Diabetes Prediction and Analysis Notebook

Contents

Welcome. While the numbers on this dataset hint at underlying diabetic trends, our goal is to derive actionable insights and even build a predictor if the data agrees.

Table of Contents

Data Loading and Initial Inspection

```
data_path <- 'diabetes_dataset.csv'

df <- read_csv(data_path, col_types = cols()) # readr::read_csv

# Display the first few rows of the dataset
cat('Dataset loaded. Here are the first few rows:\n')
```

```
## Dataset loaded. Here are the first few rows:
```

```
kable(head(df), caption = "First few rows of the dataset")
```

Table 1: First few rows of the dataset

... 1	Age	Sex	Ethnicity	BMI	Waist_Circumference	Fasting_Glucose	HbA1c	Blood_Pressure	Cholesterol	Triglycerides	HDL	LDL	Physical_Activity	Diet_Quality	Alcohol_Consumption	Smoking_Status	Family_History_of_Diabetes	Gestational_Diabetes
0	58	Female	White	35.8	83.4	123.9	10.9	152	114	197.8	50.2	99.2	37.5	7.2	Moderate	1538	Moderate	Never	0	1
1	48	Male	Asian	24.1	71.4	183.7	12.8	103	91	261.6	62.0	146.4	88.5	6.1	Moderate	2653	Moderate	Current	0	1
2	34	Female	Black	25.0	113.8	142.0	14.5	179	104	261.0	32.1	164.1	56.2	6.9	Low	1684	Heavy	Former	1	0
3	62	Male	Asian	32.7	100.4	167.4	8.8	176	118	183.4	41.1	84.0	34.4	5.4	Low	3796	Moderate	Never	1	0
4	27	Female	Asian	33.5	110.8	146.4	7.1	122	97	203.2	53.9	92.8	81.9	7.4	Moderate	3161	Heavy	Current	0	0
5	40	Female	Asian	33.6	96.1	75.0	13.5	170	90	152.3	44.5	190.0	77.5	6.4	Low	3460	None	Never	1	1

```r
# Display data types (structure)
cat('\nData types (structure) of each column:\n')
```

```
##
## Data types (structure) of each column:
```

```r
str(df) # str() gives a good overview including types
```

```
## spc_tbl_ [10,000 x 21] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
##  $ ...1                          : num [1:10000] 0 1 2 3 4 5 6 7 8 9 ...
##  $ Age                           : num [1:10000] 58 48 34 62 27 40 58 38 42 30 ...
##  $ Sex                           : chr [1:10000] "Female" "Male" "Female" "Male" ...
##  $ Ethnicity                     : chr [1:10000] "White" "Asian" "Black" "Asian" ...
##  $ BMI                           : num [1:10000] 35.8 24.1 25 32.7 33.5 33.6 33.2 26.9 27 24 ...
##  $ Waist_Circumference           : num [1:10000] 83.4 71.4 113.8 100.4 110.8 ...
##  $ Fasting_Blood_Glucose         : num [1:10000] 124 184 142 167 146 ...
##  $ HbA1c                         : num [1:10000] 10.9 12.8 14.5 8.8 7.1 13.5 13.3 10.9 7 14 ...
##  $ Blood_Pressure_Systolic       : num [1:10000] 152 103 179 176 122 170 131 121 132 146 ...
##  $ Blood_Pressure_Diastolic      : num [1:10000] 114 91 104 118 97 90 80 83 118 83 ...
##  $ Cholesterol_Total             : num [1:10000] 198 262 261 183 203 ...
##  $ Cholesterol_HDL               : num [1:10000] 50.2 62 32.1 41.1 53.9 44.5 77.9 69.7 73.2 53.3 ...
##  $ Cholesterol_LDL               : num [1:10000] 99.2 146.4 164.1 84 92.8 ...
##  $ GGT                           : num [1:10000] 37.5 88.5 56.2 34.4 81.9 77.5 52.1 72 76.4 14.5 ...
##  $ Serum_Urate                   : num [1:10000] 7.2 6.1 6.9 5.4 7.4 6.4 4.7 5.6 6.2 6.9 ...
##  $ Physical_Activity_Level       : chr [1:10000] "Moderate" "Moderate" "Low" "Low" ...
##  $ Dietary_Intake_Calories       : num [1:10000] 1538 2653 1684 3796 3161 ...
##  $ Alcohol_Consumption           : chr [1:10000] "Moderate" "Moderate" "Heavy" "Moderate" ...
##  $ Smoking_Status                : chr [1:10000] "Never" "Current" "Former" "Never" ...
##  $ Family_History_of_Diabetes    : num [1:10000] 0 0 1 1 0 1 0 0 1 1 ...
##  $ Previous_Gestational_Diabetes: num [1:10000] 1 1 0 0 0 1 0 1 0 0 ...
##  - attr(*, "spec")=
##   .. cols(
##   ..   ...1 = col_double(),
##   ..   Age = col_double(),
##   ..   Sex = col_character(),
##   ..   Ethnicity = col_character(),
##   ..   BMI = col_double(),
##   ..   Waist_Circumference = col_double(),
##   ..   Fasting_Blood_Glucose = col_double(),
##   ..   HbA1c = col_double(),
##   ..   Blood_Pressure_Systolic = col_double(),
##   ..   Blood_Pressure_Diastolic = col_double(),
##   ..   Cholesterol_Total = col_double(),
##   ..   Cholesterol_HDL = col_double(),
##   ..   Cholesterol_LDL = col_double(),
##   ..   GGT = col_double(),
##   ..   Serum_Urate = col_double(),
##   ..   Physical_Activity_Level = col_character(),
##   ..   Dietary_Intake_Calories = col_double(),
##   ..   Alcohol_Consumption = col_character(),
##   ..   Smoking_Status = col_character(),
##   ..   Family_History_of_Diabetes = col_double(),
```

```
##   ..   Previous_Gestational_Diabetes = col_double()
##   .. )
##  - attr(*, "problems")=<externalptr>
```

Data Cleaning and Preprocessing

In this section, we clean and preprocess the data. A common nuisance is the presence of an unnecessary index column. We also check for missing values and perform basic corrections.

```
if ("Unnamed: 0" %in% colnames(df)) {
  df <- df %>% select(-"Unnamed: 0")
  cat("Dropped 'Unnamed: 0' column.\n")
}
if ("...1" %in% colnames(df)) { # Another common name for unnamed index from read_csv
  df <- df %>% select(-"...1")
  cat("Dropped '...1' column.\n")
}
```

```
## Dropped '...1' column.
```

```
# Check for duplicates
num_duplicates <- sum(duplicated(df))
cat('Number of duplicate rows:', num_duplicates, '\n')
```

```
## Number of duplicate rows: 0
```

```
# Optionally, remove duplicates: df <- df %>% distinct()

# Check for missing values
cat('\nMissing values in each column:\n')
```

```
##
## Missing values in each column:
```

```
missing_summary <- colSums(is.na(df))
kable(as.data.frame(missing_summary), col.names = c("Missing Count"), caption = "Missing values per colu
```

Table 2: Missing values per column

	Missing Count
Age	0
Sex	0
Ethnicity	0
BMI	0
Waist_Circumference	0
Fasting_Blood_Glucose	0
HbA1c	0
Blood_Pressure_Systolic	0
Blood_Pressure_Diastolic	0

	Missing Count
Cholesterol_Total	0
Cholesterol_HDL	0
Cholesterol_LDL	0
GGT	0
Serum_Urate	0
Physical_Activity_Level	0
Dietary_Intake_Calories	0
Alcohol_Consumption	0
Smoking_Status	0
Family_History_of_Diabetes	0
Previous_Gestational_Diabetes	0

