Jason Hou STA 363 Final Project

Jason Hou

2022-12-09

Contents

List of Figures

List of Tables

Abstract

Stroke, defined as the brain disease that happens when the blood supply to part of the brain was blocked or when a blood vessel in the brain breaks, is a leading cause of death and a major cause of severe disability among adults in the United States.According to CDC, more than 795000 people suffer from stroke every years in the United States. While being regarded as a factor that leads to serious consequences and even mortality, stroke is preventable and treatable(CDC).Therefore, it is important to conduct screening on the population in order to identify people who have higher risk of having stroke, and provide preventive support on these people. The primary goal of this project is to build a classifier for medical professionals to conduct screening for risk of stroke. Three techniques, the k-Nearest Neighbors, penalized regression, and forest were implemented to build this classifier.

Section 1: Data and Motivation

Our goal of this project is to build a classifier that helps to predict whether someone is more likely or less likely to have stroke based on the health condition, health measures and related information provided. This classifier can be utilized to assist the prevention of stroke event along with other testing techniques, and facilitate the diagnosis process through providing a preliminary screening, which can help medical professionals to identify people who are more likely to suffer stroke, and implement preventive care if needed.

The data that we are dealing on is a data consist of 5110 rows and 12 columns.The data set record the 12 features(4 numeric features and 8 categorical features) of 5110 patients who have either had suffered or have not suffered from stroke. We have the information on the patient's biological gender, the age of each patient in years,whether a patient has heart disease/hypertension, marital status, smoking status, stroke status and other related information. For other features not mentioned, please refer to the following link for a detailed description of features in this data set.(Fedesoriano) Link: https://www.kaggle.com/datasets/fedesoriano/stroke-prediction-dataset

Section 2: Data Cleaning and EDA

```
originalStroke<-originalStroke[,-1]
```

We started with converting the data types of several features to more appropriate type. There are three features that need to be converted from numeric data to categorical data, which are hypertension, heart_disease, and stroke. These three features were converted from numeric type to categorical type. Other categorical features that are originally in character type were also converted into factor type for the model building later.

```
originalStroke$bmi<-as.numeric(originalStroke$bmi)
```

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

```
originalStroke$stroke<-as.factor(originalStroke$stroke)
originalStroke$hypertension<-as.factor(originalStroke$hypertension)
originalStroke$heart_disease<-as.factor(originalStroke$heart_disease)

#originalStroke$stroke<-as.factor(originalStroke$stroke)
```

```
#Change all categorical data that are in character type to factor type
originalStroke <- as.data.frame(unclass(originalStroke),stringsAsFactors=TRUE)
```

We then check for NA's in the data set. The result was shown in the table below.

```
# Check missing data
knitr::kable(sum(is.na(originalStroke))
,caption = "Number of Missing Values")
```

Table 1: Number of Missing Values

x
201

```
NA_plot<-vis_miss(originalStroke)
```

```
## Warning: `gather_()` was deprecated in tidyr 1.2.0.
## i Please use `gather()` instead.
## i The deprecated feature was likely used in the visdat package.
##   Please report the issue at <https://github.com/ropensci/visdat/issues>.
```

```
NA_plot
```

```
bmi_na_row<-which(is.na(originalStroke$bmi))
```

```
knitr::kable(table(originalStroke$stroke[bmi_na_row]),caption = "Stroke Status of Rows with Missing Valu
```

Table 2: Stroke Status of Rows with Missing Value

Var1	Freq
0	161
1	40

Table 1 and Figure 1 show that 201 rows of missing values were found from the data set, which were only present in the column "bmi". According to Table 2.2, there are 161 patients who suffered from stroke and 40 patients who did not suffer from stroke in patients with their BMI value missing(Here, 0 = the patient had stroke, 1 = the patient have not had stroke). We decide to remove these rows with missing values in BMI.

```
## Remove NAs
stroke<-na.omit(originalStroke)
```

```
#View Classese of stroke
stroke_classes<-ggplot(data = originalStroke, mapping = aes(x=stroke,fill=stroke))+
  geom_bar()+
  xlab("Stroke Status")

stroke_classes
```

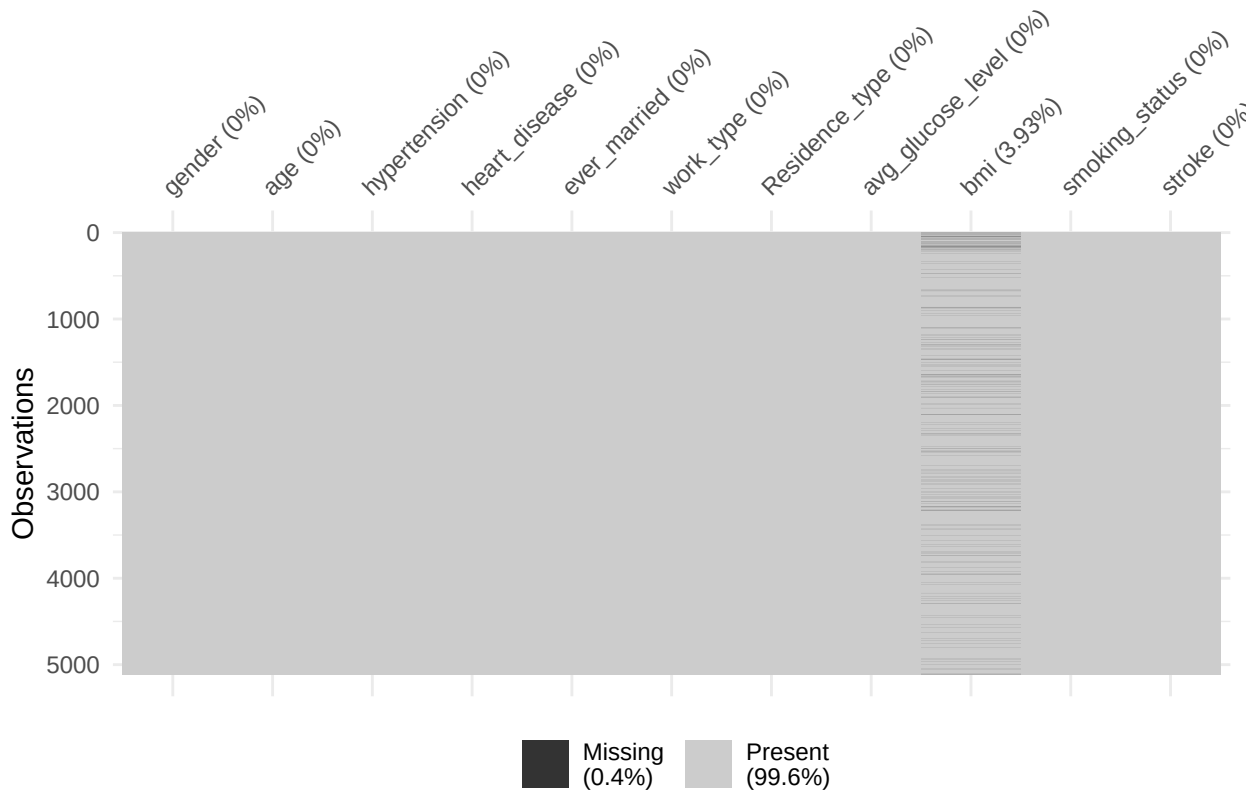


Figure 1: Graph of NA in Each Column

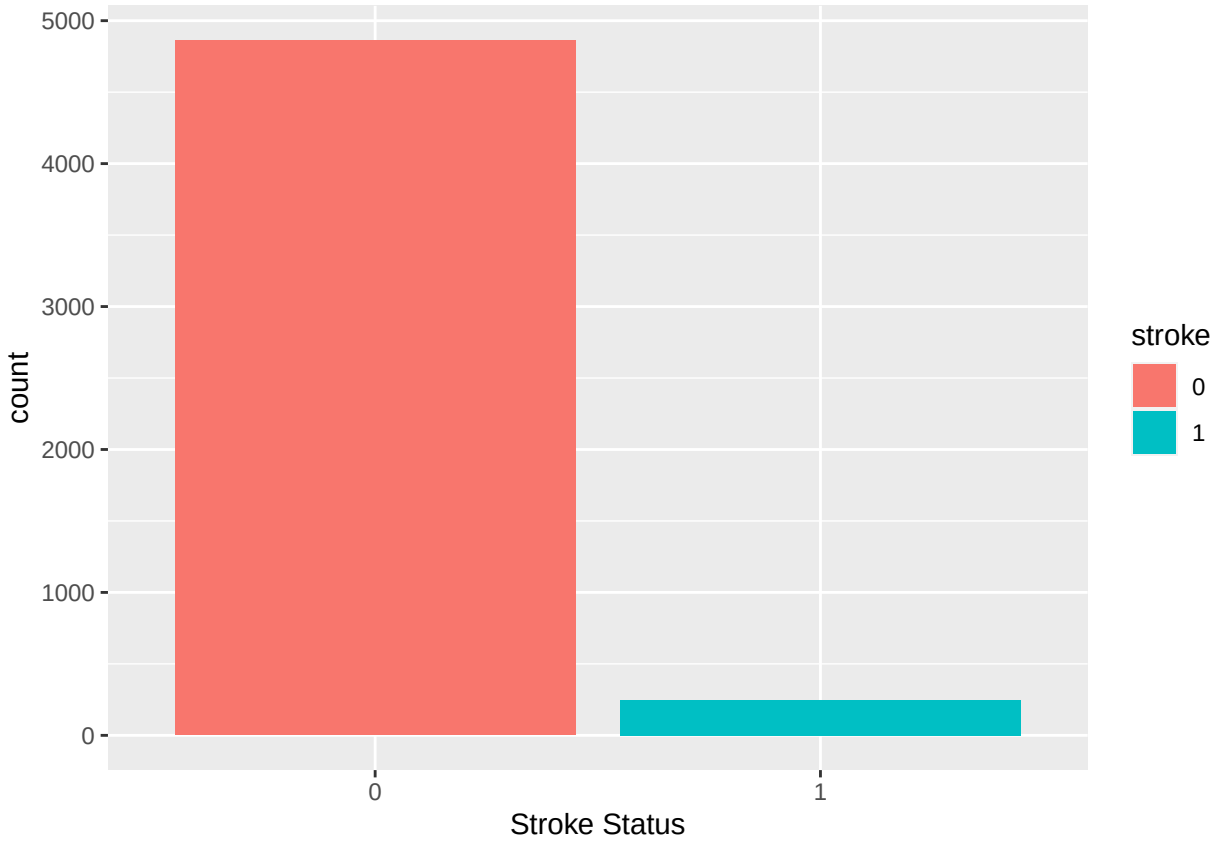


Figure 2: Classes of Stroke

```
#Utilizing smote_nc to synthesize additional data points for the minority class in the data set.
set.seed(114514)
test<-SMOTE_NC(stroke,"stroke",perc_maj = 25,5)
```

```
## Variables are continous and categorical, SMOTE_NC could be used.
```

```
##   |                                                                      |
```

```
#write.csv(test, file = "Smote0.25.csv")
#test <- as.data.frame(unclass(test),stringsAsFactors=TRUE)
```

We can see from Figure 2 that the class of stroke status was imbalanced, therefore, we utilized the technique called Synthetic Minority Oversampling Technique(SMOTE) to synthesize data points for the minority class, therefore increase the proportion of the minority class in the data set and enable us to build an usable classifier.

