R Notebook

This is an [R Markdown](http://rmarkdown.rstudio.com) Notebook. When you execute code within the notebook, the results appear beneath the code.

Try executing this chunk by clicking the *Run* button within the chunk or by placing your cursor inside it and pressing *Ctrl+Shift+Enter*.

if (!require(caret)) install.packages("caret", dependencies=TRUE)

## Loading required package: caret

## Loading required package: ggplot2

## Loading required package: lattice

if (!require(corrplot)) install.packages("corrplot", dependencies=TRUE)

## Loading required package: corrplot

## corrplot 0.95 loaded

library(ggplot2)  
library(dplyr)

##   
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':  
##   
## filter, lag

## The following objects are masked from 'package:base':  
##   
## intersect, setdiff, setequal, union

library(caret)  
library(randomForest)

## randomForest 4.7-1.2

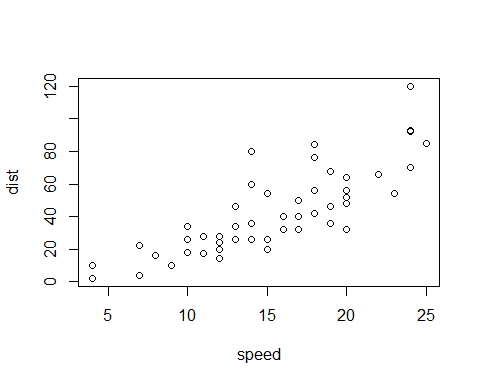
## Type rfNews() to see new features/changes/bug fixes.

##   
## Attaching package: 'randomForest'

## The following object is masked from 'package:dplyr':  
##   
## combine

## The following object is masked from 'package:ggplot2':  
##   
## margin

library(corrplot)  
library(reshape2)  
  
  
  
plot(cars)

 # 1: Data Processing # 2: Exploratory Data Analysis(EDA) # 3: Train-test split # 4: Modeling # 5: Evaluation

# Load the dataset

stroke\_data <- read.csv("healthcare-dataset-stroke-data.csv")

# Remove ‘id’ column as it is not useful for prediction

stroke\_data <- stroke\_data[, !names(stroke\_data) %in% "id"]

# Check for missing values

sum(is.na(stroke\_data))

## [1] 0

# Replace non-numeric values in ‘bmi’ with NA and impute missing values with mean

stroke\_data$bmi <- as.numeric(gsub("[^0-9.]", "", stroke\_data$bmi))  
stroke\_data$bmi[is.na(stroke\_data$bmi)] <- mean(stroke\_data$bmi, na.rm = TRUE)

#Imputing missing values in ‘bmi’ with mean

stroke\_data$bmi[is.na(stroke\_data$bmi)] <- mean(stroke\_data$bmi, na.rm = TRUE)

# Convert categorical variables to factors

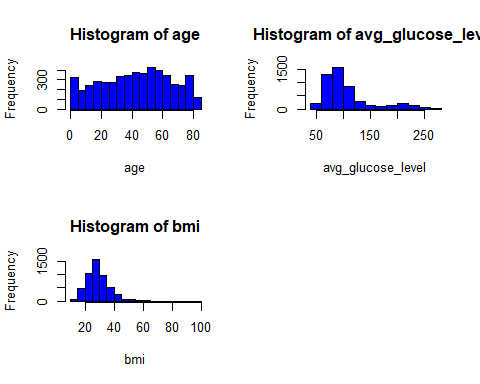
cat\_cols <- c("gender", "hypertension", "heart\_disease", "ever\_married",  
 "work\_type", "Residence\_type", "smoking\_status", "stroke")  
stroke\_data[cat\_cols] <- lapply(stroke\_data[cat\_cols], as.factor)

# Distribution of Numerical Features

numeric\_features <- stroke\_data[, sapply(stroke\_data, is.numeric)]  
par(mfrow = c(2,2))  
lapply(names(numeric\_features), function(col) hist(numeric\_features[[col]],  
 main = paste("Histogram of", col), xlab = col, col = "blue"))