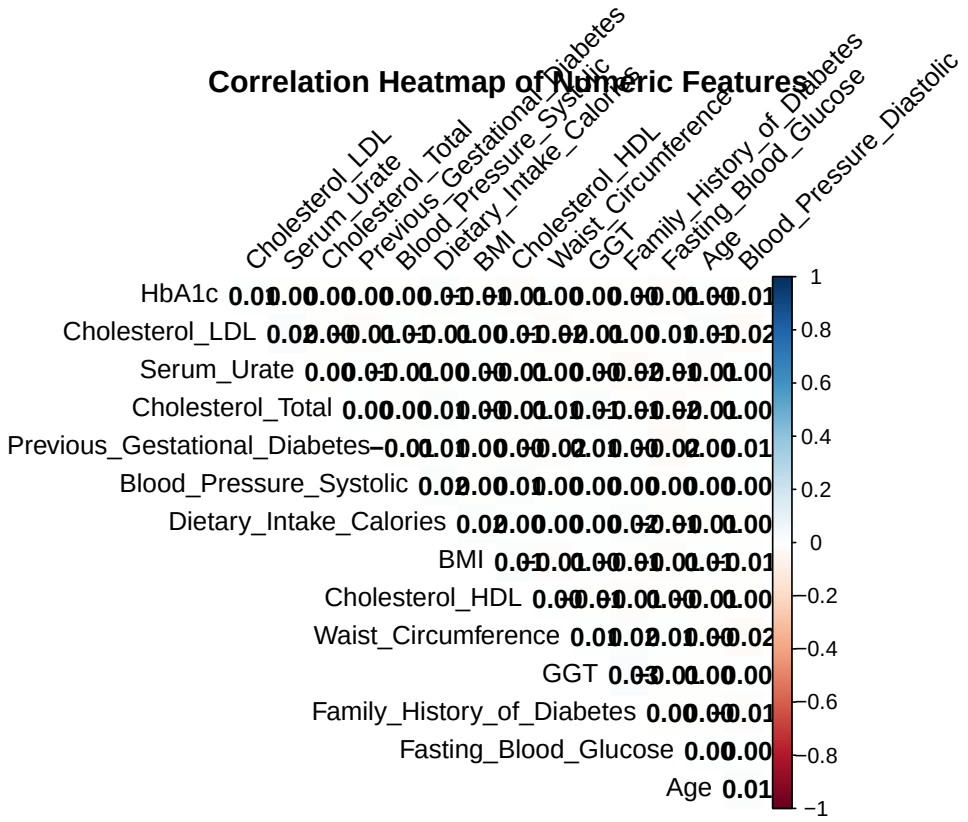
```r
# Drop rows with any missing values to keep things simple
df <- df %>% drop_na()
cat('\nData shape after cleaning (rows, columns):', paste(dim(df), collapse=", "), '\n')
```

```
##
## Data shape after cleaning (rows, columns): 10000, 20
```

Exploratory Data Analysis

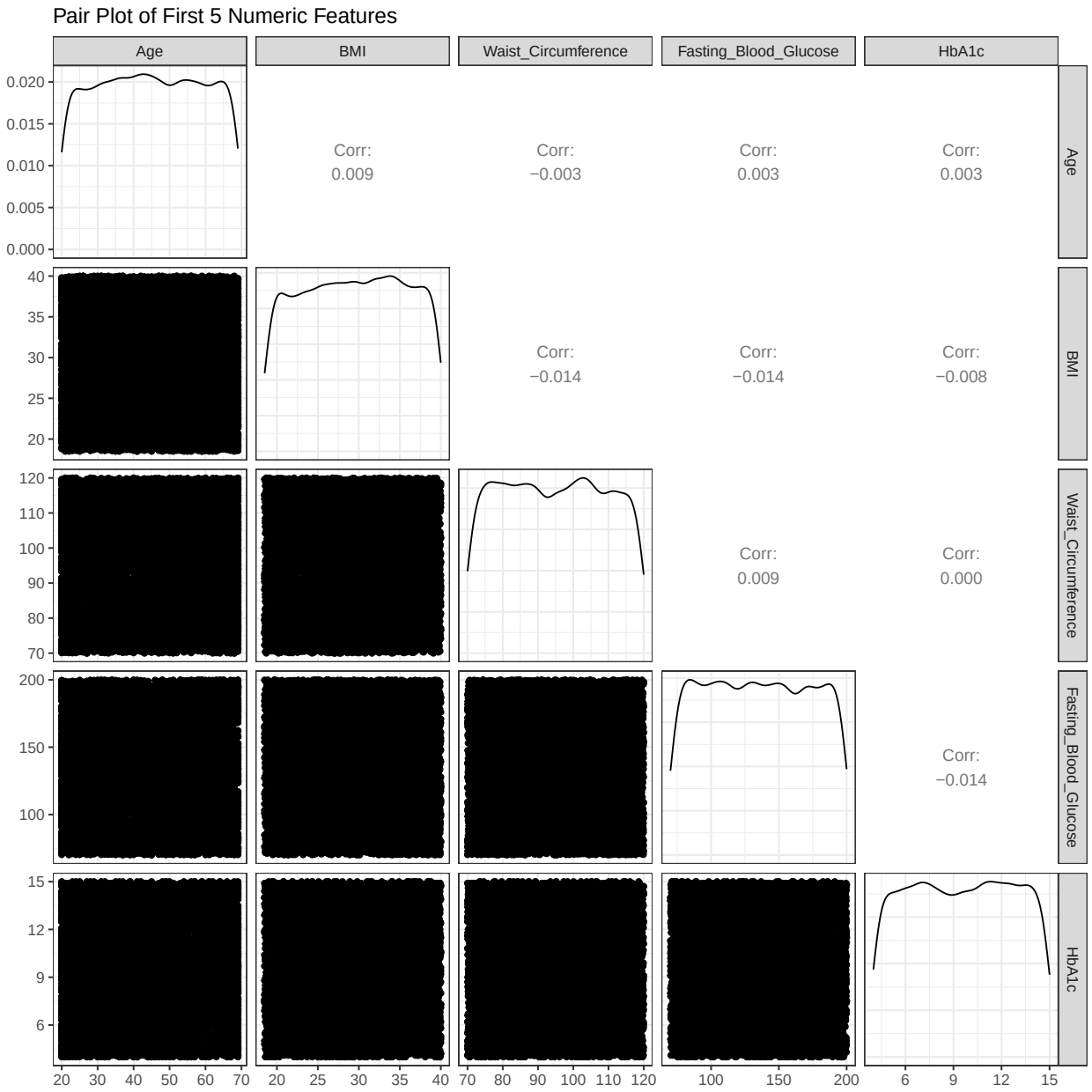
Now we begin the numerical deep dive. We create a correlation heatmap (only if sufficient numeric columns exist) and other essential visualizations to understand distribution, relationships, and trends within the data. Our goal is to understand underlying patterns that might inform the predictor later on.

```r
# Defensive check for empty column names in df
if (any(colnames(df) == "")) {
  cat("Warning: Found column(s) with empty names in 'df'. Removing them to prevent errors.\n")
  df <- df[, colnames(df) != "", drop = FALSE]
}

# Extract numeric columns for correlation analysis
numeric_df <- df %>% select_if(is.numeric)

if (ncol(numeric_df) >= 2) {
  corr_matrix <- cor(numeric_df, use = "pairwise.complete.obs")

  # Using corrplot for a nice heatmap
  corrplot(corr_matrix,
           method = "color",
           type = "upper", # Show upper triangle
           order = "hclust", # Order by hierarchical clustering
           addCoef.col = "black", # Add correlation coefficients
           tl.col = "black", tl.srt = 45, # Text label color and rotation
           diag = FALSE, # Don't display diagonal
           # title = "Correlation Heatmap of Numeric Features", # Title within corrplot
           mar = c(0,0,1,0)) # Adjust margins if title is used
  title("Correlation Heatmap of Numeric Features", line = -1) # Add title using base graphics way
} else {
  cat('Not enough numeric columns for a correlation heatmap (need at least 2).\n')
}
```



