# Load required libraries

library(tidyverse)

library(dplyr)

library(ggplot2)

library(gridExtra)

# Set working directory and load the CSV file

setwd(dirname(file.choose()))

getwd()

Stroke\_data <- read.csv("Stroke\_data.csv", stringsAsFactors = FALSE)

# Print the structure of Stroke\_data

str(Stroke\_data)

# Check for missing values:

sum(is.na(Stroke\_data))

colSums(is.na(Stroke\_data))

colSums(Stroke\_data == "N/A")

# Check for duplicate values

n\_occur <- data.frame(table(Stroke\_data$pid))

n\_occur[n\_occur$Freq > 1,]

sum(Stroke\_data$gender == "Other")

sum(Stroke\_data$smoking == "Unknown")

# Visualization

# Create and arrange multiple plots using grid.arrange

p1 <- Stroke\_data %>%

ggplot(aes(x = gender, fill = risk)) +

geom\_bar()

p2 <- Stroke\_data %>%

ggplot(aes(x = nhiss, fill = risk)) +

geom\_bar()

p3 <- Stroke\_data %>%

ggplot(aes(x = mrs, fill = risk)) +

geom\_bar()

p4 <- Stroke\_data %>%

ggplot(aes(x = systolic, fill = risk)) +

geom\_bar()

p5 <- Stroke\_data %>%

ggplot(aes(x = distolic, fill = risk)) +

geom\_bar()

p6 <- Stroke\_data %>%

ggplot(aes(x = glucose, fill = risk)) +

geom\_bar()

p7 <- Stroke\_data %>%

ggplot(aes(x = paralysis, fill = risk)) +

geom\_bar()

p8 <- Stroke\_data %>%

ggplot(aes(x = smoking, fill = risk)) +

geom\_bar()

p9 <- Stroke\_data %>%

ggplot(aes(x = cholestrol, fill = risk)) +

geom\_bar()

p10 <- Stroke\_data %>%

ggplot(aes(x = tos, fill = risk)) +

geom\_bar()

grid.arrange(grobs = list(p1, p2),

ncol = 2,

top = "Count of Levels for Each Factor"

)

grid.arrange(grobs = list(p3, p4),

ncol = 2,

top = "Count of Levels for Each Factor"

)

grid.arrange(grobs = list(p5, p6),

ncol = 2,

top = "Count of Levels for Each Factor"

)

grid.arrange(grobs = list(p7, p8),

ncol = 2,

top = "Count of Levels for Each Factor"

)

grid.arrange(grobs = list(p9, p10),

ncol = 2,

top = "Count of Levels for Each Factor"

)

# Create and arrange multiple plots using grid.arrange

dat\_prop <- Stroke\_data %>%

group\_by(gender) %>%

summarise(prop = sum(risk == "1")/length(gender))

p1 <- dat\_prop %>%

ggplot(aes(x = gender, y = prop)) +

geom\_col(fill = "#00BFC4")

dat\_prop <- Stroke\_data %>%

group\_by(nhiss) %>%

summarise(prop = sum(risk == "1")/length(nhiss))

p2 <- dat\_prop %>%

ggplot(aes(x = nhiss, y = prop)) +

geom\_col(fill = "#00BFC4")

dat\_prop <- Stroke\_data %>%

group\_by(mrs) %>%

summarise(prop = sum(risk == "1")/length(mrs))

p3 <- dat\_prop %>%

ggplot(aes(x = mrs, y = prop)) +

geom\_col(fill = "#00BFC4")

dat\_prop <- Stroke\_data %>%

group\_by(systolic) %>%

summarise(prop = sum(risk == "1")/length(systolic))

p4 <- dat\_prop %>%

ggplot(aes(x = systolic, y = prop)) +

geom\_col(fill = "#00BFC4")

dat\_prop <- Stroke\_data %>%

group\_by(distolic) %>%

summarise(prop = sum(risk == "1")/length(distolic))

p5 <- dat\_prop %>%

ggplot(aes(x = distolic, y = prop)) +

geom\_col(fill = "#00BFC4")

dat\_prop <- Stroke\_data %>%

group\_by(glucose) %>%

summarise(prop = sum(risk == "1")/length(glucose))

