Stroke Prediction Project

Ying Mi

2021-05-21

Stroke Prediction Dataset is used to predict whether a patient is likely to get stroke based on the input parameters like gender, age, various diseases, and smoking status. Each row in the data provides relevant information about the patient.

Instruction & Overview

1. Instruction

This project is to use stroke prediction dataset, preprocess the data, find pattern of data, create Train and Final Hold-out test Sets, Develop multiple algorithm & models in train set. Using final hold-out test set to validate prediction. Evaluate prediction through confusionMatrix to find the best model for this dataset prediction.

2. Dataset Overview

Dataset is originally located at kaggle as .csv file. In order to easily access without account and streamline the download process, data file is loaded in github. It includes 5,110 observations/patients and 12 variable.

```r
knitr::kable(t(dim(dat)),caption = "Initial dataset dimension")
```

Table 1: Initial dataset dimension

5110	12

Below are the detail for these 12 variables in initial dataset

Categorical outcome:

```
stroke: 1 if the patient had a stroke or 0 if not
```

Categorical features:

```
gender: "Male", "Female" or "Other"
hypertension: 0 if the patient doesn't have hypertension,
              1 if the patient has hypertension
heart_disease: 0 if the patient doesn't have any heart diseases,
               1 if the patient has a heart disease
ever_married: "No" or "Yes"
work_type: "children", "Govt_jov", "Never_worked", "Private" or "Self-employed"
Residence_type: "Rural" or "Urban"
smoking_status: "formerly smoked", "never smoked", "smokes" or "Unknown"
```

Quantitative feature:

```
age: age of the patient
avg_glucose_level: average glucose level in blood
bmi: body mass index
id: unique identifier
```

We can get basic descriptive statistic for initial dataset through describe function:

```
library(Hmisc)
print(describe(dat))
#> dat
#>
#>  12  Variables      5110  Observations
#> --------------------------------------------------------------------------------
#> id
#>        n  missing distinct     Info     Mean      Gmd      .05      .10
#>     5110        0     5110        1    36518    24436     3590     6972
#>      .25      .50      .75      .90      .95
#>    17741    36932    54682    65668    69218
#>
#> lowest :    67    77    84    91    99, highest: 72911 72914 72915 72918 72940
#> --------------------------------------------------------------------------------
#> gender
#>        n  missing distinct
#>     5110        0        3
#>
#> Value      Female   Male  Other
#> Frequency    2994   2115      1
#> Proportion  0.586  0.414  0.000
#> --------------------------------------------------------------------------------
#> age
#>        n  missing distinct     Info     Mean      Gmd      .05      .10
#>     5110        0      104        1    43.23    26.03        5       11
#>      .25      .50      .75      .90      .95
#>       25       45       61       75       79
#>
#> lowest :  0.08  0.16  0.24  0.32  0.40, highest: 78.00 79.00 80.00 81.00 82.00
#> --------------------------------------------------------------------------------
#> hypertension
#>        n  missing distinct     Info      Sum     Mean      Gmd
#>     5110        0        2    0.264      498  0.09746    0.176
#>
#> --------------------------------------------------------------------------------
#> heart_disease
#>        n  missing distinct     Info      Sum     Mean      Gmd
#>     5110        0        2    0.153      276  0.05401   0.1022
#>
#> --------------------------------------------------------------------------------
#> ever_married
#>        n  missing distinct
#>     5110        0        2
#>
#> Value         No   Yes
#> Frequency   1757  3353
#> Proportion 0.344 0.656
#> --------------------------------------------------------------------------------
#> work_type
#>        n  missing distinct
#>     5110        0        5
#>
#> lowest : children      Govt_job      Never_worked  Private       Self-employed
#> highest: children      Govt_job      Never_worked  Private       Self-employed
#>
#> Value           children      Govt_job  Never_worked       Private
#> Frequency            687           657            22          2925
#> Proportion         0.134         0.129         0.004         0.572
#>
#> Value      Self-employed
#> Frequency            819
#> Proportion         0.160
#> --------------------------------------------------------------------------------
#> Residence_type
#>        n  missing distinct
#>     5110        0        2
#>
#> Value      Rural Urban
#> Frequency   2514  2596
#> Proportion 0.492 0.508
#> --------------------------------------------------------------------------------
#> avg_glucose_level
#>        n  missing distinct     Info     Mean      Gmd      .05      .10
#>     5110        0     3979        1    106.1    45.38    60.71    65.79
#>      .25      .50      .75      .90      .95
#>    77.24    91.88   114.09   192.18   216.29
#>
#> lowest :  55.12  55.22  55.23  55.25  55.26, highest: 266.59 267.60 267.61 267.76 271.74
#> --------------------------------------------------------------------------------
#> bmi
#>        n  missing distinct
#>     5110        0      419
#>
#> lowest : 10.3 11.3 11.5 12   12.3, highest: 71.9 78   92   97.6 N/A
#> --------------------------------------------------------------------------------
#> smoking_status
#>        n  missing distinct
#>     5110        0        4
#>
#> Value      formerly smoked    never smoked          smokes         Unknown
#> Frequency              885            1892             789            1544
#> Proportion           0.173           0.370           0.154           0.302
#> --------------------------------------------------------------------------------
#> stroke
#>        n  missing distinct     Info      Sum     Mean      Gmd
#>     5110        0        2    0.139      249  0.04873  0.09273
#>
#> --------------------------------------------------------------------------------
```

Methods/Analysis

1. Data Cleaning

Pre-processing data is to transform predictors & outcome, make them consistent, increase data reliability, assure the data are useful and functional toward the intended end exploration,analysis and modeling.

a. Replace missing values with NA

Through basic descriptive statistic, we find All values of 'N/A' and 'Unknown' are supposed to be missing values in different variables. In order to transform data type with same way and also deal with missing value easily later, we use NA to represent all missing values in dataset first.

```
dat[dat=='N/A']<-NA
dat[dat=='Unknown']<-NA
```

b. Transform data type for categorical features & outcome to factor, quantitative feature to numeric

```
library(descriptr)
datAdj<- data.frame(dat) %>%
  mutate(
  id=as.numeric(id),
  gender=factor(gender),
  age=as.numeric(age),
  hypertension=factor(ifelse(hypertension==1, "Yes","No")),
  heart_disease=factor(ifelse(heart_disease==1,"Yes","No")),
  ever_married=factor(ever_married),
  work_type=factor(work_type),
  Residence_type=factor(Residence_type),
  avg_glucose_level=as.numeric(avg_glucose_level),
  bmi=as.numeric(bmi),
  smoking_status=factor(smoking_status),
  stroke=factor(as.numeric(stroke)))
```

