**# Necessary libraries**

library(forecast)

library(zoo)

library(ggplot2)

library(dplyr)

library(tidyr)

library(knitr)

library(tibble)

library(tidyverse)

library(Metrics)

library(smooth)

library(randomForest)

library(FNN)

library(nnfor)

library(prophet)

library(caret)

library(GA)

library(reshape2)

library(xgboost)

**Analysis for Under-5 Female Mortality Rates in West Africa**

**# Linear Intrepolation**

data <- data.frame(

Year = 1980:2022,

GNQ = c(NA, NA, 191.36, 187.19, 183.29, 179.7, 176.62, 174.11, 172.12, 170.43, 168.83, 167.22, 165.83, 164.47, 163.09, 161.65, 159.78, 157.13, 154.19, 150.57, 146.7, 142.58, 138.38, 134.08, 129.77, 125.17, 120.68, 116.17, 111.74, 107.62, 103.71, 100.1, 96.602, 93.199, 89.845, 86.55, 83.577, 80.926, 77.911, 74.951, 72.399, 69.923, 67.394),

GNB = c(NA, NA, NA, NA, NA, 232.88, 228.23, 224.17, 219.73, 215.38, 210.8, 206.29, 201.92, 197.26, 192.53, 187.82, 182.91, 177.83, 172.73, 167.88, 162.6, 157.18, 151.7, 146.05, 140.18, 134.16, 128.17, 122.34, 116.66, 111.05, 105.44, 100.34, 95.704, 91.589, 87.829, 84.501, 81.364, 78.55, 75.712, 73.013, 70.49, 68.098, 65.769))

data$GNQ <- na.approx(data$GNQ, rule = 2)

data$GNB <- na.approx(data$GNB, rule = 2)

head(data)

# Define the data (a simplified version with the given table structure)

data <- list(

BEN = c(200.83, 196.89, 193.1, 189.49, 185.97, 182.54, 179.04, 175.49, 171.91, 168.35, 164.65, 160.95, 157.28, 153.78, 150.26, 146.84, 143.54, 140.36, 137.13, 133.54, 129.97, 126.4, 123.28, 120.26, 117.34, 114.49, 111.76, 109.19, 106.55, 104.02, 101.64, 99.275, 97.074, 94.852, 92.611, 90.421, 88.394, 86.269, 84.13, 81.838, 79.489, 77.019, 74.778),

BFA = c(228.65, 223.26, 218.56, 214.17, 210.15, 205.71, 201.18, 196.93, 193.52, 191.27, 190.12, 189.81, 189.83, 189.86, 189.34, 187.98, 185.44, 182.38, 178.86, 175.28, 171.47, 167.39, 162.76, 157.8, 152.28, 146.03, 139.58, 133.26, 127.09, 121.28, 115.96, 111.14, 106.71, 102.52, 98.779, 95.212, 91.827, 88.466, 85.28, 82.335, 79.251, 76.574, 73.887),

GIN = c(272.33, 267.33, 262.56, 258.23, 253.92, 249.54, 245, 240.39, 235.54, 230.39, 224.85, 218.72, 212.52, 205.69, 198.53, 191.36, 184.22, 177.38, 170.47, 163.59, 156.77, 150, 143.7, 137.87, 132.57, 128.04, 124.1, 120.64, 117.67, 115.07, 112.74, 110.77, 108.78, 106.8, 105.62, 103.83, 101.39, 99.583, 97.695, 95.579, 93.357, 90.876, 88.443),

GNB = c(232.88, 232.88, 232.88, 232.88, 232.88, 232.88, 228.23, 224.17, 219.73, 215.38, 210.8, 206.29, 201.92, 197.26, 192.53, 187.82, 182.91, 177.83, 172.73, 167.88, 162.6, 157.18, 151.7, 146.05, 140.18, 134.16, 128.17, 122.34, 116.66, 111.05, 105.44, 100.34, 95.704, 91.589, 87.829, 84.501, 81.364, 78.55, 75.712, 73.013, 70.49, 68.098, 65.769),

LBR = c(228.98, 226.3, 224.25, 223.28, 223.55, 225.45, 228.95, 234.01, 239.61, 245.06, 249.11, 251.1, 250.54, 247.22, 241.3, 233.24, 224, 213.9, 202.96, 190.85, 178.46, 165.74, 153.32, 141.34, 130.39, 120.84, 112.66, 106.02, 100.52, 96.11, 92.504, 89.514, 87.065, 84.947, 85.221, 81.5, 79.602, 77.82, 75.91, 73.811, 71.65, 69.445, 67.225),

MLI = c(292.95, 284.67, 276.5, 268.41, 260.53, 253.26, 246.33, 239.42, 232.86, 226.94, 221.73, 217.18, 213.02, 209.02, 205.78, 202.37, 198.87, 194.94, 190.54, 185.48, 179.95, 173.91, 167.91, 162.05, 156.47, 150.99, 145.78, 140.87, 136.17, 131.78, 127.59, 123.64, 119.82, 116.11, 112.72, 109.44, 106.41, 103.48, 100.51, 97.268, 94.331, 91.433, 88.429),

MRT = c(156.07, 151.38, 146.63, 141.46, 135.75, 129.43, 123.35, 117.88, 113.49, 110.31, 108.17, 106.82, 105.89, 105.12, 104.4, 103.38, 102.01, 100.2, 97.791, 94.58, 90.591, 86.048, 81.117, 75.847, 70.779, 66.011, 61.744, 58.259, 55.392, 53.096, 51.267, 49.739, 48.388, 47.076, 45.774, 44.408, 43.081, 41.683, 40.286, 38.994, 37.638, 36.412, 35.12),

CPV = c(74.54, 73.49, 72.86, 72.34, 71.42, 69.77, 67.15, 63.79, 60.32, 57.1, 54.53, 52.83, 51.95, 51.37, 50.61, 49.23, 46.77, 43.35, 39.4, 35.28, 31.59, 28.73, 26.78, 25.57, 24.98, 24.8, 24.82, 24.75, 24.48, 23.9, 22.99, 21.85, 20.57, 19.22, 17.85, 16.53, 15.35, 14.31, 13.44, 12.73, 12.11, 11.6, 11.13),

TCD = c(229.48, 227.27, 224.63, 222.03, 219.39, 216.62, 213.6, 210.6, 207.56, 204.64, 201.84, 198.99, 196.39, 193.96, 191.55, 189.37, 187.19, 184.47, 181.58, 178.3, 175.03, 171.55, 168.08, 164.61, 161.35, 157.94, 154.61, 150.83, 147.11, 143.2, 139.23, 135.35, 131.58, 127.95, 124.45, 120.81, 117.29, 113.58, 109.91, 106.2, 102.85, 99.404, 96.231),

CIV = c(155.87, 151.78, 148.33, 145.5, 143.07, 141.61, 140.46, 139.78, 139.56, 139.93, 140.24, 140.86, 141.3, 141.38, 140.91, 140.06, 138.8, 136.99, 134.62, 131.83, 128.62, 125.27, 121.66, 117.97, 114.13, 110.09, 106.37, 102.54, 98.637, 94.707, 91.419, 88.443, 85.677, 82.741, 79.602, 76.84, 74.63, 72.457, 70.049, 67.555, 65.409, 63.433, 61.482),

GNQ = c(191.36, 191.36, 191.36, 187.19, 183.29, 179.7, 176.62, 174.11, 172.12, 170.43, 168.83, 167.22, 165.83, 164.47, 163.09, 161.65, 159.78, 157.13, 154.19, 150.57, 146.7, 142.58, 138.38, 134.08, 129.77, 125.17, 120.68, 116.17, 111.74, 107.62, 103.71, 100.1, 96.602, 93.199, 89.845, 86.55, 83.577, 80.926, 77.911, 74.951, 72.399, 69.923, 67.394),

GMB = c(218.69, 212.42, 206.11, 199.41, 192.9, 186.34, 179.98, 173.64, 167.69, 161.94, 156.34, 150.94, 145.7, 140.5, 135.35, 130.14, 124.96, 119.99, 115.04, 110.19, 105.7, 101.22, 96.831, 92.538, 88.442, 84.432, 80.626, 77.009, 73.554, 70.314, 67.256, 64.272, 61.5, 58.867, 56.351, 54.003, 51.65, 49.518, 47.552, 45.723, 43.944, 42.364, 40.798),

GHA = c(158.81, 157.23, 155.8, 153.64, 150.58, 146.43, 141.49, 135.85, 130.02, 124.41, 119.45, 115.21, 111.94, 109.53, 107.86, 106.25, 104.47, 102.19, 99.322, 95.799, 92.031, 88.179, 84.433, 81.158, 78.287, 75.786, 73.547, 71.238, 68.635, 65.89, 63.004, 60.024, 57.1, 54.417, 51.989, 49.672, 47.576, 45.608, 43.844, 42.083, 40.495, 39.113, 37.769),

NER = c(321.21, 320.77, 322.92, 326.94, 331.58, 335.68, 338.76, 339.5, 338.37, 335.45, 330.47, 322.71, 313.64, 302.56, 290.4, 277.6, 264.84, 253.29, 242.87, 233.92, 225.05, 215.12, 203.83, 192.02, 180.27, 169.16, 159.39, 150.85, 143.64, 137.94, 133.42, 130, 127.57, 125.84, 124.43, 123.19, 122.05, 120.94, 119.92, 118.8, 117.53, 115.94, 113.83),

NGA = c(202.31, 199.33, 197.51, 196.55, 196.41, 196.88, 197.45, 197.92, 198.53, 198.59, 198.28, 197.85, 197.5, 197, 195.8, 194.02, 191.49, 188.05, 183.46, 178.64, 173.65, 168.12, 162.43, 156.65, 151.03, 145.78, 140.8, 136.74, 133.26, 130.13, 127.6, 125.38, 123.53, 121.83, 120.45, 119.25, 117.81, 115.79, 113.29, 110.33, 107.47, 104.33, 100.91),

SEN = c(197.36, 190.43, 184.17, 177.74, 170.74, 163.33, 155.23, 147.36, 140.37, 134.91, 131.26, 129.31, 128.79, 129.38, 130.47, 131.39, 132.03, 131.85, 130.41, 127.49, 122.74, 116.22, 108.51, 100.58, 92.856, 85.743, 79.37, 73.77, 68.913, 64.721, 61.043, 57.681, 54.575, 51.641, 48.809, 46.187, 43.699, 41.385, 39.284, 37.453, 35.79, 34.366, 32.959),

SLE = c(275.31, 271.84, 268.41, 265.3, 262.37, 259.63, 256.86, 254.35, 252.02, 249.68, 247.61, 245.36, 243.02, 240.73, 238.34, 235.5, 232.3, 228.46, 224.21, 219.93, 214.74, 209.84, 204.54, 199.09, 193.47, 187.35, 180.98, 174.12, 167.04, 160.13, 152.99, 146.07, 139.71, 133.64, 131.71, 132.69, 117.94, 113.38, 109, 104.89, 100.98, 97.39, 93.834),

TGO = c(170.07, 166.02, 162.1, 158.4, 154.8, 151.46, 148.31, 145.36, 142.76, 140.28, 138.07, 136.15, 134.07, 131.83, 129.48, 126.83, 123.89, 120.9, 117.78, 114.41, 110.97, 107.7, 104.53, 101.47, 98.521, 95.734, 93.083, 90.405, 87.752, 85.191, 82.71, 80.095, 77.449, 74.901, 72.412, 70.036, 67.809, 65.551, 63.374, 61.207, 59.207, 57.243, 55.347))

**# Prepare the Data for Plotting**

years <- 1980:2022

plot\_data <- data %>%

as.data.frame() %>%

mutate(Year = years) %>%

pivot\_longer(-Year, names\_to = "Country", values\_to = "MortalityRate")

ggplot(plot\_data, aes(x = Year, y = MortalityRate, color = Country)) +geom\_line(size = 1) +theme\_minimal() +labs(title = "Under-5 Female Mortality Rate",x = "Year",y = "Mortality Rate (per 1000 live births)",color = "Country") +theme(plot.title = element\_text(hjust = 0.5, size = 10, face = "bold"),axis.title = element\_text(size = 10),axis.text = element\_text(size = 10),legend.position = "right")

**# Calculat the Means and Standard Deviations**

means <- sapply(data, mean)

sds <- sapply(data, sd)

results <- data.frame(Country = names(data), Mean = means, SD = sds)

print(results)

**# K-means Clustering**

set.seed(123)

data\_df <- as.data.frame(data)

data\_df$Year <- seq(1980, 2022)

data\_long <- melt(data\_df, id.vars = "Year", variable.name = "Country", value.name = "Value")

transposed\_data <- as.data.frame(t(data\_df[,-ncol(data\_df)]))

rownames(transposed\_data) <- colnames(data\_df)[-ncol(data\_df)]

colnames(transposed\_data) <- seq(1980, 2022)

transposed\_data\_scaled <- scale(transposed\_data)

k <- 4 # Number of clusters

kmeans\_result <- kmeans(transposed\_data\_scaled, centers = k)

cluster\_assignments <- data.frame(Country = rownames(transposed\_data), Cluster = kmeans\_result$cluster)

print(cluster\_assignments)

data\_long <- left\_join(data\_long, cluster\_assignments, by = "Country")

ggplot(data\_long, aes(x = Year, y = Value, color = factor(Cluster), group = Country)) +geom\_line() +labs(title = "K-means Clustering of Under-5 Female Mortality",x = "Year",y = "Mortality Rate",color = "Cluster") +theme\_minimal() +theme(plot.title = element\_text(hjust = 0.5, size = 10, face = "bold"),axis.title = element\_text(size = 10),axis.text = element\_text(size = 10),legend.position = "right")

**# Resilience index for each country**

set.seed(123)

calculate\_resilience <- function(values) {

start <- values[1]

end <- values[length(values)]

(start - end) / start \* 100}

resilience\_index <- sapply(data, calculate\_resilience)

ranks <- rank(-resilience\_index, ties.method = "min")

resilience\_df <- data.frame(Country = names(resilience\_index),Resilience\_Index = resilience\_index,Rank = ranks)

results <- resilience\_df[order(results$Rank), ]

print(results)

ggplot(resilience\_df, aes(x = reorder(Country, Resilience\_Index), y = Resilience\_Index)) +

geom\_bar(stat = "identity", fill = "steelblue") +

coord\_flip() +

labs(title = "Resilience Index of Female U5MR by Country", x = "Country", y = "Resilience Index (%)") +

theme\_minimal()

**# Out-of-Sample Forecast for BEN UN-5 Female Mortality Rates**

set.seed(123)

BEN <- c(200.83, 196.89, 193.1, 189.49, 185.97, 182.54, 179.04, 175.49, 171.91, 168.35, 164.65, 160.95, 157.28, 153.78, 150.26, 146.84, 143.54, 140.36, 137.13, 133.54, 129.97, 126.4, 123.28, 120.26, 117.34, 114.49, 111.76, 109.19, 106.55, 104.02, 101.64, 99.275, 97.074, 94.852, 92.611, 90.421, 88.394, 86.269, 84.13, 81.838, 79.489, 77.019, 74.778)

ben\_ts <- ts(BEN, start = 1980, frequency = 1)

training\_data <- window(ben\_ts, start=1980, end=2017)

testing\_data <- window(ben\_ts, start=2018, end=2022)

arima\_model <- auto.arima(training\_data)

arima\_forecast <- forecast(arima\_model, h = length(testing\_data))$mean

ets\_model <- ets(training\_data)

ets\_forecast <- forecast(ets\_model, h = length(testing\_data))$mean

mlp\_model <- mlp(training\_data)

mlp\_forecast <- forecast(mlp\_model, h = length(testing\_data))$mean

train\_df <- data.frame(

ds = seq.Date(from = as.Date("1980-01-01"), by = "year", length.out = length(training\_data)),

y = as.numeric(training\_data))

prophet\_model <- prophet(train\_df, yearly.seasonality = FALSE, weekly.seasonality = FALSE, daily.seasonality = FALSE)

future <- data.frame(ds = seq.Date(from = max(train\_df$ds) + 1, by = "year", length.out = length(testing\_data)))

prophet\_forecast <- predict(prophet\_model, future)$yhat

lagged\_data <- data.frame(Target = training\_data[-1],Lag1 = training\_data[-length(training\_data)])

xgb\_model <- xgboost(data = as.matrix(lagged\_data$Lag1), label = lagged\_data$Target, nrounds = 100)

xgb\_forecast <- numeric(length(testing\_data))

last\_value <- training\_data[length(training\_data)]

for (i in 1:length(testing\_data)) {

xgb\_forecast[i] <- predict(xgb\_model, newdata = as.matrix(last\_value))

last\_value <- xgb\_forecast[i]}

hybrid\_forecast\_function <- function(weights, f1, f2) {

weights[1] \* as.numeric(f1) + weights[2] \* as.numeric(f2)}

fitness\_function <- function(weights, f1, f2, actual) {

hybrid\_forecast <- hybrid\_forecast\_function(weights, f1, f2)

sqrt(mean((actual - hybrid\_forecast)^2))}

optimize\_weights <- function(f1, f2, actual) {

ga\_model <- ga(type = "real-valued",fitness = function(w) -fitness\_function(w, f1, f2, actual),lower = c(0, 0),upper = c(1, 1),popSize = 50,maxiter = 100)

return(ga\_model@solution)}

weights <- optimize\_weights(arima\_forecast, ets\_forecast, testing\_data)

arima\_ets\_forecast <- hybrid\_forecast\_function(weights, arima\_forecast, ets\_forecast)

weights <- optimize\_weights(arima\_forecast, mlp\_forecast, testing\_data)

arima\_mlp\_forecast <- hybrid\_forecast\_function(weights, arima\_forecast, mlp\_forecast)

weights <- optimize\_weights(arima\_forecast, xgb\_forecast, testing\_data)

arima\_xgb\_forecast <- hybrid\_forecast\_function(weights, arima\_forecast, xgb\_forecast)

weights <- optimize\_weights(arima\_forecast, prophet\_forecast, testing\_data)

arima\_prophet\_forecast <- hybrid\_forecast\_function(weights, arima\_forecast, prophet\_forecast)

calculate\_metrics <- function(actual, forecast) {

rmse <- sqrt(mean((actual - forecast)^2))

mape <- mean(abs((actual - forecast) / actual)) \* 100

return(c(RMSE = rmse, MAPE = mape))}

model\_metrics <- data.frame(

Model = c("ARIMA", "ETS", "MLP", "Prophet", "XGBoost", "ARIMA\_ETS", "ARIMA\_MLP", "ARIMA\_XGBoost", "ARIMA\_Prophet"),

do.call(rbind, list(

calculate\_metrics(testing\_data, arima\_forecast),

calculate\_metrics(testing\_data, ets\_forecast),

calculate\_metrics(testing\_data, mlp\_forecast),

calculate\_metrics(testing\_data, prophet\_forecast),

calculate\_metrics(testing\_data, xgb\_forecast),

calculate\_metrics(testing\_data, arima\_ets\_forecast),

calculate\_metrics(testing\_data, arima\_mlp\_forecast),

calculate\_metrics(testing\_data, arima\_xgb\_forecast),

calculate\_metrics(testing\_data, arima\_prophet\_forecast))))

print("Model Performance Metrics (RMSE and MAE):")

print(model\_metrics)

all\_forecasts <- data.frame(

Year = seq(2018, 2022),

Actual = as.numeric(testing\_data),

ARIMA = as.numeric(arima\_forecast),

ETS = as.numeric(ets\_forecast),

MLP = as.numeric(mlp\_forecast),

Prophet = as.numeric(prophet\_forecast),

XGBoost = as.numeric(xgb\_forecast),

ARIMA\_ETS = as.numeric(arima\_ets\_forecast),

ARIMA\_MLP = as.numeric(arima\_mlp\_forecast),

ARIMA\_Prophet = as.numeric(arima\_prophet\_forecast),

ARIMA\_XGBoost = as.numeric(arima\_xgb\_forecast))

melted\_forecasts <- melt(all\_forecasts, id.vars = "Year", variable.name = "Model", value.name = "Value")

ggplot(data = melted\_forecasts, aes(x = Year, y = Value, color = Model)) +

geom\_line() +

labs(title = "BEN", y = "Mortality Rate", x = "Year") +

theme\_minimal()

**# Out-of-Sample Forecast for BFA UN-5 Female Mortality Rates**

BFA = c(228.65, 223.26, 218.56, 214.17, 210.15, 205.71, 201.18, 196.93, 193.52, 191.27, 190.12, 189.81, 189.83, 189.86, 189.34, 187.98, 185.44, 182.38, 178.86, 175.28, 171.47, 167.39, 162.76, 157.8, 152.28, 146.03, 139.58, 133.26, 127.09, 121.28, 115.96, 111.14, 106.71, 102.52, 98.779, 95.212, 91.827, 88.466, 85.28, 82.335, 79.251, 76.574, 73.887)

set.seed(123)

bfa\_ts <- ts(BFA, start = 1980, frequency = 1)

training\_data <- window(bfa\_ts, start=1980, end=2017)

testing\_data <- window(bfa\_ts, start=2018, end=2022)

arima\_model <- auto.arima(training\_data)

arima\_forecast <- forecast(arima\_model, h = length(testing\_data))$mean

ets\_model <- ets(training\_data)

ets\_forecast <- forecast(ets\_model, h = length(testing\_data))$mean

mlp\_model <- mlp(training\_data)

mlp\_forecast <- forecast(mlp\_model, h = length(testing\_data))$mean

train\_df <- data.frame(

ds = seq.Date(from = as.Date("1980-01-01"), by = "year", length.out = length(training\_data)),

y = as.numeric(training\_data))

prophet\_model <- prophet(train\_df, yearly.seasonality = FALSE, weekly.seasonality = FALSE, daily.seasonality = FALSE)

future <- data.frame(ds = seq.Date(from = max(train\_df$ds) + 1, by = "year", length.out = length(testing\_data)))

prophet\_forecast <- predict(prophet\_model, future)$yhat

lagged\_data <- data.frame(

Target = training\_data[-1],

Lag1 = training\_data[-length(training\_data)])

xgb\_model <- xgboost(data = as.matrix(lagged\_data$Lag1), label = lagged\_data$Target, nrounds = 100)

xgb\_forecast <- numeric(length(testing\_data))

last\_value <- training\_data[length(training\_data)]

for (i in 1:length(testing\_data)) {

xgb\_forecast[i] <- predict(xgb\_model, newdata = as.matrix(last\_value))

last\_value <- xgb\_forecast[i]}

hybrid\_forecast\_function <- function(weights, f1, f2) {

weights[1] \* as.numeric(f1) + weights[2] \* as.numeric(f2)}

fitness\_function <- function(weights, f1, f2, actual) {

hybrid\_forecast <- hybrid\_forecast\_function(weights, f1, f2)

sqrt(mean((actual - hybrid\_forecast)^2))}

optimize\_weights <- function(f1, f2, actual) {

ga\_model <- ga(

type = "real-valued",

fitness = function(w) -fitness\_function(w, f1, f2, actual),

lower = c(0, 0),

upper = c(1, 1),

popSize = 50,

maxiter = 100)

return(ga\_model@solution)}

weights <- optimize\_weights(arima\_forecast, ets\_forecast, testing\_data)

arima\_ets\_forecast <- hybrid\_forecast\_function(weights, arima\_forecast, ets\_forecast)

weights <- optimize\_weights(arima\_forecast, mlp\_forecast, testing\_data)

arima\_mlp\_forecast <- hybrid\_forecast\_function(weights, arima\_forecast, mlp\_forecast)

weights <- optimize\_weights(arima\_forecast, xgb\_forecast, testing\_data)

arima\_xgb\_forecast <- hybrid\_forecast\_function(weights, arima\_forecast, xgb\_forecast)

weights <- optimize\_weights(arima\_forecast, prophet\_forecast, testing\_data)

arima\_prophet\_forecast <- hybrid\_forecast\_function(weights, arima\_forecast, prophet\_forecast)

calculate\_metrics <- function(actual, forecast) {

rmse <- sqrt(mean((actual - forecast)^2))

mape <- mean(abs((actual - forecast) / actual)) \* 100

return(c(RMSE = rmse, MAPE = mape))}

model\_metrics <- data.frame(

Model = c("ARIMA", "ETS", "MLP", "Prophet", "XGBoost", "ARIMA\_ETS", "ARIMA\_MLP", "ARIMA\_XGBoost", "ARIMA\_Prophet"),

do.call(rbind, list(

calculate\_metrics(testing\_data, arima\_forecast),

calculate\_metrics(testing\_data, ets\_forecast),

calculate\_metrics(testing\_data, mlp\_forecast),

calculate\_metrics(testing\_data, prophet\_forecast),

calculate\_metrics(testing\_data, xgb\_forecast),

calculate\_metrics(testing\_data, arima\_ets\_forecast),

calculate\_metrics(testing\_data, arima\_mlp\_forecast),

calculate\_metrics(testing\_data, arima\_xgb\_forecast),

calculate\_metrics(testing\_data, arima\_prophet\_forecast))))

print("Model Performance Metrics (RMSE and MAE):")

print(model\_metrics)

all\_forecasts <- data.frame(

Year = seq(2018, 2022),

Actual = as.numeric(testing\_data),

ARIMA = as.numeric(arima\_forecast),

ETS = as.numeric(ets\_forecast),

MLP = as.numeric(mlp\_forecast),

Prophet = as.numeric(prophet\_forecast),

XGBoost = as.numeric(xgb\_forecast),

ARIMA\_ETS = as.numeric(arima\_ets\_forecast),

ARIMA\_MLP = as.numeric(arima\_mlp\_forecast),

ARIMA\_Prophet = as.numeric(arima\_prophet\_forecast),

ARIMA\_XGBoost = as.numeric(arima\_xgb\_forecast))

melted\_forecasts <- melt(all\_forecasts, id.vars = "Year", variable.name = "Model", value.name = "Value")

ggplot(data = melted\_forecasts, aes(x = Year, y = Value, color = Model)) +geom\_line() +labs(title = "BFA", y = "Mortality Rate", x = "Year") +theme\_minimal()

**# Out-of-Sample Forecast for GIN UN-5 Female Mortality Rates**

set.seed(123)

GIN <- c(272.33, 267.33, 262.56, 258.23, 253.92, 249.54, 245, 240.39, 235.54, 230.39, 224.85, 218.72, 212.52, 205.69, 198.53, 191.36, 184.22, 177.38, 170.47, 163.59, 156.77, 150, 143.7, 137.87, 132.57, 128.04, 124.1, 120.64, 117.67, 115.07, 112.74, 110.77, 108.78, 106.8, 105.62, 103.83, 101.39, 99.583, 97.695, 95.579, 93.357, 90.876, 88.443)

gin\_ts <- ts(GIN, start = 1980, frequency = 1)

training\_data <- window(gin\_ts, start=1980, end=2017)

testing\_data <- window(gin\_ts, start=2018, end=2022)

arima\_model <- auto.arima(training\_data)

arima\_forecast <- forecast(arima\_model, h = length(testing\_data))$mean

ets\_model <- ets(training\_data)

ets\_forecast <- forecast(ets\_model, h = length(testing\_data))$mean

mlp\_model <- mlp(training\_data)

mlp\_forecast <- forecast(mlp\_model, h = length(testing\_data))$mean

train\_df <- data.frame(

ds = seq.Date(from = as.Date("1980-01-01"), by = "year", length.out = length(training\_data)),

y = as.numeric(training\_data))

prophet\_model <- prophet(train\_df, yearly.seasonality = FALSE, weekly.seasonality = FALSE, daily.seasonality = FALSE)

future <- data.frame(ds = seq.Date(from = max(train\_df$ds) + 1, by = "year", length.out = length(testing\_data)))

prophet\_forecast <- predict(prophet\_model, future)$yhat

lagged\_data <- data.frame(

Target = training\_data[-1],

Lag1 = training\_data[-length(training\_data)])

xgb\_model <- xgboost(data = as.matrix(lagged\_data$Lag1), label = lagged\_data$Target, nrounds = 100)

xgb\_forecast <- numeric(length(testing\_data))

last\_value <- training\_data[length(training\_data)]

for (i in 1:length(testing\_data)) {

xgb\_forecast[i] <- predict(xgb\_model, newdata = as.matrix(last\_value))

last\_value <- xgb\_forecast[i]}

hybrid\_forecast\_function <- function(weights, f1, f2) {

weights[1] \* as.numeric(f1) + weights[2] \* as.numeric(f2)}

fitness\_function <- function(weights, f1, f2, actual) {

hybrid\_forecast <- hybrid\_forecast\_function(weights, f1, f2)

sqrt(mean((actual - hybrid\_forecast)^2))}

optimize\_weights <- function(f1, f2, actual) {

ga\_model <- ga(

type = "real-valued",

fitness = function(w) -fitness\_function(w, f1, f2, actual),

lower = c(0, 0),

upper = c(1, 1),

popSize = 50,

maxiter = 100)

return(ga\_model@solution)}

weights <- optimize\_weights(arima\_forecast, ets\_forecast, testing\_data)

arima\_ets\_forecast <- hybrid\_forecast\_function(weights, arima\_forecast, ets\_forecast)

weights <- optimize\_weights(arima\_forecast, mlp\_forecast, testing\_data)

arima\_mlp\_forecast <- hybrid\_forecast\_function(weights, arima\_forecast, mlp\_forecast)

weights <- optimize\_weights(arima\_forecast, xgb\_forecast, testing\_data)

arima\_xgb\_forecast <- hybrid\_forecast\_function(weights, arima\_forecast, xgb\_forecast)

weights <- optimize\_weights(arima\_forecast, prophet\_forecast, testing\_data)

arima\_prophet\_forecast <- hybrid\_forecast\_function(weights, arima\_forecast, prophet\_forecast)

calculate\_metrics <- function(actual, forecast) {

rmse <- sqrt(mean((actual - forecast)^2))

mape <- mean(abs((actual - forecast) / actual)) \* 100

return(c(RMSE = rmse, MAPE = mape))}

model\_metrics <- data.frame(

Model = c("ARIMA", "ETS", "MLP", "Prophet", "XGBoost", "ARIMA\_ETS", "ARIMA\_MLP", "ARIMA\_XGBoost", "ARIMA\_Prophet"),

do.call(rbind, list(

calculate\_metrics(testing\_data, arima\_forecast),

calculate\_metrics(testing\_data, ets\_forecast),

calculate\_metrics(testing\_data, mlp\_forecast),

calculate\_metrics(testing\_data, prophet\_forecast),

calculate\_metrics(testing\_data, xgb\_forecast),

calculate\_metrics(testing\_data, arima\_ets\_forecast),

calculate\_metrics(testing\_data, arima\_mlp\_forecast),

calculate\_metrics(testing\_data, arima\_xgb\_forecast),

calculate\_metrics(testing\_data, arima\_prophet\_forecast))))

print("Model Performance Metrics (RMSE and MAE):")

print(model\_metrics)

all\_forecasts <- data.frame(

Year = seq(2018, 2022),

Actual = as.numeric(testing\_data),

ARIMA = as.numeric(arima\_forecast),

ETS = as.numeric(ets\_forecast),

MLP = as.numeric(mlp\_forecast),

Prophet = as.numeric(prophet\_forecast),

XGBoost = as.numeric(xgb\_forecast),

ARIMA\_ETS = as.numeric(arima\_ets\_forecast),

ARIMA\_MLP = as.numeric(arima\_mlp\_forecast),

ARIMA\_Prophet = as.numeric(arima\_prophet\_forecast),

ARIMA\_XGBoost = as.numeric(arima\_xgb\_forecast))

melted\_forecasts <- melt(all\_forecasts, id.vars = "Year", variable.name = "Model", value.name = "Value")

ggplot(data = melted\_forecasts, aes(x = Year, y = Value, color = Model)) +

geom\_line() +

labs(title = "GIN", y = "Mortality Rate", x = "Year") +

theme\_minimal()

