```r
# Code converted to pdf using https://tarikjaber.github.io/Code-to-PDF/
# Word converted all comments into headings :(

# B01003963, B00778496, B01006361, B01009661

# IMPORTING AND LOADING LIBRARIES
packages ← c("tidyverse", "ggplot2", "dplyr", "DataExplorer", "caret", "corrplot","pROC", "randomForest")
install_if_missing ← function(p) {
  if (!require(p, character.only = TRUE)) {
    install.packages(p, dependencies = TRUE)
    library(p, character.only = TRUE)
  }
}
invisible(sapply(packages, install_if_missing))



# SECTION ONE: Loading the dataset

# Load the dataset  (pleasenmake sure cardio_train.csv is in your working directory)
data ← read.csv("data/cardio_train.csv", sep = ";")

head(data)
str(data)
summary(data)
print("Data Loading Completed.")



# SECTION TWO: Data Cleansing

# Load the dataset (make sure cardio_train.csv is in your working directory)
df ← read.csv("data/cardio_train.csv", sep = ";")


# View basic information about the dataset
print("Structure of dataset:")
str(df)

print("Summary statistics:")
summary(df)

print("Missing values per column:")
print(colSums(is.na(df)))

print("Column names:")
print(names(df))
```

```r

print("First few rows of dataset:")
head(df)

# Basic Exploratory Data Analysis, more in eda.R
# Distribution of target variable (cardio)
ggplot(df, aes(x = factor(cardio))) +
  geom_bar(fill = "steelblue") +
  labs(title = "Distribution of Cardiovascular Disease", x = "Cardio (0=No, 1=Yes)", y = "Count")

# Correlation matrix plot
# Notable correlations:
#   gender&height, height&weight, gender&smoke, chol&gluc, age&cardio, weight&cardio, chol&cardio
numeric_df <- df[sapply(df, is.numeric)]
cor_matrix <- cor(numeric_df)
corrplot(cor_matrix, method = "color", type = "upper", tl.cex = 0.8)

# Data Cleaning
# Removing outliers for height, weight, and blood pressure
df <- df %>% filter(height > 100 & height < 250)
df <- df %>% filter(weight > 30 & weight < 200)
df <- df %>% filter(ap_hi > 80 & ap_hi < 250)
df <- df %>% filter(ap_lo > 40 & ap_lo < 200)

# Remove duplicate rows
df <- df[!duplicated(df), ]

# Add new column for age in years (Dataset uses days) and round down to the nearest whole number
df <- df%>%
  mutate(age_years = floor(age / 365))

# Feature Engineering
# Create BMI variable
df$bmi <- df$weight / ((df$height / 100)^2)

df <- df %>%
  mutate(bmi = weight / ((height / 100) ^ 2))  # height is in cm

# Convert categorical variables to factors
df$gender <- as.factor(df$gender)
df$cholesterol <- as.factor(df$cholesterol)
df$gluc <- as.factor(df$gluc)
df$smoke <- as.factor(df$smoke)
df$alco <- as.factor(df$alco)
df$active <- as.factor(df$active)
df$cardio <- as.factor(df$cardio)

# Save the cleaned dataset
```

```r
write.csv(df, "data/cardio_cleaned.csv", row.names = FALSE)

print("Data exploration, cleaning, and preprocessing completed. Cleaned dataset saved as cardio_cleaned.csv")




# SECTION THREE: Exploratory Data Analysis - THIS SHOULD MAYBE BE THREE
# I need to sort this out a little bit, not sure if should use data or df here
# Maybe should rename variables to reflect status, like initial, cleaned, model_ready
# Should be saving plots, not just displaying them
#library(scales)

# Add new column for age in years (Dataset uses days)
data <- data%>%
  mutate(age_years = age / 365)

# Age Distribution
gg_age <- ggplot(df, aes(x = age_years)) +
  geom_histogram(bins = 30, fill = "steelblue") +