The SMOTE technique is an oversampling technique that utilizes the technique called k-Nearest Neighbor algorithm to synthesize data.In this project we choose to use SMOTE_NC, which is a modification of SMOTE that is able to handle data set with both categorical and numeric features.For numeric features in the data set, SMOTE_NC randomly choose k data points from the minority class in the data set and synthesize new data points on the line segment between each of the two data points. For categorical features, SMOTE_NC selects the most frequent category of the nearest neighbor data points and assign it to the new synthesized data point. (Imbalanced learn)

This technique allows us to mitigate the imbalance in the data set with less concern on the issue of overfitting in regular oversampling, which make the model giving better predictions on training data but also able to making correct prediction on new data provided(Wijaya).

An experimental package created by Dongyuan Wu from github was utilized to conduct SMOTE_NC for our data set that consists of both categorical and numeric features(Wu).

EDA

```
ggplot(data=test,aes(x=bmi))+
  geom_boxplot()
```

We can see from Figure 3 that the distribution of patients' BMI is rightly skewed.

```
positive_stroke<-test$stroke==1
positive_stroke_set<-data.frame("Stroke"=as.factor(positive_stroke),"Gender"=test$gender)

stroke_gender<-positive_stroke_set%>%
  group_by(Gender)%>%
  summarise(n = n()) %>%
  ggplot(aes(x = Gender, y = n,fill=Gender))+
  geom_col()+
  labs(y="Count of Stroke Events")
stroke_gender
```

We can see from Figure 4 that there are more female patients among all patients who had stroke.

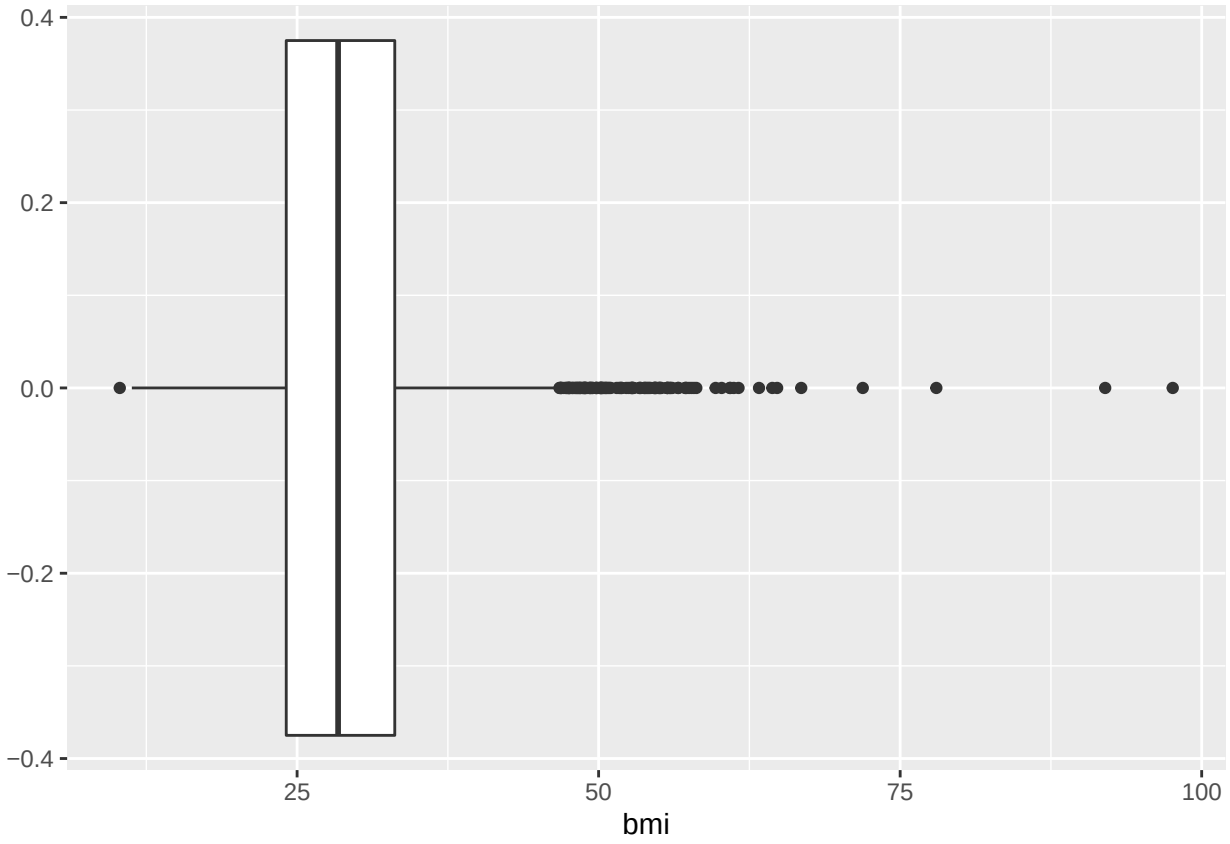


Figure 3: Distribution of BMI

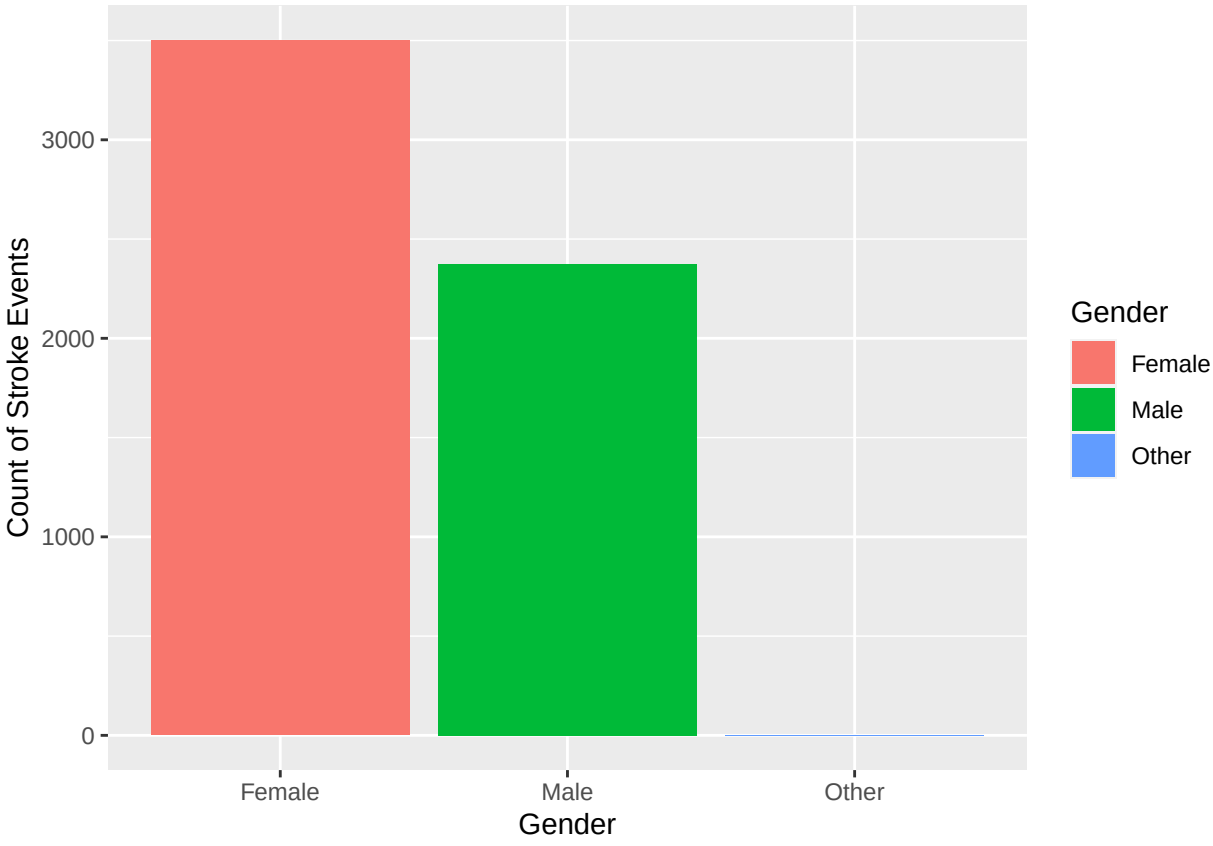


Figure 4: Stroke Event Across Gender

```
positive_stroke_work<-data.frame("Stroke"=as.factor(positive_stroke),"Work"=test$work_type)

stroke_work<-positive_stroke_work%>%
  group_by(Work)%>%
  summarise(n = n()) %>%
  ggplot(aes(x = Work, y = n,fill=Work))+
  geom_col()+
  labs(y="Count of Stroke Events")+theme(axis.text.x = element_text(angle = 45, vjust = 1, hjust=1))
stroke_work
```