After transformation, dataset is much easier to explore and analysis. We can using ds_screener function and explore data level and missing values summary through the following table :

```
print(ds_screener(datAdj))
#> -------------------------------------------------------------------------------------------------------
#> |   Column Name    |  Data Type  |                          Levels                          |  Missing  |  Missing (%)
#> -------------------------------------------------------------------------------------------------------
#> |        id        |   numeric   |                            NA                            |     0     |       0
#> |      gender      |   factor    |                    Female Male Other                     |     0     |       0
#> |       age        |   numeric   |                            NA                            |     0     |       0
#> |  hypertension    |   factor    |                          No Yes                          |     0     |       0
#> |  heart_disease   |   factor    |                          No Yes                          |     0     |       0
#> |  ever_married    |   factor    |                          No Yes                          |     0     |       0
#> |    work_type     |   factor    |children Govt_job Never_worked Private Self-employed|     0     |       0
#> | Residence_type   |   factor    |                       Rural Urban                        |     0     |       0
#> |avg_glucose_level|   numeric   |                            NA                            |     0     |       0
#> |       bmi        |   numeric   |                            NA                            |    201    |     3.93
#> | smoking_status   |   factor    |         formerly smoked never smoked smokes              |   1544    |    30.22
#> |      stroke      |   factor    |                           0 1                            |     0     |       0
#> ----------------------------------------------------------------------------------------------------------
#>
#>  Overall Missing Values           1745
#>  Percentage of Missing Values     2.85 %
#>  Rows with Missing Values         1684
#>  Columns With Missing Values      2
```

c. Deal with missing values

From the summary table, missing values are found in smoking_status and bmi variables. Since large proportion of missing value's observation, simple exclusion can cause bias in the estimation of parameters and reduce statistical power. So different methods are used to recode missing data.

Recode missing data

a) bmi

Use mean for same class to replace missing data: Split ages into age groups by every 5 years, then use average BMI values to replace NA value for same gender and age group patients

```
library(AMR)
# Split ages into age groups by every 5 years
datAdj<-datAdj %>%mutate(agegroup=age_groups(age, split_at = "fives"))
# Get average BMI values by gender and age group
datAdj<-datAdj %>%group_by(gender,agegroup) %>%mutate(bmi_avg=mean(bmi,na.rm=T))
# replace BMI missing value with average BMI value with same gender and age group
datAdj$bmi[is.na(datAdj$bmi)]  <- datAdj$bmi_avg[is.na(datAdj$bmi)]
```

b) smoking_status

I) Patient Age <15

In the United States and much of Europe, research found regular lighting up typically begins between 15 and 16 years old, so replace smoking_status with 'never smoked' for patient age <15

```
# Replace smoking_status with 'never smoked' for patient age <15
datAdj$smoking_status[datAdj$age<15]<-"never smoked"
```

II) Patient Age >=15

Use mean for same class to replace missing data: for same gender and age group patients, use most frequent non-NA value in group to replace NA values

```
datAdj<-datAdj %>%
  group_by(gender,agegroup) %>%
  add_count(smoking_status) %>%
  mutate(smoking_status= if_else(is.na(smoking_status),
                                 smoking_status[which.max(n)],
                                 smoking_status)) %>%
  select(-n) %>%
  ungroup()
```

Using ds_screenerfunction to explore missing values summary again after missing value replacement.

```
print(ds_screener(datAdj %>% select(-c("bmi_avg","agegroup"))))
#> -----------------------------------------------------------------------------------------------------------------
#> |   Column Name   |  Data Type  |                       Levels                       |  Missing  |  Missing (%)
#> -----------------------------------------------------------------------------------------------------------------
#> |       id        |   numeric   |                         NA                         |     0     |      0
#> |     gender      |   factor    |                 Female Male Other                  |     0     |      0
#> |       age       |   numeric   |                         NA                         |     0     |      0
#> |  hypertension   |   factor    |                       No Yes                       |     0     |      0
#> |  heart_disease  |   factor    |                       No Yes                       |     0     |      0
#> |  ever_married   |   factor    |                       No Yes                       |     0     |      0
#> |    work_type    |   factor    |children Govt_job Never_worked Private Self-employed|     0     |      0
#> | Residence_type  |   factor    |                    Rural Urban                     |     0     |      0
#> |avg_glucose_level|   numeric   |                         NA                         |     0     |      0
#> |       bmi       |   numeric   |                         NA                         |     0     |      0
#> | smoking_status  |   factor    |        formerly smoked never smoked smokes         |    57     |     1.12
#> |     stroke      |   factor    |                        0 1                         |     0     |      0
#> -----------------------------------------------------------------------------------------------------------------
#>
#>  Overall Missing Values           57
#>  Percentage of Missing Values     0.09 %
#>  Rows with Missing Values         57
#>  Columns With Missing Values      1
```

Exclude uncoded missing value's rows

Since % of missing value rows is decreased from 33.54% to 1.12%, excluding the rest of missing value rows won't affect whole modeling. We remove them from initial dataset.

```
# Exclude NA observation and remove temporary columns and structure dataset
datFin <-na.omit(datAdj)%>% select(-c("bmi_avg","agegroup"));datFin <-data.frame(datFin)
```

d. Deal with outliers in the dataset

In gender column, only have one observation dropped in Other category. It is an outlier with very small sample size. We can remove it from dataset.

```
# Remove observation that gender is other
datFin<- datFin %>% filter(gender!='Other')
```

