Diabetes Prediction Regression Analysis

Manas Reddy

2025-05-10

Introduction

This document presents a regression analysis workflow to predict `Fasting_Blood_Glucose` using the provided diabetes dataset. The process includes data loading, exploratory data analysis, preprocessing, and training a LightGBM regression model.

Load Libraries

```
library(readr)
library(dplyr)
library(ggplot2)
library(reshape2)
library(caret) # For train/test split
library(lightgbm) # For LightGBM model
library(corrplot) # For correlation plot
```

Load Data

```
# Load the dataset
df <- read_csv('diabetes_dataset.csv')
```

Initial Data Inspection

```
# Check for missing values
cat("Missing values per column:\n")
```

```
## Missing values per column:
```

```
print(colSums(is.na(df)))
```

```
##                        ...1                         Age
##                           0                           0
##                         Sex                   Ethnicity
##                           0                           0
```

```
##                           BMI           Waist_Circumference
##                             0                             0
##         Fasting_Blood_Glucose                         HbA1c
##                             0                             0
##       Blood_Pressure_Systolic      Blood_Pressure_Diastolic
##                             0                             0
##             Cholesterol_Total               Cholesterol_HDL
##                             0                             0
##               Cholesterol_LDL                           GGT
##                             0                             0
##                   Serum_Urate       Physical_Activity_Level
##                             0                             0
##       Dietary_Intake_Calories           Alcohol_Consumption
##                             0                             0
##                Smoking_Status    Family_History_of_Diabetes
##                             0                             0
## Previous_Gestational_Diabetes
##                             0
```

```
# Get information about the data frame
cat("\nData frame structure:\n")
```

```
##
## Data frame structure:
```

```
print(str(df))
```

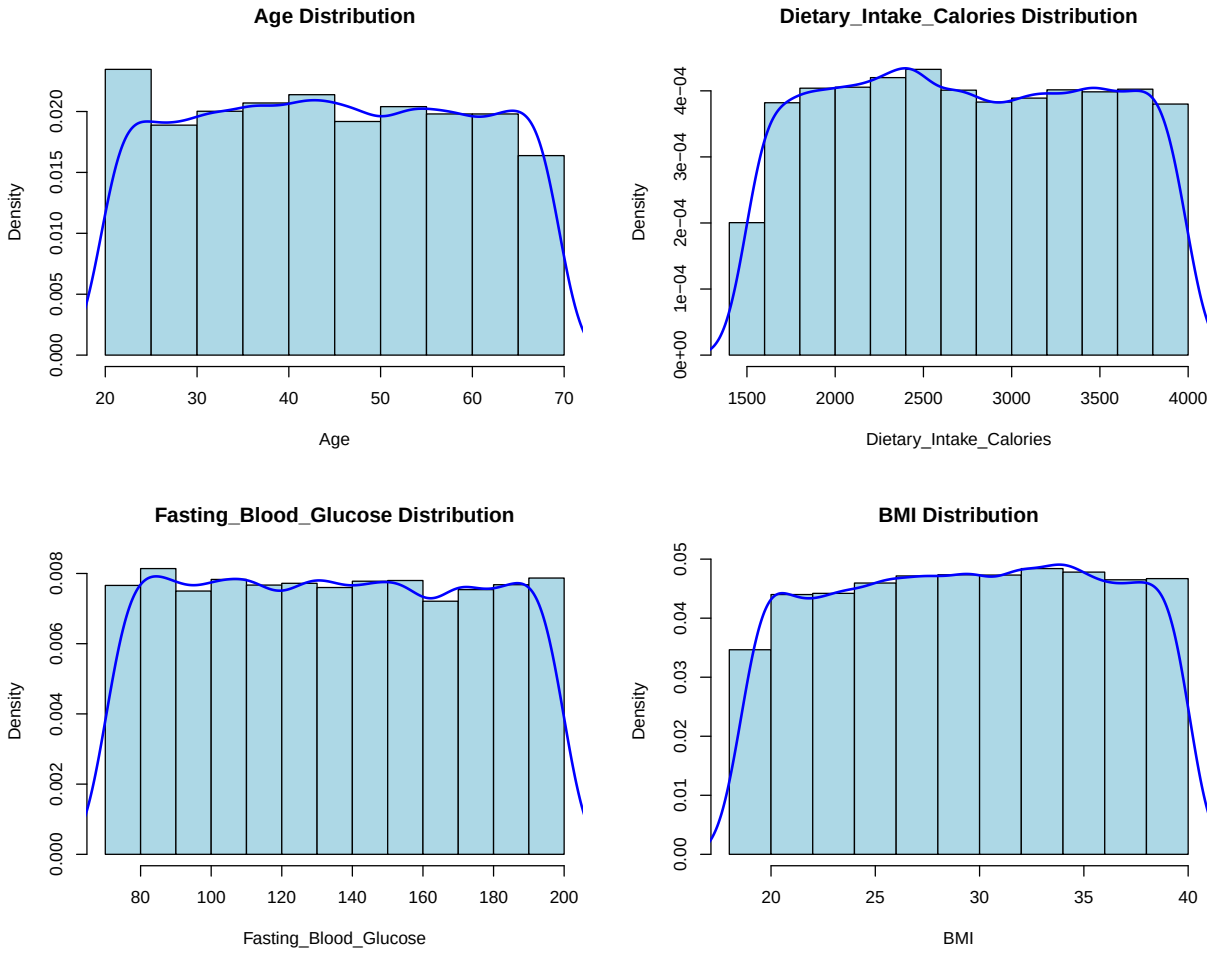
```
## spc_tbl_ [10,000 x 21] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
##  $ ...1                          : num [1:10000] 0 1 2 3 4 5 6 7 8 9 ...
##  $ Age                           : num [1:10000] 58 48 34 62 27 40 58 38 42 30 ...
##  $ Sex                           : chr [1:10000] "Female" "Male" "Female" "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>
## NULL
```