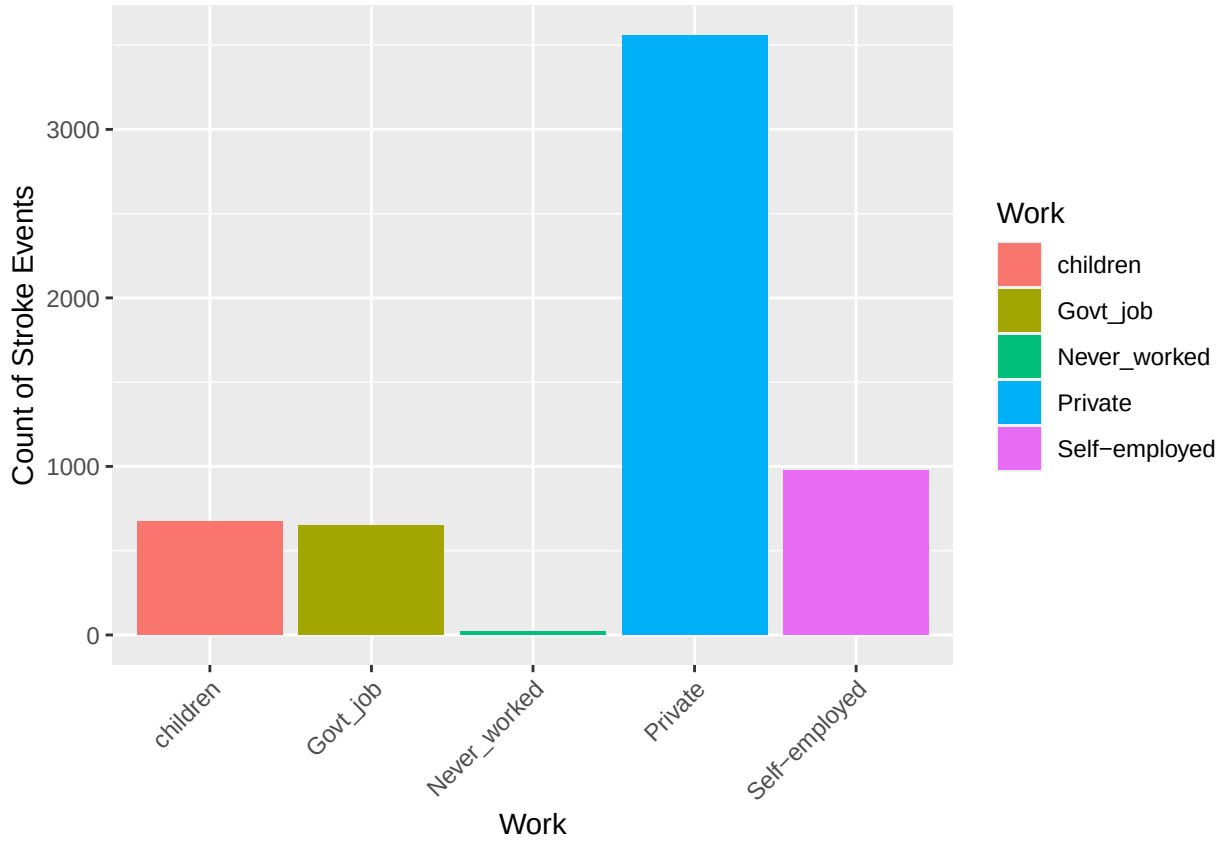


Figure 5: Stroke Event Across Work Type

We can see from Figure 5 that their are more patients who had stroke that has private as their work type.

```
ggplot(data=test,aes(x=avg_glucose_level))+
  geom_boxplot()
```

We can see from Figure 6 that the distribution of average glucose level of patients is rightly skewed.

```
ggplot(data=test,aes(x=age,y=bmi))+
  geom_point()+
  geom_smooth()
```

```
## `geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'
```

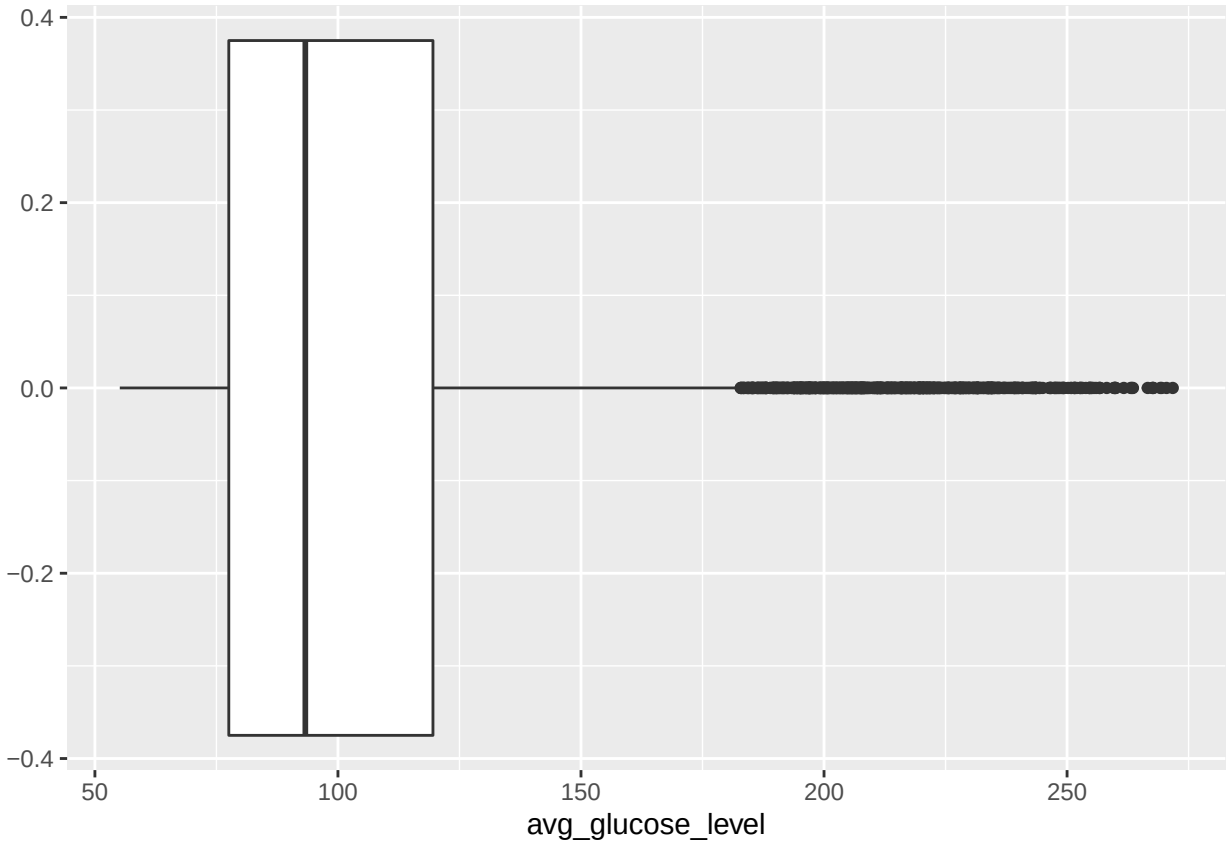


Figure 6: Distribution of Average Glucose Level

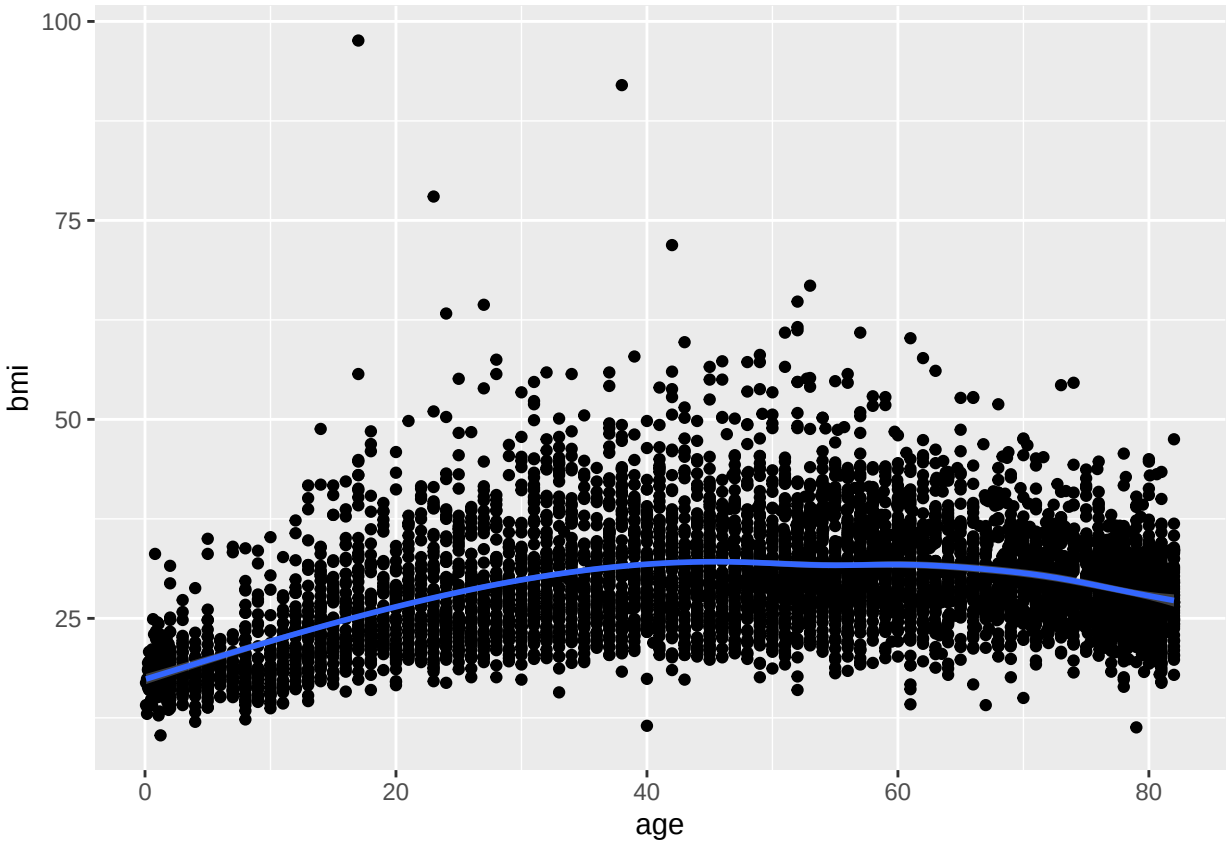


Figure 7: Scatterplot of Patients' Age vs BMI

We can see from Figure 7 that patients' age and their BMI may have correlation.

```
ggplot(data=test,aes(x=avg_glucose_level,y=bmi))+
  geom_point()+
  geom_smooth()
```

```
## `geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'
```

