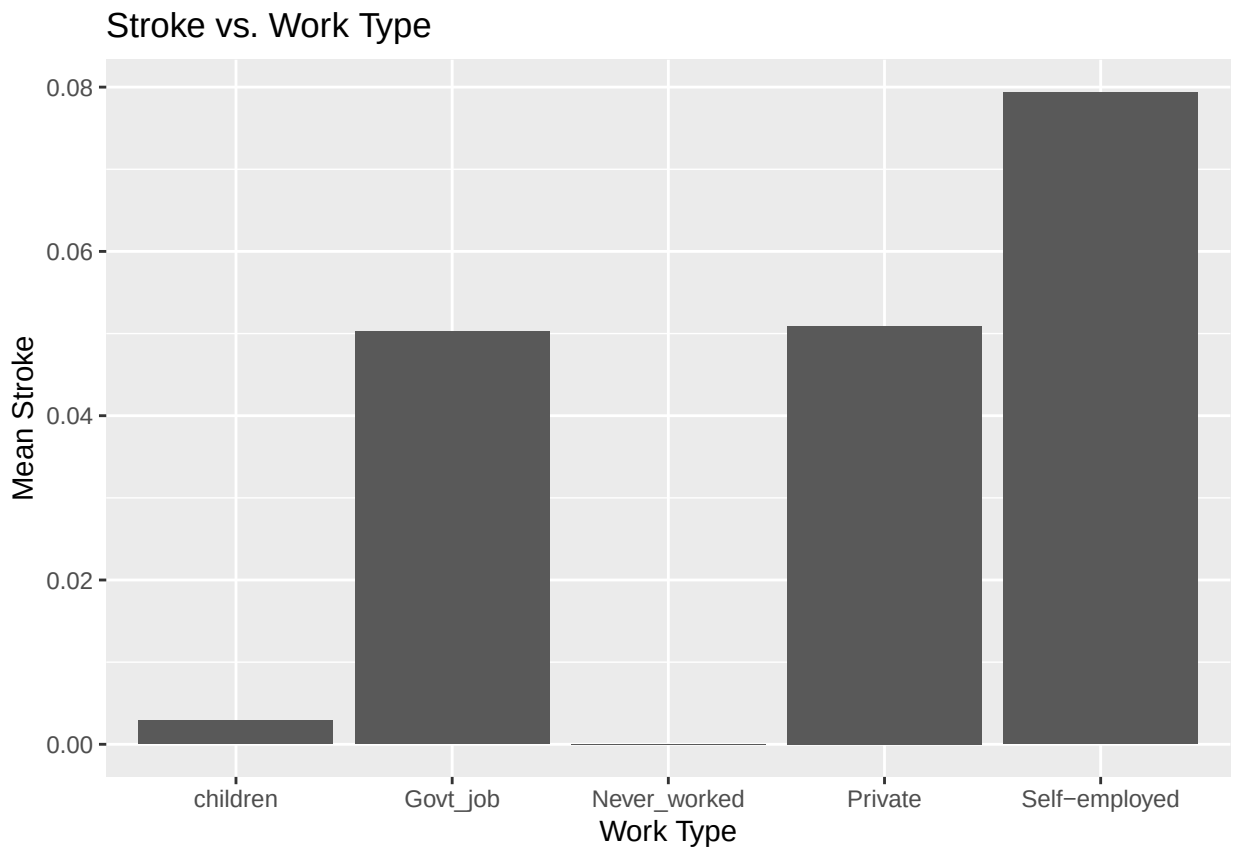

```
ggplot(stroke_data, aes(x=heart_disease,y=stroke))+
  geom_bar(stat = "summary", fun = "mean", position = "dodge")+
  labs(title = "Stroke vs. Heart Disease",
       x = "Heart Disease",
       y = "Mean Stroke")
```