Through all adjustments, we can get dataset for modeling

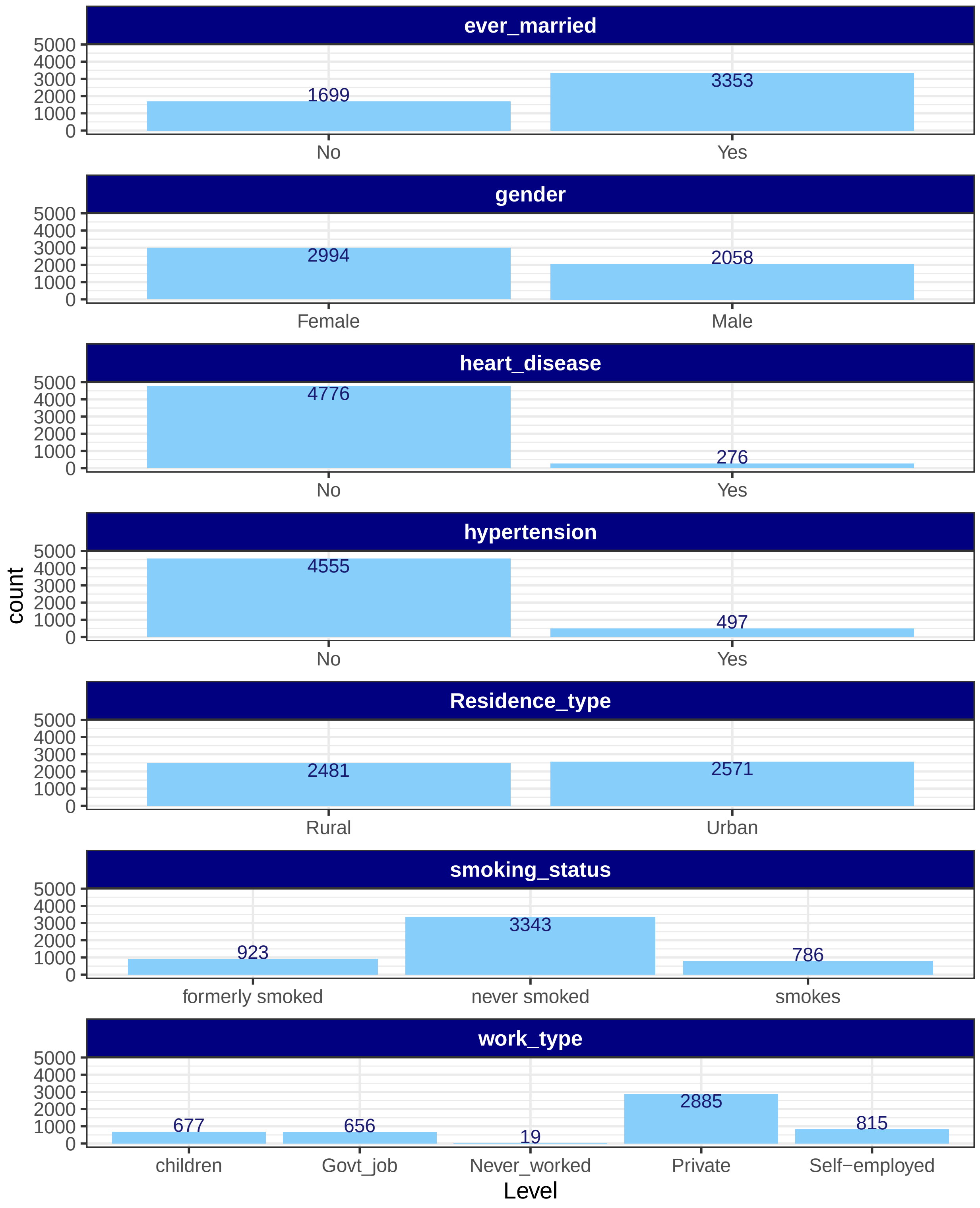
```r
# Quick view of dataset for modeling
set.seed(1, sample.kind="Rounding")
knitr::kable(datFin %>%group_by(stroke) %>%sample_n(3)%>%t(),caption = "Quick view of dataset for modeling")
```

Table 2: Quick view of dataset for modeling

id	10245	14889	36366	32729	18587	68023
gender	Female	Male	Male	Female	Female	Male
age	54	64	77	81	76	79
hypertension	No	No	No	No	No	No
heart_disease	No	No	No	No	No	No
ever_married	Yes	Yes	Yes	Yes	No	Yes
work_type	Self-employed	Govt_job	Govt_job	Private	Private	Private
Residence_type	Rural	Rural	Urban	Rural	Urban	Rural
avg_glucose_level	77.52	113.68	64.40	184.40	89.96	72.73
bmi	35.80000	24.20000	27.80000	27.50000	28.99198	28.40000
smoking_status	never smoked	never smoked	never smoked	never smoked	never smoked	never smoked
stroke	0	0	0	1	1	1

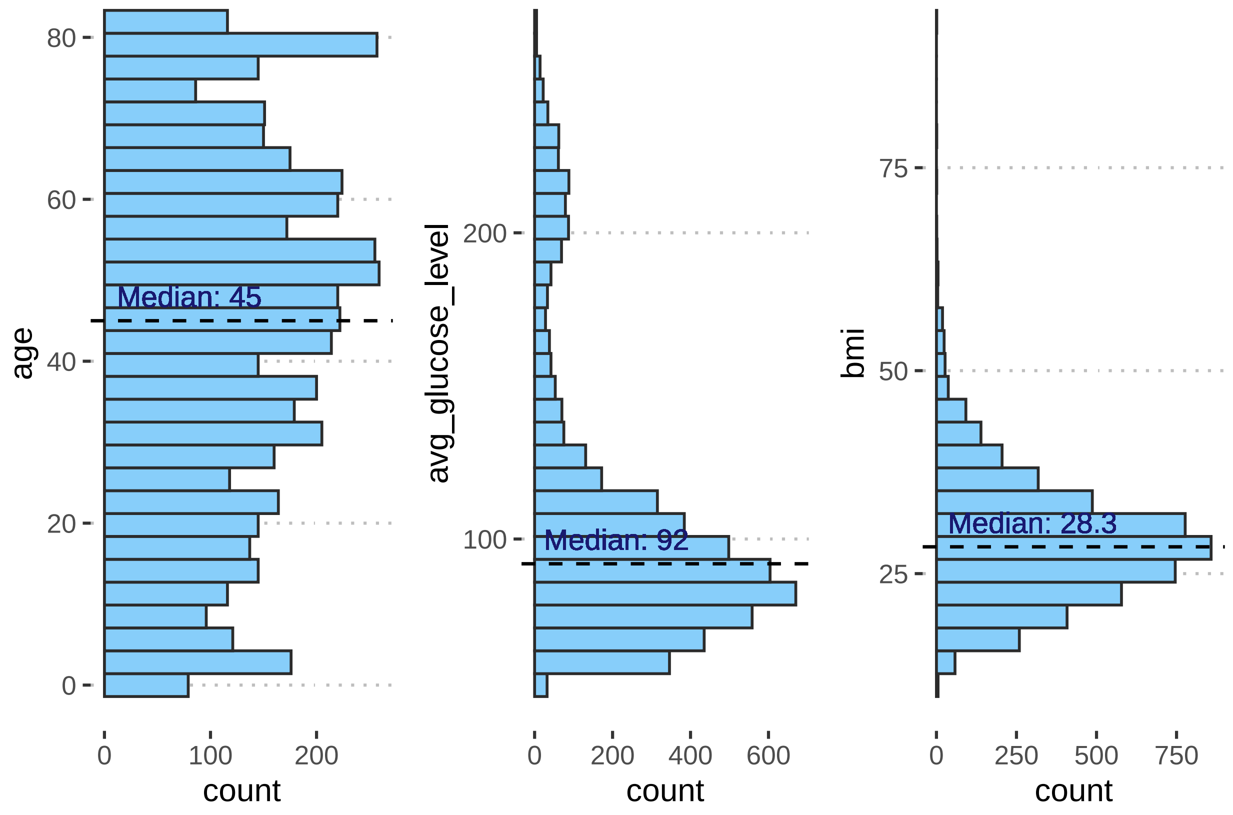
2. Data Exploration

a. Counts of categorical features



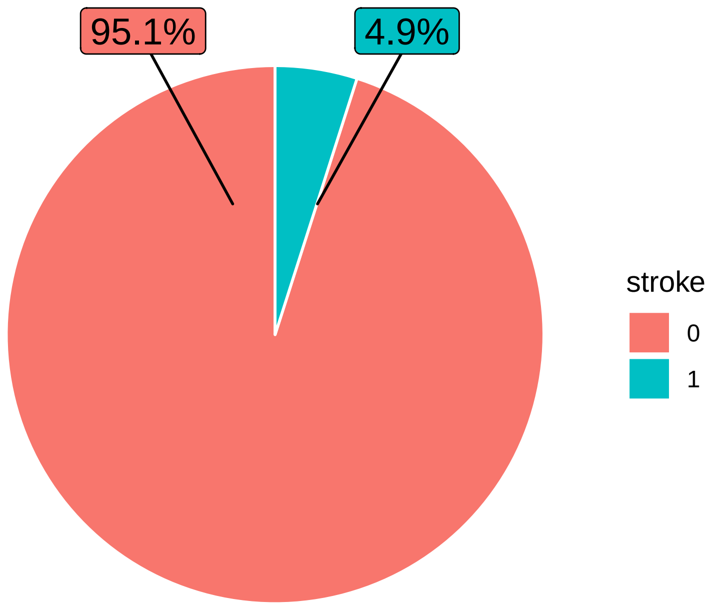
Through graph, we can see the distribution of categorical features. Majority patients don't have heart disease, don't have hypertension and never have smoked.

