library(glmnet)

library(naniar)

library(GGally)

library(visdat)

library(ggplot2)

library(dplyr)

library(tidyr)

library(plm)

library(psych)

library(forecast)

library(corrplot)

library(readxl)

MISSING\_DATA <- read\_excel("MISSING DATA.xlsx")

View(MISSING\_DATA)

attach(MISSING\_DATA)

# MISSING VALUES DETECTION AND HANDLING

# Calculate the total number of missing values in each variable

missing\_counts <- colSums(is.na(MISSING\_DATA))

print(missing\_counts)

miss\_var\_summary(CD)

# Heatplot of missingness across the entire data frame

vis\_miss(MISSING\_DATA) +

ggtitle("Missing Values Visualization") +

theme(plot.title = element\_text(hjust = 0.5))

#Visualizing missing data

gg\_miss\_var(MISSING\_DATA, show\_pct = TRUE) +

ggtitle("Missing Values Visualization")+

theme(plot.title = element\_text(hjust = 0.5))

# Z score

# Extract the relevant numeric columns

numeric\_columns <- MISSING\_DATA[, c("GDP", "PovertyHeadCount", "EducationEnrolment", "UnemploymentRate", "PopulationGrowthRate", "LifeExpectancy")]

# Calculate z-scores for each numeric column

z\_scores <- scale(numeric\_columns)

# Set the threshold for outlier detection (adjust as needed)

threshold <- 2

# Identify outliers based on the absolute z-scores exceeding the threshold

outliers <- abs(z\_scores) > threshold

# Summarize the number of outliers in each column

outliers\_summary <- colSums(outliers)

# Print the summary

print(outliers\_summary)

# Boxplot visualisation

# Boxplot to visualize distribution and outliers

par(mfrow = c(1,1))

boxplot(numeric\_columns, col = ifelse(outliers, "red", "blue"), main = "Numeric Columns Boxplot", names = c("GDP", "PH", "EEL", "UR", "P", "LE"))

# Add a legend

legend("topright", legend = c("Normal", "Outlier"), fill = c("blue", "red"))

par(mfrow = c(1, 1))

# Create a single boxplot with outliers highlighted in red

boxplot(numeric\_columns, col = ifelse(outliers, "red", "blue"),

main = "Numeric Columns Boxplot",

names = c("GDP", "PovertyHeadCount", "EducationEnrolment", "UnemploymentRate", "PopulationGrowthRate", "LifeExpectancy"))

# Add a legend with red color for outliers

legend("topright", legend = c("Normal", "Outlier"), fill = c("blue", "red"), border = "white")

# Numeric columns

numeric\_columns <- MISSING\_DATA[, c("GDP", "PovertyHeadCount", "EducationEnrolment", "UnemploymentRate", "PopulationGrowthRate", "LifeExpectancy")]

for (col\_name in numeric\_columns) {

col\_values <- MISSING\_DATA[[col\_name]]

# Calculate median and interquartile range

median\_val <- median(col\_values, na.rm = TRUE)

iqr <- IQR(col\_values, na.rm = TRUE)

# Define upper and lower bounds for outliers

upper\_bound <- median\_val + 1.5 \* iqr

lower\_bound <- median\_val - 1.5 \* iqr

# Identify and replace outliers

outliers <- col\_values > upper\_bound | col\_values < lower\_bound

if (any(!is.na(outliers) & outliers)) {

print(paste("Handling outliers in column:", col\_name))

# Print original median

print(paste("Original median:", median\_val))

# Replace outliers with median

MISSING\_DATA[outliers, col\_name] <- median\_val

# Print new median

print(paste("New median:", median(MISSING\_DATA[[col\_name]], na.rm = TRUE)))

}

}

print(MISSING\_DATA)

# Numeric columns

numeric\_columns <- c("GDP", "EducationEnrolment", "PovertyHeadCount", "UnemploymentRate", "PopulationGrowthRate", "LifeExpectancy")

for (col\_name in numeric\_columns) {

col\_values <- MISSING\_DATA[[col\_name]]

# Calculate median and interquartile range

median\_val <- median(col\_values, na.rm = TRUE)

iqr <- IQR(col\_values, na.rm = TRUE)

# Define upper and lower bounds for outliers

upper\_bound <- median\_val + 1.5 \* iqr

lower\_bound <- median\_val - 1.5 \* iqr

# Identify and replace outliers

outliers <- col\_values > upper\_bound | col\_values < lower\_bound

if (any(!is.na(outliers) & outliers)) {

print(paste("Handling outliers in column:", col\_name))

# Print original median

print(paste("Original median:", median\_val))

# Replace outliers with median

MISSING\_DATA[outliers, col\_name] <- as.numeric(median\_val)

# Print new median

print(paste("New median:", median(MISSING\_DATA[[col\_name]], na.rm = TRUE)))

