Diabetes Prediction

Quyen Di Sabino

2024-09-06

A. Introduction:

A.1. Describes the dataset and variables:

In this data analysis project, we focus on a healthcare dataset that contains crucial measurements related to diabetes, collected from a sample of 768 individuals. The dataset includes various features such as the number of pregnancies, glucose levels, blood pressure, skin thickness, insulin levels, BMI, diabetes pedigree function, and age, along with an outcome variable indicating the presence or absence of diabetes.

A.2. Summarizes the goal of the project:

The primary objective of this project is to perform comprehensive data cleaning and preprocessing to prepare the dataset for further analysis. One of the key preprocessing tasks involves handling specific data issues, such as replacing zero values in critical columns with the mean of non-zero values. This approach ensures that the dataset is robust and free from anomalies that could distort subsequent analyses or predictive modeling.

A.3. Key steps that were performed.

A.3.1. Install (if needed) and load necessary libraries\
A.3.2. Load and inspect the data\
A.3.3. Data cleaning\
A.3.4. Data statistic and visualization of relationships\
A.3.5. Predictive Modeling\

B. Methods/Analysis

B.1. Install If Needed and Load Necessary Libraries

```
if(!require(tidyverse)) install.packages("tidyverse")
```

```
## Loading required package: tidyverse
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.4     v readr     2.1.5
## v forcats   1.0.0     v stringr   1.5.1
## v ggplot2   3.5.1     v tibble    3.2.1
## v lubridate 1.9.3     v tidyr     1.3.1
```

```
## v purrr     1.0.2
## -- Conflicts ---------------------------------------- tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
```

```
if(!require(caret)) install.packages("caret")
```

```
## Loading required package: caret
## Loading required package: lattice
##
## Attaching package: 'caret'
##
## The following object is masked from 'package:purrr':
##
##     lift
```

```
if(!require(Metrics)) install.packages("Metrics")
```

```
## Loading required package: Metrics
##
## Attaching package: 'Metrics'
##
## The following objects are masked from 'package:caret':
##
##     precision, recall
```

```
if(!require(knitr)) install.packages("knitr")
```

```
## Loading required package: knitr
```

```
if(!require(e1071)) install.packages("e1071")
```

```
## Loading required package: e1071
```

```
if(!require(gridExtra))install.packages("gridExtra")
```

```
## Loading required package: gridExtra
##
## Attaching package: 'gridExtra'
##
## The following object is masked from 'package:dplyr':
##
##     combine
```

```
if(!require(pheatmap))install.packages("pheatmap")
```

```
## Loading required package: pheatmap
```

```r
library(tidyverse)
library(caret)
library(Metrics)
library(knitr)
library(e1071)
library(gridExtra)
library(pheatmap)
```

B.2. Load and Inspect the Data

```r
# Controls the number of digits to print when printing numeric values
options(digits = 6)

# Set options to avoid scientific notation
options(scipen = 10)

# Load data
dat <- read_csv("diabetes.csv")
```

```
## Rows: 768 Columns: 9
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## dbl (9): Pregnancies, Glucose, BloodPressure, SkinThickness, Insulin, BMI, D...
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```r
# Data dimension
dim(dat)
```

```
## [1] 768   9
```

```r
# Inspect the structure of the dataset
str(dat)
```

```
## spc_tbl_ [768 x 9] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
##  $ Pregnancies             : num [1:768] 6 1 8 1 0 5 3 10 2 8 ...
##  $ Glucose                 : num [1:768] 148 85 183 89 137 116 78 115 197 125 ...
##  $ BloodPressure           : num [1:768] 72 66 64 66 40 74 50 0 70 96 ...
##  $ SkinThickness           : num [1:768] 35 29 0 23 35 0 32 0 45 0 ...
##  $ Insulin                 : num [1:768] 0 0 0 94 168 0 88 0 543 0 ...
##  $ BMI                     : num [1:768] 33.6 26.6 23.3 28.1 43.1 25.6 31 35.3 30.5 0 ...
##  $ DiabetesPedigreeFunction: num [1:768] 0.627 0.351 0.672 0.167 2.288 ...
##  $ Age                     : num [1:768] 50 31 32 21 33 30 26 29 53 54 ...
##  $ Outcome                 : num [1:768] 1 0 1 0 1 0 1 0 1 1 ...
##  - attr(*, "spec")=
##   .. cols(
##   ..   Pregnancies = col_double(),
##   ..   Glucose = col_double(),
##   ..   BloodPressure = col_double(),
```

```
##   ..   SkinThickness = col_double(),
##   ..   Insulin = col_double(),
##   ..   BMI = col_double(),
##   ..   DiabetesPedigreeFunction = col_double(),
##   ..   Age = col_double(),
##   ..   Outcome = col_double()
##   .. )
##  - attr(*, "problems")=<externalptr>
```

```
# View data
View(dat)

# Summary statistics
summary(dat)
```

```
##   Pregnancies       Glucose    BloodPressure   SkinThickness     Insulin
##  Min.   : 0.00   Min.   :  0   Min.   :  0.0   Min.   : 0.0   Min.   :  0.0
##  1st Qu.: 1.00   1st Qu.: 99   1st Qu.: 62.0   1st Qu.: 0.0   1st Qu.:  0.0
##  Median : 3.00   Median :117   Median : 72.0   Median :23.0   Median : 30.5
##  Mean   : 3.85   Mean   :121   Mean   : 69.1   Mean   :20.5   Mean   : 79.8
##  3rd Qu.: 6.00   3rd Qu.:140   3rd Qu.: 80.0   3rd Qu.:32.0   3rd Qu.:127.2
##  Max.   :17.00   Max.   :199   Max.   :122.0   Max.   :99.0   Max.   :846.0
##       BMI       DiabetesPedigreeFunction      Age          Outcome
##  Min.   : 0.0   Min.   :0.078            Min.   :21.0   Min.   :0.000
##  1st Qu.:27.3   1st Qu.:0.244            1st Qu.:24.0   1st Qu.:0.000
##  Median :32.0   Median :0.372            Median :29.0   Median :0.000
##  Mean   :32.0   Mean   :0.472            Mean   :33.2   Mean   :0.349
##  3rd Qu.:36.6   3rd Qu.:0.626            3rd Qu.:41.0   3rd Qu.:1.000
##  Max.   :67.1   Max.   :2.420            Max.   :81.0   Max.   :1.000
```

```
# Frequency table of the Outcome
table(dat$Outcome)
```

```
##
##   0   1
## 500 268
```