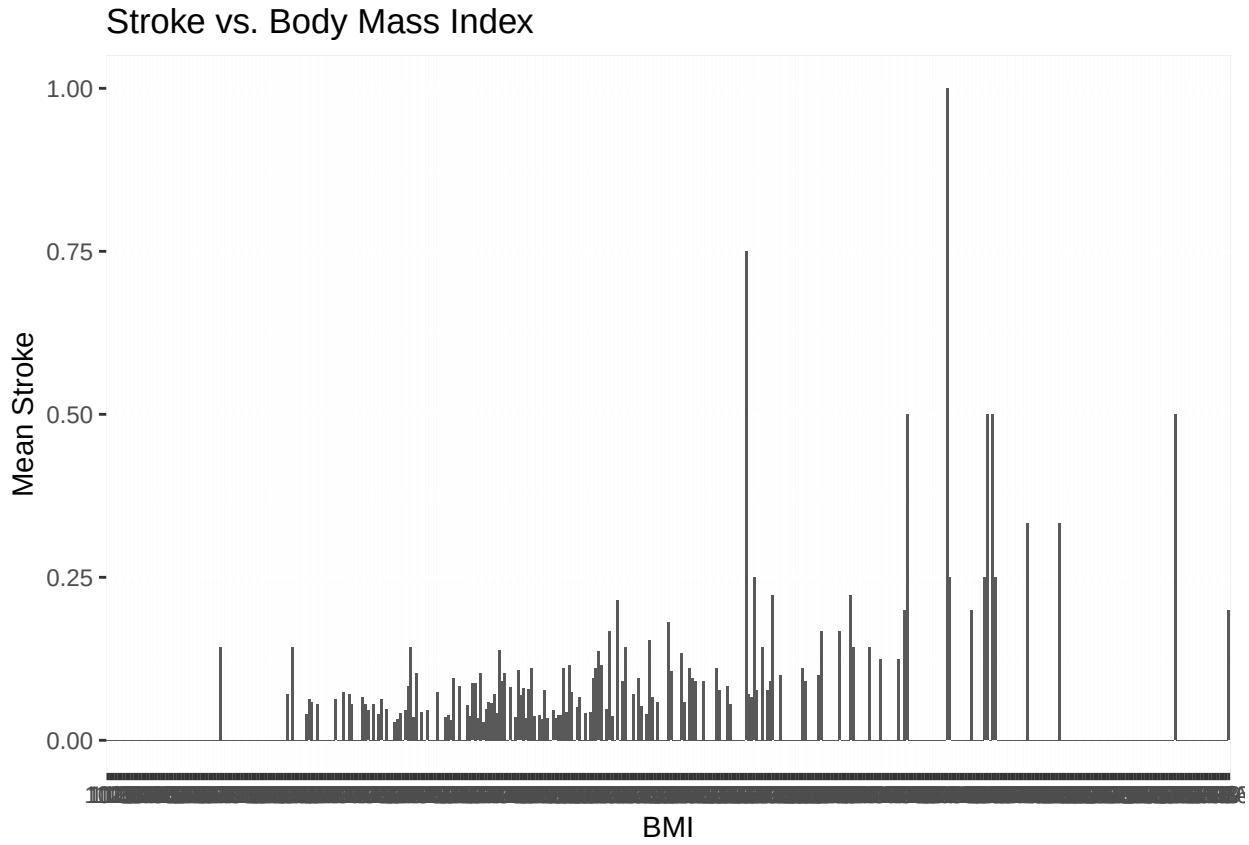

```
ggplot(stroke_data, aes(x=ever_married,y=stroke))+
  geom_bar(stat = "summary", fun = "mean", position = "dodge")+
  labs(title = "Stroke vs. Marriage",
       x = "Marriage",
       y = "Mean Stroke")
```



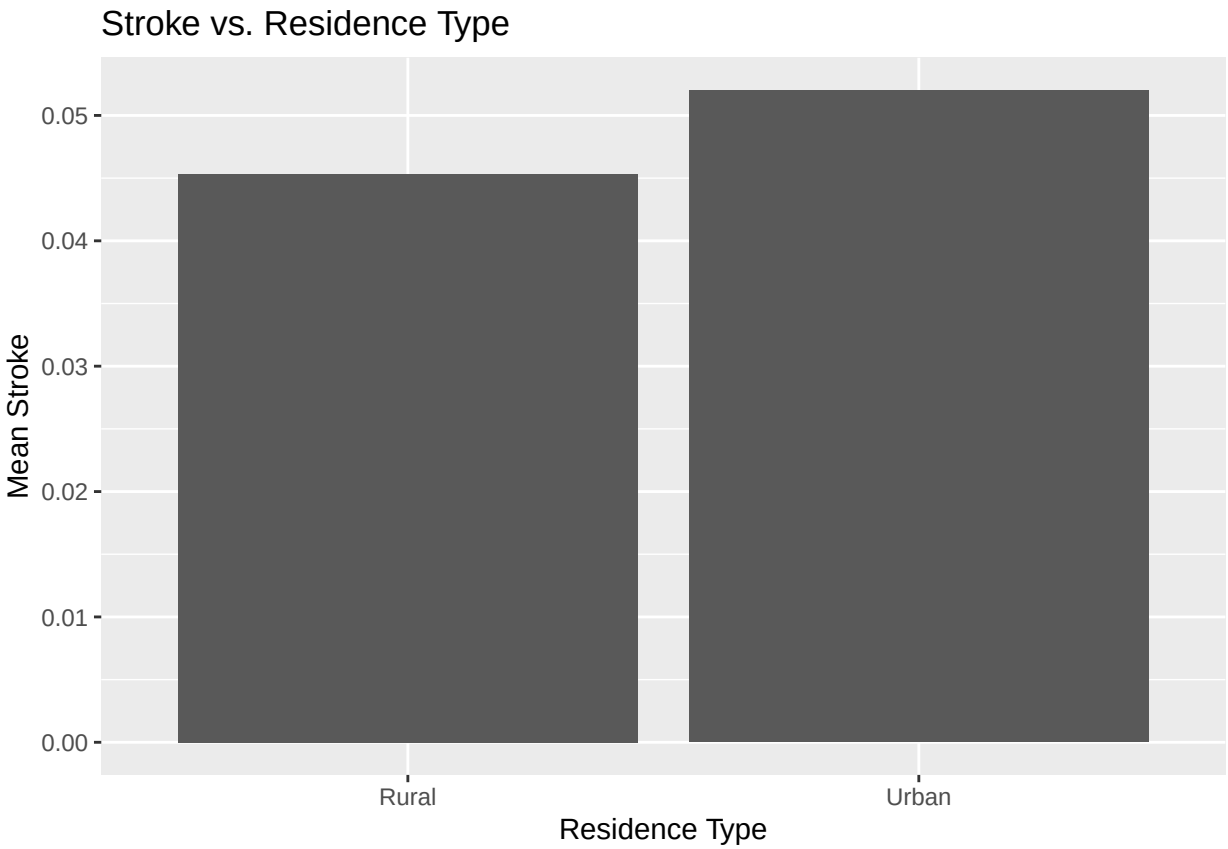

```
ggplot(stroke_data, aes(x=work_type,y=stroke))+
  geom_bar(stat = "summary", fun = "mean", position = "dodge")+
  labs(title = "Stroke vs. Work Type",
       x = "Work Type",
       y = "Mean Stroke")
```