} else {

print(paste("No outliers found in column:", col\_name))

}

}

# Calculate the mean of each variable

means <- colMeans(MISSING\_DATA[, 3:8], na.rm = TRUE)

MISSING\_DATA = as.data.frame(lapply(MISSING\_DATA, function(x) ifelse(is.na(x), means, x)))

missing\_counts <- colSums(is.na(MISSING\_DATA))

print(missing\_counts)

# Line plot for GDP over time

ggplot(MISSING\_DATA, aes(x = Year, y = GDP, group = Country, color = Country)) +

geom\_line() +

theme(plot.title = element\_text(hjust = 0.5))+

labs(title = "GDP Over Time",

x = "Year",

y = "GDP")+

theme(axis.text.x = element\_text(angle = 45, hjust = 1))

# Create a line plot for Life Expectancy over time

ggplot(MISSING\_DATA, aes(x = Year, y = LifeExpectancy,group = Country, color = Country)) +

geom\_line() +

theme(plot.title = element\_text(hjust = 0.5))+

labs(title = "Life Expectancy Over Time",

x = "Year",

y = "Life Expectancy")

# Create a scatterplot with correlation value

attach(data)

ggplot(MISSING\_DATA, aes(x = PovertyHeadCount, y = GDP))+

geom\_point() +

geom\_smooth(method = "lm", se = FALSE, color = "blue")+

annotate("text", x = max(data$PH), y = max(data$GDP),

label = paste("Correlation:", round(cor(data$PH, data$GDP), 2)),

hjust = 1, vjust = 1, size = 4, color = "blue") +

theme(plot.title = element\_text(hjust = 0.5))+

labs(title = "Scatterplot of Poverty Headcount vs. GDP",

x = "Poverty Headcount(% Population)",

y = "GDP(US $)")

# Create a bar plot for Population Growth Rate by Country

ggplot(MISSING\_DATA, aes(x = Country, y = PopulationGrowthRate, color = Country)) +

geom\_bar(stat = "identity") +

labs(title = "Population Growth Rate by Country",

x = "Country",

y = "Population Growth Rate") +

theme(axis.text.x = element\_text(angle = 45, hjust = 1))+

theme(plot.title = element\_text(hjust = 0.5))

# Create a line plot for Education Enrollment over time

MISSING\_DATA$Year <- as.factor(MISSING\_DATA$Year)

ggplot(MISSING\_DATA, aes(x = Year, y = EducationEnrolment,group = Country, color = Country)) +

geom\_line() +

theme(plot.title = element\_text(hjust = 0.5))+

labs(title = "Education Enrollment Over Time",

x = "Year",

y = "Education Enrollment")

# Custom mode function

calculate\_mode <- function(x) {

uniq\_x <- unique(x)

uniq\_x[which.max(tabulate(match(x, uniq\_x)))]

}

# Calculate summary statistics

summary\_stats <- data.frame(

Variable = colnames(data)[3:8],

Mean = sapply(data[, 3:8], mean, na.rm = TRUE),

Median = sapply(data[, 3:8], median, na.rm = TRUE),

Mode = sapply(data[, 3:8], calculate\_mode),

SD = sapply(data[, 3:8], sd, na.rm = TRUE),

Skewness = sapply(data[, 3:8], skew, na.rm = TRUE),

Kurtosis = sapply(data[, 3:8], kurtosi, na.rm = TRUE)

)

summary\_stats

# Calculate the correlation matrix

correlation\_matrix <- cor(MISSING\_DATA[,c("GDP", "PovertyHeadCount", "EducationEnrolment", "UnemploymentRate", "PopulationGrowthRate", "LifeExpectancy")])

correlation\_matrix

# Hypothesis 1

# Create two vectors for life expectancy (LE) based on PH groups

le\_below\_average <- MISSING\_DATA$LifeExpectancy[MISSING\_DATA$PovertyHeadCount < mean(MISSING\_DATA$PovertyHeadCount)]

le\_above\_average <- MISSING\_DATA$LifeExpectancy[MISSING\_DATA$PovertyHeadCount >= mean(MISSING\_DATA$PovertyHeadCount)]

# Perform the two-sample t-test

t\_test\_result <- t.test(le\_below\_average, le\_above\_average)

# Print the t-test result

print(t\_test\_result)

# Hypothesis 2

# Calculate the median GDP

mean\_gdp <- mean(MISSING\_DATA$GDP)

# Use the median as the threshold

threshold <- mean\_gdp

# Check the value of the threshold

threshold

# Hypothesis 2: Is there a significant difference in EEL between high GDP and low GDP countries?

# Null Hypothesis (H0): There is no significant difference.

# Alternative Hypothesis (H1): There is a significant difference.

# Subset the data into high GDP and low GDP countries based on a threshold

high\_gdp\_countries <- MISSING\_DATA[MISSING\_DATA$GDP >= threshold, ]

low\_gdp\_countries <- MISSING\_DATA[MISSING\_DATA$GDP < threshold, ]