p6 <- dat\_prop %>%

ggplot(aes(x = glucose, y = prop)) +

geom\_col(fill = "#00BFC4")

dat\_prop <- Stroke\_data %>%

group\_by(paralysis) %>%

summarise(prop = sum(risk == "1")/length(paralysis))

p7 <- dat\_prop %>%

ggplot(aes(x = paralysis, y = prop)) +

geom\_col(fill = "#00BFC4")

dat\_prop <- Stroke\_data %>%

group\_by(smoking) %>%

summarise(prop = sum(risk == "1")/length(smoking))

p8 <- dat\_prop %>%

ggplot(aes(x = smoking, y = prop)) +

geom\_col(fill = "#00BFC4")

dat\_prop <- Stroke\_data %>%

group\_by(cholestrol) %>%

summarise(prop = sum(risk == "1")/length(cholestrol))

p9 <- dat\_prop %>%

ggplot(aes(x = cholestrol, y = prop)) +

geom\_col(fill = "#00BFC4")

dat\_prop <- Stroke\_data %>%

group\_by(tos) %>%

summarise(prop = sum(risk == "1")/length(tos))

p10 <- dat\_prop %>%

ggplot(aes(x = tos, y = prop)) +

geom\_col(fill = "#00BFC4")

grid.arrange(grobs = list(p1, p2),

ncol = 2,

top = "Proportion of Strokes\_Risk for Each Factor"

)

grid.arrange(grobs = list(p3, p4),

ncol = 2,

top = "Proportion of Strokes\_Risk for Each Factor"

)

grid.arrange(grobs = list(p5, p6),

ncol = 2,

top = "Proportion of Strokes\_Risk for Each Factor"

)

grid.arrange(grobs = list(p7, p8),

ncol = 2,

top = "Proportion of Strokes\_Risk for Each Factor"

)

grid.arrange(grobs = list(p9, p10),

ncol = 2,

top = "Proportion of Strokes\_Risk for Each Factor"

)

# Create and arrange multiple plots using grid.arrange

p1 <- Stroke\_data %>%

ggplot(aes(x = gender, y = age, color = risk)) +

geom\_boxplot() +

theme(legend.position="none")

p2 <- Stroke\_data %>%

ggplot(aes(x = nhiss, y = age, color = risk)) +

geom\_boxplot() +

theme(legend.position="none")

p3 <- Stroke\_data %>%

ggplot(aes(x = mrs, y = age, color = risk)) +

geom\_boxplot() +

theme(legend.position="none")

p4 <- Stroke\_data %>%

ggplot(aes(x = systolic, y = age, color = risk)) +

geom\_boxplot() +

theme(legend.position="none")

p5 <- Stroke\_data %>%

ggplot(aes(x = distolic, y = age, color = risk)) +

geom\_boxplot() +

theme(legend.position="none")

p6 <- Stroke\_data %>%

ggplot(aes(x = glucose, y = age, color = risk)) +

geom\_boxplot() +

theme(legend.position="none")

p7 <- Stroke\_data %>%

ggplot(aes(x = paralysis, y = age, color = risk)) +

geom\_boxplot() +

theme(legend.position="none")

p8 <- Stroke\_data %>%

ggplot(aes(x = smoking, y = age, color = risk)) +

geom\_boxplot() +

theme(legend.position="none")

p9 <- Stroke\_data %>%

ggplot(aes(x = cholestrol, y = age, color = risk)) +

geom\_boxplot() +

theme(legend.position="none")

p10 <- Stroke\_data %>%

ggplot(aes(x = tos, y = age, color = risk)) +

geom\_boxplot() +

theme(legend.position="none")

grid.arrange(grobs = list(p1, p2),

ncol = 2,

top = "Risk and Age Across Factors"

)

grid.arrange(grobs = list(p3, p4),

ncol = 2,

top = "Risk and Age Across Factors"

)

grid.arrange(grobs = list(p5, p6),

ncol = 2,

top = "Risk and Age Across Factors"

)

grid.arrange(grobs = list(p7, p8),

ncol = 2,

top = "Risk and Age Across Factors"

)

grid.arrange(grobs = list(p9, p10),

ncol = 2,

top = "Risk and Age Across Factors"

