Stroke Prediction

2024-03-16

#importing the data  
library(tidyverse)

## ── Attaching core tidyverse packages ──────────────────────── tidyverse 2.0.0 ──  
## ✔ dplyr 1.1.4 ✔ readr 2.1.5  
## ✔ forcats 1.0.0 ✔ stringr 1.5.1  
## ✔ ggplot2 3.4.4 ✔ tibble 3.2.1  
## ✔ lubridate 1.9.3 ✔ tidyr 1.3.1  
## ✔ purrr 1.0.2   
## ── Conflicts ────────────────────────────────────────── tidyverse\_conflicts() ──  
## ✖ dplyr::filter() masks stats::filter()  
## ✖ dplyr::lag() masks stats::lag()  
## ℹ Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors

library(dplyr)  
  
df <- read.csv("healthcare-dataset-stroke-data.csv", na.strings = "N/A")  
head(df)

## id gender age hypertension heart\_disease ever\_married work\_type  
## 1 9046 Male 67 0 1 Yes Private  
## 2 51676 Female 61 0 0 Yes Self-employed  
## 3 31112 Male 80 0 1 Yes Private  
## 4 60182 Female 49 0 0 Yes Private  
## 5 1665 Female 79 1 0 Yes Self-employed  
## 6 56669 Male 81 0 0 Yes Private  
## Residence\_type avg\_glucose\_level bmi smoking\_status stroke  
## 1 Urban 228.69 36.6 formerly smoked 1  
## 2 Rural 202.21 NA never smoked 1  
## 3 Rural 105.92 32.5 never smoked 1  
## 4 Urban 171.23 34.4 smokes 1  
## 5 Rural 174.12 24.0 never smoked 1  
## 6 Urban 186.21 29.0 formerly smoked 1

#simple preprocessing  
#removing id column  
df <- df[, !(names(df) %in% "id")]  
colnames(df)

## [1] "gender" "age" "hypertension"   
## [4] "heart\_disease" "ever\_married" "work\_type"   
## [7] "Residence\_type" "avg\_glucose\_level" "bmi"   
## [10] "smoking\_status" "stroke"

df$ever\_married = factor(df$ever\_married)  
df$work\_type = factor(df$work\_type)  
df$Residence\_type = factor(df$Residence\_type)  
df$smoking\_status = factor(df$smoking\_status)

#added later  
df$hypertension <- as.numeric(factor(df$hypertension))  
df$heart\_disease <- as.numeric(factor(df$heart\_disease))

df <- df[df$gender != "Other", ]  
df$gender = factor(df$gender)

#removing row with other- only 1 row for that   
str(df)

## 'data.frame': 5109 obs. of 11 variables:  
## $ gender : Factor w/ 2 levels "Female","Male": 2 1 2 1 1 2 2 1 1 1 ...  
## $ age : num 67 61 80 49 79 81 74 69 59 78 ...  
## $ hypertension : num 1 1 1 1 2 1 2 1 1 1 ...  
## $ heart\_disease : num 2 1 2 1 1 1 2 1 1 1 ...  
## $ ever\_married : Factor w/ 2 levels "No","Yes": 2 2 2 2 2 2 2 1 2 2 ...  
## $ work\_type : Factor w/ 5 levels "children","Govt\_job",..: 4 5 4 4 5 4 4 4 4 4 ...  
## $ Residence\_type : Factor w/ 2 levels "Rural","Urban": 2 1 1 2 1 2 1 2 1 2 ...  
## $ avg\_glucose\_level: num 229 202 106 171 174 ...  
## $ bmi : num 36.6 NA 32.5 34.4 24 29 27.4 22.8 NA 24.2 ...  
## $ smoking\_status : Factor w/ 4 levels "formerly smoked",..: 1 2 2 3 2 1 2 2 4 4 ...  
## $ stroke : int 1 1 1 1 1 1 1 1 1 1 ...