b. Distribution of continuous variables



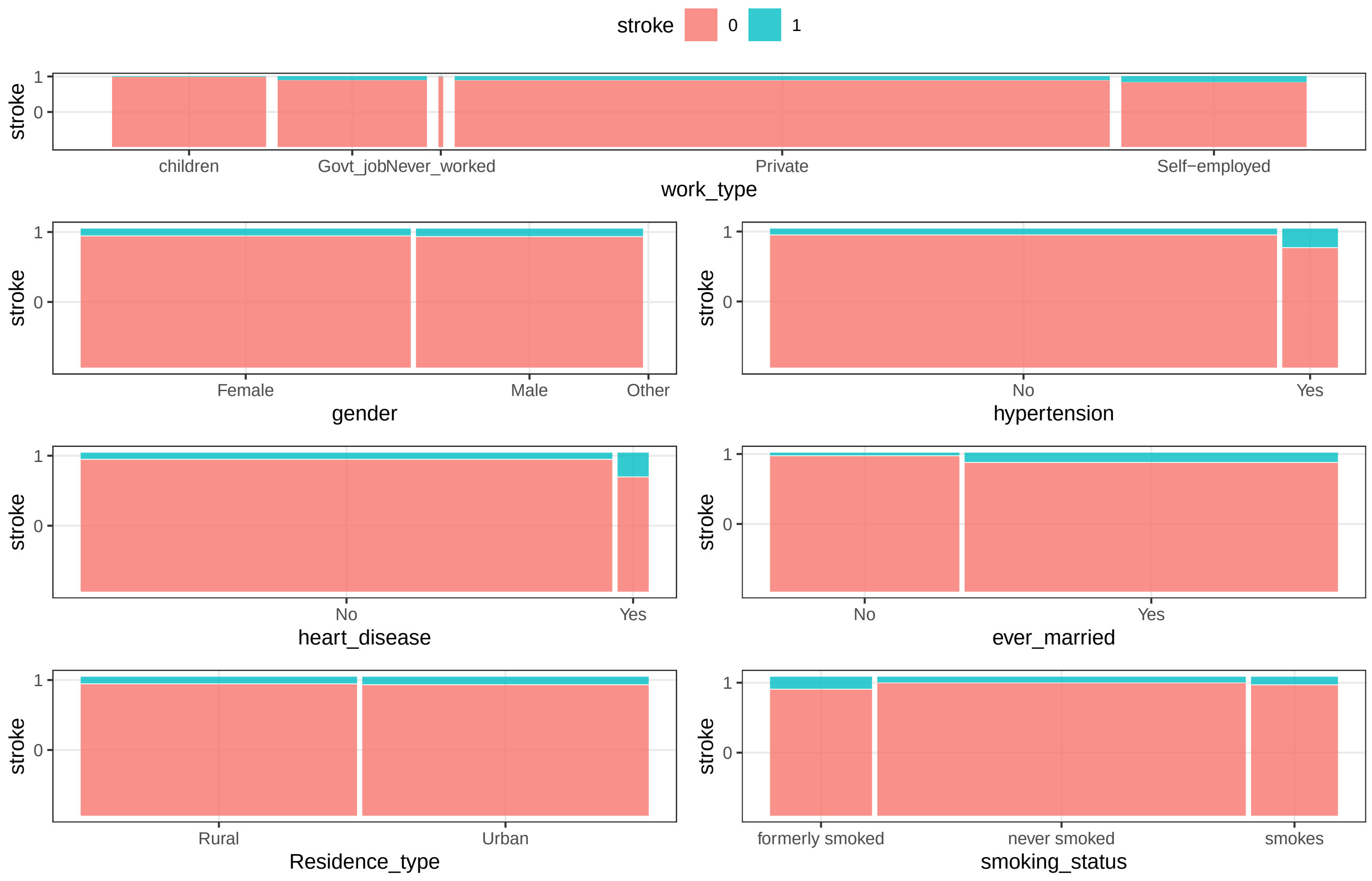
From graphs, patient's median age is 45, median average glucose level is 92 and median bmi is 28.3.

c. Distribution of outcome



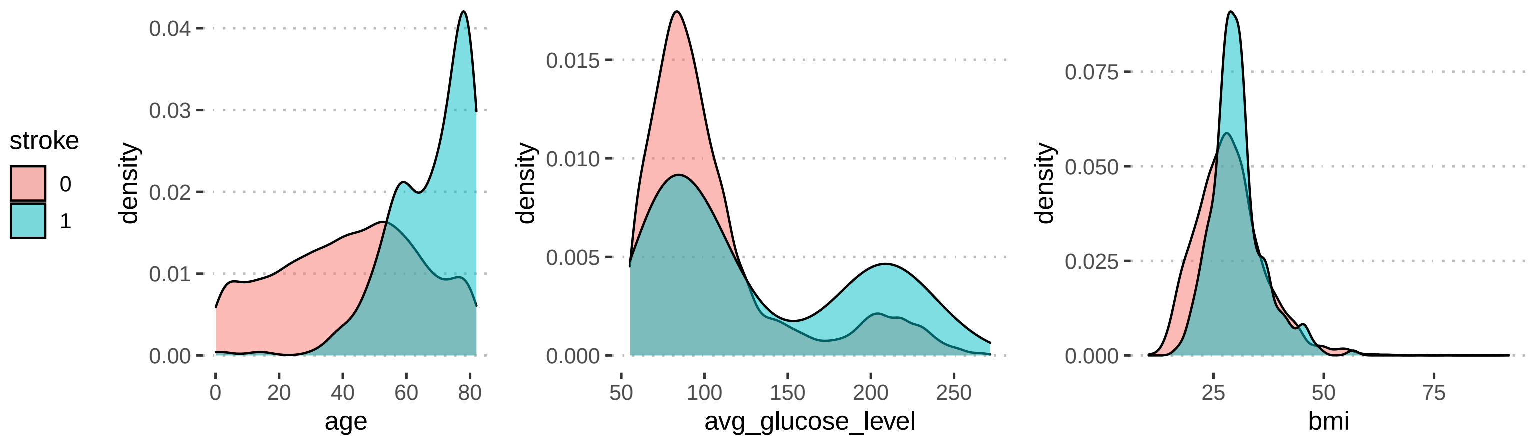
Based on graph, 95.1% patients don't have stroke, only 4.9% patients have stroke.

d. Distribution of categorical features vs stroke outcome



Through graphs, stroke patients more likely have hypertension or heart disease or smoke history or are adults with job.

e. Density of continuous variables vs stroke outcome



From density graphs, stroke patients are with higher proportion in older age, with higher proportion for higher average glucose level and higher proportion for bmi between 25-40 compared with non-stroke patients.

