Code Appendix

```
setwd(dirname(rstudioapi::getSourceEditorContext()$path))#Makes current folder working directory

#Install and load necessary packages


install.packages("DataExplorer")

install.packages("corrplot")

install.packages("ggplot2")

install.packages("e1071")

install.packages("caret")

install.packages("gridExtra")

install.packages("plotly")

install.packages("RColorBrewer")

install.packages("reshape2")

install.packages("lattice")

install.packages("dplyr")

install.packages("tidyr")

install.packages("stringr")

install.packages("tidyverse")

install.packages("plot.matrix")

install.packages("kernlab")

install.packages("GGally")

install.packages("pROC")



# Load the packages


library(caret)
```

```
library(ggplot2)
library(tidyverse)
library(corrplot)
library(gridExtra)
library(GGally)
library(knitr)
library(dplyr)
library(DataExplorer)
library(e1071)
library(plotly)
library(plot.matrix)
library(kernlab)
library(pROC)



#Importing of Data set
NEW_DB <- read.csv("DB_R 1.csv",stringsAsFactors =TRUE,header=TRUE)

                              #EXPLORATORY DATA ANALYSIS(EDA)
#checking THE STRUCTURE OF THE DATASET
str(NEW_DB)
#checking THE SUMMARY O MY DATASET
summary(NEW_DB)
anyNA(NEW_DB)
#plotting missing values
plot_missing(NEW_DB)

#plotting MY data
```

```
plot_bar(NEW_DB)


# Create a bar plot for gender
ggplot(data = NEW_DB) +
  geom_bar(mapping = aes(x = gender), fill = "orange") +
  labs(title = "Gender Distribution", x = "Gender", y = "Count")


# Create a bar plot for smoking history
ggplot(data = NEW_DB) +
  geom_bar(mapping = aes(x = smoking_history), fill = "lightgreen") +
  labs(title = "Smoking History Distribution", x = "Smoking History", y =
"Count")


# Create a bar plot for hypertension
ggplot(data = NEW_DB) +
  geom_bar(mapping = aes(x = factor(hypertension), fill =
factor(hypertension))) +
  labs(title = "Hypertension Distribution", x = "Hypertension", y = "Count") +
  scale_fill_manual(values = c("lightcoral", "lightblue"), name =
"Hypertension")



# Create a box plot for Blood Glucose Level by Diabetes Status with a custom
title
ggplot(data =NEW_DB, mapping = aes(x = factor(diabetes), y =
blood_glucose_level)) +
  geom_boxplot(color = "black") +
  labs(title = "Boxplot of Blood Glucose Level by Diabetes Status", x =
"Diabetes Status", y = "Blood Glucose Level")
```

```
# Create a density plot for Blood Glucose Level by Diabetes Status
ggplot(NEW_DB, aes(x = blood_glucose_level, fill = factor(diabetes))) +
  geom_density(alpha = 0.5) +
  labs(title = "Density Plot of Blood Glucose Level by Diabetes Status", x = "Blood Glucose Level", fill = "Diabetes Status")


#Investigate the proportion of individuals with heart disease by gender.
#This stacked bar chart shows the proportion of individuals with and without heart disease for each gender.
ggplot(NEW_DB, aes(x = gender, fill = factor(heart_disease))) +
  geom_bar(position = "fill", alpha = 0.7) +
  labs(title = "Heart Disease Proportion by Gender", x = "Gender", y = "Proportion") +
  scale_fill_manual(values = c("0" = "skyblue", "1" = "orange"))


#Bmi distribution by gender
# Compare the distribution of BMI across different genders.
boxplot(bmi ~ gender, data = NEW_DB, main = "BMI Distribution by Gender", col = c("skyblue", "pink"))


#distribution of HbA1c levels in the data set.
hist(NEW_DB$HbA1c_level, main = "Distribution of HbA1c Levels", xlab = "HbA1c Level")


#blood glucose level by diabetes status
#Compare blood glucose levels between individuals with and without diabetes.
boxplot(blood_glucose_level ~ diabetes, data = NEW_DB, main = "Blood Glucose Level by Diabetes Status", col = c("skyblue", "pink"))
```

```
#Visualize the distribution of smoking history
pie(table(NEW_DB$smoking_history), main = "Smoking History", col = c("skyblue", "pink", "lightgreen"))


#Compare HbA1c levels based on the diabetes status.