)

# Create and arrange multiple plots using grid.arrange

p1 <- Stroke\_data %>%

ggplot(aes(x = gender, y = glucose, color = risk)) +

geom\_boxplot() +

theme(legend.position="none")

p2 <- Stroke\_data %>%

ggplot(aes(x = nhiss, y = glucose, color = risk)) +

geom\_boxplot() +

theme(legend.position="none")

p3 <- Stroke\_data %>%

ggplot(aes(x = mrs, y = glucose, color = risk)) +

geom\_boxplot() +

theme(legend.position="none")

p4 <- Stroke\_data %>%

ggplot(aes(x = systolic, y = glucose, color = risk)) +

geom\_boxplot() +

theme(legend.position="none")

p5 <- Stroke\_data %>%

ggplot(aes(x = distolic, y = glucose, color = risk)) +

geom\_boxplot() +

theme(legend.position="none")

p6 <- Stroke\_data %>%

ggplot(aes(x = paralysis, y = glucose, color = risk)) +

geom\_boxplot() +

theme(legend.position="none")

p7 <- Stroke\_data %>%

ggplot(aes(x = smoking, y = glucose, color = risk)) +

geom\_boxplot() +

theme(legend.position="none")

p8 <- Stroke\_data %>%

ggplot(aes(x = tos, y = glucose, color = risk)) +

geom\_boxplot() +

theme(legend.position="none")

grid.arrange(grobs = list(p1, p2),

ncol = 2,

top = "Risk and Glucose Level Across Factors"

)

grid.arrange(grobs = list(p3, p4),

ncol = 2,

top = "Risk and Glucose Level Across Factors"

)

grid.arrange(grobs = list(p5, p6),

ncol = 2,

top = "Risk and Glucose Level Across Factors"

)

grid.arrange(grobs = list(p7, p8),

ncol = 2,

top = "Risk and Glucose Level Across Factors"

)

# Create and arrange multiple plots using grid.arrange

p1 <- Stroke\_data %>%

ggplot(aes(x = gender, y = bmi, color = risk)) +

geom\_boxplot() +

theme(legend.position="none")

p2 <- Stroke\_data %>%

ggplot(aes(x = nhiss, y = bmi, color = risk)) +

geom\_boxplot() +

theme(legend.position="none")

p3 <- Stroke\_data %>%

ggplot(aes(x = mrs, y = bmi, color = risk)) +

geom\_boxplot() +

theme(legend.position="none")

p4 <- Stroke\_data %>%

ggplot(aes(x = systolic, y = bmi, color = risk)) +

geom\_boxplot() +

theme(legend.position="none")

p5 <- Stroke\_data %>%

ggplot(aes(x = distolic, y = bmi, color = risk)) +

geom\_boxplot() +

theme(legend.position="none")

p6 <- Stroke\_data %>%

ggplot(aes(x = glucose, y = bmi, color = risk)) +

geom\_boxplot() +

theme(legend.position="none")

p7 <- Stroke\_data %>%

ggplot(aes(x = paralysis, y = bmi, color = risk)) +

geom\_boxplot() +

theme(legend.position="none")

p8 <- Stroke\_data %>%

ggplot(aes(x = smoking, y = bmi, color = risk)) +

geom\_boxplot() +

theme(legend.position="none")

p9 <- Stroke\_data %>%

ggplot(aes(x = cholestrol, y = bmi, color = risk)) +

geom\_boxplot() +

theme(legend.position="none")

p10 <- Stroke\_data %>%

ggplot(aes(x = tos, y = bmi, color = risk)) +

geom\_boxplot() +

theme(legend.position="none")

grid.arrange(grobs = list(p1, p2),

ncol = 2,

top = "Stroke\_Risk and BMI Across Factors"

)

grid.arrange(grobs = list(p3, p4),

ncol = 2,

top = "Stroke\_Risk and BMI Across Factors"

)

grid.arrange(grobs = list(p5, p6),

ncol = 2,

top = "Stroke\_Risk and BMI Across Factors"

)

grid.arrange(grobs = list(p7, p8),

ncol = 2,

top = "Stroke\_Risk and BMI Across Factors"