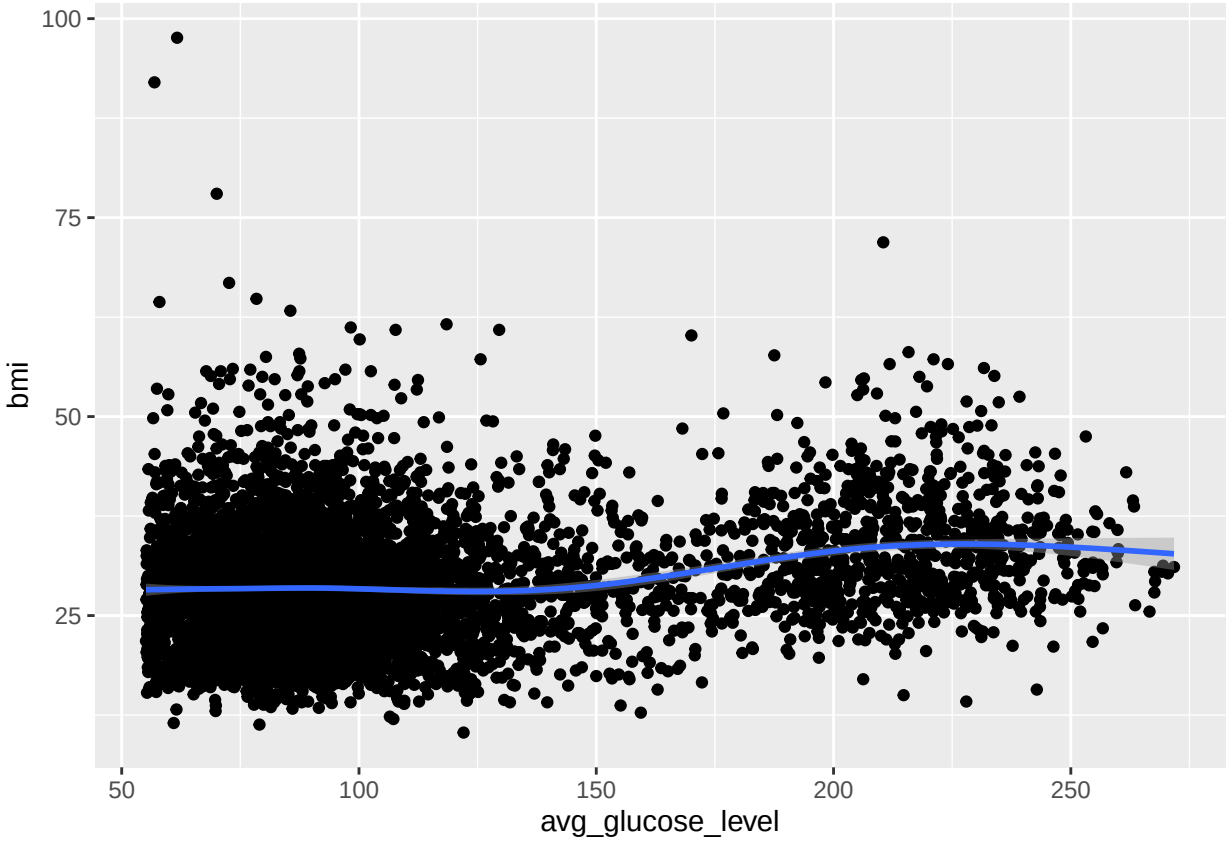


Figure 8: Scatterplot of Blood Glucose Level vs BMI

We can see from Figure 8 that patients' BMI and their average blood glucose level may have correlation.

```
ggplot(data=test,aes(x=age,y=bmi,color=stroke))+geom_point()
```

We can see from Figure 2.9 that there are more older adults who had stroke.

Section 3: Method 1:k Nearest Neighbor(KNN)

Section 3.1: Introduction

We begin building the classifier by implementing a technique called K-Nearest Neighbor(KNN) to predict the stroke status of the patients.It uses the stroke status of the k nearest neighbor data points to predict the stroke status of any given data point in the data set. We employed this technique to predict patients' stroke status since it does not have explicit training before making predictions, therefore allowing the data of new patients to be added without the need of training the model again(Kumar).