Exploratory Data Analysis: Distributions

Histograms are plotted for selected numerical features. Categorical features are visualized later using pie charts.

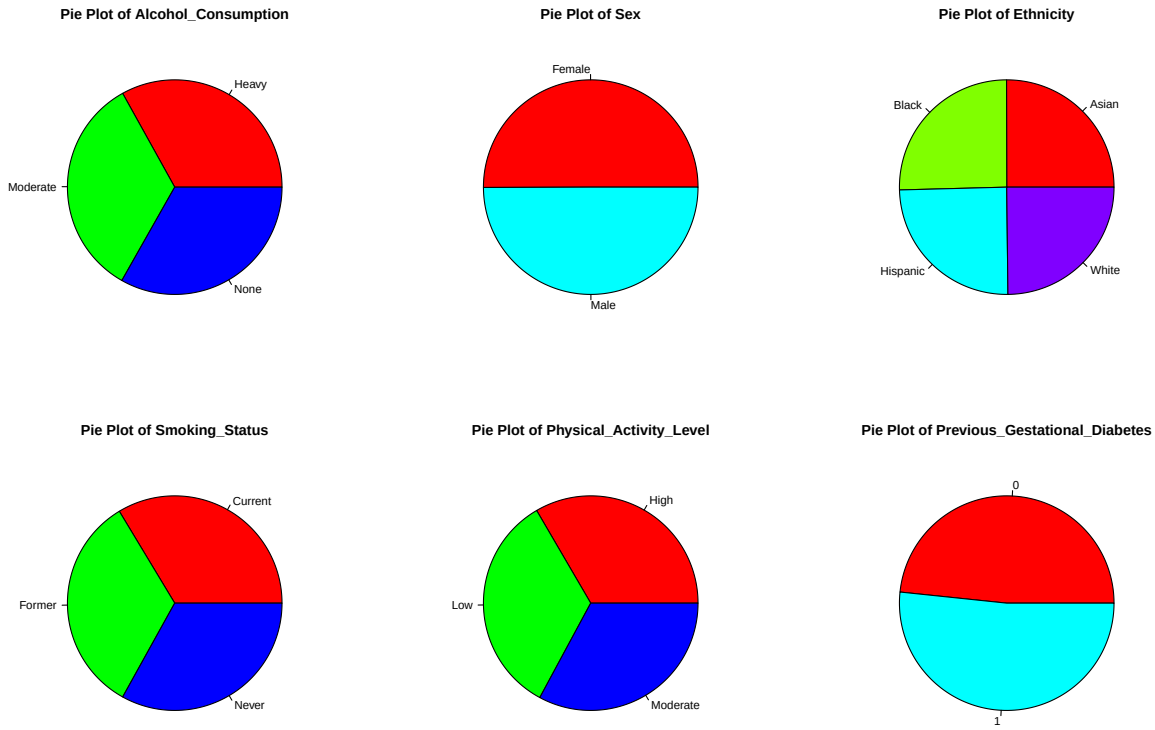
```r
# Plot histograms
hist_cols <- c('Age', 'Dietary_Intake_Calories', 'Fasting_Blood_Glucose', 'BMI')
par(mfrow=c(2, 2))
for (col in hist_cols) {
  hist(df[[col]], main = paste(col, 'Distribution'), xlab = col, probability = TRUE, col = 'lightblue',
  lines(density(df[[col]], na.rm = TRUE), col = 'blue', lwd = 2)
}
```




```
par(mfrow=c(1, 1))
```