)

grid.arrange(grobs = list(p9, p10),

ncol = 2,

top = "Stroke\_Risk and BMI Across Factors"

)

# Create and arrange multiple plots using grid.arrange

p1 <- Stroke\_data %>%

ggplot(aes(x = age, fill = risk)) +

geom\_density(alpha = 0.5) +

theme(legend.position="none")

p2 <- Stroke\_data %>%

ggplot(aes(x = glucose, fill = risk)) +

geom\_density(alpha = 0.5) +

theme(legend.position="none")

p3 <- Stroke\_data %>%

ggplot(aes(x = bmi, fill = risk)) +

geom\_density(alpha = 0.5) +

theme(legend.position="none")

p4 <- Stroke\_data %>%

ggplot(aes(x = age, fill = risk)) +

geom\_histogram() +

theme(legend.position="none")

p5 <- Stroke\_data %>%

ggplot(aes(x = glucose, fill = risk)) +

geom\_histogram() +

theme(legend.position="none")

p6 <- Stroke\_data %>%

ggplot(aes(x = bmi, fill = risk)) +

geom\_histogram() +

theme(legend.position="none")

grid.arrange(grobs = list(p1, p2, p3,

p4, p5, p6),

nrow = 2,

top = "Distribution of Continuous Variables"

)

# Create a new data frame called ischemic\_stroke with selected columns from Stroke\_data

ischemic\_stroke <- data.frame(Stroke\_data$sn.no, Stroke\_data$age, Stroke\_data$nhiss,Stroke\_data$mrs, Stroke\_data$systolic, Stroke\_data$distolic, Stroke\_data$glucose,Stroke\_data$paralysis, Stroke\_data$smoking,Stroke\_data$bmi, Stroke\_data$cholestrol,Stroke\_data$tos, Stroke\_data$risk)

# Rename the column names in the ischemic\_stroke data frame

colnames(ischemic\_stroke) <- c("SN.No","Age","NHISS","MRS","Systolic","Distolic","Glucose","Paralysis","Smoking","BMI","Cholestrol","TOS","Risk")

# Display the structure of the ischemic\_stroke data frame

str(ischemic\_stroke)

# View the ischemic\_stroke data frame

View(ischemic\_stroke)

# Create a boxplot for the ischemic\_stroke data frame

boxplot(ischemic\_stroke,main="Boxplot for Stroke Prediction",xlab="Dependent and Independant Variables",ylab="count",col="Bisque")

# Define a function called replace\_outlier that replaces extreme values with percentiles

replace\_outlier <- function(x){

for (i in which(sapply(x, is.numeric))) {

quantiles <- quantile( x[,i], c(.05, .95 ), na.rm =TRUE)

x[,i] = ifelse(x[,i] < quantiles[1] , quantiles[1], x[,i])

x[,i] = ifelse(x[,i] > quantiles[2] , quantiles[2], x[,i])}

x}

# Replace the outliers in the ischemic\_stroke data frame using the replace\_outlier function

ischemic\_stroke = replace\_outlier(ischemic\_stroke)

# Create a boxplot for the ischemic\_stroke data frame with outliers replaced

boxplot(ischemic\_stroke, xlab="numerical values", ylab="Count", col="Bisque", main="Outliers Replaced :")

#Normalization (3 Methods Used):

#1st Method min-max scaling:

ischemic\_stroke.mms<- apply(ischemic\_stroke , MARGIN = 2, FUN = function(x) (x - min(x))/diff(range(x)))

boxplot(ischemic\_stroke.mms, main= "Min-Max Scaling ",xlab="numerical values",ylab="count")

#2nd Method z-score:

ischemic\_stroke.z1 <- apply(ischemic\_stroke, MARGIN = 2, FUN = function(x) (x - mean(x))/sd(x))

ischemic\_stroke.z2 <- apply(ischemic\_stroke, MARGIN = 2, FUN = function(x) (x - mean(x))/(2\*sd(x)))

boxplot(ischemic\_stroke.z1,main= "Standard deviation 1",xlab="numerical values",ylab="count")

boxplot(ischemic\_stroke.z2,main= "Standard deviation 2",xlab="numerical values",ylab="count")