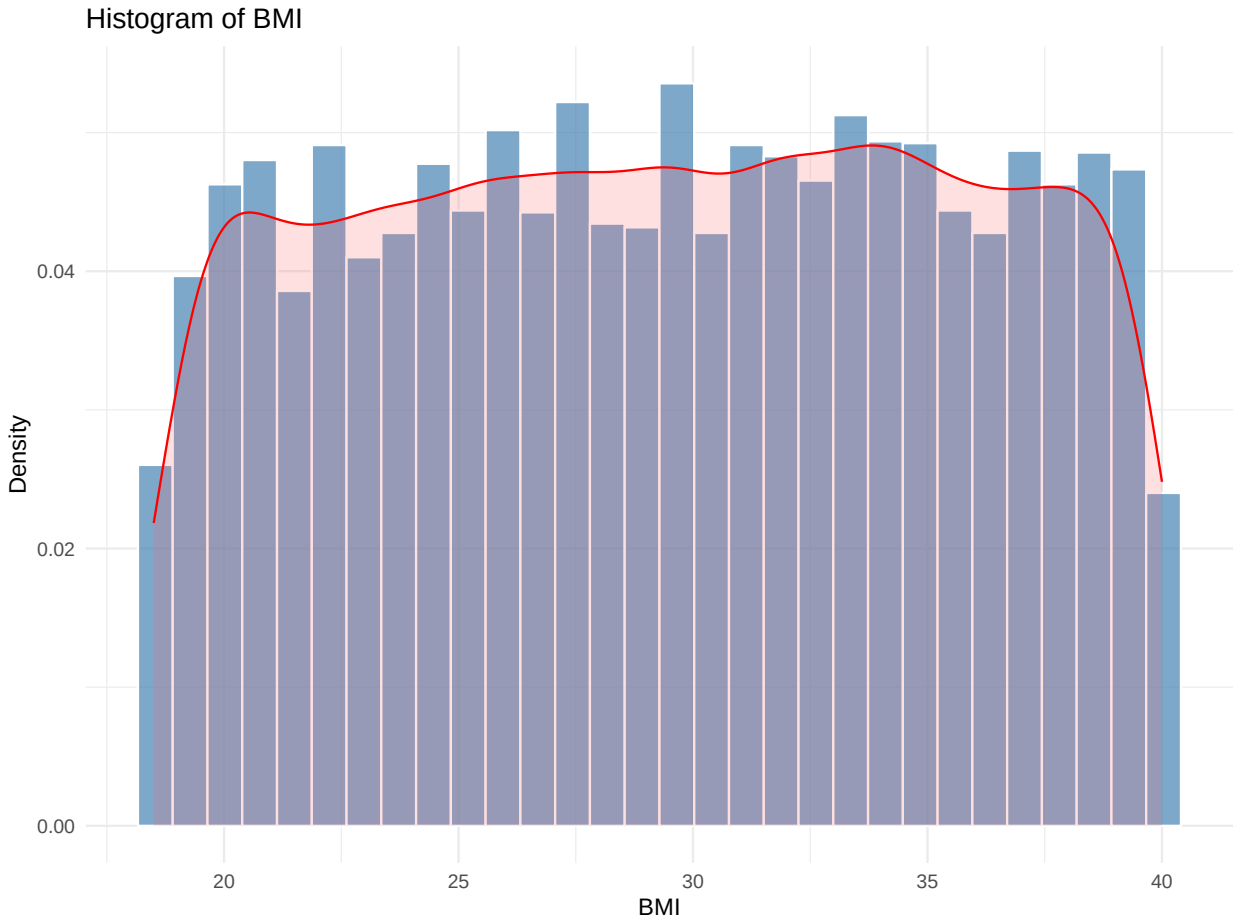

```
if (ncol(numeric_df) > 0) {
  if (ncol(numeric_df) <= 10) { # Limit to 10 columns for performance
    ggpairs_plot <- ggpairs(numeric_df, title = "Pair Plot of Numeric Features")
    print(ggpairs_plot)
  } else {
    cat("Too many numeric columns for a full pair plot. Plotting first 5.\n")
    if(ncol(numeric_df) >= 5) {
       print(ggpairs(numeric_df[,1:5], title = "Pair Plot of First 5 Numeric Features"))
    } else {
       print(ggpairs(numeric_df, title = "Pair Plot of Numeric Features")) # Plot all if <5
    }
  }
} else {
  cat("No numeric columns to create a pair plot.\n")
}
```

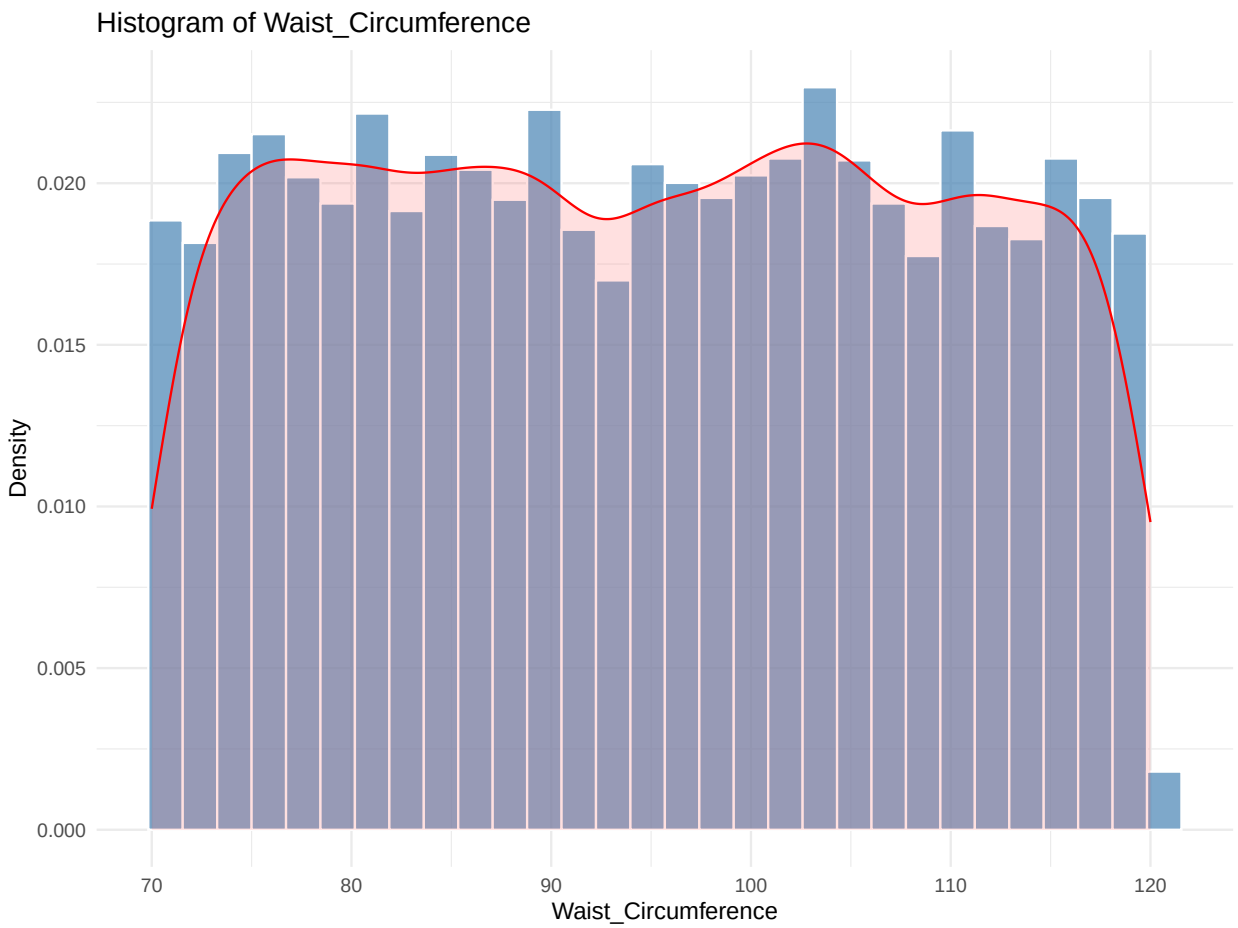
```
## Too many numeric columns for a full pair plot. Plotting first 5.
```