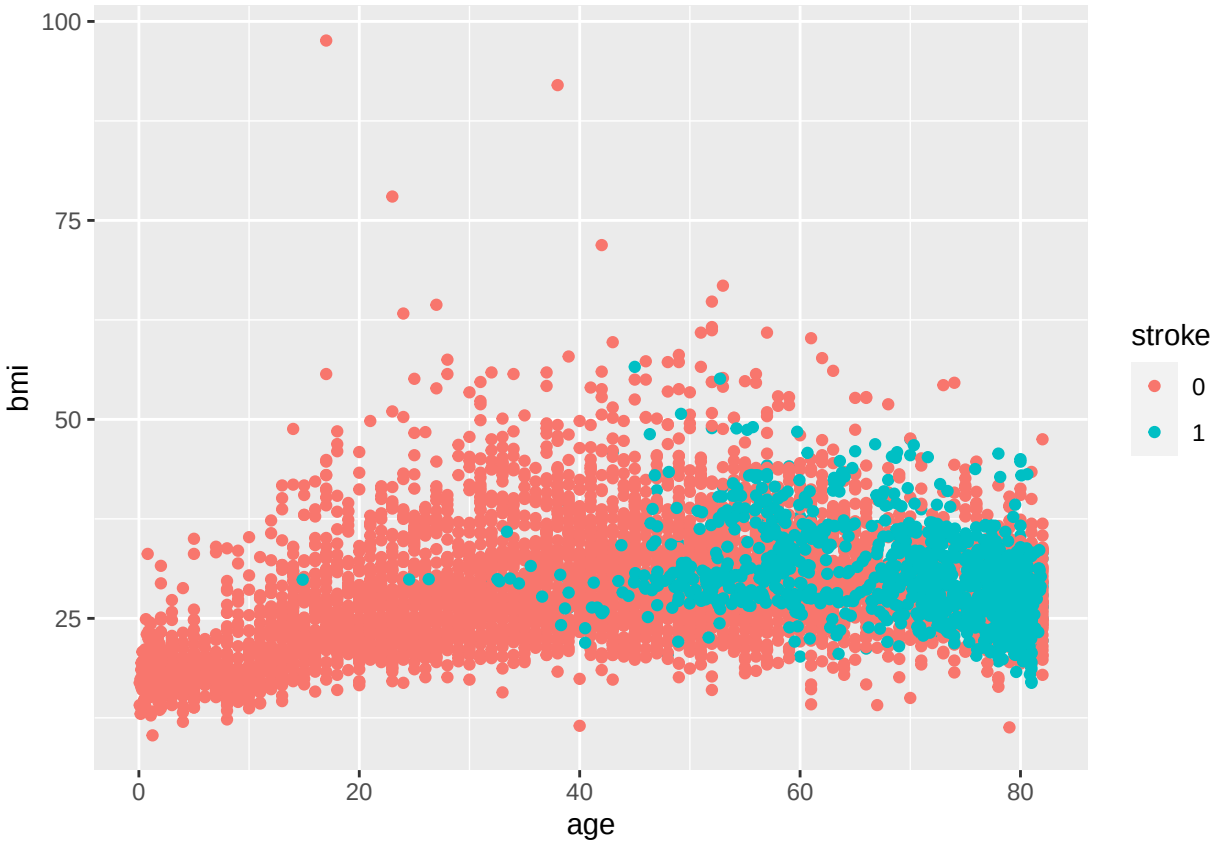


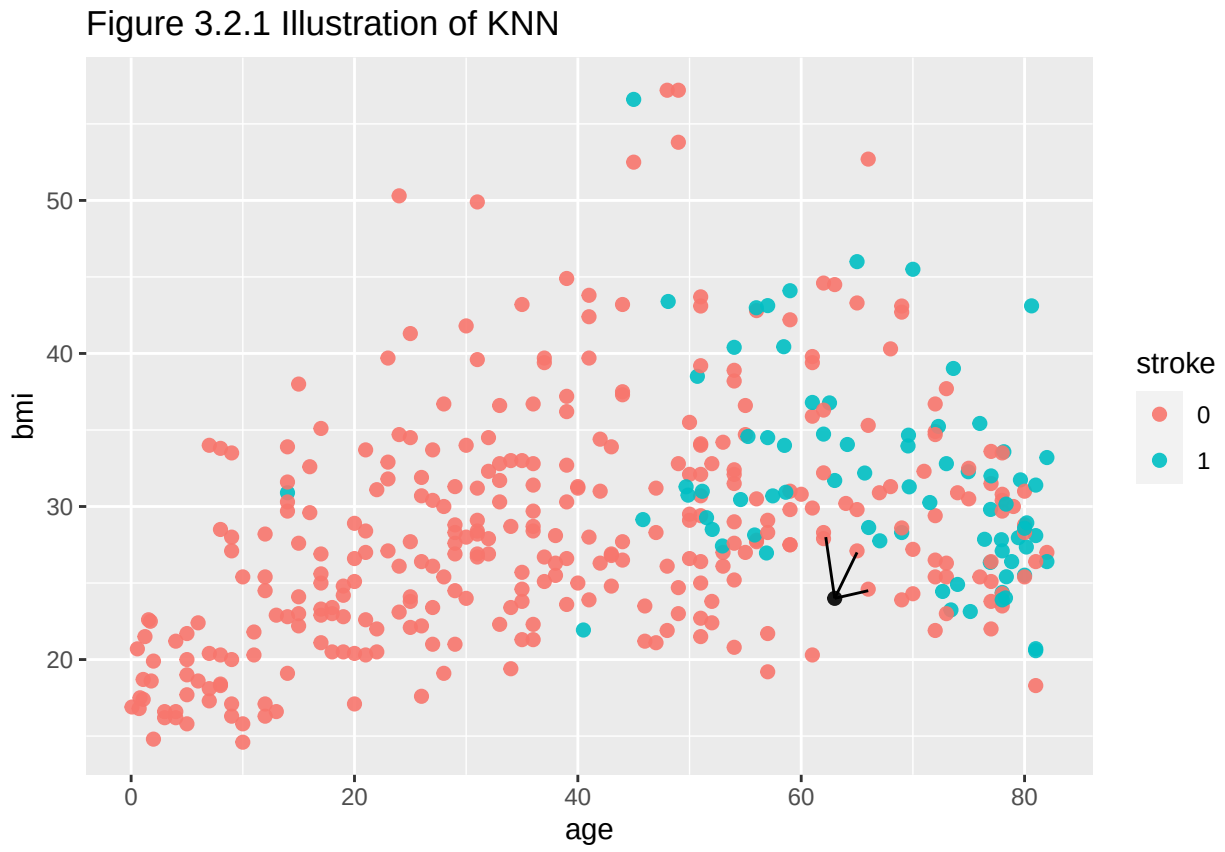
Figure 9: Age vs BMI Plot Respect to Stroke Status

Sectio 3.2 Method

Figure 3.2.1 is an illustration of the kNN technique with only age and bmi as the features utilized in kNN, where 0 = the patient had a stroke, 1 = the patient have not had stroke(For demonstration purpose, we only sampled 400 rows of data from the cleaned stroke data set).

```
set.seed(114514)
numRow<-nrow(test)
chosenDemo<-sample(1:numRow,400,replace = FALSE)
set_demo<-test[chosenDemo,]
```

```
demo_data<-data.frame("Age"= rep(NA,1),"BMI"=rep(NA,1),"Stroke"=rep(NA,1))
demo_data$Age[1]<-63
demo_data$BMI[1]<-24
demo_data$Stroke<-1
#demoTest<-example_data%>%
 # mutate(dengue_status_test=as.character(dengue_status))
ggplot()+
  geom_point(data = set_demo,aes(x=age,y=bmi,col=stroke),size=2,alpha=0.9,shape=19)+
  geom_point(data =demo_data,aes(x=Age,y=BMI),size=2,alpha=0.9,shape=19)+
  geom_segment(aes(x=63,xend=62.2,y=24,yend=28))+
  geom_segment(aes(x=63,xend=66,y=24,yend=24.5))+
  geom_segment(aes(x=63,xend=65.,y=24,yend=27))+
  ggtitle("Figure 3.2.1 Illustration of KNN")
```



In figure 3.1, we have a new data point representing a patient who is 63 years old with BMI equals to

24(shown as the black dot on the graph).If we set the k value for k Nearest Neighbors as 3. The kNN algorithm will find the stroke status of the 3 nearest data point to the data point of this patient, as shown in Figure 3.1 with black line segments connecting 3 neighbor points and the black dot. The algorithm the assigns the stroke status of majority of the 3 neighbor points as the predicted stroke status of this patient. Since all 3 neighbor points has 0 as their stroke status, this patient was predicted that he/she/them have not had a stroke.

We utilized 10-fold Cross Validation to assess the predictive accuracy of kNN on predict stroke of patients in the stroke data set. To do this we randomly divided the stroke data set into 10 folds with equal size, where each fold contains $\frac{1}{10}$ of the rows in stroke data set. We then started with treating fold 1 as the new test set of stroke data and the remaining 9 folds as the training data. We repeated this process for 10 times to train the model and obtain predictions on the training data. This method was used along with the process of determine the best k value for kNN.That is, we conduct 10-fold Cross Validation for each k value, and calculate the sensitivity(true positive rate,percent of patients who had strokes that we correctly predicted), specificity(true negative rate, percent of patients who have not had stroke that we correctly predicted) and the geometric mean of them.The geometric mean is calculated through taking the square root of sensitivity and specificity, the closer the two metrics are, the larger the geometric mean. The k value that provides the largest geometric mean is regarded as the best k value for this prediction through kNN. A line graph of geometric mean respect to each k value was used to indicate the best k value for our prediction.

```
#Create a data set with only numeric features for the KNN
numericOnly<-test[,c(2,8,9,11)]
```

```
set.seed(114514)
n<-nrow(numericOnly)
nk<-30
storage<-data.frame("K"=rep(NA,nk),
                    "Sensitivity" = rep(NA,nk),
                    "Specificity" = rep(NA,nk),
                    "GeoMean" = rep(NA,nk))

#Set a seed for sampling and create a pool and set fold for K fold CV
set.seed(114514)
pool<-rep(1:10,ceiling(n/10))

fold<-sample(pool,n,replace = FALSE)

#Outer loop for increment k
for(k in 1:nk){
  storage$K[k]<-k

  storage_inner<-data.frame("YHat" = rep(NA,n))
  #Inner loop for 10 fold CV
  for(i in 1:10){
    #Find data in each fold
    infold<-which(fold == i)

    #Create training and testing sets
    Train_Stroke<-numericOnly[-infold,]
    Test_Stroke<-numericOnly[infold,]
   #(set.seed(114514))
    #Run Knn
    k_preds<-knn(Train_Stroke[,c(1,2,3)],Test_Stroke[,c(1,2,3)],k=k,cl=Train_Stroke$stroke)
```

```
    #store predicted result from each fold to storage_inner and obtain predictions to     the full data
    storage_inner$YHat[infold]<-as.numeric(as.character(k_preds))
  }
    #Find rows with positive and negative result
    true1K<-which(numericOnly$stroke == 1)
    true0K<-which(numericOnly$stroke == 0)

    #Compute the amount of rows corresponding to each result
    ntrue1K<-length(true1K)
    ntrue0K<-length(true0K)
    #Compute sensitivity and specificity, as well as GeoMetric Mean for determining the best value for
    sensitivity<-sum(storage_inner$YHat[true1K] == 1)/ntrue1K
    storage$Sensitivity[k]<-sensitivity
    specificity<-sum(storage_inner$YHat[true0K] == 0)/ntrue0K
    storage$Specificity[k]<-specificity
    storage$GeoMean[k]<-sqrt(sensitivity*specificity)

    #if(k==8){
     # YHatOut <- storage_inner$YHat
    #}

}
```

```
#Create plot to find the best k for KNN
knnActualPlot<-ggplot(storage,aes(K,GeoMean))+
  geom_line()+
  labs(caption = paste("Geometric Mean, ReD Line at K=", which.max(storage$GeoMean)),title = "Figure 3.1
  (10 Fold Cross Validation)",y = " ")+
  geom_vline(xintercept = which.max(storage$GeoMean),lty = 2,col="red")
#Show Plot
knnActualPlot
```

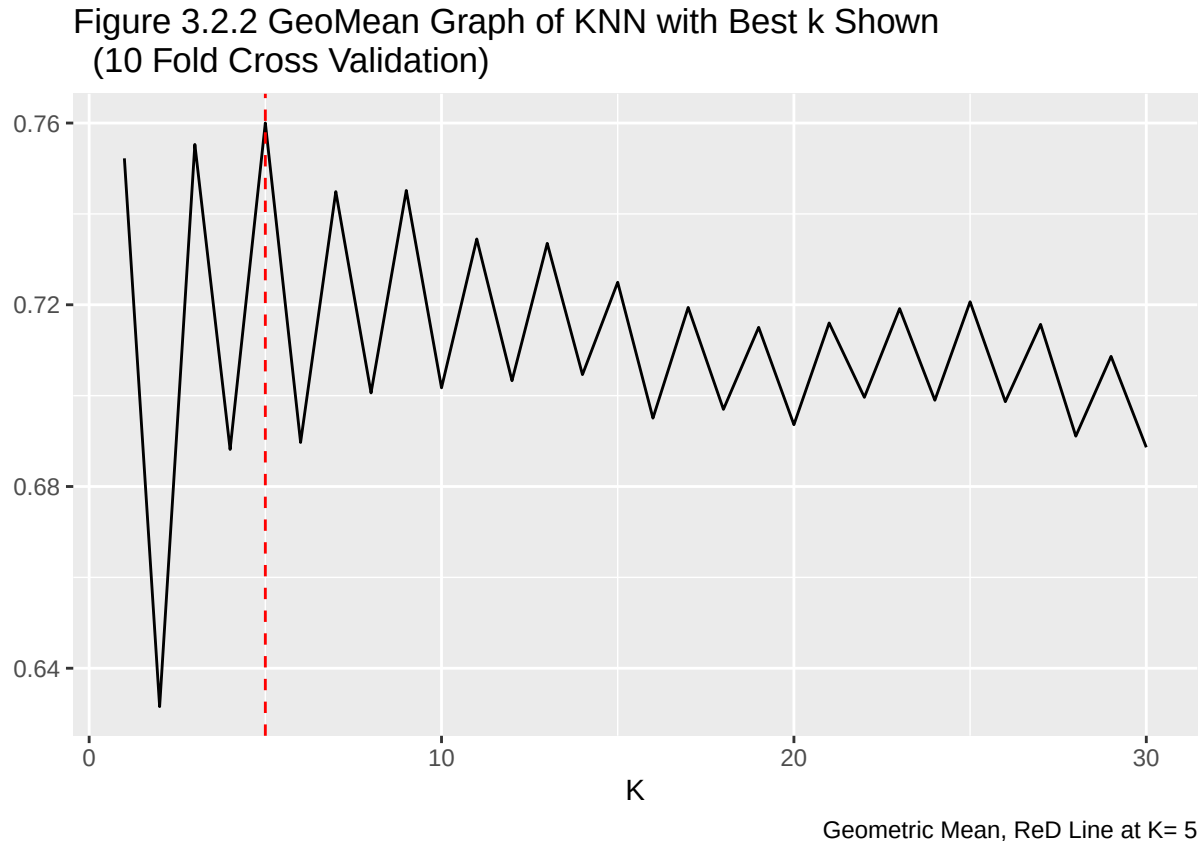


Figure 3.2.2 shows that k=5 is the best k value for the 10-fold Cross Validation, as it provides the largest geometric mean in k values from 1-30, therefore providing the most balance between sensitivity(true positive rate) and specificity(true negative rate) of the prediction.

