Crystal System Talents Data Science Course

Capstone Project

Capstone

Project Title:

Stroke Risk Prediction

Group C

Group Member	
Neaw Aik Ka (L)	**TP065116**
Siah Yao Liang	**TP061861**
Tai Rui Xian	**TP060922**

Table of Contents

1 Introduction

A stroke, also known as a cerebrovascular accident (CVA), is a medical emergency that occurs when the blood supply to a part of the brain is disrupted or reduced. This lack of blood flow can cause brain cells to die, leading to permanent brain damage or even death. The effects of a stroke vary depending on which part of the brain is affected and the severity of the damage. A stroke is characterized by sudden weakness or numbness on one side of the body, difficulty speaking or understanding speech, sudden vision problems, severe headaches, and loss of balance or coordination. Hence, when experiencing stroke symptoms, it is critical to seek immediate medical attention. Medication to

dissolve blood clots, surgery to repair damaged blood vessels, and rehabilitation to help the person regain function and independence are all possible treatments.

We will conduct a stroke risk prediction using a stroke dataset in this proposal. Based on input parameters such as gender, age, various diseases, and smoking status, this dataset is used to predict whether a patient is likely to have a stroke. The status of a stroke patient will be detected earlier to avoid missing the best treatment time. This dataset includes 5110 observations with a total of 12 attributes.

2 Problem Statement

Stroke is the second-leading cause of mortality and the primary global source of disability. According to the Global Stroke Factsheet published in 2022, the lifetime chance of having a stroke has increased by 50% in the last 17 years, with 1 in 4 individuals now thought to experience one. Over the course of their lifespan, 1 in 4 people over the age of 25 will experience a stroke. This year, 12.2 million individuals will experience their first stroke, and 6.5 million of them will pass away. The number of stroke victims worldwide exceeds 110 million.

Stroke frequency has increased by 70% since 1990, stroke fatalities have increased by 43%, stroke prevalence has increased by 102%, and disability-adjusted life years have increased by 143%. The most remarkable aspect is that 86% of stroke-related fatalities and 89% of disability-adjusted life years worldwide occur in low- and middle-income nations. Families with limited means are facing an unprecedented challenge as a result of the disproportionate impact faced by lower- and lower-middle-income nations.

The World Stroke Organization also states that metabolic factors, including high systolic blood pressure, a high body mass index, a high fasting plasma glucose level, a high total cholesterol level, and a low glomerular filtration rate, are responsible for 71.0% (64.6-77.1) of the incidence of stroke. Smoking, eating poorly, and not getting enough exercise account for occurrences of the stroke load, while environmental hazards like lead exposure and air pollution account for 37.8%. Through examining some attributes that could lead to stroke, we try to predict and build a model that helps individuals identify the probability of themselves getting artery diseases in a simple and affordable way.

3 Aim / Objectives / Deliverables

- To predict the likelihood of stroke risk cases happening by using simple data.

- To develop an interface that can be used by anyone to make stroke predictions.
- Identify the simplest indicators that lead to stroke.
- Determined the appropriate data mining techniques for the prepared dataset.
- Implement suitable data mining techniques for the dataset.

4 Scope

- Exploratory Analysis. Explore suitable analytics methods and algorithms.
- Data Collection, Data Preparation, Data transformation
- Model Building
- Evaluation of results with the appropriate visualization
- Explore hidden insight in the dataset.

5 Similar System

Citation	Brief summary	Models or techniques used	Findings/ Limitations
(Carlos Fernandez-Lozano1, 2021)	Researching clinical, biochemical, and neuroimaging with outcomes on stroke patients and generating a prediction.	Random Forest	Random forest algorithm effectively used in long-term outcome prediction of mortality and morbidity of stroke patient
(Gangavarapu Sailasya1, 2021)	Based on CDC, stroke is the fifth-leading cause of death in the US. Research is conducted for predicting the occurrence of a stroke.	Logistic regression, Decision tree, random forest, KK, SVM and VB	Out of all the algorithms chosen, Naïve Bayes Classification performs best with an accuracy of 82%.
(A.Sudha, 2012)	This paper predicts classification algorithm like Decision tree, Naïve bayes and neural networks used for predicting stroke diseases and	Decision tree, Naive Bayes, Neural Network	Observation shows that neural network performance is having more accuracy, when

	principal component analysis algorithm for reducing the attributes.		compared with other two classification methods.
(Leila Amini, 2013)	Collection of data in cases of various diseases in medical sciences. For analyzing data, data mining techniques, *K*-nearest neighbor and C4.5 decision tree using WEKA is applied.	KNN, C4.5, WEKA algorithms, refences features selection	The two algorithms, C4.5 decision tree algorithm and K-nearest neighbor, can be used in order to predict stroke in high-risk groups.
(Aditya Khosla, 2010)	In this study, it compares the Cox proportional hazards model with a machine learning approach for stroke prediction on the CHS dataset. Specifically, they consider the common problems of data imputation, feature selection, and prediction in medical datasets.	Supervised learning, Machine learning approaches, classification and regression trees and feature selection	Markov decision processes
(RichardMcKinley1, 2017)	Several clinical trials have recently proven the efficacy of mechanical thrombectomy for treating ischemic stroke, within a six-hour window for therapy and introduced fully automated method to estimate penumbra.	FASTER	Random forest classifiers

6 Exploratory Analysis and Data Preparation

Exploratory analysis and data preparation is the process of investigating and comprehending a dataset's structure and characteristics. Various methods will be implemented in order to find the patterns,

connections, and anomalies in the data. The intent of the process is to acquire the data accessible to undergo further modeling and analysis.

6.1 Attribute Information

The description of the variables that are contained in a dataset is referred to as attribute information. The name of the variable, the range of values it can accept, and a brief explanation of its relevance or meaning are included. Understanding a dataset's structure and properties, as well as providing direction for the analysis and modeling process, requires attribute information. The table shown below is the raw attribute information available in our dataset.

No.	Variables Name	Attribute Information
1.	id	unique identifier
2.	gender	"Male", "Female" or "Other"
3.	age	age of the patient
4.	hypertension	0 if the patient doesn't have hypertension, 1 if the patient has hypertension.
5.	heart_disease	0 if the patient doesn't have any heart diseases, 1 if the patient has a heart disease.
6.	ever_married	"No" or "Yes".
7.	work_type	"children", "Govt_jov", "Never_worked", "Private" or "Self-employed".
8.	Residence_type	"Rural" or "Urban"
9.	avg_glucose_level	average glucose level in blood
10.	bmi	body mass index
11.	smoking_status	"formerly smoked", "never smoked", "smokes", or "Unknown"*
12.	stroke	1 if the patient had a stroke or 0 if not.

6.2 Dataset Source link

The source link of the dataset we used in the capstone project.

https://www.kaggle.com/datasets/fedesoriano/stroke-prediction-dataset

6.3 Data Import

The process of importing data and setting the working directory into the R programming environment for further analysis and modeling. This includes defining the variable types and

establishing the location and format of the data. In case to assure data correctness and integrity, data import is a vital stage in the data analysis pipeline.

```
#-------------------------------------- DATA IMPORT ------------------------------------------

library(readr)
library(dplyr)
library(psych)
library(lattice)
library(pastecs)
library(DataExplorer)
library(ggplot2)
library(ggridges)
library(scales)
library(reshape2)
library(remotes)
library(fastDummies)
library(caret)
library(caTools)
library(grid)
library(DMwR)
library(class)
library(R6)
library(lightgbm)
library(pROC)
library(rpart)
library(rpart.plot)
library(e1071)
library(yardstick)
library(randomForest)
library(ROCR)

#set working directory
setwd('D:/Crystal/Capstone_project')
stroke_ds<- read.csv("healthcare-dataset-stroke-data.csv",header=TRUE)
```

In the data import section, we have involved all the required libraries for the project with the library () syntax. It is more convenient to execute when desired to go through the modeling. Other than that, the setwd () syntax is included for setting up the working directory for the data. A working directory is useful when manipulating a file or importing and exporting a file, it will become the default location to save and access the file. To load the dataset into the working environment, we assign the dataset with the variable name stroke_ds, enter the file name, load the CSV file with the read.csv function, and give a TRUE value for the header which indicates the CSV file contains the header. Then the library, working directory, and importing of the dataset are completed.

6.4 Data Exploration

Data exploration is a process of examining and analyzing a dataset to find patterns and connections from a certain dataset. It is the purpose to get insights and a better understanding of the data, it entails visualizing and summarizing the data using statistical techniques, and it will assist the data analyst to get a more accurate result in the outcome. Data exploration is an essential step in data analysis and

aids in directing the following steps feature engineering, modeling, and data cleansing. In our case, we had determined the missing data and handled it in this stage.

```
#-------------------------------------------------------------------------------------------------
#------------------------------------ DATA EXPLORATION -------------------------------------------

#View(stroke_ds)
str(stroke_ds) #view dataset structure
dim(stroke_ds)

stroke_ds$bmi<-as.numeric(stroke_ds$bmi) #convert bmi from chr to num
stroke_ds$age<-as.integer(stroke_ds$age) #convert age from num to integer
```

```
> str(stroke_ds) #view dataset structure
'data.frame':   5110 obs. of  12 variables:
 $ id               : int  9046 51676 31112 60182 1665 56669 53882 10434 27419 60491 ...
 $ gender           : chr  "Male" "Female" "Male" "Female" ...
 $ age              : num  67 61 80 49 79 81 74 69 59 78 ...
 $ hypertension     : int  0 0 0 0 1 0 1 0 0 0 ...
 $ heart_disease    : int  1 0 1 0 0 0 1 0 0 0 ...
 $ ever_married     : chr  "Yes" "Yes" "Yes" "Yes" ...
 $ work_type        : chr  "Private" "Self-employed" "Private" "Private" ...
 $ Residence_type   : chr  "Urban" "Rural" "Rural" "Urban" ...
 $ avg_glucose_level: num  229 202 106 171 174 ...
 $ bmi              : chr  "36.6" "N/A" "32.5" "34.4" ...
 $ smoking_status   : chr  "formerly smoked" "never smoked" "never smoked" "smokes" ...
 $ stroke           : int  1 1 1 1 1 1 1 1 1 1 ...
> dim(stroke_ds)
[1] 5110   12
```

In this stage, it is another essential part to prepare the dataset for modeling. In the first case, we view the structure with the str () function and view the dimension with the dim () function for stroke_ds which represents the variable name of the dataset. We can result with the total row and variables available in the data frame and meanwhile all the variable structures in list format. The list will indicate the data type of the variables and the sample data value in the variable aside from the variable name and datatype. In the same process, we converted two variables which are bmi and age from data type chr to num and data type num to int. It is due to the reason to make the dataset more logical and reasonable; we can view the latest dataset structure with the same syntax.

6.4.1 Missing Data

Missing data refers to values that are missing from a dataset due to their omission from collection, loss, or corruption during the data collection or processing process. Since missing data can introduce bias, lower statistical power, and skew conclusions, it can significantly affect the accuracy of data analysis. The right procedures for handling missing values, such as mean imputation, multiple imputations, or deletion of missing data, must be carefully considered deal with missing data.

```
> stroke_ds %>%
+   filter(!complete.cases(.)) #%>%   View()#BMI has missing
      id gender age hypertension heart_disease ever_married     work_type Residence_type avg_glucose_level bmi  smoking_status
1  51676 Female  61            0             0          Yes Self-employed          Rural            202.21  NA   never smoked
2  27419 Female  59            0             0          Yes       Private          Rural             76.15  NA         Unknown
3   8213   Male  78            0             1          Yes       Private          Urban            219.84  NA         Unknown
4  25226   Male  57            0             1           No      Govt_job          Urban            217.08  NA         Unknown
5  61843   Male  58            0             0          Yes       Private          Rural            189.84  NA         Unknown
6  69160   Male  59            0             0          Yes       Private          Rural            211.78  NA formerly smoked
7   1845 Female  63            0             0          Yes       Private          Urban             90.90  NA formerly smoked
8  37937 Female  75            0             1           No Self-employed          Urban            109.78  NA         Unknown
9  18587 Female  76            0             0           No       Private          Urban             89.96  NA         Unknown
10 15102   Male  78            1             0          Yes       Private          Urban             75.32  NA formerly smoked
11  8752 Female  63            0             0          Yes      Govt_job          Urban            197.54  NA   never smoked
12 66400   Male  78            0             0          Yes       Private          Urban            237.75  NA formerly smoked
13  7356   Male  75            0             0          Yes       Private          Urban            104.72  NA         Unknown
14 70676 Female  76            0             0          Yes      Govt_job          Rural             62.57  NA formerly smoked
15 45805 Female  51            0             0          Yes       Private          Urban            165.31  NA   never smoked
16 26015 Female  66            0             0          Yes Self-employed          Urban            101.45  NA         Unknown
17 70042   Male  58            0             0          Yes       Private          Urban             71.20  NA         Unknown
18  2346   Male  58            0             0          Yes       Private          Urban             82.30  NA          smokes
19 36706 Female  76            0             0          Yes Self-employed          Urban            106.41  NA formerly smoked
20 14164 Female  72            0             0          Yes       Private          Urban            219.91  NA         Unknown
```

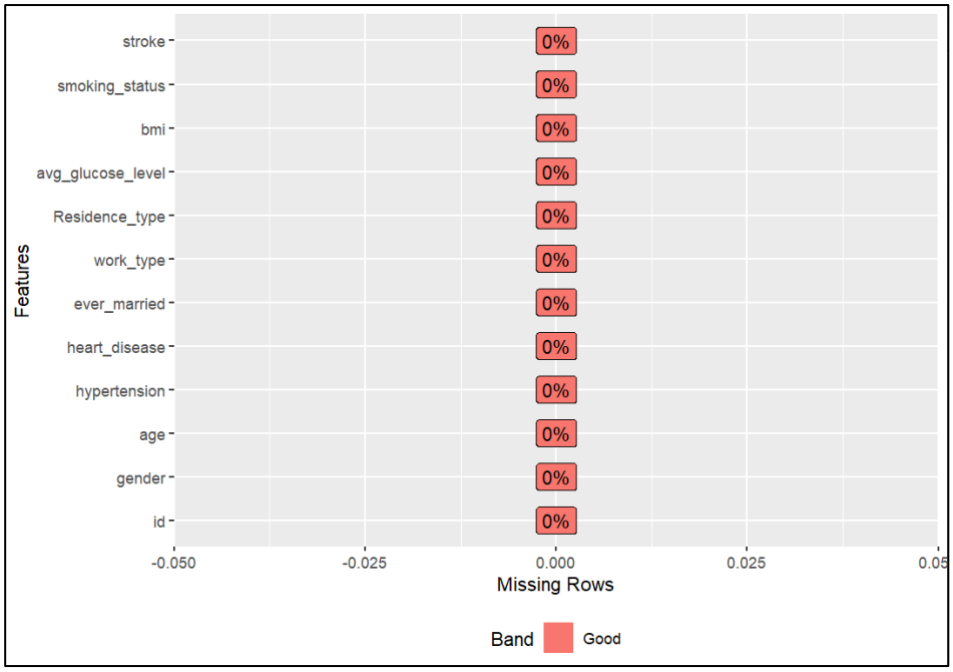
```
> unique(stroke_ds$gender) #no missing
[1] "Male"   "Female" "Other"
> mean(stroke_ds$age) #no missing
[1] 43.21526
> unique(stroke_ds$hypertension) #no missing
[1] 0 1
> unique(stroke_ds$heart_disease) #no missing
[1] 1 0
> unique(stroke_ds$ever_married) #no missing
[1] "Yes" "No"
> unique(stroke_ds$work_type) #no missing
[1] "Private"       "Self-employed" "Govt_job"      "children"      "Never_worked"
> unique(stroke_ds$Residence_type) #no missing
[1] "Urban" "Rural"
> mean(stroke_ds$avg_glucose_level) #no missing
[1] 106.1477
> unique(stroke_ds$smoking_status) #no missing but have unknown
[1] "formerly smoked" "never smoked"    "smokes"          "Unknown"
> unique(stroke_ds$stroke) #no missing
[1] 1 0
> mean(stroke_ds$bmi) #have missing
[1] NA
```

To detect the missing data from the dataset, we implemented the filter function to show the possible missing data row in the console and placed the unique() and mean() function to check the existence of missing values in each of the variables. In the result, the bmi variable is containing missing data with an NA value.

6.4.2 Handle variable BMI Missing Value with Mean method.

In purpose to obtain more accurate data, handling missing values for the dataset is necessary and, in our situation, we handle the BMI variable from the stroke_ds dataset by imputing the mean of itself with the specific function and syntax. Meanwhile for the gender variable, even if it doesn't contain any missing value, however, the value of other is considered removed from the gender column. Since it might not have any information from the value, hence we undergo to remove it from the dataset column with the filter function and assign back the new dataset to the same variable name stroke_ds.

```
#HANDLE BMI MISSING VALUE BY IMPUTING MEAN OF IT
stroke_ds$bmi[is.na(stroke_ds$bmi)]<- mean(stroke_ds$bmi,na.rm = TRUE)

# since there is only one "Other" in gender column so we will remove it
stroke_ds<- stroke_ds %>% filter(!gender=="Other")

plot_missing(stroke_ds)# check missing data in all columns
```

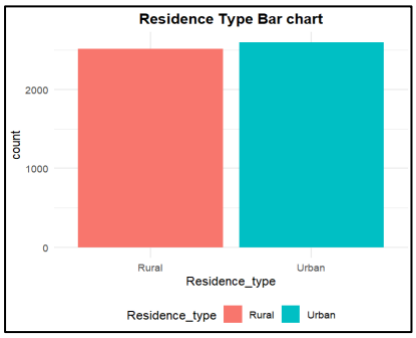


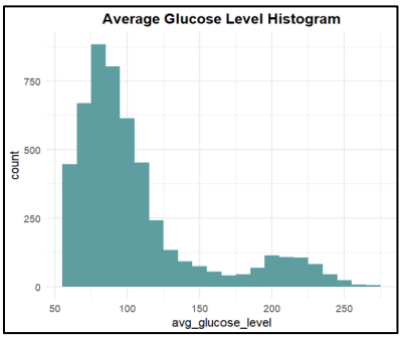
From the plot_missing() function, we can implement it to check the missing data in all variables columns with visualize method. As the result shown in the plot in percentage, it indicates all the variables column is placed with zero percent missing row on it. This plot conveys that all the missing data is successfully removed and handled from the dataset.