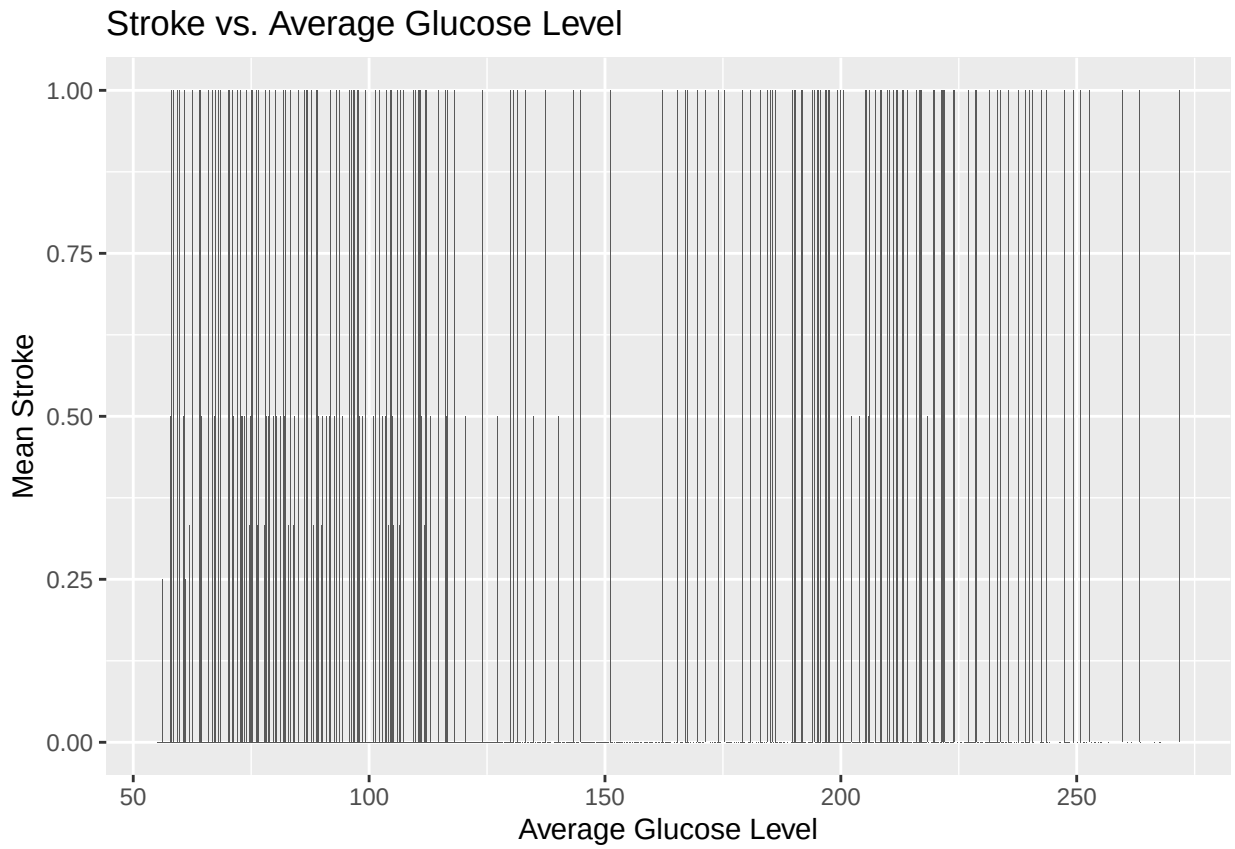

```
ggplot(stroke_data, aes(x=bmi,y=stroke))+
  geom_bar(stat = "summary", fun = "mean", position = "dodge")+
  labs(title = "Stroke vs. Body Mass Index",
       x = "BMI",
       y = "Mean Stroke")
```