## [[1]]  
## $breaks  
## [1] 0 5 10 15 20 25 30 35 40 45 50 55 60 65 70 75 80 85  
##   
## $counts  
## [1] 320 187 237 281 268 277 329 345 374 365 431 392 339 255 245 349 116  
##   
## $density  
## [1] 0.012524462 0.007318982 0.009275930 0.010998043 0.010489237 0.010841487  
## [7] 0.012876712 0.013502935 0.014637965 0.014285714 0.016868885 0.015342466  
## [13] 0.013268102 0.009980431 0.009589041 0.013659491 0.004540117  
##   
## $mids  
## [1] 2.5 7.5 12.5 17.5 22.5 27.5 32.5 37.5 42.5 47.5 52.5 57.5 62.5 67.5 72.5  
## [16] 77.5 82.5  
##   
## $xname  
## [1] "numeric\_features[[col]]"  
##   
## $equidist  
## [1] TRUE  
##   
## attr(,"class")  
## [1] "histogram"  
##   
## [[2]]  
## $breaks  
## [1] 40 60 80 100 120 140 160 180 200 220 240 260 280  
##   
## $counts  
## [1] 220 1312 1599 860 298 153 85 149 229 150 46 9  
##   
## $density  
## [1] 2.152642e-03 1.283757e-02 1.564579e-02 8.414873e-03 2.915851e-03  
## [6] 1.497065e-03 8.317025e-04 1.457926e-03 2.240705e-03 1.467710e-03  
## [11] 4.500978e-04 8.806262e-05  
##   
## $mids  
## [1] 50 70 90 110 130 150 170 190 210 230 250 270  
##   
## $xname  
## [1] "numeric\_features[[col]]"  
##   
## $equidist  
## [1] TRUE  
##   
## attr(,"class")  
## [1] "histogram"  
##   
## [[3]]  
## $breaks  
## [1] 10 15 20 25 30 35 40 45 50 55 60 65 70 75 80 85 90 95 100  
##   
## $counts  
## [1] 44 493 1070 1610 985 500 253 76 46 20 8 1 1 1 0  
## [16] 0 1 1  
##   
## $density  
## [1] 1.722114e-03 1.929550e-02 4.187867e-02 6.301370e-02 3.855186e-02  
## [6] 1.956947e-02 9.902153e-03 2.974560e-03 1.800391e-03 7.827789e-04  
## [11] 3.131115e-04 3.913894e-05 3.913894e-05 3.913894e-05 0.000000e+00  
## [16] 0.000000e+00 3.913894e-05 3.913894e-05  
##   
## $mids  
## [1] 12.5 17.5 22.5 27.5 32.5 37.5 42.5 47.5 52.5 57.5 62.5 67.5 72.5 77.5 82.5  
## [16] 87.5 92.5 97.5  
##   
## $xname  
## [1] "numeric\_features[[col]]"  
##   
## $equidist  
## [1] TRUE  
##   
## attr(,"class")  
## [1] "histogram"

par(mfrow = c(1,1))

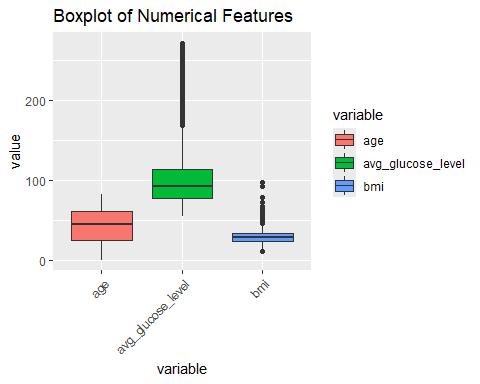


# Convert numeric features to long format explicitly

numeric\_features\_long <- melt(numeric\_features, id.vars = NULL, measure.vars = names(numeric\_features))

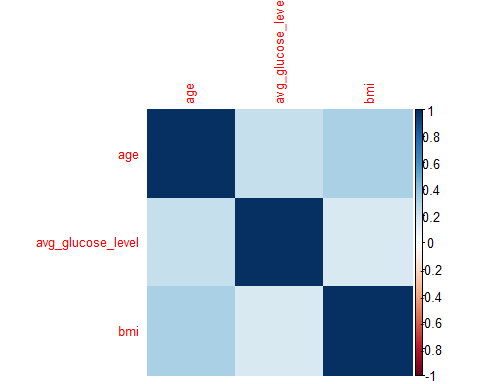
# Boxplot for Outlier Detection

ggplot(numeric\_features\_long, aes(x = variable, y = value)) +  
 geom\_boxplot(aes(fill = variable)) +  
 theme(axis.text.x = element\_text(angle = 45, hjust = 1)) +  
 ggtitle("Boxplot of Numerical Features")