  xlab("Age (Years)") +
  ggtitle("Age Distribution of Patients")
gg_age

# Cholesterol Levels (categorical)
gg_chol <- ggplot(df, aes(x = cholesterol, fill = cholesterol)) +
  geom_bar() +
  xlab("Cholesterol Level") +
  ggtitle("Cholesterol Levels in Dataset") +
  theme(legend.position = "none")
gg_chol


# Comparing Gender Distribution
# Important for ethical considerations, need to consider how this data effects the results and if it's a fair distribution
# Since there's a far greater number (roughly 65%)of female patients we need to consider why in the report
ggplot(data, aes(x=factor(gender))) +
  geom_bar(fill= "steelblue") +
  xlab("Gender (1 = Women, 2 = Men)") +
  ggtitle("Gender Distribution of Patients")

# Comparing gender distribution against CVD
# Very slightly more men have CVD proportionally than women but statistically insignificant
ggplot(data, aes(x = factor(gender), fill = factor(cardio))) +
  geom_bar(position = "fill") +
```

```
  scale_y_continuous(labels = scales::percent) +
  xlab("Gender (1 = Women, 2 = Men)") +
  ylab("Proportion of Patients") +
  ggtitle("Gender by Cardiovascular Disease Status") +
  labs(fill = "CVD (0 = No, 1 = Yes)")

# Comparing gender distribution against Age
# Accounts for difference in number of men and women
ggplot(data, aes(x = age_years, fill = factor(gender))) +
  geom_density(alpha = 0.4, adjust = 1.5) +  # Adjusting for a smoother curve
  xlab("Age (Years)") +
  ylab("Density") +
  ggtitle("Normalised Age Distribution against Gender") +
  labs(fill = "Gender (1 = Women, 2 = Men)")

# Comparing gender distribution against height
ggplot(data, aes(x = height, fill = factor(gender))) +
  geom_density(alpha = 0.4, adjust = 1.5) +  # Adjusting for a smoother curve
  xlab("Height (cm)") +
  ylab("Density") +
  ggtitle("Normalised Height Distribution against Gender") +
  labs(fill = "Gender (1 = Women, 2 = Men)")

# Comparing gender distribution against weight
ggplot(data, aes(x = weight, fill = factor(gender))) +
  geom_density(alpha = 0.4, adjust = 1.5) +  # Adjusting for a smoother curve
  xlab("Weight (Kg)") +
  ylab("Density") +
  ggtitle("Normalised Weight Distribution against Gender") +
  labs(fill = "Gender (1 = Women, 2 = Men)")

# Comparing smoking by gender
# Smoking is far more prevalent in men
ggplot(data, aes(x = factor(gender), fill = factor(smoke))) +
  geom_bar(position = "fill") +
  scale_y_continuous(labels = scales::percent) +
  xlab("Gender (1 = Women, 2 = Men)") +
  ylab("Proportion of Patients") +
  ggtitle("Smoking against Gender") +
  labs(fill = "Smoking (0 = No, 1 = Yes)")

# Comparing Alcohol consumption by gender
# Alcohol consumption is also more prevalent in men, but less so than smoking
ggplot(data, aes(x = factor(gender), fill = factor(alco))) +
  geom_bar(position = "fill") +
  scale_y_continuous(labels = scales::percent) +
  xlab("Gender (1 = Women, 2 = Men)") +
  ylab("Proportion of Patients") +
  ggtitle("Alcohol Consumption against Gender") +
  labs(fill = "Alcohol (0 = No, 1 = Yes)")

```

```r
# Comparing cholesterol levels against gender
# Women in the population have a generally higher cholesterol level than men
ggplot(data, aes(x = factor(gender), fill = factor(cholesterol))) +
  geom_bar(position = "fill") +
  scale_y_continuous(labels = scales::percent) +
  xlab("Gender (1 = Women, 2 = Men)") +
  ylab("Proportion of Patients") +
  ggtitle("Cholesterol Levels against Gender") +
