# Data Science in Health Project: CHD Logistic Prediction Model Comparison

## **Overview**

This data science project aims to analyze health data collected from individuals to predict the likelihood of developing coronary heart disease (CHD) within ten years. Insights produced in this project could save countless lives and provide tremendous benefit in the medical space. The project utilizes logistic regression to build a predictive model based on various health indicators.

This was not the initial project I originally intended to do; it was a mobile health dataset from body sensors containing over ten thousand data points. Because of my computer, it took forever to run and it became evident that I had to switch plans. During that time, I thought about a disease that runs in my family and decided to use it as inspiration for this project.

## Literature Review of Previous Works

Nishat, M., Ahmed, S., Hasan, M. M., Ali, M. H., Saha, R., & Mahmud, M. (2021). Performance Evaluation and Comparative Analysis of Different Machine Learning Algorithms in Predicting Cardiovascular Disease.

The previous study utilized various machine learning methods to predict CHD. Utilizing data from the University of California, Irvine repository, twelve algorithms were assessed using default hyperparameters, grid search cross-validation, and random search cross-validation methods. Both accuracy and computational time were measured, with hard and soft voting ensemble classifiers achieving 92% accuracy. Adaboost algorithm demonstrated superior precision and specificity compared to ensemble classifiers. The analysis extensively compares algorithm performance across multiple metrics including accuracy, precision, sensitivity, specificity, F1 score, and ROC-AUC.

Even though there were many models that intrigued me, I decided to personally use logistic regression because of its simplicity, efficiency, and clinical acceptance. Attempts of data analysis on the dataset posted on kaggle showed the following:\

- Men are more likely to get heart disease than women. As people get older, smoke more cigarettes, or have higher blood pressure, their chances of getting heart disease also go up.

- Having higher total cholesterol doesn't seem to make much difference in the chance of getting heart disease. This might be because the cholesterol test includes both good and bad cholesterol. Glucose levels also don't have a big impact on the chance of getting heart disease, only a tiny bit.

- The model we used predicted heart disease correctly 88% of the time. It's better at saying who doesn't have heart disease than who does.

## **Methodology**

I will be experimenting with min-max normalization and using the top absolute valued correlated variables associated with the variable ‘TenYearCHD’ which describes whether a subject has cardiovascular disease or not. The top correlated values were selected based on their absolute values.The objective of the project is to play around with the data and double check the claims mentioned in previous analysis.

## Preliminaries

### **Loading Dataset and Packages**

The project utilizes several R packages for data manipulation, visualization, and model training. Key packages include caret, pROC, ggplot2, and dplyr. The dataset is loaded using the read.csv() function from the foreign package. R studio was used as the main code editor to compile the project and its resources.

Loading Dataset and Packages

```{r}
#required packages
list.of.packages <-
c("foreign","rjags","dplyr","ggplot2","plotly","reshape2","bnlearn","nnet","caret","pROC","penalized","caret")

#install if necessary
new.packages <- list.of.packages[!(list.of.packages %in% installed.packages()[,"Package"])]
if(length(new.packages)) install.packages(new.packages)

#load all packages
lapply(list.of.packages, library, character.only = TRUE)
```

```{r}
# Example script to read data
data <- read.csv('/Users/unclenamo/Desktop/Zhaw/Data Science for Health Project /Data Science in Health Final
Project Folder/framingham.csv')
```

```{r}
head(data,n = 10)
```

## Conditional Indexing, Selection, and Initial Visualization

### **Handling Missing Values**

Missing values in the dataset are removed using the na.omit() function to ensure data integrity and consistency. Without removing the null values, the dataset’s dimensions would have been problematic in the analysis and training phase.

###

~~~
```{r}
str(data)
```
~~~

~~~
```{r}
clean_data <- na.omit(data)
head(clean_data)
```
~~~

~~~
```{r}
str(clean_data)
```
~~~

### **Descriptive Statistics**

Descriptive statistics such as mean, median, minimum, maximum, standard deviation, and quartiles are calculated for numeric variables in the dataset to gain insights into the distribution of health indicators. A function was created to compare male and female subjects as well as the average statistics of subjects with CHD and without CHD. I had problems with this function because I had misplaced the variables ‘clean_data’ and ‘data’ causing the difference not to be shown.

~~~
```{r}
# Check for NULL values
is.null(clean_data)

# Summary of dataframe
summary(clean_data)

# Structure of dataframe
str(clean_data)

```
~~~

~~~
```{r}
calculate_descriptive_statistics <- function(clean_data) {
    # Select specific numeric columns
    numeric_cols <- c("age", "education", "cigsPerDay", "totChol", "sysBP", "diaBP", "BMI","heartRate",
"glucose")