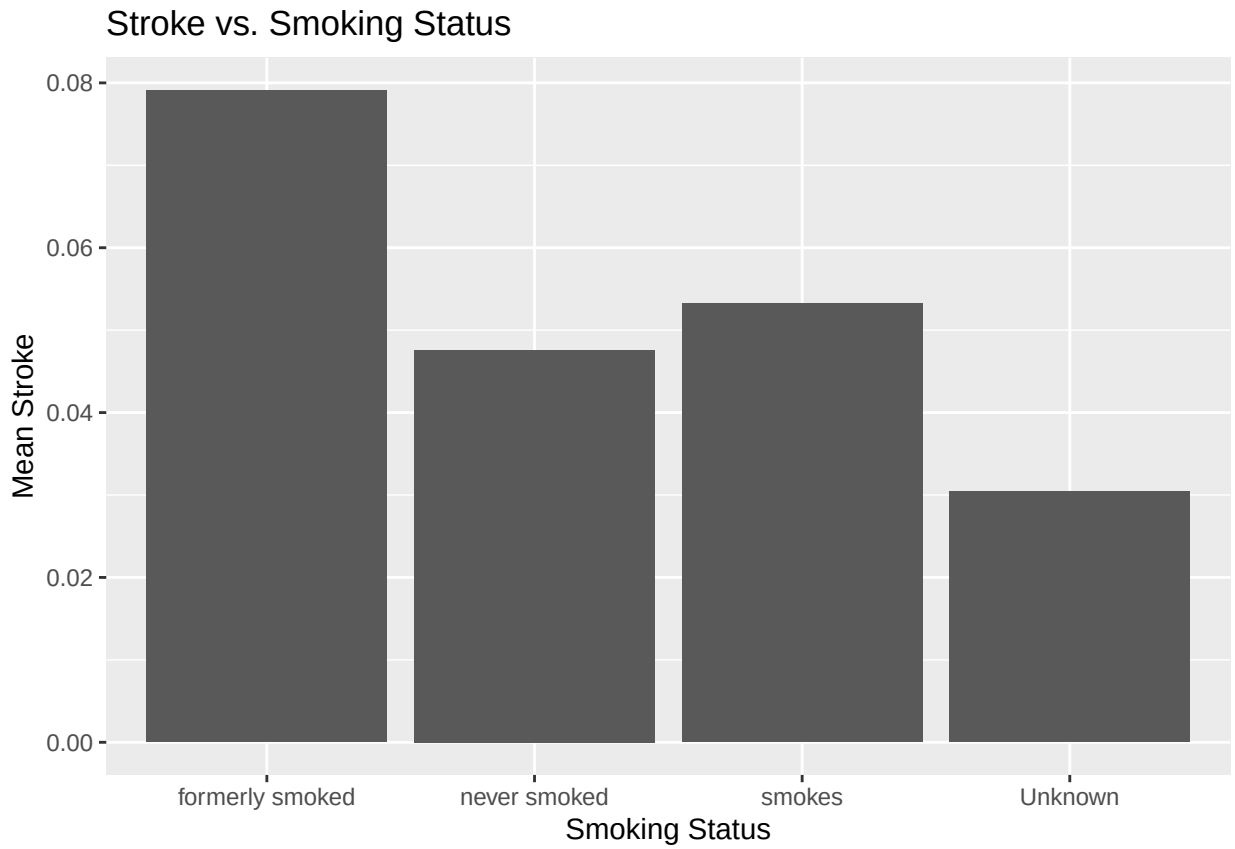

```
ggplot(stroke_data, aes(x=Residence_type,y=stroke))+
  geom_bar(stat = "summary", fun = "mean", position = "dodge")+
  labs(title = "Stroke vs. Residence Type",
       x = "Residence Type",
       y = "Mean Stroke")
```




```
ggplot(stroke_data, aes(x=avg_glucose_level,y=stroke))+
  geom_bar(stat = "summary", fun = "mean", position = "dodge")+
  labs(title = "Stroke vs. Average Glucose Level",
       x = "Average Glucose Level",
       y = "Mean Stroke")
```




```
ggplot(stroke_data, aes(x=smoking_status,y=stroke))+
  geom_bar(stat = "summary", fun = "mean", position = "dodge")+
  labs(title = "Stroke vs. Smoking Status",
       x = "Smoking Status",
       y = "Mean Stroke")
```



Viewing the bar plots above we can see that there is very little data on those who have had a stroke compared to those who have not. Since there is no specification on whether the data was randomly selected, we have to take results with caution because the results can potentially be skewed to show a lower stroke prediction than reality.

Data Cleaning

```
# Remove NA values
stroke_data_cleaned <- stroke_data %>%

  mutate(smoking_status = ifelse(smoking_status == "Unknown", "N/A", smoking_status)) %>%

  mutate(Residence_type = as.factor(ifelse(Residence_type == "Urban", 1, 0))) %>%

  mutate(ever_married = as.factor(ifelse(ever_married == "Yes", 1, 0))) %>%

  mutate(hypertension = as.factor(hypertension)) %>%

  mutate(heart_disease = as.factor(heart_disease)) %>%

  mutate(stroke = as.factor(stroke)) %>%

  mutate(gender = factor(stroke_data$gender, levels = c("Other", "Male", "Female"), labels = c(0, 1, 2))
```

```
  mutate(smoking_status = factor(stroke_data$smoking_status, levels = c("never smoked", "formerly smoke

  mutate(work_type = factor(stroke_data$work_type, levels = c("children", "Govt_job", "Never_worked", "

  mutate(bmi = as.numeric(bmi)) %>%

  filter_all(all_vars(!grepl("N/A", as.character(.)))) %>%

  na.omit(stroke_data)
```

Train/Test Split

```
# assign dummy variables to all factors
response <- stroke_data_cleaned$stroke
stroke_data <- model.matrix(~.-1, data = stroke_data_cleaned, response)

# Splitting index, using 80% for training data
set.seed(1)
split_index <- sample(1:nrow(stroke_data), size = 0.8 * nrow(stroke_data))

# assign values at train/test indeces
train <- stroke_data[split_index,]
test <- stroke_data[-split_index,]

x.train <- train[, -ncol(train)]
y.train <- train[, ncol(train)]

x.test <- test[, -ncol(test)]
y.test <- test[, ncol(test)]
```

Let's first check the prediction accuracy using a non-parametric method, k-NN classification.

k-NN Classification

```
# knn - train the classifier and make predictions on the TRAINING set!
pred.YTtrain = knn(train=x.train, test=x.train, cl=y.train, k=2)

# Confusion matrix on training data to get training error rate
conf.train <- table(predicted=pred.YTtrain, true=y.train)
conf.train
```

```
##           true
## predicted    0    1
##         0 2595  139
##         1    0    6
```