**# Out-of-Sample Forecast for GNB UN-5 Female Mortality Rates**

set.seed(123)

GNB <- c(232.88, 232.88, 232.88, 232.88, 232.88, 232.88, 228.23, 224.17, 219.73, 215.38, 210.8, 206.29, 201.92, 197.26, 192.53, 187.82, 182.91, 177.83, 172.73, 167.88, 162.6, 157.18, 151.7, 146.05, 140.18, 134.16, 128.17, 122.34, 116.66, 111.05, 105.44, 100.34, 95.704, 91.589, 87.829, 84.501, 81.364, 78.55, 75.712, 73.013, 70.49, 68.098, 65.769)

gnb\_ts <- ts(GNB, start = 1980, frequency = 1)

training\_data <- window(gnb\_ts, start=1980, end=2017)

testing\_data <- window(gnb\_ts, start=2018, end=2022)

arima\_model <- auto.arima(training\_data)

arima\_forecast <- forecast(arima\_model, h = length(testing\_data))$mean

ets\_model <- ets(training\_data)

ets\_forecast <- forecast(ets\_model, h = length(testing\_data))$mean

mlp\_model <- mlp(training\_data)

mlp\_forecast <- forecast(mlp\_model, h = length(testing\_data))$mean

train\_df <- data.frame(

ds = seq.Date(from = as.Date("1980-01-01"), by = "year", length.out = length(training\_data)),

y = as.numeric(training\_data))

prophet\_model <- prophet(train\_df, yearly.seasonality = FALSE, weekly.seasonality = FALSE, daily.seasonality = FALSE)

future <- data.frame(ds = seq.Date(from = max(train\_df$ds) + 1, by = "year", length.out = length(testing\_data)))

prophet\_forecast <- predict(prophet\_model, future)$yhat

lagged\_data <- data.frame(

Target = training\_data[-1],

Lag1 = training\_data[-length(training\_data)])

xgb\_model <- xgboost(data = as.matrix(lagged\_data$Lag1), label = lagged\_data$Target, nrounds = 100)

xgb\_forecast <- numeric(length(testing\_data))

last\_value <- training\_data[length(training\_data)]

for (i in 1:length(testing\_data)) {

xgb\_forecast[i] <- predict(xgb\_model, newdata = as.matrix(last\_value))

last\_value <- xgb\_forecast[i]}

hybrid\_forecast\_function <- function(weights, f1, f2) {

weights[1] \* as.numeric(f1) + weights[2] \* as.numeric(f2)}

fitness\_function <- function(weights, f1, f2, actual) {

hybrid\_forecast <- hybrid\_forecast\_function(weights, f1, f2)

sqrt(mean((actual - hybrid\_forecast)^2))}

optimize\_weights <- function(f1, f2, actual) {

ga\_model <- ga(

type = "real-valued",

fitness = function(w) -fitness\_function(w, f1, f2, actual),

lower = c(0, 0),

upper = c(1, 1),

popSize = 50,

maxiter = 100)

return(ga\_model@solution)}

weights <- optimize\_weights(arima\_forecast, ets\_forecast, testing\_data)

arima\_ets\_forecast <- hybrid\_forecast\_function(weights, arima\_forecast, ets\_forecast)

weights <- optimize\_weights(arima\_forecast, mlp\_forecast, testing\_data)

arima\_mlp\_forecast <- hybrid\_forecast\_function(weights, arima\_forecast, mlp\_forecast)

weights <- optimize\_weights(arima\_forecast, xgb\_forecast, testing\_data)

arima\_xgb\_forecast <- hybrid\_forecast\_function(weights, arima\_forecast, xgb\_forecast)

weights <- optimize\_weights(arima\_forecast, prophet\_forecast, testing\_data)

arima\_prophet\_forecast <- hybrid\_forecast\_function(weights, arima\_forecast, prophet\_forecast)

calculate\_metrics <- function(actual, forecast) {

rmse <- sqrt(mean((actual - forecast)^2))

mape <- mean(abs((actual - forecast) / actual)) \* 100

return(c(RMSE = rmse, MAPE = mape))}

model\_metrics <- data.frame(

Model = c("ARIMA", "ETS", "MLP", "Prophet", "XGBoost", "ARIMA\_ETS", "ARIMA\_MLP", "ARIMA\_XGBoost", "ARIMA\_Prophet"),

do.call(rbind, list(

calculate\_metrics(testing\_data, arima\_forecast),

calculate\_metrics(testing\_data, ets\_forecast),

calculate\_metrics(testing\_data, mlp\_forecast),

calculate\_metrics(testing\_data, prophet\_forecast),

calculate\_metrics(testing\_data, xgb\_forecast),

calculate\_metrics(testing\_data, arima\_ets\_forecast),

calculate\_metrics(testing\_data, arima\_mlp\_forecast),

calculate\_metrics(testing\_data, arima\_xgb\_forecast),

calculate\_metrics(testing\_data, arima\_prophet\_forecast))))

print("Model Performance Metrics (RMSE and MAE):")

print(model\_metrics)

all\_forecasts <- data.frame(

Year = seq(2018, 2022),

Actual = as.numeric(testing\_data),

ARIMA = as.numeric(arima\_forecast),

ETS = as.numeric(ets\_forecast),

MLP = as.numeric(mlp\_forecast),

Prophet = as.numeric(prophet\_forecast),

XGBoost = as.numeric(xgb\_forecast),

ARIMA\_ETS = as.numeric(arima\_ets\_forecast),

ARIMA\_MLP = as.numeric(arima\_mlp\_forecast),

ARIMA\_Prophet = as.numeric(arima\_prophet\_forecast),

ARIMA\_XGBoost = as.numeric(arima\_xgb\_forecast))

melted\_forecasts <- melt(all\_forecasts, id.vars = "Year", variable.name = "Model", value.name = "Value")

ggplot(data = melted\_forecasts, aes(x = Year, y = Value, color = Model)) +

geom\_line() +

labs(title = "GNB", y = "Mortality Rate", x = "Year") +

theme\_minimal()

**# Out-of-Sample Forecast for LBR UN-5 Female Mortality Rates**

set.seed(123)

LBR <- c(228.98, 226.3, 224.25, 223.28, 223.55, 225.45, 228.95, 234.01, 239.61, 245.06, 249.11, 251.1, 250.54, 247.22, 241.3, 233.24, 224, 213.9, 202.96, 190.85, 178.46, 165.74, 153.32, 141.34, 130.39, 120.84, 112.66, 106.02, 100.52, 96.11, 92.504, 89.514, 87.065, 84.947, 85.221, 81.5, 79.602, 77.82, 75.91, 73.811, 71.65, 69.445, 67.225)

lbr\_ts <- ts(LBR, start = 1980, frequency = 1)

training\_data <- window(lbr\_ts, start=1980, end=2017)

testing\_data <- window(lbr\_ts, start=2018, end=2022)

arima\_model <- auto.arima(training\_data)

arima\_forecast <- forecast(arima\_model, h = length(testing\_data))$mean

ets\_model <- ets(training\_data)

ets\_forecast <- forecast(ets\_model, h = length(testing\_data))$mean

mlp\_model <- mlp(training\_data)

mlp\_forecast <- forecast(mlp\_model, h = length(testing\_data))$mean

train\_df <- data.frame(

ds = seq.Date(from = as.Date("1980-01-01"), by = "year", length.out = length(training\_data)),

y = as.numeric(training\_data))

prophet\_model <- prophet(train\_df, yearly.seasonality = FALSE, weekly.seasonality = FALSE, daily.seasonality = FALSE)

future <- data.frame(ds = seq.Date(from = max(train\_df$ds) + 1, by = "year", length.out = length(testing\_data)))

prophet\_forecast <- predict(prophet\_model, future)$yhat

lagged\_data <- data.frame(

Target = training\_data[-1],

Lag1 = training\_data[-length(training\_data)])

xgb\_model <- xgboost(data = as.matrix(lagged\_data$Lag1), label = lagged\_data$Target, nrounds = 100)

xgb\_forecast <- numeric(length(testing\_data))

last\_value <- training\_data[length(training\_data)]

for (i in 1:length(testing\_data)) {

xgb\_forecast[i] <- predict(xgb\_model, newdata = as.matrix(last\_value))

last\_value <- xgb\_forecast[i]}

hybrid\_forecast\_function <- function(weights, f1, f2) {

weights[1] \* as.numeric(f1) + weights[2] \* as.numeric(f2)}

fitness\_function <- function(weights, f1, f2, actual) {

hybrid\_forecast <- hybrid\_forecast\_function(weights, f1, f2)

sqrt(mean((actual - hybrid\_forecast)^2))}

optimize\_weights <- function(f1, f2, actual) {

ga\_model <- ga(

type = "real-valued",

fitness = function(w) -fitness\_function(w, f1, f2, actual),

lower = c(0, 0),

upper = c(1, 1),

popSize = 50,

maxiter = 100)

return(ga\_model@solution)}

weights <- optimize\_weights(arima\_forecast, ets\_forecast, testing\_data)

arima\_ets\_forecast <- hybrid\_forecast\_function(weights, arima\_forecast, ets\_forecast)

weights <- optimize\_weights(arima\_forecast, mlp\_forecast, testing\_data)

arima\_mlp\_forecast <- hybrid\_forecast\_function(weights, arima\_forecast, mlp\_forecast)

weights <- optimize\_weights(arima\_forecast, xgb\_forecast, testing\_data)

arima\_xgb\_forecast <- hybrid\_forecast\_function(weights, arima\_forecast, xgb\_forecast)

weights <- optimize\_weights(arima\_forecast, prophet\_forecast, testing\_data)

arima\_prophet\_forecast <- hybrid\_forecast\_function(weights, arima\_forecast, prophet\_forecast)

calculate\_metrics <- function(actual, forecast) {

rmse <- sqrt(mean((actual - forecast)^2))

mape <- mean(abs((actual - forecast) / actual)) \* 100

return(c(RMSE = rmse, MAPE = mape))}

model\_metrics <- data.frame(

Model = c("ARIMA", "ETS", "MLP", "Prophet", "XGBoost", "ARIMA\_ETS", "ARIMA\_MLP", "ARIMA\_XGBoost", "ARIMA\_Prophet"),

do.call(rbind, list(

calculate\_metrics(testing\_data, arima\_forecast),

calculate\_metrics(testing\_data, ets\_forecast),

calculate\_metrics(testing\_data, mlp\_forecast),

calculate\_metrics(testing\_data, prophet\_forecast),

calculate\_metrics(testing\_data, xgb\_forecast),

calculate\_metrics(testing\_data, arima\_ets\_forecast),

calculate\_metrics(testing\_data, arima\_mlp\_forecast),

calculate\_metrics(testing\_data, arima\_xgb\_forecast),

calculate\_metrics(testing\_data, arima\_prophet\_forecast))))

print("Model Performance Metrics (RMSE and MAE):")

print(model\_metrics)

all\_forecasts <- data.frame(

Year = seq(2018, 2022),

Actual = as.numeric(testing\_data),

ARIMA = as.numeric(arima\_forecast),

ETS = as.numeric(ets\_forecast),

MLP = as.numeric(mlp\_forecast),

Prophet = as.numeric(prophet\_forecast),

XGBoost = as.numeric(xgb\_forecast),

ARIMA\_ETS = as.numeric(arima\_ets\_forecast),

ARIMA\_MLP = as.numeric(arima\_mlp\_forecast),

ARIMA\_Prophet = as.numeric(arima\_prophet\_forecast),

ARIMA\_XGBoost = as.numeric(arima\_xgb\_forecast))

melted\_forecasts <- melt(all\_forecasts, id.vars = "Year", variable.name = "Model", value.name = "Value")

ggplot(data = melted\_forecasts, aes(x = Year, y = Value, color = Model)) +

geom\_line() +

labs(title = "LBR", y = "Mortality Rate", x = "Year") +

theme\_minimal()

**# Out-of-Sample Forecast for MLI UN-5 Female Mortality Rates**

set.seed(123)

MLI <- c(292.95, 284.67, 276.5, 268.41, 260.53, 253.26, 246.33, 239.42, 232.86, 226.94, 221.73, 217.18, 213.02, 209.02, 205.78, 202.37, 198.87, 194.94, 190.54, 185.48, 179.95, 173.91, 167.91, 162.05, 156.47, 150.99, 145.78, 140.87, 136.17, 131.78, 127.59, 123.64, 119.82, 116.11, 112.72, 109.44, 106.41, 103.48, 100.51, 97.268, 94.331, 91.433, 88.429)

mli\_ts <- ts(MLI, start = 1980, frequency = 1)

training\_data <- window(mli\_ts, start=1980, end=2017)

testing\_data <- window(mli\_ts, start=2018, end=2022)

arima\_model <- auto.arima(training\_data)

arima\_forecast <- forecast(arima\_model, h = length(testing\_data))$mean

ets\_model <- ets(training\_data)

ets\_forecast <- forecast(ets\_model, h = length(testing\_data))$mean

mlp\_model <- mlp(training\_data)

mlp\_forecast <- forecast(mlp\_model, h = length(testing\_data))$mean

train\_df <- data.frame(

ds = seq.Date(from = as.Date("1980-01-01"), by = "year", length.out = length(training\_data)),

y = as.numeric(training\_data))

prophet\_model <- prophet(train\_df, yearly.seasonality = FALSE, weekly.seasonality = FALSE, daily.seasonality = FALSE)

future <- data.frame(ds = seq.Date(from = max(train\_df$ds) + 1, by = "year", length.out = length(testing\_data)))

prophet\_forecast <- predict(prophet\_model, future)$yhat

lagged\_data <- data.frame(

Target = training\_data[-1],

Lag1 = training\_data[-length(training\_data)])

xgb\_model <- xgboost(data = as.matrix(lagged\_data$Lag1), label = lagged\_data$Target, nrounds = 100)

xgb\_forecast <- numeric(length(testing\_data))

last\_value <- training\_data[length(training\_data)]

for (i in 1:length(testing\_data)) {

xgb\_forecast[i] <- predict(xgb\_model, newdata = as.matrix(last\_value))

last\_value <- xgb\_forecast[i]}

hybrid\_forecast\_function <- function(weights, f1, f2) {

weights[1] \* as.numeric(f1) + weights[2] \* as.numeric(f2)}

fitness\_function <- function(weights, f1, f2, actual) {

hybrid\_forecast <- hybrid\_forecast\_function(weights, f1, f2)

sqrt(mean((actual - hybrid\_forecast)^2))}

optimize\_weights <- function(f1, f2, actual) {

ga\_model <- ga(

type = "real-valued",

fitness = function(w) -fitness\_function(w, f1, f2, actual),

lower = c(0, 0),

upper = c(1, 1),

popSize = 50,

maxiter = 100)

return(ga\_model@solution)}

weights <- optimize\_weights(arima\_forecast, ets\_forecast, testing\_data)

arima\_ets\_forecast <- hybrid\_forecast\_function(weights, arima\_forecast, ets\_forecast)

weights <- optimize\_weights(arima\_forecast, mlp\_forecast, testing\_data)

arima\_mlp\_forecast <- hybrid\_forecast\_function(weights, arima\_forecast, mlp\_forecast)

weights <- optimize\_weights(arima\_forecast, xgb\_forecast, testing\_data)

arima\_xgb\_forecast <- hybrid\_forecast\_function(weights, arima\_forecast, xgb\_forecast)

weights <- optimize\_weights(arima\_forecast, prophet\_forecast, testing\_data)

arima\_prophet\_forecast <- hybrid\_forecast\_function(weights, arima\_forecast, prophet\_forecast)

calculate\_metrics <- function(actual, forecast) {

rmse <- sqrt(mean((actual - forecast)^2))

mape <- mean(abs((actual - forecast) / actual)) \* 100

return(c(RMSE = rmse, MAPE = mape))}

model\_metrics <- data.frame(

Model = c("ARIMA", "ETS", "MLP", "Prophet", "XGBoost", "ARIMA\_ETS", "ARIMA\_MLP", "ARIMA\_XGBoost", "ARIMA\_Prophet"),

do.call(rbind, list(

calculate\_metrics(testing\_data, arima\_forecast),

calculate\_metrics(testing\_data, ets\_forecast),

calculate\_metrics(testing\_data, mlp\_forecast),

calculate\_metrics(testing\_data, prophet\_forecast),

calculate\_metrics(testing\_data, xgb\_forecast),

calculate\_metrics(testing\_data, arima\_ets\_forecast),

calculate\_metrics(testing\_data, arima\_mlp\_forecast),

calculate\_metrics(testing\_data, arima\_xgb\_forecast),

calculate\_metrics(testing\_data, arima\_prophet\_forecast))))

print("Model Performance Metrics (RMSE and MAE):")

print(model\_metrics)

all\_forecasts <- data.frame(

Year = seq(2018, 2022),

Actual = as.numeric(testing\_data),

ARIMA = as.numeric(arima\_forecast),

ETS = as.numeric(ets\_forecast),

MLP = as.numeric(mlp\_forecast),

Prophet = as.numeric(prophet\_forecast),

XGBoost = as.numeric(xgb\_forecast),

ARIMA\_ETS = as.numeric(arima\_ets\_forecast),

ARIMA\_MLP = as.numeric(arima\_mlp\_forecast),

ARIMA\_Prophet = as.numeric(arima\_prophet\_forecast),

ARIMA\_XGBoost = as.numeric(arima\_xgb\_forecast))

melted\_forecasts <- melt(all\_forecasts, id.vars = "Year", variable.name = "Model", value.name = "Value")

ggplot(data = melted\_forecasts, aes(x = Year, y = Value, color = Model)) +

geom\_line() +

labs(title = "MLI", y = "Mortality Rate", x = "Year") +

theme\_minimal()

**# Out-of-Sample Forecast for MRT UN-5 Female Mortality Rates**

set.seed(123)

MRT <- c(156.07, 151.38, 146.63, 141.46, 135.75, 129.43, 123.35, 117.88, 113.49, 110.31, 108.17, 106.82, 105.89, 105.12, 104.4, 103.38, 102.01, 100.2, 97.791, 94.58, 90.591, 86.048, 81.117, 75.847, 70.779, 66.011, 61.744, 58.259, 55.392, 53.096, 51.267, 49.739, 48.388, 47.076, 45.774, 44.408, 43.081, 41.683, 40.286, 38.994, 37.638, 36.412, 35.12)

mrt\_ts <- ts(MRT, start = 1980, frequency = 1)

training\_data <- window(mrt\_ts, start=1980, end=2017)

testing\_data <- window(mrt\_ts, start=2018, end=2022)

arima\_model <- auto.arima(training\_data)

arima\_forecast <- forecast(arima\_model, h = length(testing\_data))$mean

ets\_model <- ets(training\_data)

ets\_forecast <- forecast(ets\_model, h = length(testing\_data))$mean

mlp\_model <- mlp(training\_data)

mlp\_forecast <- forecast(mlp\_model, h = length(testing\_data))$mean

train\_df <- data.frame(

ds = seq.Date(from = as.Date("1980-01-01"), by = "year", length.out = length(training\_data)),

y = as.numeric(training\_data))

prophet\_model <- prophet(train\_df, yearly.seasonality = FALSE, weekly.seasonality = FALSE, daily.seasonality = FALSE)

future <- data.frame(ds = seq.Date(from = max(train\_df$ds) + 1, by = "year", length.out = length(testing\_data)))

prophet\_forecast <- predict(prophet\_model, future)$yhat

lagged\_data <- data.frame(

Target = training\_data[-1],

Lag1 = training\_data[-length(training\_data)])

xgb\_model <- xgboost(data = as.matrix(lagged\_data$Lag1), label = lagged\_data$Target, nrounds = 100)

xgb\_forecast <- numeric(length(testing\_data))

last\_value <- training\_data[length(training\_data)]

for (i in 1:length(testing\_data)) {

xgb\_forecast[i] <- predict(xgb\_model, newdata = as.matrix(last\_value))

last\_value <- xgb\_forecast[i]}

hybrid\_forecast\_function <- function(weights, f1, f2) {

weights[1] \* as.numeric(f1) + weights[2] \* as.numeric(f2)}

fitness\_function <- function(weights, f1, f2, actual) {

hybrid\_forecast <- hybrid\_forecast\_function(weights, f1, f2)

sqrt(mean((actual - hybrid\_forecast)^2))}

optimize\_weights <- function(f1, f2, actual) {

ga\_model <- ga(

type = "real-valued",

fitness = function(w) -fitness\_function(w, f1, f2, actual),

lower = c(0, 0),

upper = c(1, 1),

popSize = 50,

maxiter = 100)

return(ga\_model@solution)}

weights <- optimize\_weights(arima\_forecast, ets\_forecast, testing\_data)

arima\_ets\_forecast <- hybrid\_forecast\_function(weights, arima\_forecast, ets\_forecast)

weights <- optimize\_weights(arima\_forecast, mlp\_forecast, testing\_data)

arima\_mlp\_forecast <- hybrid\_forecast\_function(weights, arima\_forecast, mlp\_forecast)

weights <- optimize\_weights(arima\_forecast, xgb\_forecast, testing\_data)

arima\_xgb\_forecast <- hybrid\_forecast\_function(weights, arima\_forecast, xgb\_forecast)

weights <- optimize\_weights(arima\_forecast, prophet\_forecast, testing\_data)

arima\_prophet\_forecast <- hybrid\_forecast\_function(weights, arima\_forecast, prophet\_forecast)

calculate\_metrics <- function(actual, forecast) {

rmse <- sqrt(mean((actual - forecast)^2))

mape <- mean(abs((actual - forecast) / actual)) \* 100

return(c(RMSE = rmse, MAPE = mape))}

model\_metrics <- data.frame(

Model = c("ARIMA", "ETS", "MLP", "Prophet", "XGBoost", "ARIMA\_ETS", "ARIMA\_MLP", "ARIMA\_XGBoost", "ARIMA\_Prophet"),

do.call(rbind, list(

calculate\_metrics(testing\_data, arima\_forecast),

calculate\_metrics(testing\_data, ets\_forecast),

calculate\_metrics(testing\_data, mlp\_forecast),

calculate\_metrics(testing\_data, prophet\_forecast),

calculate\_metrics(testing\_data, xgb\_forecast),

calculate\_metrics(testing\_data, arima\_ets\_forecast),

calculate\_metrics(testing\_data, arima\_mlp\_forecast),

calculate\_metrics(testing\_data, arima\_xgb\_forecast),

calculate\_metrics(testing\_data, arima\_prophet\_forecast))))

print("Model Performance Metrics (RMSE and MAE):")

print(model\_metrics)

all\_forecasts <- data.frame(

Year = seq(2018, 2022),

Actual = as.numeric(testing\_data),

ARIMA = as.numeric(arima\_forecast),

ETS = as.numeric(ets\_forecast),

MLP = as.numeric(mlp\_forecast),

Prophet = as.numeric(prophet\_forecast),

XGBoost = as.numeric(xgb\_forecast),

ARIMA\_ETS = as.numeric(arima\_ets\_forecast),

ARIMA\_MLP = as.numeric(arima\_mlp\_forecast),

ARIMA\_Prophet = as.numeric(arima\_prophet\_forecast),

ARIMA\_XGBoost = as.numeric(arima\_xgb\_forecast))

melted\_forecasts <- melt(all\_forecasts, id.vars = "Year", variable.name = "Model", value.name = "Value")

ggplot(data = melted\_forecasts, aes(x = Year, y = Value, color = Model)) +

geom\_line() +

labs(title = "MRT", y = "Mortality Rate", x = "Year") +

theme\_minimal()

**# Out-of-Sample Forecast for CPV UN-5 Female Mortality Rates**

set.seed(123)

CPV <- c(74.54, 73.49, 72.86, 72.34, 71.42, 69.77, 67.15, 63.79, 60.32, 57.1, 54.53, 52.83, 51.95, 51.37, 50.61, 49.23, 46.77, 43.35, 39.4, 35.28, 31.59, 28.73, 26.78, 25.57, 24.98, 24.8, 24.82, 24.75, 24.48, 23.9, 22.99, 21.85, 20.57, 19.22, 17.85, 16.53, 15.35, 14.31, 13.44, 12.73, 12.11, 11.6, 11.13)

cpv\_ts <- ts(CPV, start = 1980, frequency = 1)

training\_data <- window(cpv\_ts, start=1980, end=2017)

testing\_data <- window(cpv\_ts, start=2018, end=2022)

arima\_model <- auto.arima(training\_data)

arima\_forecast <- forecast(arima\_model, h = length(testing\_data))$mean

ets\_model <- ets(training\_data)

ets\_forecast <- forecast(ets\_model, h = length(testing\_data))$mean

mlp\_model <- mlp(training\_data)

mlp\_forecast <- forecast(mlp\_model, h = length(testing\_data))$mean

train\_df <- data.frame(

ds = seq.Date(from = as.Date("1980-01-01"), by = "year", length.out = length(training\_data)),

y = as.numeric(training\_data))

prophet\_model <- prophet(train\_df, yearly.seasonality = FALSE, weekly.seasonality = FALSE, daily.seasonality = FALSE)

future <- data.frame(ds = seq.Date(from = max(train\_df$ds) + 1, by = "year", length.out = length(testing\_data)))

prophet\_forecast <- predict(prophet\_model, future)$yhat

lagged\_data <- data.frame(

Target = training\_data[-1],

Lag1 = training\_data[-length(training\_data)])

xgb\_model <- xgboost(data = as.matrix(lagged\_data$Lag1), label = lagged\_data$Target, nrounds = 100)

xgb\_forecast <- numeric(length(testing\_data))

last\_value <- training\_data[length(training\_data)]

for (i in 1:length(testing\_data)) {

xgb\_forecast[i] <- predict(xgb\_model, newdata = as.matrix(last\_value))

last\_value <- xgb\_forecast[i]}

hybrid\_forecast\_function <- function(weights, f1, f2) {

weights[1] \* as.numeric(f1) + weights[2] \* as.numeric(f2)}

fitness\_function <- function(weights, f1, f2, actual) {

hybrid\_forecast <- hybrid\_forecast\_function(weights, f1, f2)

sqrt(mean((actual - hybrid\_forecast)^2))}

optimize\_weights <- function(f1, f2, actual) {

ga\_model <- ga(

type = "real-valued",

fitness = function(w) -fitness\_function(w, f1, f2, actual),

lower = c(0, 0),

upper = c(1, 1),

popSize = 50,

maxiter = 100)

return(ga\_model@solution)}

weights <- optimize\_weights(arima\_forecast, ets\_forecast, testing\_data)

arima\_ets\_forecast <- hybrid\_forecast\_function(weights, arima\_forecast, ets\_forecast)

weights <- optimize\_weights(arima\_forecast, mlp\_forecast, testing\_data)

arima\_mlp\_forecast <- hybrid\_forecast\_function(weights, arima\_forecast, mlp\_forecast)

weights <- optimize\_weights(arima\_forecast, xgb\_forecast, testing\_data)

arima\_xgb\_forecast <- hybrid\_forecast\_function(weights, arima\_forecast, xgb\_forecast)

weights <- optimize\_weights(arima\_forecast, prophet\_forecast, testing\_data)

arima\_prophet\_forecast <- hybrid\_forecast\_function(weights, arima\_forecast, prophet\_forecast)

calculate\_metrics <- function(actual, forecast) {

rmse <- sqrt(mean((actual - forecast)^2))

mape <- mean(abs((actual - forecast) / actual)) \* 100

return(c(RMSE = rmse, MAPE = mape))}

model\_metrics <- data.frame(

Model = c("ARIMA", "ETS", "MLP", "Prophet", "XGBoost", "ARIMA\_ETS", "ARIMA\_MLP", "ARIMA\_XGBoost", "ARIMA\_Prophet"),

do.call(rbind, list(

calculate\_metrics(testing\_data, arima\_forecast),

calculate\_metrics(testing\_data, ets\_forecast),

calculate\_metrics(testing\_data, mlp\_forecast),

calculate\_metrics(testing\_data, prophet\_forecast),

calculate\_metrics(testing\_data, xgb\_forecast),

calculate\_metrics(testing\_data, arima\_ets\_forecast),

calculate\_metrics(testing\_data, arima\_mlp\_forecast),

calculate\_metrics(testing\_data, arima\_xgb\_forecast),

calculate\_metrics(testing\_data, arima\_prophet\_forecast))))

print("Model Performance Metrics (RMSE and MAE):")

print(model\_metrics)

all\_forecasts <- data.frame(

Year = seq(2018, 2022),

Actual = as.numeric(testing\_data),

ARIMA = as.numeric(arima\_forecast),

ETS = as.numeric(ets\_forecast),

MLP = as.numeric(mlp\_forecast),

Prophet = as.numeric(prophet\_forecast),

XGBoost = as.numeric(xgb\_forecast),

ARIMA\_ETS = as.numeric(arima\_ets\_forecast),

ARIMA\_MLP = as.numeric(arima\_mlp\_forecast),

ARIMA\_Prophet = as.numeric(arima\_prophet\_forecast),

ARIMA\_XGBoost = as.numeric(arima\_xgb\_forecast))

melted\_forecasts <- melt(all\_forecasts, id.vars = "Year", variable.name = "Model", value.name = "Value")

ggplot(data = melted\_forecasts, aes(x = Year, y = Value, color = Model)) +

geom\_line() +

labs(title = "CPV", y = "Mortality Rate", x = "Year") +

theme\_minimal()

**# Out-of-Sample Forecast for TCD UN-5 Female Mortality Rates**

set.seed(123)