#This box plot compares HbA1c levels for different diabetes statuses, with different colors for each gender.
library(ggplot2)
ggplot(NEW_DB, aes(x = diabetes, y = HbA1c_level, fill = gender)) +
  geom_boxplot() +
  labs(title = "HbA1c Level by Diabetes Status", x = "Diabetes Status", y = "HbA1c Level") +
  scale_fill_manual(values = c("skyblue", "pink"))


# Exclude specified columns
db_plot <- select(NEW_DB, -gender, -heart_disease, -hypertension, -diabetes)
# Set up colors
boxplot_colors <- c("skyblue", "lightgreen", "lightcoral", "pink", "orange")
# Plot boxplots for all numeric variables
plot(db_plot[, sapply(db_plot, is.numeric)])
# Increase the size of the plotting device
dev.new(width = 10, height = 10)
# Set up a layout for multiple plots
par(mfrow = c(3, 3))  # 3 rows, 3 columns
# Create individual boxplots for each numeric variable
# Create individual boxplots for each numeric variable
for (i in seq_along(db_plot)) {
  if (is.numeric(db_plot[[i]])) {
    # Reduce margins for individual boxplots
```

```
    par(mar = c(3, 3, 1, 1))
    boxplot(db_plot[[i]], main = names(db_plot)[i], col = boxplot_colors[i])
  }
}
# Reset the layout
par(mfrow = c(1, 1))

# Reset the layout
par(mfrow = c(1, 1))



#few statistics
#check the column names
 names(NEW_DB)

# Selecting columns with numeric data
numeric_columns <- NEW_DB[, c("age", "bmi", "HbA1c_level",
"blood_glucose_level")]
# Summary statistics
summary_stats <- summary(numeric_columns)
# Standard deviation
sd_values <- apply(numeric_columns, 2, sd)
# Variance
variance_values <- apply(numeric_columns, 2, var)
# Display the results
print("Summary Statistics:")
print(summary_stats)
print("Standard Deviation:")
```

```
print(sd_values)

print("Variance:")

print(variance_values)

                    #DATA CLEANING

#CONVERTING ALL VARIABLES EXCEPT TARGET VARIABLE TO NUMERIC

NEW_DB$gender <- as.numeric(NEW_DB$gender)

NEW_DB$age <- as.numeric(NEW_DB$age)

NEW_DB$hypertension <- as.numeric(NEW_DB$hypertension)

NEW_DB$heart_disease <- as.numeric(NEW_DB$heart_disease)

NEW_DB$smoking_history <- as.numeric(NEW_DB$smoking_history)

NEW_DB$bmi <- as.numeric(NEW_DB$bmi)

NEW_DB$HbA1c_level <- as.numeric(NEW_DB$HbA1c_level)

NEW_DB$blood_glucose_level <- as.numeric(NEW_DB$blood_glucose_level)

str(NEW_DB)

                    #CORRELATION

# Feature selection by excluding the "diabetes" column

df_core <- select(NEW_DB, -diabetes)

# Correlation plot

correlation_matrix <- cor(df_core)

corrplot(correlation_matrix, method = "circle", type = "lower")

# Find highly correlated variables

highly_correlated <- findCorrelation(correlation_matrix, cutoff = .6, verbose = TRUE, names = TRUE)

highly_correlated


#Feature scaling or Normalization.