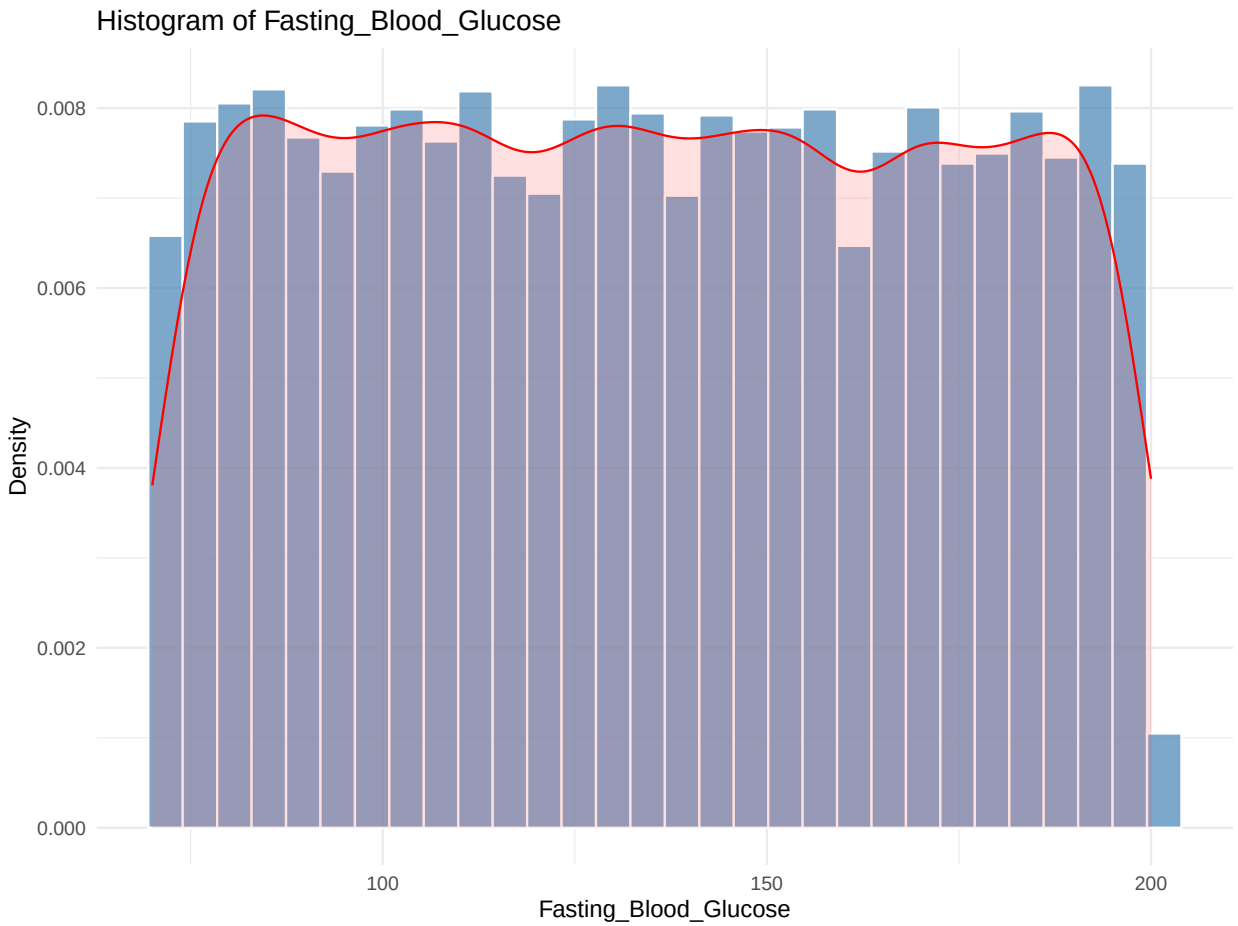


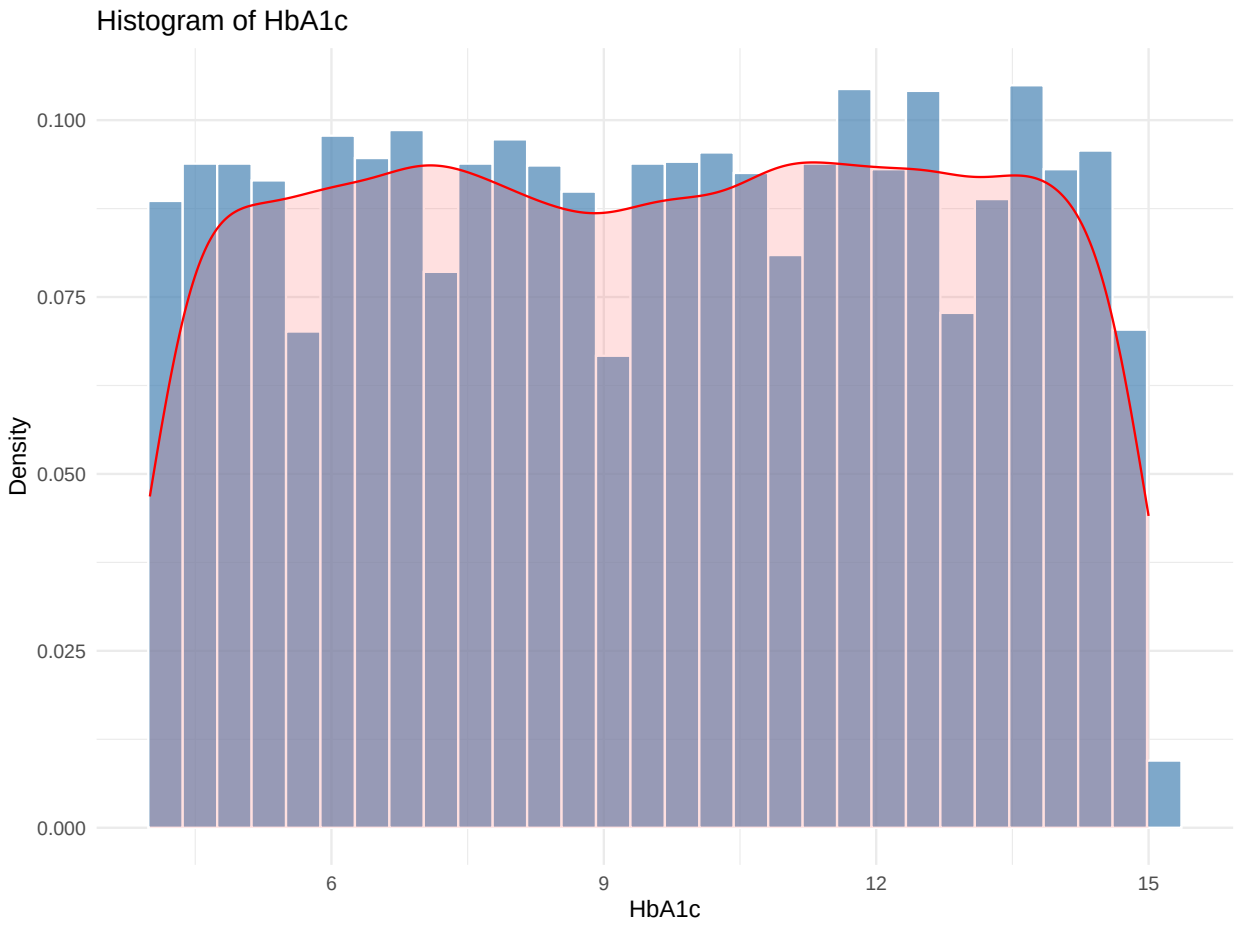

```r
# Plotting histograms for a few key numeric features
features_to_plot <- c('BMI', 'Waist_Circumference', 'Fasting_Blood_Glucose', 'HbA1c')

for (feature in features_to_plot) {
  if (feature %in% colnames(df) && is.numeric(df[[feature]])) {
    p <- ggplot(df, aes_string(x = feature)) +
      geom_histogram(aes(y = ..density..), bins = 30, fill = "steelblue", color = "white", alpha = 0.7) +
      geom_density(alpha = .2, fill = "#FF6666", color="red") +