TCD <- c(229.48, 227.27, 224.63, 222.03, 219.39, 216.62, 213.6, 210.6, 207.56, 204.64, 201.84, 198.99, 196.39, 193.96, 191.55, 189.37, 187.19, 184.47, 181.58, 178.3, 175.03, 171.55, 168.08, 164.61, 161.35, 157.94, 154.61, 150.83, 147.11, 143.2, 139.23, 135.35, 131.58, 127.95, 124.45, 120.81, 117.29, 113.58, 109.91, 106.2, 102.85, 99.404, 96.231)

tcd\_ts <- ts(TCD, start = 1980, frequency = 1)

training\_data <- window(tcd\_ts, start=1980, end=2017)

testing\_data <- window(tcd\_ts, start=2018, end=2022)

arima\_model <- auto.arima(training\_data)

arima\_forecast <- forecast(arima\_model, h = length(testing\_data))$mean

ets\_model <- ets(training\_data)

ets\_forecast <- forecast(ets\_model, h = length(testing\_data))$mean

mlp\_model <- mlp(training\_data)

mlp\_forecast <- forecast(mlp\_model, h = length(testing\_data))$mean

train\_df <- data.frame(

ds = seq.Date(from = as.Date("1980-01-01"), by = "year", length.out = length(training\_data)),

y = as.numeric(training\_data))

prophet\_model <- prophet(train\_df, yearly.seasonality = FALSE, weekly.seasonality = FALSE, daily.seasonality = FALSE)

future <- data.frame(ds = seq.Date(from = max(train\_df$ds) + 1, by = "year", length.out = length(testing\_data)))

prophet\_forecast <- predict(prophet\_model, future)$yhat

lagged\_data <- data.frame(

Target = training\_data[-1],

Lag1 = training\_data[-length(training\_data)])

xgb\_model <- xgboost(data = as.matrix(lagged\_data$Lag1), label = lagged\_data$Target, nrounds = 100)

xgb\_forecast <- numeric(length(testing\_data))

last\_value <- training\_data[length(training\_data)]

for (i in 1:length(testing\_data)) {

xgb\_forecast[i] <- predict(xgb\_model, newdata = as.matrix(last\_value))

last\_value <- xgb\_forecast[i]}

hybrid\_forecast\_function <- function(weights, f1, f2) {

weights[1] \* as.numeric(f1) + weights[2] \* as.numeric(f2)}

fitness\_function <- function(weights, f1, f2, actual) {

hybrid\_forecast <- hybrid\_forecast\_function(weights, f1, f2)

sqrt(mean((actual - hybrid\_forecast)^2))}

optimize\_weights <- function(f1, f2, actual) {

ga\_model <- ga(

type = "real-valued",

fitness = function(w) -fitness\_function(w, f1, f2, actual),

lower = c(0, 0),

upper = c(1, 1),

popSize = 50,

maxiter = 100)

return(ga\_model@solution)}

weights <- optimize\_weights(arima\_forecast, ets\_forecast, testing\_data)

arima\_ets\_forecast <- hybrid\_forecast\_function(weights, arima\_forecast, ets\_forecast)

weights <- optimize\_weights(arima\_forecast, mlp\_forecast, testing\_data)

arima\_mlp\_forecast <- hybrid\_forecast\_function(weights, arima\_forecast, mlp\_forecast)

weights <- optimize\_weights(arima\_forecast, xgb\_forecast, testing\_data)

arima\_xgb\_forecast <- hybrid\_forecast\_function(weights, arima\_forecast, xgb\_forecast)

weights <- optimize\_weights(arima\_forecast, prophet\_forecast, testing\_data)

arima\_prophet\_forecast <- hybrid\_forecast\_function(weights, arima\_forecast, prophet\_forecast)

calculate\_metrics <- function(actual, forecast) {

rmse <- sqrt(mean((actual - forecast)^2))

mape <- mean(abs((actual - forecast) / actual)) \* 100

return(c(RMSE = rmse, MAPE = mape))}

model\_metrics <- data.frame(

Model = c("ARIMA", "ETS", "MLP", "Prophet", "XGBoost", "ARIMA\_ETS", "ARIMA\_MLP", "ARIMA\_XGBoost", "ARIMA\_Prophet"),

do.call(rbind, list(

calculate\_metrics(testing\_data, arima\_forecast),

calculate\_metrics(testing\_data, ets\_forecast),

calculate\_metrics(testing\_data, mlp\_forecast),

calculate\_metrics(testing\_data, prophet\_forecast),

calculate\_metrics(testing\_data, xgb\_forecast),

calculate\_metrics(testing\_data, arima\_ets\_forecast),

calculate\_metrics(testing\_data, arima\_mlp\_forecast),

calculate\_metrics(testing\_data, arima\_xgb\_forecast),

calculate\_metrics(testing\_data, arima\_prophet\_forecast))))

print("Model Performance Metrics (RMSE and MAE):")

print(model\_metrics)

all\_forecasts <- data.frame(

Year = seq(2018, 2022),

Actual = as.numeric(testing\_data),

ARIMA = as.numeric(arima\_forecast),

ETS = as.numeric(ets\_forecast),

MLP = as.numeric(mlp\_forecast),

Prophet = as.numeric(prophet\_forecast),

XGBoost = as.numeric(xgb\_forecast),

ARIMA\_ETS = as.numeric(arima\_ets\_forecast),

ARIMA\_MLP = as.numeric(arima\_mlp\_forecast),

ARIMA\_Prophet = as.numeric(arima\_prophet\_forecast),

ARIMA\_XGBoost = as.numeric(arima\_xgb\_forecast))

melted\_forecasts <- melt(all\_forecasts, id.vars = "Year", variable.name = "Model", value.name = "Value")

ggplot(data = melted\_forecasts, aes(x = Year, y = Value, color = Model)) +

geom\_line() +

labs(title = "TCD", y = "Mortality Rate", x = "Year") +

theme\_minimal()

**# Out-of-Sample Forecast for CIV UN-5 Female Mortality Rates**

set.seed(123)

CIV <- c(155.87, 151.78, 148.33, 145.5, 143.07, 141.61, 140.46, 139.78, 139.56, 139.93, 140.24, 140.86, 141.3, 141.38, 140.91, 140.06, 138.8, 136.99, 134.62, 131.83, 128.62, 125.27, 121.66, 117.97, 114.13, 110.09, 106.37, 102.54, 98.637, 94.707, 91.419, 88.443, 85.677, 82.741, 79.602, 76.84, 74.63, 72.457, 70.049, 67.555, 65.409, 63.433, 61.482)

civ\_ts <- ts(CIV, start = 1980, frequency = 1)

training\_data <- window(civ\_ts, start=1980, end=2017)

testing\_data <- window(civ\_ts, start=2018, end=2022)

arima\_model <- auto.arima(training\_data)

arima\_forecast <- forecast(arima\_model, h = length(testing\_data))$mean

ets\_model <- ets(training\_data)

ets\_forecast <- forecast(ets\_model, h = length(testing\_data))$mean

mlp\_model <- mlp(training\_data)

mlp\_forecast <- forecast(mlp\_model, h = length(testing\_data))$mean

train\_df <- data.frame(

ds = seq.Date(from = as.Date("1980-01-01"), by = "year", length.out = length(training\_data)),

y = as.numeric(training\_data))

prophet\_model <- prophet(train\_df, yearly.seasonality = FALSE, weekly.seasonality = FALSE, daily.seasonality = FALSE)

future <- data.frame(ds = seq.Date(from = max(train\_df$ds) + 1, by = "year", length.out = length(testing\_data)))

prophet\_forecast <- predict(prophet\_model, future)$yhat

lagged\_data <- data.frame(

Target = training\_data[-1],

Lag1 = training\_data[-length(training\_data)])

xgb\_model <- xgboost(data = as.matrix(lagged\_data$Lag1), label = lagged\_data$Target, nrounds = 100)

xgb\_forecast <- numeric(length(testing\_data))

last\_value <- training\_data[length(training\_data)]

for (i in 1:length(testing\_data)) {

xgb\_forecast[i] <- predict(xgb\_model, newdata = as.matrix(last\_value))

last\_value <- xgb\_forecast[i]}

hybrid\_forecast\_function <- function(weights, f1, f2) {

weights[1] \* as.numeric(f1) + weights[2] \* as.numeric(f2)}

fitness\_function <- function(weights, f1, f2, actual) {

hybrid\_forecast <- hybrid\_forecast\_function(weights, f1, f2)

sqrt(mean((actual - hybrid\_forecast)^2))}

optimize\_weights <- function(f1, f2, actual) {

ga\_model <- ga(

type = "real-valued",

fitness = function(w) -fitness\_function(w, f1, f2, actual),

lower = c(0, 0),

upper = c(1, 1),

popSize = 50,

maxiter = 100)

return(ga\_model@solution)}

weights <- optimize\_weights(arima\_forecast, ets\_forecast, testing\_data)

arima\_ets\_forecast <- hybrid\_forecast\_function(weights, arima\_forecast, ets\_forecast)

weights <- optimize\_weights(arima\_forecast, mlp\_forecast, testing\_data)

arima\_mlp\_forecast <- hybrid\_forecast\_function(weights, arima\_forecast, mlp\_forecast)

weights <- optimize\_weights(arima\_forecast, xgb\_forecast, testing\_data)

arima\_xgb\_forecast <- hybrid\_forecast\_function(weights, arima\_forecast, xgb\_forecast)

weights <- optimize\_weights(arima\_forecast, prophet\_forecast, testing\_data)

arima\_prophet\_forecast <- hybrid\_forecast\_function(weights, arima\_forecast, prophet\_forecast)

calculate\_metrics <- function(actual, forecast) {

rmse <- sqrt(mean((actual - forecast)^2))

mape <- mean(abs((actual - forecast) / actual)) \* 100

return(c(RMSE = rmse, MAPE = mape))}

model\_metrics <- data.frame(

Model = c("ARIMA", "ETS", "MLP", "Prophet", "XGBoost", "ARIMA\_ETS", "ARIMA\_MLP", "ARIMA\_XGBoost", "ARIMA\_Prophet"),

do.call(rbind, list(

calculate\_metrics(testing\_data, arima\_forecast),

calculate\_metrics(testing\_data, ets\_forecast),

calculate\_metrics(testing\_data, mlp\_forecast),

calculate\_metrics(testing\_data, prophet\_forecast),

calculate\_metrics(testing\_data, xgb\_forecast),

calculate\_metrics(testing\_data, arima\_ets\_forecast),

calculate\_metrics(testing\_data, arima\_mlp\_forecast),

calculate\_metrics(testing\_data, arima\_xgb\_forecast),

calculate\_metrics(testing\_data, arima\_prophet\_forecast))))

print("Model Performance Metrics (RMSE and MAE):")

print(model\_metrics)

all\_forecasts <- data.frame(

Year = seq(2018, 2022),

Actual = as.numeric(testing\_data),

ARIMA = as.numeric(arima\_forecast),

ETS = as.numeric(ets\_forecast),

MLP = as.numeric(mlp\_forecast),

Prophet = as.numeric(prophet\_forecast),

XGBoost = as.numeric(xgb\_forecast),

ARIMA\_ETS = as.numeric(arima\_ets\_forecast),

ARIMA\_MLP = as.numeric(arima\_mlp\_forecast),

ARIMA\_Prophet = as.numeric(arima\_prophet\_forecast),

ARIMA\_XGBoost = as.numeric(arima\_xgb\_forecast))

melted\_forecasts <- melt(all\_forecasts, id.vars = "Year", variable.name = "Model", value.name = "Value")

ggplot(data = melted\_forecasts, aes(x = Year, y = Value, color = Model)) +

geom\_line() +

labs(title = "CIV", y = "Mortality Rate", x = "Year") +

theme\_minimal()

**# Out-of-Sample Forecast for GNQ UN-5 Female Mortality Rates**

set.seed(123)

GNQ <- c(191.36, 191.36, 191.36, 187.19, 183.29, 179.7, 176.62, 174.11, 172.12, 170.43, 168.83, 167.22, 165.83, 164.47, 163.09, 161.65, 159.78, 157.13, 154.19, 150.57, 146.7, 142.58, 138.38, 134.08, 129.77, 125.17, 120.68, 116.17, 111.74, 107.62, 103.71, 100.1, 96.602, 93.199, 89.845, 86.55, 83.577, 80.926, 77.911, 74.951, 72.399, 69.923, 67.394)

gnq\_ts <- ts(GNQ, start = 1980, frequency = 1)

training\_data <- window(gnq\_ts, start=1980, end=2017)

testing\_data <- window(gnq\_ts, start=2018, end=2022)

arima\_model <- auto.arima(training\_data)

arima\_forecast <- forecast(arima\_model, h = length(testing\_data))$mean

ets\_model <- ets(training\_data)

ets\_forecast <- forecast(ets\_model, h = length(testing\_data))$mean

mlp\_model <- mlp(training\_data)

mlp\_forecast <- forecast(mlp\_model, h = length(testing\_data))$mean

train\_df <- data.frame(

ds = seq.Date(from = as.Date("1980-01-01"), by = "year", length.out = length(training\_data)),

y = as.numeric(training\_data))

prophet\_model <- prophet(train\_df, yearly.seasonality = FALSE, weekly.seasonality = FALSE, daily.seasonality = FALSE)

future <- data.frame(ds = seq.Date(from = max(train\_df$ds) + 1, by = "year", length.out = length(testing\_data)))

prophet\_forecast <- predict(prophet\_model, future)$yhat

lagged\_data <- data.frame(

Target = training\_data[-1],

Lag1 = training\_data[-length(training\_data)])

xgb\_model <- xgboost(data = as.matrix(lagged\_data$Lag1), label = lagged\_data$Target, nrounds = 100)

xgb\_forecast <- numeric(length(testing\_data))

last\_value <- training\_data[length(training\_data)]

for (i in 1:length(testing\_data)) {

xgb\_forecast[i] <- predict(xgb\_model, newdata = as.matrix(last\_value))

last\_value <- xgb\_forecast[i]}

hybrid\_forecast\_function <- function(weights, f1, f2) {

weights[1] \* as.numeric(f1) + weights[2] \* as.numeric(f2)}

fitness\_function <- function(weights, f1, f2, actual) {

hybrid\_forecast <- hybrid\_forecast\_function(weights, f1, f2)

sqrt(mean((actual - hybrid\_forecast)^2))}

optimize\_weights <- function(f1, f2, actual) {

ga\_model <- ga(

type = "real-valued",

fitness = function(w) -fitness\_function(w, f1, f2, actual),

lower = c(0, 0),

upper = c(1, 1),

popSize = 50,

maxiter = 100)

return(ga\_model@solution)}

weights <- optimize\_weights(arima\_forecast, ets\_forecast, testing\_data)

arima\_ets\_forecast <- hybrid\_forecast\_function(weights, arima\_forecast, ets\_forecast)

weights <- optimize\_weights(arima\_forecast, mlp\_forecast, testing\_data)

arima\_mlp\_forecast <- hybrid\_forecast\_function(weights, arima\_forecast, mlp\_forecast)

weights <- optimize\_weights(arima\_forecast, xgb\_forecast, testing\_data)

arima\_xgb\_forecast <- hybrid\_forecast\_function(weights, arima\_forecast, xgb\_forecast)

weights <- optimize\_weights(arima\_forecast, prophet\_forecast, testing\_data)

arima\_prophet\_forecast <- hybrid\_forecast\_function(weights, arima\_forecast, prophet\_forecast)

calculate\_metrics <- function(actual, forecast) {

rmse <- sqrt(mean((actual - forecast)^2))

mape <- mean(abs((actual - forecast) / actual)) \* 100

return(c(RMSE = rmse, MAPE = mape))}

model\_metrics <- data.frame(

Model = c("ARIMA", "ETS", "MLP", "Prophet", "XGBoost", "ARIMA\_ETS", "ARIMA\_MLP", "ARIMA\_XGBoost", "ARIMA\_Prophet"),

do.call(rbind, list(

calculate\_metrics(testing\_data, arima\_forecast),

calculate\_metrics(testing\_data, ets\_forecast),

calculate\_metrics(testing\_data, mlp\_forecast),

calculate\_metrics(testing\_data, prophet\_forecast),

calculate\_metrics(testing\_data, xgb\_forecast),

calculate\_metrics(testing\_data, arima\_ets\_forecast),

calculate\_metrics(testing\_data, arima\_mlp\_forecast),

calculate\_metrics(testing\_data, arima\_xgb\_forecast),

calculate\_metrics(testing\_data, arima\_prophet\_forecast))))

print("Model Performance Metrics (RMSE and MAE):")

print(model\_metrics)

all\_forecasts <- data.frame(

Year = seq(2018, 2022),

Actual = as.numeric(testing\_data),

ARIMA = as.numeric(arima\_forecast),

ETS = as.numeric(ets\_forecast),

MLP = as.numeric(mlp\_forecast),

Prophet = as.numeric(prophet\_forecast),

XGBoost = as.numeric(xgb\_forecast),

ARIMA\_ETS = as.numeric(arima\_ets\_forecast),

ARIMA\_MLP = as.numeric(arima\_mlp\_forecast),

ARIMA\_Prophet = as.numeric(arima\_prophet\_forecast),

ARIMA\_XGBoost = as.numeric(arima\_xgb\_forecast))

melted\_forecasts <- melt(all\_forecasts, id.vars = "Year", variable.name = "Model", value.name = "Value")

ggplot(data = melted\_forecasts, aes(x = Year, y = Value, color = Model)) +

geom\_line() +

labs(title = "GNQ", y = "Mortality Rate", x = "Year") +

theme\_minimal()

**# Out-of-Sample Forecast for GMB UN-5 Female Mortality Rates**

set.seed(123)

GMB <- c(218.69, 212.42, 206.11, 199.41, 192.9, 186.34, 179.98, 173.64, 167.69, 161.94, 156.34, 150.94, 145.7, 140.5, 135.35, 130.14, 124.96, 119.99, 115.04, 110.19, 105.7, 101.22, 96.831, 92.538, 88.442, 84.432, 80.626, 77.009, 73.554, 70.314, 67.256, 64.272, 61.5, 58.867, 56.351, 54.003, 51.65, 49.518, 47.552, 45.723, 43.944, 42.364, 40.798)

gmb\_ts <- ts(GMB, start = 1980, frequency = 1)

training\_data <- window(gmb\_ts, start=1980, end=2017)

testing\_data <- window(gmb\_ts, start=2018, end=2022)

arima\_model <- auto.arima(training\_data)

arima\_forecast <- forecast(arima\_model, h = length(testing\_data))$mean

ets\_model <- ets(training\_data)

ets\_forecast <- forecast(ets\_model, h = length(testing\_data))$mean

mlp\_model <- mlp(training\_data)

mlp\_forecast <- forecast(mlp\_model, h = length(testing\_data))$mean

train\_df <- data.frame(

ds = seq.Date(from = as.Date("1980-01-01"), by = "year", length.out = length(training\_data)),

y = as.numeric(training\_data))

prophet\_model <- prophet(train\_df, yearly.seasonality = FALSE, weekly.seasonality = FALSE, daily.seasonality = FALSE)

future <- data.frame(ds = seq.Date(from = max(train\_df$ds) + 1, by = "year", length.out = length(testing\_data)))

prophet\_forecast <- predict(prophet\_model, future)$yhat

lagged\_data <- data.frame(

Target = training\_data[-1],

Lag1 = training\_data[-length(training\_data)])

xgb\_model <- xgboost(data = as.matrix(lagged\_data$Lag1), label = lagged\_data$Target, nrounds = 100)

xgb\_forecast <- numeric(length(testing\_data))

last\_value <- training\_data[length(training\_data)]

for (i in 1:length(testing\_data)) {

xgb\_forecast[i] <- predict(xgb\_model, newdata = as.matrix(last\_value))

last\_value <- xgb\_forecast[i]}

hybrid\_forecast\_function <- function(weights, f1, f2) {

weights[1] \* as.numeric(f1) + weights[2] \* as.numeric(f2)}

fitness\_function <- function(weights, f1, f2, actual) {

hybrid\_forecast <- hybrid\_forecast\_function(weights, f1, f2)

sqrt(mean((actual - hybrid\_forecast)^2))}

optimize\_weights <- function(f1, f2, actual) {

ga\_model <- ga(

type = "real-valued",

fitness = function(w) -fitness\_function(w, f1, f2, actual),

lower = c(0, 0),

upper = c(1, 1),

popSize = 50,

maxiter = 100)

return(ga\_model@solution)}

weights <- optimize\_weights(arima\_forecast, ets\_forecast, testing\_data)

arima\_ets\_forecast <- hybrid\_forecast\_function(weights, arima\_forecast, ets\_forecast)

weights <- optimize\_weights(arima\_forecast, mlp\_forecast, testing\_data)

arima\_mlp\_forecast <- hybrid\_forecast\_function(weights, arima\_forecast, mlp\_forecast)

weights <- optimize\_weights(arima\_forecast, xgb\_forecast, testing\_data)

arima\_xgb\_forecast <- hybrid\_forecast\_function(weights, arima\_forecast, xgb\_forecast)

weights <- optimize\_weights(arima\_forecast, prophet\_forecast, testing\_data)

arima\_prophet\_forecast <- hybrid\_forecast\_function(weights, arima\_forecast, prophet\_forecast)

calculate\_metrics <- function(actual, forecast) {

rmse <- sqrt(mean((actual - forecast)^2))

mape <- mean(abs((actual - forecast) / actual)) \* 100

return(c(RMSE = rmse, MAPE = mape))}

model\_metrics <- data.frame(

Model = c("ARIMA", "ETS", "MLP", "Prophet", "XGBoost", "ARIMA\_ETS", "ARIMA\_MLP", "ARIMA\_XGBoost", "ARIMA\_Prophet"),

do.call(rbind, list(

calculate\_metrics(testing\_data, arima\_forecast),

calculate\_metrics(testing\_data, ets\_forecast),

calculate\_metrics(testing\_data, mlp\_forecast),

calculate\_metrics(testing\_data, prophet\_forecast),

calculate\_metrics(testing\_data, xgb\_forecast),

calculate\_metrics(testing\_data, arima\_ets\_forecast),

calculate\_metrics(testing\_data, arima\_mlp\_forecast),

calculate\_metrics(testing\_data, arima\_xgb\_forecast),

calculate\_metrics(testing\_data, arima\_prophet\_forecast))))

print("Model Performance Metrics (RMSE and MAE):")

print(model\_metrics)

all\_forecasts <- data.frame(

Year = seq(2018, 2022),

Actual = as.numeric(testing\_data),

ARIMA = as.numeric(arima\_forecast),

ETS = as.numeric(ets\_forecast),

MLP = as.numeric(mlp\_forecast),

Prophet = as.numeric(prophet\_forecast),

XGBoost = as.numeric(xgb\_forecast),

ARIMA\_ETS = as.numeric(arima\_ets\_forecast),

ARIMA\_MLP = as.numeric(arima\_mlp\_forecast),

ARIMA\_Prophet = as.numeric(arima\_prophet\_forecast),

ARIMA\_XGBoost = as.numeric(arima\_xgb\_forecast))

melted\_forecasts <- melt(all\_forecasts, id.vars = "Year", variable.name = "Model", value.name = "Value")

ggplot(data = melted\_forecasts, aes(x = Year, y = Value, color = Model)) +

geom\_line() +

labs(title = "GMB", y = "Mortality Rate", x = "Year") +

theme\_minimal()

**# Out-of-Sample Forecast for GHA UN-5 Female Mortality Rates**

set.seed(123)

GHA <- c(158.81, 157.23, 155.8, 153.64, 150.58, 146.43, 141.49, 135.85, 130.02, 124.41, 119.45, 115.21, 111.94, 109.53, 107.86, 106.25, 104.47, 102.19, 99.322, 95.799, 92.031, 88.179, 84.433, 81.158, 78.287, 75.786, 73.547, 71.238, 68.635, 65.89, 63.004, 60.024, 57.1, 54.417, 51.989, 49.672, 47.576, 45.608, 43.844, 42.083, 40.495, 39.113, 37.769)

gha\_ts <- ts(GHA, start = 1980, frequency = 1)

training\_data <- window(gha\_ts, start=1980, end=2017)

testing\_data <- window(gha\_ts, start=2018, end=2022)

arima\_model <- auto.arima(training\_data)

arima\_forecast <- forecast(arima\_model, h = length(testing\_data))$mean

ets\_model <- ets(training\_data)

ets\_forecast <- forecast(ets\_model, h = length(testing\_data))$mean

mlp\_model <- mlp(training\_data)

mlp\_forecast <- forecast(mlp\_model, h = length(testing\_data))$mean

train\_df <- data.frame(

ds = seq.Date(from = as.Date("1980-01-01"), by = "year", length.out = length(training\_data)),

y = as.numeric(training\_data))

prophet\_model <- prophet(train\_df, yearly.seasonality = FALSE, weekly.seasonality = FALSE, daily.seasonality = FALSE)

future <- data.frame(ds = seq.Date(from = max(train\_df$ds) + 1, by = "year", length.out = length(testing\_data)))

prophet\_forecast <- predict(prophet\_model, future)$yhat

lagged\_data <- data.frame(

Target = training\_data[-1],

Lag1 = training\_data[-length(training\_data)])

xgb\_model <- xgboost(data = as.matrix(lagged\_data$Lag1), label = lagged\_data$Target, nrounds = 100)

xgb\_forecast <- numeric(length(testing\_data))

last\_value <- training\_data[length(training\_data)]

for (i in 1:length(testing\_data)) {

xgb\_forecast[i] <- predict(xgb\_model, newdata = as.matrix(last\_value))

last\_value <- xgb\_forecast[i]}

hybrid\_forecast\_function <- function(weights, f1, f2) {

weights[1] \* as.numeric(f1) + weights[2] \* as.numeric(f2)}

fitness\_function <- function(weights, f1, f2, actual) {

hybrid\_forecast <- hybrid\_forecast\_function(weights, f1, f2)

sqrt(mean((actual - hybrid\_forecast)^2))}

optimize\_weights <- function(f1, f2, actual) {

ga\_model <- ga(

type = "real-valued",

fitness = function(w) -fitness\_function(w, f1, f2, actual),

lower = c(0, 0),

upper = c(1, 1),

popSize = 50,

maxiter = 100)

return(ga\_model@solution)}

weights <- optimize\_weights(arima\_forecast, ets\_forecast, testing\_data)

arima\_ets\_forecast <- hybrid\_forecast\_function(weights, arima\_forecast, ets\_forecast)

weights <- optimize\_weights(arima\_forecast, mlp\_forecast, testing\_data)

arima\_mlp\_forecast <- hybrid\_forecast\_function(weights, arima\_forecast, mlp\_forecast)

weights <- optimize\_weights(arima\_forecast, xgb\_forecast, testing\_data)

arima\_xgb\_forecast <- hybrid\_forecast\_function(weights, arima\_forecast, xgb\_forecast)

weights <- optimize\_weights(arima\_forecast, prophet\_forecast, testing\_data)

arima\_prophet\_forecast <- hybrid\_forecast\_function(weights, arima\_forecast, prophet\_forecast)

calculate\_metrics <- function(actual, forecast) {

rmse <- sqrt(mean((actual - forecast)^2))

mape <- mean(abs((actual - forecast) / actual)) \* 100

return(c(RMSE = rmse, MAPE = mape))}

model\_metrics <- data.frame(

Model = c("ARIMA", "ETS", "MLP", "Prophet", "XGBoost", "ARIMA\_ETS", "ARIMA\_MLP", "ARIMA\_XGBoost", "ARIMA\_Prophet"),

do.call(rbind, list(

calculate\_metrics(testing\_data, arima\_forecast),

calculate\_metrics(testing\_data, ets\_forecast),

calculate\_metrics(testing\_data, mlp\_forecast),

calculate\_metrics(testing\_data, prophet\_forecast),

calculate\_metrics(testing\_data, xgb\_forecast),

calculate\_metrics(testing\_data, arima\_ets\_forecast),

calculate\_metrics(testing\_data, arima\_mlp\_forecast),

calculate\_metrics(testing\_data, arima\_xgb\_forecast),

calculate\_metrics(testing\_data, arima\_prophet\_forecast))))

print("Model Performance Metrics (RMSE and MAE):")

print(model\_metrics)

all\_forecasts <- data.frame(

Year = seq(2018, 2022),

Actual = as.numeric(testing\_data),

ARIMA = as.numeric(arima\_forecast),

ETS = as.numeric(ets\_forecast),

MLP = as.numeric(mlp\_forecast),

Prophet = as.numeric(prophet\_forecast),

XGBoost = as.numeric(xgb\_forecast),

ARIMA\_ETS = as.numeric(arima\_ets\_forecast),

ARIMA\_MLP = as.numeric(arima\_mlp\_forecast),

ARIMA\_Prophet = as.numeric(arima\_prophet\_forecast),

ARIMA\_XGBoost = as.numeric(arima\_xgb\_forecast))

melted\_forecasts <- melt(all\_forecasts, id.vars = "Year", variable.name = "Model", value.name = "Value")

ggplot(data = melted\_forecasts, aes(x = Year, y = Value, color = Model)) +

geom\_line() +

labs(title = "GHA", y = "Mortality Rate", x = "Year") +

theme\_minimal()

**# Out-of-Sample Forecast for NER UN-5 Female Mortality Rates**

set.seed(123)

NER <- c(321.21, 320.77, 322.92, 326.94, 331.58, 335.68, 338.76, 339.5, 338.37, 335.45, 330.47, 322.71, 313.64, 302.56, 290.4, 277.6, 264.84, 253.29, 242.87, 233.92, 225.05, 215.12, 203.83, 192.02, 180.27, 169.16, 159.39, 150.85, 143.64, 137.94, 133.42, 130, 127.57, 125.84, 124.43, 123.19, 122.05, 120.94, 119.92, 118.8, 117.53, 115.94, 113.83)

ner\_ts <- ts(NER, start = 1980, frequency = 1)

training\_data <- window(ner\_ts, start=1980, end=2017)

testing\_data <- window(ner\_ts, start=2018, end=2022)

arima\_model <- auto.arima(training\_data)

arima\_forecast <- forecast(arima\_model, h = length(testing\_data))$mean

ets\_model <- ets(training\_data)

ets\_forecast <- forecast(ets\_model, h = length(testing\_data))$mean

mlp\_model <- mlp(training\_data)

mlp\_forecast <- forecast(mlp\_model, h = length(testing\_data))$mean

train\_df <- data.frame(

ds = seq.Date(from = as.Date("1980-01-01"), by = "year", length.out = length(training\_data)),

y = as.numeric(training\_data))

prophet\_model <- prophet(train\_df, yearly.seasonality = FALSE, weekly.seasonality = FALSE, daily.seasonality = FALSE)

future <- data.frame(ds = seq.Date(from = max(train\_df$ds) + 1, by = "year", length.out = length(testing\_data)))

prophet\_forecast <- predict(prophet\_model, future)$yhat

lagged\_data <- data.frame(

Target = training\_data[-1],

Lag1 = training\_data[-length(training\_data)])

xgb\_model <- xgboost(data = as.matrix(lagged\_data$Lag1), label = lagged\_data$Target, nrounds = 100)

xgb\_forecast <- numeric(length(testing\_data))

last\_value <- training\_data[length(training\_data)]

for (i in 1:length(testing\_data)) {

xgb\_forecast[i] <- predict(xgb\_model, newdata = as.matrix(last\_value))

last\_value <- xgb\_forecast[i]}

hybrid\_forecast\_function <- function(weights, f1, f2) {

weights[1] \* as.numeric(f1) + weights[2] \* as.numeric(f2)}

fitness\_function <- function(weights, f1, f2, actual) {

hybrid\_forecast <- hybrid\_forecast\_function(weights, f1, f2)

sqrt(mean((actual - hybrid\_forecast)^2))}

optimize\_weights <- function(f1, f2, actual) {

ga\_model <- ga(

type = "real-valued",

fitness = function(w) -fitness\_function(w, f1, f2, actual),

lower = c(0, 0),

upper = c(1, 1),

popSize = 50,

maxiter = 100)

return(ga\_model@solution)}

weights <- optimize\_weights(arima\_forecast, ets\_forecast, testing\_data)

arima\_ets\_forecast <- hybrid\_forecast\_function(weights, arima\_forecast, ets\_forecast)

weights <- optimize\_weights(arima\_forecast, mlp\_forecast, testing\_data)

arima\_mlp\_forecast <- hybrid\_forecast\_function(weights, arima\_forecast, mlp\_forecast)

weights <- optimize\_weights(arima\_forecast, xgb\_forecast, testing\_data)

arima\_xgb\_forecast <- hybrid\_forecast\_function(weights, arima\_forecast, xgb\_forecast)

weights <- optimize\_weights(arima\_forecast, prophet\_forecast, testing\_data)

arima\_prophet\_forecast <- hybrid\_forecast\_function(weights, arima\_forecast, prophet\_forecast)

calculate\_metrics <- function(actual, forecast) {

rmse <- sqrt(mean((actual - forecast)^2))

mape <- mean(abs((actual - forecast) / actual)) \* 100

return(c(RMSE = rmse, MAPE = mape))}

model\_metrics <- data.frame(

Model = c("ARIMA", "ETS", "MLP", "Prophet", "XGBoost", "ARIMA\_ETS", "ARIMA\_MLP", "ARIMA\_XGBoost", "ARIMA\_Prophet"),

do.call(rbind, list(

calculate\_metrics(testing\_data, arima\_forecast),

calculate\_metrics(testing\_data, ets\_forecast),

calculate\_metrics(testing\_data, mlp\_forecast),

calculate\_metrics(testing\_data, prophet\_forecast),

calculate\_metrics(testing\_data, xgb\_forecast),

calculate\_metrics(testing\_data, arima\_ets\_forecast),

calculate\_metrics(testing\_data, arima\_mlp\_forecast),

calculate\_metrics(testing\_data, arima\_xgb\_forecast),

calculate\_metrics(testing\_data, arima\_prophet\_forecast))))

print("Model Performance Metrics (RMSE and MAE):")

print(model\_metrics)

all\_forecasts <- data.frame(

Year = seq(2018, 2022),

Actual = as.numeric(testing\_data),

ARIMA = as.numeric(arima\_forecast),

ETS = as.numeric(ets\_forecast),

MLP = as.numeric(mlp\_forecast),

Prophet = as.numeric(prophet\_forecast),

XGBoost = as.numeric(xgb\_forecast),

ARIMA\_ETS = as.numeric(arima\_ets\_forecast),

ARIMA\_MLP = as.numeric(arima\_mlp\_forecast),

ARIMA\_Prophet = as.numeric(arima\_prophet\_forecast),

ARIMA\_XGBoost = as.numeric(arima\_xgb\_forecast))

melted\_forecasts <- melt(all\_forecasts, id.vars = "Year", variable.name = "Model", value.name = "Value")

ggplot(data = melted\_forecasts, aes(x = Year, y = Value, color = Model)) +

geom\_line() +

labs(title = "NER", y = "Mortality Rate", x = "Year") +

theme\_minimal()