  labs(fill = "Cholesterol Level (1 = Normal, 2 = Above Normal, 3 = Well Above Normal)")

# Comparing Systolic and Diastolic blood pressure against gender
# Not entirely happy about how these look, might be better way to represent
# However women have a greater max blood pressure than men
ggplot(data, aes(x = factor(gender), y = ap_hi, fill = factor(gender))) +
  geom_boxplot() +
  xlab("Gender (1 = Women, 2 = Men)") +
  ylab("Systolic Blood Pressure") +
  ggtitle("Systolic Blood Pressure against Gender")

ggplot(data, aes(x = factor(gender), y = ap_lo, fill = factor(gender))) +
  geom_boxplot() +
  xlab("Gender (1 = Women, 2 = Men)") +
  ylab("Diastolic Blood Pressure") +
  ggtitle("Diastolic Blood Pressure against Gender")

# Comparing Glucose levels between men and women
# Women have slightly higher glucose levels
ggplot(data, aes(x = factor(gender), fill = factor(gluc))) +
  geom_bar(position = "fill") +
  scale_y_continuous(labels = scales::percent) +
  xlab("Gender (1 = Women, 2 = Men)") +
  ylab("Proportion of Patients") +
  ggtitle("Glucose Levels against Gender") +
  labs(fill = "Glucose Level (1 = Normal, 2 = Above Normal, 3 = Well Above Normal)")

# Comparing activity by gender
# Activity levels are very similar
ggplot(data, aes(x = factor(gender), fill = factor(active))) +
  geom_bar(position = "fill") +
  scale_y_continuous(labels = scales::percent) +
  xlab("Gender (1 = Women, 2 = Men)") +
  ylab("Proportion of Patients") +
  ggtitle("Activity against Gender") +
  labs(fill = "Activity (0 = Not Active, 1 = Active)")

# Visualize age distribution
ggplot(data, aes(x = age_years)) +
  geom_histogram(bins = 30) +
  xlab("Age (Years)") +
```

```
  ggtitle("Age Distribution Of Patients")

# Visualize distribution of cholesterol levels
ggplot(data, aes(x = factor(cholesterol))) +
  geom_bar() +
  xlab("Cholesterol Level (1: Normal, 2: Above Normal, 3: Well Above Normal)") +
  ggtitle("Cholesterol Level Distribution")

# Cardiovascular disease class balance
gg_cardio <- ggplot(df, aes(x = factor(cardio))) +
  geom_bar(fill = "darkred") +
  xlab("Cardiovascular Disease (0 = No, 1 = Yes)") +
  ggtitle("Distribution of Cardiovascular Disease Cases")
gg_cardio

# Age vs Disease status (boxplot)
gg_age_cardio <- ggplot(df, aes(x = factor(cardio), y = age_years, fill = factor(cardio))) +
  geom_boxplot() +
  xlab("Cardiovascular Disease (0 = No, 1 = Yes)") +
  ylab("Age (Years)") +
  ggtitle("Age Distribution by Cardiovascular Disease") +
  theme(legend.position = "none")
gg_age_cardio

# Smoking vs. Disease (stacked proportion)

gg_smoke <- ggplot(df, aes(x = factor(smoke), fill = factor(cardio))) +
  geom_bar(position = "fill") +
  scale_y_continuous(labels = scales::percent) +
  xlab("Smoking (0 = No, 1 = Yes)") +
  ylab("Proportion of Patients") +
  ggtitle("Smoking Habits of Cardiovascular Disease Status") +
  labs(fill = "CVD (0 = No, 1 = Yes)")
gg_smoke

# Correlation matrix for numeric variables
numeric_vars <- df%>%select_if(is.numeric)
cor_matrix <- cor(numeric_vars)
corrplot(cor_matrix, method = "color", type = "upper", tl.cex = 0.7)

# BMI distribution by disease
gg_bmi <- ggplot(df, aes(x = bmi, fill = factor(cardio))) +
  geom_histogram(position = "identity", alpha = 0.5, bins = 30) +
  xlab("BMI") +
  ggtitle("BMI Distribution by Cardiovascular Disease") +
  labs(fill = "CVD (0 = No, 1 = Yes)")
gg_bmi