The training confusion matrix shows that out of 2595 people who did not have a stroke, the model predicted all 2595 to not have a stroke. However, out of the 145 people who in fact did have a stroke, the model predicted 139 to not have a stroke. The k-NN model was only able to detect 6 stroke positive cases out of

the 145 people. This implies that the model's True Positive Rate is low, because it is classifying only 4.13% of the positive stroke cases correctly. This can be due to the fact mentioned earlier regarding limited amount of stroke-positive data available for the model to correctly learn which factors contribute to having a stroke. Training predictions typically perform better, now let's apply this model to the test data set to see how well it performs.

```
# Make predictions on test set
pred.YTest <- knn(train = x.train, test = x.test, cl = y.train, k = 2)

# Get confusion matrix
conf.test <- table(pred.YTest, y.test)
conf.test
```

```
##           y.test
## pred.YTest   0   1
##          0 647  34
##          1   4   1
```

Similar to the predictions on the training set, the k-NN model worked best at correctly classifying the no stroke, and again predicted a large percentage to not have a stroke despite them actually having one. Only one observation that actually had a stroke was correctly classified. Despite the model correctly classifying 99% of observations who did not have a stroke, it was only able to correctly classify 2.8% of those that did have a stroke. This again, may be due to the limited amount of stroke-positive data in the original set to work with.

Ridge Regression

```
# find tuning parameter for ridge regression
cvout <- cv.glmnet(x.train, y.train, alpha = 0, family='binomial')
lambda_opt <- cvout$lambda.min

# Ridge Regression
fit <- glmnet(x.train, y.train, family = 'binomial', alpha = 0)
ridge.pred <- predict(fit, s = lambda_opt, newx = x.test, newy = y.test, type = "class")

ridge_coef <- coef(fit, s = lambda_opt)

# Confusion Matrix
conf.ridge <- table(ridge.pred, y.test)
conf.ridge
```

```
##           y.test
## ridge.pred   0   1
##          0 651  35
```

The Ridge regression model use the concept of regularization that apply a shrinkage penalty to each of the beta coefficients, to develop an optimal model. The shrinkage penalty is a combination of a lambda value multiplied by the L2 norm squared. The lambda value is chosen through cross-validation methods, and the fitted regression model is then used to predict the outcome on the testing set. Finally, a confusion matrix is created to evaluate the model's performance. Since the confusion matrix only has 1 row, implying that there were no predictions for positive-stroke. This suggests that the positive-stroke was not well represented by the data.

Lasso Regression

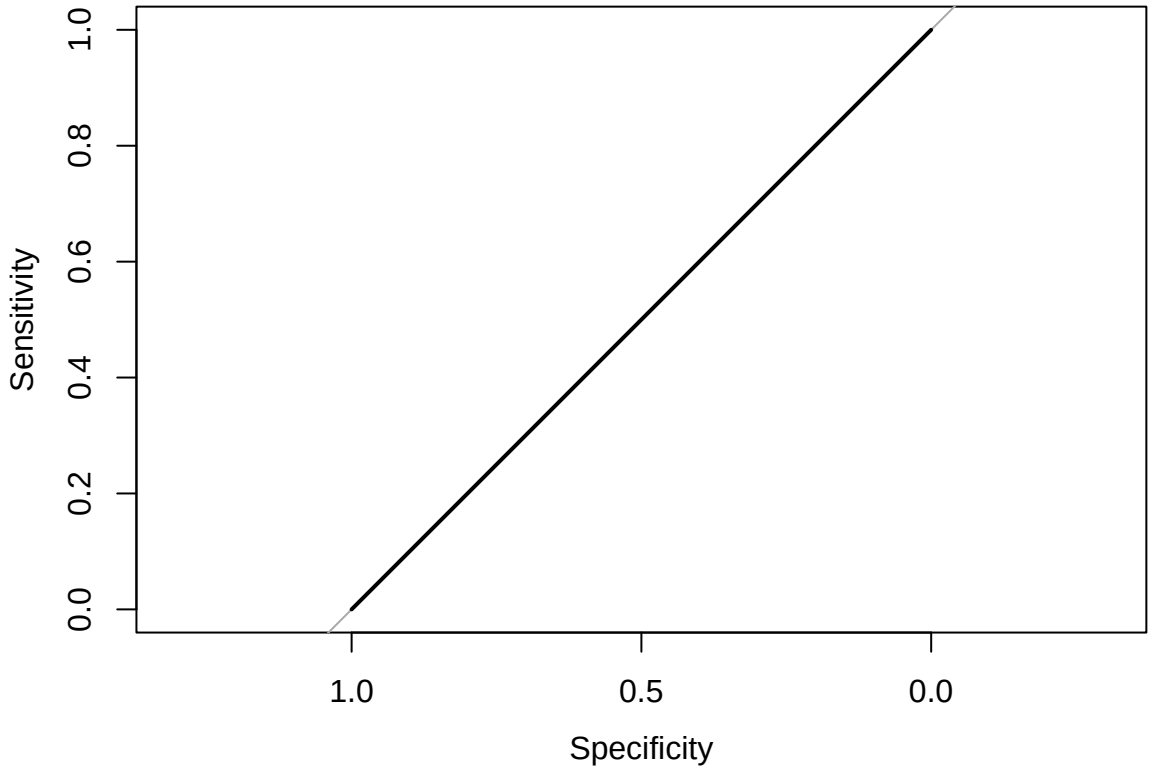
```
# find tuning parameter for ridge regression
cvout <- cv.glmnet(x.train, y.train, alpha = 1, family='binomial')
lambda_opt <- cvout$lambda.min

# Lasso Regression
fit <- glmnet(x.train, y.train, family = 'binomial', alpha = 1)
lasso.pred <- predict(fit, s = lambda_opt, newx = x.test , type = "class")

lasso_coef <- coef(fit, s = lambda_opt)

# Confusion Matrix
conf.lasso <- table(lasso.pred, y.test)
conf.lasso
```

```
##           y.test
## lasso.pred   0   1
##          0 651  35
```

Similar to Ridge regression, Lasso regression is also a regularization technique that uses a shrinkage penalty to modify the beta coefficients and develop an optimal model. This shrinkage penalty is a combination of a chosen lambda value and the L1 norm. Perhaps due to the skewed data, the Ridge and Lasso matrices are leading to the same result.