B.3 Data Cleaning

```
# B.3.1. Missing values

# Identify and count any missing values in each column of a data frame dat
missing_values <- colSums(is.na(dat))
print(missing_values) # zero missing values
```

```
##              Pregnancies                  Glucose            BloodPressure
##                        0                        0                        0
##            SkinThickness                  Insulin                      BMI
##                        0                        0                        0
## DiabetesPedigreeFunction                      Age                  Outcome
##                        0                        0                        0
```

```r
# B.3.2. Zero valuses

# Identify and count any zero values that may require attention.
n_zeros <- sapply(dat, function(x) sum(x==0))
print(n_zeros)
```

```
##              Pregnancies                  Glucose            BloodPressure
##                      111                        5                       35
##            SkinThickness                  Insulin                      BMI
##                      227                      374                       11
## DiabetesPedigreeFunction                      Age                  Outcome
##                        0                        0                      500
```

```r
# These non sense zeros in BloodPressure, BMI, Glucose, Insulin, SkinThickness are not actual observati
# They may be due to :
# Measurement error: device and testing
# Severe medical condition of the patients
# Data entry mistake
# Calculation error.

# Replace zeros in columns where they don't make sense (e.g., BloodPressure, BMI, Glucose, Insulin, Ski
# Function to replace zeros with column mean
replace_zeros_with_mean <- function(x) {
  mean_value <- mean(x[x != 0], na.rm = TRUE) # Calculate mean of non-zero values
  x[x == 0] <- mean_value # Replace zeros with the mean value
  return(x)
}

# Columns with nonsense zeros
columns_with_nonsense_zeros <- c("BloodPressure", "BMI", "Glucose", "Insulin", "SkinThickness")

# Apply the function to columns with nonsense zeros using lapply
dat[columns_with_nonsense_zeros] <- lapply(dat[columns_with_nonsense_zeros], replace_zeros_with_mean)


# B.3.3. Outlinears

# Identify outlinears
dat %>% select("Insulin") %>% filter(Insulin > 500) %>% nrow()
```

```
## [1] 9
```