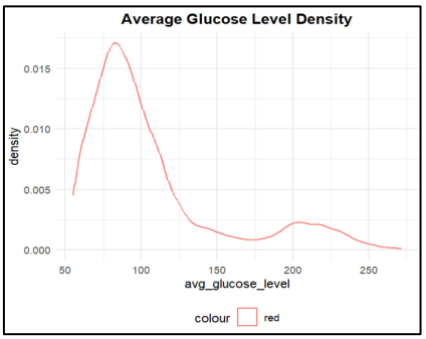
6.5 General Overview

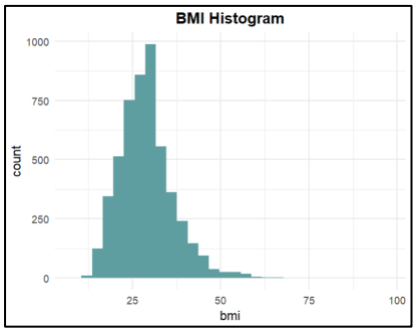
```
#------------------------------ GENERAL OVERVIEW --------------------------------------
#install.packages("ggridges")

describe(stroke_ds)#check status of data
stat.desc(stroke_ds)
summary(stroke_ds)
plot_str(stroke_ds)#visualize for data attributes
```

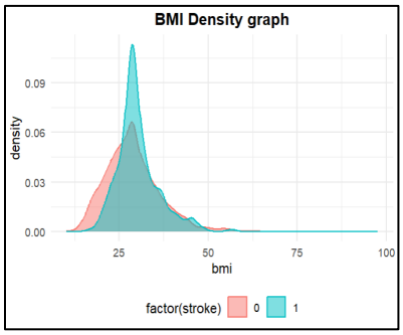
All the graph was implemented for all the variables available in the dataset. A box plot is included to illustrate hypertension and age variables.

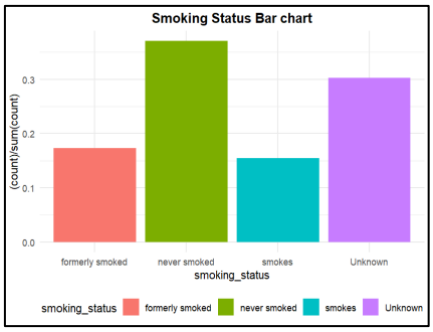




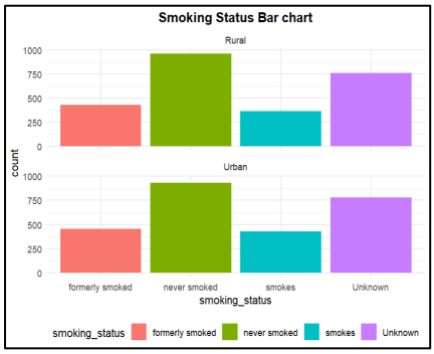


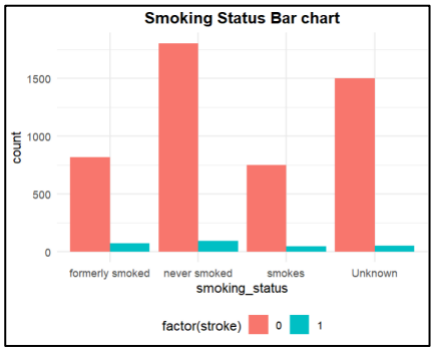


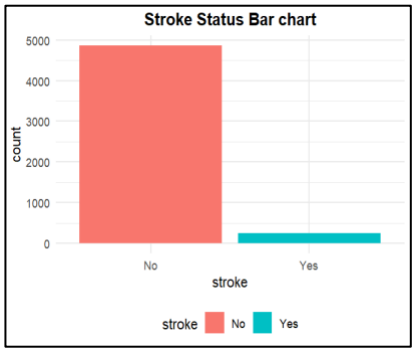


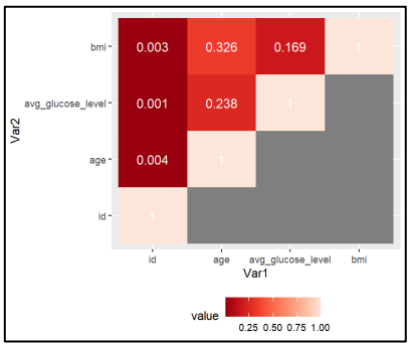


Different variables were implemented with different visualizations such as histogram, bar chart, line chart, box plot, density graph and heat map in case to achieve the more appropriate visualization and better visual performance.









Other than that, after reflecting all the visualization for all the 12 variables in the raw dataset, a heatmap also had involved to show the correlation matrix from the numeric variables between the four variables, bmi, avg_glucose_level, age, and id. The correlation matrix and lower triangular are duplicates of the upper triangle elements. The 'NA' value is used to avoid redundancy in the final output.

6.6 Data Transformation & Data Pre-Processing

The procedure for data analysis requires both pre-processing and data transformation. Data transformation includes data cleansing, data normalization, and feature engineering, which entails turning the original dataset into a format that can be used for analysis. Data pre-processing is used to prepare the data for analysis using statistical techniques and methodologies including data imputation, outlier detection, and dimensionality reduction. It aids to ensure data consistency, enhancing precision, and facilitating effective modeling and prediction.

```
#-------------------------------- DATA TRANSFORMATION ----------------------------------------
#-------------------------------- DATA PRE-PROCESSING ----------------------------------------
library(dplyr)
# remove index column
stroke_ds <- select(stroke_ds, -id)
```

According to the code snippet above, after loading the library ‘dplyr’, we removed the index column with select () and ‘- ‘functions. While we convey that it is not a required column for analysis, we decided to drop it from the dataset while the index column is in the column to show the numbering of the data row.

6.6.1 Standardize Columns and Variables Name

```
#adjust spelling for
#1. work type
#2. smoking status
#3. gender
#4. Residence_type
stroke_ds <- stroke_ds %>% mutate(work_type = ifelse (work_type == 'Govt_job','gov_job',work_type))%>%
                           mutate(work_type = ifelse (work_type == 'Never_worked','never_worked',work_type))%>%
                           mutate(work_type = ifelse (work_type == 'Private','private',work_type))%>%
                           mutate(work_type = ifelse (work_type == 'Self-employed','self_employed',work_type))%>%
                           mutate(gender = ifelse (gender == 'Male','male',gender))%>%
                           mutate(gender = ifelse (gender == 'Female','female',gender))%>%
                           mutate(smoking_status = ifelse (smoking_status == 'Unknown','unknown', smoking_status))%>%
                           mutate(Residence_type = ifelse (Residence_type == 'Unknown','unknown', Residence_type))%>%
                           mutate(Residence_type = ifelse (Residence_type == 'Urban','urban', Residence_type))%>%
                           mutate(Residence_type = ifelse (Residence_type == 'Rural','rural', Residence_type))
# Change the column name
colnames(stroke_ds)[colnames(stroke_ds)=="Residence_type"] <- "residence_type"
```

Based on the raw dataset we obtain from the resources, there is a few columns, and variables' name are not standardized with others or required. According to our situation, we implemented the mutate and if else functions to rename. For example, rename for work type ‘Private’ to ‘private’, as required for all the variable's names, it is recommended to be all in lowercase. Variables of work type, gender, smoking_status, residence type, and column name residence type had all been involved in rename process for Tidier.

6.6.2 Label Encoding

Label encoding is a method for transforming categorical data into numerical form by giving each category an individual integer value (e.g., 0, 1). By optimizing the encoding of categorical features, it helps optimal data processing and modeling in machine learning methods that require quantitative input data.

```
#Encoding ----
stroke_ds$ever_married = factor(stroke_ds$ever_married,
                                levels = c('No', 'Yes'),
                                labels = c(0, 1))

stroke_ds$gender = factor(stroke_ds$gender,
                          levels = c('female', 'male'),
                          labels = c(0, 1))

stroke_ds$smoking_status = factor(stroke_ds$smoking_status,
                          levels = c('unknown', 'never smoked', 'formerly smoked', 'smokes'),
                          labels = c(0, 1, 2, 3))
```

To implement the label encoding for the three appropriate variables in our dataset stroke_ds, we put the factor () function in place as well as levels () and labels () syntax. All three variables have been assigned into numerical values such as 0 and 1. Below is the result indicating.

	gender	age	hypertension	heart_disease	ever_married	work_type	residence_type	avg_glucose_level	bmi	smoking_status
1	1	67	0	1	1	private	urban	228.69	36.60000	2
2	0	61	0	0	1	self_employed	rural	202.21	28.89324	1
3	1	80	0	1	1	private	rural	105.92	32.50000	1
4	0	49	0	0	1	private	urban	171.23	34.40000	3
5	0	79	1	0	1	self_employed	rural	174.12	24.00000	1
6	1	81	0	0	1	private	urban	186.21	29.00000	2
7	1	74	1	1	1	private	rural	70.09	27.40000	1
8	0	69	0	0	0	private	urban	94.39	22.80000	1
9	0	59	0	0	1	private	rural	76.15	28.89324	0
10	0	78	0	0	1	private	urban	58.57	24.20000	0

6.6.3 One-Hot Encoding

```
#---------------------------------- One-Hot Encoding ----------------------------------
library(remotes)
library(fastDummies)
library(caret)

# 1. One-hot encoding -work_type
work_type_dummy <- dummyVars(" ~ work_type", data = stroke_ds, sep = "_")
work_type_encoded <- predict(work_type_dummy, stroke_ds)
# Add the encoded columns to original dataset
stroke_ds <- cbind(stroke_ds, work_type_encoded)

# 2. One-hot encoding -residence_type
residence_type_dummy <- dummyVars(" ~ residence_type", data = stroke_ds, sep = "_")
residence_type_encoded <- predict(residence_type_dummy, stroke_ds)
# Add the encoded columns to original dataset
stroke_ds <- cbind(stroke_ds, residence_type_encoded)

# remove work_type and resident_type original raw column
stroke_ds <- select(stroke_ds, -work_type)
stroke_ds <- select(stroke_ds, -residence_type)
```

One-hot encoding process transforms categorical data into a binary representation, with a binary vector of zeros and ones representing each category. It enables appropriate data processing and modeling by maintaining the uniqueness of categorical features while eliminating numerical correlations between categories. To achieve the one-hot encoding, we execute the dummyVars, predict and cbind function meanwhile also drop off the raw column.

work_typechildren	work_typegov_job	work_typenever_worked	work_typeprivate	work_typeself_employed	residence_typerural	residence_typeurban
0	0	0	1	0	0	1
0	0	0	0	1	1	0
0	0	0	1	0	1	0
0	0	0	1	0	0	1
0	0	0	0	1	1	0

6.6.4 Categorize and Encoding Variable

```
#------------------------------ Categorize + (Label Encoding/One-Hot Encoding) -------------------------------

#1. Categorize age status
stroke_ds <- stroke_ds %>% mutate(age_status = ifelse(age >=0 & age <= 2, "babies/toodler",
                                      ifelse(age <= 12, "children",
                                      ifelse(age <= 17, "adolescent",
                                      ifelse(age <=30, "young adult",
                                      ifelse(age <=64, "middle-aged adult",
                                      ifelse(age >64, "older adult", "older adult")))))))
#Label Encoding age status
stroke_ds$age_status = factor(stroke_ds$age_status,
                      levels = c('babies/toodler', 'children', 'adolescent', 'young adult', 'middle-aged adult', 'older adult'),
                      labels = c(0, 1, 2, 3, 4, 5))

#2. Categorize avg_glucose_level_status
stroke_ds <- stroke_ds %>% mutate(avg_glucose_level_status = ifelse(avg_glucose_level <=79, "hypoglycemia",
                                                             ifelse(avg_glucose_level <= 99, "normal level",
                                                             ifelse(avg_glucose_level < 125, "pre diabetic",
                                                             ifelse(avg_glucose_level >=125, "diabetic", "diabetic")))))

#One-Hot Encoding avg_glucose_level_status
stroke_ds$avg_glucose_level_status = factor(stroke_ds$avg_glucose_level_status,
                         levels = c('hypoglycemia', 'normal level', 'pre diabetic', 'diabetic'),
                         labels = c(0, 1, 2, 3))
```

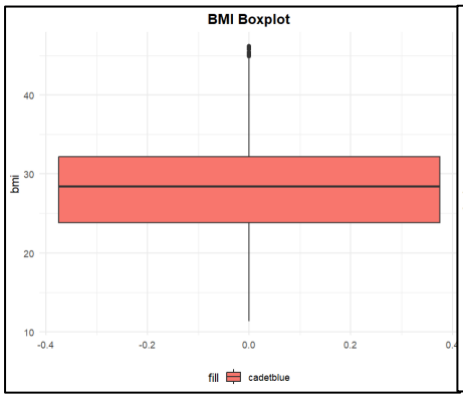
```
#---- BMI Encoding ----
#1. Categorize bmi_status
stroke_ds <- stroke_ds %>% mutate(bmi_status = ifelse(bmi <18.5, "underweight",
                                               ifelse(bmi <= 24.9, "normal weight",
                                               ifelse(bmi <= 29.9, "overweight",
                                               ifelse(bmi >= 30, "obese", "obese")))))
#Label Encoding bmi_status
stroke_ds$bmi_status = factor(stroke_ds$bmi_status,
                              levels = c('underweight', 'normal weight', 'overweight', 'obese'),
                              labels = c(0, 1, 2, 3))
```

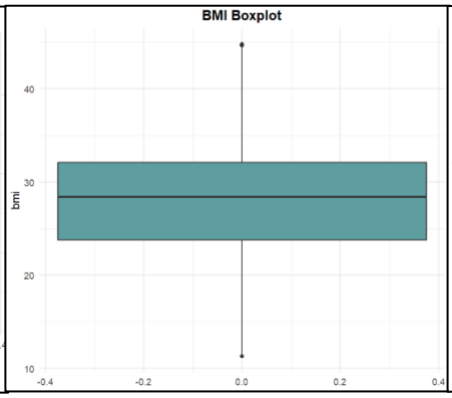
age_status	avg_glucose_level_status
5	3
4	3
5	2
4	3

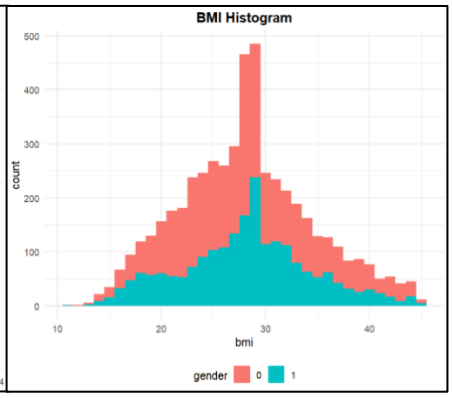
According to the two variables age, bmi, and average glucose level, it is a numerical value, for developing the modeling, we execute it with categorize it in an official range and turn the range into a numeric value with label encoding, hence the result will show within 0-5 or else. Each of the numeric numbers will represent a different range of values.

6.6.5 Outliers Detection and Replacement

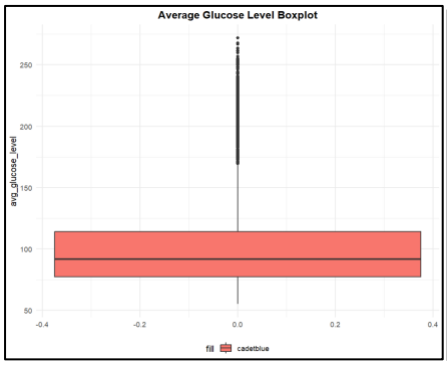
Data points identified as outliers differ significantly from other observations in a dataset due to measurement error, random variation, or other causes. Through data exploration, visualization, etc. like trimming, winsorization, or removal, they can be recognized and dealt with as they will cause a major impact on the outcomes of analysis and modeling.

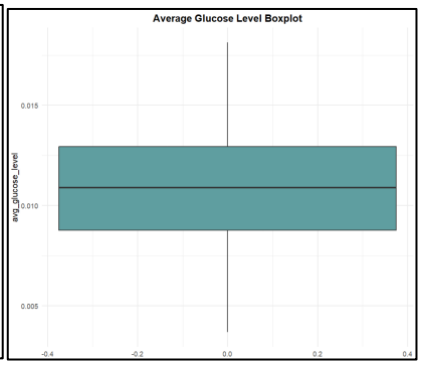


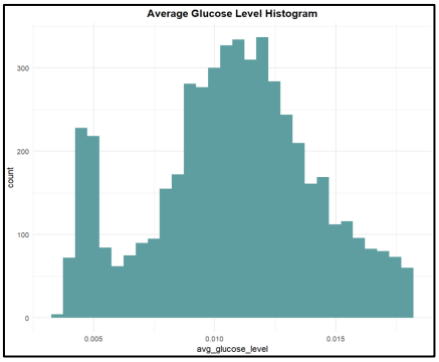





```
> quantile(stroke_ds$bmi)# quantile checking
  0%  25%  50%  75% 100%
11.3 23.8 28.4 32.2 46.2
```







According to the histogram and box plot above, it indicates the BMI and average glucose level variable in visualization. The boxplot in red color is the pattern before we replace the outlier, and the green color box plot will be the boxplot after replacing the outlier for the variable. On the other hand, the quantile of the variable is also illustrated.