   # Filter the data based on selected numeric columns
   numeric_data <- clean_data[, numeric_cols]


  # Calculate descriptive statistics
  descriptive_stats <- apply(clean_data[, numeric_cols], 2, function(x) {
    mean_val <- mean(x, na.rm = TRUE)
    median_val <- median(x, na.rm = TRUE)
    min_val <- min(x, na.rm = TRUE)
    max_val <- max(x, na.rm = TRUE)
    sd_val <- sd(x, na.rm = TRUE)
    q1 <- quantile(x, probs = 0.25, na.rm = TRUE)
    q3 <- quantile(x, probs = 0.75, na.rm = TRUE)
    iqr <- q3 - q1

    result <- c(mean = mean_val,
                median = median_val,
                min = min_val,
                max = max_val,
                sd = sd_val,
                q1 = q1,
                q3 = q3,
                IQR = iqr)

    return(result)
  })

  # Create a dataframe from the results
  descriptive_stats_df <- t(as.data.frame(descriptive_stats))
  colnames(descriptive_stats_df) <- c("Mean", "Median", "Min", "Max", "SD", "Q1", "Q3", "IQR")

  return(descriptive_stats_df)
}

```
~~~

~~~
```{r}
# Assuming 'data' is the name of your dataset
~~~

~~~
descriptive_stats <- calculate_descriptive_statistics(clean_data)
print(descriptive_stats)
```

### Male vs Female Selection

```{r}
# Select rows where 'male' is equal to 0
female_data <- clean_data[clean_data$male == 0, ]

# Select rows where 'male' is equal to 1
male_data <- clean_data[clean_data$male == 1, ]
```

```{r}
print(female_data)
```

```{r}
print(male_data)
```

```{r}
# Assuming 'data' is the name of your dataset
descriptive_stats_female <- calculate_descriptive_statistics(female_data)
print(descriptive_stats_female)
```

```{r}
# Assuming 'data' is the name of your dataset
descriptive_stats_male <- calculate_descriptive_statistics(male_data)
print(descriptive_stats_male)
```

```{r}
# Select rows where 'TenYearCHD' is equal to 0
no_chd_data <-clean_data[clean_data$TenYearCHD == 0, ]

# Select rows where 'TenYearCHD' is equal to 1
chd_data <- clean_data[clean_data$TenYearCHD == 1, ]

```

```{r}
print(no_chd_data)
```

```{r}
print(chd_data)
```

```{r}
count(chd_data)
```

```{r}
count(no_chd_data)
```

```{r}
# Assuming 'data' is the name of your dataset
descriptive_stats_nochd <- calculate_descriptive_statistics(no_chd_data)
print(descriptive_stats_nochd)

```

```{r}
# Assuming 'data' is the name of your dataset
descriptive_stats_chd <- calculate_descriptive_statistics(chd_data)
print(descriptive_stats_chd)
```

#### Comparison Function CHD vs No CHD

```{r}
# Assuming 'data' is the name of your dataset

# Calculate descriptive statistics for individuals with and without CHD
chd_stats <- calculate_descriptive_statistics(clean_data[clean_data$TenYearCHD == 1, ])
no_chd_stats <- calculate_descriptive_statistics(clean_data[clean_data$TenYearCHD == 0, ])

# Combine the statistics into a single dataframe for comparison
comparison <- data.frame(Feature = rownames(chd_stats),
                         CHD_Mean = chd_stats[, "Mean"],
~~~

~~~
                              No_CHD_Mean = no_chd_stats[, "Mean"],
                              Difference = chd_stats[, "Mean"] - no_chd_stats[, "Mean"])

# Print the comparison
print(comparison)


```

## Visualization Pre-Training

The project includes various data visualizations to explore relationships between different health variables
and their impact on the likelihood of developing CHD. Visualizations include scatter plots, bar charts, and
violin plots.