# Correlation matrix

cor\_matrix <- cor(numeric\_features, use = "complete.obs")  
corrplot(cor\_matrix, method = "color", tl.cex = 0.8)



# Outlier Detection and Removal (Using IQR)

outlier\_removal <- function(x) {  
 Q1 <- quantile(x, 0.25, na.rm = TRUE)  
 Q3 <- quantile(x, 0.75, na.rm = TRUE)  
 IQR <- Q3 - Q1  
 x[x < (Q1 - 1.5 \* IQR) | x > (Q3 + 1.5 \* IQR)] <- NA  
 return(x)  
}  
  
stroke\_data[names(numeric\_features)] <- lapply(stroke\_data[names(numeric\_features)], outlier\_removal)  
stroke\_data <- na.omit(stroke\_data)

# Standardization (Scaling Numeric Features)

stroke\_data[names(numeric\_features)] <- scale(stroke\_data[names(numeric\_features)])

# Splitting data into training and testing sets

set.seed(123)  
train\_index <- createDataPartition(stroke\_data$stroke, p = 0.8, list = FALSE)  
train\_data <- stroke\_data[train\_index, ]  
test\_data <- stroke\_data[-train\_index, ]

# Logistic Regression Model

stroke\_model <- glm(stroke ~ ., data = train\_data, family = binomial)  
pred\_probs <- predict(stroke\_model, test\_data, type = "response")  
pred\_labels <- ifelse(pred\_probs > 0.5, 1, 0)  
conf\_matrix <- confusionMatrix(factor(pred\_labels, levels = levels(test\_data$stroke)), test\_data$stroke)  
print(conf\_matrix)

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction 0 1  
## 0 845 33  
## 1 0 0  
##   
## Accuracy : 0.9624   
## 95% CI : (0.9476, 0.974)  
## No Information Rate : 0.9624   
## P-Value [Acc > NIR] : 0.5461   
##   
## Kappa : 0   
##   
## Mcnemar's Test P-Value : 2.54e-08   
##   
## Sensitivity : 1.0000   
## Specificity : 0.0000   
## Pos Pred Value : 0.9624   
## Neg Pred Value : NaN   
## Prevalence : 0.9624   
## Detection Rate : 0.9624   
## Detection Prevalence : 1.0000   
## Balanced Accuracy : 0.5000   
##   
## 'Positive' Class : 0   
##

# Decision Tree Model

tree\_model <- train(stroke ~ ., data = train\_data, method = "rpart")  
tree\_pred <- predict(tree\_model, test\_data)  
tree\_conf\_matrix <- confusionMatrix(tree\_pred, test\_data$stroke)  
print(tree\_conf\_matrix)

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction 0 1  
## 0 845 33  
## 1 0 0  
##   
## Accuracy : 0.9624   
## 95% CI : (0.9476, 0.974)  
## No Information Rate : 0.9624   
## P-Value [Acc > NIR] : 0.5461   
##   
## Kappa : 0   
##   
## Mcnemar's Test P-Value : 2.54e-08   
##   
## Sensitivity : 1.0000   
## Specificity : 0.0000   
## Pos Pred Value : 0.9624   
## Neg Pred Value : NaN   
## Prevalence : 0.9624   
## Detection Rate : 0.9624   
## Detection Prevalence : 1.0000   
## Balanced Accuracy : 0.5000   
##   
## 'Positive' Class : 0   
##

# Random Forest Model

rf\_model <- randomForest(stroke ~ ., data = train\_data, ntree = 100)  
rf\_pred <- predict(rf\_model, test\_data)  
rf\_conf\_matrix <- confusionMatrix(rf\_pred, test\_data$stroke)  
print(rf\_conf\_matrix)

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction 0 1  
## 0 845 32  
## 1 0 1  
##   
## Accuracy : 0.9636   
## 95% CI : (0.9489, 0.9749)  
## No Information Rate : 0.9624   
## P-Value [Acc > NIR] : 0.4755   
##   
## Kappa : 0.0567   
##   
## Mcnemar's Test P-Value : 4.251e-08   
##   
## Sensitivity : 1.0000   
## Specificity : 0.0303   
## Pos Pred Value : 0.9635   
## Neg Pred Value : 1.0000   
## Prevalence : 0.9624   
## Detection Rate : 0.9624   
## Detection Prevalence : 0.9989   
## Balanced Accuracy : 0.5152   
##   
## 'Positive' Class : 0   
##