6.6.6 Normalization

The process of normalization rescales numerical data characteristics to a standard scale, usually between 0 and 1. This assist in decreasing bias towards larger features in data analysis and machine learning and enhances algorithm performance.

```
#---------------------------------- normalize ====
normalize <- function(x){
  return (x - min(x)) / ((max(x) - min(x)))
}
stroke_ds <- as.data.frame(lapply(stroke_ds, as.numeric))
stroke_ds <- as.data.frame(lapply(stroke_ds[,], normalize))
quantile(stroke_ds$bmi) # new quantile checking
```

```
> quantile(stroke_ds$bmi)
  0%  25%  50%  75% 100%
 0.0 12.5 17.1 20.8 33.5
```

The normalize () function takes the vector as input and applies the following formula to each value with the first three-row of code from the left figure. Following the normalization had implemented the lapply function and meanwhile set the value to numeric with the as.numeric syntax. Coming we check the variable BMI quantile after normalization.

6.6.7 SMOTE and Splitting Dataset (Train set and Test set)

SMOTE (Synthetic Minority Over-sampling technique) is a data augmentation approach that generates synthetic data for the minority class in order to balance imbalanced datasets. The available dataset is split into training and testing sets to assess the effectiveness of machine learning models.

```
#---------------------------------- SMOTE and splitting====
stroke_ds$stroke = factor(stroke_ds$stroke,
                          levels = c(1, 0),
                          labels = c('Yes', 'No'))
library(caTools)
set.seed(123)
split <- sample.split(stroke_ds$stroke, SplitRatio = 0.7)
strokeTrain <- subset(stroke_ds, split==TRUE)
strokeTest <- subset(stroke_ds, split==FALSE)
dim(strokeTrain)
dim(strokeTest)
```

According to the process above, it will execute for the SMOTE and splitting stage for the dataset. In the first case, we encode the stroke variable and set.seed with value 123. The split dataset ratio falls with a value of 0.7 between the train and test datasets.

Check Bias Sampling

```
> prop.table(table(strokeTrain$stroke))

      Yes        No
0.5151515 0.4848485
> prop.table(table(strokeTest$stroke))

      Yes        No
0.5151515 0.4848485
```

```
> #for training dataset
> table(strokeTrain$stroke)

 Yes   No
2958 2784
> strokeTrain <- SMOTE(stroke~., strokeTrain,perc.over = 1600, perc.under = 100)
> as.data.frame(table(strokeTrain$stroke))
  Var1  Freq
1  Yes 44544
2   No 47328
```

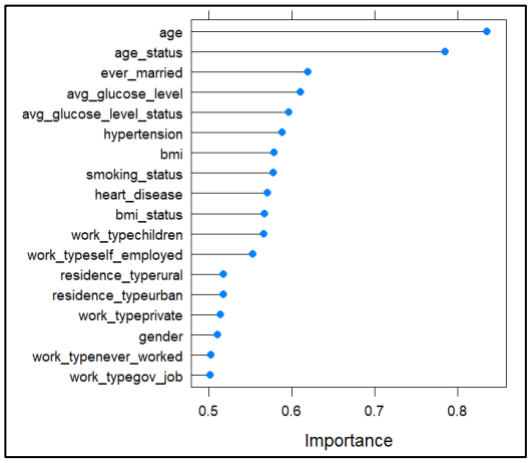
```
>  #for testing dataset
> table(strokeTest$stroke)

 Yes   No
1275 1200
> strokeTest <- SMOTE(stroke~., strokeTest,perc.over = 1600, perc.under = 100)
> as.data.frame(table(strokeTest$stroke))
  Var1  Freq
1  Yes 19200
2   No 20400
```

Balanced data will be essential in the dataset, it can assist in preventing the bias towards the majority class, improve the model's performance and reduce the chances of making an error. Regarding from the proportion and table of unique value 'Yes' and 'No' under variable stroke the train and test dataset had been balanced from the previous stage with the closest value.

6.6.8 Rank Features by Importance

Ranking features by importance is used to find the features in a dataset that have the greatest impact on the output variable. Several feature selection methods including correlation analysis, model-based methods such as Lasso regression and Random Forests. The ranking is useful for determining the most pertinent features in a machine learning model and enhancing its accuracy.

```
> importance <- varImp(model, scale=FALSE)
> print(importance)
ROC curve variable importance

                          Importance
age                           0.8345
age_status                    0.7843
ever_married                  0.6194
avg_glucose_level             0.6107
avg_glucose_level_status      0.5962
hypertension                  0.5881
bmi                           0.5783
smoking_status                0.5775
heart_disease                 0.5708
bmi_status                    0.5673
work_typechildren             0.5665
work_typeself_employed        0.5530
residence_typeurban           0.5179
residence_typerural           0.5179
work_typeprivate              0.5137
gender                        0.5104
work_typenever_worked         0.5023
work_typegov_job              0.5021
> plot(importance)
```



Based on the list and the sorted Cleveland dot plot showing, it is the rank features by importance in relation to the dataset stroke_ds. As we noticed, the highest importance in the dataset is the age variable which holds with the rate of 0.8345 and over the 18 variables overall.

6.7 Brief Summary of Exploratory Analysis and Data Preparation

After all the necessary process from data import, data transformation, and data pre-processing. The dataset has become cleaner, standardized, and balanced and it is ready for the further modeling process. In addition, below is the final attribute in table format of the latest dataset.

No.	Variables Name	Attribute Information
1.	gender	'1' = male, '2' = female
2.	age	age of the patient
3.	hypertension	0 if the patient doesn't have hypertension, 1 if the patient has hypertension.

4.	heart_disease	0 if the patient doesn't have any heart diseases, 1 if the patient has a heart disease.
5.	ever_married	'1' = Yes, '2' = No
6.	avg_glucose_level	average glucose level in blood in float
7.	bmi	body mass index
8.	smoking_status	"0 = unknown", "1 = formerly smoked", "2 = never smoked", or "3 = smokes"
9.	stroke	1 if the patient had a stroke or 0 if not.
10.	work_typechildren	One-Hot encoding: '1' = Yes or '0' = Not work type of take care children in home
11.	work_typegov_job	One-Hot encoding: '1' = Yes or '0' = Not work type of government job
12.	work_typenever_worked	One-Hot encoding: '1' = Yes or '0' = Not work type of the status never worked
13.	work_typeprivate	One-Hot encoding: '1' = Yes or '0' = Not work type with private agency
14.	work_typeself_employed	One-Hot encoding: '1' = Yes or '0' = Not work type in self-employed
15.	residence_typerural	One-Hot encoding: '1' = Yes or '0' = Not in rural residence type
16.	residence_typeurban	One-Hot encoding: '1' = Yes or '0' = Not in urban residence type
17.	age_status	"babies/toodler = 0", "children = 1", "adolescent = 2", "young adult = 3", "middle-aged adult = 4", "older adult = 5"
18.	avg_glucose_level_status	'hypoglycemia = 0', 'normal level = 1', 'pre diabetic = 2', 'diabetic = 3'
19.	bmi_status	'underweight = 0', 'normal weight = 1', 'overweight = 2', 'obese = 3'

7 Model Building and Evaluation of Results

7.1 KNN

KNN (K-Nearest Neighbor) is a non-parametric algorithm that can be deployed in both classification and regression. It clusters the data point based on its similarity which is counted using Euclidean distance and Manhattan distance most of the time. Therefore, it is also an easy algorithm that requires no parameter tuning.

```
> knn_tune$bestTune
  k
1 1
```

Before model training, we will first tune the model using the training dataset, finding the best value for clustering number k. As a result, we found that the best tuning result occurs when k = 1.

we need to convert all the features to the numeric format, including the target variable 'stroke'. After that, only we will proceed to KNN modeling using k = 1.

```
> cat("MSE_DT: ", mse_1, "\nMAE_DT: ", mae_1, "\nRMSE_DT: ", rmse_1)
MSE_DT:  0.2331313
MAE_DT:  0.2331313
RMSE_DT:  0.4828367
```

```
> print(paste("Accurary = ", round(1-classError,5)))
[1] "Accurary =  0.76687"
```

Then, we check the model performance by looking at its MSE (Mean Square Error), MAE (Mean Absolute Error), and RMSE (Root Mean Square Error). The lower value we get in these indicators, the more accurate the model is. Then, we calculate the accuracy of the model.

ROC plot is an effective classification error metric, showing the prediction result of the classification machine learning algorithm. The closer the score of ROC AUC to 1, the more excellent the modeling is. (Mulani, 2022) (Zach, 2021) Here we also include the model's ROC curve in the appendix. The AUC value of the model is 0.5

Then, we plot the confusion matrix at the same time. As a result, there are 334 True Positives (TP) and 222 False Negatives (FN) predictions.

```
> cat("Precision: ", precision_1, "\nRecall: ", recall_1, "\nF1 score: ", F1_score_1, "\n")
Precision:  0.7195208
Recall:  0.8508333
F1 score:  0.7796869
```

In the end, we check the precision, recall, and F1 score value of our model.

7.2 LightGBM

LightGBM is one of the Gradient Boosting Methods that is based on the leaf-wise decision tree algorithm. LightGBM is superior to other traditional algorithms as it is faster than them and capable

of handling enormous datasets. Hence, it is also not advisable to apply LightGBM to a small dataset since it might cause overfitting to the result. (BANERJEE, 2020)

For our stroke prediction dataset, there are more than 5000 rows of samples, therefore, LightGBM is appropriate to be deployed as a modeling algorithm.

The library will be used in the LightGBM modeling including “R6” and “lightgbm”. Then, we assign the feature data to train_x(train data) and test_x (test data) whereas the label index (stroke) is assigned to train_y and test_y.

By doing this, we establish the dataset special for LightGBM. Before training the model, we have to specify the modeling parameters' details. The parameters of the model are referenced with another prediction modeling that has been done and some suggestions from the LightGBM documentation website. (MUSTAFA GÜRKAN ÇANAKÇI, 2023) (DataTechNotes, 4 C.E.) (Mandot, 2017) Following that, we insert the parameters into the modeling function and get the evaluation result of the LightGBM algorithm in all 5 rounds. In the LightGBM model, we have to test the model parameters for many times and randomly change some values to get the best-tuned model.

```
> cat("MSE: ", mse_gbm, "\nMAE: ", mae_gbm, "\nRMSE: ", rmse_gbm)
MSE:  0.1632323
MAE:  0.1632323
RMSE:  0.4040202
```

After that, we look at the MSE (Mean Square Error), MAE (Mean Absolute Error), and RMSE (Root Mean Square Error) of the model.

```
          Accuracy : 0.8368
            95% CI : (0.8216, 0.8511)
No Information Rate : 0.5883
P-Value [Acc > NIR] : < 2.2e-16

             Kappa : 0.6753
```

Furthermore, by attaching the “pROC” package, we draw out the ROC AUC curve of the model. The ROC score we get in the LightGBM modeling is 0. 84.

```
> cat("Precision: ", precision_gbm, "\nRecall: ", recall_gbm, "\nF1 score: ", F1_score_gbm, "\n")
Precision:  0.7733516
Recall:  0.9383333
F1 score:  0.8478916
```

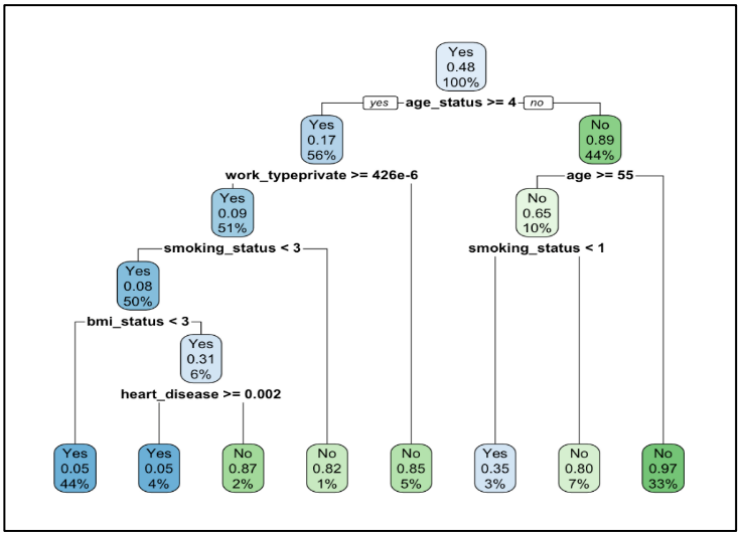
By examining the predicted result, we plot the confusion matrix at the same time. As a result, there are 358 True Positives (TP) and 87 False Negatives (FN) predictions. Through these, we also calculate the value of precision, recall, and F1 score for model comparison later.

Next, we compare the prediction and the actual label in the test dataset to check the distribution of the right and wrong predictions. However, the distribution has shown no trend and we will include the

trend graph in the appendix. The LightGBM has ranked the importance of the features during modeling, here we also check the top 5 features in the dataset, and we found that age, work type, heart disease, and BMI is the key to determining whether someone has a stroke. Furthermore, heart disease and age are also considered as one of the most important elements to determine individuals whether have a high risk of getting stroke.

7.3 Decision Tree

Decision Tree is a non-parametric algorithm that can be used for both classification and regression. It is designed in a hierarchical tree structure with root nodes, and leaves.



Here, we train the decision tree model by using the rpart() function. Then, we plot the decision tree generated.

```
> table_mat
     predict
       Yes   No
  Yes  873  402
  No    93 1107
```

```
> print(paste('Accuracy for test :', accuracy))
[1] "Accuracy for test : 0.8"
```

Then, by drawing the confusion matrix of the prediction made based on the decision tree prediction, we can determine the accuracy of the prediction.

```
> cat("MSE_DT: ", mse_dt, "\nMAE_DT: ", mae_dt, "\nRMSE_DT: ", rmse_dt)
MSE_DT:  0.2
MAE_DT:  0.2
RMSE_DT:  0.4472136
```

Next, similarly, we calculate the indicators we are using to evaluate the model performance, including MAE, MSE, RMSE, precision, recall, ROC curve, and F1 score.

```
> cat("Precision: ", precision_dt, "\nRecall: ", recall_dt, "\nF1 score: ", F1_score_dt, "\n")
Precision:  0.7335984
Recall:  0.9225
F1 score:  0.8172757
```

7.4 SVM

```
> cat("MSE_SVM: ", mse_svm, "\nMAE_SVM: ", mae_svm, "\nRMSE_SVM: ", rmse_svm)
MSE_SVM:  0.2719192
MAE_SVM:  0.2719192
RMSE_SVM:  0.5214587
```

SVM (Support Vector Machine algorithm) is a typical classification deep learning algorithm that classifies the data samples via optimal hyperplane.

```
> cat("Precision: ", precision_svm, "\nRecall: ", recall_svm, "\nF1 score: ", F1_score_svm, "\n")
Precision:  0.7060203
Recall:  0.7525
F1 score:  0.7285196
```

Here, we train the basic SVM model. Then, by using the prediction result, calculate the indicators for comparison later. Then, we plot the confusion matrix and the ROC curve.

7.5 Random Forest

Random Forest is an algorithm that is developed based on the ensemble method of multiple decision trees. It is helpful when we need to plot out the importance ranking of the features.

As mentioned above, we plot the model-building error and the feature's importance. We leave the feature importance graph in the appendix section of this document.

```
          Accuracy : 0.8642
            95% CI : (0.8501, 0.8775)
 No Information Rate : 0.6085
 P-Value [Acc > NIR] : < 2.2e-16

              Kappa : 0.7303
```

```
> cat("MSE_rf: ", mse_rf, "\nMAE_rf: ", mae_rf, "\nRMSE_rf: ", rmse_rf)
MSE_rf:  1.38303
MAE_rf:  1.123636
RMSE_rf:  1.176023
```

```
> cat("Precision: ", precision_rf, "\nRecall: ", recall_rf, "\nF1 score: ", F1_score_rf, "\n")
Precision:  0.7458824
Recall:  0.9865145
F1 score:  0.8494864
```

Then, we calculate the accuracy, MAE, etc. of this algorithm to be included in the modeling comparison later. After that, we draw the confusion matrix of the Random Forest model.

7.6 Logistic regression

Logistic regression is an algorithm to identify the relationship between the dependent variable (dependent variable) and the independent variable (explanatory variable).

In logistic regression, we need to come up with a model first, make a prediction and check on the accuracy. Then, the model has to be built for the second time with the best combination of parameters.

```
> print(tuned_model)
Generalized Linear Model

5742 samples
  18 predictor
   2 classes: '0', '1'

No pre-processing
Resampling: Cross-Validated (5 fold)
Summary of sample sizes: 4595, 4594, 4593, 4593, 4593
Resampling results:

  Accuracy   Kappa
  0.8418681  0.6825226
```

Following that, we plot the evaluation metrics and ROC curve of the tuned model. Like the modeling process previously, we will also plot the confusion matrix and check the precision, recall, and F1 score.

```
> cat("MSE_lr: ", mse_lr, "\nMAE_lr: ", mae_lr, "\nRMSE_lr: ", rmse_lr)
MSE_lr:  1.491408
MAE_lr:  1.134169
RMSE_lr:  1.221232
```

```
> cat("Precision: ", precision_lr, "\nRecall: ", recall_lr, "\nF1 score: ", F1_score_lr, "\n")
Precision:  0.7372549
Recall:  0.7004471
F1 score:  0.7183798
```

7.7 Model Comparison

In the comparison of the model, we assign the value array first and plot the comparison bar chart then.