ROC/AUC Curve

```
library(pROC)
roc <- roc(y.test, as.numeric(ridge.pred))

plot(roc)
```




```
auc(roc)
```

```
## Area under the curve: 0.5
```

We can see that even the ROC curve was impacted, resulting in a strange straight line.

The AUC value of 0.5 essentially means the prediction is equivalent to random selection.

Let's take a different approach using ensemble learning techniques.

Train/Test Split for Decision Tree and Random Forest

```
stroke_data <- read.csv("data/stroke_data.csv")

# Splitting index, using 80% for training data
set.seed(4)
split_index <- sample(1:nrow(stroke_data_cleaned), size = 0.8 * nrow(stroke_data_cleaned))

train <- stroke_data_cleaned[split_index,]
test <- stroke_data_cleaned[-split_index,]

x.train <- train[, -ncol(train)]
y.train <- train[, ncol(train)]
```

```r
x.test <- test[, -ncol(test)]
y.test <- test[, ncol(test)]
```

Decision Tree

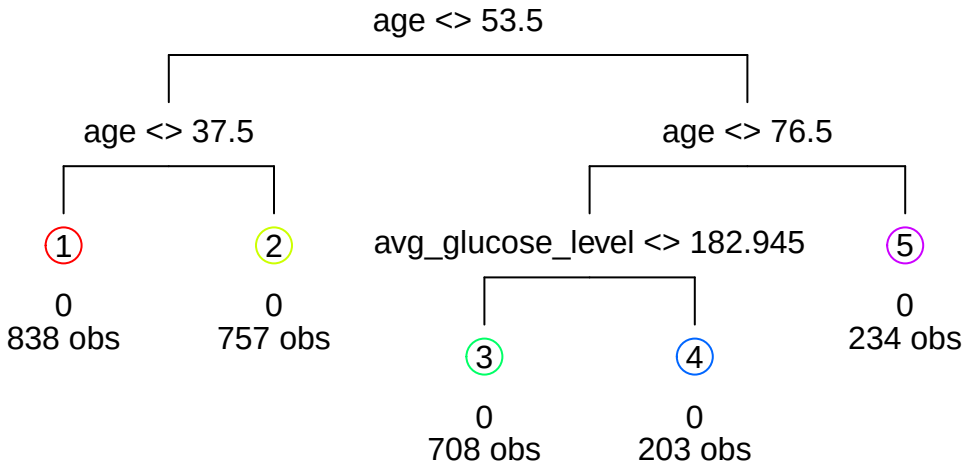
```r
# Creates one tree
stroke_tree <- tree(stroke ~ ., data = train)
print(stroke_tree)
```

```
## node), split, n, deviance, yval, (yprob)
##       * denotes terminal node
##
##  1) root 2740 1105.0 0 ( 0.94891 0.05109 )
##    2) age < 53.5 1595  197.2 0 ( 0.98871 0.01129 )
##      4) age < 37.5 838    0.0 0 ( 1.00000 0.00000 ) *
##      5) age > 37.5 757  170.2 0 ( 0.97622 0.02378 ) *
##    3) age > 53.5 1145  776.9 0 ( 0.89345 0.10655 )
##      6) age < 76.5 911  493.7 0 ( 0.92316 0.07684 )
##       12) avg_glucose_level < 182.945 708  313.2 0 ( 0.94209 0.05791 ) *
##       13) avg_glucose_level > 182.945 203  166.5 0 ( 0.85714 0.14286 ) *
##      7) age > 76.5 234  247.9 0 ( 0.77778 0.22222 ) *
```

```r
summary(stroke_tree)
```

```
##
## Classification tree:
## tree(formula = stroke ~ ., data = train)
## Variables actually used in tree construction:
## [1] "age"               "avg_glucose_level"
## Number of terminal nodes:  5
## Residual mean deviance:  0.3283 = 897.8 / 2735
## Misclassification error rate: 0.05109 = 140 / 2740
```

```r
# five fold cross validation tree and shows the best size for the tree (6)
stroke_tree_cv <- cv.tree(stroke_tree, FUN = prune.misclass, K = 5)
best_size <- stroke_tree_cv$size[which.min(stroke_tree_cv$dev)]

#pruned tree
pruned_stroke_tree <- prune.misclass(stroke_tree, best = best_size)
draw.tree(pruned_stroke_tree, nodeinfo = F)
```




```
#prediction
stroke_tree_predvals <- predict(pruned_stroke_tree, newdata = test, type = "class")
strokeTreeConfusionMat <- table(test$stroke, stroke_tree_predvals)
strokeTreeConfusionMat
```

```
##    stroke_tree_predvals
##       0   1
##   0 646   0
##   1  40   0
```

We started by creating a single decision tree, printing it, summarizing it, and observing it. You can see that the variables used to build the tree include age, average blood sugar level, and smoking status. We can hypothesize that these three variables best predict whether a person will have a stroke. The tree was then pruned and represented visually using the draw.tree function. The root node splits the data at 48.5 years, showing that age is the most important predictor of stroke in the dataset. Individuals younger than 48.5 years were classified into the non-stroke category. For people over 48.5 years of age, the decision tree then diverges at 67.5 years of age, indicating that risk increases with age. In addition, the average blood glucose level for individuals aged 48.5 to 67.5 years further differentiated risk, with a breakdown of 102.13.The subgroup with mean blood glucose level less than or equal to 102.13 was further subdivided based on smoking status. Notably, this model found no strokes in individuals younger than her 48.5 years, suggesting that stroke risk is low for this age group in our sample. In older adults, stroke was not predicted in those with average blood sugar levels above 126.8, but this may be due to potential imbalance in the data. This can also be seen in the confusion matrix, as the model also achieves the most true negative values, in addition to 35 false negative values.

