# Title: ANALYSIS OF CORONARY HEART DISEASE PREDICTION USING MACHINE LANGUAGE

#Installing Packages

install.packages("kableExtra") # For formatting Tables using Knit package

install.packages("tidyverse") # For data wrangling and visualization

install.packages("caret") # For Model Training

install.packages("GGally") # For pair plots

install.packages("e1071") # For statistical Learning

install.packages("tidyr") # For Tidying messy data

install.packages("cowplot") # For plot manipulation

install.packages("lattice") # For creating various types of plots

install.packages("ggcorrplot") # For correlation of plots

install.packages("corrplot") # For visualizing correlation matrices

install.packages("purrr") # For simplifying and enhancing vectors

install.packages("pkgsearch") # For searching packages

install.packages("ROCR") # For visualizing and evaluating the performance of classifications

install.packages("cvAUC") # For using cross-Validation

install.packages("ranger") # For classification and regression tasks

install.packages("dplyr") # For changing factors

install.packages("tictoc") # For measuring time taken for code execution

install.packages("ggplot2") # For plotting graphs

install.packages("knitr") # For flexible formatting

install.packages("kernlab") # For implementing Support Vector Machines (SVM)

install.packages("DataExplore") # For automating data handling and visualization

install.packages("randomForest") # For data classification and regression

install.packages("gridExtra") # For useful extensions to work with grid graphics

install.packages("yardstick")

install.packages("remotes")

remotes::install\_github("cran/DMwR")

# Loading Packages

library(kableExtra)

library(tidyverse)

library(caret)

library(GGally)

library(e1071)

library(tidyr)

library(cowplot)

library(lattice)

library(ggcorrplot)

library(corrplot)

library(purrr)

library(pkgsearch)

library(ROCR)

library(cvAUC)

library(ranger)

library(dplyr)

library(tictoc)

library(ggplot2)

library(knitr)

library(kernlab)

library(DataExplorer)

library(randomForest)

library(gridExtra)

library(yardstick)

library(DMwR)

# Setting working directory

getwd()

setwd(dirname(rstudioapi::getSourceEditorContext()$path)) # sets working directory to current folder

# Importing the dataset

cardio\_data <- read.csv("data\_cardiovascular\_risk.csv", header=TRUE, stringsAsFactors=TRUE)

cardio <- cardio\_data

# Checking the structure and summary of the dataset

str(cardio)

summary(cardio)

# Checking the dimension of the dataset

dim(cardio)

# Getting the names of the columns in the dataset

# Renaming columns

names(cardio)[names(cardio) == "sex"] <- "gender"

names(cardio)[names(cardio) == "is\_smoking"] <- "smoker"

names(cardio)[names(cardio) == "TenYearCHD"] <- "outcome"

names(cardio)

names(cardio)

# Removing the id variable

cardio <- cardio %>% select(-"id")

# List types for each attribute & check the structure again

sapply(cardio, class)

str(cardio)

# Get the number of missing values & Plot a chart for every attribute

sum(is.na(cardio))

options(repr.plot.width = 10, repr.plot.height = 4)

plot\_missing(cardio)

# Replace missing values with median value

cardio$glucose = replace\_na(cardio$glucose, median(cardio$glucose, na.rm = TRUE))

cardio$BMI = replace\_na(cardio$BMI, median(cardio$BMI, na.rm = TRUE))

# Drop the other missing value because it is less than 5% of the data and plot a second chart for every attribute

sum(is.na(cardio))

cardio <- na.omit(cardio)

options(repr.plot.width=10, repr.plot.height=4)

plot\_missing(cardio)

# Checking the current data dimension after droping missing values

# Values have reduced to 3202

dim(cardio)