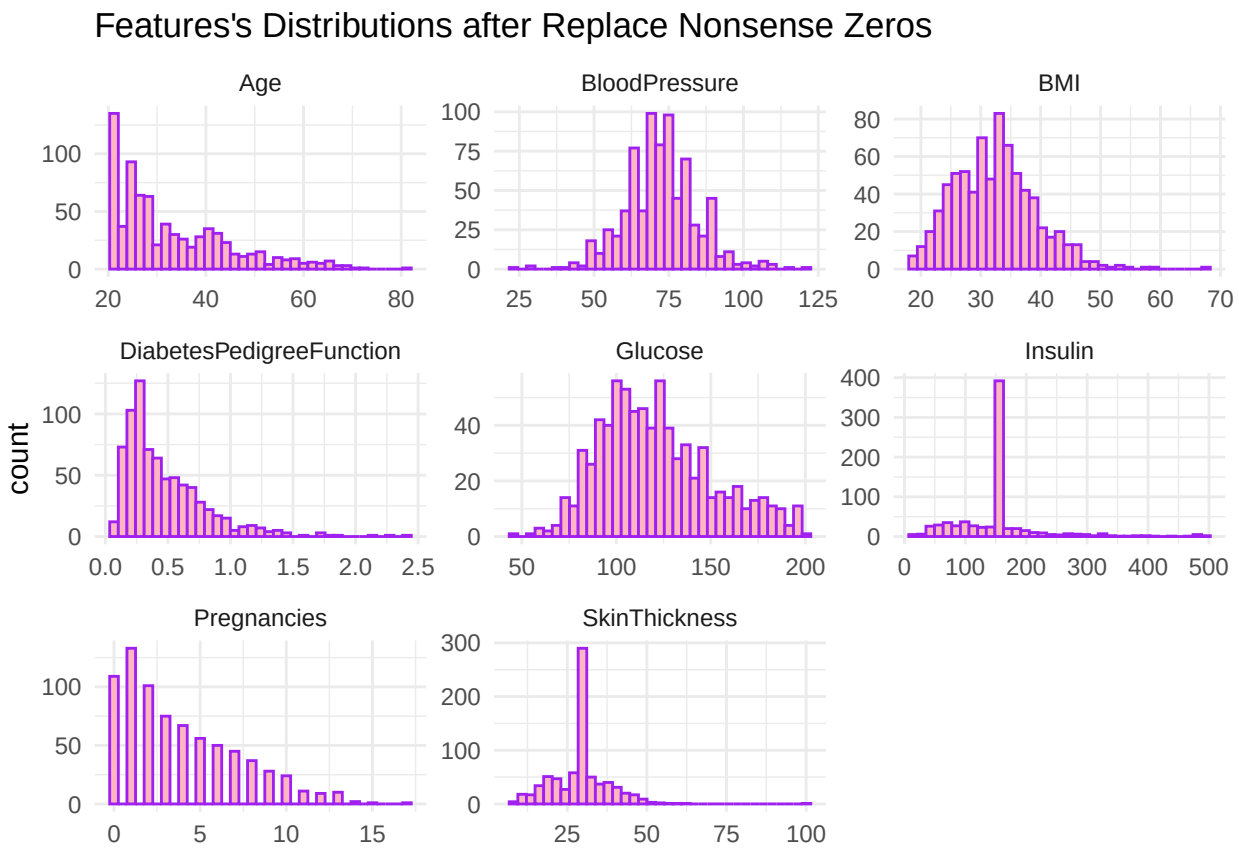
```r
# Elevated insulin levels might occur in individuals with type 2 diabetes or a precursor.
# But this significant high insulin may be due to rare insulin-secreting tumor or out linear

# Remove rows where insulin > 500
dat <- dat[dat$Insulin <= 500, ]
```

B.4. Exploratory Data Analysis (EDA)