Data Exploration Codes

```
####################
# Data Exploration #
####################

library(tidyverse)
library(ggplot2)
library(ggpubr)

#####################################
# a. Counts of categorical features #
#####################################

#visualized all categorical feature's counts
datFreqExpCat<-gather(subset(datFin, select = -c(age,avg_glucose_level,bmi,stroke)),
                      "Attribute","Level",-id)
datFreqExpCat%>%
  ggplot(aes(x = Level)) +
  theme_pubclean()+
  geom_bar(fill = "lightskyblue") +geom_text(stat='Count',aes(label=..count..),
                                             colour="midnightblue",
                                             vjust = "inward",
                                             size=3)+
  facet_wrap(vars(Attribute),scales = "free_x", ncol =1)+
  theme_bw()+
  theme(strip.background =element_rect(fill="navy"))+
  theme(strip.text = element_text(colour = 'white',size = 10, face='bold' ))

###########################################
# b. Distribution of continuous variables #
###########################################

# Visualize distribution of age and explore median values for age
p1<-ggplot(datFin, aes(x = age)) +
  geom_histogram(bins = 30, color = "grey17", fill = "lightskyblue") +
  theme_pubclean()+
  geom_vline(aes(xintercept = median(age)),
             linetype = "dashed", size = 0.6)+
  geom_text(aes(x=median(age)+3, label=paste("Median:", round(median(age),digit=0)),
                y=80), colour="midnightblue") +coord_flip()

# Visualize distribution of avg_glucose_level and explore median values for avg_glucose_level
p2<-ggplot(datFin, aes(x = avg_glucose_level)) +
  geom_histogram(bins = 30, color = "grey17", fill = "lightskyblue") +
  theme_pubclean()+
  geom_vline(aes(xintercept = median(avg_glucose_level)),
             linetype = "dashed", size = 0.6)+
  geom_text(aes(x=median(avg_glucose_level)+8,
                label=paste("Median:", round(median(avg_glucose_level),digit=0))
                , y=210), colour="midnightblue")  +coord_flip()

# Visualize distribution of bmi and explore median values for bmi
p3<-ggplot(datFin, aes(x = bmi)) +
  geom_histogram(bins = 30, color = "grey17", fill = "lightskyblue") +
  theme_pubclean()+
```

```r
  geom_vline(aes(xintercept = median(bmi)),
             linetype = "dashed", size = 0.6)+
  geom_text(aes(x=median(bmi)+3,
                label=paste("Median:", round(median(bmi),digit=1))
                , y=300), colour="midnightblue")  +coord_flip()

# Bring 3 distribution graphs together in one row
library(gridExtra)
grid.arrange(p1, p2, p3, nrow = 1)

##############################
# c. Distribution of outcome #
##############################

# Pie chart for stroke's distribution
library(ggrepel)
datFin %>%
  group_by(stroke) %>%
  summarise(n = n()) %>%
  mutate(prop = n / sum(n)) %>% mutate(label=paste0(round(prop*100,1),'%'),
                                       cumulative = cumsum(n),
                                       midpoint = cumulative+ n / 2)%>%
  ggplot(aes(x ="", n, fill = stroke)) +
  geom_col(color="white") +
  coord_polar(theta = "y", start=0) +
  geom_label_repel(aes(label = label), size=5, show.legend = F, nudge_x = 1) +
  theme_void()

##############################################################
# d. Distribution of categorical features vs stroke outcome #
##############################################################

library(ggmosaic)
# Explore stroke distribution by gender
mosaicsg<-ggplot(data = datFin) +
  geom_mosaic(aes(x = product(stroke, gender), fill=stroke))+
  theme_bw()+
  guides(fill=FALSE)
# Explore stroke distribution by hypertension
mosaicshy<-ggplot(data = datFin) +
  geom_mosaic(aes(x = product(stroke, hypertension), fill=stroke))+
  theme_bw()+
  guides(fill=FALSE)
# Explore stroke distribution by heart_disease
mosaicsh<-ggplot(data = datFin) +
  geom_mosaic(aes(x = product(stroke, heart_disease), fill=stroke))+
  theme_bw()+
  guides(fill=FALSE)
# Explore stroke distribution by ever_married
mosaicsm<-ggplot(data = datFin) +
  geom_mosaic(aes(x = product(stroke, ever_married), fill=stroke))+
  theme_bw()+
  guides(fill=FALSE)
# Explore stroke distribution by work_type
mosaicsw<-ggplot(data = datFin) +
  geom_mosaic(aes(x = product(stroke, work_type), fill=stroke))+
  theme_bw()+
  theme(legend.position="top")
# Explore stroke distribution by Residence_type
mosaicsr<-ggplot(data = datFin) +
  geom_mosaic(aes(x = product(stroke, Residence_type), fill=stroke))+
  theme_bw()+
  guides(fill=FALSE)
# Explore stroke distribution by smoking_status
mosaicss<-ggplot(data = datFin) +
  geom_mosaic(aes(x = product(stroke, smoking_status), fill=stroke))+
  theme_bw()+
  guides(fill=FALSE)

# Bring distribution graphs together
grid.arrange(mosaicsw,mosaicsg,mosaicshy, mosaicsh,mosaicsm,mosaicsr,mosaicss,ncol = 2,
             layout_matrix = rbind(c(1, 1),
                                   c(2, 3),
                                   c(4, 5),
                                   c(6, 7)
             )
)

########################################################
# e. Density of continuous variables vs stroke outcome #
########################################################
# Explore density by stroke and age
p1s<-ggplot(datFin, aes(x = age,fill = stroke)) +
  geom_density(alpha = 0.5) +
  theme_pubclean()+theme(legend.position="left")
# Explore density by stroke and avg_glucose_level
p2s<-ggplot(datFin, aes(x = avg_glucose_level,fill = stroke)) +
  geom_density(alpha = 0.5) + guides(fill=FALSE)+
  theme_pubclean()
# Explore density by stroke and bmi
p3s<-ggplot(datFin, aes(x = bmi,fill = stroke)) +
  geom_density(alpha = 0.5) + guides(fill=FALSE)+
  theme_pubclean()
# Bring distribution graphs together in one row
grid.arrange(p1s, p2s, p3s, nrow = 1)
```

3. Mothod & Data Analysis

Since outcome is binomial, we decide to use Logistic regression model to predict stroke first.