#checking for NA values

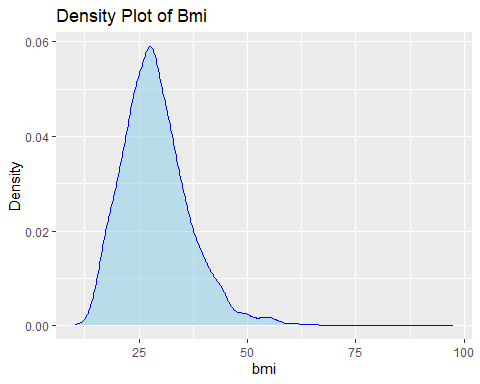
na\_values <- sum(is.na(df))  
na\_values\_per\_column <- colSums(is.na(df))  
na\_values\_per\_column

## gender age hypertension heart\_disease   
## 0 0 0 0   
## ever\_married work\_type Residence\_type avg\_glucose\_level   
## 0 0 0 0   
## bmi smoking\_status stroke   
## 201 0 0

#bmi has 201 na values. Check the distribution to see if mean imputation or meadian imputation should be done

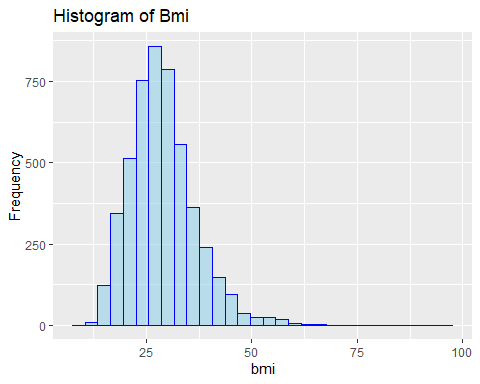
library(ggplot2)  
  
  
# Create density plot  
density\_plot <- ggplot(df, aes(x = bmi)) +  
 geom\_density(fill = "skyblue", color = "blue", alpha = 0.5) +  
 labs(title = "Density Plot of Bmi",  
 x = "bmi",  
 y = "Density")  
  
# Create histogram  
histogram\_plot <- ggplot(df, aes(x = bmi)) +  
 geom\_histogram(fill = "skyblue", color = "blue", alpha = 0.5, bins = 30) +  
 labs(title = "Histogram of Bmi",  
 x = "bmi",  
 y = "Frequency")  
  
# Display both plots  
print(density\_plot)

## Warning: Removed 201 rows containing non-finite values (`stat\_density()`).



print(histogram\_plot)

## Warning: Removed 201 rows containing non-finite values (`stat\_bin()`).



#right skewed so impute with median

median\_bmi <- median(df$bmi, na.rm = TRUE)  
df$bmi[is.na(df$bmi)] <- median\_bmi

#Descriptive statistics

summary(df)

## gender age hypertension heart\_disease ever\_married  
## Female:2994 Min. : 0.08 Min. :1.000 Min. :1.000 No :1756   
## Male :2115 1st Qu.:25.00 1st Qu.:1.000 1st Qu.:1.000 Yes:3353   
## Median :45.00 Median :1.000 Median :1.000   
## Mean :43.23 Mean :1.097 Mean :1.054   
## 3rd Qu.:61.00 3rd Qu.:1.000 3rd Qu.:1.000   
## Max. :82.00 Max. :2.000 Max. :2.000   
## work\_type Residence\_type avg\_glucose\_level bmi   
## children : 687 Rural:2513 Min. : 55.12 Min. :10.30   
## Govt\_job : 657 Urban:2596 1st Qu.: 77.24 1st Qu.:23.80   
## Never\_worked : 22 Median : 91.88 Median :28.10   
## Private :2924 Mean :106.14 Mean :28.86   
## Self-employed: 819 3rd Qu.:114.09 3rd Qu.:32.80   
## Max. :271.74 Max. :97.60   
## smoking\_status stroke   
## formerly smoked: 884 Min. :0.00000   
## never smoked :1892 1st Qu.:0.00000   
## smokes : 789 Median :0.00000   
## Unknown :1544 Mean :0.04874   
## 3rd Qu.:0.00000   
## Max. :1.00000

The summary statistics provided represent data on various attributes for a group of individuals, including gender, age, health conditions, marital status, work type, residence type, average glucose level, body mass index (BMI), smoking status, and stroke occurrence.

The dataset consists of 5,109 individuals, with a gender distribution of 2,994 females and 2,115 males; there are no individuals identified as ‘Other’. Age in the dataset ranges from 0.08 to 82 years, with a median age of 45 years, indicating a middle-aged population. In terms of health conditions, 9.748% have hypertension, and 5.402% have heart disease.