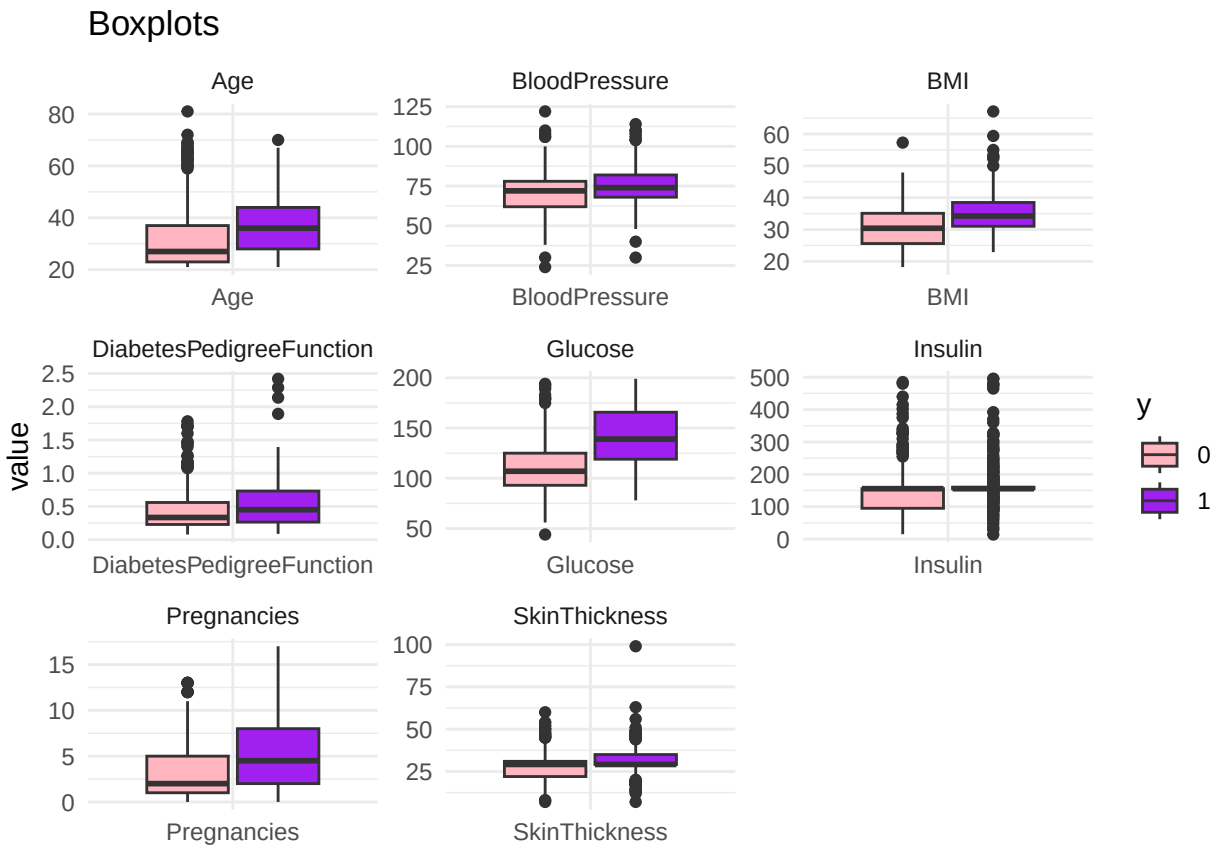
```r
# B.4.1. Data statistic and visualization of relationships

# Data statistics
summary_df <- dat %>%
  # diff(range(.)) computes the difference between the maximum and minimum values.
  summarise(across(everything(), list(mean = mean, sd = sd, median = median, range = ~ diff(range(.))))) %>%
  pivot_longer(cols = everything(), names_to = c(".value", "feature"), names_sep = "_")

# Print the resulting summary data frame
print(summary_df)
```

```
## # A tibble: 4 x 10
##   feature Pregnancies Glucose BloodPressure SkinThickness Insulin   BMI
##   <chr>         <dbl>   <dbl>         <dbl>         <dbl>   <dbl> <dbl>
## 1 mean           3.86   121.           72.4         29.1    150.  32.4
## 2 sd             3.38    30.0          12.1          8.75    67.7  6.85
## 3 median         3      117            72.4         29.2    156.  32.4
## 4 range         17      155            98           92      481   48.9
## # i 3 more variables: DiabetesPedigreeFunction <dbl>, Age <dbl>, Outcome <dbl>
```

```r
# Histograms for numerical variables
dat %>%
  select(-"Outcome") %>%
  gather(key = "feature", value = "value") %>%
  ggplot(aes(value)) +
  geom_histogram(bins = 35, fill = "lightpink", colour = "purple") +
  facet_wrap(~feature, scales = "free") + # Creates separate plots for each feature with free axis scal
  theme_minimal() +
  labs(title = "Features's Distributions after Replace Nonsense Zeros", x = NULL)
```




```r
# Tall bars due to replacement of nonsense zeros

# Data prepare for creating multiple boxpots
x <- dat %>% select(-"Outcome")
y <- dat %>%
  pull(Outcome) %>% # extract the 'Outcome' column as a vector
  as.factor()

# Reshape data to long format and create boxplots
data.frame(y, x) %>%
  gather(key = "feature", value = "value", -y) %>%
  ggplot(aes(feature, value, fill = y)) +
  geom_boxplot() +
  facet_wrap(~feature, scales = "free") +
  scale_fill_manual(values = c("0" = "lightpink", "1" = "purple")) + # Custom colors
  theme_minimal() +
  labs(title = "Boxplots", x = NULL)
```




```
# Median values/distribution of each variable are generally higher in patients with diabetes


# B.4.2. Correlation analysis

# Test the correlation between two continuous variables: SkinThickness and Insulin
cor.test(dat$SkinThickness, dat$Insulin)
```

```
##
##  Pearson's product-moment correlation
##
## data:  dat$SkinThickness and dat$Insulin
## t = 3.981, df = 757, p-value = 0.0000751
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  0.0727958 0.2122164
## sample estimates:
##      cor
## 0.143217
```

```
#  p-value = 0.0000751 indicates that the correlation is statistically significant at the 0.05 level.

# Test the correlation between two continuous variables: Age and BMI
cor.test(dat$Age, dat$BMI)
```