Random Forest

```
# Creates Random Forest
stroke_randomForest <- randomForest(stroke ~ ., data = train, importance = TRUE)
print(stroke_randomForest)
```

```
##
## Call:
##  randomForest(formula = stroke ~ ., data = train, importance = TRUE)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 3
##
##         OOB estimate of  error rate: 5.26%
## Confusion matrix:
##      0 1 class.error
## 0 2596 4 0.001538462
## 1  140 0 1.000000000
```

```
importance(stroke_randomForest)
```

```
##                           0          1 MeanDecreaseAccuracy MeanDecreaseGini
## id                -1.2274884 -0.1226940           -1.2103733        50.965742
## gender             0.4243524 -0.6550392            0.1165018         6.372579
## age               18.2459806 23.9755682           23.4728700        49.613438
## hypertension       1.5788697  2.4164088            2.2603605         5.451261
## heart_disease     -0.4055853  9.5040118            3.3363136         5.844463
## ever_married      16.0460495 -5.2219287           14.5033693         4.517881
## work_type          6.5182096 -3.4300936            5.1262723         9.929694
## Residence_type     0.6807150 -1.1005913            0.2922909         6.501883
## avg_glucose_level -4.8134270  6.3181930           -2.3699341        55.953719
## bmi                3.9770506 -7.0623812            1.4211396        47.108331
## smoking_status    -0.4539264  0.4607556           -0.2786994        11.075062
```

We analyzed the data through a random forest model with 500 trees. The out-of-the-box (OOB) estimate of the model's error rate is 5.36%, indicating a high level of accuracy. The confusion matrix provided additional observations with 2592 true negatives, 3 false positives, 144 false negatives, and 1 true positive. This sample showed high particularity, as the number of true negatives was higher compared to false positives. However, sensitivity is a problem due to the large number of negative effects. The model's ability to detect actual stroke events is critical and requires improvement.

Train/Test Split:

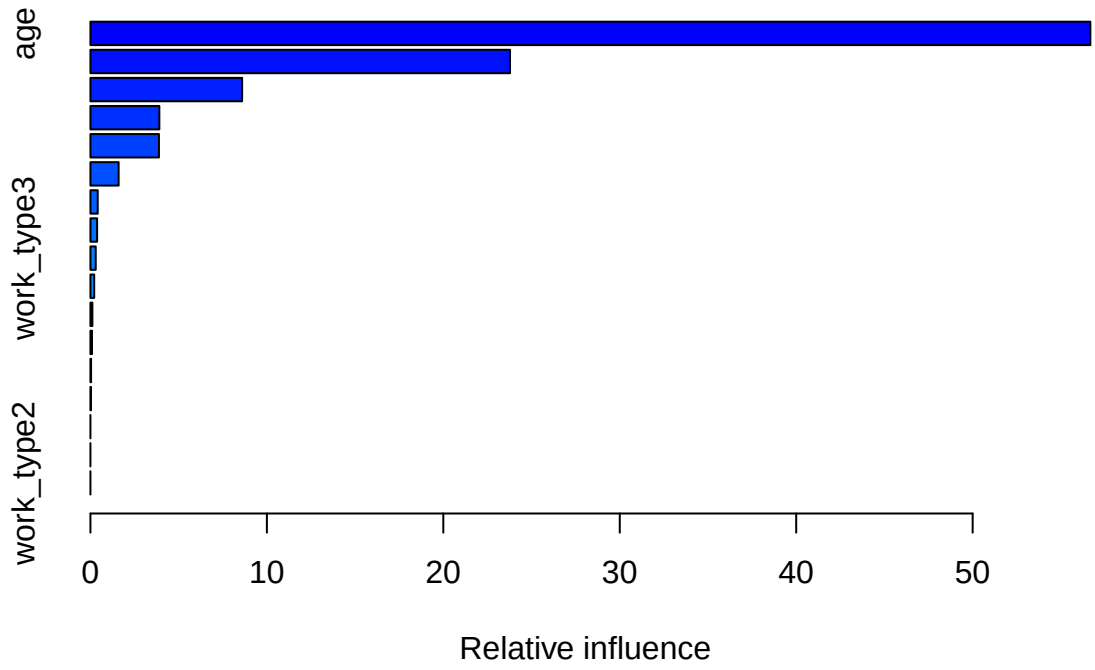
```
# assign dummy variables to all factors
response <- stroke_data_cleaned$stroke
stroke_data <- model.matrix(~.-1, data = stroke_data_cleaned, response)


# Splitting index, using 80% for training data
```

```r
set.seed(5)
split_index <- sample(1:nrow(stroke_data), size = 0.8 * nrow(stroke_data))

# assign values at train/test indices
train <- stroke_data[split_index,]
test <- stroke_data[-split_index,]

x.train <- train[, -ncol(train)]
y.train <- train[, ncol(train)]

x.test <- test[, -ncol(test)]
y.test <- test[, ncol(test)]
```

Now let us visualize a boosting algorithm.