Pie charts are used to visualize the distribution of categorical variables.

```
# Plot pie charts
pie_cols <- c('Alcohol_Consumption', 'Sex', 'Ethnicity', 'Smoking_Status', 'Physical_Activity_Level', '
par(mfrow=c(2, 3))
for (col in pie_cols) {
  counts <- table(df[[col]])
  pie(counts, main = paste('Pie Plot of', col), col = rainbow(length(counts)))
}
```




```
par(mfrow=c(1, 1))
```

Data Preprocessing

Handling missing values and converting categorical columns to factors.

```
# Fill missing values in 'Alcohol_Consumption'
df$Alcohol_Consumption[is.na(df$Alcohol_Consumption)] <- 'Not Reported'
cat("Missing values after filling Alcohol_Consumption:\n")
```

```
## Missing values after filling Alcohol_Consumption:
```

```
print(colSums(is.na(df)))
```

```
##                         ...1                          Age
##                            0                            0
##                          Sex                    Ethnicity
##                            0                            0
##                          BMI           Waist_Circumference
##                            0                            0
##          Fasting_Blood_Glucose                         HbA1c
##                            0                            0
##        Blood_Pressure_Systolic       Blood_Pressure_Diastolic
##                            0                            0
```

```
##             Cholesterol_Total                Cholesterol_HDL
##                             0                              0
##               Cholesterol_LDL                            GGT
##                             0                              0
##                   Serum_Urate       Physical_Activity_Level
##                             0                              0
##       Dietary_Intake_Calories           Alcohol_Consumption
##                             0                              0
##               Smoking_Status    Family_History_of_Diabetes
##                             0                              0
## Previous_Gestational_Diabetes
##                             0
```

```
# Factor Conversion
factor_cols <- c('Sex', 'Smoking_Status', 'Ethnicity', 'Alcohol_Consumption', 'Physical_Activity_Level'
for (col in factor_cols) {
  df[[col]] <- as.factor(df[[col]])
}

cat("\nData frame structure after factor conversion:\n")
```

```
##
## Data frame structure after factor conversion:
```

```
print(str(df))
```