# Relationship between alcohol status intake and cardiovascular disease
```

```r
gg_alcohol <- ggplot(data, aes(x = factor(alco), fill = factor(cardio))) +
  geom_bar(position = "fill") +
  scale_y_continuous(labels = scales::percent) +
  xlab("Alcohol Consumption (0 = No, 1 = Yes)") +
  ylab("Proportion of Patients") +
  ggtitle("Alcohol Consumption by Cardiovascular Disease Status") +
  labs(fill = "CVD (0 = No, 1 = Yes)")

# Display gg_alcohol
gg_alcohol

# Required variables are treated correctly
df$gluc <- factor(df$gluc, levels = c(1, 2, 3),
                  labels = c("Normal", "Above Normal", "Well Above Normal"))
df$active <- factor(df$active, levels = c(0, 1), labels = c("Inactive",
"Active"))
df$cardio <- factor(df$cardio, levels = c(0, 1), labels = c("No", "Yes"))


# Height Distribution
gg_height <- ggplot(df, aes(x = height)) +
  geom_histogram(bins = 30, fill = "seagreen") +
  xlab("Height (cm)") +
  ggtitle("Height Distribution of Patients")
gg_height

# Weight Distribution
gg_weight <- ggplot(df, aes(x = weight)) +
  geom_histogram(bins = 30, fill = "purple") +
  xlab("Weight (kg)") +
  ggtitle("Weight Distribution of Patients")
gg_weight

# Glucose Levels by Disease Status
gg_gluc <- ggplot(df, aes(x = gluc, fill = cardio)) +
  geom_bar(position = "fill") +
  scale_y_continuous(labels = scales::percent) +
  xlab("Glucose Level") +
  ylab("Proportion of Patients") +
  ggtitle("Glucose Levels by Cardiovascular Disease Status") +
  labs(fill = "CVD")
gg_gluc

# Physical Activity by Disease Status
gg_active <- ggplot(df, aes(x = active, fill = cardio)) +
  geom_bar(position = "fill") +
  scale_y_continuous(labels = scales::percent) +
  xlab("Physical Activity") +
  ylab("Proportion of Patients") +
  ggtitle("Physical Activity by Cardiovascular Disease Status") +
  labs(fill = "CVD")
```

```
gg_active


# SECTION FOUR: Feature Engineering

# Load cleaned data
df ← read.csv("data/cardio_cleaned.csv")

# Drop unnecessary columns
# ID as not relevant to our research
# COMMENTING THIS OUT TEMPORARILY, NEED TO CHECK SOMETHING, GETTING ERROR, NOT SHOWING UP IN COLUMN NAMES
df ← df %>%
  select(-id)

# Scale numeric features
# So easier to compare
numeric_cols ← c("age_years", "height", "weight", "ap_hi", "ap_lo", "bmi")
df[numeric_cols] ← scale(df[numeric_cols])

# Changes chol & gluc to meaningful labels
df$cholesterol ← factor(df$cholesterol, levels = c(1,2,3),
                        labels = c("normal", "above_normal", "well_above_normal"))
df$gluc ← factor(df$gluc, levels = c(1,2,3),
                 labels = c("normal", "above_normal", "well_above_normal"))

# Convert chol & gluc to dummy variables, one-hot encoding
df ← cbind(df, model.matrix(~ cholesterol + gluc - 1, data = df))
df ← df %>% select(-cholesterol, -gluc)

# Convert cardio values to factors
# Changing this temporarily, model will need this as 0 or 1
df$cardio ← factor(df$cardio, levels = c(0,1), labels = c("No", "Yes"))
#df$cardio ← factor(df$cardio, levels = c(0,1))


# Save
write.csv(df, "data/cardio_model_ready.csv")


# SECTION FIVE: Data Splitting

# Using the feature engineered dataset
# Selecting the whole dataset
df ← read.csv("data/cardio_model_ready.csv")

# Setting a seed so can be performed again
set.seed(1)

# Creating the partition
```

```r
# 70% training, 30% testing
partition ← createDataPartition(df$cardio, p = 0.7, list=FALSE)

# Allocating training and testing data
trainingData ← df[partition, ]