For the **Accuracy** of the model, Random Forest gets the highest accuracy among all of the models.

For **MAE**, Logistic Regression is ranked as the lowest one.

In **MSE**, Logistic Regression also gets the lowest error value.

In terms of the **RMSE**, Logistic Regression has the least value.

For **precision value**, the Random Forest gets the highest mark.

For the **recall value**, the Random Forest model gets the first place in comparison.

For the **F1 score**, Random Forest scored more outstanding than any other model we made.

In the **ROC curve** graph, Random Forest has the curve that is the closest to the top, and the highest AUC score (0.868).

7.7.1 Model Choosing and Enhancement

Since the dataset has been processed with SMOTE technique, the dataset has been balanced. MAE, MSE, and RMSE might not be our top consideration indicators in choosing the model. Consequently, we chose the result according to the accuracy, precision, recall, F1 score, and ROC curve ranking. Random Forest will be our choice in the end.

After that, we will enhance the random forest model by using feature selection and calculating the latest scoring in terms of accuracy, AUC, Precision, etc. Lastly, we implement the prediction model to the shiny framework. Users can predict the probability of having a stroke by entering the required data into the interface. Meanwhile, we also will include the scoring and framework interface image in the appendix section.

8 Conclusion

In a nutshell, after processing with the data pre-processing and up until processing with modeling the outcome has been determined. When passing the analysis, the most suitable and accurate model in our analysis will be **RANDOM FOREST** after comparing with a total of 6 models which include LightGBM, KNN, decision tree, SVM, random forest, and logistic regression. We have identified that age, marital status, heart disease, BMI, and average glucose level are the most important indicators

that determine whether one has a stroke. By implementing the shiny framework, our work enables people to simply enter the data that are commonly known to predict their stroke likelihood.

9 References

Carlos Fernandez-Lozano1, 2. P.-A.-Y.-G.-D.-G.-Y.-R. (2021). Random forest-based prediction of stroke outcome. *scientific reports*, 1-12.

Gangavarapu Sailasya1, G. L. (2021). Analyzing the Performance of Stroke Prediction using ML Classification Algorithms . *International Journal of Advanced Computer Science and Applications*, 1-7.

Leila Amini, R. A. (4 May, 2013). *Prediction and Control of Stroke by Data Mining.* Retrieved from PubMed Cnetral: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3678226/

Band, A. (2020, May 23). *How to find the optimal value of K in KNN?* Towards Data Science. https://towardsdatascience.com/how-to-find-the-optimal-value-of-k-in-knn-35d936e554eb#:~:text=The%20optimal%20K%20value%20usually,be%20aware%20of%20the%20outliers.

BANERJEE, P. (2020, July 21). *LightGBM Classifier in Python | Kaggle*. Kaggle: Your Machine Learning and Data Science Community; Kaggle. https://www.kaggle.com/code/prashant111/lightgbm-classifier-in-python

DataTechNotes. (4 C.E.). *DataTechNotes: LightGBM Regression Example in R*. DataTechNotes. https://www.datatechnotes.com/2022/04/lightgbm-regression-example-in-r.html

Mandot, P. (2017, August 17). *What is LightGBM, How to implement it? How to fine tune the parameters? | by Pushkar Mandot | Medium*. Medium; Medium. https://medium.com/@pushkarmandot/https-medium-com-pushkarmandot-what-is-lightgbm-how-to-implement-it-how-to-fine-tune-the-parameters-60347819b7fc

Mulani, S. (2022, August 3). *Plotting ROC curve in R Programming | DigitalOcean*. DigitalOcean | The Cloud for Builders; DigitalOcean. https://www.digitalocean.com/community/tutorials/plot-roc-curve-r-programming

MUSTAFA GÜRKAN ÇANAKÇI. (2023, March 31). *Prediction with 7-Classification Models | ROC AUC | Kaggle*. Kaggle: Your Machine Learning and Data Science Community; Kaggle. https://www.kaggle.com/code/mechatronixs/prediction-with-7-classification-models-roc-auc

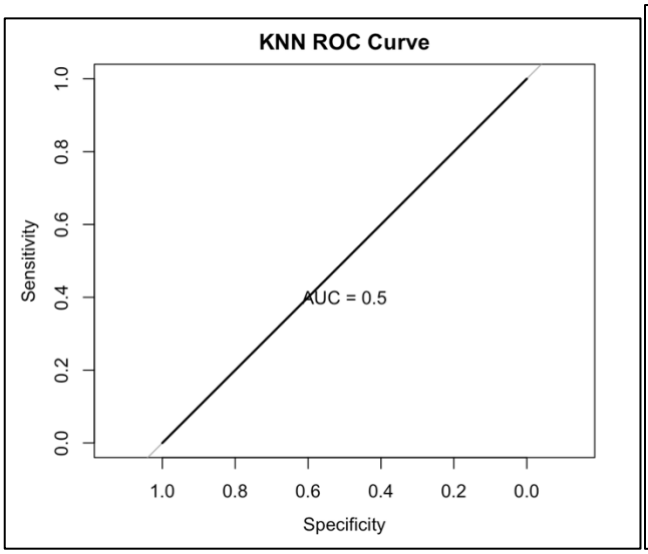
Zach. (2021, August 9). *How to Interpret a ROC Curve (With Examples) - Statology*. Statology. https://www.statology.org/interpret-roc-curve/#:~:text=The%20more%20that%20the%20ROC,1%2C%20the%20better%20the%20model.

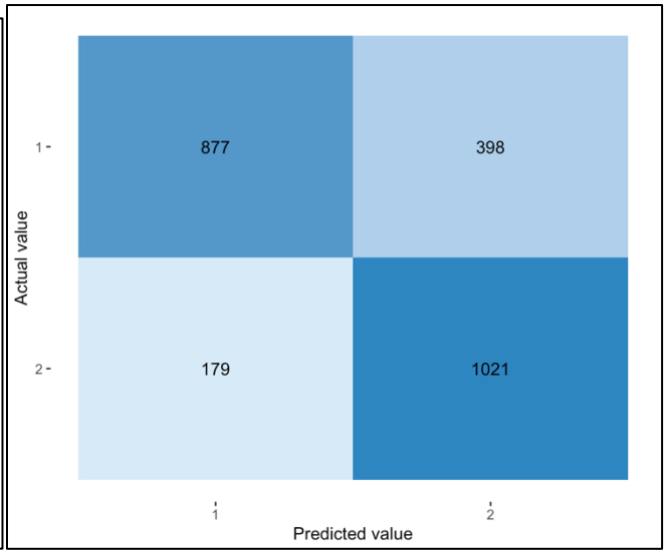
10 Appendix

10.1 KNN

```r
# KNN ====
# install.packages('class')
library(class)

strokeTrainknn <- strokeTrain
strokeTestknn <- strokeTest

# model tuning
control <- trainControl(method = "repeatedcv", number = 10, repeats = 3, search = "grid")
set.seed(123)
grid <- data.frame(k = seq(1, 30, by = 2))
knn_tune <- train(stroke ~ ., data = strokeTrainknn, method = "knn",
                  trControl = control, tuneGrid = grid)
knn_tune$bestTune

strokeTrainknn$stroke = factor(strokeTrain$stroke,
                           levels = c('Yes', 'No'),
                           labels = c(1,0))
strokeTestknn$stroke = factor(strokeTest$stroke,
                            levels = c('Yes', 'No'),
                            labels = c(1,0))
strokeTrainknn <- as.data.frame(lapply(strokeTrainknn, as.numeric))
strokeTestknn <- as.data.frame(lapply(strokeTestknn, as.numeric))

# k = 1
knn_1 <- knn(train=strokeTrainknn, test=strokeTestknn, cl=strokeTrain$stroke,k=1, prob=T)
cm_1 <- confusionMatrix(table(strokeTest$stroke, knn_1))

mse_1 <- mean((strokeTestknn$stroke - as.numeric(knn_1))^2)
mae_1 <- mean(abs(strokeTestknn$stroke - as.numeric(knn_1)))
rmse_1 <- sqrt(mean((strokeTestknn$stroke - as.numeric(knn_1))^2))
cat("MSE_DT: ", mse_1, "\nMAE_DT: ", mae_1, "\nRMSE_DT: ", rmse_1)

classError <- mean(as.factor(as.integer(knn_1)) != strokeTestknn$stroke)
print(paste("Accurary = ", round(1-classError,5)))
```

```r
# install.packages(c("pROC", "ROCR))
library(pROC, ROCR)
prob <- attr(knn_1, "prob")
prob
prob <- 2 * ifelse(knn_1 == "-1", 1- prob, prob) - 1

auc_knn <- roc(strokeTest$stroke, prob)$auc
plot(roc(strokeTest$stroke, prob), main = "KNN ROC Curve")
text(0.5, 0.4, paste("AUC =", round(auc_knn, 3)))

# install.packages("yardstick")
library(yardstick)
cm <- conf_mat(table(strokeTestknn$stroke, as.numeric(knn_1)))
autoplot(cm ,type='heatmap')+
  scale_fill_gradient(low="#D6EAF8",high = "#2E86C1") +
  xlab("Predicted value") + ylab("Actual value")

# Calculate precision, recall and F1 score
TN_1 <- cm_1$table[1,1] # True negatives
FP_1 <- cm_1$table[1,2] # False positives
FN_1 <- cm_1$table[2,1] # False negatives
TP_1 <- cm_1$table[2,2] # True positives

precision_1 <- TP_1 / (TP_1 + FP_1)
recall_1 <- TP_1 / (TP_1 + FN_1)
F1_score_1 <- 2 * precision_1 * recall_1 / (precision_1 + recall_1)

cat("Precision: ", precision_1, "\nRecall: ", recall_1, "\nF1 score: ", F1_score_1, "\n")
```

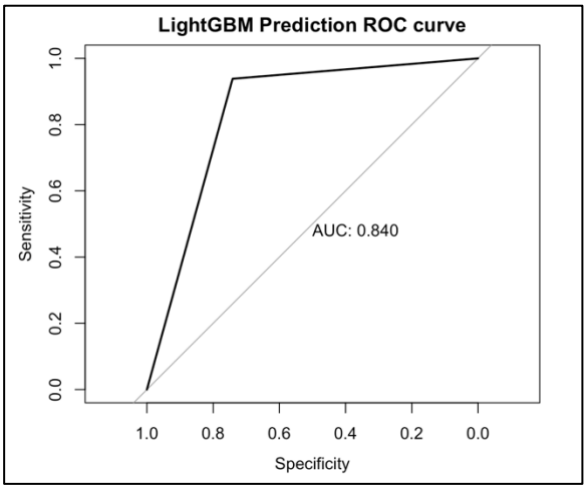

KNN ROC Curve
Sensitivity
1.0
0.8
0.6
0.4
0.2
0.0
AUC = 0.5
1.0
0.8
0.6
0.4
0.2
0.0
Specificity

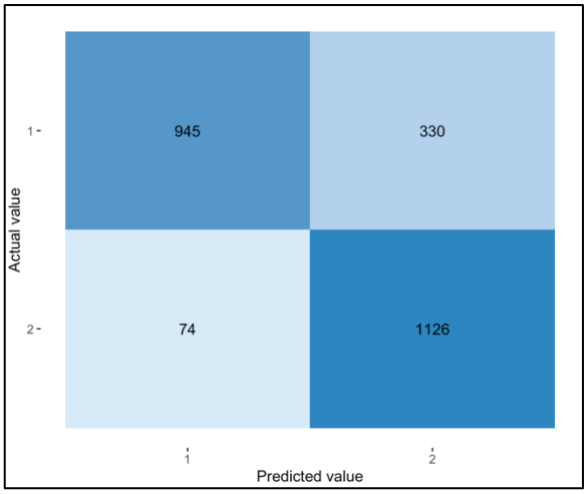

Actual value
1
2
877
398
179
1021
1
2
Predicted value

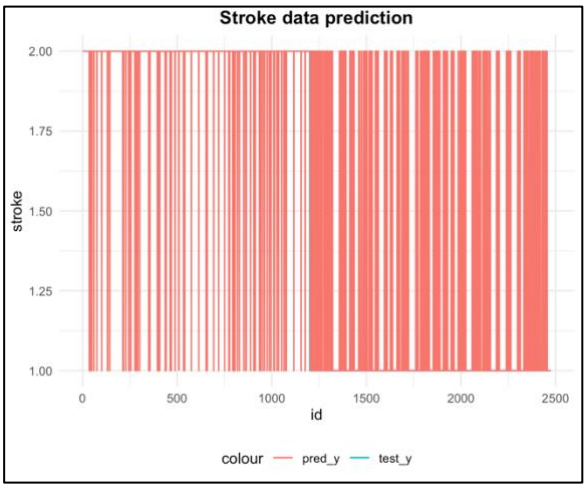
10.2 LightGBM

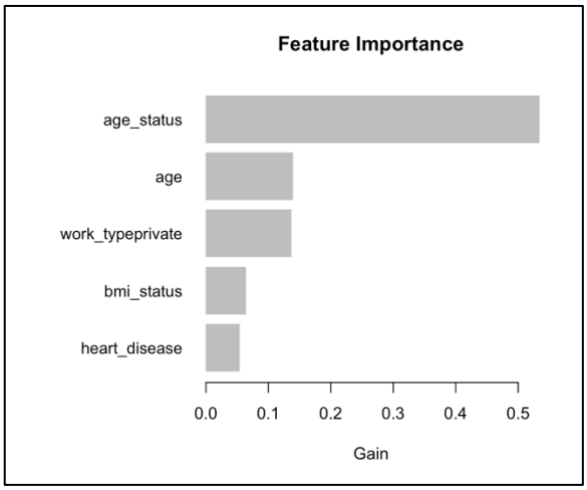
```r
# lightGBM ====
#install.packages("lightgbm")
library(R6)
library(lightgbm)
train_x <- dplyr::select(strokeTrain, -stroke)
train_y <- as.integer(strokeTrain$stroke)

test_x <- dplyr::select(strokeTest, -stroke)
test_y <- as.integer(strokeTest$stroke)

dtrain <- lgb.Dataset(data = as.matrix(train_x), label=train_y)
dtest <- lgb.Dataset.create.valid(dtrain, as.matrix(test_x), label=test_y)

params <- list(objective = "regression", metric = "l2", min_data = 1L,
               learning_rate = .0015, force_row_wise = T,max_depth = 100,min_data_in_lead = 50,
               boosting_type = 'gbdt', bagging_fraction = 0.6, force_row_wise = T,
               unbalance = F, num_leaves = 10, max_bin = 10, num_iterations = 100,
               feature_fraction = 0.8)
set.seed(123)
valids <- list(test = dtest)
model <- lgb.train(params = params, data = dtrain, nrounds = 5L, valids = valids)
lgb.get.eval.result(model, "test", "l2")

pred_y <- predict(model, as.matrix(test_x), reshape = T)
pred_y <- as.numeric(round(pred_y, 0))

mse_gbm <- mean((test_y - pred_y)^2)
mae_gbm <- caret::MAE(test_y, pred_y)
rmse_gbm <- caret::RMSE(test_y, pred_y)
cat("MSE: ", mse_gbm, "\nMAE: ", mae_gbm, "\nRMSE: ", rmse_gbm)

# install.packages("pROC")
library(pROC)
gbm_auc <- roc(test_y, pred_y)
gbm_auc$auc
plot(gbm_auc, main = "LightGBM Prediction ROC curve", print.auc = T)
```

```r
cm_gbm <- confusionMatrix(as.factor(test_y), as.factor(pred_y))
cm_gbm

library(yardstick)
cm <- conf_mat(table(test_y, pred_y))
autoplot(cm ,type='heatmap')+
  scale_fill_gradient(low="#D6EAF8",high = "#2E86C1") +
  xlab("Predicted value") + ylab("Actual value")

df <- data.frame(test_y, pred_y)
df$id <- 1:nrow(df)

ggplot() + geom_line(data = df, aes(x = id, y = test_y, color = 'test_y')) +
  geom_line(data = df, aes(x=id, y = pred_y, color = 'pred_y')) +
  ggtitle("Stroke data prediction") + theme_minimal() + ylab('stroke') +
  theme(legend.position = "bottom", plot.title = element_text(hjust=0.5, face = "bold"))

tree_imp <- lgb.importance(model, percentage = TRUE)
lgb.plot.importance(tree_imp, top_n = 5L, measure = "Gain")

# Calculate precision, recall and F1 score
TN_gbm <- cm_gbm$table[1,1] # True negatives
FP_gbm <- cm_gbm$table[1,2] # False positives
FN_gbm <- cm_gbm$table[2,1] # False negatives
TP_gbm <- cm_gbm$table[2,2] # True positives

precision_gbm <- TP_gbm / (TP_gbm + FP_gbm)
recall_gbm <- TP_gbm / (TP_gbm + FN_gbm)
F1_score_gbm <- 2 * precision_gbm * recall_gbm / (precision_gbm + recall_gbm)

cat("Precision: ", precision_gbm, "\nRecall: ", recall_gbm, "\nF1 score: ", F1_score_gbm, "\n")
```