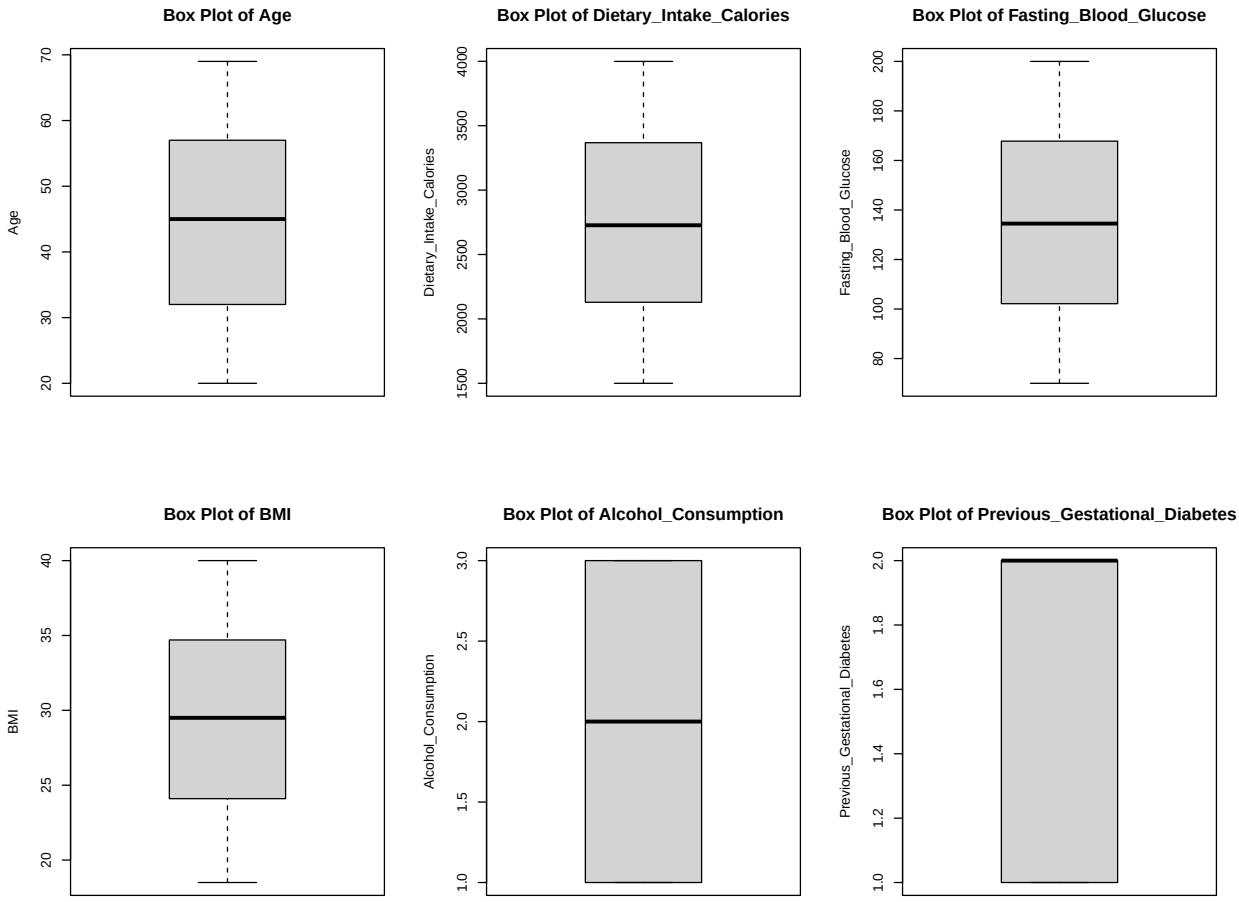
```
## spc_tbl_ [10,000 x 21] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
##  $ ...1                          : num [1:10000] 0 1 2 3 4 5 6 7 8 9 ...
##  $ Age                           : num [1:10000] 58 48 34 62 27 40 58 38 42 30 ...
##  $ Sex                           : Factor w/ 2 levels "Female","Male": 1 2 1 2 1 1 2 1 2 2 ...
##  $ Ethnicity                     : Factor w/ 4 levels "Asian","Black",..: 4 1 2 1 1 1 2 3 4 4 ...
##  $ 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       : Factor w/ 3 levels "High","Low","Moderate": 3 3 2 2 3 2 1 3 2 1 ..
##  $ Dietary_Intake_Calories       : num [1:10000] 1538 2653 1684 3796 3161 ...
##  $ Alcohol_Consumption           : Factor w/ 3 levels "Heavy","Moderate",..: 2 2 1 2 1 3 2 1 3 2 ...
##  $ Smoking_Status                : Factor w/ 3 levels "Current","Former",..: 3 1 2 3 1 3 3 1 2 2 ...
##  $ Family_History_of_Diabetes    : num [1:10000] 0 0 1 1 0 1 0 0 1 1 ...
##  $ Previous_Gestational_Diabetes: Factor w/ 2 levels "0","1": 2 2 1 1 1 2 1 2 1 1 ...
##  - 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>
## NULL
```