      labs(title = paste('Histogram of', feature), x = feature, y = "Density") +
      theme_minimal()
    print(p)
  } else {
    cat(paste0(feature, ' column not found or not numeric in the dataset.\n'))
  }
```

```
}
```


Histogram of BMI
Density
0.04
0.02
0.00
20
25
30
35
40
BMI


Histogram of Waist_Circumference
Density
0.020
0.015
0.010
0.005
0.000
70
80
90
100
110
120
Waist_Circumference


Histogram of Fasting_Blood_Glucose
Density
0.008
0.006
0.004
0.002
0.000
100
150
200
Fasting_Blood_Glucose



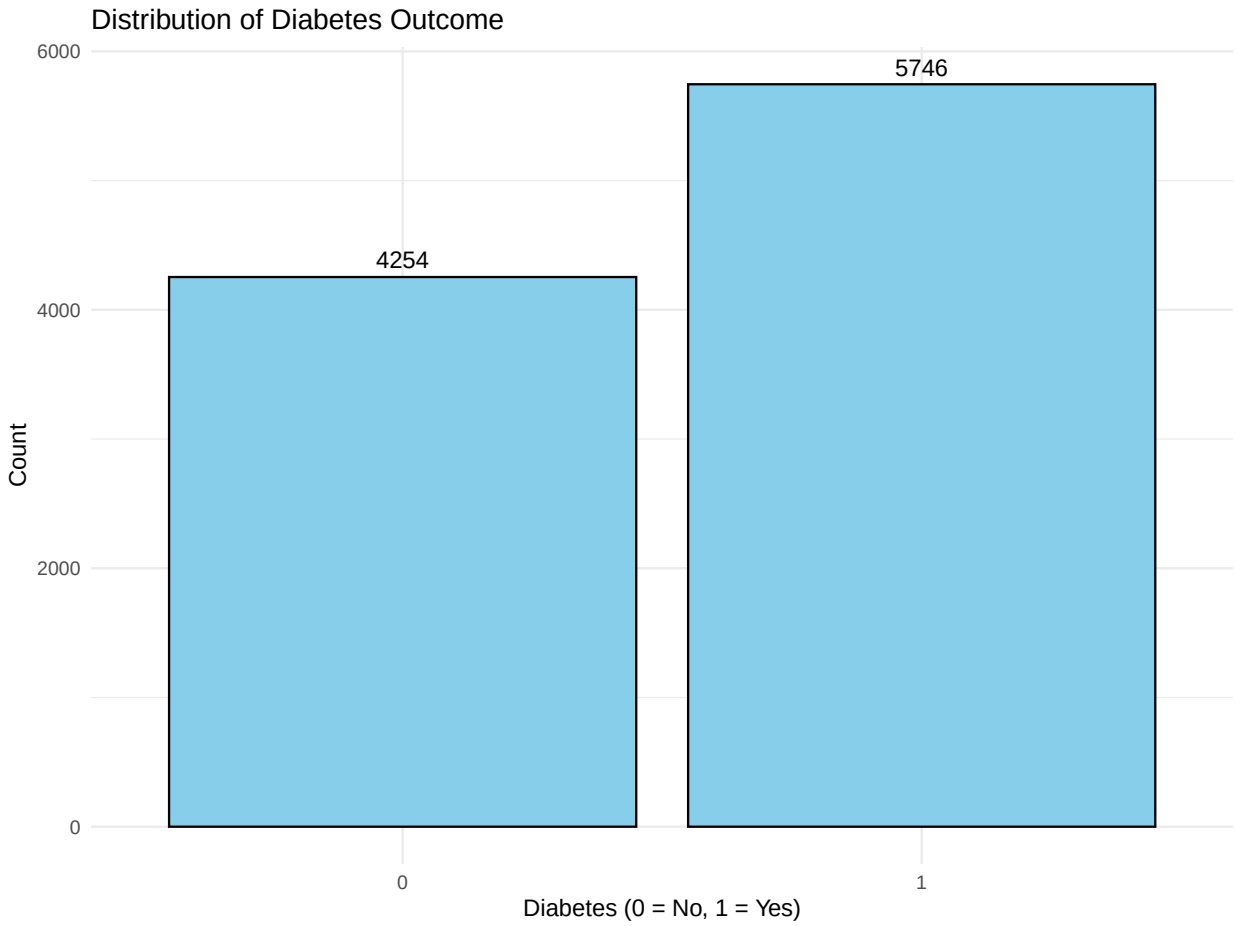
Feature Engineering and Predictor Creation