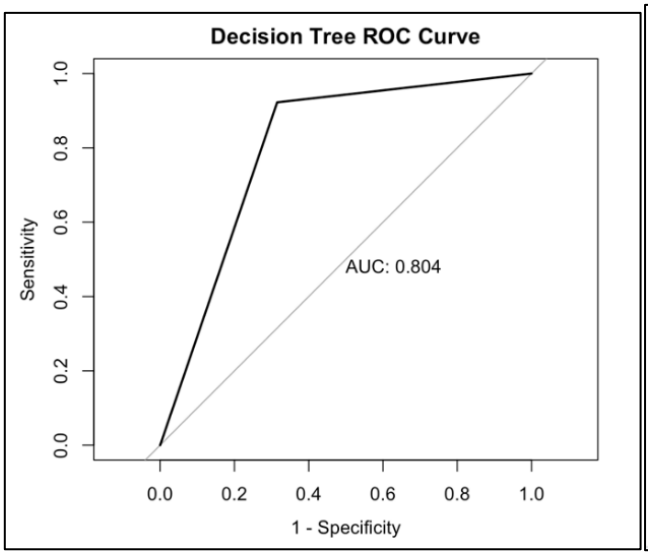


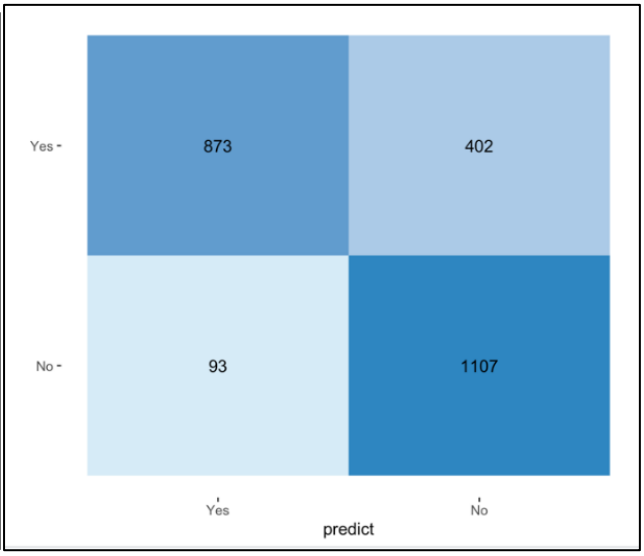
10.3 Decision Tree

```
#Decison tree ====
#install.packages('rpart')
#install.packages
predict(object, ...)
library(rpart)
library(rpart.plot)

testing <- rpart(stroke ~.,data=strokeTrain,method = 'class') #build decison tree model
rpart.plot(testing)  #plot decision tree
predict <- predict(testing,strokeTest,type='class') #prediction
```

```r
predicted_probs<-as.numeric(predict)   #output in prob for ROC plot purpose
predict_value<-predict(testing,strokeTest,type='vector')
#output in vector for finding MSE, MAE, RMSE purpose

table_mat <- table(strokeTest$stroke, predict) #show confusion matrix table
table_mat

accuracy<-sum(diag(table_mat)/sum(table_mat)) # show accurracy
print(paste('Accuracy for test :', accuracy))

cm_dt <- confusionMatrix(table(strokeTest$stroke, predict)) #show confusion matrix table
cm_dt  #accuracy

# calculate MSE MAE RMSE
x <- as.numeric(strokeTest$stroke) #change target variable from yes/no to 1/2
mse_dt <- mean((x - predict_value)^2)
mae_dt <- mean(abs(x - predict_value))
rmse_dt <- sqrt(mean((x - predict_value)^2))
compare_dt <- c(mse_dt, mae_dt, rmse_dt)
cat("MSE_DT: ", mse_dt, "\nMAE_DT: ", mae_dt, "\nRMSE_DT: ", rmse_dt)
```

```r
library(pROC)
roc_obj <- roc(strokeTest$stroke,predicted_probs ) #AUC/ROC score
plot(roc_obj, main = "Decision Tree ROC Curve", print.auc = TRUE, legacy.axes = TRUE) #plot ROC

cm <- conf_mat(table(strokeTest$stroke, predict))
autoplot(cm ,type='heatmap')+
  scale_fill_gradient(low="#D6EAF8",high = "#2E86C1")

# Calculate precision, recall and F1 score
TN_dt <- cm_dt$table[1,1] # True negatives
FP_dt <- cm_dt$table[1,2] # False positives
FN_dt <- cm_dt$table[2,1] # False negatives
TP_dt <- cm_dt$table[2,2] # True positives

precision_dt <- TP_dt / (TP_dt + FP_dt)
recall_dt <- TP_dt / (TP_dt + FN_dt)
F1_score_dt <- 2 * precision_dt * recall_dt / (precision_dt + recall_dt)

cat("Precision: ", precision_dt, "\nRecall: ", recall_dt, "\nF1 score: ", F1_score_dt, "\n")
```



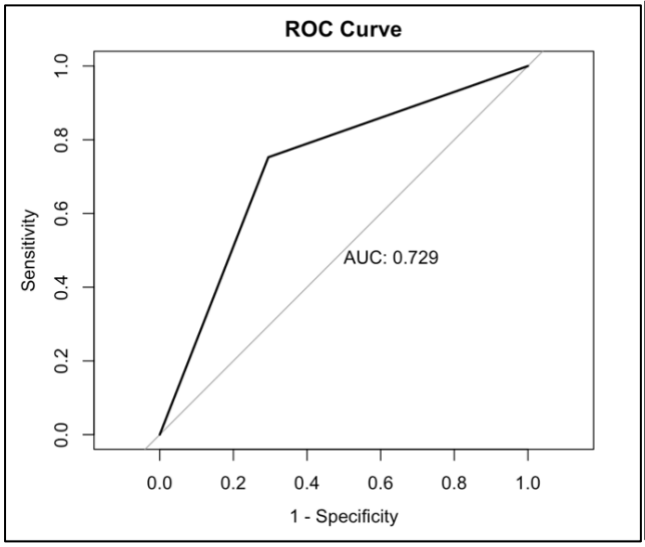


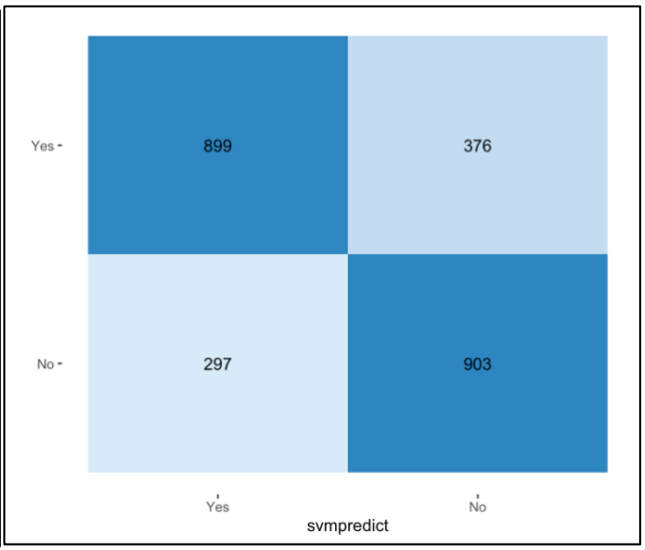
10.4 SVM

```r
#SVM ====
#install.packages('e1071')
#install.packages('yardstick')
library(e1071)
library(yardstick)

svmtrain <- svm(formula = stroke ~ age, #build svm model
                data = strokeTrain,
                type = 'C-classification',
                kernel = 'linear')

svmpredict <- predict(svmtrain, newdata = strokeTest) #predict
svmpredict_value<-as.numeric(svmpredict)
#convert output to numeric for finding MSE,MAE,RMSE purpose

cm_svm <- confusionMatrix(table(strokeTest$stroke, svmpredict))
#use conf_mat from yardstick library to visualize
cm_svm

x<- as.numeric(strokeTest$stroke)
mse_svm <- mean((x - svmpredict_value)^2)
mae_svm<- mean(abs(x - svmpredict_value))
rmse_svm <- sqrt(mean((x - svmpredict_value)^2))
cat("MSE_SVM: ", mse_svm, "\nMAE_SVM: ", mae_svm, "\nRMSE_SVM: ", rmse_svm)
```

```r
library(pROC)
roc_obj <- roc(strokeTest$stroke,svmpredict_value ) #AUC/ROC score
plot(roc_obj, main = "ROC Curve", print.auc = TRUE, legacy.axes = TRUE) #plot ROC

cm <- conf_mat(table(strokeTest$stroke, svmpredict))
autoplot(cm ,type='heatmap')+
  scale_fill_gradient(low="#D6EAF8",high = "#2E86C1")

# Calculate precision, recall and F1 score
TN_svm <- cm_svm$table[1,1] # True negatives
FP_svm <- cm_svm$table[1,2] # False positives
FN_svm <- cm_svm$table[2,1] # False negatives
TP_svm <- cm_svm$table[2,2] # True positives

precision_svm <- TP_svm / (TP_svm + FP_svm)
recall_svm <- TP_svm / (TP_svm + FN_svm)
F1_score_svm <- 2 * precision_svm * recall_svm / (precision_svm + recall_svm)

cat("Precision: ", precision_svm, "\nRecall: ", recall_svm, "\nF1 score: ", F1_score_svm, "\n")
```

```r
library(pROC)
roc_obj <- roc(strokeTest$stroke,svmpredict_value ) #AUC/ROC score
plot(roc_obj, main = "ROC Curve", print.auc = TRUE, legacy.axes = TRUE) #plot ROC

cm <- conf_mat(table(strokeTest$stroke, svmpredict))
autoplot(cm ,type='heatmap')+
  scale_fill_gradient(low="#D6EAF8",high = "#2E86C1")

# Calculate precision, recall and F1 score
TN_svm <- cm_svm$table[1,1] # True negatives
FP_svm <- cm_svm$table[1,2] # False positives
FN_svm <- cm_svm$table[2,1] # False negatives
TP_svm <- cm_svm$table[2,2] # True positives

precision_svm <- TP_svm / (TP_svm + FP_svm)
recall_svm <- TP_svm / (TP_svm + FN_svm)
F1_score_svm <- 2 * precision_svm * recall_svm / (precision_svm + recall_svm)

cat("Precision: ", precision_svm, "\nRecall: ", recall_svm, "\nF1 score: ", F1_score_svm, "\n")
```


ROC Curve
Sensitivity
1 - Specificity
AUC: 0.729
0.0
0.2
0.4
0.6
0.8
1.0

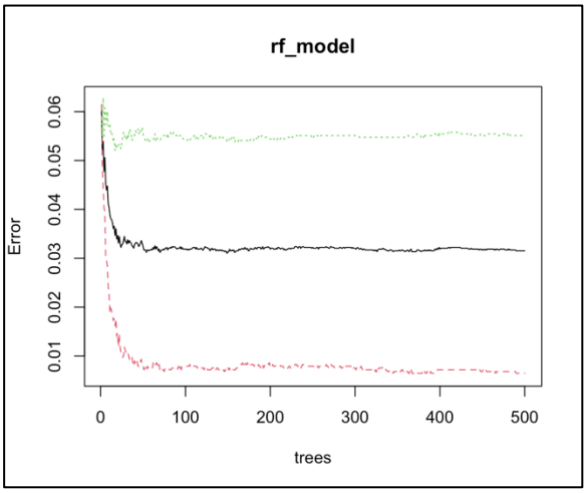

Yes
No
899
376
297
903
Yes
No
svmpredict

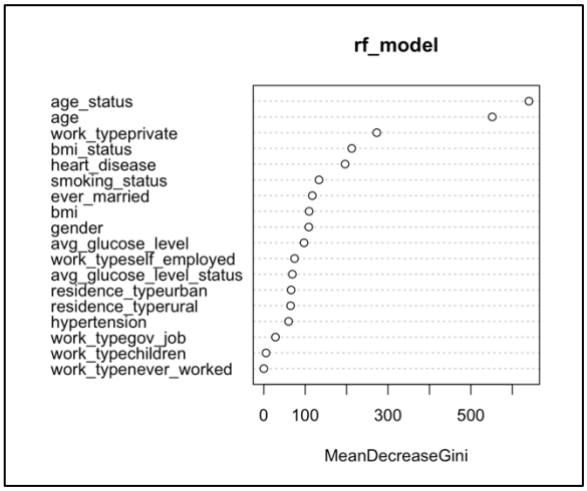
10.5 Random Forest

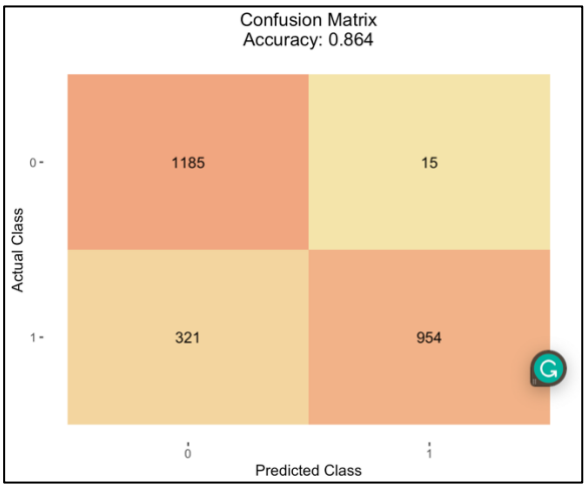
```r
# Random Forest====
# install.packages("randomForest")
library(randomForest)
library(e1071)
library(yardstick)

#Train the random forest model
rf_model <- randomForest(stroke ~ ., data = strokeTrain)
plot(rf_model)
print(rf_model)
varImpPlot(rf_model)
var_importance <- importance(rf_model)
print(var_importance)
sorted_importance <- sort(var_importance, decreasing = TRUE)
print(sorted_importance)

for (i in seq_along(sorted_importance)) {
  var_name <- names(strokeTrain)[i]
  importance_value <- sorted_importance[i]
  print(paste(var_name, ":", importance_value))
}

#Make predictions on test set
rf_preds <- predict(rf_model,strokeTest)
table_mat <-confusionMatrix(table(strokeTest$stroke, rf_preds))
table_mat

# Evaluate the performance for model
# Example of using AUC as a metric
library(pROC)
rf_preds <- as.numeric(as.character(rf_preds))
rf_preds <- ordered(rf_preds, levels = c("0", "1"))
print(rf_preds)

table(strokeTest$stroke)

rf_auc <- auc(strokeTest$stroke, rf_preds)
str(rf_auc)
```

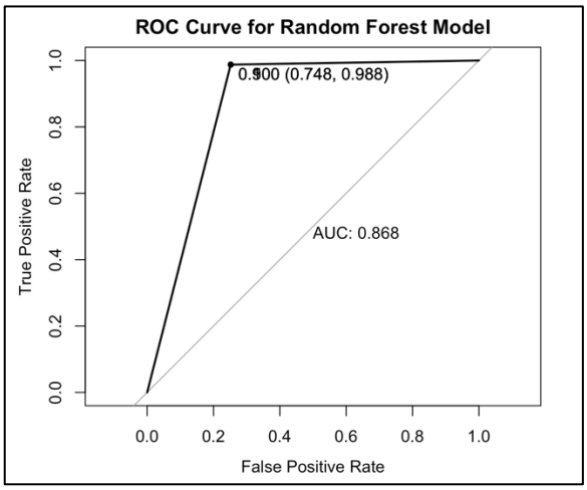
```r
# Evaluate the accuracy of the random forest model
# Convert rf_preds to numeric
rf_preds <- as.numeric(as.character(rf_preds))
# Evaluate the accuracy of the random forest model
rf_acc <- sum(rf_preds == strokeTest$stroke) / nrow(strokeTest)
print(rf_acc)

# Calculate MSE, MAE, RMSE -----
test_rf <- as.integer(strokeTest$stroke)
mse_rf <- mean((test_rf - rf_preds)^2)
mae_rf <- mean(abs(test_rf - rf_preds))
rmse_rf <- sqrt(mean((test_rf - rf_preds)^2))
cat("MSE_rf: ", mse_rf, "\nMAE_rf: ", mae_rf, "\nRMSE_rf: ", rmse_rf)

# Create confusion matrix
cm_rf <- confusionMatrix(as.factor(rf_preds), strokeTest$stroke)
print(cm_rf)

# Visualize CM and ROC -----
library(ggplot2)
library(ROCR)
# Confusion Matrix plot
cmv_rf <- conf_mat(table(strokeTest$stroke, rf_preds))
autoplot(cmv_rf, type = 'heatmap', label = TRUE) +
  scale_fill_gradient(low = "#F3E5AB", high = "#F1A680") +
  labs(title = paste("Confusion Matrix\nAccuracy:", round(cm_rf$overall["Accuracy"], 3)),
       x = "Predicted Class", y = "Actual Class")+
  theme(plot.title = element_text(hjust = 0.5))

# ROC curve plot
roc_rf <- roc(strokeTest$stroke, rf_preds, levels = rev(levels(strokeTest$stroke)))
plot(roc_rf, print.auc = TRUE, legacy.axes = TRUE,
     main = "ROC Curve for Random Forest Model",
     xlab = "False Positive Rate", ylab = "True Positive Rate",
     print.thres = c(0.1, 0.5, 0.9))
```

```r
# Calculate precision, recall and F1 score
TN_rf <- cm_rf$table[1,1] # True negatives
FP_rf <- cm_rf$table[1,2] # False positives
FN_rf <- cm_rf$table[2,1] # False negatives
TP_rf <- cm_rf$table[2,2] # True positives

precision_rf <- TP_rf / (TP_rf + FP_rf)
recall_rf <- TP_rf / (TP_rf + FN_rf)
F1_score_rf <- 2 * precision_rf * recall_rf / (precision_rf + recall_rf)

cat("Precision: ", precision_rf, "\nRecall: ", recall_rf, "\nF1 score: ", F1_score_rf, "\n")
```