Regarding marital status, 3,353 individuals are married, while 1,756 are not. Work type categories include children (687 individuals), government jobs (657), never worked (22), private sector jobs (2,924), and self-employed (819). The individuals are almost evenly split between rural (2,513) and urban (2,596) residence types.

The average glucose level in the group is 106.14 mg/dL, with a range from 55.12 to 271.74 mg/dL. The BMI values range from 10.3 to 97.6, with a mean of 28.86, indicating that the average individual is overweight according to WHO standards.

As for smoking status, the dataset includes 884 individuals who formerly smoked, 1,892 who never smoked, 789 who currently smoke, and 1,544 whose smoking status is unknown. Lastly, the prevalence of stroke in this population is 4.874%, which aligns with the dataset’s median and mean values being close to 0, suggesting that the majority of individuals have not experienced a stroke.

#checking the class distribution

table(df$stroke)

##   
## 0 1   
## 4860 249

#From all of the above analysis, we can see that the dataset is imbalanced. We will balance the dataset through the process of oversampling.

prop.table(table(df$stroke))

##   
## 0 1   
## 0.95126248 0.04873752

# The process of oversampling - SMOTE

library(ROSE)

## Warning: package 'ROSE' was built under R version 4.3.3

## Loaded ROSE 0.0-4

library(caret)

## Loading required package: lattice

##   
## Attaching package: 'caret'

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

#split the data into training and testing set  
set.seed(25) # for reproducibility  
sample\_size <- floor(0.75 \* nrow(df)) # 75% for training  
train\_indices <- sample(seq\_len(nrow(df)), size = sample\_size)  
  
train\_set <- df[train\_indices, ]  
test\_set <- df[-train\_indices, ]

train\_set$stroke <- factor(train\_set$stroke, levels = c(0, 1))  
test\_set$stroke <- factor(test\_set$stroke, levels = c(0, 1))

balanced\_train\_set <- ovun.sample(stroke ~ ., data = train\_set, method = "over")$data

The oversampling methodology report describes the utilization of the ovun.sample() function from the ROSE package in R to address class imbalance in a dataset, focusing particularly on balancing the ‘stroke’ variable. This approach involves taking the original ‘train\_set’ and applying oversampling to augment the minority class. The function is set with method = “over”, directing it to increase the minority class’s presence by creating synthetic samples that are statistically similar to existing ones, though not identical. The result is the ‘balanced\_train\_set’, a modified version of the original dataset with a better balance between classes, aiming to improve the outcomes of predictive modeling by providing a more equitable data foundation. This method is primarily designed to reduce the skewness in class distribution, thus potentially enhancing the performance of subsequent analyses and model training.

\*\*\*\*\*\*Logistic Regerssion\*\*\*\*\*\*\*\*\*\*\*\*\*

#Fitting model with balanced train

X\_train <- select(balanced\_train\_set, -stroke) # Exclude the response variable  
y\_train <- balanced\_train\_set$stroke # Directly access the column  
  
# Convert y\_train to numeric if it's a factor and not already numeric  
y\_train <- as.numeric(as.character(y\_train))  
  
  
#standardizing the dataset  
preProcValues <- preProcess(X\_train, method = c("center", "scale"))  
X\_train <- predict(preProcValues, X\_train)

# Assuming y\_train is a binary outcome and you're fitting a logistic regression model  
# Here, family = "binomial" tells glm() to fit a logistic regression model  
  
# If X\_train is a matrix, we need to convert it back to a data frame or similar structure that includes column names  
# This step assumes column names were stored or can be recreated. If X\_train was converted to a matrix without  
# preserving column names, adjust as necessary.  
X\_train\_df <- as.data.frame(X\_train) # Convert matrix back to data frame if necessary  
  
# Create a formula for the glm function. This depends on having column names in X\_train\_df.  
# Assuming column names are present and your features are named appropriately in X\_train\_df.  
formula <- as.formula(paste("y\_train ~", paste(colnames(X\_train\_df), collapse = " + ")))  
  