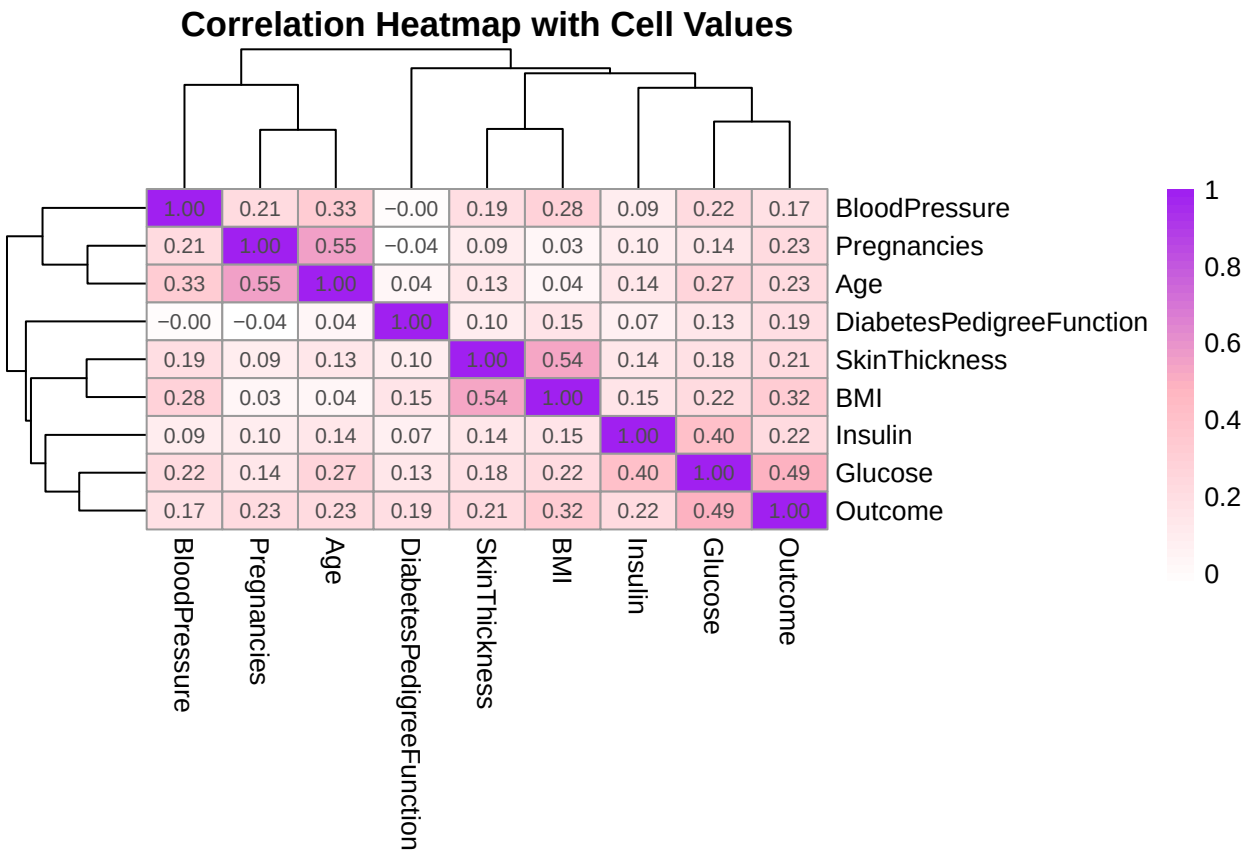
```
##
```

```
##  Pearson's product-moment correlation
##
## data:  dat$Age and dat$BMI
## t = 0.9676, df = 757, p-value = 0.334
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  -0.036107  0.106043
## sample estimates:
##      cor
## 0.035146
```

```
# p-value = 0.334 indicates that the correlation is not statistically significant at the 0.05 level.

# perform t-test on Glucose
t.test(Glucose ~ as.factor(Outcome), dat)
```

```
##
##  Welch Two Sample t-test
##
## data:  Glucose by as.factor(Outcome)
## t = -14.75, df = 452.3, p-value <2e-16
## alternative hypothesis: true difference in means between group 0 and group 1 is not equal to 0
## 95 percent confidence interval:
##  -35.3544 -27.0408
## sample estimates:
## mean in group 0 mean in group 1
##         110.338         141.536
```

```
# p-value < 2e-16. There is a significant difference between the mean_glucose of two groups '0' and '1'

# Calculate the correlation matrix
cor_matrix <- round(cor(dat),2)

# Create the heatmap with cell values
pheatmap(cor_matrix,
         main = "Correlation Heatmap with Cell Values",
         color = colorRampPalette(c("white", "lightpink", "purple"))(50),
         display_numbers = TRUE,  # Show numbers in cells
         number_format = "%.2f",  # Format of the numbers
         fontsize_number = 8 # Size of the numbers
)
```




```
# No strong correlations between variables
# Few moderate correlations between variables
# Glucose level has most effective to the outcome
# BloodPressure has least effective to the outcome
```

B.5. Predictive Modeling

```
# Scale date to ensure that data is uniformly prepared for analysis and modeling
x_scaled <- dat %>%
  select(-"Outcome") %>%
  scale()

# Extract the 'Outcome' column as a vector
y <- dat %>%
  pull(Outcome) %>%
  as.factor()

# Seed for reproducibility of data generation
set.seed(1, sample.kind = "Rounding")

# Split data into training and testing sets:

# Ensure that our training and testing sets are representative of the entire dataset
test_index <- createDataPartition(y, times = 1, p = 0.2, list = FALSE)
```

```r
# Split the dataset into training and testing sets
test_x <- x_scaled[test_index,]
test_y <- y[test_index]

train_x <- x_scaled[-test_index,]
train_y <- y[-test_index]

# Train control used for all methods. Specifies 10-fold cross-validation for model evaluation.
train_control <- trainControl(method = "cv", number = 10)


#----------------------------------------
# B.5.1. Generalized Linear Model (GLM)
#----------------------------------------

# Train the GLM model
train_glm <- train(train_x, train_y,
                   method = "glm",
                   family = binomial,
                   trControl = train_control
                   )

# Predict on the testing set
glm_preds <- predict(train_glm, test_x)

# Create a tibble to store model's accuracy, accuracy
accuracy <- tibble(Model = "Generalized Linear Model (GLM)",
                   Accuracy = round(mean(glm_preds == test_y), 5) # Compare predictions with the actual
                   )

# Print accuracy tibble
accuracy %>% kable()
```