#3rd Method soft Max Scaling:

library(DMwR2)

help(SoftMax)

sts <- apply(ischemic\_stroke, MARGIN = 2, FUN = function(x) (SoftMax(x,lambda = 1, mean(x), sd(x))))

boxplot (sts, main = "Soft Max, lambda = 1")

sts <- apply(ischemic\_stroke, MARGIN = 2, FUN = function(x) (SoftMax(x,lambda = 2, mean(x), sd(x))))

boxplot (sts, main = "Soft Max, lambda = 2")

sts <- apply(ischemic\_stroke, MARGIN = 2, FUN = function(x) (SoftMax(x,lambda = 3, mean(x), sd(x))))

boxplot (sts, main = "Soft Max, lambda = 3")

sts <- apply(ischemic\_stroke, MARGIN = 2, FUN = function(x) (SoftMax(x,lambda = 4, mean(x), sd(x))))

boxplot (sts, main = "Soft Max, lambda = 4")

sts <- apply(ischemic\_stroke, MARGIN = 2, FUN = function(x) (SoftMax(x,lambda = 5, mean(x), sd(x))))

boxplot (sts, main = "Soft Max, lambda = 5")

#Checking the Given Variables are Normally Distributed through Ks test:

#If norm > 0.05 then it's Normally Distributed.

ks.test(ischemic\_stroke$SN.No,"pnorm", mean(ischemic\_stroke$SN.No), sd(ischemic\_stroke$SN.No))

ks.test(ischemic\_stroke$Age,"pnorm", mean(ischemic\_stroke$Age), sd(ischemic\_stroke$Age))

ks.test(ischemic\_stroke$NHISS,"pnorm", mean(ischemic\_stroke$NHISS), sd(ischemic\_stroke$NHISS))

ks.test(ischemic\_stroke$Systolic,"pnorm", mean(ischemic\_stroke$Systolic), sd(ischemic\_stroke$Systolic))

ks.test(ischemic\_stroke$Glucose,"pnorm", mean(ischemic\_stroke$Glucose), sd(ischemic\_stroke$Glucose))

ks.test(ischemic\_stroke$Smoking,"pnorm", mean(ischemic\_stroke$Smoking), sd(ischemic\_stroke$Smoking))

ks.test(ischemic\_stroke$Cholestrol,"pnorm", mean(ischemic\_stroke$Cholestrol), sd(ischemic\_stroke$Cholestrol))

ks.test(ischemic\_stroke$Risk,"pnorm", mean(ischemic\_stroke$Risk), sd(ischemic\_stroke$Risk))

# Correlations among numeric variables

cor.matrix <- cor(ischemic\_stroke, use = "pairwise.complete.obs", method = "pearson")

round(cor.matrix, digits = 2)

cor.df <- as.data.frame(cor.matrix)

View(cor.df)

round(cor.df,2)

#Co-relation Map

library(corrgram)

corrgram(ischemic\_stroke, order=FALSE, cor.method = "pearson", lower.panel=panel.cor,

upper.panel=panel.pie, text.panel=panel.txt, main="Patients with Ischemic Stroke (pearson correlation)")

#Partial correlation

library(ppcor)

#calculate partial correlation using Pearson

pcor.test(ischemic\_stroke$Age,ischemic\_stroke$Risk,ischemic\_stroke$NHISS)

pcor.test(ischemic\_stroke$Age,ischemic\_stroke$NHISS,ischemic\_stroke$Risk)

#Kaiser-Meyer-Olkin statistics: if overall MSA > 0.6, proceed to factor analysis

library(psych)

KMO(cor(ischemic\_stroke))

#Calculate KMO statistic

KMO\_result <- KMO(ischemic\_stroke)

#Check if overall MSA is greater than 0.6

if(KMO\_result$MSA > 0.6){

#Proceed to factor analysis

fa\_result <- fa(ischemic\_stroke)

print(fa\_result)

} else {

print("Overall measure of sampling adequacy is not greater than 0.6. Factor analysis cannot be performed.")