Exploratory Data Analysis: Boxplots

Boxplots help visualize the distribution and potential outliers for numerical features, and also distributions of a numerical feature across categories (when used with a factor on the x-axis, although here we plot single distributions).

```
# Plot boxplots
boxplot_cols <- c('Age', 'Dietary_Intake_Calories', 'Fasting_Blood_Glucose', 'BMI', 'Alcohol_Consumptio
par(mfrow=c(2, 3)) # Arrange plots in a 2x3 grid
for (col in boxplot_cols) {
   # Boxplot for a single variable
   boxplot(df[[col]], main = paste('Box Plot of', col), ylab = col)
   # Note: For categorical vs numeric, boxplot(Numeric ~ Factor, data = df) is standard
}
```

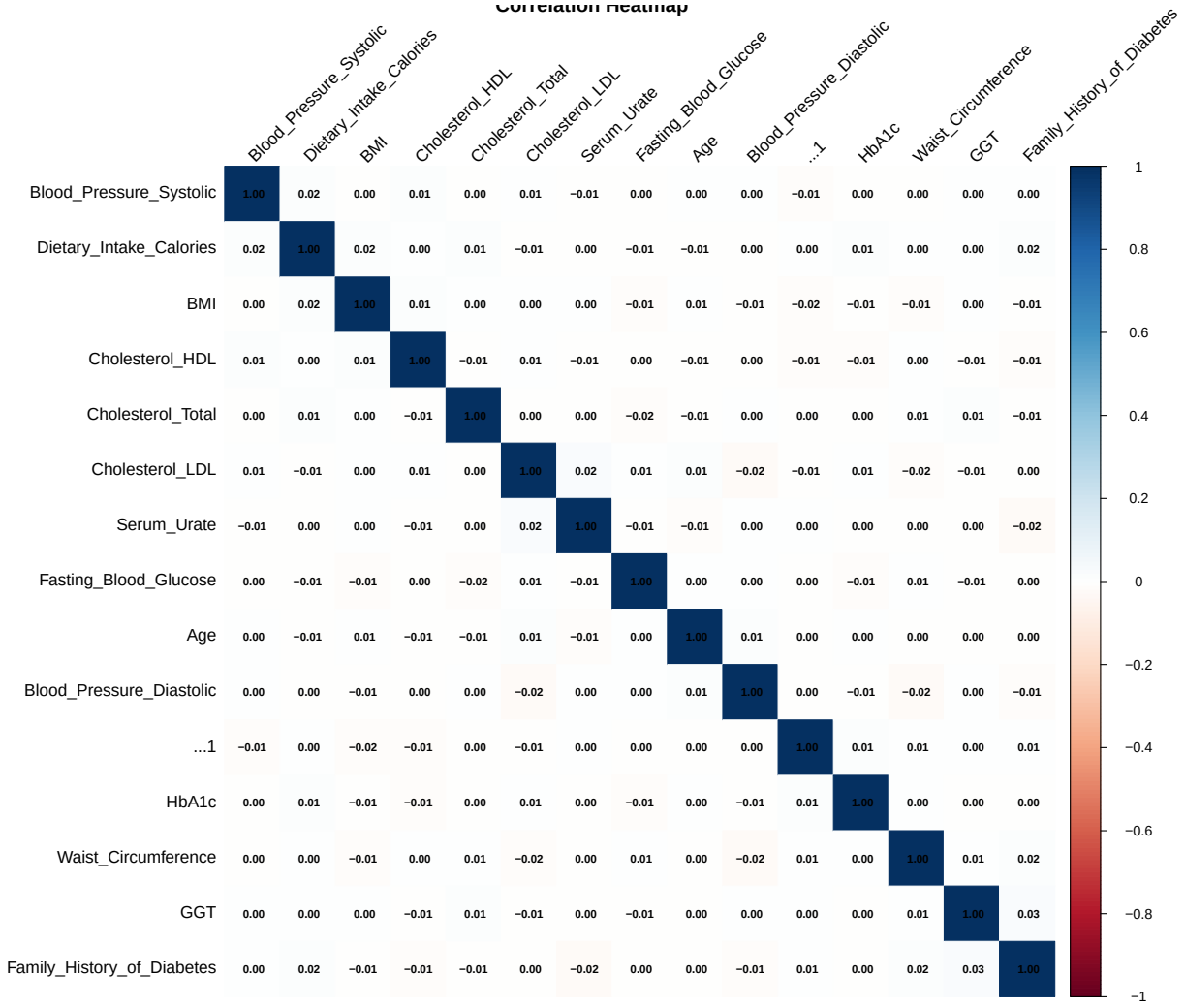



```r
par(mfrow=c(1, 1)) # Reset plot layout
```