> Logistic regression is a powerful statistical way of modeling a binomial outcome with one or more explanatory variables. It measures the relationship between the categorical dependent variable and one or more independent variables by estimating probabilities using a logistic function, which is the cumulative logistic distribution. It is represented with the following equation:
>
> $$log_b \frac{p}{1-p} = \beta_0 + \beta_1 x_1 + \beta_2 x_2 .... + \beta_n x_n$$
>
> In here p is the probability of the event that $Y = 1$

After modeling, we find simple model can't work well since even if we get very high accuracy however no any stroke patient is predicted. It is because stroke dataset is an unbalanced dataset in which there is a disproportionate ratio of observations for stroke and un-stroke patients. It generates bias in prediction model.

In order to solve this problem, we apply one of oversample methodology - SMOTE (i.e. Synthetic Minority Oversampling Technique) to adjust train dataset.

> SMOTE (Chawla et. al. 2002) is a well-known algorithm, to artificially generate new examples of the minority class using the nearest neighbors of these cases. Also, the majority class examples are also under-sampled, leading to a more balanced dataset. The algorithm is using bootstrapping and k-nearest neighbor to synthetically create additional observations of rare event.

Through SMOTE adjustment, train dataset becomes balanced. Then we reapply regression model and using 10-folder cross validation way to get reasonable prediction model.

After that, we also try random forest and KNN model in balanced train dataset and also get reasonable prediction models.

> For Random forest, it is a supervised learning algorithm. The "forest" it builds, is an ensemble of decision trees, usually trained with the "bagging" method. The general idea of the bagging method is that a combination of learning models increases the overall result. Its predictions for unseen samples x' can be made by averaging the predictions from all the individual regression trees on x':
>
> $$\hat{f} = \frac{1}{B} \sum_{b=1}^{B} f_b(x')$$
>
> For K nearest neighbors, it is a simple algorithm that stores all available cases and classifies new cases based on a similarity measure (e.g., distance functions). KNN has been used in statistical estimation and pattern recognition already in the beginning of 1970's as a non-parametric technique. A case is classified by a majority vote of its neighbors, with the case being assigned to the class most common amongst its K nearest neighbors measured by a distance function.
>
> $$\sum_{i=1}^{k} |x_i - y_i|$$

Finally through comparison of different prediction models, we get the best model for this stroke prediction.

Results

1. Finalize dataset for modeling, split train and test dataset

Since unique identifier doesn't affect stroke outcome, we remove this column from dataset for modeling. Also tranform all factor features to numeric columns for modeling. Then we split dataset with 90% as train dataset and 10% as test dataset.

```r
library(caret)

# Remove id column
datFin=subset(datFin, select = -c(id) )
# Adjust outcome to integer data type
datFin$stroke<- as.integer(as.character(datFin$stroke))
# Adjust factor features to numeric features
datFin<-mutate_if(datFin, is.factor, ~ as.numeric((.x)))

# Using 90% of final dataset as train dataset, 10% of final dataset as validation dataset.
set.seed(123, sample.kind="Rounding")
Validation_index <- createDataPartition(y = datFin$stroke, times = 1, p = 0.1, list = FALSE)
Train <- datFin[-Validation_index,]
Validation <- datFin[Validation_index,]
```

2. Estimate Logistic regression model in unblance dataset in R

```r
# Build GLM model with binary outcome (i.e. Logistic regression model) through train dataset
LogisticModel <- glm(as.factor(stroke) ~ gender+age+hypertension+heart_disease+ever_married+
                       work_type+avg_glucose_level+bmi+smoking_status+Residence_type,
                     data =Train, family = "binomial")
# Calculate the predicted probabilities in the form of P(y=1|x) at validation dataset
LogisticProbabilities <- LogisticModel %>% predict(Validation, type = "response")
# If P(y=1|x) > 0.5 then y = 1 otherwise y=0.
Logisticpredicted <- ifelse(LogisticProbabilities > 0.5,1, 0)
# Using confusionMatix to check results
LogisticFin<-confusionMatrix(as.factor(Logisticpredicted),
                             as.factor(Validation$stroke),
                             positive = "1")

print(LogisticFin)
#> Confusion Matrix and Statistics
#>
#>           Reference
#> Prediction   0   1
#>          0 490  16
#>          1   0   0
#>
#>                Accuracy : 0.9684
#>                  95% CI : (0.9492, 0.9818)
#>     No Information Rate : 0.9684
#>     P-Value [Acc > NIR] : 0.5659710
#>
#>                   Kappa : 0
#>
#>  Mcnemar's Test P-Value : 0.0001768
```

```
#>
#>             Sensitivity : 0.00000
#>             Specificity : 1.00000
#>          Pos Pred Value :     NaN
#>          Neg Pred Value : 0.96838
#>              Prevalence : 0.03162
#>          Detection Rate : 0.00000
#>    Detection Prevalence : 0.00000
#>       Balanced Accuracy : 0.50000
#>
#>        'Positive' Class : 1
#>
```

Through Confusion Matrix and Statistics, we can see there is a very high accuracy rate 96.8%, however with 0 specificity rate and no any stroke case is predicted in validation dataset. It is because stroke dataset is an unbalance dataset. So train dataset need to be adjusted first before modeling method is applied.

3. Adjust unbalance train dataset using SMOTE way in R

Before adjustment, 5% of train dataset are positive stroke cases, matching with whole dataset's proportion. However this is clearly a skewed dataset, causing data bias in prediction model.

```
# Explore proportion and counts of stroke outcome for train dataset
TrainStrokepropInitial <- Train %>%
  count(stroke) %>%            # summarise counts
  mutate(prop = prop.table(n))    # summarise proportion

knitr::kable(as.data.frame(TrainStrokepropInitial),
             caption = "Train dataset stroke outcome cases & proportion before adjustment")
```

Table 3: Train dataset stroke outcome cases & proportion before adjustment

stroke	n	prop
0	4313	0.9487462
1	233	0.0512538

We use SMOTE way to adjust unbalance train dataset, we create extra positive observations, set perc.over = 100 to double the quantity of stroke cases, and set perc.under=200 to keep half of what was created as unstroke cases.

```
# Using SMOTE way to adjust unblanced train dataset
library(DMwR)
Train$stroke<-as.factor(Train$stroke)
trainSOMTE <- SMOTE(stroke ~ ., Train, perc.over = 100, perc.under=200)
```

Through adjustment, we check the stroke outcome balance with proportion and equalize the train data set between stroke and unstroke cases.

```
# Explore proportion and counts of stroke outcome for train dataset after adjustments
TrainStrokepropAdj <- trainSOMTE %>%
  count(stroke) %>%            # summarise counts
  mutate(prop = prop.table(n))    # summarise proportion
```

```
knitr::kable(as.data.frame(TrainStrokepropAdj),
             caption = "Train dataset stroke outcome cases & proportion after adjustment")
```

Table 4: Train dataset stroke outcome cases & proportion after adjustment

stroke	n	prop
0	466	0.5
1	466	0.5

4. Logistic, Random forest, KNN models training in balanced train dataset and validation through confusion matrix in R

We apply logistic regression, random forest or KNN model in adjusted train dataset. In order to prevent overfitting, we set up 10-fold cross validation to train model.