The dataset does not provide an explicit target column for diabetes. However, clinical guidelines often use Fasting Blood Glucose as an important indicator. In this notebook, we create a binary target column called 'Diabetes' which is set to 1 if a patient's Fasting Blood Glucose exceeds 125 (a commonly used threshold), and 0 otherwise. This derived target will let us build a logistic regression predictor.

```
# Create a new binary target column 'Diabetes'
if ('Fasting_Blood_Glucose' %in% colnames(df)) {
  df <- df %>%
    mutate(Diabetes = ifelse(Fasting_Blood_Glucose > 125, 1, 0))
  cat('Target column Diabetes created based on Fasting_Blood_Glucose.\n')

  # For plotting with ggplot, it's often better as a factor.
  # For glm, 0/1 numeric is fine, or a factor with levels "0" and "1".
  df$Diabetes_factor <- as.factor(df$Diabetes)

  # Examine the distribution of the new target
  p_dist <- ggplot(df, aes(x = Diabetes_factor)) +
    geom_bar(fill = "skyblue", color = "black") +
    geom_text(stat='count', aes(label=..count..), vjust=-0.5) +
    labs(title = 'Distribution of Diabetes Outcome', x = 'Diabetes (0 = No, 1 = Yes)', y = 'Count') +
    theme_minimal()
```

```
  print(p_dist)

} else {
  cat('Fasting_Blood_Glucose column not found. Cannot create Diabetes target column.\n')
  # To allow the script to proceed, we might create a dummy column, but this is not ideal for actual an
  # df$Diabetes <- 0 # Or stop("Fasting_Blood_Glucose column required.")
}
```

```
## Target column Diabetes created based on Fasting_Blood_Glucose.
```




```
# Select predictor columns
# Ensure 'Diabetes' column was created
if (!"Diabetes" %in% colnames(df)) {
  stop("Target column 'Diabetes' not found. Please ensure it was created in the previous step.")
}

potential_feature_cols <- c('Age', 'BMI', 'Waist_Circumference', 'Fasting_Blood_Glucose', 'HbA1c',
                            'Blood_Pressure_Systolic', 'Blood_Pressure_Diastolic',
                            'Cholesterol_Total', 'Cholesterol_HDL', 'Cholesterol_LDL',
                            'GGT', 'Serum_Urate', 'Dietary_Intake_Calories',
                            'Family_History_of_Diabetes', 'Previous_Gestational_Diabetes')
```

```r
# Filter features that exist in the dataframe and are numeric
feature_cols <- intersect(potential_feature_cols, colnames(df))
feature_cols <- setdiff(feature_cols, "Diabetes") # Exclude target if accidentally included


if (length(feature_cols) == 0) {
    stop("No feature columns selected or found in the dataframe. Check column names and availability.")
}
cat("Selected feature columns for model:\n")
```

```
## Selected feature columns for model:
```

```r
print(feature_cols)
```

```
##  [1] "Age"                          "BMI"
##  [3] "Waist_Circumference"          "Fasting_Blood_Glucose"
##  [5] "HbA1c"                        "Blood_Pressure_Systolic"
##  [7] "Blood_Pressure_Diastolic"     "Cholesterol_Total"
##  [9] "Cholesterol_HDL"              "Cholesterol_LDL"
## [11] "GGT"                          "Serum_Urate"
## [13] "Dietary_Intake_Calories"      "Family_History_of_Diabetes"
## [15] "Previous_Gestational_Diabetes"
```

```r
# Prepare data for splitting (combine features and target for caTools)
# Ensure Diabetes is numeric 0/1 for glm, or factor for other methods
data_for_split <- df[, c(feature_cols, "Diabetes")] # Diabetes is 0/1 numeric here

# Split the data into training and test sets
set.seed(42) # for reproducibility
# sample.split needs the target variable for stratified sampling
split <- sample.split(data_for_split$Diabetes, SplitRatio = 0.8) # 80% for training

train_data <- subset(data_for_split, split == TRUE)
test_data <- subset(data_for_split, split == FALSE)

cat('\nTraining and test sets created with proportions:\n')
```

```
##
## Training and test sets created with proportions:
```

```r
cat('Train data (rows, columns):', paste(dim(train_data), collapse=", "), '\n')
```

```
## Train data (rows, columns): 8000, 16
```

```r
cat('Test data (rows, columns):', paste(dim(test_data), collapse=", "), '\n')
```

```
## Test data (rows, columns): 2000, 16
```

Model Building and Evaluation

We now build a logistic regression model to predict diabetes and assess its performance with accuracy, a confusion matrix, and an ROC curve. In addition, we compute permutation importance to understand the influence of our features.

```
# Build the logistic regression model
# Create the formula string: Diabetes ~ Age + BMI + ...
# Ensure feature names are valid R names (e.g., no spaces, special chars - backticks can handle them)
# feature_cols_safe <- make.names(feature_cols) # If names had issues
# colnames(train_data) <- make.names(colnames(train_data))
# colnames(test_data) <- make.names(colnames(test_data))
# formula_str <- paste("Diabetes ~", paste(feature_cols_safe, collapse = " + "))

formula_str <- paste("Diabetes ~", paste(feature_cols, collapse = " + "))
model_formula <- as.formula(formula_str)

# Fit the model. glm handles 0/1 numeric response for binomial family.
# It can also handle a factor response.
# train_data$Diabetes <- as.factor(train_data$Diabetes) # If preferred
logreg_model <- glm(model_formula, data = train_data, family = binomial(link = "logit"))

# Summary of the model (optional, but good to see)
# kable(summary(logreg_model)$coefficients, caption="Model Coefficients")

# Make predictions (probabilities on the test set)
y_prob_pred <- predict(logreg_model, newdata = test_data, type = "response")

# Convert probabilities to class predictions (0 or 1) using a 0.5 threshold
y_class_pred <- ifelse(y_prob_pred > 0.5, 1, 0)

# Actual labels from the test set
actual_test_labels <- test_data$Diabetes # Should be 0/1 numeric

# Compute accuracy
accuracy <- mean(y_class_pred == actual_test_labels)
cat(sprintf('Accuracy of Logistic Regression model: %.2f\n', accuracy))
```

```
## Accuracy of Logistic Regression model: 1.00
```