Section 3.3: Results

We assesses the predictive accuracy of kNN through the following predictive metric: sensitivity(true positive rate),specificity(true negative rate), accuracy, and classification error rate(CER). The sensitivity indicates the percentage of patients who had stroke that we correctly predicted on their higher risk of having stroke in all patients who had stroke. The specificity indicates the percentage of patients who have not had stroke that we correctly predicted on their lower risk of having stroke in all patients who have not suffered from stroke. Accuracy indicates the percentage of patients we correctly predicted on in all patients. CER indicates the percentage of patients that we failed to make correct prediction on whether they had stroke in all patients.

Below is a confusion matrix of the prediction using kNN with k value equals to 5.

```
knitr::kable(table("Prediction" = storage_inner$YHat,"Actual" = numericOnly$stroke),caption = "Confusio
```

Table 3: Confusion Matrix of Predicting Stroke(10-fold Cross Validation with k=5

	0	1
0	4343	572
1	357	603

From Table 3, we have:

```
KNN_metrics<-data.frame("Sensitivity"= 0.514,"Specificity" = 0.928,"Accuracy" = 0.844,"CER" = 0.156)
knitr::kable(KNN_metrics,caption = "Performance Metrics KNN(K=5)")
```

Table 4: Performance Metrics KNN(K=5)

Sensitivity	Specificity	Accuracy	CER
0.514	0.928	0.844	0.156

According to the result obtained from 10-fold Cross Validation, kNN allowed us to reach a sensitivity of 0.514 and a specificity of 0.928, and an overall accuracy of 0.844 on predicting for patients in the cleaned data set on whether each of them are more likely or less likely to have stroke. Despite the accuracy of 0.83 that indicates we made accurate predictions on 83% of the patients on their risk of having stroke, the sensitivity of our prediction revealed that we only predicted accurately for 51.4% of the patients on their higher risk of having stroke.

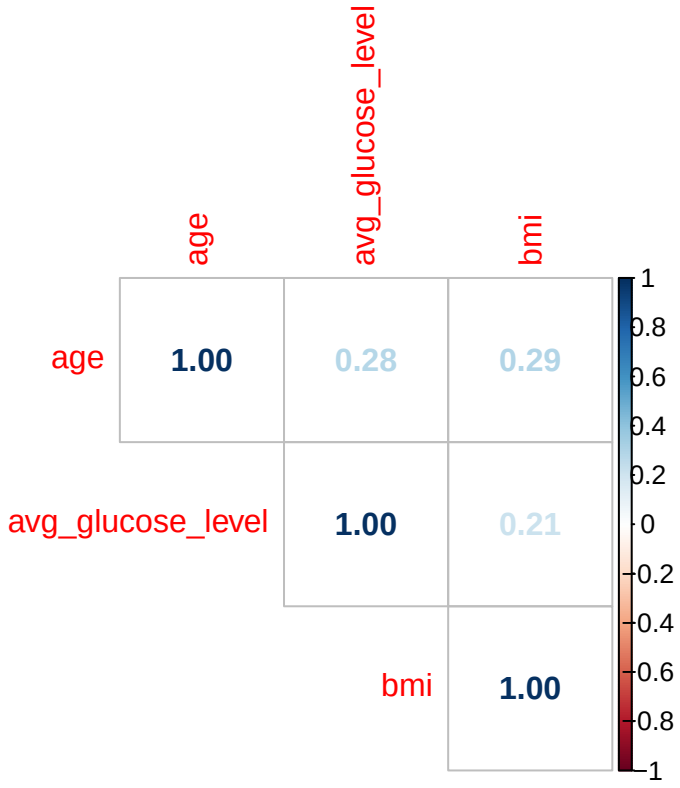
Section 4: Method 2: Ridge Regression

Section 4.1: Introduction

The kNN algorithm we used in section 3 would only accept numeric features, therefore the 8 remaining categorical features were not used for the prediction. Since we found in section 2 that the correlations might exist in between features, we created a correlation plot for the numeric features to further investigate the presence of correlations between features.

```
M<-cor(test[,c(2,8,9)])
#Create correlation plot
corrplot(M, method= "number",type = "upper",title="Figure 4.1 Correlation Plot of Numeric Variables in S
```

Figure 4.1 Correlation Plot of Numeric Variables in Stroke Data Set



We can see from the Figure 4.1 that correlations present between the numeric features in the data set, indicating that multicolinearity exists across features, therefore utilizing penalized regression models is appropriate for the data set.

Therefore, we decided to implement penalized regression techniques, along with logistic regression, to build the second classifier that not only utilizes both numeric and categorical features on making prediction, but also mitigate the impact of multicolinearity (correlations between features) in the prediction.

Section 4.2: Method

We used ridge regression to create the a prediction model for classification, as it provides the effect of shrinkage. In other words, shrinking the coefficients of correlated features towards 0 while still maintaining all the features in the cleaned data set when building the model, thus restrains the impact from them and mitigates the effect caused by multicollinearity(correlated features in the data set). To do this we need to find the tuning parameter that provides the best shrinking effect.

The general form of our model is the following:

$$log(\frac{\boldsymbol{Y}}{1-\boldsymbol{Y}}) = \boldsymbol{X_D\beta} + \boldsymbol{\epsilon}$$

, where XD is our design matrix that consists of all the features remained in the data set after data cleaning.

The deviance of the logistic regression model is defined as a measurement of how much the fitted logistic regression model deviates from a model that perfectly predicts each of the observation. In other words, deviance refers to the goodness of fit. Therefore, the smaller the deviance is, the better the model fits on the observed response(Kjytay and Kjytay).

For using ridge regression along with logistic regression, we uses Deviance, plus the penalty term $\lambda\hat{\boldsymbol{\beta}}^{\mathbf{T}}\hat{\boldsymbol{\beta}}$, which constrains the regression coefficient when correlated feature exist.

$$Deviance + \lambda\hat{\boldsymbol{\beta}}^{\mathbf{T}}\hat{\boldsymbol{\beta}}$$

Our goal is to find $\boldsymbol{\beta}$ coefficients that minimizes the following:

$$Deviance + \lambda\hat{\boldsymbol{\beta}}^{\mathbf{T}}\hat{\boldsymbol{\beta}} = (\boldsymbol{log}(\frac{\boldsymbol{Y}}{\mathbf{1}-\boldsymbol{Y}}) - \boldsymbol{X}_{\mathbf{D}}\hat{\boldsymbol{\beta}})^{T}(\boldsymbol{log}(\frac{\boldsymbol{Y}}{\mathbf{1}-\boldsymbol{Y}}) - \boldsymbol{X}_{\mathbf{D}}\hat{\boldsymbol{\beta}}) + \lambda\hat{\boldsymbol{\beta}}^{\mathbf{T}}\hat{\boldsymbol{\beta}}$$