Model	Accuracy
Generalized Linear Model (GLM)	0.73203

```r
#-----------------------------------------
# B.5.2. Linear Discriminant Analysis (LDA)
#-----------------------------------------

# Train the LDA model
train_lda <- train(train_x, train_y,
                   method = "lda",
                   trControl = train_control
                   )

# Predict on the testing set
lda_preds <- predict(train_lda, test_x)

# Calculate the accuracy then add on to the accuracy tibble
accuracy <- rbind(accuracy,
```

```
                  tibble(Model = "Linear Discriminant Analysis (LDA)",
                         Accuracy = round(mean(lda_preds == test_y), 5)
                         )
                  )

# Print accuracy tibble
accuracy %>% kable()
```

Model	Accuracy
Generalized Linear Model (GLM)	0.73203
Linear Discriminant Analysis (LDA)	0.72549

```
#-----------------------------------------------
# B.5.3. Quadratic Discriminant Analysis (QDA)
#-----------------------------------------------

# Train the QDA model
train_qda <- train(train_x, train_y,
                   method = "qda",
                   trControl = train_control
                   )

# Predict on the testing set
qda_preds <- predict(train_qda, test_x)

# Calculate the accuracy then add on to the accuracy tibble
accuracy <- rbind(accuracy,
                  tibble(Model = "Quadratic Discriminant Analysis (QDA)",
                         Accuracy = round(mean(qda_preds == test_y), 5)
                         )
                  )

# Print accuracy tibble
accuracy %>% kable()
```

Model	Accuracy
Generalized Linear Model (GLM)	0.73203
Linear Discriminant Analysis (LDA)	0.72549
Quadratic Discriminant Analysis (QDA)	0.67320

```
#-------------------------------------------------------
# B.5.4. Locally Estimated Scatterplot Smoothing (LOESS)
#-------------------------------------------------------

# Train the LOESS model.
train_loess <- train(train_x, train_y,
                     method = "gamLoess",
                     trControl = train_control
                     )
```

```
## Loading required package: gam

## Loading required package: splines

## Loading required package: foreach

##
## Attaching package: 'foreach'

## The following objects are masked from 'package:purrr':
##
##     accumulate, when

## Loaded gam 1.22-4
```

```
# Predict on the testing set
loess_preds <- predict(train_loess, test_x)

# Calculate the accuracy then add on to the accuracy tibble
accuracy <- rbind(accuracy,
                  tibble(Model = "Locally weighted scatterplot smoothing (LOESS)",
                         Accuracy = round(mean(loess_preds == test_y), 5)
                         )
                  )

# Print accuracy tibble
accuracy %>% kable()
```

Model	Accuracy
Generalized Linear Model (GLM)	0.73203
Linear Discriminant Analysis (LDA)	0.72549
Quadratic Discriminant Analysis (QDA)	0.67320
Locally weighted scatterplot smoothing (LOESS)	0.78431

```
#-----------------------------------
# B.5.5. K-Nearest Neighbors (KNN)
#-----------------------------------

# Define the tuning grid for specifying a range of hyperparameters or 'neighbors' to tune
tuning <- data.frame(k = seq(3, 21, 2))

# Train the KNN model
train_knn <- train(train_x, train_y,
                   method = "knn",
                   tuneGrid = tuning,
                   trControl = train_control
                   )

# Predict on the testing set
knn_preds <- predict(train_knn, test_x)
```