# Fit the model  
glm\_model <- glm(formula, family = "binomial", data = cbind(X\_train\_df, y\_train = y\_train))  
  
# Summary of the model  
summary(glm\_model)

##   
## Call:  
## glm(formula = formula, family = "binomial", data = cbind(X\_train\_df,   
## y\_train = y\_train))  
##   
## Coefficients:  
## Estimate Std. Error z value Pr(>|z|)   
## (Intercept) 0.94711 0.26175 3.618 0.000296 \*\*\*  
## genderMale 0.03677 0.06068 0.606 0.544530   
## age 1.66915 0.05224 31.953 < 2e-16 \*\*\*  
## hypertension 0.19522 0.03061 6.378 1.79e-10 \*\*\*  
## heart\_disease 0.05623 0.03151 1.784 0.074386 .   
## ever\_marriedYes -0.03082 0.09594 -0.321 0.748027   
## work\_typeGovt\_job -0.94111 0.26794 -3.512 0.000444 \*\*\*  
## work\_typeNever\_worked -12.01532 219.09726 -0.055 0.956266   
## work\_typePrivate -0.87900 0.26102 -3.367 0.000759 \*\*\*  
## work\_typeSelf-employed -1.07536 0.27431 -3.920 8.85e-05 \*\*\*  
## Residence\_typeUrban 0.02436 0.05925 0.411 0.680967   
## avg\_glucose\_level 0.19942 0.03196 6.239 4.40e-10 \*\*\*  
## bmi 0.12927 0.03281 3.940 8.14e-05 \*\*\*  
## smoking\_statusnever smoked -0.48338 0.07849 -6.158 7.36e-10 \*\*\*  
## smoking\_statussmokes 0.03112 0.09163 0.340 0.734102   
## smoking\_statusUnknown -0.18427 0.08915 -2.067 0.038744 \*   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## (Dispersion parameter for binomial family taken to be 1)  
##   
## Null deviance: 10151.7 on 7322 degrees of freedom  
## Residual deviance: 7107.5 on 7307 degrees of freedom  
## AIC: 7139.5  
##   
## Number of Fisher Scoring iterations: 13

#Preprocessing the test dataset

X\_test <- select(test\_set, -stroke) # Exclude the target variable  
y\_test <- test\_set$stroke # Access the target column  
  
# Convert y\_test to numeric if it's a factor and not already numeric  
y\_test <- as.numeric(as.character(y\_test))  
  
X\_test\_df <- as.data.frame(X\_test)  
  
X\_test\_df <- predict(preProcValues, X\_test\_df)

#Calculating the accuracy metrics

glm\_probs <- predict(glm\_model, newdata = X\_test\_df, type = "response")  
predicted\_classes <- ifelse(glm\_probs > 0.5, 1, 0)  
table(predicted\_classes, y\_test)

## y\_test  
## predicted\_classes 0 1  
## 0 859 11  
## 1 355 53

mean(predicted\_classes == y\_test)

## [1] 0.713615

#AUC and ROC curves

# Load necessary library  
library(pROC)

## Type 'citation("pROC")' for a citation.

##   
## Attaching package: 'pROC'

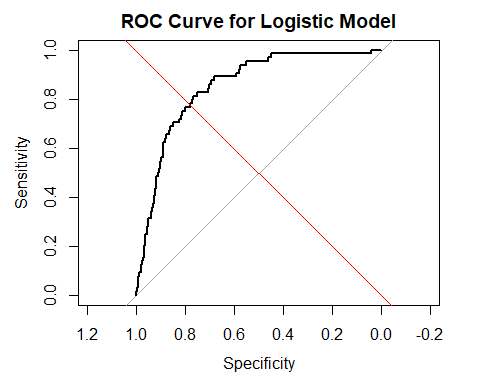
## The following objects are masked from 'package:stats':  
##   
## cov, smooth, var

# Ensure y\_test is in the correct format (needs to be numeric or a factor for pROC)  
# If y\_test is not a numeric vector, convert it from factors or characters to numeric  
y\_test\_numeric <- as.numeric(as.character(unlist(test\_set$stroke))) - 1 # Assuming '1' and '0' are your class labels  
  
# Generate the ROC curve  
roc\_curve <- roc(y\_test\_numeric, as.numeric(glm\_probs))

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

## Setting direction: controls < cases

# Plot the ROC curve  
plot(roc\_curve, main="ROC Curve for Logistic Model")  
abline(a=0, b=1, col="red") # Adds a reference line for random guessing