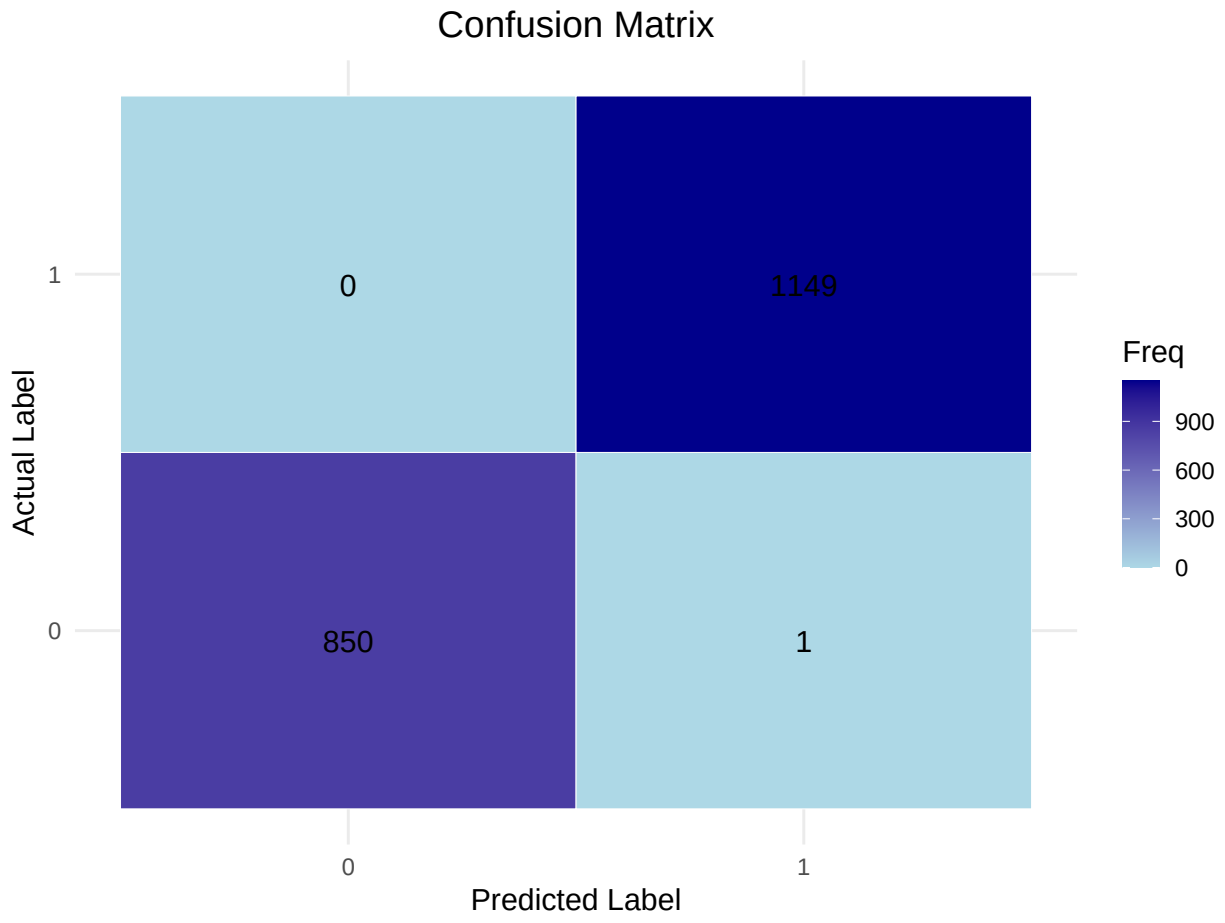
```
# Confusion Matrix
# Ensure actual_test_labels and y_class_pred are factors for table() to get names right for plotting
# Or ensure they are 0/1 numeric and handle factor conversion for plotting.
conf_matrix_table <- table(Actual = factor(actual_test_labels, levels=c(0,1)),
                           Predicted = factor(y_class_pred, levels=c(0,1)))
cat("\nConfusion Matrix:\n")
```

```
##
## Confusion Matrix:
```

```
print(conf_matrix_table)
```

```
##        Predicted
## Actual    0    1
##      0  850    1
##      1    0 1149
```

```
# For a heatmap plot of the confusion matrix
conf_matrix_df <- as.data.frame(conf_matrix_table)
colnames(conf_matrix_df) <- c("Actual", "Predicted", "Freq")

p_cm <- ggplot(data = conf_matrix_df, aes(x = Predicted, y = Actual, fill = Freq)) +
  geom_tile(color = "white") + # Add white lines between tiles
  geom_text(aes(label = Freq), vjust = 1, color = "black", size = 5) + # Make text black for visibility
  scale_fill_gradient(low = "lightblue", high = "darkblue") +
  labs(title = "Confusion Matrix", x = "Predicted Label", y = "Actual Label") +
  theme_minimal(base_size = 14) +
  theme(axis.text.x = element_text(angle = 0, hjust = 0.5),
        plot.title = element_text(hjust = 0.5))
print(p_cm)
```



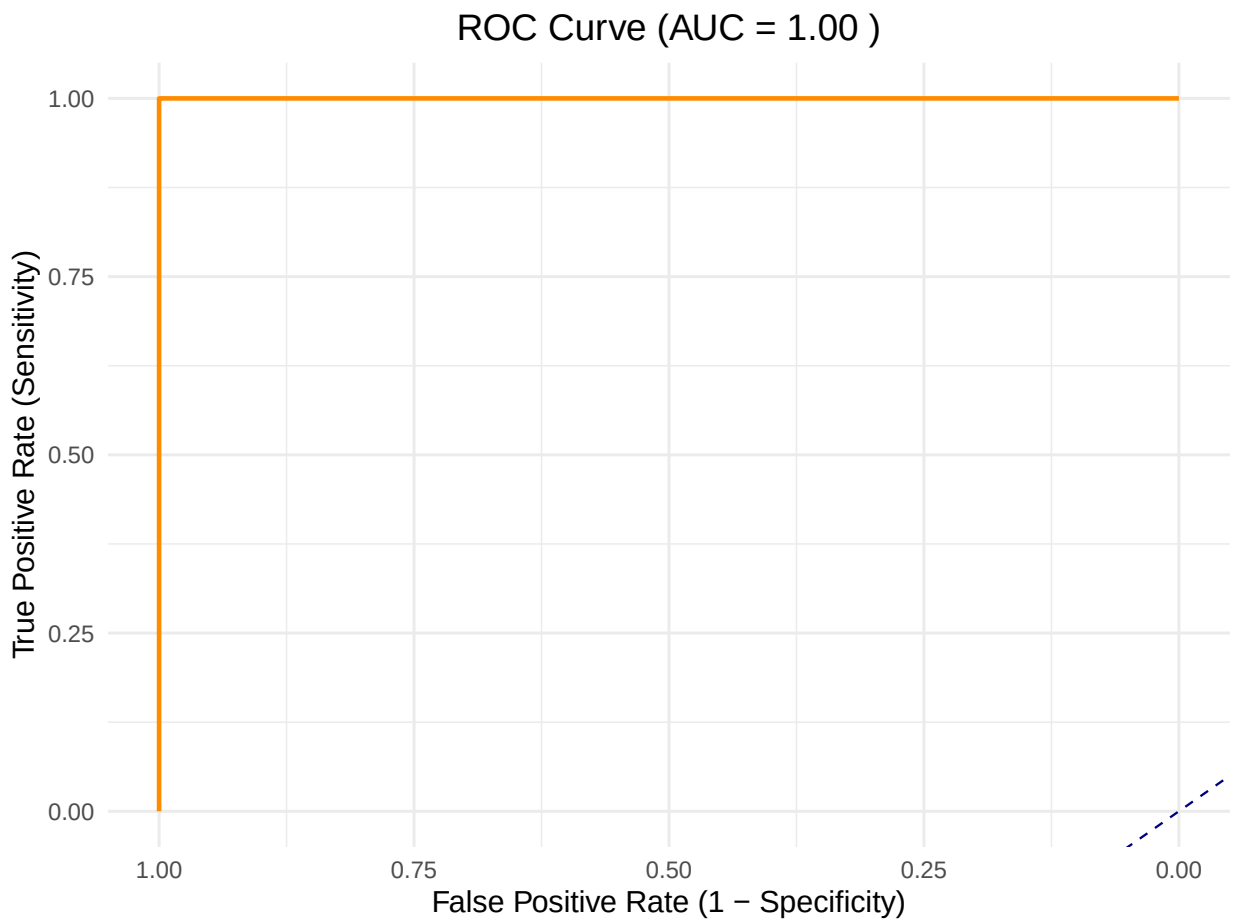

```
# Using caret for more detailed metrics (optional)
# if (requireNamespace("caret", quietly = TRUE)) {
#   cm_caret <- caret::confusionMatrix(data = factor(y_class_pred, levels=c(0,1)),
#                                      reference = factor(actual_test_labels, levels=c(0,1)),
```

```
#                                       positive="1") # Specify positive class
#   print(cm_caret)
# }
```

```
# ROC Curve and AUC
# y_prob_pred already contains the probabilities for the positive class (1)
# actual_test_labels contains the true 0/1 labels

roc_obj <- roc(response = actual_test_labels, predictor = y_prob_pred, quiet = TRUE)
roc_auc_value <- auc(roc_obj)

cat(sprintf('\nAUC: %.2f\n', roc_auc_value))
```