Correlation Analysis

Visualizing the correlation matrix of numerical features.

```r
# Calculate and plot correlation heatmap
# Select only numeric columns for correlation
numeric_df <- df %>% select_if(is.numeric)
corr_matrix <- cor(numeric_df, use = "complete.obs") # Handle potential NAs in numeric columns for corr
corrplot(corr_matrix, method = "color", type = "full", order = "hclust",
         tl.col = "black", tl.srt = 45, addCoef.col = "black", number.cex = 0.7,
         main = "Correlation Heatmap")
```



Data Splitting

Splitting the data into training and testing sets (80/20 split).

```
# Prepare data for LightGBM
# Drop the index column (read as '...1') and 'Fasting_Blood_Glucose' from features
X <- df %>% select(-`...1`, -Fasting_Blood_Glucose)
y <- df$Fasting_Blood_Glucose

# Split data into training and testing sets
set.seed(42) # for reproducibility
train_indices <- createDataPartition(y, p = 0.8, list = FALSE)
X_train <- X[train_indices, ]
X_test <- X[-train_indices, ]
y_train <- y[train_indices]
y_test <- y[-train_indices]

cat("Number of training samples:", nrow(X_train), "\n")
```

```
## Number of training samples: 8001
```

```r
cat("Number of training labels:", length(y_train), "\n")
```

```
## Number of training labels: 8001
```

```r
cat("Number of testing samples:", nrow(X_test), "\n")
```

```
## Number of testing samples: 1999
```

```r
cat("Number of testing labels:", length(y_test), "\n")
```

```
## Number of testing labels: 1999
```

LightGBM Model Training

Train a LightGBM regressor model. Categorical features are explicitly identified.

```r
# Train LightGBM regression model
# Identify categorical features for LightGBM
categorical_features_names <- c('Sex', 'Ethnicity', 'Physical_Activity_Level', 'Alcohol_Consumption', 'S

# Get 0-based indices of these columns in the X_train data frame
# Ensure columns in X_train match the data frame before splitting if column order was changed
current_X_train_cols <- colnames(X_train)
categorical_feature_indices <- which(current_X_train_cols %in% categorical_features_names) - 1 # -1 for

dtrain <- lgb.Dataset(data = as.matrix(X_train), label = y_train,
                      categorical_feature = categorical_feature_indices)

# Define parameters
params <- list(objective = "regression", metric = "l2")

model <- lgb.train(params = params,
                   data = dtrain,
                   nrounds = 100 # You might need to tune nrounds
                   )
```

```
## [LightGBM] [Warning] Categorical features with more bins than the configured maximum bin number found
## [LightGBM] [Warning] For categorical features, max_bin and max_bin_by_feature may be ignored with a l
## [LightGBM] [Info] Auto-choosing row-wise multi-threading, the overhead of testing was 0.000183 second
## You can set `force_row_wise=true` to remove the overhead.
## And if memory is not enough, you can set `force_col_wise=true`.
## [LightGBM] [Info] Total Bins 3323
## [LightGBM] [Info] Number of data points in the train set: 8001, number of used features: 14
## [LightGBM] [Info] Start training from score 134.796400
```