4a. Apply Logistic regression model to balanced train dataset and rebuild data model in R

```
trainSOMTE$stroke<-as.integer(as.character(trainSOMTE$stroke))
set.seed(123)

# Set up 10-fold cross validation in train dataset to prevent Overfitting
train.control <- trainControl(method = "cv", number = 10)
# Apply GLM model with binary outcome (i.e. Logistic regression model) to train dataset
# with 10-fold cross validation
SmoteLogisticModelmodel <- train(form = factor(stroke)~gender+age+hypertension+
                                   heart_disease+ever_married+
                                   work_type+avg_glucose_level+bmi+smoking_status+
                                   Residence_type, data = trainSOMTE,
                                 method = "glm",family = "binomial",
                                 trControl = train.control)
# Explore GLM model with binary outcome
SmoteLogisticModelmodel
#> Generalized Linear Model
#>
#> 932 samples
#>  10 predictor
#>   2 classes: '0', '1'
#>
#> No pre-processing
#> Resampling: Cross-Validated (10 fold)
#> Summary of sample sizes: 839, 839, 839, 838, 838, 839, ...
#> Resampling results:
#>
#>   Accuracy   Kappa
#>   0.7648695  0.5296801
```

```
# Calculate the predicted probabilities at validation dataset
SmoteLogisticProbabilities <- SmoteLogisticModelmodel%>% predict(Validation, type = "prob")
# Look at stroke case outcome, if predicted probabilities > 0.5 then stroke case otherwise un-stroke case
SmoteLogisticpredicted <- ifelse(SmoteLogisticProbabilities[,"1"] > 0.5,1, 0)
# Using confusionMatix to check results
SmoteLogisticFin<-confusionMatrix(relevel(factor(SmoteLogisticpredicted),ref ="1"),
                                  relevel(factor(Validation$stroke), ref ="1"),
                                  positive = "1")
print(SmoteLogisticFin)
#> Confusion Matrix and Statistics
#>
#>           Reference
#> Prediction   1   0
#>          1  13 115
#>          0   3 375
#>
#>                Accuracy : 0.7668
#>                  95% CI : (0.7275, 0.803)
#>     No Information Rate : 0.9684
#>     P-Value [Acc > NIR] : 1
#>
#>                   Kappa : 0.1317
#>
#>  Mcnemar's Test P-Value : <2e-16
#>
#>             Sensitivity : 0.81250
#>             Specificity : 0.76531
#>          Pos Pred Value : 0.10156
#>          Neg Pred Value : 0.99206
#>              Prevalence : 0.03162
#>          Detection Rate : 0.02569
#>    Detection Prevalence : 0.25296
#>       Balanced Accuracy : 0.78890
#>
#>        'Positive' Class : 1
#>
```

4b. Apply Random forest model to balanced train dataset and build data model in R

```
library(randomForest)
# Apply Random forest model to train dataset with 10-fold cross validation
# Tune parameter for Number of variables available for splitting at each tree node.

tunegrid <-expand.grid(.mtry=seq(1,10,1))

SmoteRandomforestModel <- train(form = factor(stroke)~gender+age+hypertension+
                                  heart_disease+ever_married+
                                  work_type+avg_glucose_level+bmi+
                                  smoking_status+Residence_type, data = trainSOMTE,
                                method = "rf",
                                tuneGrid=tunegrid,
                                trControl = train.control)
```

```
# Explore Random forest model
SmoteRandomforestModel
#> Random Forest
#>
#> 932 samples
#>  10 predictor
#>   2 classes: '0', '1'
#>
#> No pre-processing
#> Resampling: Cross-Validated (10 fold)
#> Summary of sample sizes: 838, 839, 839, 839, 840, 839, ...
#> Resampling results across tuning parameters:
#>
#>   mtry  Accuracy   Kappa
#>    1    0.7736164  0.5472384
#>    2    0.8101544  0.6203059
#>    3    0.8122935  0.6245880
#>    4    0.8219135  0.6438156
#>    5    0.8218784  0.6437637
#>    6    0.8176121  0.6352245
#>    7    0.8154613  0.6309618
#>    8    0.8143975  0.6288121
#>    9    0.8068821  0.6137887
#>   10    0.8101193  0.6202433
#>
#> Accuracy was used to select the optimal model using the largest value.
#> The final value used for the model was mtry = 4.

# Calculate the predicted probabilities at validation dataset
SmoteRandomforestProbabilities <- SmoteRandomforestModel %>%
  predict(Validation, type = "prob")

# Look at stroke case outcome, if predicted probabilities > 0.5 then stroke case otherwise un-stroke case
SmoteRandomforestpredicted <- ifelse(SmoteRandomforestProbabilities[,"1"] > 0.5
                                     ,1, 0)
# Using confusionMatix to check results
SmoteRandomforestFin<-confusionMatrix(relevel(factor(SmoteRandomforestpredicted),ref ="1"),
                                      relevel(factor(Validation$stroke),ref ="1"),
                                      positive = "1")

SmoteRandomforestFin
#> Confusion Matrix and Statistics
#>
#>           Reference
#> Prediction   1   0
#>          1  12 130
#>          0   4 360
#>
#>                Accuracy : 0.7352
#>                  95% CI : (0.6944, 0.7731)
#>     No Information Rate : 0.9684
#>     P-Value [Acc > NIR] : 1
#>
#>                   Kappa : 0.1008
#>
#>  Mcnemar's Test P-Value : <2e-16
```

```
#>
#>             Sensitivity : 0.75000
#>             Specificity : 0.73469
#>          Pos Pred Value : 0.08451
#>          Neg Pred Value : 0.98901
#>              Prevalence : 0.03162
#>          Detection Rate : 0.02372
#>    Detection Prevalence : 0.28063
#>       Balanced Accuracy : 0.74235
#>
#>        'Positive' Class : 1
#>
```