# EXPLORATORY DATA ANALYSIS (DATA VISUALIZATION)

# Changing the numeric variables with new labels for better visualization

cardio$outcome <- factor(cardio$outcome, labels = c("Yes", "No"))

cardio$gender <- factor(cardio$gender, labels = c("Female", "Male"))

cardio$education <- factor(cardio$education, labels = c("Junior School", "High School", "College", "University"))

cardio$smoker <- factor(cardio$smoker, labels = c("Smoker", "Non-Smoker"))

cardio$BPMeds <- factor(cardio$BPMeds, labels = c("Medication", "No-Medication"))

cardio$prevalentStroke <- factor(cardio$prevalentStroke, labels = c("Yes", "No"))

cardio$prevalentHyp <- factor(cardio$prevalentHyp, labels = c("Yes", "No"))

cardio$diabetes <- factor(cardio$diabetes, labels = c("Yes", "No"))

cardio %>% glimpse()

# Histograms for attributes

cardio\_data %>%

ggplot(mapping = aes(age,color = "sex"))+

geom\_histogram(col = I("#8B2252"), fill = '#FFF0F5')+

facet\_wrap(~ sex) +

ggtitle("Age Distribution")

cardio\_data %>%

ggplot(mapping = aes(BMI,color = "BMI"))+

geom\_histogram(col = I("#8B2252"), fill = '#FFF0F5')+

facet\_wrap(~ sex) +

ggtitle("BMI Distribution")

cardio\_data %>%

ggplot(mapping = aes(totChol,color = "totC"))+

geom\_histogram(col = I("#8B2252"), fill = '#FFF0F5')+

facet\_wrap(~ sex) +

ggtitle("Total Cholesterol")

cardio\_data %>%

ggplot(mapping = aes(heartRate,))+

geom\_histogram(col = I("#8B2252"), fill = '#FFF0F5')+

facet\_wrap(~ sex) +

ggtitle("Heart Rate")

cardio\_data %>%

ggplot(mapping = aes(glucose,))+

geom\_histogram(col = I("#8B2252"), fill = "#FFF0F5")+

facet\_wrap(~ sex) +

ggtitle("Glucose Level")

# Clustered Bar charts for attributes

# Smoker and prevalence of stroke distribution

smoker = ggplot(cardio, aes(smoker , fill = gender)) +

geom\_bar(stat = "count", position = "dodge") +

scale\_fill\_manual(values=c("#CD1076","#79CDCD")) +

labs(title = "Smoker Distribution by Gender", x = "") +

theme\_bw(base\_size = 8) + theme(legend.position="bottom")

stroke = ggplot(cardio, aes(prevalentStroke , fill = gender)) +

geom\_bar(stat = "count", position = "dodge") +

scale\_fill\_manual(values=c("#CD1076","#79CDCD")) +

labs(title = "Prevalence of Stroke by Gender", x = "") +

theme\_bw(base\_size = 8) + theme(legend.position="bottom")

# Plotting Chart

plot\_grid(smoker,stroke, ncol = 2, nrow = 1)

# Chart for BPMeds and Prevalence of Hypertension

BPMeds = ggplot(cardio, aes(BPMeds , fill = gender)) +

geom\_bar(stat = "count", position = "dodge") +

scale\_fill\_manual(values=c("#CD1076","#79CDCD")) +

labs(title = "Blood Pressure Medication by Gender", x = "") +

theme\_bw(base\_size = 8) + theme(legend.position="bottom")

Hyper = ggplot(cardio, aes(prevalentHyp , fill = gender)) +

geom\_bar(stat = "count", position = "dodge") +

scale\_fill\_manual(values=c("#CD1076","#79CDCD")) +

labs(title = "Prevalence of Hypertension by Gender", x = "") +

theme\_bw(base\_size = 8) + theme(legend.position="bottom")

# Plotting Chart

plot\_grid(BPMeds,Hyper, ncol = 2, nrow = 1)

# Compariaring some of the variables with the Target Variable

# \*\*\* Count of Cardiovascular Disease \*\*\*

a = ggplot(cardio, aes(outcome, fill = outcome)) +

geom\_bar(stat = "count") + scale\_fill\_manual(values=c("#CD1076","#79CDCD")) +

labs(title = "Risk of Cardiovascular Disease") + theme\_bw(base\_size = 8) +

theme(legend.position="bottom")