**# Out-of-Sample Forecast for NGA UN-5 Female Mortality Rates**

set.seed(123)

NGA <- c(202.31, 199.33, 197.51, 196.55, 196.41, 196.88, 197.45, 197.92, 198.53, 198.59, 198.28, 197.85, 197.5, 197, 195.8, 194.02, 191.49, 188.05, 183.46, 178.64, 173.65, 168.12, 162.43, 156.65, 151.03, 145.78, 140.8, 136.74, 133.26, 130.13, 127.6, 125.38, 123.53, 121.83, 120.45, 119.25, 117.81, 115.79, 113.29, 110.33, 107.47, 104.33, 100.91)

nga\_ts <- ts(NGA, start = 1980, frequency = 1)

training\_data <- window(nga\_ts, start=1980, end=2017)

testing\_data <- window(nga\_ts, start=2018, end=2022)

arima\_model <- auto.arima(training\_data)

arima\_forecast <- forecast(arima\_model, h = length(testing\_data))$mean

ets\_model <- ets(training\_data)

ets\_forecast <- forecast(ets\_model, h = length(testing\_data))$mean

mlp\_model <- mlp(training\_data)

mlp\_forecast <- forecast(mlp\_model, h = length(testing\_data))$mean

train\_df <- data.frame(

ds = seq.Date(from = as.Date("1980-01-01"), by = "year", length.out = length(training\_data)),

y = as.numeric(training\_data))

prophet\_model <- prophet(train\_df, yearly.seasonality = FALSE, weekly.seasonality = FALSE, daily.seasonality = FALSE)

future <- data.frame(ds = seq.Date(from = max(train\_df$ds) + 1, by = "year", length.out = length(testing\_data)))

prophet\_forecast <- predict(prophet\_model, future)$yhat

lagged\_data <- data.frame(

Target = training\_data[-1],

Lag1 = training\_data[-length(training\_data)])

xgb\_model <- xgboost(data = as.matrix(lagged\_data$Lag1), label = lagged\_data$Target, nrounds = 100)

xgb\_forecast <- numeric(length(testing\_data))

last\_value <- training\_data[length(training\_data)]

for (i in 1:length(testing\_data)) {

xgb\_forecast[i] <- predict(xgb\_model, newdata = as.matrix(last\_value))

last\_value <- xgb\_forecast[i]}

hybrid\_forecast\_function <- function(weights, f1, f2) {

weights[1] \* as.numeric(f1) + weights[2] \* as.numeric(f2)}

fitness\_function <- function(weights, f1, f2, actual) {

hybrid\_forecast <- hybrid\_forecast\_function(weights, f1, f2)

sqrt(mean((actual - hybrid\_forecast)^2))}

optimize\_weights <- function(f1, f2, actual) {

ga\_model <- ga(

type = "real-valued",

fitness = function(w) -fitness\_function(w, f1, f2, actual),

lower = c(0, 0),

upper = c(1, 1),

popSize = 50,

maxiter = 100)

return(ga\_model@solution)}

weights <- optimize\_weights(arima\_forecast, ets\_forecast, testing\_data)

arima\_ets\_forecast <- hybrid\_forecast\_function(weights, arima\_forecast, ets\_forecast)

weights <- optimize\_weights(arima\_forecast, mlp\_forecast, testing\_data)

arima\_mlp\_forecast <- hybrid\_forecast\_function(weights, arima\_forecast, mlp\_forecast)

weights <- optimize\_weights(arima\_forecast, xgb\_forecast, testing\_data)

arima\_xgb\_forecast <- hybrid\_forecast\_function(weights, arima\_forecast, xgb\_forecast)

weights <- optimize\_weights(arima\_forecast, prophet\_forecast, testing\_data)

arima\_prophet\_forecast <- hybrid\_forecast\_function(weights, arima\_forecast, prophet\_forecast)

calculate\_metrics <- function(actual, forecast) {

rmse <- sqrt(mean((actual - forecast)^2))

mape <- mean(abs((actual - forecast) / actual)) \* 100

return(c(RMSE = rmse, MAPE = mape))}

model\_metrics <- data.frame(

Model = c("ARIMA", "ETS", "MLP", "Prophet", "XGBoost", "ARIMA\_ETS", "ARIMA\_MLP", "ARIMA\_XGBoost", "ARIMA\_Prophet"),

do.call(rbind, list(

calculate\_metrics(testing\_data, arima\_forecast),

calculate\_metrics(testing\_data, ets\_forecast),

calculate\_metrics(testing\_data, mlp\_forecast),

calculate\_metrics(testing\_data, prophet\_forecast),

calculate\_metrics(testing\_data, xgb\_forecast),

calculate\_metrics(testing\_data, arima\_ets\_forecast),

calculate\_metrics(testing\_data, arima\_mlp\_forecast),

calculate\_metrics(testing\_data, arima\_xgb\_forecast),

calculate\_metrics(testing\_data, arima\_prophet\_forecast))))

print("Model Performance Metrics (RMSE and MAE):")

print(model\_metrics)

all\_forecasts <- data.frame(

Year = seq(2018, 2022),

Actual = as.numeric(testing\_data),

ARIMA = as.numeric(arima\_forecast),

ETS = as.numeric(ets\_forecast),

MLP = as.numeric(mlp\_forecast),

Prophet = as.numeric(prophet\_forecast),

XGBoost = as.numeric(xgb\_forecast),

ARIMA\_ETS = as.numeric(arima\_ets\_forecast),

ARIMA\_MLP = as.numeric(arima\_mlp\_forecast),

ARIMA\_Prophet = as.numeric(arima\_prophet\_forecast),

ARIMA\_XGBoost = as.numeric(arima\_xgb\_forecast))

melted\_forecasts <- melt(all\_forecasts, id.vars = "Year", variable.name = "Model", value.name = "Value")

ggplot(data = melted\_forecasts, aes(x = Year, y = Value, color = Model)) +

geom\_line() +

labs(title = "NGA", y = "Mortality Rate", x = "Year") +

theme\_minimal()

**# Out-of-Sample Forecast for SEN UN-5 Female Mortality Rates**

set.seed(123)

SEN <- c(197.36, 190.43, 184.17, 177.74, 170.74, 163.33, 155.23, 147.36, 140.37, 134.91, 131.26, 129.31, 128.79, 129.38, 130.47, 131.39, 132.03, 131.85, 130.41, 127.49, 122.74, 116.22, 108.51, 100.58, 92.856, 85.743, 79.37, 73.77, 68.913, 64.721, 61.043, 57.681, 54.575, 51.641, 48.809, 46.187, 43.699, 41.385, 39.284, 37.453, 35.79, 34.366, 32.959)

sen\_ts <- ts(SEN, start = 1980, frequency = 1)

training\_data <- window(sen\_ts, start=1980, end=2017)

testing\_data <- window(sen\_ts, start=2018, end=2022)

arima\_model <- auto.arima(training\_data)

arima\_forecast <- forecast(arima\_model, h = length(testing\_data))$mean

ets\_model <- ets(training\_data)

ets\_forecast <- forecast(ets\_model, h = length(testing\_data))$mean

mlp\_model <- mlp(training\_data)

mlp\_forecast <- forecast(mlp\_model, h = length(testing\_data))$mean

train\_df <- data.frame(

ds = seq.Date(from = as.Date("1980-01-01"), by = "year", length.out = length(training\_data)),

y = as.numeric(training\_data))

prophet\_model <- prophet(train\_df, yearly.seasonality = FALSE, weekly.seasonality = FALSE, daily.seasonality = FALSE)

future <- data.frame(ds = seq.Date(from = max(train\_df$ds) + 1, by = "year", length.out = length(testing\_data)))

prophet\_forecast <- predict(prophet\_model, future)$yhat

lagged\_data <- data.frame(

Target = training\_data[-1],

Lag1 = training\_data[-length(training\_data)])

xgb\_model <- xgboost(data = as.matrix(lagged\_data$Lag1), label = lagged\_data$Target, nrounds = 100)

xgb\_forecast <- numeric(length(testing\_data))

last\_value <- training\_data[length(training\_data)]

for (i in 1:length(testing\_data)) {

xgb\_forecast[i] <- predict(xgb\_model, newdata = as.matrix(last\_value))

last\_value <- xgb\_forecast[i]}

hybrid\_forecast\_function <- function(weights, f1, f2) {

weights[1] \* as.numeric(f1) + weights[2] \* as.numeric(f2)}

fitness\_function <- function(weights, f1, f2, actual) {

hybrid\_forecast <- hybrid\_forecast\_function(weights, f1, f2)

sqrt(mean((actual - hybrid\_forecast)^2))}

optimize\_weights <- function(f1, f2, actual) {

ga\_model <- ga(

type = "real-valued",

fitness = function(w) -fitness\_function(w, f1, f2, actual),

lower = c(0, 0),

upper = c(1, 1),

popSize = 50,

maxiter = 100)

return(ga\_model@solution)}

weights <- optimize\_weights(arima\_forecast, ets\_forecast, testing\_data)

arima\_ets\_forecast <- hybrid\_forecast\_function(weights, arima\_forecast, ets\_forecast)

weights <- optimize\_weights(arima\_forecast, mlp\_forecast, testing\_data)

arima\_mlp\_forecast <- hybrid\_forecast\_function(weights, arima\_forecast, mlp\_forecast)

weights <- optimize\_weights(arima\_forecast, xgb\_forecast, testing\_data)

arima\_xgb\_forecast <- hybrid\_forecast\_function(weights, arima\_forecast, xgb\_forecast)

weights <- optimize\_weights(arima\_forecast, prophet\_forecast, testing\_data)

arima\_prophet\_forecast <- hybrid\_forecast\_function(weights, arima\_forecast, prophet\_forecast)

calculate\_metrics <- function(actual, forecast) {

rmse <- sqrt(mean((actual - forecast)^2))

mape <- mean(abs((actual - forecast) / actual)) \* 100

return(c(RMSE = rmse, MAPE = mape))}

model\_metrics <- data.frame(

Model = c("ARIMA", "ETS", "MLP", "Prophet", "XGBoost", "ARIMA\_ETS", "ARIMA\_MLP", "ARIMA\_XGBoost", "ARIMA\_Prophet"),

do.call(rbind, list(

calculate\_metrics(testing\_data, arima\_forecast),

calculate\_metrics(testing\_data, ets\_forecast),

calculate\_metrics(testing\_data, mlp\_forecast),

calculate\_metrics(testing\_data, prophet\_forecast),

calculate\_metrics(testing\_data, xgb\_forecast),

calculate\_metrics(testing\_data, arima\_ets\_forecast),

calculate\_metrics(testing\_data, arima\_mlp\_forecast),

calculate\_metrics(testing\_data, arima\_xgb\_forecast),

calculate\_metrics(testing\_data, arima\_prophet\_forecast))))

print("Model Performance Metrics (RMSE and MAE):")

print(model\_metrics)

all\_forecasts <- data.frame(

Year = seq(2018, 2022),

Actual = as.numeric(testing\_data),

ARIMA = as.numeric(arima\_forecast),

ETS = as.numeric(ets\_forecast),

MLP = as.numeric(mlp\_forecast),

Prophet = as.numeric(prophet\_forecast),

XGBoost = as.numeric(xgb\_forecast),

ARIMA\_ETS = as.numeric(arima\_ets\_forecast),

ARIMA\_MLP = as.numeric(arima\_mlp\_forecast),

ARIMA\_Prophet = as.numeric(arima\_prophet\_forecast),

ARIMA\_XGBoost = as.numeric(arima\_xgb\_forecast))

melted\_forecasts <- melt(all\_forecasts, id.vars = "Year", variable.name = "Model", value.name = "Value")

ggplot(data = melted\_forecasts, aes(x = Year, y = Value, color = Model)) +

geom\_line() +

labs(title = "SEN", y = "Mortality Rate", x = "Year") +

theme\_minimal()

**# Out-of-Sample Forecast for SLE UN-5 Female Mortality Rates**

set.seed(123)

SLE <- c(275.31, 271.84, 268.41, 265.3, 262.37, 259.63, 256.86, 254.35, 252.02, 249.68, 247.61, 245.36, 243.02, 240.73, 238.34, 235.5, 232.3, 228.46, 224.21, 219.93, 214.74, 209.84, 204.54, 199.09, 193.47, 187.35, 180.98, 174.12, 167.04, 160.13, 152.99, 146.07, 139.71, 133.64, 131.71, 132.69, 117.94, 113.38, 109, 104.89, 100.98, 97.39, 93.834)

sle\_ts <- ts(SLE, start = 1980, frequency = 1)

training\_data <- window(sle\_ts, start=1980, end=2017)

testing\_data <- window(sle\_ts, start=2018, end=2022)

arima\_model <- auto.arima(training\_data)

arima\_forecast <- forecast(arima\_model, h = length(testing\_data))$mean

ets\_model <- ets(training\_data)

ets\_forecast <- forecast(ets\_model, h = length(testing\_data))$mean

mlp\_model <- mlp(training\_data)

mlp\_forecast <- forecast(mlp\_model, h = length(testing\_data))$mean

train\_df <- data.frame(

ds = seq.Date(from = as.Date("1980-01-01"), by = "year", length.out = length(training\_data)),

y = as.numeric(training\_data))

prophet\_model <- prophet(train\_df, yearly.seasonality = FALSE, weekly.seasonality = FALSE, daily.seasonality = FALSE)

future <- data.frame(ds = seq.Date(from = max(train\_df$ds) + 1, by = "year", length.out = length(testing\_data)))

prophet\_forecast <- predict(prophet\_model, future)$yhat

lagged\_data <- data.frame(

Target = training\_data[-1],

Lag1 = training\_data[-length(training\_data)])

xgb\_model <- xgboost(data = as.matrix(lagged\_data$Lag1), label = lagged\_data$Target, nrounds = 100)

xgb\_forecast <- numeric(length(testing\_data))

last\_value <- training\_data[length(training\_data)]

for (i in 1:length(testing\_data)) {

xgb\_forecast[i] <- predict(xgb\_model, newdata = as.matrix(last\_value))

last\_value <- xgb\_forecast[i]}

hybrid\_forecast\_function <- function(weights, f1, f2) {

weights[1] \* as.numeric(f1) + weights[2] \* as.numeric(f2)}

fitness\_function <- function(weights, f1, f2, actual) {

hybrid\_forecast <- hybrid\_forecast\_function(weights, f1, f2)

sqrt(mean((actual - hybrid\_forecast)^2))}

optimize\_weights <- function(f1, f2, actual) {

ga\_model <- ga(

type = "real-valued",

fitness = function(w) -fitness\_function(w, f1, f2, actual),

lower = c(0, 0),

upper = c(1, 1),

popSize = 50,

maxiter = 100)

return(ga\_model@solution)}

weights <- optimize\_weights(arima\_forecast, ets\_forecast, testing\_data)

arima\_ets\_forecast <- hybrid\_forecast\_function(weights, arima\_forecast, ets\_forecast)

weights <- optimize\_weights(arima\_forecast, mlp\_forecast, testing\_data)

arima\_mlp\_forecast <- hybrid\_forecast\_function(weights, arima\_forecast, mlp\_forecast)

weights <- optimize\_weights(arima\_forecast, xgb\_forecast, testing\_data)

arima\_xgb\_forecast <- hybrid\_forecast\_function(weights, arima\_forecast, xgb\_forecast)

weights <- optimize\_weights(arima\_forecast, prophet\_forecast, testing\_data)

arima\_prophet\_forecast <- hybrid\_forecast\_function(weights, arima\_forecast, prophet\_forecast)

calculate\_metrics <- function(actual, forecast) {

rmse <- sqrt(mean((actual - forecast)^2))

mape <- mean(abs((actual - forecast) / actual)) \* 100

return(c(RMSE = rmse, MAPE = mape))}

model\_metrics <- data.frame(

Model = c("ARIMA", "ETS", "MLP", "Prophet", "XGBoost", "ARIMA\_ETS", "ARIMA\_MLP", "ARIMA\_XGBoost", "ARIMA\_Prophet"),

do.call(rbind, list(

calculate\_metrics(testing\_data, arima\_forecast),

calculate\_metrics(testing\_data, ets\_forecast),

calculate\_metrics(testing\_data, mlp\_forecast),

calculate\_metrics(testing\_data, prophet\_forecast),

calculate\_metrics(testing\_data, xgb\_forecast),

calculate\_metrics(testing\_data, arima\_ets\_forecast),

calculate\_metrics(testing\_data, arima\_mlp\_forecast),

calculate\_metrics(testing\_data, arima\_xgb\_forecast),

calculate\_metrics(testing\_data, arima\_prophet\_forecast))))

print("Model Performance Metrics (RMSE and MAE):")

print(model\_metrics)

all\_forecasts <- data.frame(

Year = seq(2018, 2022),

Actual = as.numeric(testing\_data),

ARIMA = as.numeric(arima\_forecast),

ETS = as.numeric(ets\_forecast),

MLP = as.numeric(mlp\_forecast),

Prophet = as.numeric(prophet\_forecast),

XGBoost = as.numeric(xgb\_forecast),

ARIMA\_ETS = as.numeric(arima\_ets\_forecast),

ARIMA\_MLP = as.numeric(arima\_mlp\_forecast),

ARIMA\_Prophet = as.numeric(arima\_prophet\_forecast),

ARIMA\_XGBoost = as.numeric(arima\_xgb\_forecast))

melted\_forecasts <- melt(all\_forecasts, id.vars = "Year", variable.name = "Model", value.name = "Value")

ggplot(data = melted\_forecasts, aes(x = Year, y = Value, color = Model)) +

geom\_line() +

labs(title = "SLE", y = "Mortality Rate", x = "Year") +

theme\_minimal()

**# Out-of-Sample Forecast for TGO UN-5 Female Mortality Rates**

set.seed(123)

TGO <- c(170.07, 166.02, 162.1, 158.4, 154.8, 151.46, 148.31, 145.36, 142.76, 140.28, 138.07, 136.15, 134.07, 131.83, 129.48, 126.83, 123.89, 120.9, 117.78, 114.41, 110.97, 107.7, 104.53, 101.47, 98.521, 95.734, 93.083, 90.405, 87.752, 85.191, 82.71, 80.095, 77.449, 74.901, 72.412, 70.036, 67.809, 65.551, 63.374, 61.207, 59.207, 57.243, 55.347)

tgo\_ts <- ts(TGO, start = 1980, frequency = 1)

training\_data <- window(tgo\_ts, start=1980, end=2017)

testing\_data <- window(tgo\_ts, start=2018, end=2022)

arima\_model <- auto.arima(training\_data)

arima\_forecast <- forecast(arima\_model, h = length(testing\_data))$mean

ets\_model <- ets(training\_data)

ets\_forecast <- forecast(ets\_model, h = length(testing\_data))$mean

mlp\_model <- mlp(training\_data)

mlp\_forecast <- forecast(mlp\_model, h = length(testing\_data))$mean

train\_df <- data.frame(

ds = seq.Date(from = as.Date("1980-01-01"), by = "year", length.out = length(training\_data)),

y = as.numeric(training\_data))

prophet\_model <- prophet(train\_df, yearly.seasonality = FALSE, weekly.seasonality = FALSE, daily.seasonality = FALSE)

future <- data.frame(ds = seq.Date(from = max(train\_df$ds) + 1, by = "year", length.out = length(testing\_data)))

prophet\_forecast <- predict(prophet\_model, future)$yhat

lagged\_data <- data.frame(

Target = training\_data[-1],

Lag1 = training\_data[-length(training\_data)])

xgb\_model <- xgboost(data = as.matrix(lagged\_data$Lag1), label = lagged\_data$Target, nrounds = 100)

xgb\_forecast <- numeric(length(testing\_data))

last\_value <- training\_data[length(training\_data)]

for (i in 1:length(testing\_data)) {

xgb\_forecast[i] <- predict(xgb\_model, newdata = as.matrix(last\_value))

last\_value <- xgb\_forecast[i]}

hybrid\_forecast\_function <- function(weights, f1, f2) {

weights[1] \* as.numeric(f1) + weights[2] \* as.numeric(f2)}

fitness\_function <- function(weights, f1, f2, actual) {

hybrid\_forecast <- hybrid\_forecast\_function(weights, f1, f2)

sqrt(mean((actual - hybrid\_forecast)^2))}

optimize\_weights <- function(f1, f2, actual) {

ga\_model <- ga(

type = "real-valued",

fitness = function(w) -fitness\_function(w, f1, f2, actual),

lower = c(0, 0),

upper = c(1, 1),

popSize = 50,

maxiter = 100)

return(ga\_model@solution)}

weights <- optimize\_weights(arima\_forecast, ets\_forecast, testing\_data)

arima\_ets\_forecast <- hybrid\_forecast\_function(weights, arima\_forecast, ets\_forecast)

weights <- optimize\_weights(arima\_forecast, mlp\_forecast, testing\_data)

arima\_mlp\_forecast <- hybrid\_forecast\_function(weights, arima\_forecast, mlp\_forecast)

weights <- optimize\_weights(arima\_forecast, xgb\_forecast, testing\_data)

arima\_xgb\_forecast <- hybrid\_forecast\_function(weights, arima\_forecast, xgb\_forecast)

weights <- optimize\_weights(arima\_forecast, prophet\_forecast, testing\_data)

arima\_prophet\_forecast <- hybrid\_forecast\_function(weights, arima\_forecast, prophet\_forecast)

calculate\_metrics <- function(actual, forecast) {

rmse <- sqrt(mean((actual - forecast)^2))

mape <- mean(abs((actual - forecast) / actual)) \* 100

return(c(RMSE = rmse, MAPE = mape))}

model\_metrics <- data.frame(

Model = c("ARIMA", "ETS", "MLP", "Prophet", "XGBoost", "ARIMA\_ETS", "ARIMA\_MLP", "ARIMA\_XGBoost", "ARIMA\_Prophet"),

do.call(rbind, list(

calculate\_metrics(testing\_data, arima\_forecast),

calculate\_metrics(testing\_data, ets\_forecast),

calculate\_metrics(testing\_data, mlp\_forecast),

calculate\_metrics(testing\_data, prophet\_forecast),

calculate\_metrics(testing\_data, xgb\_forecast),

calculate\_metrics(testing\_data, arima\_ets\_forecast),

calculate\_metrics(testing\_data, arima\_mlp\_forecast),

calculate\_metrics(testing\_data, arima\_xgb\_forecast),

calculate\_metrics(testing\_data, arima\_prophet\_forecast))))

print("Model Performance Metrics (RMSE and MAE):")

print(model\_metrics)

all\_forecasts <- data.frame(

Year = seq(2018, 2022),

Actual = as.numeric(testing\_data),

ARIMA = as.numeric(arima\_forecast),

ETS = as.numeric(ets\_forecast),

MLP = as.numeric(mlp\_forecast),

Prophet = as.numeric(prophet\_forecast),

XGBoost = as.numeric(xgb\_forecast),

ARIMA\_ETS = as.numeric(arima\_ets\_forecast),

ARIMA\_MLP = as.numeric(arima\_mlp\_forecast),

ARIMA\_Prophet = as.numeric(arima\_prophet\_forecast),

ARIMA\_XGBoost = as.numeric(arima\_xgb\_forecast))

melted\_forecasts <- melt(all\_forecasts, id.vars = "Year", variable.name = "Model", value.name = "Value")

ggplot(data = melted\_forecasts, aes(x = Year, y = Value, color = Model)) +

geom\_line() +

labs(title = "TGO", y = "Mortality Rate", x = "Year") +

theme\_minimal()

**Sensitivity Analysis for Under-5 Female Mortality Rates in West Africa**

**# Sensitivity Analysis for BEN UN-5 Female Mortality using Time Horizon: 3, 5, and 7 years**

set.seed(123)

BEN <- c(200.83, 196.89, 193.1, 189.49, 185.97, 182.54, 179.04, 175.49, 171.91, 168.35, 164.65, 160.95, 157.28, 153.78, 150.26, 146.84, 143.54, 140.36, 137.13, 133.54, 129.97, 126.4, 123.28, 120.26, 117.34, 114.49, 111.76, 109.19, 106.55, 104.02, 101.64, 99.275, 97.074, 94.852, 92.611, 90.421, 88.394, 86.269, 84.13, 81.838, 79.489, 77.019, 74.778)

ben\_ts <- ts(BEN, start = 1980, frequency = 1)

horizons <- c(3, 5, 7)

results <- data.frame(Horizon = integer(), Model = character(), RMSE = numeric())

for (h in horizons) {

training\_data <- window(ben\_ts, start = 1980, end = 2017)

testing\_data <- window(ben\_ts, start = 2018, end = 2017 + h)

arima\_model <- auto.arima(training\_data)

arima\_forecast <- forecast(arima\_model, h = length(testing\_data))$mean

ets\_model <- ets(training\_data)

ets\_forecast <- forecast(ets\_model, h = length(testing\_data))$mean

mlp\_model <- mlp(training\_data)

mlp\_forecast <- forecast(mlp\_model, h = length(testing\_data))$mean

train\_df <- data.frame(

ds = seq.Date(from = as.Date("1980-01-01"), by = "year", length.out = length(training\_data)),

y = as.numeric(training\_data))

prophet\_model <- prophet(train\_df, yearly.seasonality = FALSE)

future <- data.frame(ds = seq.Date(from = max(train\_df$ds) + 1, by = "year", length.out = length(testing\_data)))

prophet\_forecast <- predict(prophet\_model, future)$yhat

lagged\_data <- data.frame(Target = training\_data[-1], Lag1 = training\_data[-length(training\_data)])

xgb\_model <- xgboost(data = as.matrix(lagged\_data$Lag1), label = lagged\_data$Target, nrounds = 100)

xgb\_forecast <- numeric(length(testing\_data))

last\_value <- training\_data[length(training\_data)]

for (i in 1:length(testing\_data)) {

xgb\_forecast[i] <- predict(xgb\_model, newdata = as.matrix(last\_value))

last\_value <- xgb\_forecast[i]}

hybrid\_forecast\_function <- function(weights, f1, f2) {

weights[1] \* as.numeric(f1) + weights[2] \* as.numeric(f2)}

optimize\_weights <- function(f1, f2, actual) {ga\_model <- ga(type = "real-valued", fitness = function(w) -sqrt(mean((actual - hybrid\_forecast\_function(w, f1, f2))^2)), lower = c(0, 0), upper = c(1, 1), popSize = 50, maxiter = 100)

return(ga\_model@solution)}

weights <- optimize\_weights(arima\_forecast, ets\_forecast, testing\_data)

arima\_ets\_forecast <- hybrid\_forecast\_function(weights, arima\_forecast, ets\_forecast)

weights <- optimize\_weights(arima\_forecast, mlp\_forecast, testing\_data)

arima\_mlp\_forecast <- hybrid\_forecast\_function(weights, arima\_forecast, mlp\_forecast)

weights <- optimize\_weights(arima\_forecast, prophet\_forecast, testing\_data)

arima\_prophet\_forecast <- hybrid\_forecast\_function(weights, arima\_forecast, prophet\_forecast)

weights <- optimize\_weights(arima\_forecast, xgb\_forecast, testing\_data)

arima\_xgb\_forecast <- hybrid\_forecast\_function(weights, arima\_forecast, xgb\_forecast)

calculate\_rmse <- function(actual, forecast) {

sqrt(mean((actual - forecast)^2))}

models <- list(

ARIMA = arima\_forecast,

ETS = ets\_forecast,

MLP = mlp\_forecast,

Prophet = prophet\_forecast,

XGBoost = xgb\_forecast,

ARIMA\_ETS = arima\_ets\_forecast,

ARIMA\_MLP = arima\_mlp\_forecast,

ARIMA\_Prophet = arima\_prophet\_forecast,

ARIMA\_XGBoost = arima\_xgb\_forecast)

for (model\_name in names(models)) {

rmse <- calculate\_rmse(testing\_data, models[[model\_name]])

results <- rbind(results, data.frame(Horizon = h, Model = model\_name, RMSE = rmse))}}

print("Sensitivity Analysis Results:")

print(results)

ggplot(results, aes(x = Horizon, y = RMSE, color = Model, group = Model)) +geom\_line() +geom\_point() +labs(title = "BEN", y = "RMSE", x = "Forecast Horizon (Years)") +theme\_minimal()

**# Sensitivity Analysis for BFA UN-5 Female Mortality using Time Horizon: 3, 5, and 7 years**

set.seed(123)

