```r
########### Import library necessary ##############################################################################

library(ggplot2)
library(tidyverse)
library(GGally)
library(corrplot)
library(nortest)
library(pastecs)
library(data.table)
library(caret)
library(dataPreparation)
library(caTools)
library(microbenchmark)
library(ggpubr)

########## Import dataset ###########################################################################################
raw <- read.csv('cardio_train.csv',sep = ";")

########## Handle Missing Value and Data Formating ##########################################################################

# Check for any missing values
colSums(is.na(raw))

# Define categorical list
cat.list <- c("id","gender","cholesterol","gluc","smoke","alco","active","cardio")

# Convert numerical columns to categorical data
raw[cat.list ] <- lapply(raw[cat.list ], factor)

# Define numerical list
num.list <- c("age","height","weight","ap_hi","ap_lo")

# Reduce "age" measurement unit from day to year
raw <- mutate(raw, age = age / 365.25)

########## Exploratory Data Analysis #############################################################################

# Check structure of dataframe
str(raw)

# Check descriptive statistic of data
summary(raw)

# Univariate Plots - Plot and export boxplot as jpeg
plot.box <- function(x,y){
  plot <- boxplot(x, xlab=y) + theme_minimal()
  return(plot)
}

for (i in 1:length(num.list)){
  jpeg(file=sprintf("plot_box_%s.jpeg",num.list[i]))
  print(plot.box(raw[num.list][,i],num.list[i]))
  dev.off()
  print(i)}

# Remove outliers of feature “ap_hi” and “ap_lo” and redraw the box plot
outliers4 <- boxplot(raw$ap_hi, plot=FALSE)$out
raw <- raw[-which(raw$ap_hi %in% outliers4),]
boxplot(raw$ap_hi,xlab="ap_hi")
outliers5 <- boxplot(raw$ap_lo, plot=FALSE)$out
raw <- raw[-which(raw$ap_lo %in% outliers5),]
boxplot(raw$ap_lo,xlab="ap_lo")

# Check descriptive statistic of data again after removed outlier in ap_lo and ap_hi
summary(raw[,2:13])

```

```r
# Check Distribution - Plot and export histogram of numerical variables as jpeg
plot.hist <- function(x,y){
  plot <- ggplot(raw,aes(x)) + geom_histogram(fill='blue',bins = 20,alpha=0.5)+ labs(x = y) + theme_minimal()
  return(plot)
}

for (i in 1:length(num.list)){
  jpeg(file=sprintf("plot_hist_%s.jpeg",num.list[i]))
  print(plot.hist(raw[num.list][,i],num.list[i]))
  dev.off()
  print(i)}

# Univariate Plots - Plot and export bar chart of categorical variables as jpeg
plot.bar <- function(x,y){
  plot <- ggplot(raw,aes(x)) + geom_bar(aes(fill=factor(x)),alpha=0.5)+ labs(x = y) + theme_minimal()+labs(fill = y)
  return(plot)
}

for (i in 2:length(cat.list)){
  jpeg(file=sprintf("plot_bar_%s.jpeg",cat.list[i]))
  print(plot.bar(raw[cat.list][,i],cat.list[i]))
  dev.off()
  print(i)}

# Multivariate Plots - Plot and export scatter plot as jpeg
plot.sca <- function(x,y,z,m,n,p){
  plot <- ggplot(raw,aes(x=x,y=y)) + geom_point(aes(color=factor(z)),alpha=0.5)+ theme_minimal() + labs(x=m, y=n,col=p)
  return(plot)
}

for (i in 1:length(num.list)){
  for (l in 1:length(num.list)){
    if (num.list[i] != num.list[l]){
      jpeg(file=sprintf("plot_scatter_%s_%s.jpeg",num.list[i],num.list[l]))

      print(plot.sca(raw[num.list][,i],raw[num.list][,l],raw$cardio,num.list[i],num.list[l],"cardio"))
      dev.off()
      print(i)
    }
  }
}

# Plot and export correlation matrix as jpeg
num.cols <- sapply(raw, is.numeric)
cor.data <- cor(raw[,num.cols])
jpeg(file="corr.jpeg")
corrplot(cor.data,method = "number")
dev.off()

############# Train and Test Set ###############################################################################################

set.seed(123)
raw_train <- raw[,2:ncol(raw)]
split = sample.split(raw_train $cardio, SplitRatio = 0.75)
training_set = subset(raw_train, split == TRUE)
test_set = subset(raw_train, split == FALSE)

summary(training_set)

X_train = training_set[,1:ncol(raw_train)-1]
y_train = training_set[,ncol(raw_train)]
X_test = test_set[,1:ncol(raw_train)-1]