testData ← df[-partition, ]

# Checking proportion of cardio data is roughly even
table(trainingData$cardio)
table(testData$cardio)

# Saving for use in model
write.csv(trainingData, "data/trainingData.csv", row.names = FALSE)
write.csv(testData, "data/testData.csv", row.names = FALSE)

print("Data splitting completed. Datasets saved as testData.csv and
trainingData.csv")

# SECTION SIX: Creation and evaluation of a simple logistic regression model
# We need to consider why we've used this and alternative things
# Do we want to see if we can predict it based on only one category like age
or BMI?
# This model has an accuracy of roughly 0.7279 which is pretty decent!
# This is adapted from the lab with the logistic regression tutorial
# cholesterol well above normal coming up as NA

log_model ← glm(cardio ~., data= trainingData,
family=binomial(link="logit"))

# Make Predictions
log_predictions ← predict(log_model, testData, type = "response")
log_predicted_classes ← ifelse(log_predictions > 0.5, 1, 0)


# Begin evaluation of model using a confusion matrix
# Big issues here, data and reference should be factors with the same levels
# Fix here, but if have time need to go back and check
log_conf_matrix ← confusionMatrix(
  factor(log_predicted_classes, levels = levels(factor(testData$cardio))),
  factor(testData$cardio, levels = levels(factor(testData$cardio)))
)

print(log_conf_matrix)
# Get F1 from the confusion matrix
log_conf_matrix$byClass["F1"]


# Printing a summary of the model here
summary(log_model)

```

```r
# Get feature importance using p-values
p_values <- model_summary$coefficients[-1, 4]
importance_df_p <- data.frame(
  Feature = names(p_values),
  P_Value = p_values,
  Importance = -log10(p_values)
)
importance_df_p <- importance_df_p[order(-importance_df_p$Importance),]

most_significant_feature <- importance_df_p[1, "Feature"]
most_significant_pvalue <- importance_df_p[1, "P_Value"]

# Print paste seems to work for formatting strings
print(paste("Most statistically significant feature:", most_significant_feature,
            "with p-value:", most_significant_pvalue))

# computing odds ratios and confidence intervals
exp(coef(log_model))
exp(cbind(OR = coef(log_model), confint(log_model)))

# AUC ROC - Sensitivity vs specificity
roc_object <- roc( testData$cardio, log_predictions)
rocCurve <- ggroc(roc_object) + ggtitle("ROC Curve for Logistic Regression Model")
ggsave("results/logistic/roc.png", plot = rocCurve)

# AUC = 0.7897, the closer the auc is to 1, the better the model
# Close to 1: Good at distinguishing between positive and negative classes
# Close to 0.5: Performs no better than random guessing
# https://www.geeksforgeeks.org/plotting-roc-curve-in-r-programming/
auc(roc_object)

# Trying out logistic curves
# Original go results in a very spiky plot, this makes things readable
# The shaded section shows the confidence interval, automatically 95%
ageLC <- ggplot(testData, aes(x = age_years, y = log_predictions)) +
  geom_smooth(method = "loess", color = "steelblue") +
  labs(title = "Smoothed Logistic Regression Curve for age",
       x = "Age in years",
       y = "Probability of CVD")
ggsave("results/logistic/cvdByAgeLC.png", plot = ageLC)

bmiLC <- ggplot(testData, aes(x = bmi, y = log_predictions)) +
  geom_smooth(method = "loess", color = "steelblue") +
  labs(title = "Logistic Regression Curve for BMI",