# Perform a two-sample t-test to compare EEL between the two groups

t\_test\_result <- t.test(high\_gdp\_countries$EducationEnrolment, low\_gdp\_countries$EducationEnrolment)

# Display the results

t\_test\_result

# Create a matrix of independent variables

data = MISSING\_DATA

X <- as.matrix(data[, c("EducationEnrolment", "PovertyHeadCount", "UnemploymentRate", "PopulationGrowthRate", "LifeExpectancy")])

# Create a vector of the dependent variable

Y <- MISSING\_DATA$GDP

# Fit a Ridge Regression model

ridge\_model <- glmnet(X, Y, alpha=0) # Alpha=0 specifies Ridge Regression

# Plot the cross-validated mean squared error (MSE) as a function of lambda

plot(ridge\_model)

# Choose the lambda with the minimum cross-validated MSE

best\_lambda <- ridge\_model$lambda.min

# Refit the model with the best lambda

best\_ridge\_model <- glmnet(X, Y, alpha=0, lambda=best\_lambda)

# To get the coefficients

coefficients(best\_ridge\_model)

# Matrix setup: Create a matrix of independent variables (socio-economic indicators) and the dependent variable (GDP).

data = MISSING\_DATA

X <- as.matrix(data[, c("EducationEnrolment", "PovertyHeadCount", "UnemploymentRate", "PopulationGrowthRate", "LifeExpectancy")])

y <- data$GDP

# Lasso Model: Build a Lasso Regression model using the "glmnet" function.

lasso\_model <- glmnet(X, y, alpha = 1) # Setting alpha to 1 indicates Lasso regression.

# Cross-Validation: Perform k-fold cross-validation (e.g., 10-fold) to select the optimal lambda value.

cv\_model <- cv.glmnet(X, y, alpha = 1) # Setting alpha to 1 for Lasso regression.

# Find the optimal lambda value with minimum mean squared error (MSE):

optimal\_lambda <- cv\_model$lambda.min

# Fit Lasso Model: Fit the Lasso Regression model using the optimal lambda.

lasso\_fit <- glmnet(X, y, alpha = 1, lambda = optimal\_lambda)

# View Coefficients: Use the "coef" function to view the coefficients of the Lasso model.

lasso\_coefficients <- coef(lasso\_fit)

# Interpret Coefficients: Interpret the coefficients to understand which socio-economic indicators are most influential.

# Print the coefficients to view their values.

print(lasso\_coefficients)

#ARIMA MODEL

time\_series\_data <- ts(data$GDP, frequency = 1)

# Fit an ARIMA model

arima\_model <- auto.arima(time\_series\_data)

# Print the summary of the ARIMA model

summary(arima\_model)

# Plot the forecasts

plot(forecast(arima\_model))

#VAR

data$Year <- as.numeric(data$Year)

# Ensure the data is in a time series format (assuming yearly data)

ts\_data <- ts(data[, c("GDP", "PovertyHeadCount", "UnemploymentRate")], start = min(data$Year), frequency = 1) # Assuming yearly data

# Load the 'vars' package

library(vars)

# Fit the VAR model

var\_model <- VAR(ts\_data, p = 2)

# View the model summary

summary(var\_model)

#Exponential Smoothing

time\_series\_data <- ts(data$GDP, frequency = 1)

# Fit a Holt-Winters Exponential Smoothing model

hw\_model <- ets(time\_series\_data)

# Print the summary of the Holt-Winters model

summary(hw\_model)

# Plot the forecasts

plot(forecast(hw\_model))

# Numeric columns

numeric\_columns <- MISSING\_DATA[, c("GDP", "PovertyHeadCount", "EducationEnrolment", "UnemploymentRate", "PopulationGrowthRate", "LifeExpectancy")]

# Outlier Handling with Median

for (col\_name in numeric\_columns) {

if (col\_name %in% colnames(MISSING\_DATA)) {

col\_values <- MISSING\_DATA[, col\_name]

outliers <- abs(scale(col\_values)) > 3

if (any(outliers)) {

print(paste("Handling outliers in column:", col\_name))

# Print original median

print(paste("Original median:", median(col\_values, na.rm = TRUE)))

# Replace outliers with median

col\_values[outliers] <- median(col\_values, na.rm = TRUE)

MISSING\_DATA[, col\_name] <- col\_values

# Print new median

print(paste("New median:", median(col\_values, na.rm = TRUE)))

}

} else {

print(paste("Column not found:", col\_name))

}

}

# Print updated dataset

print("Updated Dataset:")

print(MISSING\_DATA)