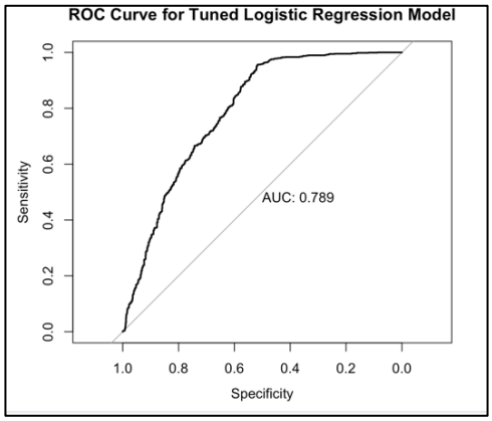


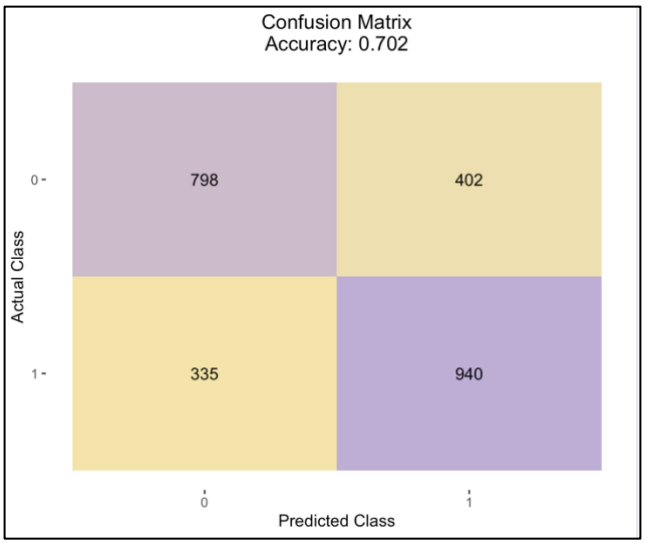


10.6 Logistic Regression

```r
# Logistic Regression ====

# install.packages("MASS")
# install.packages("caret")
library(caret)
library(MASS)
library(e1071)
library(yardstick)
#Fit a logistic regression model:
model <- glm(stroke~., data = strokeTrain, family = binomial())
summary(model)
#Make Predictions on test set + evaluate
probabilities <- predict(model, newdata = strokeTest, type = "response")
predicted_classes <- ifelse(probabilities > 0.25, 1, 0)

cm_lr <- confusionMatrix(as.factor(as.matrix(predicted_classes)), strokeTest$stroke)
cm_lr_metrics <- cm_lr$byClass
#Tune the model by selecting the best combination of hyperparameters:
control <- trainControl(method="cv", number=5, verboseIter = FALSE)
tuned_model <- train(stroke ~ ., data = strokeTrain,
                     method = "glm",
                     family = "binomial",
                     trControl = control,
                     tuneLength = 10)
print(tuned_model)
```

```r
#---- Calculate MSE, MAE, RMSE
test_lr <- as.integer(strokeTest$stroke)
mse_lr <- mean((test_lr - probabilities)^2)
mae_lr <- mean(abs(test_lr - probabilities))
rmse_lr <- sqrt(mean((test_lr - probabilities)^2))
cat("MSE_lr: ", mse_lr, "\nMAE_lr: ", mae_lr, "\nRMSE_lr: ", rmse_lr)

# Visualize CM and ROC ---- Logistic Regression
library(ggplot2)
library(ROCR)

# ROC curve plot
roc_lr <- roc(strokeTest$stroke, probabilities)
# Plot ROC curve and calculate AUC
plot(roc_lr, main = "ROC Curve for Tuned Logistic Regression Model\n",
     print.auc = TRUE, col = "black")
```

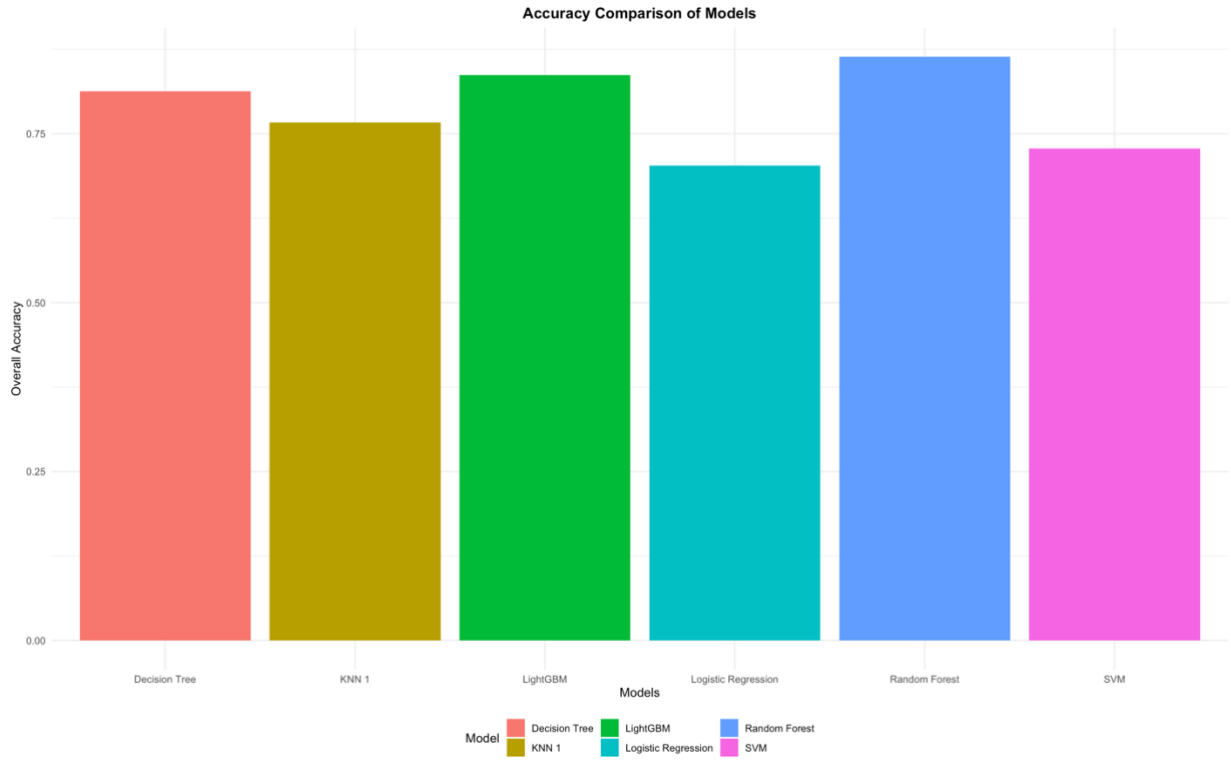


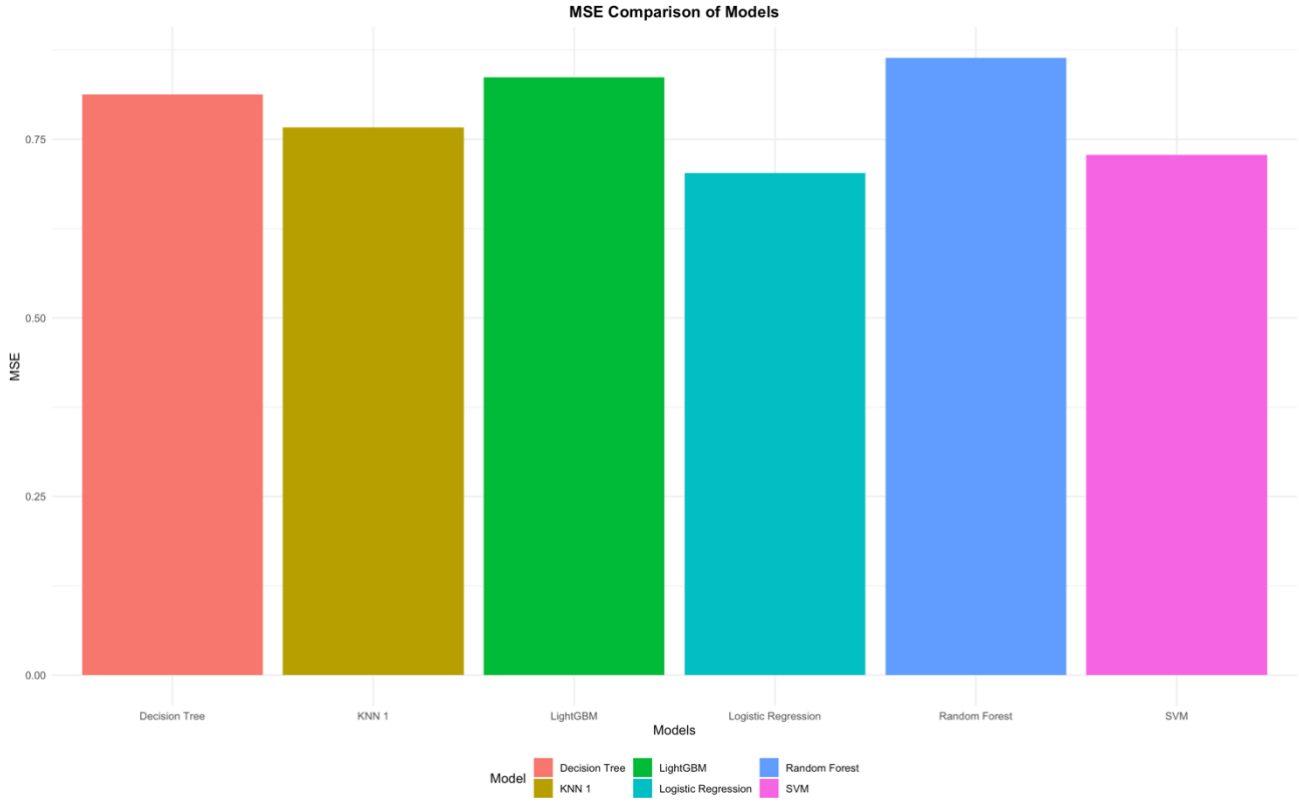


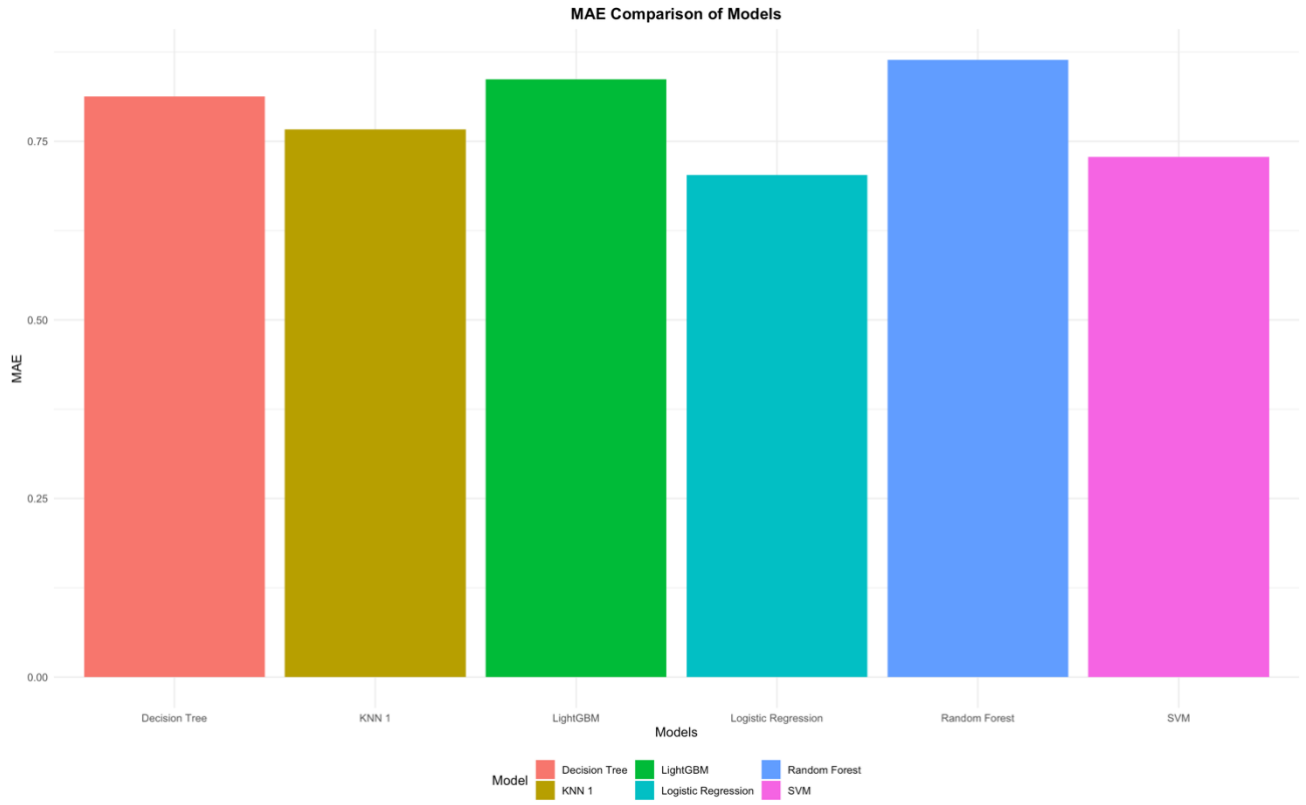
10.7 Model Comparison

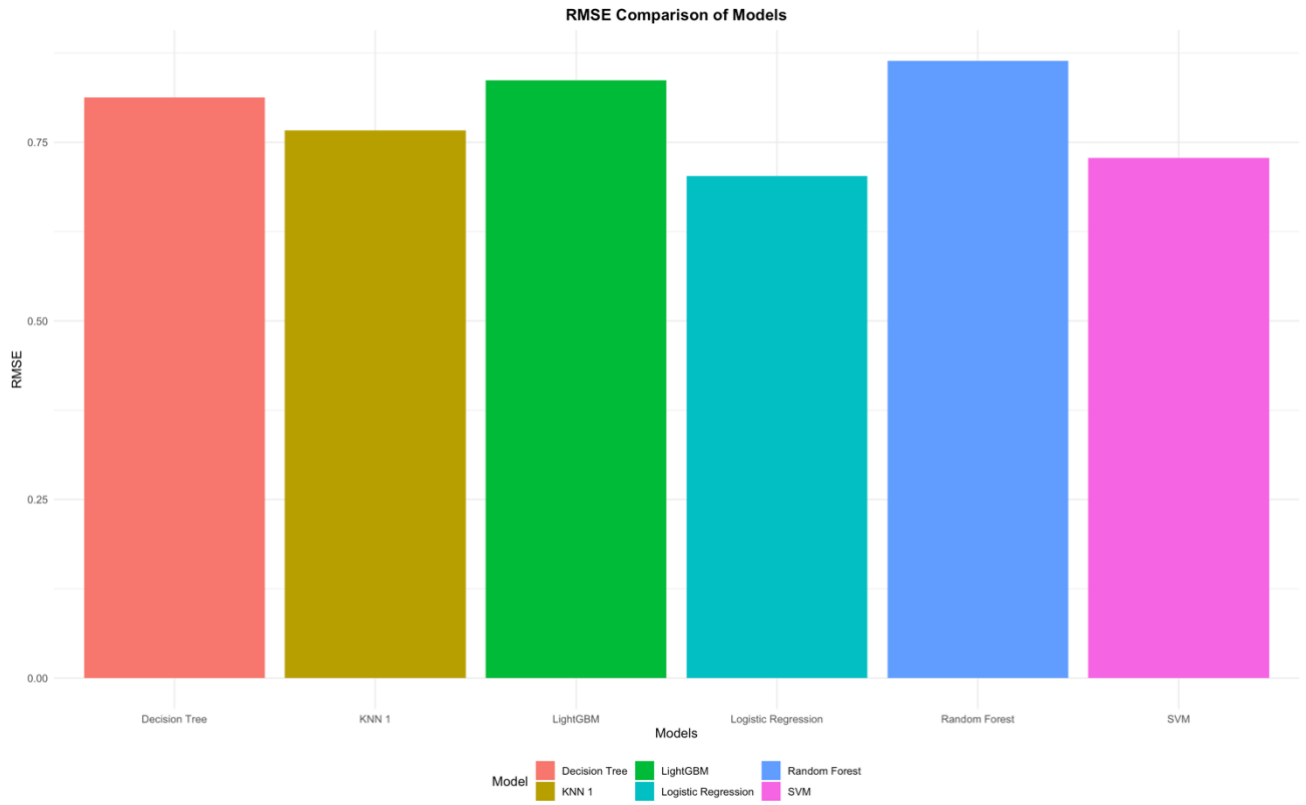
```
model_compare <- data.frame(Model, Precision)
ggplot(aes(x=Model, y=Accuracy, fill=Model), data=model_compare) +
  geom_bar(stat='identity') + ggtitle('Precision Comparison of Models') +
  xlab('Models') + ylab('Precision') + theme_minimal() +
  theme(legend.position = "bottom", plot.title = element_text(hjust=0.5, face = "bold"))

model_compare <- data.frame(Model, Recall)
ggplot(aes(x=Model, y=Accuracy, fill=Model), data=model_compare) +
  geom_bar(stat='identity') + ggtitle('Recall Comparison of Models') +
  xlab('Models') + ylab('Recall') + theme_minimal() +
  theme(legend.position = "bottom", plot.title = element_text(hjust=0.5, face = "bold"))

model_compare <- data.frame(Model, F1_score)
ggplot(aes(x=Model, y=Accuracy, fill=Model), data=model_compare) +
  geom_bar(stat='identity') + ggtitle('F1 score Comparison of Models') +
  xlab('Models') + ylab('F1 score') + theme_minimal() +
  theme(legend.position = "bottom", plot.title = element_text(hjust=0.5, face = "bold"))
```