BFA = c(228.65, 223.26, 218.56, 214.17, 210.15, 205.71, 201.18, 196.93, 193.52, 191.27, 190.12, 189.81, 189.83, 189.86, 189.34, 187.98, 185.44, 182.38, 178.86, 175.28, 171.47, 167.39, 162.76, 157.8, 152.28, 146.03, 139.58, 133.26, 127.09, 121.28, 115.96, 111.14, 106.71, 102.52, 98.779, 95.212, 91.827, 88.466, 85.28, 82.335, 79.251, 76.574, 73.887)

bfa\_ts <- ts(BFA, start = 1980, frequency = 1)

horizons <- c(3, 5, 7)

results <- data.frame(Horizon = integer(), Model = character(), RMSE = numeric())

for (h in horizons) {

training\_data <- window(bfa\_ts, start = 1980, end = 2017)

testing\_data <- window(bfa\_ts, start = 2018, end = 2017 + h)

arima\_model <- auto.arima(training\_data)

arima\_forecast <- forecast(arima\_model, h = length(testing\_data))$mean

ets\_model <- ets(training\_data)

ets\_forecast <- forecast(ets\_model, h = length(testing\_data))$mean

mlp\_model <- mlp(training\_data)

mlp\_forecast <- forecast(mlp\_model, h = length(testing\_data))$mean

train\_df <- data.frame(

ds = seq.Date(from = as.Date("1980-01-01"), by = "year", length.out = length(training\_data)),

y = as.numeric(training\_data))

prophet\_model <- prophet(train\_df, yearly.seasonality = FALSE)

future <- data.frame(ds = seq.Date(from = max(train\_df$ds) + 1, by = "year", length.out = length(testing\_data)))

prophet\_forecast <- predict(prophet\_model, future)$yhat

lagged\_data <- data.frame(Target = training\_data[-1], Lag1 = training\_data[-length(training\_data)])

xgb\_model <- xgboost(data = as.matrix(lagged\_data$Lag1), label = lagged\_data$Target, nrounds = 100)

xgb\_forecast <- numeric(length(testing\_data))

last\_value <- training\_data[length(training\_data)]

for (i in 1:length(testing\_data)) {

xgb\_forecast[i] <- predict(xgb\_model, newdata = as.matrix(last\_value))

last\_value <- xgb\_forecast[i]}

hybrid\_forecast\_function <- function(weights, f1, f2) {

weights[1] \* as.numeric(f1) + weights[2] \* as.numeric(f2)}

optimize\_weights <- function(f1, f2, actual) {ga\_model <- ga(type = "real-valued", fitness = function(w) -sqrt(mean((actual - hybrid\_forecast\_function(w, f1, f2))^2)), lower = c(0, 0), upper = c(1, 1), popSize = 50, maxiter = 100)

return(ga\_model@solution)}

weights <- optimize\_weights(arima\_forecast, ets\_forecast, testing\_data)

arima\_ets\_forecast <- hybrid\_forecast\_function(weights, arima\_forecast, ets\_forecast)

weights <- optimize\_weights(arima\_forecast, mlp\_forecast, testing\_data)

arima\_mlp\_forecast <- hybrid\_forecast\_function(weights, arima\_forecast, mlp\_forecast)

weights <- optimize\_weights(arima\_forecast, prophet\_forecast, testing\_data)

arima\_prophet\_forecast <- hybrid\_forecast\_function(weights, arima\_forecast, prophet\_forecast)

weights <- optimize\_weights(arima\_forecast, xgb\_forecast, testing\_data)

arima\_xgb\_forecast <- hybrid\_forecast\_function(weights, arima\_forecast, xgb\_forecast)

calculate\_rmse <- function(actual, forecast) {

sqrt(mean((actual - forecast)^2))}

models <- list(

ARIMA = arima\_forecast,

ETS = ets\_forecast,

MLP = mlp\_forecast,

Prophet = prophet\_forecast,

XGBoost = xgb\_forecast,

ARIMA\_ETS = arima\_ets\_forecast,

ARIMA\_MLP = arima\_mlp\_forecast,

ARIMA\_Prophet = arima\_prophet\_forecast,

ARIMA\_XGBoost = arima\_xgb\_forecast)

for (model\_name in names(models)) {

rmse <- calculate\_rmse(testing\_data, models[[model\_name]])

results <- rbind(results, data.frame(Horizon = h, Model = model\_name, RMSE = rmse))}}

print("Sensitivity Analysis Results:")

print(results)

ggplot(results, aes(x = Horizon, y = RMSE, color = Model, group = Model)) +geom\_line() +geom\_point() +labs(title = "BFA", y = "RMSE", x = "Forecast Horizon (Years)") +theme\_minimal()

**# Sensitivity Analysis for GIN UN-5 Female Mortality using Time Horizon: 3, 5, and 7 years**

set.seed(123)

GIN = c(272.33, 267.33, 262.56, 258.23, 253.92, 249.54, 245, 240.39, 235.54, 230.39, 224.85, 218.72, 212.52, 205.69, 198.53, 191.36, 184.22, 177.38, 170.47, 163.59, 156.77, 150, 143.7, 137.87, 132.57, 128.04, 124.1, 120.64, 117.67, 115.07, 112.74, 110.77, 108.78, 106.8, 105.62, 103.83, 101.39, 99.583, 97.695, 95.579, 93.357, 90.876, 88.443)

gin\_ts <- ts(GIN, start = 1980, frequency = 1)

horizons <- c(3, 5, 7)

results <- data.frame(Horizon = integer(), Model = character(), RMSE = numeric())

for (h in horizons) {

training\_data <- window(gin\_ts, start = 1980, end = 2017)

testing\_data <- window(gin\_ts, start = 2018, end = 2017 + h)

arima\_model <- auto.arima(training\_data)

arima\_forecast <- forecast(arima\_model, h = length(testing\_data))$mean

ets\_model <- ets(training\_data)

ets\_forecast <- forecast(ets\_model, h = length(testing\_data))$mean

mlp\_model <- mlp(training\_data)

mlp\_forecast <- forecast(mlp\_model, h = length(testing\_data))$mean

train\_df <- data.frame(

ds = seq.Date(from = as.Date("1980-01-01"), by = "year", length.out = length(training\_data)),

y = as.numeric(training\_data))

prophet\_model <- prophet(train\_df, yearly.seasonality = FALSE)

future <- data.frame(ds = seq.Date(from = max(train\_df$ds) + 1, by = "year", length.out = length(testing\_data)))

prophet\_forecast <- predict(prophet\_model, future)$yhat

lagged\_data <- data.frame(Target = training\_data[-1], Lag1 = training\_data[-length(training\_data)])

xgb\_model <- xgboost(data = as.matrix(lagged\_data$Lag1), label = lagged\_data$Target, nrounds = 100)

xgb\_forecast <- numeric(length(testing\_data))

last\_value <- training\_data[length(training\_data)]

for (i in 1:length(testing\_data)) {

xgb\_forecast[i] <- predict(xgb\_model, newdata = as.matrix(last\_value))

last\_value <- xgb\_forecast[i]}

hybrid\_forecast\_function <- function(weights, f1, f2) {

weights[1] \* as.numeric(f1) + weights[2] \* as.numeric(f2)}

optimize\_weights <- function(f1, f2, actual) {ga\_model <- ga(type = "real-valued", fitness = function(w) -sqrt(mean((actual - hybrid\_forecast\_function(w, f1, f2))^2)), lower = c(0, 0), upper = c(1, 1), popSize = 50, maxiter = 100)

return(ga\_model@solution)}

weights <- optimize\_weights(arima\_forecast, ets\_forecast, testing\_data)

arima\_ets\_forecast <- hybrid\_forecast\_function(weights, arima\_forecast, ets\_forecast)

weights <- optimize\_weights(arima\_forecast, mlp\_forecast, testing\_data)

arima\_mlp\_forecast <- hybrid\_forecast\_function(weights, arima\_forecast, mlp\_forecast)

weights <- optimize\_weights(arima\_forecast, prophet\_forecast, testing\_data)

arima\_prophet\_forecast <- hybrid\_forecast\_function(weights, arima\_forecast, prophet\_forecast)

weights <- optimize\_weights(arima\_forecast, xgb\_forecast, testing\_data)

arima\_xgb\_forecast <- hybrid\_forecast\_function(weights, arima\_forecast, xgb\_forecast)

calculate\_rmse <- function(actual, forecast) {

sqrt(mean((actual - forecast)^2))}

models <- list(

ARIMA = arima\_forecast,

ETS = ets\_forecast,

MLP = mlp\_forecast,

Prophet = prophet\_forecast,

XGBoost = xgb\_forecast,

ARIMA\_ETS = arima\_ets\_forecast,

ARIMA\_MLP = arima\_mlp\_forecast,

ARIMA\_Prophet = arima\_prophet\_forecast,

ARIMA\_XGBoost = arima\_xgb\_forecast)

for (model\_name in names(models)) {

rmse <- calculate\_rmse(testing\_data, models[[model\_name]])

results <- rbind(results, data.frame(Horizon = h, Model = model\_name, RMSE = rmse))}}

print("Sensitivity Analysis Results:")

print(results)

ggplot(results, aes(x = Horizon, y = RMSE, color = Model, group = Model)) +geom\_line() +geom\_point() +labs(title = "GIN", y = "RMSE", x = "Forecast Horizon (Years)") +theme\_minimal()

**# Sensitivity Analysis for GNB UN-5 Female Mortality using Time Horizon: 3, 5, and 7 years**

set.seed(123)

GNB = c(232.88, 232.88, 232.88, 232.88, 232.88, 232.88, 228.23, 224.17, 219.73, 215.38, 210.8, 206.29, 201.92, 197.26, 192.53, 187.82, 182.91, 177.83, 172.73, 167.88, 162.6, 157.18, 151.7, 146.05, 140.18, 134.16, 128.17, 122.34, 116.66, 111.05, 105.44, 100.34, 95.704, 91.589, 87.829, 84.501, 81.364, 78.55, 75.712, 73.013, 70.49, 68.098, 65.769)

gnb\_ts <- ts(GNB, start = 1980, frequency = 1)

horizons <- c(3, 5, 7)

results <- data.frame(Horizon = integer(), Model = character(), RMSE = numeric())

for (h in horizons) {

training\_data <- window(gnb\_ts, start = 1980, end = 2017)

testing\_data <- window(gnb\_ts, start = 2018, end = 2017 + h)

arima\_model <- auto.arima(training\_data)

arima\_forecast <- forecast(arima\_model, h = length(testing\_data))$mean

ets\_model <- ets(training\_data)

ets\_forecast <- forecast(ets\_model, h = length(testing\_data))$mean

mlp\_model <- mlp(training\_data)

mlp\_forecast <- forecast(mlp\_model, h = length(testing\_data))$mean

train\_df <- data.frame(

ds = seq.Date(from = as.Date("1980-01-01"), by = "year", length.out = length(training\_data)),

y = as.numeric(training\_data))

prophet\_model <- prophet(train\_df, yearly.seasonality = FALSE)

future <- data.frame(ds = seq.Date(from = max(train\_df$ds) + 1, by = "year", length.out = length(testing\_data)))

prophet\_forecast <- predict(prophet\_model, future)$yhat

lagged\_data <- data.frame(Target = training\_data[-1], Lag1 = training\_data[-length(training\_data)])

xgb\_model <- xgboost(data = as.matrix(lagged\_data$Lag1), label = lagged\_data$Target, nrounds = 100)

xgb\_forecast <- numeric(length(testing\_data))

last\_value <- training\_data[length(training\_data)]

for (i in 1:length(testing\_data)) {

xgb\_forecast[i] <- predict(xgb\_model, newdata = as.matrix(last\_value))

last\_value <- xgb\_forecast[i]}

hybrid\_forecast\_function <- function(weights, f1, f2) {

weights[1] \* as.numeric(f1) + weights[2] \* as.numeric(f2)}

optimize\_weights <- function(f1, f2, actual) {ga\_model <- ga(type = "real-valued", fitness = function(w) -sqrt(mean((actual - hybrid\_forecast\_function(w, f1, f2))^2)), lower = c(0, 0), upper = c(1, 1), popSize = 50, maxiter = 100)

return(ga\_model@solution)}

weights <- optimize\_weights(arima\_forecast, ets\_forecast, testing\_data)

arima\_ets\_forecast <- hybrid\_forecast\_function(weights, arima\_forecast, ets\_forecast)

weights <- optimize\_weights(arima\_forecast, mlp\_forecast, testing\_data)

arima\_mlp\_forecast <- hybrid\_forecast\_function(weights, arima\_forecast, mlp\_forecast)

weights <- optimize\_weights(arima\_forecast, prophet\_forecast, testing\_data)

arima\_prophet\_forecast <- hybrid\_forecast\_function(weights, arima\_forecast, prophet\_forecast)

weights <- optimize\_weights(arima\_forecast, xgb\_forecast, testing\_data)

arima\_xgb\_forecast <- hybrid\_forecast\_function(weights, arima\_forecast, xgb\_forecast)

calculate\_rmse <- function(actual, forecast) {

sqrt(mean((actual - forecast)^2))}

models <- list(

ARIMA = arima\_forecast,

ETS = ets\_forecast,

MLP = mlp\_forecast,

Prophet = prophet\_forecast,

XGBoost = xgb\_forecast,

ARIMA\_ETS = arima\_ets\_forecast,

ARIMA\_MLP = arima\_mlp\_forecast,

ARIMA\_Prophet = arima\_prophet\_forecast,

ARIMA\_XGBoost = arima\_xgb\_forecast)

for (model\_name in names(models)) {

rmse <- calculate\_rmse(testing\_data, models[[model\_name]])

results <- rbind(results, data.frame(Horizon = h, Model = model\_name, RMSE = rmse))}}

print("Sensitivity Analysis Results:")

print(results)

ggplot(results, aes(x = Horizon, y = RMSE, color = Model, group = Model)) +geom\_line() +geom\_point() +labs(title = "GNB", y = "RMSE", x = "Forecast Horizon (Years)") +theme\_minimal()

**# Sensitivity Analysis for LBR UN-5 Female Mortality using Time Horizon: 3, 5, and 7 years**

set.seed(123)

LBR = c(228.98, 226.3, 224.25, 223.28, 223.55, 225.45, 228.95, 234.01, 239.61, 245.06, 249.11, 251.1, 250.54, 247.22, 241.3, 233.24, 224, 213.9, 202.96, 190.85, 178.46, 165.74, 153.32, 141.34, 130.39, 120.84, 112.66, 106.02, 100.52, 96.11, 92.504, 89.514, 87.065, 84.947, 85.221, 81.5, 79.602, 77.82, 75.91, 73.811, 71.65, 69.445, 67.225)

lbr\_ts <- ts(LBR, start = 1980, frequency = 1)

horizons <- c(3, 5, 7)

results <- data.frame(Horizon = integer(), Model = character(), RMSE = numeric())

for (h in horizons) {

training\_data <- window(lbr\_ts, start = 1980, end = 2017)

testing\_data <- window(lbr\_ts, start = 2018, end = 2017 + h)

arima\_model <- auto.arima(training\_data)

arima\_forecast <- forecast(arima\_model, h = length(testing\_data))$mean

ets\_model <- ets(training\_data)

ets\_forecast <- forecast(ets\_model, h = length(testing\_data))$mean

mlp\_model <- mlp(training\_data)

mlp\_forecast <- forecast(mlp\_model, h = length(testing\_data))$mean

train\_df <- data.frame(

ds = seq.Date(from = as.Date("1980-01-01"), by = "year", length.out = length(training\_data)),

y = as.numeric(training\_data))

prophet\_model <- prophet(train\_df, yearly.seasonality = FALSE)

future <- data.frame(ds = seq.Date(from = max(train\_df$ds) + 1, by = "year", length.out = length(testing\_data)))

prophet\_forecast <- predict(prophet\_model, future)$yhat

lagged\_data <- data.frame(Target = training\_data[-1], Lag1 = training\_data[-length(training\_data)])

xgb\_model <- xgboost(data = as.matrix(lagged\_data$Lag1), label = lagged\_data$Target, nrounds = 100)

xgb\_forecast <- numeric(length(testing\_data))

last\_value <- training\_data[length(training\_data)]

for (i in 1:length(testing\_data)) {

xgb\_forecast[i] <- predict(xgb\_model, newdata = as.matrix(last\_value))

last\_value <- xgb\_forecast[i]}

hybrid\_forecast\_function <- function(weights, f1, f2) {

weights[1] \* as.numeric(f1) + weights[2] \* as.numeric(f2)}

optimize\_weights <- function(f1, f2, actual) {ga\_model <- ga(type = "real-valued", fitness = function(w) -sqrt(mean((actual - hybrid\_forecast\_function(w, f1, f2))^2)), lower = c(0, 0), upper = c(1, 1), popSize = 50, maxiter = 100)

return(ga\_model@solution)}

weights <- optimize\_weights(arima\_forecast, ets\_forecast, testing\_data)

arima\_ets\_forecast <- hybrid\_forecast\_function(weights, arima\_forecast, ets\_forecast)

weights <- optimize\_weights(arima\_forecast, mlp\_forecast, testing\_data)

arima\_mlp\_forecast <- hybrid\_forecast\_function(weights, arima\_forecast, mlp\_forecast)

weights <- optimize\_weights(arima\_forecast, prophet\_forecast, testing\_data)

arima\_prophet\_forecast <- hybrid\_forecast\_function(weights, arima\_forecast, prophet\_forecast)

weights <- optimize\_weights(arima\_forecast, xgb\_forecast, testing\_data)

arima\_xgb\_forecast <- hybrid\_forecast\_function(weights, arima\_forecast, xgb\_forecast)

calculate\_rmse <- function(actual, forecast) {

sqrt(mean((actual - forecast)^2))}

models <- list(

ARIMA = arima\_forecast,

ETS = ets\_forecast,

MLP = mlp\_forecast,

Prophet = prophet\_forecast,

XGBoost = xgb\_forecast,

ARIMA\_ETS = arima\_ets\_forecast,

ARIMA\_MLP = arima\_mlp\_forecast,

ARIMA\_Prophet = arima\_prophet\_forecast,

ARIMA\_XGBoost = arima\_xgb\_forecast)

for (model\_name in names(models)) {

rmse <- calculate\_rmse(testing\_data, models[[model\_name]])

results <- rbind(results, data.frame(Horizon = h, Model = model\_name, RMSE = rmse))}}

print("Sensitivity Analysis Results:")

print(results)

ggplot(results, aes(x = Horizon, y = RMSE, color = Model, group = Model)) +geom\_line() +geom\_point() +labs(title = "LBR", y = "RMSE", x = "Forecast Horizon (Years)") +theme\_minimal()

**# Sensitivity Analysis for MLI UN-5 Female Mortality using Time Horizon: 3, 5, and 7 years**

set.seed(123)

MLI = c(292.95, 284.67, 276.5, 268.41, 260.53, 253.26, 246.33, 239.42, 232.86, 226.94, 221.73, 217.18, 213.02, 209.02, 205.78, 202.37, 198.87, 194.94, 190.54, 185.48, 179.95, 173.91, 167.91, 162.05, 156.47, 150.99, 145.78, 140.87, 136.17, 131.78, 127.59, 123.64, 119.82, 116.11, 112.72, 109.44, 106.41, 103.48, 100.51, 97.268, 94.331, 91.433, 88.429)

mli\_ts <- ts(MLI, start = 1980, frequency = 1)

horizons <- c(3, 5, 7)

results <- data.frame(Horizon = integer(), Model = character(), RMSE = numeric())

for (h in horizons) {

training\_data <- window(mli\_ts, start = 1980, end = 2017)

testing\_data <- window(mli\_ts, start = 2018, end = 2017 + h)

arima\_model <- auto.arima(training\_data)

arima\_forecast <- forecast(arima\_model, h = length(testing\_data))$mean

ets\_model <- ets(training\_data)

ets\_forecast <- forecast(ets\_model, h = length(testing\_data))$mean

mlp\_model <- mlp(training\_data)

mlp\_forecast <- forecast(mlp\_model, h = length(testing\_data))$mean

train\_df <- data.frame(

ds = seq.Date(from = as.Date("1980-01-01"), by = "year", length.out = length(training\_data)),

y = as.numeric(training\_data))

prophet\_model <- prophet(train\_df, yearly.seasonality = FALSE)

future <- data.frame(ds = seq.Date(from = max(train\_df$ds) + 1, by = "year", length.out = length(testing\_data)))

prophet\_forecast <- predict(prophet\_model, future)$yhat

lagged\_data <- data.frame(Target = training\_data[-1], Lag1 = training\_data[-length(training\_data)])

xgb\_model <- xgboost(data = as.matrix(lagged\_data$Lag1), label = lagged\_data$Target, nrounds = 100)

xgb\_forecast <- numeric(length(testing\_data))

last\_value <- training\_data[length(training\_data)]

for (i in 1:length(testing\_data)) {

xgb\_forecast[i] <- predict(xgb\_model, newdata = as.matrix(last\_value))

last\_value <- xgb\_forecast[i]}

hybrid\_forecast\_function <- function(weights, f1, f2) {

weights[1] \* as.numeric(f1) + weights[2] \* as.numeric(f2)}

optimize\_weights <- function(f1, f2, actual) {ga\_model <- ga(type = "real-valued", fitness = function(w) -sqrt(mean((actual - hybrid\_forecast\_function(w, f1, f2))^2)), lower = c(0, 0), upper = c(1, 1), popSize = 50, maxiter = 100)

return(ga\_model@solution)}

weights <- optimize\_weights(arima\_forecast, ets\_forecast, testing\_data)

arima\_ets\_forecast <- hybrid\_forecast\_function(weights, arima\_forecast, ets\_forecast)

weights <- optimize\_weights(arima\_forecast, mlp\_forecast, testing\_data)

arima\_mlp\_forecast <- hybrid\_forecast\_function(weights, arima\_forecast, mlp\_forecast)

weights <- optimize\_weights(arima\_forecast, prophet\_forecast, testing\_data)

arima\_prophet\_forecast <- hybrid\_forecast\_function(weights, arima\_forecast, prophet\_forecast)

weights <- optimize\_weights(arima\_forecast, xgb\_forecast, testing\_data)

arima\_xgb\_forecast <- hybrid\_forecast\_function(weights, arima\_forecast, xgb\_forecast)

calculate\_rmse <- function(actual, forecast) {

sqrt(mean((actual - forecast)^2))}

models <- list(

ARIMA = arima\_forecast,

ETS = ets\_forecast,

MLP = mlp\_forecast,

Prophet = prophet\_forecast,

XGBoost = xgb\_forecast,

ARIMA\_ETS = arima\_ets\_forecast,

ARIMA\_MLP = arima\_mlp\_forecast,

ARIMA\_Prophet = arima\_prophet\_forecast,

ARIMA\_XGBoost = arima\_xgb\_forecast)

for (model\_name in names(models)) {

rmse <- calculate\_rmse(testing\_data, models[[model\_name]])

results <- rbind(results, data.frame(Horizon = h, Model = model\_name, RMSE = rmse))}}

print("Sensitivity Analysis Results:")

print(results)

ggplot(results, aes(x = Horizon, y = RMSE, color = Model, group = Model)) +geom\_line() +geom\_point() +labs(title = "MLI", y = "RMSE", x = "Forecast Horizon (Years)") +theme\_minimal()

**# Sensitivity Analysis for MRT UN-5 Female Mortality using Time Horizon: 3, 5, and 7 years**

set.seed(123)

MRT = c(156.07, 151.38, 146.63, 141.46, 135.75, 129.43, 123.35, 117.88, 113.49, 110.31, 108.17, 106.82, 105.89, 105.12, 104.4, 103.38, 102.01, 100.2, 97.791, 94.58, 90.591, 86.048, 81.117, 75.847, 70.779, 66.011, 61.744, 58.259, 55.392, 53.096, 51.267, 49.739, 48.388, 47.076, 45.774, 44.408, 43.081, 41.683, 40.286, 38.994, 37.638, 36.412, 35.12)

mrt\_ts <- ts(MRT, start = 1980, frequency = 1)

horizons <- c(3, 5, 7)

results <- data.frame(Horizon = integer(), Model = character(), RMSE = numeric())

for (h in horizons) {

training\_data <- window(mrt\_ts, start = 1980, end = 2017)

testing\_data <- window(mrt\_ts, start = 2018, end = 2017 + h)

arima\_model <- auto.arima(training\_data)

arima\_forecast <- forecast(arima\_model, h = length(testing\_data))$mean

ets\_model <- ets(training\_data)

ets\_forecast <- forecast(ets\_model, h = length(testing\_data))$mean

mlp\_model <- mlp(training\_data)

mlp\_forecast <- forecast(mlp\_model, h = length(testing\_data))$mean

train\_df <- data.frame(

ds = seq.Date(from = as.Date("1980-01-01"), by = "year", length.out = length(training\_data)),

y = as.numeric(training\_data))

prophet\_model <- prophet(train\_df, yearly.seasonality = FALSE)

future <- data.frame(ds = seq.Date(from = max(train\_df$ds) + 1, by = "year", length.out = length(testing\_data)))

prophet\_forecast <- predict(prophet\_model, future)$yhat

lagged\_data <- data.frame(Target = training\_data[-1], Lag1 = training\_data[-length(training\_data)])

xgb\_model <- xgboost(data = as.matrix(lagged\_data$Lag1), label = lagged\_data$Target, nrounds = 100)

xgb\_forecast <- numeric(length(testing\_data))

last\_value <- training\_data[length(training\_data)]

for (i in 1:length(testing\_data)) {

xgb\_forecast[i] <- predict(xgb\_model, newdata = as.matrix(last\_value))

last\_value <- xgb\_forecast[i]}

hybrid\_forecast\_function <- function(weights, f1, f2) {

weights[1] \* as.numeric(f1) + weights[2] \* as.numeric(f2)}

optimize\_weights <- function(f1, f2, actual) {ga\_model <- ga(type = "real-valued", fitness = function(w) -sqrt(mean((actual - hybrid\_forecast\_function(w, f1, f2))^2)), lower = c(0, 0), upper = c(1, 1), popSize = 50, maxiter = 100)

return(ga\_model@solution)}

weights <- optimize\_weights(arima\_forecast, ets\_forecast, testing\_data)

arima\_ets\_forecast <- hybrid\_forecast\_function(weights, arima\_forecast, ets\_forecast)

weights <- optimize\_weights(arima\_forecast, mlp\_forecast, testing\_data)

arima\_mlp\_forecast <- hybrid\_forecast\_function(weights, arima\_forecast, mlp\_forecast)

weights <- optimize\_weights(arima\_forecast, prophet\_forecast, testing\_data)

arima\_prophet\_forecast <- hybrid\_forecast\_function(weights, arima\_forecast, prophet\_forecast)

weights <- optimize\_weights(arima\_forecast, xgb\_forecast, testing\_data)

arima\_xgb\_forecast <- hybrid\_forecast\_function(weights, arima\_forecast, xgb\_forecast)

calculate\_rmse <- function(actual, forecast) {

sqrt(mean((actual - forecast)^2))}

models <- list(

ARIMA = arima\_forecast,

ETS = ets\_forecast,

MLP = mlp\_forecast,

Prophet = prophet\_forecast,

XGBoost = xgb\_forecast,

ARIMA\_ETS = arima\_ets\_forecast,

ARIMA\_MLP = arima\_mlp\_forecast,

ARIMA\_Prophet = arima\_prophet\_forecast,

ARIMA\_XGBoost = arima\_xgb\_forecast)

for (model\_name in names(models)) {

rmse <- calculate\_rmse(testing\_data, models[[model\_name]])

results <- rbind(results, data.frame(Horizon = h, Model = model\_name, RMSE = rmse))}}

print("Sensitivity Analysis Results:")

print(results)

ggplot(results, aes(x = Horizon, y = RMSE, color = Model, group = Model)) +geom\_line() +geom\_point() +labs(title = "MRT", y = "RMSE", x = "Forecast Horizon (Years)") +theme\_minimal()

**# Sensitivity Analysis for CPV UN-5 Female Mortality using Time Horizon: 3, 5, and 7 years**

set.seed(123)

CPV = c(74.54, 73.49, 72.86, 72.34, 71.42, 69.77, 67.15, 63.79, 60.32, 57.1, 54.53, 52.83, 51.95, 51.37, 50.61, 49.23, 46.77, 43.35, 39.4, 35.28, 31.59, 28.73, 26.78, 25.57, 24.98, 24.8, 24.82, 24.75, 24.48, 23.9, 22.99, 21.85, 20.57, 19.22, 17.85, 16.53, 15.35, 14.31, 13.44, 12.73, 12.11, 11.6, 11.13)

cpv\_ts <- ts(CPV, start = 1980, frequency = 1)

horizons <- c(3, 5, 7)

results <- data.frame(Horizon = integer(), Model = character(), RMSE = numeric())

for (h in horizons) {

training\_data <- window(cpv\_ts, start = 1980, end = 2017)

testing\_data <- window(cpv\_ts, start = 2018, end = 2017 + h)

arima\_model <- auto.arima(training\_data)

arima\_forecast <- forecast(arima\_model, h = length(testing\_data))$mean

ets\_model <- ets(training\_data)

ets\_forecast <- forecast(ets\_model, h = length(testing\_data))$mean

mlp\_model <- mlp(training\_data)

mlp\_forecast <- forecast(mlp\_model, h = length(testing\_data))$mean

train\_df <- data.frame(

ds = seq.Date(from = as.Date("1980-01-01"), by = "year", length.out = length(training\_data)),

y = as.numeric(training\_data))

prophet\_model <- prophet(train\_df, yearly.seasonality = FALSE)

future <- data.frame(ds = seq.Date(from = max(train\_df$ds) + 1, by = "year", length.out = length(testing\_data)))

prophet\_forecast <- predict(prophet\_model, future)$yhat

lagged\_data <- data.frame(Target = training\_data[-1], Lag1 = training\_data[-length(training\_data)])

xgb\_model <- xgboost(data = as.matrix(lagged\_data$Lag1), label = lagged\_data$Target, nrounds = 100)

xgb\_forecast <- numeric(length(testing\_data))

last\_value <- training\_data[length(training\_data)]

for (i in 1:length(testing\_data)) {

xgb\_forecast[i] <- predict(xgb\_model, newdata = as.matrix(last\_value))

last\_value <- xgb\_forecast[i]}

hybrid\_forecast\_function <- function(weights, f1, f2) {

weights[1] \* as.numeric(f1) + weights[2] \* as.numeric(f2)}

optimize\_weights <- function(f1, f2, actual) {ga\_model <- ga(type = "real-valued", fitness = function(w) -sqrt(mean((actual - hybrid\_forecast\_function(w, f1, f2))^2)), lower = c(0, 0), upper = c(1, 1), popSize = 50, maxiter = 100)

return(ga\_model@solution)}

weights <- optimize\_weights(arima\_forecast, ets\_forecast, testing\_data)

arima\_ets\_forecast <- hybrid\_forecast\_function(weights, arima\_forecast, ets\_forecast)

weights <- optimize\_weights(arima\_forecast, mlp\_forecast, testing\_data)

arima\_mlp\_forecast <- hybrid\_forecast\_function(weights, arima\_forecast, mlp\_forecast)

weights <- optimize\_weights(arima\_forecast, prophet\_forecast, testing\_data)

arima\_prophet\_forecast <- hybrid\_forecast\_function(weights, arima\_forecast, prophet\_forecast)

weights <- optimize\_weights(arima\_forecast, xgb\_forecast, testing\_data)

arima\_xgb\_forecast <- hybrid\_forecast\_function(weights, arima\_forecast, xgb\_forecast)

calculate\_rmse <- function(actual, forecast) {

sqrt(mean((actual - forecast)^2))}

models <- list(

ARIMA = arima\_forecast,

ETS = ets\_forecast,

MLP = mlp\_forecast,

Prophet = prophet\_forecast,

XGBoost = xgb\_forecast,

ARIMA\_ETS = arima\_ets\_forecast,

ARIMA\_MLP = arima\_mlp\_forecast,

ARIMA\_Prophet = arima\_prophet\_forecast,

ARIMA\_XGBoost = arima\_xgb\_forecast)

for (model\_name in names(models)) {

rmse <- calculate\_rmse(testing\_data, models[[model\_name]])

results <- rbind(results, data.frame(Horizon = h, Model = model\_name, RMSE = rmse))}}

print("Sensitivity Analysis Results:")

print(results)

ggplot(results, aes(x = Horizon, y = RMSE, color = Model, group = Model)) +geom\_line() +geom\_point() +labs(title = "CPV", y = "RMSE", x = "Forecast Horizon (Years)") +theme\_minimal()