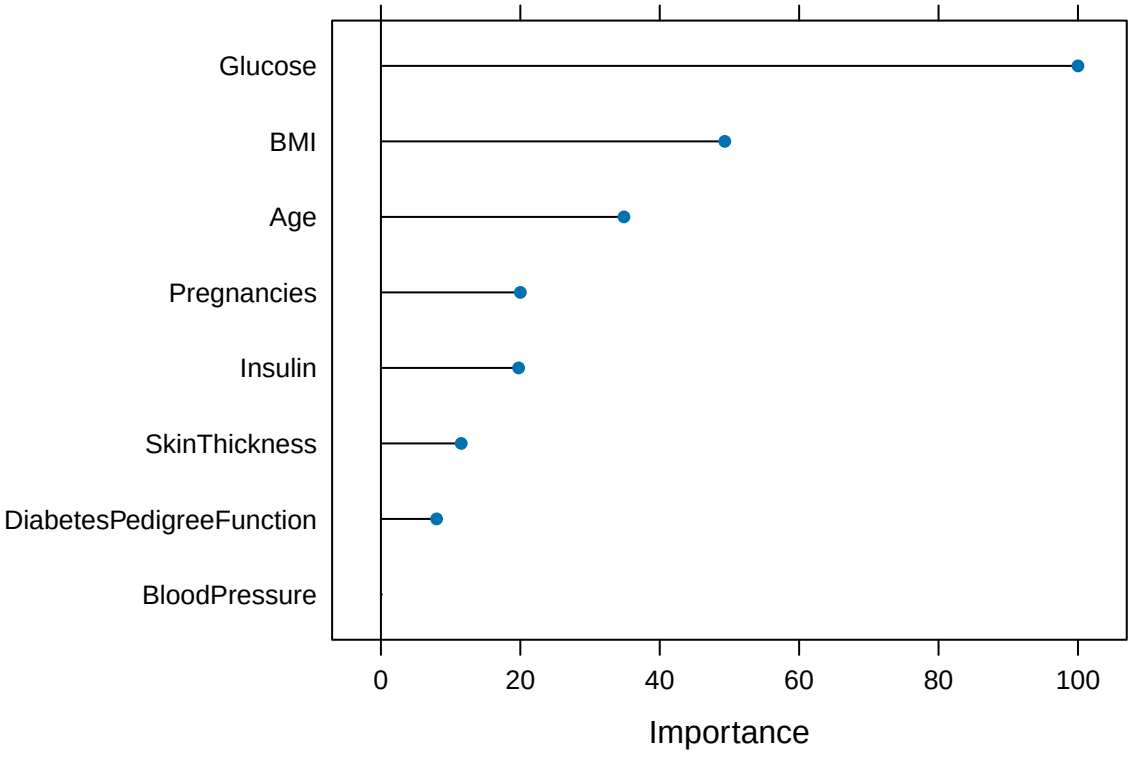
```r
# Calculate the accuracy then add on to the accuracy tibble
accuracy <- rbind(accuracy,
                  tibble(Model = "K-Nearest Neighbors (KNN)",
                         Accuracy = round(mean(knn_preds == test_y), 5)
                         )
                  )

# Print accuracy tibble
accuracy %>% kable()
```

Model	Accuracy
Generalized Linear Model (GLM)	0.73203
Linear Discriminant Analysis (LDA)	0.72549
Quadratic Discriminant Analysis (QDA)	0.67320
Locally weighted scatterplot smoothing (LOESS)	0.78431
K-Nearest Neighbors (KNN)	0.77778

```r
#------------------------------
# B.5.6. Random Forests (RF)
#------------------------------

set.seed(1, sample.kind = "Rounding")

# Create a tuning data frame with different values for mtry.
# mtry is the number of features to consider for splitting at each node in a decision tree within the R
tuning <- data.frame(mtry = c(3, 5, 7, 9))

# Train the RF model.
train_rf <- train(train_x, train_y,
                  method = "rf",
                  tuneGrid = tuning,
                  trControl = train_control,
                  importance = TRUE
                  )

# Predict on the testing set
rf_preds <- predict(train_rf, test_x)

# Calculate the accuracy then add on to the accuracy tibble
accuracy <- rbind(accuracy,
                  tibble(Model = "Random Forests (RF)",
                         Accuracy = round(mean(rf_preds == test_y), 5)
                         )
                  )

# Print accuracy tibble
accuracy %>% kable()
```

Model	Accuracy
Generalized Linear Model (GLM)	0.73203
Linear Discriminant Analysis (LDA)	0.72549
Quadratic Discriminant Analysis (QDA)	0.67320
Locally weighted scatterplot smoothing (LOESS)	0.78431
K-Nearest Neighbors (KNN)	0.77778
Random Forests (RF)	0.75817

```
# Compute variable importance
var_imp <- varImp(train_rf)

# Visualize variable importance
plot(var_imp)
```




```
# Glucose, BMI features are most influential in our model.
# Blood pressure has importance score of zero, suggesting it has zero influence in predicting Diabetes


#--------------------------
# B.5.7. Naive Bayes (NB)
#--------------------------

# Train the NB model
train_nb <- train(train_x, y = train_y,
```

```
                  method = "naive_bayes",
                  trControl = train_control
                  )

# Predict on the testing set
nb_preds <- predict(train_nb, test_x)

# Calculate the accuracy then add on to the accuracy tibble
accuracy <- rbind(accuracy,
                  tibble(Model = "Naive Bayes (NB)",
                         Accuracy = round(mean(nb_preds == test_y), 5)
                         )
                  )

# Print accuracy tibble
accuracy %>% kable()
```

Model	Accuracy
Generalized Linear Model (GLM)	0.73203
Linear Discriminant Analysis (LDA)	0.72549
Quadratic Discriminant Analysis (QDA)	0.67320
Locally weighted scatterplot smoothing (LOESS)	0.78431
K-Nearest Neighbors (KNN)	0.77778
Random Forests (RF)	0.75817
Naive Bayes (NB)	0.72549

```
#------------------------
# B.5.8. Decision Trees
#------------------------

set.seed(1, sample.kind = "Rounding")

# Train the decision tree model
train_dt <- train(x = train_x, y = train_y,
                  method = "rpart",
                  trControl = train_control
                  )

# Predict on the testing set
dt_preds <- predict(train_dt, test_x)

# Calculate the accuracy then add on to the accuracy tibble
accuracy <- rbind(accuracy,
                  tibble(Model = "Decision Trees (DT)",
                         Accuracy = round(mean(dt_preds == test_y), 5)
                         )
                  )

# Print accuracy tibble
accuracy %>% kable()
```