       x = "BMI",
       y = "Predicted Probability of CVD")
ggsave("results/logistic/cvdByBMILC.png", plot = bmiLC)


```

```r
ap_hiLC <- ggplot(testData, aes(x = ap_hi, y = log_predictions)) +
  geom_smooth(method = "loess", color = "steelblue") +
  labs(title = "Logistic Regression Curve for ap_hi",
       x = "ap_hi",
       y = "Predicted Probability of CVD")
ggsave("results/logistic/cvdByap_hiLC.png", plot = ap_hiLC)

ap_loLC <- ggplot(testData, aes(x = ap_lo, y = log_predictions)) +
  geom_smooth(method = "loess", color = "steelblue") +
  labs(title = "Logistic Regression Curve for ap_lo",
       x = "ap_lo",
       y = "Predicted Probability of CVD")
ggsave("results/logistic/cvdByap_loLC.png", plot = ap_loLC)


# HEYO! Gender is NOT a continuous variable, however we have it represented as 1 or 2.
# This plot shows us that this model predicts that men (2) are more likely to have CVD,
# however it presents it as a continuous variable. The question is do we leave it as this,
# or do we find a new type of plot that represents binary variables better?
# I really like the look of this graph, it shows things clearly
genderPlotLC <- ggplot(testData, aes(x = gender, y = log_predictions)) +
  geom_smooth(method = "loess", color = "steelblue") +
  labs(title = "Logistic Regression Curve for Gender",
       x = "Gender (1 = Woman, 2 = Man)",
       y = "Predicted Probability of CVD")
ggsave("results/logistic/cvdByGenderLC.png", plot = genderPlotLC)

# Not as nice graph, but technically represents things better
genderPlot <- ggplot(testData, aes(x = gender, y = log_predictions)) +
  stat_summary(fun = mean, geom = "bar", position = position_dodge(), fill="steelblue") +
  labs(title = "Probability of CVD by Gender",
       x = "Gender (1 = Woman, 2 = Man)",
       y = "Predicted Probability of CVD")
ggsave("results/logistic/cvdByGender.png", plot = genderPlot)


smokePlot <- ggplot(testData, aes(x = smoke, y = log_predictions)) +
  stat_summary(fun = mean, geom = "bar", position = position_dodge(), fill="steelblue") +
  labs(title = "Probability of CVD by Smoking",
       x = "Smoking (0 = Does not smoke, 1 = Does smoke)",
       y = "Predicted Probability of CVD")
ggsave("results/logistic/cvdBySmoke.png", plot = smokePlot)


alcoPlot <- ggplot(testData, aes(x = smoke, y = log_predictions)) +
  stat_summary(fun = mean, geom = "bar", position = position_dodge(),
```

```
fill="steelblue") +
  labs(title = "Probability of CVD by Alcohol",
       x = "Alcohol (0 = Does not drink, 1 = Does drink)",
       y = "Predicted Probability of CVD")
ggsave("results/logistic/cvdByAlco.png", plot = alcoPlot)


activePlot ← ggplot(testData, aes(x = active, y = log_predictions)) +
  stat_summary(fun = mean, geom = "bar", position = position_dodge(),
fill="steelblue") +
  labs(title = "Probability of CVD by Activity",
       x = "Activity (0 = Inactive, 1 = Active)",
       y = "Predicted Probability of CVD")
ggsave("results/logistic/cvdByActivity.png", plot = activePlot)


# I wanted to explore the possibility of smoking interfering with the gender
predictions
# This doesn't seem to be the case, this shows the probability of men and
women who smoke and don't
# The model reckons we should all take up smoking
genderSmoke ← ggplot(testData, aes(x = interaction(gender, smoke), y =
log_predictions, fill = interaction(gender, smoke))) +
  stat_summary(fun = mean, geom = "bar", position = position_dodge()) +