**# Sensitivity Analysis for TCD UN-5 Female Mortality using Time Horizon: 3, 5, and 7 years**

set.seed(123)

TCD = c(229.48, 227.27, 224.63, 222.03, 219.39, 216.62, 213.6, 210.6, 207.56, 204.64, 201.84, 198.99, 196.39, 193.96, 191.55, 189.37, 187.19, 184.47, 181.58, 178.3, 175.03, 171.55, 168.08, 164.61, 161.35, 157.94, 154.61, 150.83, 147.11, 143.2, 139.23, 135.35, 131.58, 127.95, 124.45, 120.81, 117.29, 113.58, 109.91, 106.2, 102.85, 99.404, 96.231)

tcd\_ts <- ts(TCD, start = 1980, frequency = 1)

horizons <- c(3, 5, 7)

results <- data.frame(Horizon = integer(), Model = character(), RMSE = numeric())

for (h in horizons) {

training\_data <- window(tcd\_ts, start = 1980, end = 2017)

testing\_data <- window(tcd\_ts, start = 2018, end = 2017 + h)

arima\_model <- auto.arima(training\_data)

arima\_forecast <- forecast(arima\_model, h = length(testing\_data))$mean

ets\_model <- ets(training\_data)

ets\_forecast <- forecast(ets\_model, h = length(testing\_data))$mean

mlp\_model <- mlp(training\_data)

mlp\_forecast <- forecast(mlp\_model, h = length(testing\_data))$mean

train\_df <- data.frame(

ds = seq.Date(from = as.Date("1980-01-01"), by = "year", length.out = length(training\_data)),

y = as.numeric(training\_data))

prophet\_model <- prophet(train\_df, yearly.seasonality = FALSE)

future <- data.frame(ds = seq.Date(from = max(train\_df$ds) + 1, by = "year", length.out = length(testing\_data)))

prophet\_forecast <- predict(prophet\_model, future)$yhat

lagged\_data <- data.frame(Target = training\_data[-1], Lag1 = training\_data[-length(training\_data)])

xgb\_model <- xgboost(data = as.matrix(lagged\_data$Lag1), label = lagged\_data$Target, nrounds = 100)

xgb\_forecast <- numeric(length(testing\_data))

last\_value <- training\_data[length(training\_data)]

for (i in 1:length(testing\_data)) {

xgb\_forecast[i] <- predict(xgb\_model, newdata = as.matrix(last\_value))

last\_value <- xgb\_forecast[i]}

hybrid\_forecast\_function <- function(weights, f1, f2) {

weights[1] \* as.numeric(f1) + weights[2] \* as.numeric(f2)}

optimize\_weights <- function(f1, f2, actual) {ga\_model <- ga(type = "real-valued", fitness = function(w) -sqrt(mean((actual - hybrid\_forecast\_function(w, f1, f2))^2)), lower = c(0, 0), upper = c(1, 1), popSize = 50, maxiter = 100)

return(ga\_model@solution)}

weights <- optimize\_weights(arima\_forecast, ets\_forecast, testing\_data)

arima\_ets\_forecast <- hybrid\_forecast\_function(weights, arima\_forecast, ets\_forecast)

weights <- optimize\_weights(arima\_forecast, mlp\_forecast, testing\_data)

arima\_mlp\_forecast <- hybrid\_forecast\_function(weights, arima\_forecast, mlp\_forecast)

weights <- optimize\_weights(arima\_forecast, prophet\_forecast, testing\_data)

arima\_prophet\_forecast <- hybrid\_forecast\_function(weights, arima\_forecast, prophet\_forecast)

weights <- optimize\_weights(arima\_forecast, xgb\_forecast, testing\_data)

arima\_xgb\_forecast <- hybrid\_forecast\_function(weights, arima\_forecast, xgb\_forecast)

calculate\_rmse <- function(actual, forecast) {

sqrt(mean((actual - forecast)^2))}

models <- list(

ARIMA = arima\_forecast,

ETS = ets\_forecast,

MLP = mlp\_forecast,

Prophet = prophet\_forecast,

XGBoost = xgb\_forecast,

ARIMA\_ETS = arima\_ets\_forecast,

ARIMA\_MLP = arima\_mlp\_forecast,

ARIMA\_Prophet = arima\_prophet\_forecast,

ARIMA\_XGBoost = arima\_xgb\_forecast)

for (model\_name in names(models)) {

rmse <- calculate\_rmse(testing\_data, models[[model\_name]])

results <- rbind(results, data.frame(Horizon = h, Model = model\_name, RMSE = rmse))}}

print("Sensitivity Analysis Results:")

print(results)

ggplot(results, aes(x = Horizon, y = RMSE, color = Model, group = Model)) +geom\_line() +geom\_point() +labs(title = "TCD", y = "RMSE", x = "Forecast Horizon (Years)") +theme\_minimal()

**# Sensitivity Analysis for CIV UN-5 Female Mortality using Time Horizon: 3, 5, and 7 years**

set.seed(123)

CIV = c(155.87, 151.78, 148.33, 145.5, 143.07, 141.61, 140.46, 139.78, 139.56, 139.93, 140.24, 140.86, 141.3, 141.38, 140.91, 140.06, 138.8, 136.99, 134.62, 131.83, 128.62, 125.27, 121.66, 117.97, 114.13, 110.09, 106.37, 102.54, 98.637, 94.707, 91.419, 88.443, 85.677, 82.741, 79.602, 76.84, 74.63, 72.457, 70.049, 67.555, 65.409, 63.433, 61.482)

civ\_ts <- ts(CIV, start = 1980, frequency = 1)

horizons <- c(3, 5, 7)

results <- data.frame(Horizon = integer(), Model = character(), RMSE = numeric())

for (h in horizons) {

training\_data <- window(civ\_ts, start = 1980, end = 2017)

testing\_data <- window(civ\_ts, start = 2018, end = 2017 + h)

arima\_model <- auto.arima(training\_data)

arima\_forecast <- forecast(arima\_model, h = length(testing\_data))$mean

ets\_model <- ets(training\_data)

ets\_forecast <- forecast(ets\_model, h = length(testing\_data))$mean

mlp\_model <- mlp(training\_data)

mlp\_forecast <- forecast(mlp\_model, h = length(testing\_data))$mean

train\_df <- data.frame(

ds = seq.Date(from = as.Date("1980-01-01"), by = "year", length.out = length(training\_data)),

y = as.numeric(training\_data))

prophet\_model <- prophet(train\_df, yearly.seasonality = FALSE)

future <- data.frame(ds = seq.Date(from = max(train\_df$ds) + 1, by = "year", length.out = length(testing\_data)))

prophet\_forecast <- predict(prophet\_model, future)$yhat

lagged\_data <- data.frame(Target = training\_data[-1], Lag1 = training\_data[-length(training\_data)])

xgb\_model <- xgboost(data = as.matrix(lagged\_data$Lag1), label = lagged\_data$Target, nrounds = 100)

xgb\_forecast <- numeric(length(testing\_data))

last\_value <- training\_data[length(training\_data)]

for (i in 1:length(testing\_data)) {

xgb\_forecast[i] <- predict(xgb\_model, newdata = as.matrix(last\_value))

last\_value <- xgb\_forecast[i]}

hybrid\_forecast\_function <- function(weights, f1, f2) {

weights[1] \* as.numeric(f1) + weights[2] \* as.numeric(f2)}

optimize\_weights <- function(f1, f2, actual) {ga\_model <- ga(type = "real-valued", fitness = function(w) -sqrt(mean((actual - hybrid\_forecast\_function(w, f1, f2))^2)), lower = c(0, 0), upper = c(1, 1), popSize = 50, maxiter = 100)

return(ga\_model@solution)}

weights <- optimize\_weights(arima\_forecast, ets\_forecast, testing\_data)

arima\_ets\_forecast <- hybrid\_forecast\_function(weights, arima\_forecast, ets\_forecast)

weights <- optimize\_weights(arima\_forecast, mlp\_forecast, testing\_data)

arima\_mlp\_forecast <- hybrid\_forecast\_function(weights, arima\_forecast, mlp\_forecast)

weights <- optimize\_weights(arima\_forecast, prophet\_forecast, testing\_data)

arima\_prophet\_forecast <- hybrid\_forecast\_function(weights, arima\_forecast, prophet\_forecast)

weights <- optimize\_weights(arima\_forecast, xgb\_forecast, testing\_data)

arima\_xgb\_forecast <- hybrid\_forecast\_function(weights, arima\_forecast, xgb\_forecast)

calculate\_rmse <- function(actual, forecast) {

sqrt(mean((actual - forecast)^2))}

models <- list(

ARIMA = arima\_forecast,

ETS = ets\_forecast,

MLP = mlp\_forecast,

Prophet = prophet\_forecast,

XGBoost = xgb\_forecast,

ARIMA\_ETS = arima\_ets\_forecast,

ARIMA\_MLP = arima\_mlp\_forecast,

ARIMA\_Prophet = arima\_prophet\_forecast,

ARIMA\_XGBoost = arima\_xgb\_forecast)

for (model\_name in names(models)) {

rmse <- calculate\_rmse(testing\_data, models[[model\_name]])

results <- rbind(results, data.frame(Horizon = h, Model = model\_name, RMSE = rmse))}}

print("Sensitivity Analysis Results:")

print(results)

ggplot(results, aes(x = Horizon, y = RMSE, color = Model, group = Model)) +geom\_line() +geom\_point() +labs(title = "CIV", y = "RMSE", x = "Forecast Horizon (Years)") +theme\_minimal()

**# Sensitivity Analysis for GNQ UN-5 Female Mortality using Time Horizon: 3, 5, and 7 years**

set.seed(123)

GNQ = c(191.36, 191.36, 191.36, 187.19, 183.29, 179.7, 176.62, 174.11, 172.12, 170.43, 168.83, 167.22, 165.83, 164.47, 163.09, 161.65, 159.78, 157.13, 154.19, 150.57, 146.7, 142.58, 138.38, 134.08, 129.77, 125.17, 120.68, 116.17, 111.74, 107.62, 103.71, 100.1, 96.602, 93.199, 89.845, 86.55, 83.577, 80.926, 77.911, 74.951, 72.399, 69.923, 67.394)

gnq\_ts <- ts(GNQ, start = 1980, frequency = 1)

horizons <- c(3, 5, 7)

results <- data.frame(Horizon = integer(), Model = character(), RMSE = numeric())

for (h in horizons) {

training\_data <- window(gnq\_ts, start = 1980, end = 2017)

testing\_data <- window(gnq\_ts, start = 2018, end = 2017 + h)

arima\_model <- auto.arima(training\_data)

arima\_forecast <- forecast(arima\_model, h = length(testing\_data))$mean

ets\_model <- ets(training\_data)

ets\_forecast <- forecast(ets\_model, h = length(testing\_data))$mean

mlp\_model <- mlp(training\_data)

mlp\_forecast <- forecast(mlp\_model, h = length(testing\_data))$mean

train\_df <- data.frame(

ds = seq.Date(from = as.Date("1980-01-01"), by = "year", length.out = length(training\_data)),

y = as.numeric(training\_data))

prophet\_model <- prophet(train\_df, yearly.seasonality = FALSE)

future <- data.frame(ds = seq.Date(from = max(train\_df$ds) + 1, by = "year", length.out = length(testing\_data)))

prophet\_forecast <- predict(prophet\_model, future)$yhat

lagged\_data <- data.frame(Target = training\_data[-1], Lag1 = training\_data[-length(training\_data)])

xgb\_model <- xgboost(data = as.matrix(lagged\_data$Lag1), label = lagged\_data$Target, nrounds = 100)

xgb\_forecast <- numeric(length(testing\_data))

last\_value <- training\_data[length(training\_data)]

for (i in 1:length(testing\_data)) {

xgb\_forecast[i] <- predict(xgb\_model, newdata = as.matrix(last\_value))

last\_value <- xgb\_forecast[i]}

hybrid\_forecast\_function <- function(weights, f1, f2) {

weights[1] \* as.numeric(f1) + weights[2] \* as.numeric(f2)}

optimize\_weights <- function(f1, f2, actual) {ga\_model <- ga(type = "real-valued", fitness = function(w) -sqrt(mean((actual - hybrid\_forecast\_function(w, f1, f2))^2)), lower = c(0, 0), upper = c(1, 1), popSize = 50, maxiter = 100)

return(ga\_model@solution)}

weights <- optimize\_weights(arima\_forecast, ets\_forecast, testing\_data)

arima\_ets\_forecast <- hybrid\_forecast\_function(weights, arima\_forecast, ets\_forecast)

weights <- optimize\_weights(arima\_forecast, mlp\_forecast, testing\_data)

arima\_mlp\_forecast <- hybrid\_forecast\_function(weights, arima\_forecast, mlp\_forecast)

weights <- optimize\_weights(arima\_forecast, prophet\_forecast, testing\_data)

arima\_prophet\_forecast <- hybrid\_forecast\_function(weights, arima\_forecast, prophet\_forecast)

weights <- optimize\_weights(arima\_forecast, xgb\_forecast, testing\_data)

arima\_xgb\_forecast <- hybrid\_forecast\_function(weights, arima\_forecast, xgb\_forecast)

calculate\_rmse <- function(actual, forecast) {

sqrt(mean((actual - forecast)^2))}

models <- list(

ARIMA = arima\_forecast,

ETS = ets\_forecast,

MLP = mlp\_forecast,

Prophet = prophet\_forecast,

XGBoost = xgb\_forecast,

ARIMA\_ETS = arima\_ets\_forecast,

ARIMA\_MLP = arima\_mlp\_forecast,

ARIMA\_Prophet = arima\_prophet\_forecast,

ARIMA\_XGBoost = arima\_xgb\_forecast)

for (model\_name in names(models)) {

rmse <- calculate\_rmse(testing\_data, models[[model\_name]])

results <- rbind(results, data.frame(Horizon = h, Model = model\_name, RMSE = rmse))}}

print("Sensitivity Analysis Results:")

print(results)

ggplot(results, aes(x = Horizon, y = RMSE, color = Model, group = Model)) +geom\_line() +geom\_point() +labs(title = "GNQ", y = "RMSE", x = "Forecast Horizon (Years)") +theme\_minimal()

**# Sensitivity Analysis for GMB UN-5 Female Mortality using Time Horizon: 3, 5, and 7 years**

set.seed(123)

GMB = c(218.69, 212.42, 206.11, 199.41, 192.9, 186.34, 179.98, 173.64, 167.69, 161.94, 156.34, 150.94, 145.7, 140.5, 135.35, 130.14, 124.96, 119.99, 115.04, 110.19, 105.7, 101.22, 96.831, 92.538, 88.442, 84.432, 80.626, 77.009, 73.554, 70.314, 67.256, 64.272, 61.5, 58.867, 56.351, 54.003, 51.65, 49.518, 47.552, 45.723, 43.944, 42.364, 40.798)

gmb\_ts <- ts(GMB, start = 1980, frequency = 1)

horizons <- c(3, 5, 7)

results <- data.frame(Horizon = integer(), Model = character(), RMSE = numeric())

for (h in horizons) {

training\_data <- window(gmb\_ts, start = 1980, end = 2017)

testing\_data <- window(gmb\_ts, start = 2018, end = 2017 + h)

arima\_model <- auto.arima(training\_data)

arima\_forecast <- forecast(arima\_model, h = length(testing\_data))$mean

ets\_model <- ets(training\_data)

ets\_forecast <- forecast(ets\_model, h = length(testing\_data))$mean

mlp\_model <- mlp(training\_data)

mlp\_forecast <- forecast(mlp\_model, h = length(testing\_data))$mean

train\_df <- data.frame(

ds = seq.Date(from = as.Date("1980-01-01"), by = "year", length.out = length(training\_data)),

y = as.numeric(training\_data))

prophet\_model <- prophet(train\_df, yearly.seasonality = FALSE)

future <- data.frame(ds = seq.Date(from = max(train\_df$ds) + 1, by = "year", length.out = length(testing\_data)))

prophet\_forecast <- predict(prophet\_model, future)$yhat

lagged\_data <- data.frame(Target = training\_data[-1], Lag1 = training\_data[-length(training\_data)])

xgb\_model <- xgboost(data = as.matrix(lagged\_data$Lag1), label = lagged\_data$Target, nrounds = 100)

xgb\_forecast <- numeric(length(testing\_data))

last\_value <- training\_data[length(training\_data)]

for (i in 1:length(testing\_data)) {

xgb\_forecast[i] <- predict(xgb\_model, newdata = as.matrix(last\_value))

last\_value <- xgb\_forecast[i]}

hybrid\_forecast\_function <- function(weights, f1, f2) {

weights[1] \* as.numeric(f1) + weights[2] \* as.numeric(f2)}

optimize\_weights <- function(f1, f2, actual) {ga\_model <- ga(type = "real-valued", fitness = function(w) -sqrt(mean((actual - hybrid\_forecast\_function(w, f1, f2))^2)), lower = c(0, 0), upper = c(1, 1), popSize = 50, maxiter = 100)

return(ga\_model@solution)}

weights <- optimize\_weights(arima\_forecast, ets\_forecast, testing\_data)

arima\_ets\_forecast <- hybrid\_forecast\_function(weights, arima\_forecast, ets\_forecast)

weights <- optimize\_weights(arima\_forecast, mlp\_forecast, testing\_data)

arima\_mlp\_forecast <- hybrid\_forecast\_function(weights, arima\_forecast, mlp\_forecast)

weights <- optimize\_weights(arima\_forecast, prophet\_forecast, testing\_data)

arima\_prophet\_forecast <- hybrid\_forecast\_function(weights, arima\_forecast, prophet\_forecast)

weights <- optimize\_weights(arima\_forecast, xgb\_forecast, testing\_data)

arima\_xgb\_forecast <- hybrid\_forecast\_function(weights, arima\_forecast, xgb\_forecast)

calculate\_rmse <- function(actual, forecast) {

sqrt(mean((actual - forecast)^2))}

models <- list(

ARIMA = arima\_forecast,

ETS = ets\_forecast,

MLP = mlp\_forecast,

Prophet = prophet\_forecast,

XGBoost = xgb\_forecast,

ARIMA\_ETS = arima\_ets\_forecast,

ARIMA\_MLP = arima\_mlp\_forecast,

ARIMA\_Prophet = arima\_prophet\_forecast,

ARIMA\_XGBoost = arima\_xgb\_forecast)

for (model\_name in names(models)) {

rmse <- calculate\_rmse(testing\_data, models[[model\_name]])

results <- rbind(results, data.frame(Horizon = h, Model = model\_name, RMSE = rmse))}}

print("Sensitivity Analysis Results:")

print(results)

ggplot(results, aes(x = Horizon, y = RMSE, color = Model, group = Model)) +geom\_line() +geom\_point() +labs(title = "GMB", y = "RMSE", x = "Forecast Horizon (Years)") +theme\_minimal()

**# Sensitivity Analysis for GHA UN-5 Female Mortality using Time Horizon: 3, 5, and 7 years**

set.seed(123)

GHA = c(158.81, 157.23, 155.8, 153.64, 150.58, 146.43, 141.49, 135.85, 130.02, 124.41, 119.45, 115.21, 111.94, 109.53, 107.86, 106.25, 104.47, 102.19, 99.322, 95.799, 92.031, 88.179, 84.433, 81.158, 78.287, 75.786, 73.547, 71.238, 68.635, 65.89, 63.004, 60.024, 57.1, 54.417, 51.989, 49.672, 47.576, 45.608, 43.844, 42.083, 40.495, 39.113, 37.769)

gha\_ts <- ts(GHA, start = 1980, frequency = 1)

horizons <- c(3, 5, 7)

results <- data.frame(Horizon = integer(), Model = character(), RMSE = numeric())

for (h in horizons) {

training\_data <- window(gha\_ts, start = 1980, end = 2017)

testing\_data <- window(gha\_ts, start = 2018, end = 2017 + h)

arima\_model <- auto.arima(training\_data)

arima\_forecast <- forecast(arima\_model, h = length(testing\_data))$mean

ets\_model <- ets(training\_data)

ets\_forecast <- forecast(ets\_model, h = length(testing\_data))$mean

mlp\_model <- mlp(training\_data)

mlp\_forecast <- forecast(mlp\_model, h = length(testing\_data))$mean

train\_df <- data.frame(

ds = seq.Date(from = as.Date("1980-01-01"), by = "year", length.out = length(training\_data)),

y = as.numeric(training\_data))

prophet\_model <- prophet(train\_df, yearly.seasonality = FALSE)

future <- data.frame(ds = seq.Date(from = max(train\_df$ds) + 1, by = "year", length.out = length(testing\_data)))

prophet\_forecast <- predict(prophet\_model, future)$yhat

lagged\_data <- data.frame(Target = training\_data[-1], Lag1 = training\_data[-length(training\_data)])

xgb\_model <- xgboost(data = as.matrix(lagged\_data$Lag1), label = lagged\_data$Target, nrounds = 100)

xgb\_forecast <- numeric(length(testing\_data))

last\_value <- training\_data[length(training\_data)]

for (i in 1:length(testing\_data)) {

xgb\_forecast[i] <- predict(xgb\_model, newdata = as.matrix(last\_value))

last\_value <- xgb\_forecast[i]}

hybrid\_forecast\_function <- function(weights, f1, f2) {

weights[1] \* as.numeric(f1) + weights[2] \* as.numeric(f2)}

optimize\_weights <- function(f1, f2, actual) {ga\_model <- ga(type = "real-valued", fitness = function(w) -sqrt(mean((actual - hybrid\_forecast\_function(w, f1, f2))^2)), lower = c(0, 0), upper = c(1, 1), popSize = 50, maxiter = 100)

return(ga\_model@solution)}

weights <- optimize\_weights(arima\_forecast, ets\_forecast, testing\_data)

arima\_ets\_forecast <- hybrid\_forecast\_function(weights, arima\_forecast, ets\_forecast)

weights <- optimize\_weights(arima\_forecast, mlp\_forecast, testing\_data)

arima\_mlp\_forecast <- hybrid\_forecast\_function(weights, arima\_forecast, mlp\_forecast)

weights <- optimize\_weights(arima\_forecast, prophet\_forecast, testing\_data)

arima\_prophet\_forecast <- hybrid\_forecast\_function(weights, arima\_forecast, prophet\_forecast)

weights <- optimize\_weights(arima\_forecast, xgb\_forecast, testing\_data)

arima\_xgb\_forecast <- hybrid\_forecast\_function(weights, arima\_forecast, xgb\_forecast)

calculate\_rmse <- function(actual, forecast) {

sqrt(mean((actual - forecast)^2))}

models <- list(

ARIMA = arima\_forecast,

ETS = ets\_forecast,

MLP = mlp\_forecast,

Prophet = prophet\_forecast,

XGBoost = xgb\_forecast,

ARIMA\_ETS = arima\_ets\_forecast,

ARIMA\_MLP = arima\_mlp\_forecast,

ARIMA\_Prophet = arima\_prophet\_forecast,

ARIMA\_XGBoost = arima\_xgb\_forecast)

for (model\_name in names(models)) {

rmse <- calculate\_rmse(testing\_data, models[[model\_name]])

results <- rbind(results, data.frame(Horizon = h, Model = model\_name, RMSE = rmse))}}

print("Sensitivity Analysis Results:")

print(results)

ggplot(results, aes(x = Horizon, y = RMSE, color = Model, group = Model)) +geom\_line() +geom\_point() +labs(title = "GHA", y = "RMSE", x = "Forecast Horizon (Years)") +theme\_minimal()

**# Sensitivity Analysis for NER UN-5 Female Mortality using Time Horizon: 3, 5, and 7 years**

set.seed(123)

NER = c(321.21, 320.77, 322.92, 326.94, 331.58, 335.68, 338.76, 339.5, 338.37, 335.45, 330.47, 322.71, 313.64, 302.56, 290.4, 277.6, 264.84, 253.29, 242.87, 233.92, 225.05, 215.12, 203.83, 192.02, 180.27, 169.16, 159.39, 150.85, 143.64, 137.94, 133.42, 130, 127.57, 125.84, 124.43, 123.19, 122.05, 120.94, 119.92, 118.8, 117.53, 115.94, 113.83)

ner\_ts <- ts(NER, start = 1980, frequency = 1)

horizons <- c(3, 5, 7)

results <- data.frame(Horizon = integer(), Model = character(), RMSE = numeric())

for (h in horizons) {

training\_data <- window(ner\_ts, start = 1980, end = 2017)

testing\_data <- window(ner\_ts, start = 2018, end = 2017 + h)

arima\_model <- auto.arima(training\_data)

arima\_forecast <- forecast(arima\_model, h = length(testing\_data))$mean

ets\_model <- ets(training\_data)

ets\_forecast <- forecast(ets\_model, h = length(testing\_data))$mean

mlp\_model <- mlp(training\_data)

mlp\_forecast <- forecast(mlp\_model, h = length(testing\_data))$mean

train\_df <- data.frame(

ds = seq.Date(from = as.Date("1980-01-01"), by = "year", length.out = length(training\_data)),

y = as.numeric(training\_data))

prophet\_model <- prophet(train\_df, yearly.seasonality = FALSE)

future <- data.frame(ds = seq.Date(from = max(train\_df$ds) + 1, by = "year", length.out = length(testing\_data)))

prophet\_forecast <- predict(prophet\_model, future)$yhat

lagged\_data <- data.frame(Target = training\_data[-1], Lag1 = training\_data[-length(training\_data)])

xgb\_model <- xgboost(data = as.matrix(lagged\_data$Lag1), label = lagged\_data$Target, nrounds = 100)

xgb\_forecast <- numeric(length(testing\_data))

last\_value <- training\_data[length(training\_data)]

for (i in 1:length(testing\_data)) {

xgb\_forecast[i] <- predict(xgb\_model, newdata = as.matrix(last\_value))

last\_value <- xgb\_forecast[i]}

hybrid\_forecast\_function <- function(weights, f1, f2) {

weights[1] \* as.numeric(f1) + weights[2] \* as.numeric(f2)}

optimize\_weights <- function(f1, f2, actual) {ga\_model <- ga(type = "real-valued", fitness = function(w) -sqrt(mean((actual - hybrid\_forecast\_function(w, f1, f2))^2)), lower = c(0, 0), upper = c(1, 1), popSize = 50, maxiter = 100)

return(ga\_model@solution)}

weights <- optimize\_weights(arima\_forecast, ets\_forecast, testing\_data)

arima\_ets\_forecast <- hybrid\_forecast\_function(weights, arima\_forecast, ets\_forecast)

weights <- optimize\_weights(arima\_forecast, mlp\_forecast, testing\_data)

arima\_mlp\_forecast <- hybrid\_forecast\_function(weights, arima\_forecast, mlp\_forecast)

weights <- optimize\_weights(arima\_forecast, prophet\_forecast, testing\_data)

arima\_prophet\_forecast <- hybrid\_forecast\_function(weights, arima\_forecast, prophet\_forecast)

weights <- optimize\_weights(arima\_forecast, xgb\_forecast, testing\_data)

arima\_xgb\_forecast <- hybrid\_forecast\_function(weights, arima\_forecast, xgb\_forecast)

calculate\_rmse <- function(actual, forecast) {

sqrt(mean((actual - forecast)^2))}

models <- list(

ARIMA = arima\_forecast,

ETS = ets\_forecast,

MLP = mlp\_forecast,

Prophet = prophet\_forecast,

XGBoost = xgb\_forecast,

ARIMA\_ETS = arima\_ets\_forecast,

ARIMA\_MLP = arima\_mlp\_forecast,

ARIMA\_Prophet = arima\_prophet\_forecast,

ARIMA\_XGBoost = arima\_xgb\_forecast)

for (model\_name in names(models)) {

rmse <- calculate\_rmse(testing\_data, models[[model\_name]])

results <- rbind(results, data.frame(Horizon = h, Model = model\_name, RMSE = rmse))}}

print("Sensitivity Analysis Results:")

print(results)

ggplot(results, aes(x = Horizon, y = RMSE, color = Model, group = Model)) +geom\_line() +geom\_point() +labs(title = "NER", y = "RMSE", x = "Forecast Horizon (Years)") +theme\_minimal()

**# Sensitivity Analysis for NGA UN-5 Female Mortality using Time Horizon: 3, 5, and 7 years**

set.seed(123)

NGA = c(202.31, 199.33, 197.51, 196.55, 196.41, 196.88, 197.45, 197.92, 198.53, 198.59, 198.28, 197.85, 197.5, 197, 195.8, 194.02, 191.49, 188.05, 183.46, 178.64, 173.65, 168.12, 162.43, 156.65, 151.03, 145.78, 140.8, 136.74, 133.26, 130.13, 127.6, 125.38, 123.53, 121.83, 120.45, 119.25, 117.81, 115.79, 113.29, 110.33, 107.47, 104.33, 100.91)

nga\_ts <- ts(NGA, start = 1980, frequency = 1)

horizons <- c(3, 5, 7)

results <- data.frame(Horizon = integer(), Model = character(), RMSE = numeric())

for (h in horizons) {

training\_data <- window(nga\_ts, start = 1980, end = 2017)

testing\_data <- window(nga\_ts, start = 2018, end = 2017 + h)

arima\_model <- auto.arima(training\_data)

arima\_forecast <- forecast(arima\_model, h = length(testing\_data))$mean

ets\_model <- ets(training\_data)

ets\_forecast <- forecast(ets\_model, h = length(testing\_data))$mean

mlp\_model <- mlp(training\_data)

mlp\_forecast <- forecast(mlp\_model, h = length(testing\_data))$mean

train\_df <- data.frame(

ds = seq.Date(from = as.Date("1980-01-01"), by = "year", length.out = length(training\_data)),

y = as.numeric(training\_data))

prophet\_model <- prophet(train\_df, yearly.seasonality = FALSE)

future <- data.frame(ds = seq.Date(from = max(train\_df$ds) + 1, by = "year", length.out = length(testing\_data)))

prophet\_forecast <- predict(prophet\_model, future)$yhat

lagged\_data <- data.frame(Target = training\_data[-1], Lag1 = training\_data[-length(training\_data)])

xgb\_model <- xgboost(data = as.matrix(lagged\_data$Lag1), label = lagged\_data$Target, nrounds = 100)

xgb\_forecast <- numeric(length(testing\_data))

last\_value <- training\_data[length(training\_data)]

for (i in 1:length(testing\_data)) {

xgb\_forecast[i] <- predict(xgb\_model, newdata = as.matrix(last\_value))

last\_value <- xgb\_forecast[i]}

hybrid\_forecast\_function <- function(weights, f1, f2) {

weights[1] \* as.numeric(f1) + weights[2] \* as.numeric(f2)}

optimize\_weights <- function(f1, f2, actual) {ga\_model <- ga(type = "real-valued", fitness = function(w) -sqrt(mean((actual - hybrid\_forecast\_function(w, f1, f2))^2)), lower = c(0, 0), upper = c(1, 1), popSize = 50, maxiter = 100)

return(ga\_model@solution)}

weights <- optimize\_weights(arima\_forecast, ets\_forecast, testing\_data)

arima\_ets\_forecast <- hybrid\_forecast\_function(weights, arima\_forecast, ets\_forecast)

weights <- optimize\_weights(arima\_forecast, mlp\_forecast, testing\_data)

arima\_mlp\_forecast <- hybrid\_forecast\_function(weights, arima\_forecast, mlp\_forecast)

weights <- optimize\_weights(arima\_forecast, prophet\_forecast, testing\_data)

arima\_prophet\_forecast <- hybrid\_forecast\_function(weights, arima\_forecast, prophet\_forecast)

weights <- optimize\_weights(arima\_forecast, xgb\_forecast, testing\_data)

arima\_xgb\_forecast <- hybrid\_forecast\_function(weights, arima\_forecast, xgb\_forecast)

calculate\_rmse <- function(actual, forecast) {

sqrt(mean((actual - forecast)^2))}

models <- list(

ARIMA = arima\_forecast,

ETS = ets\_forecast,

MLP = mlp\_forecast,

Prophet = prophet\_forecast,

XGBoost = xgb\_forecast,

ARIMA\_ETS = arima\_ets\_forecast,

ARIMA\_MLP = arima\_mlp\_forecast,

ARIMA\_Prophet = arima\_prophet\_forecast,

ARIMA\_XGBoost = arima\_xgb\_forecast)

for (model\_name in names(models)) {

rmse <- calculate\_rmse(testing\_data, models[[model\_name]])

results <- rbind(results, data.frame(Horizon = h, Model = model\_name, RMSE = rmse))}}

print("Sensitivity Analysis Results:")

print(results)

ggplot(results, aes(x = Horizon, y = RMSE, color = Model, group = Model)) +geom\_line() +geom\_point() +labs(title = "NGA", y = "RMSE", x = "Forecast Horizon (Years)") +theme\_minimal()