Model	Accuracy
Generalized Linear Model (GLM)	0.73203
Linear Discriminant Analysis (LDA)	0.72549
Quadratic Discriminant Analysis (QDA)	0.67320
Locally weighted scatterplot smoothing (LOESS)	0.78431
K-Nearest Neighbors (KNN)	0.77778
Random Forests (RF)	0.75817
Naive Bayes (NB)	0.72549
Decision Trees (DT)	0.73203

```
#--------------------------------------
# B.5.9. Support Vector Machines (SVM)
#--------------------------------------
# Method learned from DataCamp tutorial.

set.seed(1, sample.kind = "Rounding")

# Train an SVM model with Radial Basis Function (RBF) Kernel
train_svm <- train(x = train_x, y = train_y,
                   method = "svmRadial",
                   trControl = train_control
                   )

# Predict on the testing set
svm_preds <- predict(train_svm, test_x)

# Calculate the accuracy then add on to the accuracy tibble
accuracy <- rbind(accuracy,
                  tibble(Model = "Support Vector Machines (SVM)",
                         Accuracy = round(mean(svm_preds == test_y), 5)
                         )
                  )

# Print accuracy tibble
accuracy %>% kable()
```

Model	Accuracy
Generalized Linear Model (GLM)	0.73203
Linear Discriminant Analysis (LDA)	0.72549
Quadratic Discriminant Analysis (QDA)	0.67320
Locally weighted scatterplot smoothing (LOESS)	0.78431
K-Nearest Neighbors (KNN)	0.77778
Random Forests (RF)	0.75817
Naive Bayes (NB)	0.72549
Decision Trees (DT)	0.73203
Support Vector Machines (SVM)	0.75817

```
#-------------------------------
# B.5.10. Creating an ensemble
#-------------------------------
```

```r
# Create an ensemble
ensemble <- cbind(glm = glm_preds == 0,
                  lda = lda_preds == 0,
                  qda = qda_preds == 0,
                  loess = loess_preds == 0,
                  rf = rf_preds == 0,
                  knn = knn_preds == 0,
                  nb = nb_preds == 0,
                  dt = dt_preds == 0,
                  svm = svm_preds == 0
                  )

# Predict on the testing set. If more models predict "0" then return "0"
ensemble_preds <- ifelse(rowMeans(ensemble) > 0.5, "0", "1")

# Calculate the accuracy then add on to the accuracy tibble
accuracy <- rbind(accuracy,
                  tibble(Model = "Ensemble",
                         Accuracy = round(mean(ensemble_preds == test_y), 5)
                         )
                  )

# Print accuracy tibble
accuracy %>% kable()
```

Model	Accuracy
Generalized Linear Model (GLM)	0.73203
Linear Discriminant Analysis (LDA)	0.72549
Quadratic Discriminant Analysis (QDA)	0.67320
Locally weighted scatterplot smoothing (LOESS)	0.78431
K-Nearest Neighbors (KNN)	0.77778
Random Forests (RF)	0.75817
Naive Bayes (NB)	0.72549
Decision Trees (DT)	0.73203
Support Vector Machines (SVM)	0.75817
Ensemble	0.76471

C. Result

```r
# Method with highest Accuracy
print(accuracy[which.max(accuracy$Accuracy),])
```

```
## # A tibble: 1 x 2
##   Model                                          Accuracy
##   <chr>                                             <dbl>
## 1 Locally weighted scatterplot smoothing (LOESS)    0.784
```

D. Conclusion:

D.1. Brief Summary of the Report:

In this project, we have addressed several critical aspects of the dataset to ensure its suitability for subsequent analysis and modeling. By focusing on the replacement of zero values with the mean of non-zero values in essential columns such as Blood Pressure, BMI, Glucose, Insulin, and Skin Thickness, we have enhanced the integrity of the data.

D.2. Potential Impact:

This step mitigates the impact of potentially erroneous or missing values, which could otherwise lead to misleading conclusions or skewed results.

D.3. Limitations:

The approach of replacing zero values with the mean of non-zero values assumes that zeros are missing or erroneous rather than actual observations. This may not be valid for all features. For instance, zeros in some columns could represent genuine values, and replacing them might distort the data distribution.

D.4. Future Work:

Addressing these potential sources of error involves a combination of improved procedures, better training, and robust systems for error detection and correction. By focusing on standardizing measurement protocols, understanding the impact of severe medical conditions, reducing data entry mistakes, and minimizing calculation errors, future work can significantly enhance the quality and reliability of the dataset. Implementing these strategies will contribute to more accurate analyses and better-informed decisions based on the data.

E. References

https://rafalab.dfci.harvard.edu/dsbook/
https://www.datacamp.com/tutorial/support-vector-machines-r
https://translate.google.com/?sl=auto&tl=en&op=translate to translate some of my text from my native language