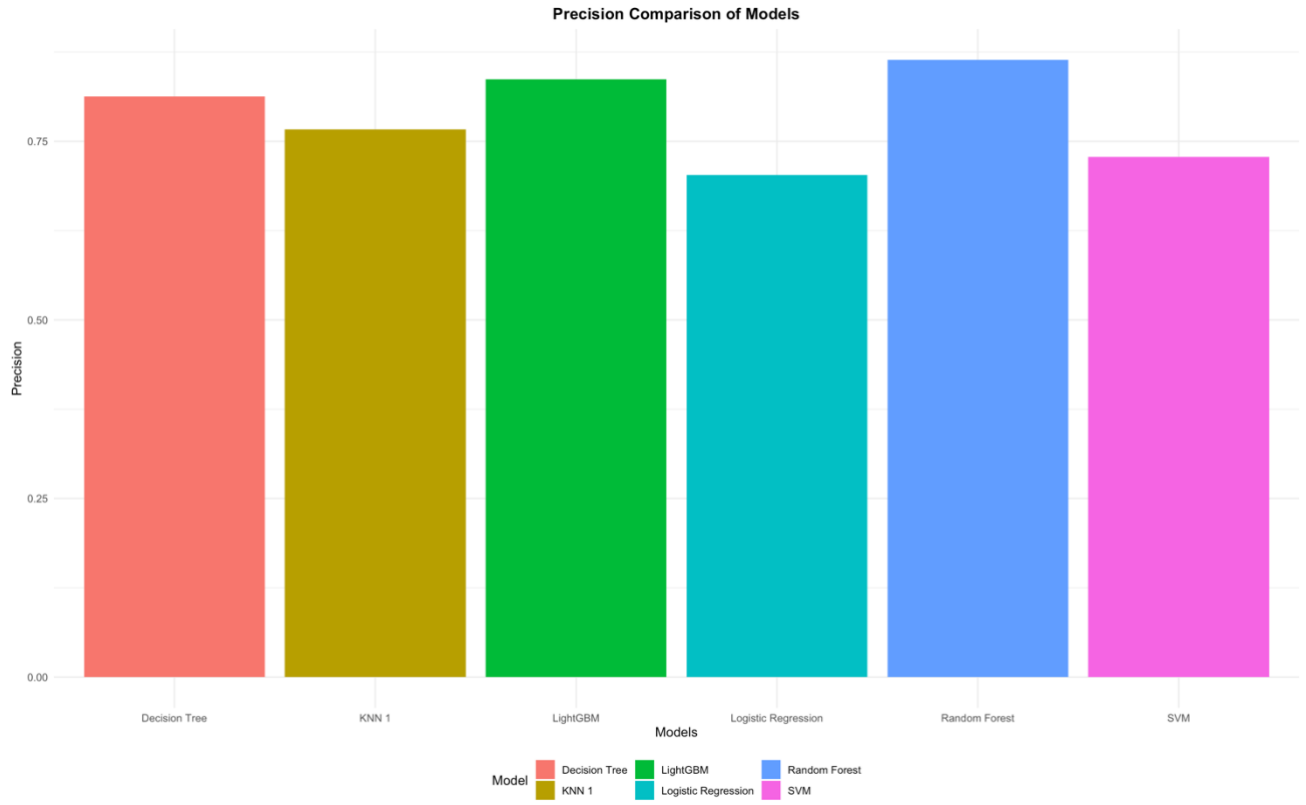
```
Accuracy <- c(cm_1$overall[1],cm_gbm$overall[1],
              cm_dt$overall[1],cm_svm$overall[1],
              cm_rf$overall[1],cm_lr$overall[1])

MSE <- c(mse_1, mse_gbm, mse_dt, mse_svm, mse_rf, mse_lr)
MAE <- c(mae_1, mae_gbm, mae_dt, mae_svm, mae_rf, mae_lr)
RMSE <- c(rmse_1, rmse_gbm, rmse_dt, rmse_svm, rmse_rf, rmse_lr)
Precision <- c(precision_1, precision_gbm, precision_dt, precision_svm, precision_rf,
               precision_lr)
Recall <- c(recall_1, recall_gbm, recall_dt, recall_svm, recall_rf, recall_lr)
F1_score <- c(F1_score_1, F1_score_gbm, F1_score_dt, F1_score_svm, F1_score_rf, F1_score_lr)
Model <- c('KNN 1','LightGBM',
           'Decision Tree', 'SVM',
           'Random Forest', 'Logistic Regression')

model_compare <- data.frame(Model, Accuracy)
ggplot(aes(x=Model, y=Accuracy, fill=Model), data=model_compare) +
  geom_bar(stat='identity') + ggtitle('Accuracy Comparison of Models') +
  xlab('Models') + ylab('Overall Accuracy') + theme_minimal() +
  theme(legend.position = "bottom", plot.title = element_text(hjust=0.5, face = "bold"))
#gbm#3lr#3gbm
model_compare <- data.frame(Model, MSE)
ggplot(aes(x=Model, y=Accuracy, fill=Model), data=model_compare) +
  geom_bar(stat='identity') + ggtitle('MSE Comparison of Models') +
  xlab('Models') + ylab('MSE') + theme_minimal() +
  theme(legend.position = "bottom", plot.title = element_text(hjust=0.5, face = "bold"))

model_compare <- data.frame(Model, MAE)
ggplot(aes(x=Model, y=Accuracy, fill=Model), data=model_compare) +
  geom_bar(stat='identity') + ggtitle('MAE Comparison of Models') +
  xlab('Models') + ylab('MAE') + theme_minimal() +
  theme(legend.position = "bottom", plot.title = element_text(hjust=0.5, face = "bold"))

model_compare <- data.frame(Model, RMSE)
ggplot(aes(x=Model, y=Accuracy, fill=Model), data=model_compare) +
  geom_bar(stat='identity') + ggtitle('RMSE Comparison of Models') +
  xlab('Models') + ylab('RMSE') + theme_minimal() +
  theme(legend.position = "bottom", plot.title = element_text(hjust=0.5, face = "bold"))
```

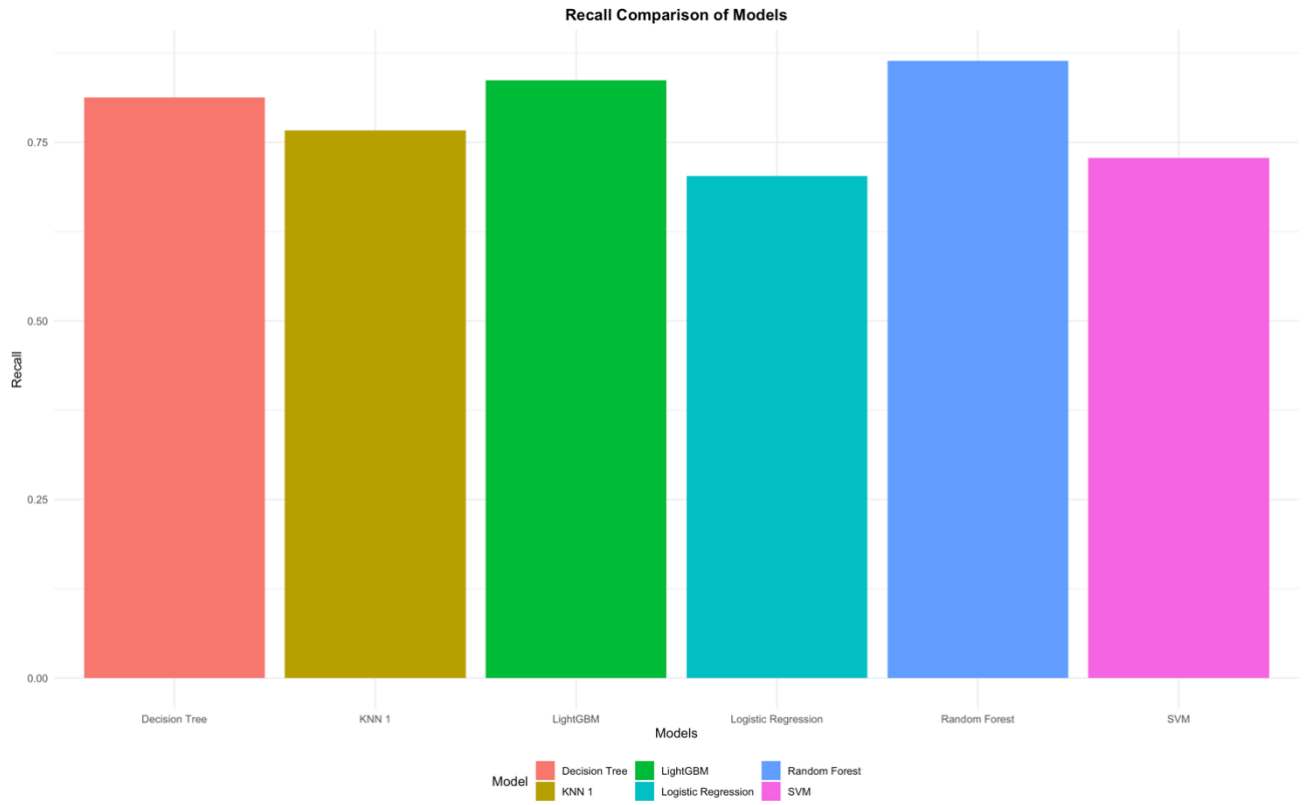

Accuracy Comparison of Models
Overall Accuracy
0.00
0.25
0.50
0.75
Decision Tree
KNN 1
LightGBM
Logistic Regression
Random Forest
SVM
Models
Model
Decision Tree
KNN 1
LightGBM
Logistic Regression
Random Forest
SVM

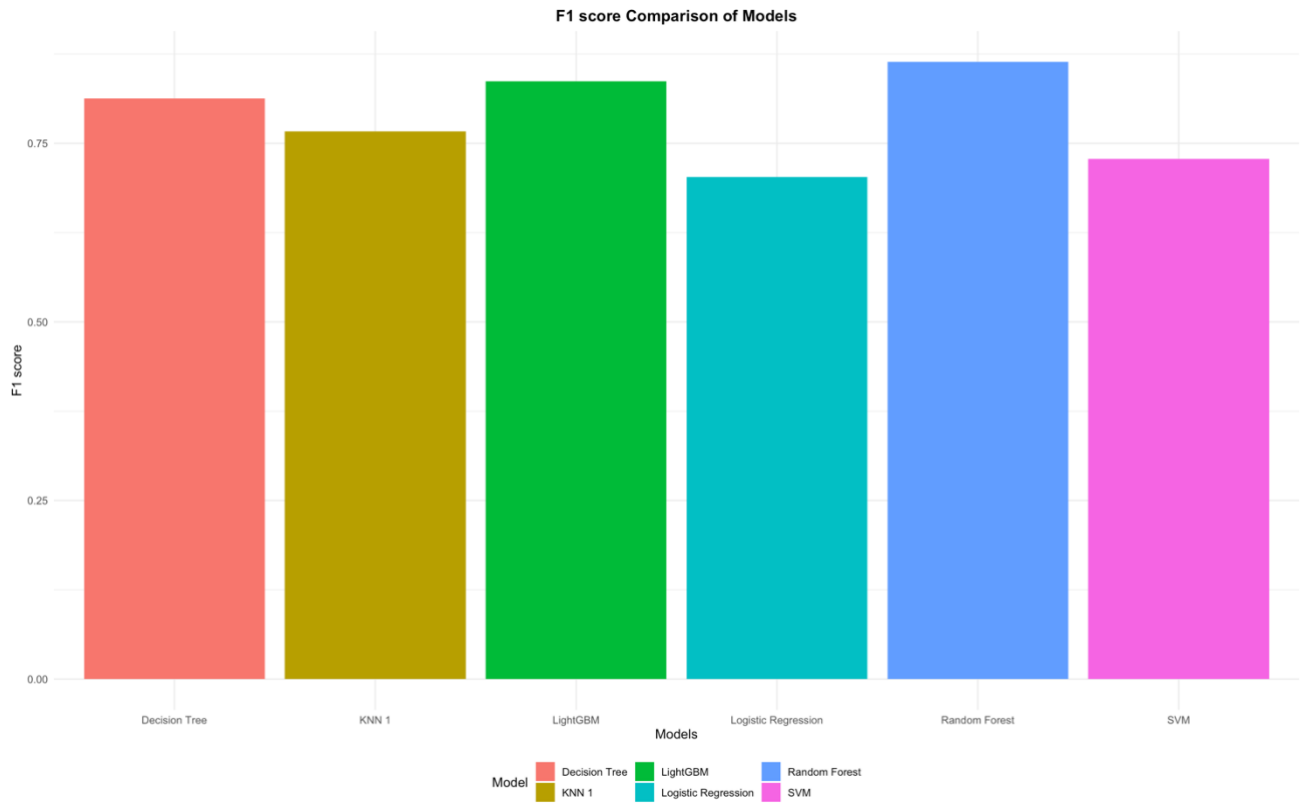

MSE Comparison of Models
MSE
0.00
0.25
0.50
0.75
Decision Tree
KNN 1
LightGBM
Logistic Regression
Random Forest
SVM
Models
Model
Decision Tree
KNN 1
LightGBM
Logistic Regression
Random Forest
SVM

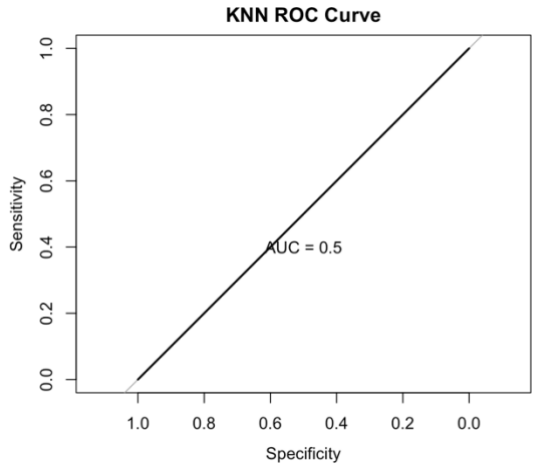

MAE Comparison of Models
MAE
0.00
0.25
0.50
0.75
Decision Tree
KNN 1
LightGBM
Logistic Regression
Random Forest
SVM
Models
Model
Decision Tree
KNN 1
LightGBM
Logistic Regression
Random Forest
SVM

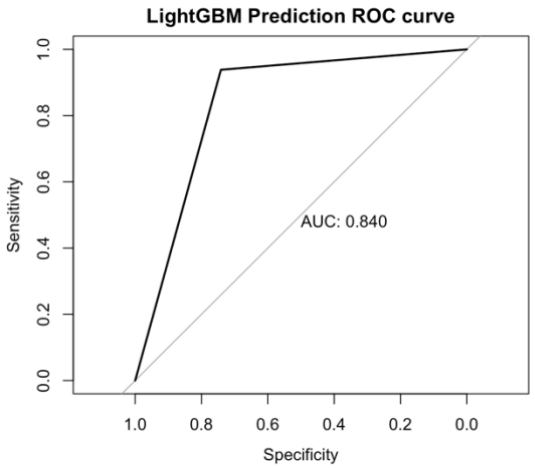

RMSE Comparison of Models
RMSE
0.00
0.25
0.50
0.75
Decision Tree
KNN 1
LightGBM
Logistic Regression
Random Forest
SVM
Models
Model
Decision Tree
KNN 1
LightGBM
Logistic Regression
Random Forest
SVM

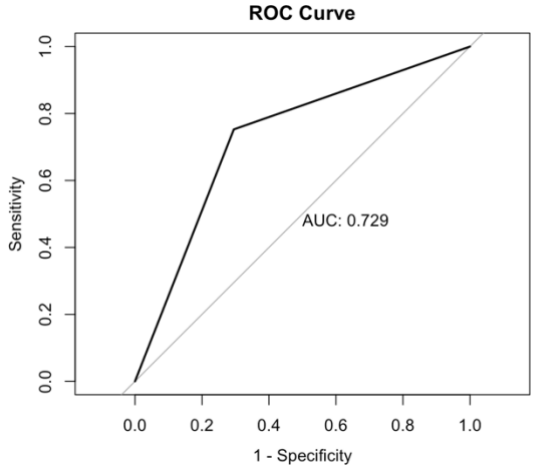

Precision Comparison of Models
Precision
0.00
0.25
0.50
0.75
Decision Tree
KNN 1
LightGBM
Logistic Regression
Random Forest
SVM
Models
Model
Decision Tree
KNN 1
LightGBM
Logistic Regression
Random Forest
SVM

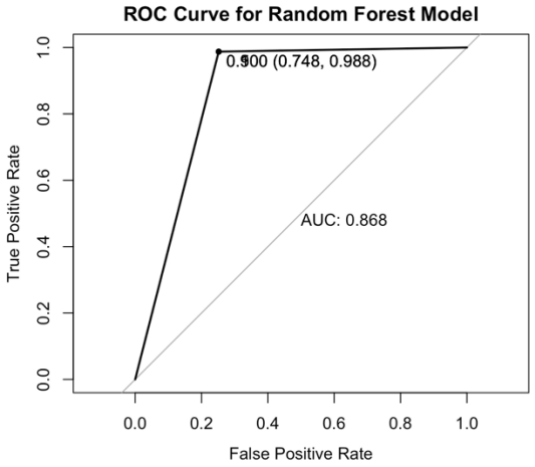

Recall Comparison of Models
Recall
0.00
0.25
0.50
0.75
Decision Tree
KNN 1
LightGBM
Logistic Regression
Random Forest
SVM
Models
Model
Decision Tree
KNN 1
LightGBM
Logistic Regression
Random Forest
SVM

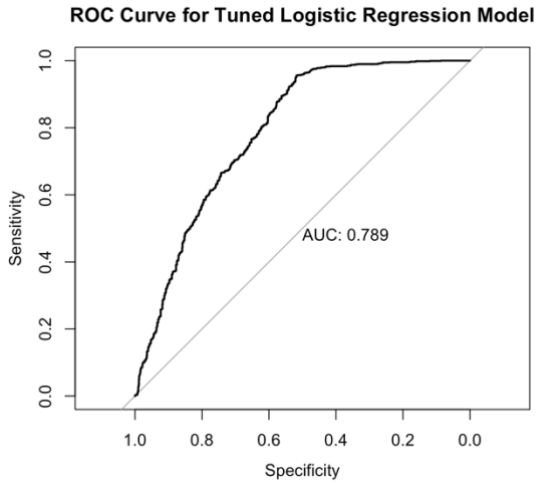

F1 score Comparison of Models
F1 score
0.00
0.25
0.50
0.75
Decision Tree
KNN 1
LightGBM
Logistic Regression
Random Forest
SVM
Models
Model
Decision Tree
KNN 1
LightGBM
Logistic Regression
Random Forest
SVM