**# Sensitivity Analysis for SEN UN-5 Female Mortality using Time Horizon: 3, 5, and 7 years**

set.seed(123)

SEN = c(197.36, 190.43, 184.17, 177.74, 170.74, 163.33, 155.23, 147.36, 140.37, 134.91, 131.26, 129.31, 128.79, 129.38, 130.47, 131.39, 132.03, 131.85, 130.41, 127.49, 122.74, 116.22, 108.51, 100.58, 92.856, 85.743, 79.37, 73.77, 68.913, 64.721, 61.043, 57.681, 54.575, 51.641, 48.809, 46.187, 43.699, 41.385, 39.284, 37.453, 35.79, 34.366, 32.959)

sen\_ts <- ts(SEN, start = 1980, frequency = 1)

horizons <- c(3, 5, 7)

results <- data.frame(Horizon = integer(), Model = character(), RMSE = numeric())

for (h in horizons) {

training\_data <- window(sen\_ts, start = 1980, end = 2017)

testing\_data <- window(sen\_ts, start = 2018, end = 2017 + h)

arima\_model <- auto.arima(training\_data)

arima\_forecast <- forecast(arima\_model, h = length(testing\_data))$mean

ets\_model <- ets(training\_data)

ets\_forecast <- forecast(ets\_model, h = length(testing\_data))$mean

mlp\_model <- mlp(training\_data)

mlp\_forecast <- forecast(mlp\_model, h = length(testing\_data))$mean

train\_df <- data.frame(

ds = seq.Date(from = as.Date("1980-01-01"), by = "year", length.out = length(training\_data)),

y = as.numeric(training\_data))

prophet\_model <- prophet(train\_df, yearly.seasonality = FALSE)

future <- data.frame(ds = seq.Date(from = max(train\_df$ds) + 1, by = "year", length.out = length(testing\_data)))

prophet\_forecast <- predict(prophet\_model, future)$yhat

lagged\_data <- data.frame(Target = training\_data[-1], Lag1 = training\_data[-length(training\_data)])

xgb\_model <- xgboost(data = as.matrix(lagged\_data$Lag1), label = lagged\_data$Target, nrounds = 100)

xgb\_forecast <- numeric(length(testing\_data))

last\_value <- training\_data[length(training\_data)]

for (i in 1:length(testing\_data)) {

xgb\_forecast[i] <- predict(xgb\_model, newdata = as.matrix(last\_value))

last\_value <- xgb\_forecast[i]}

hybrid\_forecast\_function <- function(weights, f1, f2) {

weights[1] \* as.numeric(f1) + weights[2] \* as.numeric(f2)}

optimize\_weights <- function(f1, f2, actual) {ga\_model <- ga(type = "real-valued", fitness = function(w) -sqrt(mean((actual - hybrid\_forecast\_function(w, f1, f2))^2)), lower = c(0, 0), upper = c(1, 1), popSize = 50, maxiter = 100)

return(ga\_model@solution)}

weights <- optimize\_weights(arima\_forecast, ets\_forecast, testing\_data)

arima\_ets\_forecast <- hybrid\_forecast\_function(weights, arima\_forecast, ets\_forecast)

weights <- optimize\_weights(arima\_forecast, mlp\_forecast, testing\_data)

arima\_mlp\_forecast <- hybrid\_forecast\_function(weights, arima\_forecast, mlp\_forecast)

weights <- optimize\_weights(arima\_forecast, prophet\_forecast, testing\_data)

arima\_prophet\_forecast <- hybrid\_forecast\_function(weights, arima\_forecast, prophet\_forecast)

weights <- optimize\_weights(arima\_forecast, xgb\_forecast, testing\_data)

arima\_xgb\_forecast <- hybrid\_forecast\_function(weights, arima\_forecast, xgb\_forecast)

calculate\_rmse <- function(actual, forecast) {

sqrt(mean((actual - forecast)^2))}

models <- list(

ARIMA = arima\_forecast,

ETS = ets\_forecast,

MLP = mlp\_forecast,

Prophet = prophet\_forecast,

XGBoost = xgb\_forecast,

ARIMA\_ETS = arima\_ets\_forecast,

ARIMA\_MLP = arima\_mlp\_forecast,

ARIMA\_Prophet = arima\_prophet\_forecast,

ARIMA\_XGBoost = arima\_xgb\_forecast)

for (model\_name in names(models)) {

rmse <- calculate\_rmse(testing\_data, models[[model\_name]])

results <- rbind(results, data.frame(Horizon = h, Model = model\_name, RMSE = rmse))}}

print("Sensitivity Analysis Results:")

print(results)

ggplot(results, aes(x = Horizon, y = RMSE, color = Model, group = Model)) +geom\_line() +geom\_point() +labs(title = "SEN", y = "RMSE", x = "Forecast Horizon (Years)") +theme\_minimal()

**# Sensitivity Analysis for SLE UN-5 Female Mortality using Time Horizon: 3, 5, and 7 years**

set.seed(123)

SLE = c(275.31, 271.84, 268.41, 265.3, 262.37, 259.63, 256.86, 254.35, 252.02, 249.68, 247.61, 245.36, 243.02, 240.73, 238.34, 235.5, 232.3, 228.46, 224.21, 219.93, 214.74, 209.84, 204.54, 199.09, 193.47, 187.35, 180.98, 174.12, 167.04, 160.13, 152.99, 146.07, 139.71, 133.64, 131.71, 132.69, 117.94, 113.38, 109, 104.89, 100.98, 97.39, 93.834)

sle\_ts <- ts(SLE, start = 1980, frequency = 1)

horizons <- c(3, 5, 7)

results <- data.frame(Horizon = integer(), Model = character(), RMSE = numeric())

for (h in horizons) {

training\_data <- window(sle\_ts, start = 1980, end = 2017)

testing\_data <- window(sle\_ts, start = 2018, end = 2017 + h)

arima\_model <- auto.arima(training\_data)

arima\_forecast <- forecast(arima\_model, h = length(testing\_data))$mean

ets\_model <- ets(training\_data)

ets\_forecast <- forecast(ets\_model, h = length(testing\_data))$mean

mlp\_model <- mlp(training\_data)

mlp\_forecast <- forecast(mlp\_model, h = length(testing\_data))$mean

train\_df <- data.frame(

ds = seq.Date(from = as.Date("1980-01-01"), by = "year", length.out = length(training\_data)),

y = as.numeric(training\_data))

prophet\_model <- prophet(train\_df, yearly.seasonality = FALSE)

future <- data.frame(ds = seq.Date(from = max(train\_df$ds) + 1, by = "year", length.out = length(testing\_data)))

prophet\_forecast <- predict(prophet\_model, future)$yhat

lagged\_data <- data.frame(Target = training\_data[-1], Lag1 = training\_data[-length(training\_data)])

xgb\_model <- xgboost(data = as.matrix(lagged\_data$Lag1), label = lagged\_data$Target, nrounds = 100)

xgb\_forecast <- numeric(length(testing\_data))

last\_value <- training\_data[length(training\_data)]

for (i in 1:length(testing\_data)) {

xgb\_forecast[i] <- predict(xgb\_model, newdata = as.matrix(last\_value))

last\_value <- xgb\_forecast[i]}

hybrid\_forecast\_function <- function(weights, f1, f2) {

weights[1] \* as.numeric(f1) + weights[2] \* as.numeric(f2)}

optimize\_weights <- function(f1, f2, actual) {ga\_model <- ga(type = "real-valued", fitness = function(w) -sqrt(mean((actual - hybrid\_forecast\_function(w, f1, f2))^2)), lower = c(0, 0), upper = c(1, 1), popSize = 50, maxiter = 100)

return(ga\_model@solution)}

weights <- optimize\_weights(arima\_forecast, ets\_forecast, testing\_data)

arima\_ets\_forecast <- hybrid\_forecast\_function(weights, arima\_forecast, ets\_forecast)

weights <- optimize\_weights(arima\_forecast, mlp\_forecast, testing\_data)

arima\_mlp\_forecast <- hybrid\_forecast\_function(weights, arima\_forecast, mlp\_forecast)

weights <- optimize\_weights(arima\_forecast, prophet\_forecast, testing\_data)

arima\_prophet\_forecast <- hybrid\_forecast\_function(weights, arima\_forecast, prophet\_forecast)

weights <- optimize\_weights(arima\_forecast, xgb\_forecast, testing\_data)

arima\_xgb\_forecast <- hybrid\_forecast\_function(weights, arima\_forecast, xgb\_forecast)

calculate\_rmse <- function(actual, forecast) {

sqrt(mean((actual - forecast)^2))}

models <- list(

ARIMA = arima\_forecast,

ETS = ets\_forecast,

MLP = mlp\_forecast,

Prophet = prophet\_forecast,

XGBoost = xgb\_forecast,

ARIMA\_ETS = arima\_ets\_forecast,

ARIMA\_MLP = arima\_mlp\_forecast,

ARIMA\_Prophet = arima\_prophet\_forecast,

ARIMA\_XGBoost = arima\_xgb\_forecast)

for (model\_name in names(models)) {

rmse <- calculate\_rmse(testing\_data, models[[model\_name]])

results <- rbind(results, data.frame(Horizon = h, Model = model\_name, RMSE = rmse))}}

print("Sensitivity Analysis Results:")

print(results)

ggplot(results, aes(x = Horizon, y = RMSE, color = Model, group = Model)) +geom\_line() +geom\_point() +labs(title = "SLE", y = "RMSE", x = "Forecast Horizon (Years)") +theme\_minimal()

**# Sensitivity Analysis for TGO UN-5 Female Mortality using Time Horizon: 3, 5, and 7 years**

set.seed(123)

TGO = c(170.07, 166.02, 162.1, 158.4, 154.8, 151.46, 148.31, 145.36, 142.76, 140.28, 138.07, 136.15, 134.07, 131.83, 129.48, 126.83, 123.89, 120.9, 117.78, 114.41, 110.97, 107.7, 104.53, 101.47, 98.521, 95.734, 93.083, 90.405, 87.752, 85.191, 82.71, 80.095, 77.449, 74.901, 72.412, 70.036, 67.809, 65.551, 63.374, 61.207, 59.207, 57.243, 55.347)

tgo\_ts <- ts(TGO, start = 1980, frequency = 1)

horizons <- c(3, 5, 7)

results <- data.frame(Horizon = integer(), Model = character(), RMSE = numeric())

for (h in horizons) {

training\_data <- window(tgo\_ts, start = 1980, end = 2017)

testing\_data <- window(tgo\_ts, start = 2018, end = 2017 + h)

arima\_model <- auto.arima(training\_data)

arima\_forecast <- forecast(arima\_model, h = length(testing\_data))$mean

ets\_model <- ets(training\_data)

ets\_forecast <- forecast(ets\_model, h = length(testing\_data))$mean

mlp\_model <- mlp(training\_data)

mlp\_forecast <- forecast(mlp\_model, h = length(testing\_data))$mean

train\_df <- data.frame(

ds = seq.Date(from = as.Date("1980-01-01"), by = "year", length.out = length(training\_data)),

y = as.numeric(training\_data))

prophet\_model <- prophet(train\_df, yearly.seasonality = FALSE)

future <- data.frame(ds = seq.Date(from = max(train\_df$ds) + 1, by = "year", length.out = length(testing\_data)))

prophet\_forecast <- predict(prophet\_model, future)$yhat

lagged\_data <- data.frame(Target = training\_data[-1], Lag1 = training\_data[-length(training\_data)])

xgb\_model <- xgboost(data = as.matrix(lagged\_data$Lag1), label = lagged\_data$Target, nrounds = 100)

xgb\_forecast <- numeric(length(testing\_data))

last\_value <- training\_data[length(training\_data)]

for (i in 1:length(testing\_data)) {

xgb\_forecast[i] <- predict(xgb\_model, newdata = as.matrix(last\_value))

last\_value <- xgb\_forecast[i]}

hybrid\_forecast\_function <- function(weights, f1, f2) {

weights[1] \* as.numeric(f1) + weights[2] \* as.numeric(f2)}

optimize\_weights <- function(f1, f2, actual) {ga\_model <- ga(type = "real-valued", fitness = function(w) -sqrt(mean((actual - hybrid\_forecast\_function(w, f1, f2))^2)), lower = c(0, 0), upper = c(1, 1), popSize = 50, maxiter = 100)

return(ga\_model@solution)}

weights <- optimize\_weights(arima\_forecast, ets\_forecast, testing\_data)

arima\_ets\_forecast <- hybrid\_forecast\_function(weights, arima\_forecast, ets\_forecast)

weights <- optimize\_weights(arima\_forecast, mlp\_forecast, testing\_data)

arima\_mlp\_forecast <- hybrid\_forecast\_function(weights, arima\_forecast, mlp\_forecast)

weights <- optimize\_weights(arima\_forecast, prophet\_forecast, testing\_data)

arima\_prophet\_forecast <- hybrid\_forecast\_function(weights, arima\_forecast, prophet\_forecast)

weights <- optimize\_weights(arima\_forecast, xgb\_forecast, testing\_data)

arima\_xgb\_forecast <- hybrid\_forecast\_function(weights, arima\_forecast, xgb\_forecast)

calculate\_rmse <- function(actual, forecast) {

sqrt(mean((actual - forecast)^2))}

models <- list(

ARIMA = arima\_forecast,

ETS = ets\_forecast,

MLP = mlp\_forecast,

Prophet = prophet\_forecast,

XGBoost = xgb\_forecast,

ARIMA\_ETS = arima\_ets\_forecast,

ARIMA\_MLP = arima\_mlp\_forecast,

ARIMA\_Prophet = arima\_prophet\_forecast,

ARIMA\_XGBoost = arima\_xgb\_forecast)

for (model\_name in names(models)) {

rmse <- calculate\_rmse(testing\_data, models[[model\_name]])

results <- rbind(results, data.frame(Horizon = h, Model = model\_name, RMSE = rmse))}}

print("Sensitivity Analysis Results:")

print(results)

ggplot(results, aes(x = Horizon, y = RMSE, color = Model, group = Model)) +geom\_line() +geom\_point() +labs(title = "TGO", y = "RMSE", x = "Forecast Horizon (Years)") +theme\_minimal()

**Analysis for Under-5 Male Mortality Rates in West Africa**

data <- data.frame(

Year = 1980:2022,

GNQ = c(NA, NA, 210.4, 206.2, 201.85, 198.09, 195.07, 192.62, 190.47, 188.77, 187.05, 185.44, 183.9, 182.51, 181.08, 179.5, 177.39, 174.89, 171.44, 167.91, 163.84, 159.69, 155.21, 150.74, 146.19, 141.4, 136.61, 131.77, 127.04, 122.63, 118.57, 114.55, 110.62, 106.86, 103.37, 99.931, 96.594, 93.446, 90.2, 87.159, 84.438, 81.857, 79.273),

GNB = c(NA, NA, NA, NA, NA, 255.03, 250.16, 245.56, 241.3, 237.17, 232.89, 228.42, 223.74, 219.02, 214.19, 209.2, 204.12, 199.12, 194.21, 189.18, 183.82, 178.4, 172.82, 166.88, 160.78, 154.57, 148.15, 141.69, 135.22, 128.94, 122.76, 117.07, 111.81, 107.22, 102.82, 99.118, 95.589, 92.405, 89.207, 86.104, 83.296, 80.465, 77.896))

data$GNQ <- na.approx(data$GNQ, rule = 2)

data$GNB <- na.approx(data$GNB, rule = 2)

head(data)

data <- list(

BEN = c(222.61, 218.26, 214.02, 209.88, 205.56, 201.31, 196.89, 192.56, 188.19, 183.71, 179.31, 174.92, 170.57, 166.25, 162.28, 158.55, 155.01, 151.5, 148.16, 145.25, 142.29, 139.35, 136.26, 133.13, 130.12, 127.33, 124.65, 122.14, 119.87, 117.63, 115.36, 113.17, 110.87, 108.57, 106.29, 104.03, 101.72, 99.419, 96.985, 94.427, 91.807, 89.172, 86.516),

BFA = c(248.21, 241.81, 236.12, 230.91, 225.96, 221.39, 216.92, 212.89, 209.53, 207.06, 205.35, 204.24, 203.63, 202.75, 201.63, 199.56, 197.01, 194.02, 190.97, 187.65, 184.08, 180.11, 175.79, 170.6, 164.54, 158.01, 151.01, 144.1, 137.52, 131.45, 126.02, 121.26, 116.9, 112.83, 109.03, 105.42, 101.74, 98.319, 94.992, 92.171, 89.15, 86.099, 83.342),

GIN = c(289.64, 284.73, 279.85, 274.71, 269.65, 264.85, 260.11, 255.59, 251.02, 246.14, 240.82, 235.09, 228.6, 222.07, 215.38, 208.5, 201.67, 194.82, 187.96, 180.99, 174.08, 167.36, 161.06, 155.22, 150.02, 145.41, 141.47, 138.11, 135.23, 132.65, 130.27, 127.9, 125.77, 123.68, 122.32, 120.47, 117.99, 116.09, 113.88, 111.45, 108.85, 106.06, 103.14),

GNB = c(255.03, 255.03, 255.03, 255.03, 255.03, 255.03, 250.16, 245.56, 241.3, 237.17, 232.89, 228.42, 223.74, 219.02, 214.19, 209.2, 204.12, 199.12, 194.21, 189.18, 183.82, 178.4, 172.82, 166.88, 160.78, 154.57, 148.15, 141.69, 135.22, 128.94, 122.76, 117.07, 111.81, 107.22, 102.82, 99.118, 95.589, 92.405, 89.207, 86.104, 83.296, 80.465, 77.896),

LBR = c(257.9, 254.9, 252.63, 251.31, 251.38, 253.12, 256.79, 261.92, 267.8, 273.4, 277.82, 279.7, 278.8, 275.22, 268.86, 260.27, 249.95, 238.37, 225.77, 212.9, 199.78, 186.18, 172.89, 160.19, 148.35, 137.69, 128.73, 121.25, 115.23, 110.34, 106.43, 103.27, 100.56, 98.226, 98.232, 94.286, 92.204, 90.23, 88.193, 85.849, 83.494, 81.188, 78.972),

MLI = c(308.142, 299.993, 291.887, 283.921, 276.039, 268.291, 261.071, 254.887, 249.398, 244.094, 239.312, 235.079, 231.208, 227.784, 224.089, 220.369, 216.334, 211.819, 206.747, 201.047, 194.925, 188.617, 182.162, 175.681, 169.571, 163.814, 158.41, 153.433, 148.558, 143.989, 139.753, 135.552, 131.75, 128.071, 124.425, 120.894, 117.44, 114.097, 111, 107.863, 104.577, 101.615, 98.6436),

MRT = c(175.72, 170.62, 165.49, 160, 153.82, 147.06, 140.41, 134.58, 129.78, 126.19, 123.84, 122.37, 121.31, 120.48, 119.73, 118.82, 117.5, 115.76, 113.16, 109.8, 105.66, 100.63, 95.058, 89.264, 83.611, 78.196, 73.4, 69.391, 66.123, 63.472, 61.309, 59.545, 57.979, 56.456, 54.91, 53.383, 51.799, 50.261, 48.754, 47.219, 45.825, 44.399, 43.022),

CPV = c(86.06, 84.55, 83.59, 82.68, 81.32, 79.32, 76.47, 72.82, 68.97, 65.45, 62.63, 60.73, 59.71, 59.1, 58.28, 56.7, 54.03, 50.18, 45.69, 41.2, 37.14, 33.89, 31.6, 30.2, 29.52, 29.35, 29.37, 29.38, 29.18, 28.57, 27.6, 26.37, 24.9, 23.33, 21.67, 20.08, 18.61, 17.35, 16.25, 15.33, 14.58, 13.94, 13.39),

TCD = c(249.96, 247.91, 245.44, 242.78, 239.9, 236.96, 234.08, 231.21, 228.28, 224.99, 221.77, 218.58, 215.61, 212.64, 209.84, 207.23, 204.53, 202.01, 199.12, 196.02, 192.78, 189.26, 185.93, 182.59, 179, 175.35, 171.4, 167.61, 163.66, 159.59, 155.43, 151.39, 147.39, 143.58, 139.72, 135.94, 132.23, 128.24, 124.19, 120.39, 116.63, 112.88, 109.33),

CIV = c(181.68, 177.17, 173.54, 170.72, 168.58, 166.68, 165.37, 164.74, 164.68, 164.99, 165.31, 165.47, 165.35, 165.32, 165, 164.22, 162.93, 161.05, 158.62, 155.67, 152.55, 149.01, 145.28, 141.51, 137.54, 133.48, 129.6, 125.59, 121.49, 116.87, 112.86, 109.34, 106, 102.43, 98.739, 95.379, 92.666, 90.033, 87.174, 84.164, 81.657, 79.277, 76.893),

GNQ = c(210.40, 210.4, 210.4, 206.2, 201.85, 198.09, 195.07, 192.62, 190.47, 188.77, 187.05, 185.44, 183.9, 182.51, 181.08, 179.5, 177.39, 174.89, 171.44, 167.91, 163.84, 159.69, 155.21, 150.74, 146.19, 141.4, 136.61, 131.77, 127.04, 122.63, 118.57, 114.55, 110.62, 106.86, 103.37, 99.931, 96.594, 93.446, 90.2, 87.159, 84.438, 81.857, 79.273),

GMB = c(240.36, 233.75, 227.15, 220.51, 213.83, 207.16, 200.42, 194.03, 187.83, 181.77, 175.79, 169.85, 163.82, 157.85, 151.91, 146.3, 140.8, 135.37, 130.17, 125.16, 120.2, 115.39, 110.75, 106.26, 101.89, 97.669, 93.579, 89.685, 86.001, 82.434, 79.091, 75.963, 72.905, 70.031, 67.323, 64.698, 62.246, 59.855, 57.645, 55.583, 53.628, 51.819, 50.137),

GHA = c(175.08, 173.71, 172.37, 170.44, 167.51, 163.42, 158.24, 152.45, 146.45, 140.61, 135.5, 131.35, 128.09, 125.85, 124.09, 122.43, 120.47, 118.1, 115.07, 111.55, 107.61, 103.46, 99.584, 96.236, 93.358, 90.743, 88.311, 85.723, 82.855, 79.624, 76.175, 72.679, 69.287, 66.103, 63.205, 60.61, 58.142, 55.832, 53.737, 51.854, 49.954, 48.205, 46.67),

NER = c(324.44, 324.26, 326.35, 330.26, 334.87, 339.29, 342.28, 343.1, 341.85, 338.55, 333.72, 327.93, 319.82, 309.77, 297.67, 284.64, 272.09, 260.7, 250.34, 240.77, 231.36, 221.43, 210.15, 198.28, 186.55, 175.33, 165.25, 156.58, 149.47, 143.91, 139.66, 136.56, 134.29, 132.54, 131.04, 129.71, 128.7, 127.5, 126.42, 125.42, 124.27, 122.71, 120.54),

NGA = c(223.14, 219.95, 217.96, 217.03, 217.09, 217.75, 218.78, 219.94, 220.46, 220.63, 220.41, 219.69, 218.6, 217.13, 215.43, 213.24, 210.34, 206.43, 202.09, 196.74, 190.96, 185.39, 179.54, 173.98, 168.45, 162.98, 158.07, 153.31, 149.1, 145.54, 142.56, 140.04, 137.88, 136.07, 134.46, 132.85, 131.03, 128.7, 126.25, 123.14, 120.11, 116.75, 113.24),

SEN = c(211.62, 205.06, 198.96, 193.02, 186.56, 179.07, 170.65, 162.2, 154.98, 149.37, 145.41, 143.37, 142.75, 143.09, 143.95, 145.06, 145.67, 145.46, 144.02, 140.99, 136, 129.15, 121.23, 112.89, 104.73, 97.138, 90.494, 84.651, 79.607, 75.247, 71.265, 67.703, 64.387, 61.287, 58.381, 55.462, 52.681, 50.216, 47.873, 45.804, 43.966, 42.293, 40.775),

SLE = c(300.39, 296.73, 293.46, 290.18, 286.98, 284.2, 281.51, 278.99, 276.47, 274.32, 272.19, 270.38, 268.39, 265.99, 263.04, 259.49, 255.39, 250.9, 245.93, 240.48, 235.21, 229.3, 223.42, 217.22, 210.66, 204.11, 197.26, 190.27, 183.01, 175.55, 168.57, 161.66, 155.12, 148.96, 147.04, 147.99, 133.07, 128.21, 123.65, 119.3, 115.07, 111.22, 107.51),

TGO = c(187.53, 183.2, 179.06, 175.14, 171.48, 168.16, 165.26, 162.69, 160.21, 158.02, 155.99, 153.96, 152, 149.99, 147.7, 145.07, 142.2, 138.9, 135.33, 131.82, 128.14, 124.36, 120.6, 116.95, 113.45, 110.11, 106.82, 103.76, 100.73, 97.691, 94.78, 92, 89.156, 86.428, 83.783, 81.165, 78.681, 76.289, 73.919, 71.605, 69.329, 67.244, 65.13))

**# Prepare the Data for Plotting**

years <- 1980:2022

plot\_data <- data %>%

as.data.frame() %>%

mutate(Year = years) %>%

pivot\_longer(-Year, names\_to = "Country", values\_to = "MortalityRate")

ggplot(plot\_data, aes(x = Year, y = MortalityRate, color = Country)) +geom\_line(size = 1) +theme\_minimal() +labs(title = "Under-5 Male Mortality Rate",x = "Year",y = "Mortality Rate (per 1000 live births)",color = "Country") +theme(plot.title = element\_text(hjust = 0.5, size = 10, face = "bold"),axis.title = element\_text(size = 10),axis.text = element\_text(size = 10),legend.position = "right")

**# Calculat the Means and Standard Deviations**

means <- sapply(data, mean)

sds <- sapply(data, sd)

results <- data.frame(Country = names(data), Mean = means, SD = sds)

print(results)

**# K-means Clustering**

set.seed(123)

data\_df <- as.data.frame(data)

data\_df$Year <- seq(1980, 2022)

data\_long <- melt(data\_df, id.vars = "Year", variable.name = "Country", value.name = "Value")

transposed\_data <- as.data.frame(t(data\_df[,-ncol(data\_df)]))

rownames(transposed\_data) <- colnames(data\_df)[-ncol(data\_df)]

colnames(transposed\_data) <- seq(1980, 2022)

transposed\_data\_scaled <- scale(transposed\_data)

k <- 4 # Number of clusters

kmeans\_result <- kmeans(transposed\_data\_scaled, centers = k)

cluster\_assignments <- data.frame(Country = rownames(transposed\_data), Cluster = kmeans\_result$cluster)

print(cluster\_assignments)

data\_long <- left\_join(data\_long, cluster\_assignments, by = "Country")

ggplot(data\_long, aes(x = Year, y = Value, color = factor(Cluster), group = Country)) +geom\_line() +labs(title = "K-means Clustering of Under-5 Male Mortality",x = "Year",y = "Mortality Rate",color = "Cluster") +theme\_minimal() +theme(plot.title = element\_text(hjust = 0.5, size = 10, face = "bold"),axis.title = element\_text(size = 10),axis.text = element\_text(size = 10),legend.position = "right")

**# Resilience index for each country**

set.seed(123)

calculate\_resilience <- function(values) {

start <- values[1]

end <- values[length(values)]

(start - end) / start \* 100}

resilience\_index <- sapply(data, calculate\_resilience)

ranks <- rank(-resilience\_index, ties.method = "min")

resilience\_df <- data.frame(Country = names(resilience\_index),Resilience\_Index = resilience\_index,Rank = ranks)

results <- resilience\_df[order(results$Rank), ]

print(results)

ggplot(resilience\_df, aes(x = reorder(Country, Resilience\_Index), y = Resilience\_Index)) +

geom\_bar(stat = "identity", fill = "steelblue") +

coord\_flip() +

labs(title = "Resilience Index of Male U5MR by Country", x = "Country", y = "Resilience Index (%)") +

theme\_minimal()