p1 <- preProcess(NEW_DB[, c(1:8)], method = c("center", "scale"))
```

```
NEW_DB_scaled <- predict(p1, NEW_DB[, c(1:8)])
NEW_DB_scaled <- cbind(NEW_DB_scaled, diabetes = NEW_DB$diabetes)

# Check the resulting data set
head(NEW_DB_scaled)
tail(NEW_DB_scaled)
#DATA PREPROCESSING

# Count occurrences of each category
count <- table(NEW_DB_scaled$diabetes)
# Calculate proportions
proportions <- prop.table(count)
# Calculate percentages
percentages <- prop.table(count) * 100
# Print the results
count
proportions
percentages

#SHUFFLING DATASET
set.seed(77)
#shuffles the row of the data frame#
NEW_DB_shuffled<-NEW_DB_scaled %>%
  sample_n(size = nrow(NEW_DB_scaled),
  replace=FALSE)

                    #MACHINE LEARNING
# Create a data partition for training (80%) and testing (20%)
```

```
intrain <- createDataPartition(y = NEW_DB_shuffled$diabetes, p = 0.8, list = FALSE)
# Split the data
training <- NEW_DB_shuffled[intrain, ]
testing <- NEW_DB_shuffled[-intrain, ]

# View the dimensions and structure of the training and testing sets
dim(training)
dim(testing)
str(training)
str(testing)
                    #Support vector machine

training$diabetes <- factor(training$diabetes, levels = c(0, 1))

# SET SEED TO ENSURE REPRODUCIBILITY OF RANDOM NUMBERS
set.seed(123)
#setting train control
trctrl <- trainControl(method = "repeatedcv",
                       number = 10,
                       repeats = 5,
                       summaryFunction = defaultSummary)


# TRAIN SVM MODEL
set.seed(7)
svm_model <- train(diabetes ~ .,
                   data = training,
```

```
                method = "svmLinear",
                trControl = trctrl,
                preProcess = c("center", "scale"),
                tuneLength = 10,
                metric = "Accuracy")
svm_model

# Predict on the testing set
test_pred <- predict(svm_model, newdata = testing)

# Display the predictions
print(test_pred)

# Calculate confusion matrix for SVM model
confusion_matrix <- confusionMatrix(table(test_pred, testing$diabetes))

# Display the confusion matrix
print(confusion_matrix)

# Print accuracy
accuracy <- confusion_matrix$overall["Accuracy"]
print(paste("Accuracy: ", round(accuracy, 4)))

#Comparing svm and KNN.
# Train the KNN model
# Train the KNN model
# Convert to factor with the same levels
training$diabetes <- as.factor(training$diabetes)
```

```
testing$diabetes <- as.factor(testing$diabetes)

# Check factor levels
levels(training$diabetes)
levels(testing$diabetes)



set.seed(7)
fit.knn <- train(diabetes ~ ., data = training,
                 method = "knn", preProcess = c("center", "scale"),
                 trControl = trainControl())
conf_matrix_knn <- confusionMatrix(predict(fit.knn, newdata = testing),
testing$diabetes)
print("Confusion Matrix for KNN:")
print(conf_matrix_knn)




# Train the SVM model
set.seed(7)
fit.svm <- train(diabetes ~ ., data = training,
                 method = "svmLinear", preProcess = c("center", "scale"),
                 trControl = trainControl())

# Make predictions on the testing set
predictions <- predict(fit.svm, newdata = testing)
```

```
# Calculate confusion matrix for SVM model
conf_matrix_svm <- confusionMatrix(predictions, testing$diabetes)

# Display the confusion matrix for SVM
cat("Confusion Matrix for SVM:\n")
print(conf_matrix_svm)



# Compare the models using resamples
comp <- resamples(list(SVM = fit.svm, KNN = fit.knn))

# Display summary of the comparison
summary(comp)
dotplot(comp)



####ROC CURVE
#ROC CURVE FOR SVM MODEL
#create a table for diabetes
y_true <- testing$diabetes
#convert to numeric
y_true <- as.numeric(as.character(y_true))
svm_scores <- predict(svm_model, testing, decision.values = TRUE)
svm_scores <- as.numeric(as.character(svm_scores))
# Create a ROC curve object
roc_curve_svm <- roc(y_true, svm_scores)
# Plot the ROC curve
plot(roc_curve_svm, col = "skyblue", main = "ROC Curve - SVM", lwd = 2)
```

```
# Add AUC value to the plot
text(0.8, 0.2, paste("AUC =", round(auc(roc_curve_svm), 2)), col = "red", cex = 1.5)



# ROC CURVE FOR KNN MODEL
# create a table for DIABETES
y_true <- testing$diabetes
knn_scores <- predict(knn_model, testing, decision.values = TRUE)
knn_scores <- as.numeric(as.character(knn_scores))
# Create a ROC curve object for KNN
roc_curve_knn <- roc(y_true, knn_scores)

# Plot the ROC curve for KNN
plot(roc_curve_knn, col = "skyblue", main = "ROC Curve - KNN", lwd = 2)  # Fix variable name typo

# Add AUC value to the plot
text(0.8, 0.2, paste("AUC =", round(auc(roc_curve_knn), 2)), col = "red", cex = 1.5)  # Fix variable name typo
```