```{r}
head(data)
```

```{r}
# Select only numeric columns
numeric_data <- clean_data[, sapply(data, is.numeric)]

# Calculate the correlation matrix for the numeric data
correlation_matrix <- cor(numeric_data)

# Print the correlation matrix
print(correlation_matrix)

```

```{r}
display_correlation_pairs <- function(correlation_matrix) {
  # Convert correlation matrix to a long-form data frame
  df <- reshape2::melt(correlation_matrix)

  # Remove NA and duplicate rows
  df <- df[complete.cases(df), ]
  df <- df[!duplicated(df), ]

  # Sort by absolute correlation value in descending order
  df <- df[order(-abs(df$value)), ]

  # Print the sorted pairs
  print(df)
}

display_correlation_pairs(correlation_matrix)
```

```{r}
# Set the size of the plot
options(repr.plot.width = 30, repr.plot.height = 15) # Adjust width and height as needed

# Create a heatmap of the correlation matrix with color scale
heatmap(correlation_matrix,
        col = colorRampPalette(c("blue", "yellow", "red"))(100),
        scale = "row",   # Add scale for rows
        symm = TRUE,     # To make the heatmap symmetric
        margins = c(10, 10))  # To provide extra space for row and column names

# Add color scale legend
legend("right",    # Position the legend to the right
       legend = c(-1, 0, 1),  # Values for the color scale (simplified)
       fill = colorRampPalette(c("blue", "yellow", "red"))(3),  # Color gradient for the legend
       title = "Correlation")  # Title for the legend


```

```{r}
print(clean_data)
```

```{r}
# Create a basic violin plot
violin_plot <- ggplot(clean_data, aes(x = factor(TenYearCHD), y = cigsPerDay)) +
  geom_violin() +
  labs(x = "TenYearCHD", y = "Cigarettes Per Day") +
  theme_minimal()

# Display the violin plot
print(violin_plot)
~~~

~~~~
```

```{r}
# Calculate the counts of individuals with and without TenYearCHD
chd_counts <- table(clean_data$TenYearCHD)

# Create the bar chart
barplot(chd_counts, col = c("blue", "red"), main = "Counts of Individuals with and without TenYearCHD",
        xlab = "TenYearCHD", ylab = "Count", legend = c("No CHD", "CHD"))



```

```{r}
# Calculate the proportion of individuals with and without CHD among smokers and non-smokers
chd_prop <- aggregate(TenYearCHD ~ currentSmoker, data = clean_data, FUN = function(x) sum(x == 1) /
length(x))
names(chd_prop) <- c("currentSmoker", "CHD_Proportion")

# Convert currentSmoker to factor for better visualization
chd_prop$currentSmoker <- factor(chd_prop$currentSmoker, levels = c(0, 1), labels = c("Non-Smoker",
"Smoker"))

# Create the grouped bar chart
grouped_bar_chart <- ggplot(chd_prop, aes(x = currentSmoker, y = CHD_Proportion, fill = currentSmoker)) +
  geom_bar(stat = "identity", position = "dodge") +
  labs(x = "Smoking Status", y = "Proportion with CHD", fill = "Smoking Status") +
  scale_fill_manual(values = c("Non-Smoker" = "blue", "Smoker" = "red")) +  # Customizing fill colors
  theme_minimal()
```

```{r}
# Display the grouped bar chart
print(grouped_bar_chart)
```

```
```

![]()

```{r}
# Load the ggplot2 library
library(ggplot2)

# Assuming 'data' is the name of your dataset
# Scatterplot for Blood Pressure (sysBP, diaBP)
bp_scatterplot <- ggplot(data, aes(x = sysBP, y = diaBP)) +
  geom_point() +
  labs(x = "Systolic Blood Pressure", y = "Diastolic Blood Pressure", title = "Blood Pressure Scatterplot") +
  theme_minimal()


print(bp_scatterplot)

```

```{r}
# Calculate the proportion of individuals with and without CHD for each level of diabetes status
diabetes_chd_prop <- aggregate(TenYearCHD ~ diabetes, data = data, FUN = function(x) mean(x == 1))
names(diabetes_chd_prop) <- c("Diabetes_Status", "CHD_Proportion")

# Convert diabetes status to factor for correct ordering in the plot
diabetes_chd_prop$Diabetes_Status <- factor(diabetes_chd_prop$Diabetes_Status, levels = c(0, 1), labels =
c("No Diabetes", "Diabetes"))

# Create a grouped bar plot
grouped_bar_plot <- ggplot(diabetes_chd_prop, aes(x = Diabetes_Status, y = CHD_Proportion, fill =
Diabetes_Status)) +
  geom_bar(stat = "identity") +
  labs(x = "Diabetes Status", y = "Proportion with CHD", fill = "Diabetes Status", title = "Proportion of CHD
by Diabetes Status") +
  theme_minimal()

# Display the grouped bar plot
print(grouped_bar_plot)
```

-   **Low Cholesterol**: Total cholesterol level below 200 mg/dL.

-   **Desirable/Medium Cholesterol**: Total cholesterol level between 200 mg/dL and 239 mg/dL.
~~~~

~~~
-   **High Cholesterol**: Total cholesterol level 240 mg/dL or higher.