# \*\*\* Gender \*\*\*

b = ggplot(cardio, aes(gender, fill = outcome)) +

geom\_bar(stat = "count", position = "dodge") +

scale\_fill\_manual(values=c("#CD1076","#79CDCD")) +

labs(title = "Gender Distribution", x = "") + theme\_bw(base\_size = 8) +

theme(legend.position="bottom")

# Plotting chart

options(repr.plot.width=12, repr.plot.height=6)

plot\_grid(a,b, ncol = 2, nrow = 1)

# \*\*\* Smoker \*\*\*

c = ggplot(cardio, aes(smoker , fill = outcome)) +

geom\_bar(stat = "count", position = "dodge") +

scale\_fill\_manual(values=c("#CD1076","#79CDCD")) +

labs(title = "Smoker Distribution", x = "") +

theme\_bw(base\_size = 8) + theme(legend.position="bottom")

# \*\*\* Blood Pressure Medication \*\*\*

d = ggplot(cardio, aes(BPMeds, fill = outcome)) +

geom\_bar(stat = "count", position = "dodge") +

scale\_fill\_manual(values=c("#CD1076","#79CDCD")) +

labs(title = "Blood Pressure Medication", x = "") +

theme\_bw(base\_size = 8) + theme(legend.position="bottom")

# Plotting Chart

plot\_grid(c,d, ncol = 2, nrow = 1)

# \*\*\* Education \*\*\*

e = ggplot(cardio, aes(education, fill = outcome)) +

geom\_bar(stat = "count", position = "dodge") +

scale\_fill\_manual(values=c('#CD1076', '#79CDCD')) +

labs(title = "Education Distribution", x = "") +

theme\_bw(base\_size = 8) + theme(legend.position="bottom")

# \*\*\* Prevalence of Stroke \*\*\*

f = ggplot(cardio, aes(prevalentStroke, fill = outcome)) +

geom\_bar(stat = "count", position = "dodge") +

scale\_fill\_manual(values=c('#CD1076', '#79CDCD')) +

labs(title = "Prevalence of Stroke", x = "") +

theme\_bw(base\_size = 8) + theme(legend.position="bottom")

plot\_grid(e,f, ncol = 2, nrow = 1)

# Changing the dataset to its original form

cardio <- cardio %>%

mutate(

outcome = as.numeric(outcome),

education = as.numeric(education),

gender = as.numeric(gender),

smoker = as.numeric(smoker),

BPMeds = as.numeric(BPMeds),

prevalentHyp = as.numeric(prevalentHyp),

prevalentStroke = as.numeric(prevalentStroke),

diabetes = as.numeric(diabetes)

)

# Histogram for all the attributes - Do i still need these?

par(mfrow=c(2,4))

for(i in 1:16) {

hist(cardio[,i], main = names(cardio)[i])

}

# Density plots for every attribute

par(mfrow=c(2,4))

for(i in 1:16) {

plot(density(cardio[,i]), main = names(cardio)[i])

}

# list the levels for the Target class

levels(cardio$outcome)

# The total counts of Negative and Positive

table(cardio$outcome)

# Data normalization scaling between 0 and 1

cardio\_clean = subset(cardio, select = -c(education) )

cardio\_clean

normalize <- function(x) {return ((x - min(x)) / (max(x) - min(x))) }

cardio\_normalized <- as.data.frame(lapply((cardio\_clean[,1:ncol(cardio\_clean)]), normalize))

cardio\_normalized

# Correlation Matrix Visualization

corrplot(cor(cardio\_normalized [,-1]))

#Define highly correlated column from the dataset

corr\_Data <- cor(cardio\_normalized)

highCorr <- findCorrelation(corr\_Data, cutoff = .60)

length(highCorr)

highCorrelated <- data.frame(highCorr)

highCorrelated

#DATA PRE-PROCESSING

#SMOTE BALANCING of Dataset

cardio\_normalized$outcome <- as.factor(cardio\_normalized$outcome)