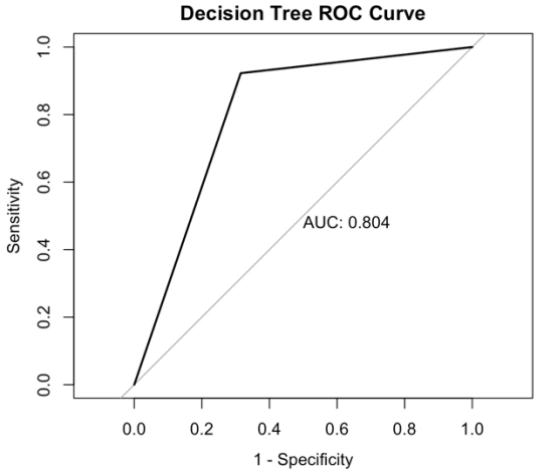

KNN ROC Curve
Sensitivity
Specificity
AUC = 0.5


LightGBM Prediction ROC curve
Sensitivity
Specificity
AUC: 0.840


ROC Curve
Sensitivity
1 - Specificity
AUC: 0.729


ROC Curve for Random Forest Model
True Positive Rate
False Positive Rate
AUC: 0.868


ROC Curve for Tuned Logistic Regression Model
Sensitivity
Specificity
AUC: 0.789


Decision Tree ROC Curve
Sensitivity
1 - Specificity
AUC: 0.804

10.7.1 Model Choosing and Enhancement

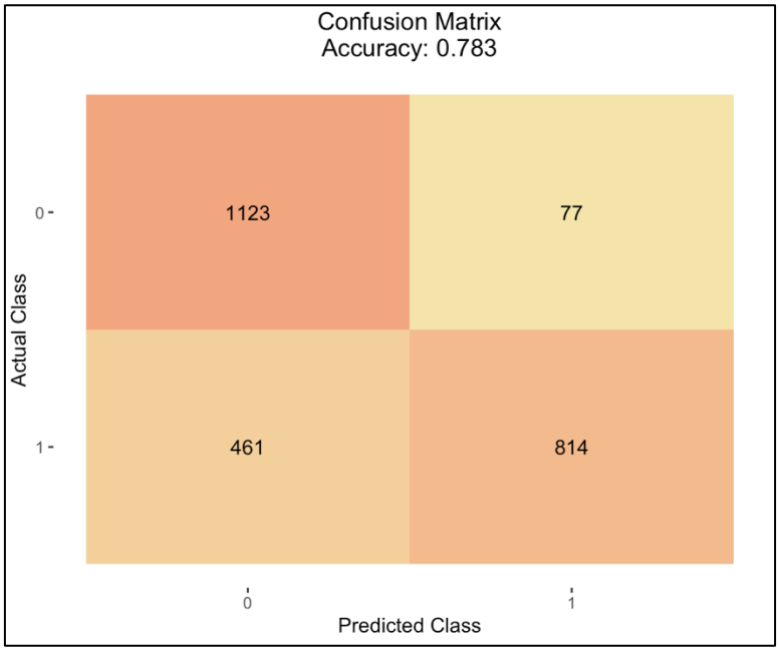
```
# Random Forest Model Enhancement ====
featureselect <- strokeTrain %>%
  dplyr::select(gender,age,ever_married,stroke,heart_disease,bmi,avg_glucose_level)
featureselect_test<- strokeTest %>%
  dplyr::select(gender,age,ever_married,stroke,heart_disease,bmi,avg_glucose_level)
rf_model <- randomForest(stroke ~ ., data = featureselect)
write.csv(featureselect, file = "strokeTrain.csv", row.names = TRUE)

#Make predictions on test set
rf_preds <- predict(rf_model,strokeTest)

library(pROC)
rf_preds <- as.numeric(as.character(rf_preds))
rf_preds <- ordered(rf_preds, levels = c("0", "1"))
rf_auc <- auc(strokeTest$stroke, rf_preds)

# Evaluate the accuracy of the random forest model
# Convert rf_preds to numeric
rf_preds <- as.numeric(as.character(rf_preds))
# Evaluate the accuracy of the random forest model
rf_acc <- sum(rf_preds == strokeTest$stroke) / nrow(strokeTest)
print(rf_acc)

# Calculate MSE, MAE, RMSE ----
test_rf <- as.integer(strokeTest$stroke)
mse_rf <- mean((test_rf - rf_preds)^2)
mae_rf <- mean(abs(test_rf - rf_preds))
rmse_rf <- sqrt(mean((test_rf - rf_preds)^2))
cat("MSE_rf: ", mse_rf, "\nMAE_rf: ", mae_rf, "\nRMSE_rf: ", rmse_rf)

# Create confusion matrix
cm_rf <- confusionMatrix(as.factor(rf_preds), strokeTest$stroke)
```

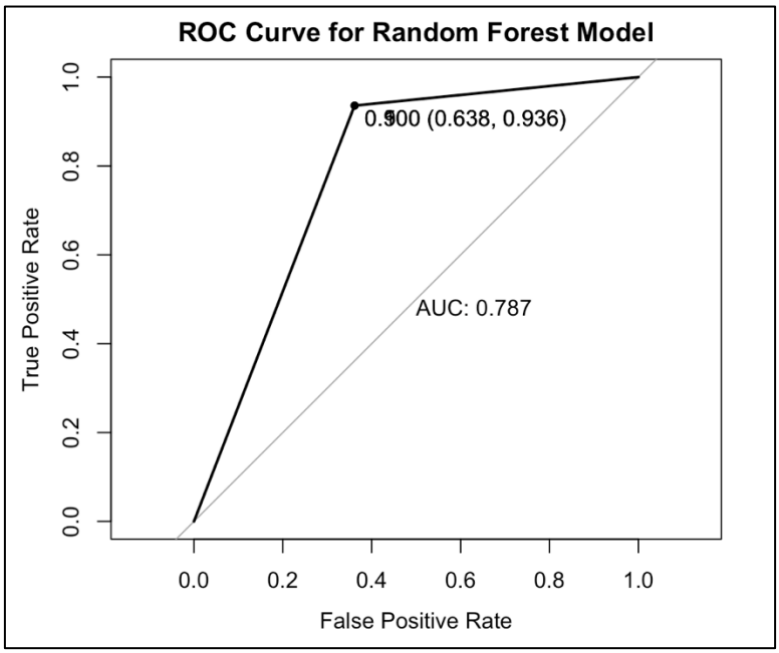
```
library(ggplot2)
library(ROCR)
# Confusion Matrix plot
cmv_rf <- conf_mat(table(strokeTest$stroke, rf_preds))
autoplot(cmv_rf, type = 'heatmap', label = TRUE) +
  scale_fill_gradient(low = "#F3E5AB", high = "#F1A680") +
  labs(title = paste("Confusion Matrix\nAccuracy:", round(cm_rf$overall["Accuracy"], 3)),
       x = "Predicted Class", y = "Actual Class")+
  theme(plot.title = element_text(hjust = 0.5))

# ROC curve plot
roc_rf <- roc(strokeTest$stroke, rf_preds, levels = rev(levels(strokeTest$stroke)))
plot(roc_rf, print.auc = TRUE, legacy.axes = TRUE,
     main = "ROC Curve for Random Forest Model",
     xlab = "False Positive Rate", ylab = "True Positive Rate",
     print.thres = c(0.1, 0.5, 0.9))

# Calculate precision, recall and F1 score
TN_rf <- cm_rf$table[1,1] # True negatives
FP_rf <- cm_rf$table[1,2] # False positives
FN_rf <- cm_rf$table[2,1] # False negatives
TP_rf <- cm_rf$table[2,2] # True positives

precision_rf <- TP_rf / (TP_rf + FP_rf)
recall_rf <- TP_rf / (TP_rf + FN_rf)
F1_score_rf <- 2 * precision_rf * recall_rf / (precision_rf + recall_rf)

cat("Precision: ", precision_rf, "\nRecall: ", recall_rf, "\nF1 score: ", F1_score_rf, "\n")
```

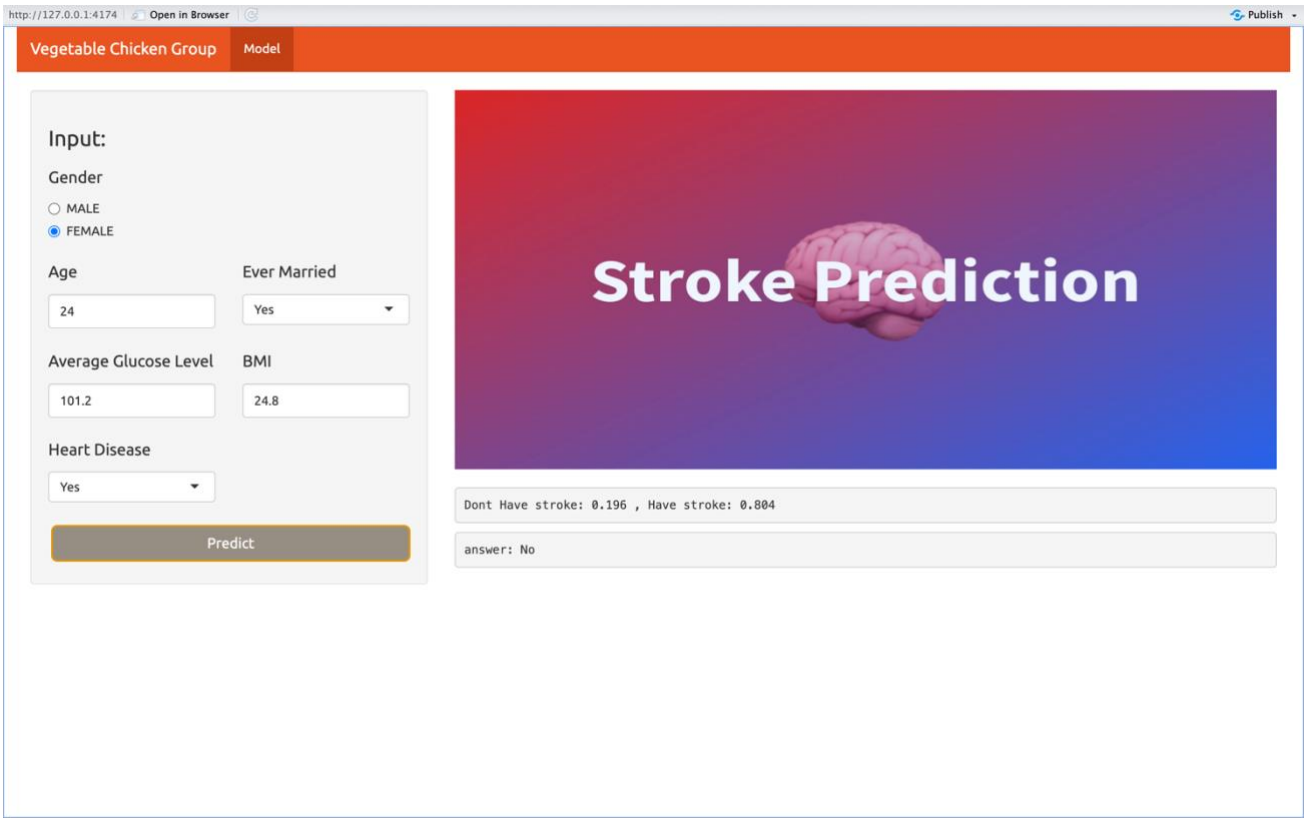
```
> print(rf_acc)
[1] 0.7826263
```

```
> cat("MSE_rf: ", mse_rf, "\nMAE_rf: ", mae_rf, "\nRMSE_rf: ", rmse_rf)
MSE_rf:  1.527677
MAE_rf:  1.155152
RMSE_rf:  1.235992
```

```
> cat("Precision: ", precision_rf, "\nRecall: ", recall_rf, "\nF1 score: ", F1_score_rf, "\n")
Precision:  0.6384314
Recall:  0.9135802
F1 score:  0.7516159
```






http://127.0.0.1:4174
Open in Browser
Publish
Vegetable Chicken Group
Model
Input:
Gender
MALE
FEMALE
Age
24
Ever Married
Yes
Average Glucose Level
101.2
BMI
24.8
Heart Disease
Yes
Predict
Stroke Prediction
Dont Have stroke: 0.196 , Have stroke: 0.804
answer: No

```
library(shiny)
library(randomForest)
library(shinythemes)
library(dplyr)
library(remotes)
library(fastDummies)
library(caret)
#--------make sure all library is installed-------------
#setwd('D:/Crystal/Capstone_project') #set the working directory
traindataset<- read.csv("/Capstone Project/strokeTrain.csv",header=TRUE)  #read the train dataset from the directory
traindataset <- select(traindataset, -X) #remove the X value which is ID
str(traindataset)
traindataset$stroke<-as.factor(traindataset$stroke)
# convert the target variable (stroke) as factor
str(traindataset) #check structure of dataset
model <- randomForest(stroke ~ ., data = traindataset) # build the random forest model
# start building the UI interface
ui <- fluidPage(theme = shinytheme("united"),
                tags$head(
                  tags$style(HTML("
                  .button {
                    background-color: #e95420;
                    color: white;
                    padding: 10px 20px;
                    border: none;
                    text-align: center;
                    text-decoration: none;
                    display: inline-block;
                    font-size: 16px;
                    margin: 4px 2px;
                    cursor: pointer;
                    border-radius: 8px;
                    width:100%;
                  }")),
                navbarPage(
                  # theme = "cerulean",  # <--- To use a theme, uncomment this
                  "Vegetable  Chicken Group",
                  tabPanel("Model",
                           sidebarLayout(position = "left",
                                         sidebarPanel(
                                           tags$h3("Input:"),
                                           radioButtons("gender", h4("Gender"),     #build radio button for gender
                                                        choices = list("MALE" = 2,
                                                                       "FEMALE" = 1),
                                                        selected = 1),
                                           fluidRow(  # 2 column in 1 row
                                             column(6,
                                                    numericInput("age",
                                                                 h4("Age"),
                                                                 value = 1)),
                                             column(6,
                                                    selectInput("evermaried", h4("Ever Married"),
                                                                choices = list("Yes" = 2, "No" = 1),
                                                                selected = 1)),
                                             column(6,
                                                    numericInput("avg_glucose_level",
                                                                 h4("Average Glucose Level"),
                                                                 value = 1)),
                                             column(6,
                                                    numericInput("bmi",
                                                                 h4("BMI"),
                                                                 value = 1)),
                                             column(6,
                                                    selectInput("heart_disease", h4("Heart Disease"),
                                                                choices = list("Yes" = 2, "No" = 1),
                                                                selected = 1)),
                                           ),
                                             actionButton("submit","Predict",class = "button") #add an action button
                                         ),
                                         mainPanel(
                                          div(
                                            style = "height: 440px; text-align: center;",
                                            imageOutput("Image")
                                          ),
                                          verbatimTextOutput("myOutput"),  #output field
                                          verbatimTextOutput("final")      #output field
                                         ),

                                  ) # mainPanel
                         ),# sidebarPanel
                  ), # Navbar 1, tabPanel
) # navbarPage
# fluidPage
#this is the place where the server backend process your model
server <- function(input, output) {
  output$Image<-renderImage({
    image_path<-"Crystal System/Capstone Project/WhatsApp Image 2023-05-14 at 1.26.31 AM.jpeg"
    list(
      src=image_path,
      alt="Stroke Image",
      width="100%",
      height="420px")
  },deleteFile = FALSE)
  formData <-reactiveValues(data = data.frame())
  observeEvent(input$submit,{  #this event will do after the action button was click
    set.seed(123)
    # get all the data and make them into a dataframe
    new_row <- data.frame(gender = input$gender,
                          age = input$age,
                          ever_married = input$evermaried,
                          avg_glucose_level = input$avg_glucose_level,
                          bmi = input$bmi,
                          heart_disease = input$heart_disease)
    formData$data <- rbind(formData$data, new_row)
    new_row <- new_row %>% mutate(avg_glucose_level = 1/avg_glucose_level) #reciprocal avg glucose level
    new_row <- as.data.frame(lapply(new_row, as.numeric))  #convert all data into numeric
    print(str(new_row)) #check the structure of input dataset
    # Perform prediction using the trained model
    prediction <- predict(model, newdata = new_row,"prob")  #predict the model and output in probability
    final <- predict(model, newdata = new_row)              #predict the model and output in yes/no
    output$myOutput <- renderText({                         #this is the output field
      doesnot_have_stroke <- prediction[1]
      have_stroke <- prediction[2]
      paste("Dont Have stroke:", doesnot_have_stroke, ", Have stroke:", have_stroke)
    })
    output$final <- renderText({
      paste("answer:",final)
    })})}
#this is the place where the UI run
shinyApp(ui = ui, server = server)
```

WORKLOAD MATRIX

Team Member	Workload
NEAW AIK KA	- LightGBM Modelling - KNN modelling - Model comparison - Exploratory data analysis - Feature engineering - Handling outliers
SIAH YAO LIANG	- Data exploration - Data cleaning - Exploratory data analysis - SVM model - Decision model - Model deployment
TAI RUI XIAN	- Feature engineering - Data exploratory analysis - Random forest model - Logistic regression - Problem statement - Literature review