}

# Test dependent variable for normality

# graphically

qqnorm(ischemic\_stroke$Age, xlab = "Theoretical Quantiles: Age Factor in Ischemic Stroke" )

qqline(ischemic\_stroke$Age,col = 2) ## red color

# K-S test

ks.test(ischemic\_stroke$Age, "pnorm", mean(ischemic\_stroke$Age), "pnorm"(ischemic\_stroke$Risk))

#Machine Learning Technique

#Setting seed and creating a sample for training and testing data

set.seed(1)

sample <- sample(nrow(ischemic\_stroke), floor(0.75 \* nrow(ischemic\_stroke)))

train\_data <- ischemic\_stroke[sample,]

test\_data <- ischemic\_stroke[-sample,]

#Loading required libraries

library(randomForest)

library(e1071)

library(e1071)

library(glmnet)

library(rpart)

library(ggplot2)

#Random Forest

set.seed(123)

rf\_model <- randomForest( Risk ~ ., data = train\_data, importance = TRUE)

rf\_model

#Predict on test data

rf\_predictions <- predict(rf\_model, newdata = test\_data)

rf\_confusion\_matrix <- table(rf\_predictions, test\_data$Risk)

rf\_confusion\_matrix

#Gaussian Naive Bayes

nb\_model <- naiveBayes(Risk ~ ., data = train\_data)

nb\_model

#Predict on test data

nb\_predictions <- predict(nb\_model, newdata = test\_data)

nb\_confusion\_matrix <- table(nb\_predictions, test\_data$Risk)

nb\_confusion\_matrix

#Support Vector Machine

svm\_model <- svm(Risk ~ ., data = train\_data)

svm\_model

#Predict on test data

svm\_predictions <- predict(svm\_model, newdata = test\_data)

svm\_confusion\_matrix <- table(svm\_predictions, test\_data$Risk)

svm\_confusion\_matrix

#Logistic Regression

set.seed(123)

x <- model.matrix(Risk ~ ., data = train\_data)[,-1]

y <- train\_data$Risk

cv\_model <- cv.glmnet(x, y, family = "multinomial")

lambda.min <- cv\_model$lambda.min

logit\_model <- glmnet(x, y, family = "multinomial", alpha = 1, lambda = lambda.min)

logit\_model

#Predict on test data

logit\_predictions <- predict(logit\_model, newx = model.matrix(Risk ~ ., data = test\_data)[,-1], type = "response")

logit\_predictions[logit\_predictions >= 0.5] <- 1

logit\_predictions[logit\_predictions < 0.5] <- 0

# Make sure both vectors have the same length

if (length(logit\_predictions) != length(test\_data$Risk)) {

logit\_predictions <- logit\_predictions[1:length(test\_data$Risk)]

}

logit\_confusion\_matrix <- table(logit\_predictions, test\_data$Risk)

logit\_confusion\_matrix

#Decision Tree

set.seed(123)

tree\_model <- rpart(Risk ~ ., data = train\_data, method = "class")

tree\_model

plot(tree\_model)

text(tree\_model, use.n = TRUE, all = TRUE, cex = 0.8)

tree\_predictions <- predict(tree\_model, newdata = test\_data, type = "class")

tree\_confusion\_matrix <- table(tree\_predictions, test\_data$Risk)

tree\_confusion\_matrix

#Comparison of all models

accuracy <- c(

rf\_acc = sum(diag(rf\_confusion\_matrix))/sum(rf\_confusion\_matrix),

nb\_acc = sum(diag(nb\_confusion\_matrix))/sum(nb\_confusion\_matrix),

svm\_acc = sum(diag(svm\_confusion\_matrix))/sum(svm\_confusion\_matrix),

logit\_acc = sum(diag(logit\_confusion\_matrix))/sum(logit\_confusion\_matrix),

tree\_acc = sum(diag(tree\_confusion\_matrix))/sum(tree\_confusion\_matrix)

)

accuracy

#Round the accuracy to 3 decimal points

accuracy <- round(accuracy, 3)

accuracy

#Plotting accuracy for all models

accuracy\_df <- data.frame(model = names(accuracy), accuracy = accuracy)

ggplot(data = accuracy\_df, aes(x = model, y = accuracy)) +

geom\_bar(stat = "identity", fill = "steelblue") +

geom\_text(aes(label = round(accuracy, 3)), vjust = -0.5) +

ggtitle("Accuracy of Different Machine Learning Models")