#BALANCING THE TARGET VARIABLE USING SMOTE

cardio\_up <- SMOTE(outcome ~ ., data = cardio\_normalized, perc.over = 100, perc.under = 300, k = 5)

# Check class distribution after SMOTE

table(cardio\_up$outcome)

# Original Data

p1 <- ggplot(cardio, mapping = aes(x = outcome)) +

geom\_bar(colour = "red", fill = "darkgrey") +

labs(title = "Original data") +

theme\_bw()

# Upsampled Data

p2 <- ggplot(cardio\_up, mapping = aes(x = outcome)) +

geom\_bar(colour = "red", fill = "darkblue") +

labs(title = "Upsampled data") +

theme\_bw()

#Subplot

grid.arrange(p1, p2, ncol = 2)

# Check the count of occurrences, proportions and percentage of each variable

count <- table(cardio\_up$outcome)

proportions <- prop.table(count)

percentages <- prop.table(count) \* 100

# Print the results

count

proportions

percentages

# MACHINE LEARNING

#Shuffle the rows of the data frame

set.seed(77)

cardio\_shuffled <- cardio\_up %>%

sample\_n(size = nrow(cardio\_up),

replace = FALSE)

cardio\_shuffled$outcome <- as.factor(cardio\_shuffled$outcome)

# PARTITIONING THE DATA

# Partition the Upsampled dataset in two part one is Training data (as 70%) and another is Test data (as 30%)

train\_index <- createDataPartition(y = cardio\_shuffled$outcome, times = 1, p = 0.7, list= FALSE)

# Split the data

train\_set <- cardio\_shuffled[train\_index, ]

test\_set <- cardio\_shuffled[-train\_index, ]

# Converting the dependent variables to factors

train\_set$outcome <- as.factor(train\_set$outcome)

test\_set$outcome <- as.factor(test\_set$outcome)

dim(train\_set)

dim(test\_set)

# set up the train control method tp compute all the computational overhead prior to training

trctrl <- trainControl(method= "repeatedcv",

number = 10,

repeats = 5)

######

# SVM

######

# train the model with the train() method

set.seed(7)

svm\_Linear <- train(outcome~.,data = train\_set,

method ="svmLinear",trControl = trctrl,

preProcess = c("center", "scale"),tuneLength =10)

svm\_Linear

# Check results for the train method

test\_pred <-predict(svm\_Linear, newdata = test\_set)

test\_pred

# Confusion matrix stats

confusionMatrix(table(test\_pred,

test\_set$outcome))

#create a grid for SVM

grid <-expand.grid(C=c(0, 0.01, 0.05, 0.1, 0.25, 0.5,0.75, 1, 1.25, 1.5, 1.75, 2.5))

# Find the best value for C

svm\_Linear\_Grid <- train(outcome~.,data = train\_set,

method = "svmLinear", trControl = trctrl,

preProcess =c("center", "scale"), tuneGrid = grid,

tuneLength = 10)

svm\_Linear\_Grid

# Plot the SVM

plot(svm\_Linear\_Grid, main = "SVM Decision Boundary Plot")

SVM\_test\_pred\_Grid <- predict(svm\_Linear\_Grid, newdata = test\_set)

SVM\_test\_Results <- confusionMatrix(table(SVM\_test\_pred\_Grid,test\_set$outcome))

# Confusion matrix Plot for SVM

Reference <- factor(c(0, 0, 1, 1))

Prediction <- factor(c(0, 1, 0, 1))

Y <- c(350 , 83, 152, 137)

svm <- data.frame(Reference, Prediction, Y)

ggplot(svm, mapping = aes(x = Reference, y = Prediction)) +

geom\_tile(aes(fill = Y), colour = "#8B0A50") +

geom\_text(aes(label = sprintf("%1.0f", Y)), vjust = 1) +

labs(title = "Support Vector Machine",) +

scale\_fill\_gradient(low = "#A2CD5A", high = "#B23AEE") +

theme\_bw() + theme(legend.position = "none")