```{r}
# Create a new variable to categorize cholesterol levels
data$Cholesterol_Category <- cut(data$totChol,
                                 breaks = c(-Inf, 200, 239, Inf),
                                 labels = c("Low", "Medium", "High"),
                                 right = FALSE)

# Calculate the proportion of individuals with and without CHD for each level of cholesterol category
cholesterol_chd_prop <- aggregate(TenYearCHD ~ Cholesterol_Category, data = data, FUN = function(x) mean(x ==
1))
names(cholesterol_chd_prop) <- c("Cholesterol_Category", "CHD_Proportion")

# Create a grouped bar plot
grouped_bar_plot_cholesterol <- ggplot(cholesterol_chd_prop, aes(x = Cholesterol_Category, y =
CHD_Proportion, fill = Cholesterol_Category)) +
  geom_bar(stat = "identity") +
  labs(x = "Cholesterol Category", y = "Proportion with CHD", fill = "Cholesterol Category", title =
"Proportion of CHD by Cholesterol Category") +
  theme_minimal()

# Display the grouped bar plot
print(grouped_bar_plot_cholesterol)


```

-   Age Range

    -   Young Adult: Age \< 40

    -   Middle-Aged Adult: 40 ≤ Age \< 65

    -   Elderly: Age ≥ 65

-   BMI

    -   Young Adult: Age \< 40

    -   Middle-Aged Adult: 40 ≤ Age \< 65

    -   Elderly: Age ≥ 65

```{r}
# Categorize BMI
data$BMI_Category <- cut(data$BMI,
                         breaks = c(-Inf, 18.5, 24.9, 29.9, Inf),
                         labels = c("Underweight", "Normal Weight", "Overweight", "Obesity"),
                         right = FALSE)

# Categorize Age
data$Age_Category <- cut(data$age,
                         breaks = c(-Inf, 40, 65, Inf),
                         labels = c("Young Adult", "Middle-Aged Adult", "Elderly"),
                         right = FALSE)

# Calculate the proportion of individuals with and without CHD for each level of BMI category
bmi_chd_prop <- aggregate(TenYearCHD ~ BMI_Category, data = data, FUN = function(x) mean(x == 1))
names(bmi_chd_prop) <- c("BMI_Category", "CHD_Proportion")

# Create a grouped bar plot for BMI
grouped_bar_plot_bmi <- ggplot(bmi_chd_prop, aes(x = BMI_Category, y = CHD_Proportion, fill = BMI_Category))
+
  geom_bar(stat = "identity") +
  labs(x = "BMI Category", y = "Proportion with CHD", fill = "BMI Category", title = "Proportion of CHD by
BMI Category") +
  theme_minimal()

# Calculate the proportion of individuals with and without CHD for each level of diabetes status
diabetes_chd_prop <- aggregate(TenYearCHD ~ diabetes, data = data, FUN = function(x) mean(x == 1))
names(diabetes_chd_prop) <- c("Diabetes_Status", "CHD_Proportion")

# Create a grouped bar plot for Diabetes Status
grouped_bar_plot_diabetes <- ggplot(diabetes_chd_prop, aes(x = Diabetes_Status, y = CHD_Proportion, fill =
Diabetes_Status)) +
  geom_bar(stat = "identity") +
  labs(x = "Diabetes Status", y = "Proportion with CHD", fill = "Diabetes Status", title = "Proportion of CHD
by Diabetes Status") +
  theme_minimal()

# Calculate the proportion of individuals with and without CHD for each age category
age_chd_prop <- aggregate(TenYearCHD ~ Age_Category, data = data, FUN = function(x) mean(x == 1))
names(age_chd_prop) <- c("Age_Category", "CHD_Proportion")
~~~

~~~
# Create a grouped bar plot for Age
grouped_bar_plot_age <- ggplot(age_chd_prop, aes(x = Age_Category, y = CHD_Proportion, fill = Age_Category))
+
  geom_bar(stat = "identity") +
  labs(x = "Age Category", y = "Proportion with CHD", fill = "Age Category", title = "Proportion of CHD by
Age Category") +
  theme_minimal()

# Display the grouped bar plots
print(grouped_bar_plot_bmi)
print(grouped_bar_plot_diabetes)
print(grouped_bar_plot_age)

```

```{r}
head(clean_data)
```

## **Data Normalization**

Numeric variables in the dataset are normalized using min-max scaling to ensure uniformity and prevent any
single variable from dominating the model due to differences in scale. Min-max normalization was used to
ensure ranges of zero to one since ROC utilizes probability.