4c. Apply KNN model to balanced train dataset and build data model in R

```
library(class)

# Adjust dataset to fit KNN model
# create feature dataset for training model
knnTrain<-subset(trainSOMTE, select = -c(stroke) )
# create feature dataset for validating model
knnValidation<-subset(Validation, select = -c(stroke) )

# Apply KNN model to train dataset with 10-fold cross validation
SmoteknnModel <- train(knnTrain,factor(trainSOMTE$stroke), method = "knn",
                      tuneLength = 10,
                      trControl = train.control)
# Explore Random forest model
SmoteknnModel
#> k-Nearest Neighbors
#>
#> 932 samples
#>  10 predictor
#>   2 classes: '0', '1'
#>
#> No pre-processing
#> Resampling: Cross-Validated (10 fold)
#> Summary of sample sizes: 838, 838, 839, 839, 839, 840, ...
#> Resampling results across tuning parameters:
#>
#>   k   Accuracy   Kappa
#>    5  0.7402883  0.4807049
#>    7  0.7510405  0.5021614
#>    9  0.7660146  0.5321825
#>   11  0.7574590  0.5150702
#>   13  0.7595056  0.5192130
#>   15  0.7573889  0.5148858
#>   17  0.7659684  0.5320400
#>   19  0.7627423  0.5256195
#>   21  0.7627192  0.5255059
#>   23  0.7637718  0.5276325
#>
#> Accuracy was used to select the optimal model using the largest value.
#> The final value used for the model was k = 9.
```

```
# Calculate the predicted probabilities at validation dataset
SmoteknnProbabilities <- SmoteknnModel %>% predict(Validation, type = "prob")

# Look at stroke case outcome, if predicted probabilities > 0.5 then stroke case otherwise un-stroke case
Smoteknnpredicted <- ifelse(SmoteknnProbabilities[,"1"] > 0.5,1, 0)

# Using confusionMatix to check results
SmoteknnFin <-confusionMatrix(relevel(factor(Smoteknnpredicted),ref ="1"),
                              relevel(factor(Validation$stroke),ref ="1"),
                              positive = "1")

SmoteknnFin
#> Confusion Matrix and Statistics
#>
#>           Reference
#> Prediction   1   0
#>          1  14 150
#>          0   2 340
#>
#>                Accuracy : 0.6996
#>                  95% CI : (0.6576, 0.7393)
#>     No Information Rate : 0.9684
#>     P-Value [Acc > NIR] : 1
#>
#>                   Kappa : 0.1039
#>
#>  Mcnemar's Test P-Value : <2e-16
#>
#>             Sensitivity : 0.87500
#>             Specificity : 0.69388
#>          Pos Pred Value : 0.08537
#>          Neg Pred Value : 0.99415
#>              Prevalence : 0.03162
#>          Detection Rate : 0.02767
#>    Detection Prevalence : 0.32411
#>       Balanced Accuracy : 0.78444
#>
#>        'Positive' Class : 1
#>
```

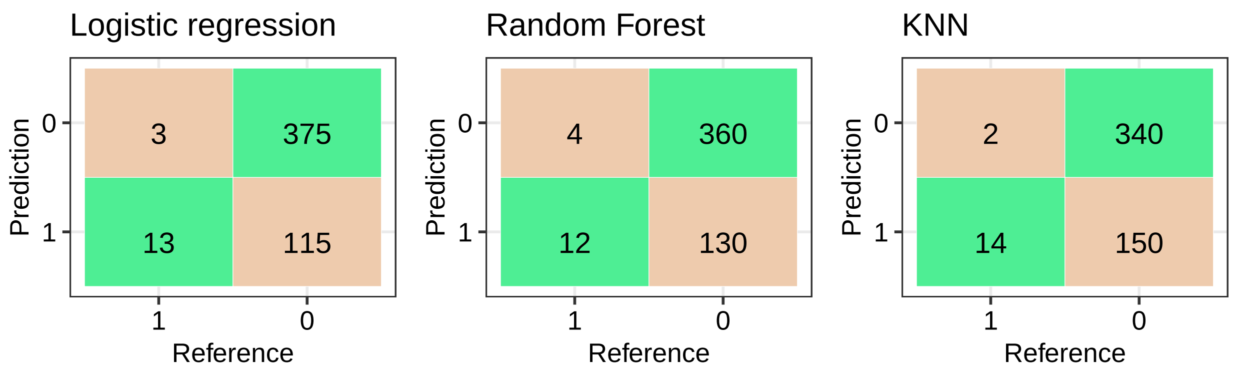
5.Comparison of 3 models:

```
# Build function to visualize confusion matrix table
plot_confusion_matrix <- function(Tb,Tl) {

  ConMT= Tb
  ConMT<-ConMT %>% mutate(colorFlag=ifelse(Prediction==Reference,1,0))

  ComMTPlot= ggplot(data =  ConMT,
                  mapping = aes(x = Reference , y = Prediction )) +
    geom_tile(aes(fill = colorFlag), colour = "white") +
    geom_text(aes(label = sprintf("%1.0f", Freq)), vjust = 1) +
    scale_fill_gradient(low="peachpuff2",high = "seagreen2") +
    theme_bw() + theme(legend.position = "none")+
    theme(axis.text.x = element_text(colour="black",size=10))+
    theme(axis.text.y = element_text(colour="black",size=10))+
    theme(text = element_text(size=10,colour="black"))+
    labs(title=Tl)
}


# visualize confusion matrix tables for 3 models
p1c<-plot_confusion_matrix(as.data.frame(SmoteLogisticFin$table),"Logistic regression")
p2c<-plot_confusion_matrix(as.data.frame(SmoteRandomforestFin$table),"Random Forest")
p3c<-plot_confusion_matrix(as.data.frame(SmoteknnFin$table),"KNN")
grid.arrange(p1c, p2c, p3c, nrow = 1)
```



```

# Build function to get each model's accuracy, F1, sensitivity, specificity
metric_confusion_matrix <- function(cmm,Tl) {

  M<-data_frame(
    Model=Tl,Metric=names(cmm$overall[1]),Value=as.numeric(cmm$overall[1])
  )
  M <- bind_rows(M,data_frame(Model=Tl,Metric=names(cmm$byClass[1]),
                              Value=as.numeric(cmm$byClass[1])))
  M <- bind_rows(M,data_frame(Model=Tl,Metric=names(cmm$byClass[2]),
                              Value=as.numeric(cmm$byClass[2])))
  M <- bind_rows(M,data_frame(Model=Tl,Metric=names(cmm$byClass[7]),
                              Value=as.numeric(cmm$byClass[7])))
}

# Get each model's accuracy, F1, sensitivity, specificity
p1m<-metric_confusion_matrix(SmoteLogisticFin,"Logistic regression")
p2m<-metric_confusion_matrix(SmoteRandomforestFin,"Random Forest")
p3m<-metric_confusion_matrix(SmoteknnFin,"KNN")

# Combine 3 model's results into one table
pm<-bind_rows(p1m,p2m,p3m)
pmprint<-pm %>%spread(Metric, Value)
knitr::kable(pmprint)
```

Model	Accuracy	F1	Sensitivity	Specificity
KNN	0.6996047	0.1555556	0.8750	0.6938776
Logistic regression	0.7667984	0.1805556	0.8125	0.7653061
Random Forest	0.7351779	0.1518987	0.7500	0.7346939

Conclusion

In this project, we use Logistic regression, Random forest and KNN models to predict stroke. Also cross validation way is used to prevent model overfitting and find optimized parameters in model.

From comparison with confusion matrix's parameters, Logistic regression model is the best model for this stroke dataset.

In order to optimize the model further, future investigation can involve to ensemble algorithm together in model.