\*\*\*\*\*\*Using regularised Model (glmnet)\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

library(glmnet)

## Loading required package: Matrix

##   
## Attaching package: 'Matrix'

## The following objects are masked from 'package:tidyr':  
##   
## expand, pack, unpack

## Loaded glmnet 4.1-8

#converting the categorical variables to dummy, that is what glmnet expects   
  
  
X\_train <- select(balanced\_train\_set, -stroke) # Exclude the response variable  
y\_train <- select(balanced\_train\_set, stroke) %>% unlist(.) # Ensure it's a factor if it's not already  
dummyVarsOut = dummyVars(~., data = X\_train, fullRank = FALSE)  
X\_train = predict(dummyVarsOut, X\_train)  
  
#standardize the dataset  
preProcValues <- preProcess(X\_train, method = c("center", "scale"))  
X\_train <- predict(preProcValues, X\_train)

#cross validation 10 fold to find the best alpha values  
set.seed(25)  
K = 10  
trainControl = trainControl(method = "cv", number = K)  
tuneGrid = expand.grid('alpha'=c(0,.25,.5,.75,1),'lambda' = seq(00, .001, length.out = 30))  
elasticOut = train(x = X\_train, y = y\_train,  
method = "glmnet",  
trControl = trainControl, tuneGrid = tuneGrid)

glmnetOut = glmnet(x = X\_train, y = y\_train, alpha = elasticOut$bestTune$alpha,  
family = 'binomial', standardize = FALSE)

#preporcessing the test dataset  
X\_test = select(test\_set, -stroke)  
y\_test = select(test\_set, stroke)  
X\_test <- predict(dummyVarsOut, newdata = X\_test)  
  
X\_test <- predict(preProcValues, X\_test)

#making predictions   
probHatTest = predict(glmnetOut, X\_test, s=elasticOut$bestTune$lambda, type = 'response')  
YhatTestGlmnet = ifelse(probHatTest > 0.5, '1', '0')

mean(YhatTestGlmnet == y\_test)

## [1] 0.7120501

y\_test <- as.character(unlist(y\_test)) # Ensure y\_test is in the same format as YhatTestGlmnet for comparison  
confusion\_matrix <- table(Predicted = YhatTestGlmnet, Actual = y\_test)  
print(confusion\_matrix)

## Actual  
## Predicted 0 1  
## 0 857 11  
## 1 357 53

# Ensure both are factors and have the same levels  
YhatTestGlmnet <- factor(YhatTestGlmnet, levels = c('0', '1'))  
y\_test <- factor(y\_test, levels = c('0', '1'))  
  
# Use confusionMatrix from caret package  
library(caret)  
conf\_matrix <- confusionMatrix(YhatTestGlmnet, y\_test)  
print(conf\_matrix)

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction 0 1  
## 0 857 11  
## 1 357 53  
##   
## Accuracy : 0.7121   
## 95% CI : (0.6864, 0.7368)  
## No Information Rate : 0.9499   
## P-Value [Acc > NIR] : 1   
##   
## Kappa : 0.15   
##   
## Mcnemar's Test P-Value : <2e-16   
##   
## Sensitivity : 0.7059   
## Specificity : 0.8281   
## Pos Pred Value : 0.9873   
## Neg Pred Value : 0.1293   
## Prevalence : 0.9499   
## Detection Rate : 0.6706   
## Detection Prevalence : 0.6792   
## Balanced Accuracy : 0.7670   
##   
## 'Positive' Class : 0   
##

#plot ROC curve and AUC value

# Load necessary library  
library(pROC)  
  
# Ensure y\_test is in the correct format (needs to be numeric or a factor for pROC)  
# If y\_test is not a numeric vector, convert it from factors or characters to numeric  
y\_test\_numeric <- as.numeric(as.character(unlist(y\_test))) - 1 # Assuming '1' and '0' are your class labels  
  
# Generate the ROC curve  
roc\_curve <- roc(y\_test\_numeric, as.numeric(probHatTest))

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

## Setting direction: controls < cases

# Plot the ROC curve  
plot(roc\_curve, main="ROC Curve for glmnet Model")  
abline(a=0, b=1, col="red") # Adds a reference line for random guessing