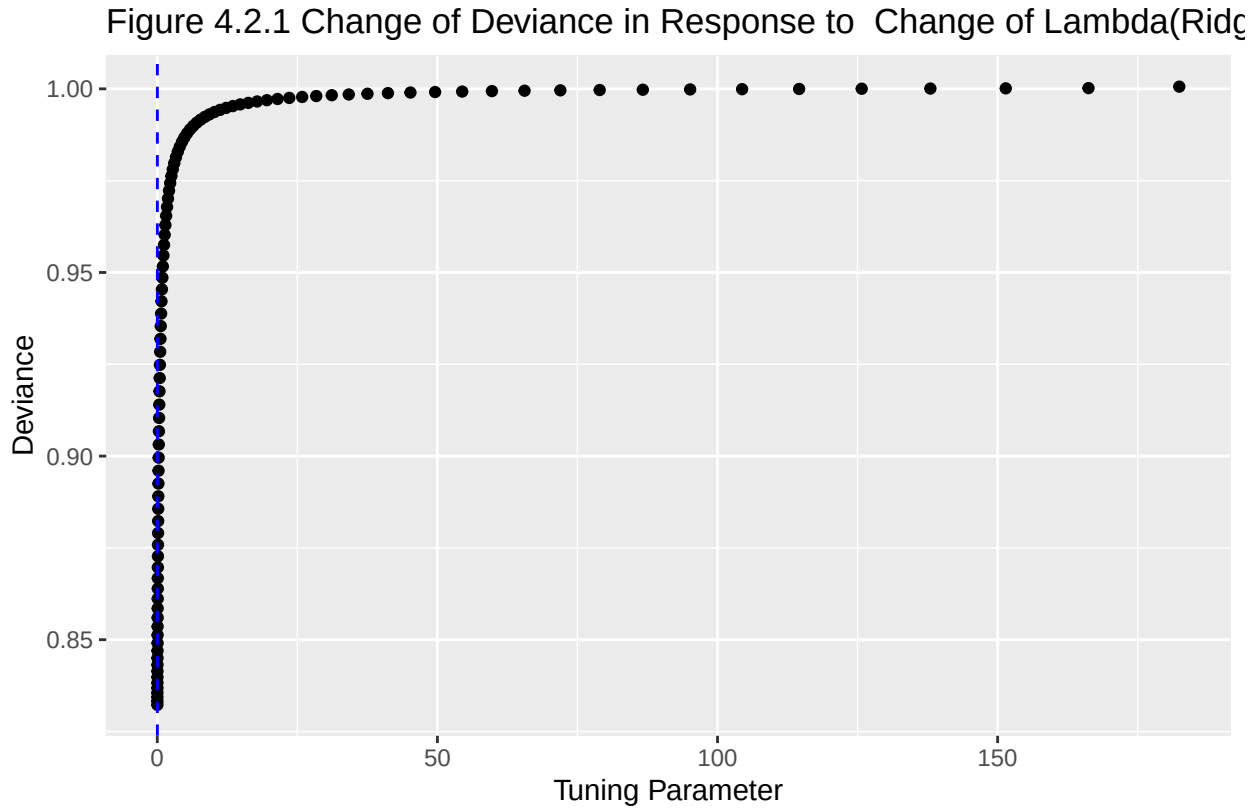
In order to implement ridge regression ,we need to find the value of tuning parameter λ that provides the most ideal shrinkage to the regression. We use the deviance to assess the predictive accuracy of the models that uses different values of λ. The logistic regression is said to be more accurate when has smaller deviance. The model with smaller deviance will perform better on predicting observed response, therefore lead to a higher predictive accuracy(Kjytay and Kjytay).

```
# Create the design matrix for ridge regression
XD <- model.matrix(stroke ~., data = test)
```

```
#Set seed for random sampling
set.seed(114514)
#Run cross validation for lambda value from 1 to 50(increment by 0.05 for each run)
ridge.modFind_Lambda <- cv.glmnet(XD[,-1], test$stroke , alpha = 0,standardize = TRUE,family = "binomial
```

```
#plot(ridge.modFind_Lambda)
```

```
ridgePlot(ridge.modFind_Lambda, metric = "Deviance" , title ="Figure 4.2.1 Change of Deviance in Respons
```




```
CV_result_ridge<-data.frame("Lambda" = ridge.modFind_Lambda$lambda,"Deviance" = ridge.modFind_Lambda$cvm
smallestRidge_Deviance<-which.min(CV_result_ridge$Deviance)
knitr::kable(CV_result_ridge[smallestRidge_Deviance,],caption = "Deviance of Model with Chosen Lambda(Ri
```

Table 5: Deviance of Model with Chosen Lambda(Ridge)

	Lambda	Deviance
100	0.0182422	0.692653

We can see from Figure 4.2.1 and Table 4.2.1 that the lowest deviance(0.69) occurred when the model is using $\lambda = 0.0182421$ as the tuning parameter the ridge regression model. Therefore, we select $\lambda = 0.0182421$ and fit the the final ridge regression model.

```
ridge.final<-glmnet(XD[,-1], test$stroke , alpha = 0,lambda = CV_result_ridge[smallestRidge_Deviance,]$l
pred_ridge<-predict(ridge.final,newx=XD[,-1],type = "response")
```

```
unadjusted_pred_ridge<-ifelse(pred_ridge>0.5,"1","0")

knitr::kable(table("Prediction" = unadjusted_pred_ridge,"Actual"= test$stroke),caption = "Confusion Matr
```

Table 6: Confusion Matrix for Prediction of Stroke Via Ridge(Unadjusted)

	0	1
0	4528	744
1	172	431

From Table 4.2.2, we have:

```
ridge_raw<-data.frame("Sensitivity"=0.367,"Specificity"=0.963,"Accuracy"=0.844,"CER"=0.156)
knitr::kable(ridge_raw,caption = "Performance Metrics Ridge(Threshold Unadjusted)")
```

Table 7: Performance Metrics Ridge(Threshold Unadjusted)

Sensitivity	Specificity	Accuracy	CER
0.367	0.963	0.844	0.156

We can see from Table 4.2.3 that although our ridge regression model reaches an overall predictive accuracy of 0.844, we only reached a sensitivity of 0.366, meaning that among all of the patients who had stroke, we are only able to accurately predict 36.6 % them on their higher risk of suffering from stroke. This is caused by the imbalanced nature of our data set(Brownlee).

If we uses the default setting 0.5 as the threshold used on predicting whether a patient has higher risk of suffering from stroke,the sensitivity(true positive rate) of our prediction model is going to be low while the specificity is going to be high since majority of the observations we use to build the model consist of patients who have not suffered from stroke(Brownlee). To mitigate the issue and reach our goal of predicting if a patient has higher risk of suffering from stroke, we utilizes ROC curve to find the ideal threshold of the prediction(Google Developers Machine Learning Crash Course).

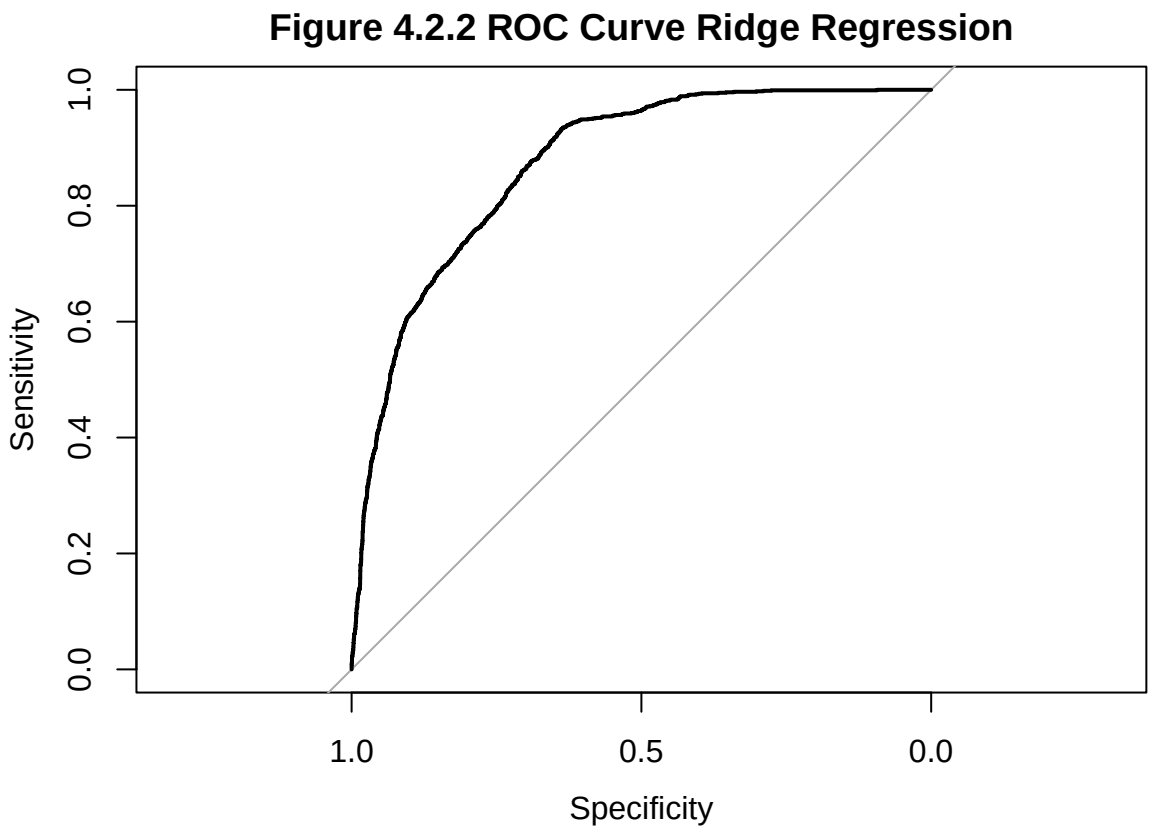
```
roc_ridge<-roc(test$stroke,pred_ridge)
```

```
## Setting levels: control = 0, case = 1
```

```
## Warning in roc.default(test$stroke, pred_ridge): Deprecated use a matrix as
## predictor. Unexpected results may be produced, please pass a numeric vector.
```

```
## Setting direction: controls < cases
```

```
plot(roc_ridge,main="Figure 4.2.2 ROC Curve Ridge Regression")
```




```
#knitr::kable(auc(roc_ridge))
holder <- data.frame("Threshold" = roc_ridge$thresholds, "Sensivity" = roc_ridge$sensitivities, "Spec" =
knitr::kable(holder[which.max(holder$GMean),]
,caption = "Threshold With Ideally Balanced Sensitivity and Specificity and Largest Geometric Mean ")
```

Table 8: Threshold With Ideally Balanced Sensitivity and Specificity and Largest Geometric Mean

	Threshold	Sensivity	Spec	GMean
3482	0.1913847	0.8604255	0.7057447	0.7792565

We can see from Table 4.2.4 that threshold = 0.191465 provides us a balance between sensitivity and specificity of the prediction model, since it has the largest largest geometric mean that indicate the ideal balance between sensitivity and specificity of the prediction model.

```
pred_ridge_Y<-ifelse(pred_ridge>holder[which.max(holder$GMean),]$Threshold,"1","0")
```

Section 4.3: Results:

Below is a confusion matrix of prediction on whether a patient in our cleaned data set has higher risk of having stroke.

```
knitr::kable(table("Prediction" = pred_ridge_Y,"Actual"= test$stroke),caption = "Confusion Matrix for Pr
```

Table 9: Confusion Matrix for Prediction of Stroke Via Ridge(threshold adjusted

	0	1
0	3317	164
1	1383	1011

From Table 4.3.1,we have;

```
ridge_metrics<-data.frame("Sensitivity"=0.860,"Specificity"=0.706,"Accuracy"=0.737,"CER"=0.263)
knitr::kable(ridge_metrics,caption = "Performance Metrics Ridge Regression(Threshold Adjusted")
```

Table 10: Performance Metrics Ridge Regression(Threshold Adjusted

Sensitivity	Specificity	Accuracy	CER
0.86	0.706	0.737	0.263

We can see from Table 4.3.2 that the ridge regression model along with logistic regression reached a sensitivity of 0.86, meaning we correctly predicted 86% of the patients on their higher risk of having stroke among all patients with higher stroke risk in our cleaned data set. The specificity of our model is 0.706, which indicates that the model correctly predicted for 70.6% of patients on their lower risk of having stroke among all patients with lower risk of having stroke. An overall accuracy of 73.7 indicates that the model correctly predicts the stroke risk status for 73.7% of the patients. A CER of 0.263 indicates that the model incorrectly predicted the stroke risk status of 26.3% of the patients.