  labs(title = "Probability of CVD by Gender and Smoking",
       x = "Group",
       y = "Mean Probability of CVD")
ggsave("results/logistic/cvdByGenderSmoking.png", plot = genderSmoke)

# Similarly, the odds of having CVD and drinking alcohol are lower across the
board
genderAlco ← ggplot(testData, aes(x = interaction(gender, alco), y =
log_predictions, fill = interaction(gender, alco))) +
  stat_summary(fun = mean, geom = "bar", position = position_dodge()) +
  labs(title = "Probability of CVD by Gender and Alcohol",
       x = "Group",
       y = "Mean Probability of CVD")
ggsave("results/logistic/cvdByGenderAlcohol.png", plot = genderAlco)

# Saving the model
# We might want to move this up before the evaluation, not sure it really
matters
saveRDS(log_model, "results/models/cardio_logistic_model.rds")

print("Logistic Regression model created and saved as
cardio_logistic_model.rds")

# SECTION SEVEN: Creation and Evaluation of a K-Nearest Neighbors (KNN) Model
# We're using KNN to classify whether a person has cardiovascular disease
based on all available features.
# This approach looks at the 'k' nearest patients and makes a prediction
```

```
based on what class most of them belong to.
# NEEDS ROC, AUC AND SUMMARY, PROB CONF INTERVALS

# Train the KNN model using 10-fold cross-validation
# We scale the data to ensure fairness in distance calculations
# The issue here is because cardio is being treated as numeric instead of categorical
# Fixed this
trainingData$cardio ← as.factor(trainingData$cardio)
testData$cardio ← as.factor(testData$cardio)

knn_model ← train(cardio ~ ., data = trainingData, method = "knn",preProcess = c("center", "scale"), trControl = trainControl(method = "cv", number = 10))

# Make predictions on the test set
knn_predictions ← predict(knn_model, testData)
knn_probs ← predict(knn_model, testData, type = "prob")[,2]            # Probabilities for class 1

# Performs best when using 9 nearest neighbours
print(knn_model)

# FIXED THIS
# Evaluate the model using a confusion matrix
knn_conf_matrix ← confusionMatrix(
  factor(knn_predictions, levels = levels(testData$cardio)),
  factor(testData$cardio, levels = levels(testData$cardio))
)

# Display the results
print("Confusion Matrix for KNN Model:")
print(knn_conf_matrix)


# Printing a summary of the model here
summary(knn_model)


# AUC ROC - Sensitivity vs specificity
# 0.7629
knn_roc_object ← roc( testData$cardio, knn_probs)
knn_rocCurve ← ggroc(knn_roc_object) + ggtitle("ROC Curve for K-NN Model")
ggsave("results/knn/roc.png", plot = rocCurve)

# AUC
auc(knn_roc_object)

knn_conf_matrix$byClass["F1"]


# Visualize predicted CVD status across BMI
```

```
620 knn_bmi_plot ← ggplot(testData, aes(x = bmi, fill = knn_predictions)) +
621   geom_density(alpha = 0.5) +
622   labs(title = "KNN: Predicted CVD Probability by BMI",
623        x = "BMI",
624        fill = "Predicted CVD")
625 ggsave("results/knn/knnByBMI.png", plot = knn_bmi_plot)
626
627 # Visualize predicted CVD by gender
628 knn_gender_plot ← ggplot(testData, aes(x = gender, fill = knn_predictions))
    + geom_bar(position = "fill") +
629   scale_y_continuous(labels = scales::percent) +
630   labs(title = "KNN: Gender Distribution by Predicted CVD",
631        x = "Gender (1 = Woman, 2 = Man)",
632        y = "Proportion of Patients",
633        fill = "Predicted CVD")