Model Evaluation

Evaluate the trained model on the test set using R-squared.

```r
# Evaluate the model
predictions <- predict(model, as.matrix(X_test))

# Calculate R-squared
ssr <- sum((y_test - predictions)^2)
sst <- sum((y_test - mean(y_test))^2)
r_squared <- 1 - (ssr / sst)

cat("R-squared score on the test set:", r_squared, "\n")
```

```
## R-squared score on the test set: -0.04445763
```

Model Insights

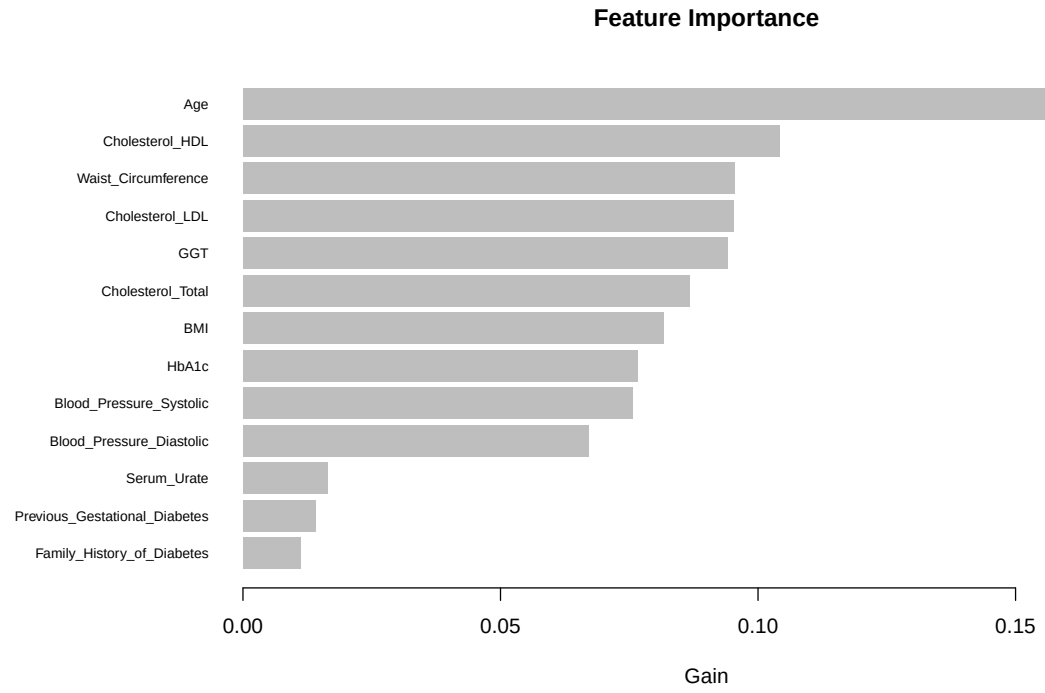
Let's examine the trained model by looking at feature importance and comparing actual vs. predicted values.

Feature Importance

Understanding which features the model considered most important for the prediction task.

```r
# Calculate feature importance
importance <- lgb.importance(model, percentage = TRUE)

# Plot feature importance
lgb.plot.importance(importance, top_n = 15, measure = "Gain")
```