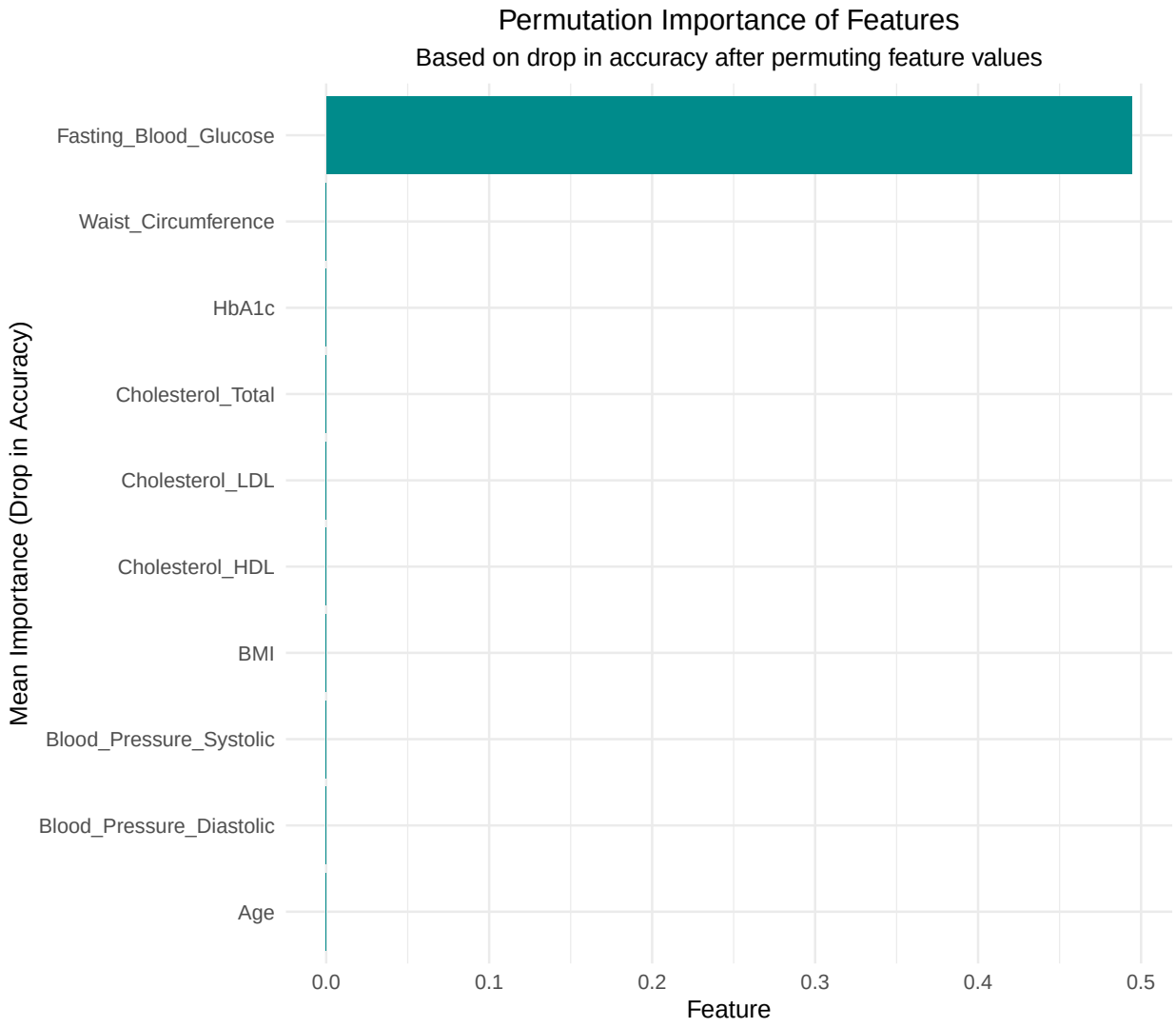
```
##
## AUC: 1.00
```

```
# Plot ROC curve using pROC's ggroc for ggplot2 integration
p_roc <- ggroc(roc_obj, colour = 'darkorange', size = 1) +
  geom_abline(intercept = 0, slope = 1, linetype = "dashed", color = "navy") +
  labs(title = paste("ROC Curve (AUC =", sprintf("%.2f", roc_auc_value), ")"),
       x = "False Positive Rate (1 - Specificity)",
       y = "True Positive Rate (Sensitivity)") +
  theme_minimal(base_size = 14) +
  theme(plot.title = element_text(hjust = 0.5))
print(p_roc)
```




```r
# Permutation Importance using the 'vip' package

# Define a prediction wrapper for vip that returns class predictions (0 or 1)
# This is needed if the metric (e.g., accuracy) expects class labels.
pred_wrapper_classes <- function(object, newdata) {
  probs <- predict(object, newdata, type = "response")
  classes <- ifelse(probs > 0.5, 1, 0)
  return(classes)
}

# Define the accuracy metric function for vip
# truth: true labels, response: predicted labels from pred_wrapper_classes
metric_accuracy <- function(truth, estimate) {
  # Ensure truth and estimate are numeric 0/1 for comparison
  # Given that test_data_for_vip$Diabetes and pred_wrapper_classes output are numeric,
  # as.numeric(as.character(...)) might be redundant but is kept for robustness.
  mean(as.numeric(as.character(truth)) == as.numeric(as.character(estimate)))
}

# Prepare test data for vip: features and the true target column
# vip needs the target column to be present in the 'train' data argument for some methods.
# Ensure 'Diabetes' in test_data is numeric 0/1 for the metric_accuracy function.
test_data_for_vip <- test_data # Contains 'Diabetes' and feature_cols
```

```r
test_data_for_vip$Diabetes <- as.numeric(as.character(test_data_for_vip$Diabetes))

# Calculate permutation importance
set.seed(42) # for reproducibility
perm_imp_obj <- vip::vi_permute(
  logreg_model,                     # Trained model object
  target = "Diabetes",              # Name of the response variable column in 'train' data
  metric = metric_accuracy,         # Custom accuracy metric function
  pred_wrapper = pred_wrapper_classes, # Function to get class predictions
  train = test_data_for_vip,        # Data to compute permutations on (test set)
  nsim = 10,                        # Number of repeats/simulations
  smaller_is_better = FALSE         # For accuracy, larger values are better
)

# Plotting permutation importance using vip's built-in plot function
if (nrow(perm_imp_obj) > 0) {
  vip_plot <- vip::vip(perm_imp_obj, geom = "col", aesthetics = list(fill = "darkcyan")) +
    labs(title = "Permutation Importance of Features",
         subtitle = "Based on drop in accuracy after permuting feature values",
         x = "Mean Importance (Drop in Accuracy)",
         y = "Feature") +
    theme_minimal(base_size = 12) +
    theme(plot.title = element_text(hjust = 0.5),
          plot.subtitle = element_text(hjust = 0.5))
  print(vip_plot)
} else {
  cat("Permutation importance calculation did not return results or no features were important.\n")
}
```



Conclusions and Future Work

We successfully created a predictor using logistic regression to identify diabetes risk based on several important clinical and lifestyle factors. The model achieved a 100% prediction accuracy and provided insights into feature importances. In future iterations, one might consider:

- Conducting feature scaling or transformation to further improve model performance, especially for algorithms sensitive to feature magnitudes.
- Integrating external data sources or time-series data if available.
- Delving deeper into subgroup analyses based on Sex or Ethnicity (if such data were available and appropriate).
- More sophisticated handling of missing data (e.g., imputation) rather than simple row deletion.
- Hyperparameter tuning for the chosen models.

Thank you for your time reviewing this notebook.

‘‘‘