y_test = test_set[,ncol(raw_train)]

# Scaling
scales <- build_scales(dataSet = training_set, cols = num.list, verbose = TRUE)
training_set_s <- fastScale(dataSet = training_set, scales = scales, verbose = TRUE)
```

```
test_set_s <- fastScale(dataSet = test_set, scales = scales, verbose = TRUE)
X_train_s <- fastScale(dataSet = X_train, scales = scales, verbose = TRUE)
X_test_s <- fastScale(dataSet = X_test, scales = scales, verbose = TRUE)

summary(training_set_s)

############# Test Harness ##############################################################################################

# Run algorithms using 10-fold cross validation
control <- trainControl(method="cv", number=10)
metric <- "Accuracy"

############# Build Models ##############################################################################################

# LDA
set.seed(7)
fit.lda <- train(cardio~., data=training_set, method="lda", metric=metric, trControl=control)

# CART
set.seed(7)
fit.cart <- train(cardio~., data=training_set, method="rpart", metric=metric, trControl=control)

# naive bayes
set.seed(7)
fit.nb <- train(cardio~., data=training_set, method="nb", metric=metric, trControl=control)

# kNN
set.seed(7)
knn.grid <- expand.grid(k=c(203,253)) # design the parameter tuning grid
fit.knn <- train(cardio~., data=training_set_s, method="knn", metric=metric, trControl=control, tuneGrid=knn.grid)

# Random Forest
set.seed(7)
rf.grid <- expand.grid(mtry=c(2,7,12)) # design the parameter tuning grid
fit.rf <- train(cardio~., data=training_set, method="rf", metric=metric, trControl=control, tuneGrid=rf.grid)


############# Measure Training and Testing Time ##################################################################################

# Measure training time
knn.grid.final <- expand.grid(k=c(fit.knn$bestTune[[1]]))
rf.grid.final <- expand.grid(mtry=c(fit.rf$bestTune[[1]]))
mbm_train <- microbenchmark("LDA" = { train1 <- train(cardio~., data=training_set, method="lda", metric=metric, trControl=control)},
                      "CART" = {train2 <- train(cardio~., data=training_set, method="rpart", metric=metric, trControl=control)},
                      "NB" = {train3 <- train(cardio~., data=training_set, method="nb", metric=metric, trControl=control)},
                      "KNN" = {train4 <- train(cardio~., data=training_set_s, method="knn", metric=metric, trControl=control, tuneGrid=knn.grid.final)},
                      "RF" = {train5 <- train(cardio~., data=training_set, method="rf", metric=metric, trControl=control, tuneGrid=rf.grid.final)},
                      times = 3,unit = "s")

# Measure testing time
mbm_test <- microbenchmark("LDA" = { test1 <- predict(train1, test_set)},
                           "CART" = {test2 <- predict(train2, test_set)},
                           "NB" = {test3 <- predict(train3, test_set)},
                           "KNN" = {test4 <- predict(train4, test_set_s)},
                           "RF" = {test5 <- predict(train5, test_set)},
                           times = 3,unit = "s")
```

```
############# Select Best Model ##########################################################################################

# compare accuracy of models
results <- resamples(list(lda=train1, cart=train2, nb=train3, kn=train4,rf=train5))
summary(results)
jpeg(file="model_performance.jpeg")
dotplot(results)
dev.off()

# compare training time of models
mbm_train
jpeg(file="training_time.jpeg")
autoplot(mbm_train)
dev.off()

# compare testing time of models
mbm_test
jpeg(file="testing_time.jpeg")
autoplot(mbm_test)
dev.off()

# summarize best model (accuracy)
print(train5)

############# Make Predictions ############################################################################################

predictions <- predict(train5, test_set)
confusionMatrix(predictions, test_set$cardio)

############# Feature Importance ############################################################################
feature.imp <- varImp(train5)
jpeg(file="feature_importance.jpeg")
ggplot(data=feature.imp)
dev.off()
```