######

# KNN

######

ctrl <- trainControl(method = "cv", verboseIter = FALSE, number = 5)

set.seed(7)

knn <- train(outcome~.,data = train\_set,

method ="knn",trControl = ctrl,

preProcess = c("center", "scale"),tuneLength =10)

knn

plot(knn, main = "KNN Decision Boundary Plot")

toc()

knn\_predict <- predict(knn, newdata = test\_set)

knn\_results <- confusionMatrix(knn\_predict, test\_set$outcome)

knn\_results

# Confusion matrix Plot for KNN

Reference <- factor(c(0, 0, 1, 1))

Prediction <- factor(c(0, 1, 0, 1))

Y <- c(322 , 111, 199, 90)

knn <- data.frame(Reference, Prediction, Y)

ggplot(knn, mapping = aes(x = Reference, y = Prediction)) +

geom\_tile(aes(fill = Y), colour = "#8B0A50") +

geom\_text(aes(label = sprintf("%1.0f", Y)), vjust = 1) +

labs(title = "K-Neares Neighbours",) +

scale\_fill\_gradient(low = "#A2CD5A", high = "#B23AEE") +

theme\_bw() + theme(legend.position = "none")

# Compare SVM result to KNN

#train the KNN model

set.seed(7)

fit.knn <- train(outcome~., data = train\_set,

method="knn", preProcess = c("center", "scale"),

trControl=trainControl())

#train the SVM model

set.seed(7)

fit.svm <-train(outcome~., data=train\_set,

method="svmLinear",preProcess =c("center", "scale"),

trControl=trainControl())

# Compare algorithms

comp <- resamples(list(SVM = fit.svm, KNN = fit.knn))

summary\_table <- summary(comp)

# Dotplot with customized colors for points and lines

bwplot(comp, fill = c("#EE6A50", "#A2CD5A"))

######################

# RF (Random Forest)

#####################

control<- trainControl(method = "cv", number = 5, verboseIter = FALSE)

grid <-data.frame(mtry = seq(1, 10, 2))

tic(msg= " Total time for rf :: ")

rf\_fit <- train(outcome ~ ., method = "rf", data = train\_set, ntree = 20, trControl = control,

tuneGrid = grid)

plot(rf\_fit)

toc()

rf\_predict <- predict(rf\_fit, newdata = test\_set)

rf\_results <- confusionMatrix(rf\_predict, test\_set$outcome)

rf\_results

# Confusion matrix Plot of RF

Reference <- factor(c(0, 0, 1, 1))

Prediction <- factor(c(0, 1, 0, 1))

Y <- c(374 , 59, 192, 97)

df <- data.frame(Reference, Prediction, Y)

ggplot(rf\_results = df, mapping = aes(x = Reference, y = Prediction)) +

geom\_tile(aes(fill = Y), colour = "black") +

geom\_text(aes(label = sprintf("%1.0f", Y)), vjust = 1) +

labs(title = "Random Forest",) +

scale\_fill\_gradient(low = "#97FFFF", high = "#97FFFF") +

theme\_bw() + theme(legend.position = "none")

# Comparison graph of RF and KNN

data <- read.table(text="

Measures,Machine\_Learning,Percent

Accuracy,K-Nearest Neighbours,72.16

Accuracy,Random Forest,78.39

Sensitivity,K-Nearest Neighbours,74.36

Sensitivity,Random Forest,86.37", header=TRUE, sep=",")

measure\_colors <- c("Accuracy" = "#008B8B", "Sensitivity" = "#FF7256")

ggplot(data, aes(x=Machine\_Learning, y=Percent, fill=Measures)) +

geom\_bar(stat="identity", position="dodge") +

geom\_text(aes(label=Percent), vjust=1.6, color="black",

position = position\_dodge(0.9), size=5) +

scale\_fill\_manual(values = measure\_colors) +

theme\_minimal()