634 ggsave("results/knn/knnByGender.png", plot = knn_gender_plot)
635
636 # Get variable importance
637 importance ← varImp(knn_model)
638 importance_df$Feature ← rownames(importance_df)
639 importance_df ← importance_df[order(-importance_df[,1]),]
640 most_important_feature ← importance_df[1, "Feature"]
641 most_important_value ← importance_df[1, 1]
642 print(paste("Most important feature:", most_important_feature,
643             "with importance value:", most_important_value))
644
645
646
647 # Save the trained model for later evaluation
648 saveRDS(knn_model, "results/models/cardio_knn_model.rds")
649 print("KNN model trained and saved as cardio_knn_model.rds")
650
651
652
653 # SECTION EIGHT: Creation and Evaluation of a Random Forest Model
654
655
656 # Ensure cardio is treated as factor for classification
657 trainingData$cardio ← as.factor(trainingData$cardio)
658 testData$cardio ← as.factor(testData$cardio)
659
660 # Train the Random Forest model
661 rf_model ← randomForest(cardio ~ ., data = trainingData, ntree = 100,
    importance = TRUE)
662
663 # Save the trained model
664 saveRDS(rf_model, "results/models/cardio_randomforest_model.rds")
665 print(" Random Forest model saved as cardio_randomforest_model.rds")
666
667 # --- Predictions ---
668 rf_predictions ← predict(rf_model, testData, type = "class")        #
```

```
Predicted classes
rf_probs ← predict(rf_model, testData, type = "prob")[,2]            # Probabilities for class 1

# --- Confusion Matrix ---
rf_conf_matrix ← confusionMatrix(
  factor(rf_predictions, levels = levels(testData$cardio)),
  factor(testData$cardio, levels = levels(testData$cardio))
)
print("Confusion Matrix for Random Forest Model:")
print(rf_conf_matrix)

# Get F1
rf_conf_matrix$byClass["F1"]


# --- ROC & AUC ---
rf_roc ← roc(testData$cardio, rf_probs)
rf_auc ← auc(rf_roc)
print(paste("AUC for Random Forest:", rf_auc))

# Save ROC plot
rf_roc_plot ← ggroc(rf_roc) + ggtitle("ROC Curve for Random Forest Model")
ggsave("results/randomforest/roc.png", plot = rf_roc_plot)

# --- Variable Importance ---
varImpPlot(rf_model, main = "Random Forest - Variable Importance")
write.csv(importance(rf_model), "results/randomforest/variable_importance.csv")

# --- Probability Summary with Confidence Intervals ---
# Bootstrap confidence interval for mean predicted probabilities
rf_ci ← t.test(rf_probs ~ testData$cardio)$conf.int
print(" 95% Confidence Interval for Predicted Probabilities (CVD vs No CVD):")
print(rf_ci)

# Save ROC stats and confidence interval summary
rf_eval ← data.frame(
  Accuracy = rf_conf_matrix$overall["Accuracy"],
  Kappa = rf_conf_matrix$overall["Kappa"],
  Sensitivity = rf_conf_matrix$byClass["Sensitivity"],
  Specificity = rf_conf_matrix$byClass["Specificity"],
  AUC = as.numeric(rf_auc),
  CI_Lower = rf_ci[1],
  CI_Upper = rf_ci[2]
)

write.csv(rf_eval, "results/randomforest/evaluation_summary.csv", row.names = FALSE)
print("Evaluation summary for Random Forest saved to results/randomforest/
```

```
evaluation_summary.csv")

# Print summary table to console
print(rf_eval)


# SECTION NINE: Any further eval & model comparison

# Reckon it's useful to combine all rocs into one plot
roc_list <- list(roc_object, knn_roc_object,rf_roc)

# Create the plot for all ROC curves
# 1= Log, 2= knn, 4= rf
all_rocs_plot <- ggroc(roc_list) +
  ggtitle("The ROC curves for Models")
ggsave("results/all_rocs_plot.png", plot =all_rocs_plot)

```