```{r}

# Select only numeric columns except for the "activity" column from the dataset
numeric_data <- clean_data[, sapply(clean_data, is.numeric)]

# Min-max scaling to normalize between 0 and 1
min_max_scaled <- apply(numeric_data, 2, function(x) (x - min(x)) / (max(x) - min(x)))

# Convert the scaled data back to a data frame
min_max_normalized_data <- as.data.frame(min_max_scaled)

head(min_max_normalized_data)


```

# Training Logistic Regression Models

## **Model Training**

### **Logistic Regression**

Logistic regression is employed as the primary machine learning model for predicting the probability of
developing CHD. The glm() function is used to train the logistic regression model, and evaluation metrics
such as accuracy, precision, recall, specificity, F1 score, and Matthews correlation coefficient (MCC) are
computed to assess model performance.

I had a tremendously difficult time with a different library during the training process with cross
validation, so I decided to take a step back and use the basic  glm() function instead. It was an error due
to multiplications of incorrect object dimensions.

-   Error in dimnames(out) \<- \*vtmp\* : length of 'dimnames' [2] not equal to array extent

The formula that was used for the logistic regression were the highly correlated values with the variable
TenYearCHD.

TenYearCHD \~ male + age + sysBP + prevalentHyp + diaBP + glucose + diabetes

```{r}
# Check unique values in TenYearCHD
unique_values <- unique(min_max_normalized_data$TenYearCHD)

# Check if there are any unexpected values
print(unique_values)

```

```{r}
print(min_max_normalized_data)
```

```{r}
# Check the number of rows for the column TenYearCHD
num_rows <- nrow(min_max_normalized_data$TenYearCHD)
print(num_rows)
```

#### Training Test Data Split
~~~

~~~
```{r}
# Split data into training and test sets
set.seed(123)  # for reproducibility
train_index <- createDataPartition(min_max_normalized_data$TenYearCHD, p = 0.8, list = FALSE)
train_data <- min_max_normalized_data[train_index, ]
test_data <- min_max_normalized_data[-train_index, ]

```

```{r}
print(min_max_normalized_data)
```

```{r}
head(test_data)
```

```{r}
print(train_data)
```

```{r}
# Ensure dimensions match
length(test_data$TenYearCHD)
length(train_data$TenYearCHD)

```

```{r}
str(test_data)
```

```{r}
str(train_data)
```

#### Model Training

```{r}

# Define the formula with specific variables
formula <- as.formula("TenYearCHD ~ male + age + sysBP + prevalentHyp + diaBP + glucose + diabetes")

# Train the logistic regression model
model <- glm(formula, data = train_data, family = binomial)

# Predict probabilities of positive class (1)
predictions <- predict(model, newdata = test_data, type = "response")

# Compute ROC curve
roc_curve <- roc(test_data$TenYearCHD, predictions)

# Plot ROC curve
plot(roc_curve, main = "ROC Curve", col = "blue", lwd = 2)

# Add AUC to the plot
legend("bottomright", legend = paste("AUC =", round(auc(roc_curve), 2)), col = "blue", lwd = 2)


```

```{r}
summary(model)
```

```{r}
levels(factor(round(predictions)))
levels(test_data$TenYearCHD)

# Get unique levels from both factors
all_levels <- union(levels(factor(round(predictions))), levels(test_data$TenYearCHD))

# Set the same levels for both factors
predictions_factor <- factor(round(predictions), levels = all_levels)
actual_values_factor <- factor(test_data$TenYearCHD, levels = all_levels)

```

```{r}
conf_mat <- confusionMatrix(predictions_factor, actual_values_factor)
# Extracting specific metrics
accuracy <- conf_mat$overall['Accuracy']
precision <- conf_mat$byClass['Pos Pred Value']
~~~

~~~
recall <- conf_mat$byClass['Sensitivity']
specificity <- conf_mat$byClass['Specificity']
f1_score <- (2 * precision * recall) / (precision + recall)
mcc <- cor(test_data$TenYearCHD, round(predictions))


# Print the evaluation metrics
cat("Accuracy:", accuracy, "\n")
cat("Precision:", precision, "\n")
cat("Recall:", recall, "\n")
cat("Specificity:", specificity, "\n")
cat("F1 Score:", f1_score, "\n")
cat("Matthews Correlation Coefficient:", mcc, "\n")

```

```{r}
# Set the threshold for predicting positive class
threshold <- 0.5

# Predict classes based on the threshold
predicted_classes <- ifelse(predictions > threshold, 1, 0)

# Create the confusion matrix
conf_matrix <- table(Actual = test_data$TenYearCHD, Predicted = predicted_classes)

# Print the confusion matrix
print(conf_matrix)


```

\
~~~