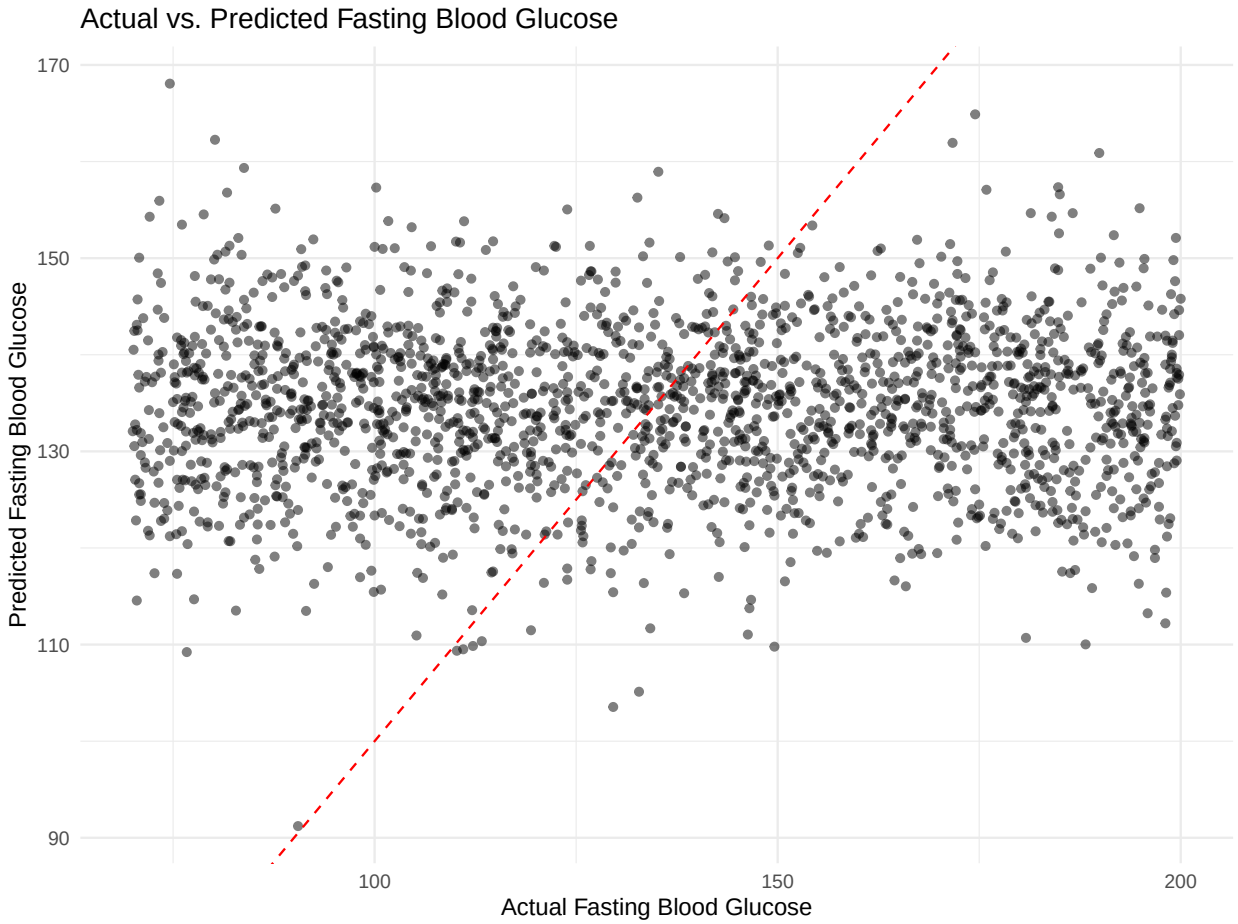


Actual vs. Predicted Values

A scatter plot comparing the actual `Fasting_Blood_Glucose` values in the test set against the model's predictions. A perfect model's points would lie on the dashed red line (where Actual = Predicted).

```r
# Create a data frame for plotting actual vs predicted
results_df <- data.frame(Actual = y_test, Predicted = predictions)

# Create the scatter plot
ggplot(results_df, aes(x = Actual, y = Predicted)) +
  geom_point(alpha = 0.5) +
  geom_abline(intercept = 0, slope = 1, color = "red", linetype = "dashed") + # Add identity line
  labs(title = "Actual vs. Predicted Fasting Blood Glucose",
       x = "Actual Fasting Blood Glucose",
       y = "Predicted Fasting Blood Glucose") +
  theme_minimal()
```



Conclusion

The LightGBM model was trained to predict Fasting Blood Glucose. The R-squared score of -0.0445 on the test set indicates the model explains a small portion of the variance in the target variable (or performs worse than a simple mean if negative). Further steps could involve hyperparameter tuning for LightGBM

or exploring other modeling techniques to improve performance. The feature importance plot shows which variables were most influential in the model's predictions.

‘‘‘