#GBM

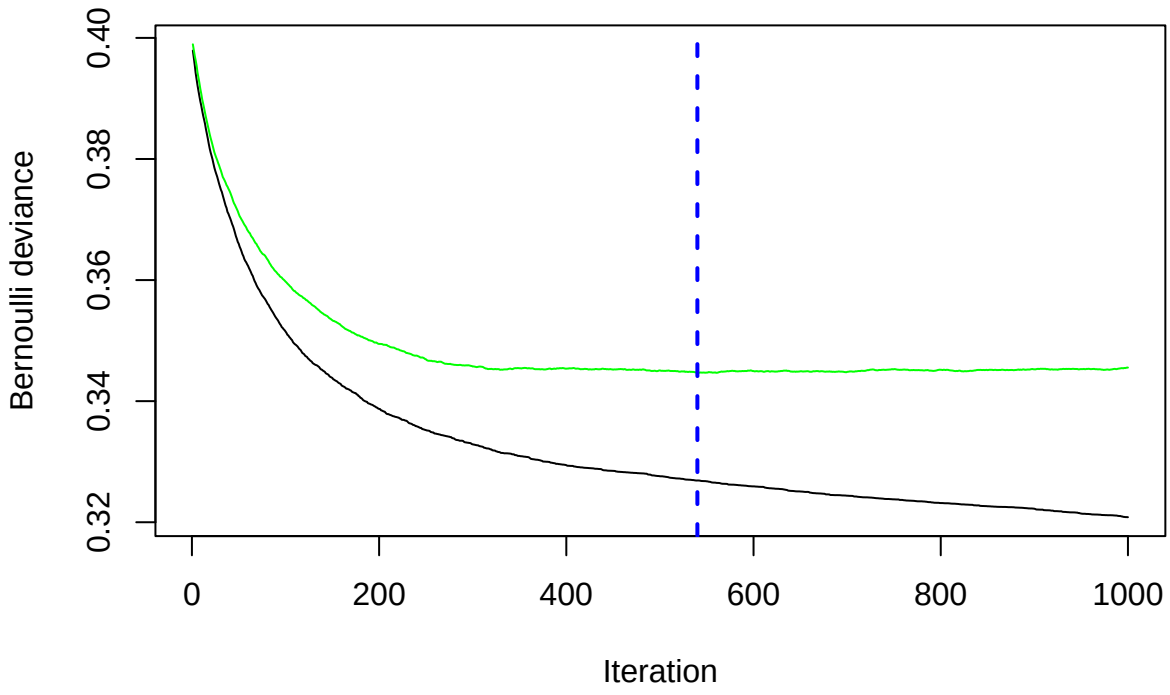
```r
set.seed(6)

# Generate Boosting Model with 5 cv folds
stroke_gbm <- gbm(stroke1 ~ ., data = as.data.frame(train), distribution = "bernoulli", n.trees = 1000,
summary(stroke_gbm)
```




```
##                              var     rel.inf
## age                          age 56.67145133
```

```
## avg_glucose_level avg_glucose_level 23.77983121
## heart_disease1       heart_disease1  8.59610772
## hypertension1         hypertension1  3.90475553
## id                               id  3.88255941
## bmi                             bmi  1.59693689
## work_type4               work_type4  0.41215766
## smoking_status1     smoking_status1  0.37766598
## work_type3               work_type3  0.29999867
## smoking_status2     smoking_status2  0.21342163
## Residence_type1     Residence_type1  0.11020011
## ever_married1         ever_married1  0.08709599
## work_type1               work_type1  0.03663424
## gender1                     gender1  0.03118362
## gender0                     gender0  0.00000000
## gender2                     gender2  0.00000000
## work_type2               work_type2  0.00000000
```

```
best_trees <- gbm.perf(stroke_gbm, method = "cv")
```




```
stroke_gbm_predvals <- predict(stroke_gbm, newdata = as.data.frame(test), n.trees = best_trees, type =
stroke_gbm_predvals <- ifelse(stroke_gbm_predvals > 0.5, 1, 0)
gbmConfusionMat <- table(y.test, stroke_gbm_predvals)
print(gbmConfusionMat)
```

```
##       stroke_gbm_predvals
```

```
## y.test   0
##      0 644
##      1  42
```

The GBM model was trained on the dataset with multiple projections using 1000 trees and a shrinkage of 0.01. The model was validated using five-fold cross-validation to mitigate overfitting and ensure generalizability. The variable importance plot and tables show which predictors are associated with stroke prediction. Age was the most significant predictor and influencer of the relationship, reinforcing the known link between age and stroke. Average blood sugar level is the second most important factor, indicating that patients with high blood sugar have a higher risk of stroke. This pattern was consistent throughout the data analysis. Similarly, the confusion matrix shows a significant true negative rate, indicating skewed data.

Conclusion and future direction for improvement

After visualizing all models, it appears the k-NN classification had the best results. There were 6 true positive classification of strokes, while classifying 99% of non-stroke observations correctly. The boosting algorithm was next best and classified 1 true positive. With this given data, k-NN worked best, which is surprising as stronger regularization methods such as Ridge and Lasso regression didn't work as well. For improvement in the future, we can try to modify the data in a way that there is an equal split of stroke and non-stroke data, to ensure that the models can learn and perform well.