The ridge regression model enabled us to predict more accurately than kNN on identifying people with higher stroke risk. However, the increased CER on prediction would lead to more incorrect prediction on people' stroke risk. Therefore, further techniques may be needed for mitigating this issue and developing better classifier on stroke risk prediction.

Section 5: Method 3 Forests

Section 5.1: Introduction

Although the ridge regression model we used in Section 4 reached a higher predictive accuracy on predicting patients with higher risk on having stroke, the classification error rate of the prediction model was 0.263, meaning the risk status of 26.3% of the patients from the cleaned data set was incorrectly predicted. Thus, we introduce the technique called Bagged Forest to build the third classifier and attempt to mitigate this situation. As it may provide higher accuracy on the prediction on stroke risk status. To do this we need to discuss about the classification tree in the next section, which are the components of the bagged forest.

Section 5.2: Method

The classification tree uses tree representation to resolve classification problems(GeeksForGeeks). Below is a sample decision tree created based on features including gender, age, hypertension, heart disease status, marriage status and stroke(response variable).

```
##sample classification tree in bagging
set.seed(114514)
select_rows<-sample(1:n,n,replace = TRUE)
select_set<-test[select_rows,c(1:5,11)]
sample_tree<-rpart(stroke~.,method = "class",data=select_set)
fancyRpartPlot(sample_tree,sub="",cex=0.75)
```

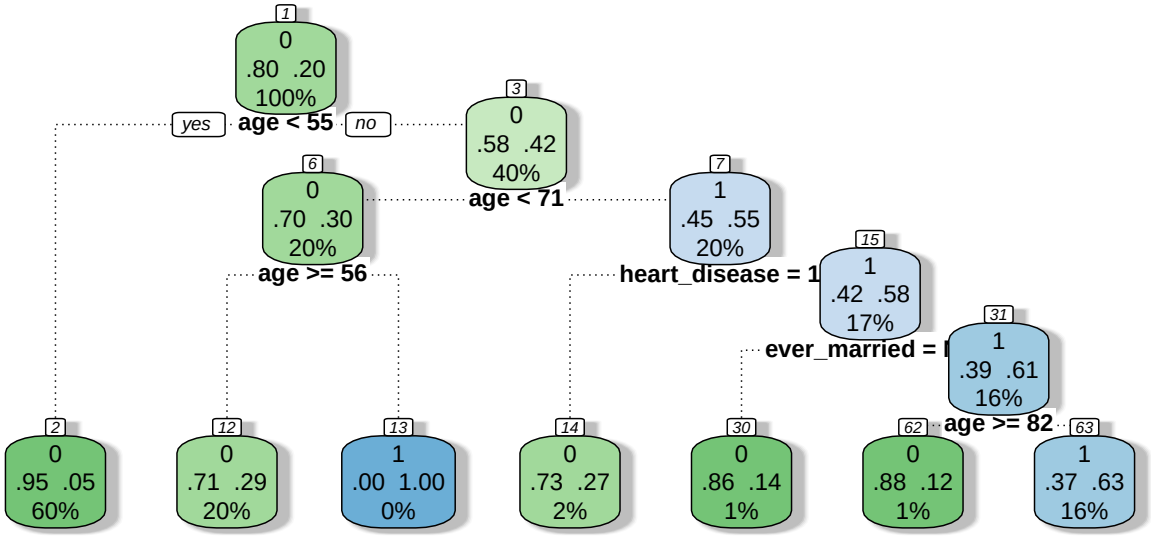


Figure 10: Sample Classification Tree (Only Use 6 features)

At the begging of this tree, we put the patients of the entire cleaned data set as the "root" of the tree, then we split the root of the tree into two nodes based on the features we used, which distribute the patients in the cleaned data set into the two nodes based on their class on this specific feature. The split was decided based on the way that gives us the smallest Gini Index (A measure of a split's effectiveness of correctly classifying a patient into a node in the tree). This process was repeated until we get the lowest Gini Index, which indicates the most ideal classifying ability of a classification tree(Dash).

However, a single decision tree is prone to overfitting, which results in a high variance on the classifier. In other words, the tree model is trained to tightly to the data set we use to create it, and made it unusable to predict for the stroke status of new patients in screening(Liberman).

The technique of bagged forest consists of a series of decision trees(in this project, classification trees). The bagged forest utilizes the bagging(abbreviation of Bootstrap Aggregation) process, which randomly draw out rows of data from our cleaned data set, and create classification trees with using these data. All the features in the data set will be used to create each classification tree. When using the bagged forest to predict the risk of stroke of a patient, each tree in the forest will make a prediction based on the data, the majority of the prediction result will be assigned to the patient as the predicted stroke risk status.

```r
#Set seed
set.seed(100)
set_for_forest<-test
#set_for_forest$hypertension <- as.numeric(set_for_forest$hypertension)
#set_for_forest$heart_disease <- as.numeric(set_for_forest$heart_disease)
bagged<-randomForest(stroke~.,data=test,mtry=10,importance=TRUE,ntree=1000,compete=FALSE)
```

```r
#bag forest prediction
for_predict<-test[,-11]
pred_bag<-predict(bagged)
#table(pred_bag)
#table("Prediction"= pred_bag,"Actual" = test$stroke)
```

```r
#Formatted Confusion Matrix for bagged forest
knitr::kable(table("Prediction"=bagged$predicted,"Actual"=test$stroke),caption="Confusion Matrix of Bagg
```

Table 11: Confusion Matrix of Bagged Forest

	0	1
0	4510	296
1	190	879

From Table 11, we have:

```r
bagged_metrics<-data.frame("Sensitivity"=0.748,"Specificity"=0.960,"Accuracy"=0.917,"CER"=0.083)
knitr::kable(bagged_metrics,caption = "Performance Metrics Bagged Forest")
```

Table 12: Performance Metrics Bagged Forest

Sensitivity	Specificity	Accuracy	CER
0.748	0.96	0.917	0.083

$$Sensitivity = \frac{879}{296 + 879} \approx 0.748$$

$$Specificity = \frac{4510}{4510 + 190} \approx 0.960$$

$$Accuracy = \frac{4511 + 785}{5875} \approx 0.917$$

Works Cited

"About Stroke." Centers for Disease Control and Prevention, Centers for Disease Control and Prevention, 2 Nov. 2022, https://www.cdc.gov/stroke/about.htm.

"Stroke Facts." Centers for Disease Control and Prevention, Centers for Disease Control and Prevention, 14 Oct. 2022, https://www.cdc.gov/stroke/facts.htm.

Fedesoriano. "Stroke Prediction Dataset." Kaggle, 26 Jan. 2021, https://www.kaggle.com/datasets/fedesoriano/stroke-prediction-dataset/code.

Imbalanced Learn. "2. Over-Sampling#." 2. Over-Sampling - Version 0.10.0.dev0, https://imbalanced-learn.org/dev/over_sampling.html#smote-variants.

Wijaya, Cornellius Yudha. "5 Smote Techniques for Oversampling Your Imbalance Data." Medium, Towards Data Science, 24 May 2022, https://towardsdatascience.com/5-smote-techniques-for-oversampling-your-imbalance-data-b8155bdbe2b5.

RolandRoland."Convert All Data Frame Character Columns to Factors." Stack Overflow, 17 Dec. 2013, https://stackoverflow.com/questions/20637360/convert-all-data-frame-character-columns-to-factors.

Wu, Dongyuan. "Dongyuanwu/RSBID: Resampling Strategies for Binary Imbalanced Datasets Version 0.0.2.0000 from Github." Version 0.0.2.0000 from GitHub, 18 July 2022, https://rdrr.io/github/dongyuanwu/RSBID/.

Singh, Deepika. "Deepika Singh." Pluralsight, 12 Nov. 2019, https://www.pluralsight.com/guides/finding-relationships-data-with-r.

Reka Solymosi (maintained and updated by David Buil-Gil and Nicolas Trajtenberg). "Making Sense of Crim Data." Chapter 3 Week 3, 8 Nov. 2022, https://maczokni.github.io/MSCD_labs/week3.html.

Kumar, Naresh. "Advantages and Disadvantages of KNN Algorithm in Machine Learning." Advantages and Disadvantages of KNN Algorithm in Machine Learning, http://theprofessionalspoint.blogspot.com/2019/02/advantages-and-disadvantages-of-knn.html.

Kjytay, and Kjytay. "What Is Deviance?" Statistical Odds & Ends, 27 Mar. 2019, https://statisticaloddsandends.wordpress.com/2019/03/27/what-is-deviance/.

Brownlee, Jason. "A Gentle Introduction to Threshold-Moving for Imbalanced Classification." MachineLearningMastery.com, 4 Jan. 2021, https://machinelearningmastery.com/threshold-moving-for-imbalanced-classification/.

"Classification: Roc Curve and AUC | Machine Learning | Google Developers." Google, Google, https://developers.google.com/machine-learning/crash-course/classification/roc-and-auc#:~:text=An%20ROC%20curve%20(receive

"Decision Tree Introduction with Example." GeeksforGeeks, 10 Nov. 2022, https://www.geeksforgeeks.org/decision-tree-introduction-example/.

https://towardsdatascience.com/decision-trees-explained-entropy-information-gain-gini-index-ccp-pruning-4d78070db36c