**# Out-of-Sample Forecast for BEN UN-5 Male Mortality Rates**

BEN = c(222.61, 218.26, 214.02, 209.88, 205.56, 201.31, 196.89, 192.56, 188.19, 183.71, 179.31, 174.92, 170.57, 166.25, 162.28, 158.55, 155.01, 151.5, 148.16, 145.25, 142.29, 139.35, 136.26, 133.13, 130.12, 127.33, 124.65, 122.14, 119.87, 117.63, 115.36, 113.17, 110.87, 108.57, 106.29, 104.03, 101.72, 99.419, 96.985, 94.427, 91.807, 89.172, 86.516)

ben\_ts <- ts(BEN, start = 1980, frequency = 1)

training\_data <- window(ben\_ts, start=1980, end=2017)

testing\_data <- window(ben\_ts, start=2018, end=2022)

arima\_model <- auto.arima(training\_data)

arima\_forecast <- forecast(arima\_model, h = length(testing\_data))$mean

ets\_model <- ets(training\_data)

ets\_forecast <- forecast(ets\_model, h = length(testing\_data))$mean

mlp\_model <- mlp(training\_data)

mlp\_forecast <- forecast(mlp\_model, h = length(testing\_data))$mean

train\_df <- data.frame(

ds = seq.Date(from = as.Date("1980-01-01"), by = "year", length.out = length(training\_data)),

y = as.numeric(training\_data))

prophet\_model <- prophet(train\_df, yearly.seasonality = FALSE, weekly.seasonality = FALSE, daily.seasonality = FALSE)

future <- data.frame(ds = seq.Date(from = max(train\_df$ds) + 1, by = "year", length.out = length(testing\_data)))

prophet\_forecast <- predict(prophet\_model, future)$yhat

lagged\_data <- data.frame(Target = training\_data[-1],Lag1 = training\_data[-length(training\_data)])

xgb\_model <- xgboost(data = as.matrix(lagged\_data$Lag1), label = lagged\_data$Target, nrounds = 100)

xgb\_forecast <- numeric(length(testing\_data))

last\_value <- training\_data[length(training\_data)]

for (i in 1:length(testing\_data)) {

xgb\_forecast[i] <- predict(xgb\_model, newdata = as.matrix(last\_value))

last\_value <- xgb\_forecast[i]}

hybrid\_forecast\_function <- function(weights, f1, f2) {

weights[1] \* as.numeric(f1) + weights[2] \* as.numeric(f2)}

fitness\_function <- function(weights, f1, f2, actual) {

hybrid\_forecast <- hybrid\_forecast\_function(weights, f1, f2)

sqrt(mean((actual - hybrid\_forecast)^2))}

optimize\_weights <- function(f1, f2, actual) {

ga\_model <- ga(type = "real-valued",fitness = function(w) -fitness\_function(w, f1, f2, actual),lower = c(0, 0),upper = c(1, 1),popSize = 50,maxiter = 100)

return(ga\_model@solution)}

weights <- optimize\_weights(arima\_forecast, ets\_forecast, testing\_data)

arima\_ets\_forecast <- hybrid\_forecast\_function(weights, arima\_forecast, ets\_forecast)

weights <- optimize\_weights(arima\_forecast, mlp\_forecast, testing\_data)

arima\_mlp\_forecast <- hybrid\_forecast\_function(weights, arima\_forecast, mlp\_forecast)

weights <- optimize\_weights(arima\_forecast, xgb\_forecast, testing\_data)

arima\_xgb\_forecast <- hybrid\_forecast\_function(weights, arima\_forecast, xgb\_forecast)

weights <- optimize\_weights(arima\_forecast, prophet\_forecast, testing\_data)

arima\_prophet\_forecast <- hybrid\_forecast\_function(weights, arima\_forecast, prophet\_forecast)

calculate\_metrics <- function(actual, forecast) {

rmse <- sqrt(mean((actual - forecast)^2))

mape <- mean(abs((actual - forecast) / actual)) \* 100

return(c(RMSE = rmse, MAPE = mape))}

model\_metrics <- data.frame(

Model = c("ARIMA", "ETS", "MLP", "Prophet", "XGBoost", "ARIMA\_ETS", "ARIMA\_MLP", "ARIMA\_XGBoost", "ARIMA\_Prophet"),

do.call(rbind, list(

calculate\_metrics(testing\_data, arima\_forecast),

calculate\_metrics(testing\_data, ets\_forecast),

calculate\_metrics(testing\_data, mlp\_forecast),

calculate\_metrics(testing\_data, prophet\_forecast),

calculate\_metrics(testing\_data, xgb\_forecast),

calculate\_metrics(testing\_data, arima\_ets\_forecast),

calculate\_metrics(testing\_data, arima\_mlp\_forecast),

calculate\_metrics(testing\_data, arima\_xgb\_forecast),

calculate\_metrics(testing\_data, arima\_prophet\_forecast))))

print("Model Performance Metrics (RMSE and MAE):")

print(model\_metrics)

all\_forecasts <- data.frame(

Year = seq(2018, 2022),

Actual = as.numeric(testing\_data),

ARIMA = as.numeric(arima\_forecast),

ETS = as.numeric(ets\_forecast),

MLP = as.numeric(mlp\_forecast),

Prophet = as.numeric(prophet\_forecast),

XGBoost = as.numeric(xgb\_forecast),

ARIMA\_ETS = as.numeric(arima\_ets\_forecast),

ARIMA\_MLP = as.numeric(arima\_mlp\_forecast),

ARIMA\_Prophet = as.numeric(arima\_prophet\_forecast),

ARIMA\_XGBoost = as.numeric(arima\_xgb\_forecast))

melted\_forecasts <- melt(all\_forecasts, id.vars = "Year", variable.name = "Model", value.name = "Value")

ggplot(data = melted\_forecasts, aes(x = Year, y = Value, color = Model)) +

geom\_line() +

labs(title = "BEN", y = "Mortality Rate", x = "Year") +

theme\_minimal()

**# Out-of-Sample Forecast for BFA UN-5 Male Mortality Rates**

BFA = c(248.21, 241.81, 236.12, 230.91, 225.96, 221.39, 216.92, 212.89, 209.53, 207.06, 205.35, 204.24, 203.63, 202.75, 201.63, 199.56, 197.01, 194.02, 190.97, 187.65, 184.08, 180.11, 175.79, 170.6, 164.54, 158.01, 151.01, 144.1, 137.52, 131.45, 126.02, 121.26, 116.9, 112.83, 109.03, 105.42, 101.74, 98.319, 94.992, 92.171, 89.15, 86.099, 83.342)

set.seed(123)

bfa\_ts <- ts(BFA, start = 1980, frequency = 1)

training\_data <- window(bfa\_ts, start=1980, end=2017)

testing\_data <- window(bfa\_ts, start=2018, end=2022)

arima\_model <- auto.arima(training\_data)

arima\_forecast <- forecast(arima\_model, h = length(testing\_data))$mean

ets\_model <- ets(training\_data)

ets\_forecast <- forecast(ets\_model, h = length(testing\_data))$mean

mlp\_model <- mlp(training\_data)

mlp\_forecast <- forecast(mlp\_model, h = length(testing\_data))$mean

train\_df <- data.frame(

ds = seq.Date(from = as.Date("1980-01-01"), by = "year", length.out = length(training\_data)),

y = as.numeric(training\_data))

prophet\_model <- prophet(train\_df, yearly.seasonality = FALSE, weekly.seasonality = FALSE, daily.seasonality = FALSE)

future <- data.frame(ds = seq.Date(from = max(train\_df$ds) + 1, by = "year", length.out = length(testing\_data)))

prophet\_forecast <- predict(prophet\_model, future)$yhat

lagged\_data <- data.frame(

Target = training\_data[-1],

Lag1 = training\_data[-length(training\_data)])

xgb\_model <- xgboost(data = as.matrix(lagged\_data$Lag1), label = lagged\_data$Target, nrounds = 100)

xgb\_forecast <- numeric(length(testing\_data))

last\_value <- training\_data[length(training\_data)]

for (i in 1:length(testing\_data)) {

xgb\_forecast[i] <- predict(xgb\_model, newdata = as.matrix(last\_value))

last\_value <- xgb\_forecast[i]}

hybrid\_forecast\_function <- function(weights, f1, f2) {

weights[1] \* as.numeric(f1) + weights[2] \* as.numeric(f2)}

fitness\_function <- function(weights, f1, f2, actual) {

hybrid\_forecast <- hybrid\_forecast\_function(weights, f1, f2)

sqrt(mean((actual - hybrid\_forecast)^2))}

optimize\_weights <- function(f1, f2, actual) {

ga\_model <- ga(

type = "real-valued",

fitness = function(w) -fitness\_function(w, f1, f2, actual),

lower = c(0, 0),

upper = c(1, 1),

popSize = 50,

maxiter = 100)

return(ga\_model@solution)}

weights <- optimize\_weights(arima\_forecast, ets\_forecast, testing\_data)

arima\_ets\_forecast <- hybrid\_forecast\_function(weights, arima\_forecast, ets\_forecast)

weights <- optimize\_weights(arima\_forecast, mlp\_forecast, testing\_data)

arima\_mlp\_forecast <- hybrid\_forecast\_function(weights, arima\_forecast, mlp\_forecast)

weights <- optimize\_weights(arima\_forecast, xgb\_forecast, testing\_data)

arima\_xgb\_forecast <- hybrid\_forecast\_function(weights, arima\_forecast, xgb\_forecast)

weights <- optimize\_weights(arima\_forecast, prophet\_forecast, testing\_data)

arima\_prophet\_forecast <- hybrid\_forecast\_function(weights, arima\_forecast, prophet\_forecast)

calculate\_metrics <- function(actual, forecast) {

rmse <- sqrt(mean((actual - forecast)^2))

mape <- mean(abs((actual - forecast) / actual)) \* 100

return(c(RMSE = rmse, MAPE = mape))}

model\_metrics <- data.frame(

Model = c("ARIMA", "ETS", "MLP", "Prophet", "XGBoost", "ARIMA\_ETS", "ARIMA\_MLP", "ARIMA\_XGBoost", "ARIMA\_Prophet"),

do.call(rbind, list(

calculate\_metrics(testing\_data, arima\_forecast),

calculate\_metrics(testing\_data, ets\_forecast),

calculate\_metrics(testing\_data, mlp\_forecast),

calculate\_metrics(testing\_data, prophet\_forecast),

calculate\_metrics(testing\_data, xgb\_forecast),

calculate\_metrics(testing\_data, arima\_ets\_forecast),

calculate\_metrics(testing\_data, arima\_mlp\_forecast),

calculate\_metrics(testing\_data, arima\_xgb\_forecast),

calculate\_metrics(testing\_data, arima\_prophet\_forecast))))

print("Model Performance Metrics (RMSE and MAE):")

print(model\_metrics)

all\_forecasts <- data.frame(

Year = seq(2018, 2022),

Actual = as.numeric(testing\_data),

ARIMA = as.numeric(arima\_forecast),

ETS = as.numeric(ets\_forecast),

MLP = as.numeric(mlp\_forecast),

Prophet = as.numeric(prophet\_forecast),

XGBoost = as.numeric(xgb\_forecast),

ARIMA\_ETS = as.numeric(arima\_ets\_forecast),

ARIMA\_MLP = as.numeric(arima\_mlp\_forecast),

ARIMA\_Prophet = as.numeric(arima\_prophet\_forecast),

ARIMA\_XGBoost = as.numeric(arima\_xgb\_forecast))

melted\_forecasts <- melt(all\_forecasts, id.vars = "Year", variable.name = "Model", value.name = "Value")

ggplot(data = melted\_forecasts, aes(x = Year, y = Value, color = Model)) +geom\_line() +labs(title = "BFA", y = "Mortality Rate", x = "Year") +theme\_minimal()

**# Out-of-Sample Forecast for GIN UN-5 Male Mortality Rates**

set.seed(123)

GIN = c(289.64, 284.73, 279.85, 274.71, 269.65, 264.85, 260.11, 255.59, 251.02, 246.14, 240.82, 235.09, 228.6, 222.07, 215.38, 208.5, 201.67, 194.82, 187.96, 180.99, 174.08, 167.36, 161.06, 155.22, 150.02, 145.41, 141.47, 138.11, 135.23, 132.65, 130.27, 127.9, 125.77, 123.68, 122.32, 120.47, 117.99, 116.09, 113.88, 111.45, 108.85, 106.06, 103.14)

gin\_ts <- ts(GIN, start = 1980, frequency = 1)

training\_data <- window(gin\_ts, start=1980, end=2017)

testing\_data <- window(gin\_ts, start=2018, end=2022)

arima\_model <- auto.arima(training\_data)

arima\_forecast <- forecast(arima\_model, h = length(testing\_data))$mean

ets\_model <- ets(training\_data)

ets\_forecast <- forecast(ets\_model, h = length(testing\_data))$mean

mlp\_model <- mlp(training\_data)

mlp\_forecast <- forecast(mlp\_model, h = length(testing\_data))$mean

train\_df <- data.frame(

ds = seq.Date(from = as.Date("1980-01-01"), by = "year", length.out = length(training\_data)),

y = as.numeric(training\_data))

prophet\_model <- prophet(train\_df, yearly.seasonality = FALSE, weekly.seasonality = FALSE, daily.seasonality = FALSE)

future <- data.frame(ds = seq.Date(from = max(train\_df$ds) + 1, by = "year", length.out = length(testing\_data)))

prophet\_forecast <- predict(prophet\_model, future)$yhat

lagged\_data <- data.frame(

Target = training\_data[-1],

Lag1 = training\_data[-length(training\_data)])

xgb\_model <- xgboost(data = as.matrix(lagged\_data$Lag1), label = lagged\_data$Target, nrounds = 100)

xgb\_forecast <- numeric(length(testing\_data))

last\_value <- training\_data[length(training\_data)]

for (i in 1:length(testing\_data)) {

xgb\_forecast[i] <- predict(xgb\_model, newdata = as.matrix(last\_value))

last\_value <- xgb\_forecast[i]}

hybrid\_forecast\_function <- function(weights, f1, f2) {

weights[1] \* as.numeric(f1) + weights[2] \* as.numeric(f2)}

fitness\_function <- function(weights, f1, f2, actual) {

hybrid\_forecast <- hybrid\_forecast\_function(weights, f1, f2)

sqrt(mean((actual - hybrid\_forecast)^2))}

optimize\_weights <- function(f1, f2, actual) {

ga\_model <- ga(

type = "real-valued",

fitness = function(w) -fitness\_function(w, f1, f2, actual),

lower = c(0, 0),

upper = c(1, 1),

popSize = 50,

maxiter = 100)

return(ga\_model@solution)}

weights <- optimize\_weights(arima\_forecast, ets\_forecast, testing\_data)

arima\_ets\_forecast <- hybrid\_forecast\_function(weights, arima\_forecast, ets\_forecast)

weights <- optimize\_weights(arima\_forecast, mlp\_forecast, testing\_data)

arima\_mlp\_forecast <- hybrid\_forecast\_function(weights, arima\_forecast, mlp\_forecast)

weights <- optimize\_weights(arima\_forecast, xgb\_forecast, testing\_data)

arima\_xgb\_forecast <- hybrid\_forecast\_function(weights, arima\_forecast, xgb\_forecast)

weights <- optimize\_weights(arima\_forecast, prophet\_forecast, testing\_data)

arima\_prophet\_forecast <- hybrid\_forecast\_function(weights, arima\_forecast, prophet\_forecast)

calculate\_metrics <- function(actual, forecast) {

rmse <- sqrt(mean((actual - forecast)^2))

mape <- mean(abs((actual - forecast) / actual)) \* 100

return(c(RMSE = rmse, MAPE = mape))}

model\_metrics <- data.frame(

Model = c("ARIMA", "ETS", "MLP", "Prophet", "XGBoost", "ARIMA\_ETS", "ARIMA\_MLP", "ARIMA\_XGBoost", "ARIMA\_Prophet"),

do.call(rbind, list(

calculate\_metrics(testing\_data, arima\_forecast),

calculate\_metrics(testing\_data, ets\_forecast),

calculate\_metrics(testing\_data, mlp\_forecast),

calculate\_metrics(testing\_data, prophet\_forecast),

calculate\_metrics(testing\_data, xgb\_forecast),

calculate\_metrics(testing\_data, arima\_ets\_forecast),

calculate\_metrics(testing\_data, arima\_mlp\_forecast),

calculate\_metrics(testing\_data, arima\_xgb\_forecast),

calculate\_metrics(testing\_data, arima\_prophet\_forecast))))

print("Model Performance Metrics (RMSE and MAE):")

print(model\_metrics)

all\_forecasts <- data.frame(

Year = seq(2018, 2022),

Actual = as.numeric(testing\_data),

ARIMA = as.numeric(arima\_forecast),

ETS = as.numeric(ets\_forecast),

MLP = as.numeric(mlp\_forecast),

Prophet = as.numeric(prophet\_forecast),

XGBoost = as.numeric(xgb\_forecast),

ARIMA\_ETS = as.numeric(arima\_ets\_forecast),

ARIMA\_MLP = as.numeric(arima\_mlp\_forecast),

ARIMA\_Prophet = as.numeric(arima\_prophet\_forecast),

ARIMA\_XGBoost = as.numeric(arima\_xgb\_forecast))

melted\_forecasts <- melt(all\_forecasts, id.vars = "Year", variable.name = "Model", value.name = "Value")

ggplot(data = melted\_forecasts, aes(x = Year, y = Value, color = Model)) +

geom\_line() +

labs(title = "GIN", y = "Mortality Rate", x = "Year") +

theme\_minimal()

**# Out-of-Sample Forecast for GNB UN-5 Male Mortality Rates**

set.seed(123)

GNB = c(255.03, 255.03, 255.03, 255.03, 255.03, 255.03, 250.16, 245.56, 241.3, 237.17, 232.89, 228.42, 223.74, 219.02, 214.19, 209.2, 204.12, 199.12, 194.21, 189.18, 183.82, 178.4, 172.82, 166.88, 160.78, 154.57, 148.15, 141.69, 135.22, 128.94, 122.76, 117.07, 111.81, 107.22, 102.82, 99.118, 95.589, 92.405, 89.207, 86.104, 83.296, 80.465, 77.896)

gnb\_ts <- ts(GNB, start = 1980, frequency = 1)

training\_data <- window(gnb\_ts, start=1980, end=2017)

testing\_data <- window(gnb\_ts, start=2018, end=2022)

arima\_model <- auto.arima(training\_data)

arima\_forecast <- forecast(arima\_model, h = length(testing\_data))$mean

ets\_model <- ets(training\_data)

ets\_forecast <- forecast(ets\_model, h = length(testing\_data))$mean

mlp\_model <- mlp(training\_data)

mlp\_forecast <- forecast(mlp\_model, h = length(testing\_data))$mean

train\_df <- data.frame(

ds = seq.Date(from = as.Date("1980-01-01"), by = "year", length.out = length(training\_data)),

y = as.numeric(training\_data))

prophet\_model <- prophet(train\_df, yearly.seasonality = FALSE, weekly.seasonality = FALSE, daily.seasonality = FALSE)

future <- data.frame(ds = seq.Date(from = max(train\_df$ds) + 1, by = "year", length.out = length(testing\_data)))

prophet\_forecast <- predict(prophet\_model, future)$yhat

lagged\_data <- data.frame(

Target = training\_data[-1],

Lag1 = training\_data[-length(training\_data)])

xgb\_model <- xgboost(data = as.matrix(lagged\_data$Lag1), label = lagged\_data$Target, nrounds = 100)

xgb\_forecast <- numeric(length(testing\_data))

last\_value <- training\_data[length(training\_data)]

for (i in 1:length(testing\_data)) {

xgb\_forecast[i] <- predict(xgb\_model, newdata = as.matrix(last\_value))

last\_value <- xgb\_forecast[i]}

hybrid\_forecast\_function <- function(weights, f1, f2) {

weights[1] \* as.numeric(f1) + weights[2] \* as.numeric(f2)}

fitness\_function <- function(weights, f1, f2, actual) {

hybrid\_forecast <- hybrid\_forecast\_function(weights, f1, f2)

sqrt(mean((actual - hybrid\_forecast)^2))}

optimize\_weights <- function(f1, f2, actual) {

ga\_model <- ga(

type = "real-valued",

fitness = function(w) -fitness\_function(w, f1, f2, actual),

lower = c(0, 0),

upper = c(1, 1),

popSize = 50,

maxiter = 100)

return(ga\_model@solution)}

weights <- optimize\_weights(arima\_forecast, ets\_forecast, testing\_data)

arima\_ets\_forecast <- hybrid\_forecast\_function(weights, arima\_forecast, ets\_forecast)

weights <- optimize\_weights(arima\_forecast, mlp\_forecast, testing\_data)

arima\_mlp\_forecast <- hybrid\_forecast\_function(weights, arima\_forecast, mlp\_forecast)

weights <- optimize\_weights(arima\_forecast, xgb\_forecast, testing\_data)

arima\_xgb\_forecast <- hybrid\_forecast\_function(weights, arima\_forecast, xgb\_forecast)

weights <- optimize\_weights(arima\_forecast, prophet\_forecast, testing\_data)

arima\_prophet\_forecast <- hybrid\_forecast\_function(weights, arima\_forecast, prophet\_forecast)

calculate\_metrics <- function(actual, forecast) {

rmse <- sqrt(mean((actual - forecast)^2))

mape <- mean(abs((actual - forecast) / actual)) \* 100

return(c(RMSE = rmse, MAPE = mape))}

model\_metrics <- data.frame(

Model = c("ARIMA", "ETS", "MLP", "Prophet", "XGBoost", "ARIMA\_ETS", "ARIMA\_MLP", "ARIMA\_XGBoost", "ARIMA\_Prophet"),

do.call(rbind, list(

calculate\_metrics(testing\_data, arima\_forecast),

calculate\_metrics(testing\_data, ets\_forecast),

calculate\_metrics(testing\_data, mlp\_forecast),

calculate\_metrics(testing\_data, prophet\_forecast),

calculate\_metrics(testing\_data, xgb\_forecast),

calculate\_metrics(testing\_data, arima\_ets\_forecast),

calculate\_metrics(testing\_data, arima\_mlp\_forecast),

calculate\_metrics(testing\_data, arima\_xgb\_forecast),

calculate\_metrics(testing\_data, arima\_prophet\_forecast))))

print("Model Performance Metrics (RMSE and MAE):")

print(model\_metrics)

all\_forecasts <- data.frame(

Year = seq(2018, 2022),

Actual = as.numeric(testing\_data),

ARIMA = as.numeric(arima\_forecast),

ETS = as.numeric(ets\_forecast),

MLP = as.numeric(mlp\_forecast),

Prophet = as.numeric(prophet\_forecast),

XGBoost = as.numeric(xgb\_forecast),

ARIMA\_ETS = as.numeric(arima\_ets\_forecast),

ARIMA\_MLP = as.numeric(arima\_mlp\_forecast),

ARIMA\_Prophet = as.numeric(arima\_prophet\_forecast),

ARIMA\_XGBoost = as.numeric(arima\_xgb\_forecast))

melted\_forecasts <- melt(all\_forecasts, id.vars = "Year", variable.name = "Model", value.name = "Value")

ggplot(data = melted\_forecasts, aes(x = Year, y = Value, color = Model)) +

geom\_line() +

labs(title = "GNB", y = "Mortality Rate", x = "Year") +

theme\_minimal()

**# Out-of-Sample Forecast for LBR UN-5 Male Mortality Rates**

set.seed(123)

LBR = c(257.9, 254.9, 252.63, 251.31, 251.38, 253.12, 256.79, 261.92, 267.8, 273.4, 277.82, 279.7, 278.8, 275.22, 268.86, 260.27, 249.95, 238.37, 225.77, 212.9, 199.78, 186.18, 172.89, 160.19, 148.35, 137.69, 128.73, 121.25, 115.23, 110.34, 106.43, 103.27, 100.56, 98.226, 98.232, 94.286, 92.204, 90.23, 88.193, 85.849, 83.494, 81.188, 78.972)

lbr\_ts <- ts(LBR, start = 1980, frequency = 1)

training\_data <- window(lbr\_ts, start=1980, end=2017)

testing\_data <- window(lbr\_ts, start=2018, end=2022)

arima\_model <- auto.arima(training\_data)

arima\_forecast <- forecast(arima\_model, h = length(testing\_data))$mean

ets\_model <- ets(training\_data)

ets\_forecast <- forecast(ets\_model, h = length(testing\_data))$mean

mlp\_model <- mlp(training\_data)

mlp\_forecast <- forecast(mlp\_model, h = length(testing\_data))$mean

train\_df <- data.frame(

ds = seq.Date(from = as.Date("1980-01-01"), by = "year", length.out = length(training\_data)),

y = as.numeric(training\_data))

prophet\_model <- prophet(train\_df, yearly.seasonality = FALSE, weekly.seasonality = FALSE, daily.seasonality = FALSE)

future <- data.frame(ds = seq.Date(from = max(train\_df$ds) + 1, by = "year", length.out = length(testing\_data)))

prophet\_forecast <- predict(prophet\_model, future)$yhat

lagged\_data <- data.frame(

Target = training\_data[-1],

Lag1 = training\_data[-length(training\_data)])

xgb\_model <- xgboost(data = as.matrix(lagged\_data$Lag1), label = lagged\_data$Target, nrounds = 100)

xgb\_forecast <- numeric(length(testing\_data))

last\_value <- training\_data[length(training\_data)]

for (i in 1:length(testing\_data)) {

xgb\_forecast[i] <- predict(xgb\_model, newdata = as.matrix(last\_value))

last\_value <- xgb\_forecast[i]}

hybrid\_forecast\_function <- function(weights, f1, f2) {

weights[1] \* as.numeric(f1) + weights[2] \* as.numeric(f2)}

fitness\_function <- function(weights, f1, f2, actual) {

hybrid\_forecast <- hybrid\_forecast\_function(weights, f1, f2)

sqrt(mean((actual - hybrid\_forecast)^2))}

optimize\_weights <- function(f1, f2, actual) {

ga\_model <- ga(

type = "real-valued",

fitness = function(w) -fitness\_function(w, f1, f2, actual),

lower = c(0, 0),

upper = c(1, 1),

popSize = 50,

maxiter = 100)

return(ga\_model@solution)}

weights <- optimize\_weights(arima\_forecast, ets\_forecast, testing\_data)

arima\_ets\_forecast <- hybrid\_forecast\_function(weights, arima\_forecast, ets\_forecast)

weights <- optimize\_weights(arima\_forecast, mlp\_forecast, testing\_data)

arima\_mlp\_forecast <- hybrid\_forecast\_function(weights, arima\_forecast, mlp\_forecast)

weights <- optimize\_weights(arima\_forecast, xgb\_forecast, testing\_data)

arima\_xgb\_forecast <- hybrid\_forecast\_function(weights, arima\_forecast, xgb\_forecast)

weights <- optimize\_weights(arima\_forecast, prophet\_forecast, testing\_data)

arima\_prophet\_forecast <- hybrid\_forecast\_function(weights, arima\_forecast, prophet\_forecast)

calculate\_metrics <- function(actual, forecast) {

rmse <- sqrt(mean((actual - forecast)^2))

mape <- mean(abs((actual - forecast) / actual)) \* 100

return(c(RMSE = rmse, MAPE = mape))}

model\_metrics <- data.frame(

Model = c("ARIMA", "ETS", "MLP", "Prophet", "XGBoost", "ARIMA\_ETS", "ARIMA\_MLP", "ARIMA\_XGBoost", "ARIMA\_Prophet"),

do.call(rbind, list(

calculate\_metrics(testing\_data, arima\_forecast),

calculate\_metrics(testing\_data, ets\_forecast),

calculate\_metrics(testing\_data, mlp\_forecast),

calculate\_metrics(testing\_data, prophet\_forecast),

calculate\_metrics(testing\_data, xgb\_forecast),

calculate\_metrics(testing\_data, arima\_ets\_forecast),

calculate\_metrics(testing\_data, arima\_mlp\_forecast),

calculate\_metrics(testing\_data, arima\_xgb\_forecast),

calculate\_metrics(testing\_data, arima\_prophet\_forecast))))

print("Model Performance Metrics (RMSE and MAE):")

print(model\_metrics)

all\_forecasts <- data.frame(

Year = seq(2018, 2022),

Actual = as.numeric(testing\_data),

ARIMA = as.numeric(arima\_forecast),

ETS = as.numeric(ets\_forecast),

MLP = as.numeric(mlp\_forecast),

Prophet = as.numeric(prophet\_forecast),

XGBoost = as.numeric(xgb\_forecast),

ARIMA\_ETS = as.numeric(arima\_ets\_forecast),

ARIMA\_MLP = as.numeric(arima\_mlp\_forecast),

ARIMA\_Prophet = as.numeric(arima\_prophet\_forecast),

ARIMA\_XGBoost = as.numeric(arima\_xgb\_forecast))

melted\_forecasts <- melt(all\_forecasts, id.vars = "Year", variable.name = "Model", value.name = "Value")

ggplot(data = melted\_forecasts, aes(x = Year, y = Value, color = Model)) +

geom\_line() +

labs(title = "LBR", y = "Mortality Rate", x = "Year") +

theme\_minimal()

**# Out-of-Sample Forecast for MLI UN-5 Female Mortality Rates**

set.seed(123)

MLI = c(308.142, 299.993, 291.887, 283.921, 276.039, 268.291, 261.071, 254.887, 249.398, 244.094, 239.312, 235.079, 231.208, 227.784, 224.089, 220.369, 216.334, 211.819, 206.747, 201.047, 194.925, 188.617, 182.162, 175.681, 169.571, 163.814, 158.41, 153.433, 148.558, 143.989, 139.753, 135.552, 131.75, 128.071, 124.425, 120.894, 117.44, 114.097, 111, 107.863, 104.577, 101.615, 98.6436)

mli\_ts <- ts(MLI, start = 1980, frequency = 1)

training\_data <- window(mli\_ts, start=1980, end=2017)

testing\_data <- window(mli\_ts, start=2018, end=2022)

arima\_model <- auto.arima(training\_data)

arima\_forecast <- forecast(arima\_model, h = length(testing\_data))$mean

ets\_model <- ets(training\_data)

ets\_forecast <- forecast(ets\_model, h = length(testing\_data))$mean

mlp\_model <- mlp(training\_data)

mlp\_forecast <- forecast(mlp\_model, h = length(testing\_data))$mean

train\_df <- data.frame(

ds = seq.Date(from = as.Date("1980-01-01"), by = "year", length.out = length(training\_data)),

y = as.numeric(training\_data))

prophet\_model <- prophet(train\_df, yearly.seasonality = FALSE, weekly.seasonality = FALSE, daily.seasonality = FALSE)

future <- data.frame(ds = seq.Date(from = max(train\_df$ds) + 1, by = "year", length.out = length(testing\_data)))

prophet\_forecast <- predict(prophet\_model, future)$yhat

lagged\_data <- data.frame(

Target = training\_data[-1],

Lag1 = training\_data[-length(training\_data)])

xgb\_model <- xgboost(data = as.matrix(lagged\_data$Lag1), label = lagged\_data$Target, nrounds = 100)

xgb\_forecast <- numeric(length(testing\_data))

last\_value <- training\_data[length(training\_data)]

for (i in 1:length(testing\_data)) {

xgb\_forecast[i] <- predict(xgb\_model, newdata = as.matrix(last\_value))

last\_value <- xgb\_forecast[i]}

hybrid\_forecast\_function <- function(weights, f1, f2) {

weights[1] \* as.numeric(f1) + weights[2] \* as.numeric(f2)}

fitness\_function <- function(weights, f1, f2, actual) {

hybrid\_forecast <- hybrid\_forecast\_function(weights, f1, f2)

sqrt(mean((actual - hybrid\_forecast)^2))}

optimize\_weights <- function(f1, f2, actual) {

ga\_model <- ga(

type = "real-valued",

fitness = function(w) -fitness\_function(w, f1, f2, actual),

lower = c(0, 0),

upper = c(1, 1),

popSize = 50,

maxiter = 100)

return(ga\_model@solution)}

weights <- optimize\_weights(arima\_forecast, ets\_forecast, testing\_data)

arima\_ets\_forecast <- hybrid\_forecast\_function(weights, arima\_forecast, ets\_forecast)

weights <- optimize\_weights(arima\_forecast, mlp\_forecast, testing\_data)

arima\_mlp\_forecast <- hybrid\_forecast\_function(weights, arima\_forecast, mlp\_forecast)

weights <- optimize\_weights(arima\_forecast, xgb\_forecast, testing\_data)

arima\_xgb\_forecast <- hybrid\_forecast\_function(weights, arima\_forecast, xgb\_forecast)

weights <- optimize\_weights(arima\_forecast, prophet\_forecast, testing\_data)

arima\_prophet\_forecast <- hybrid\_forecast\_function(weights, arima\_forecast, prophet\_forecast)

calculate\_metrics <- function(actual, forecast) {

rmse <- sqrt(mean((actual - forecast)^2))

mape <- mean(abs((actual - forecast) / actual)) \* 100

return(c(RMSE = rmse, MAPE = mape))}

model\_metrics <- data.frame(

Model = c("ARIMA", "ETS", "MLP", "Prophet", "XGBoost", "ARIMA\_ETS", "ARIMA\_MLP", "ARIMA\_XGBoost", "ARIMA\_Prophet"),

do.call(rbind, list(

calculate\_metrics(testing\_data, arima\_forecast),

calculate\_metrics(testing\_data, ets\_forecast),

calculate\_metrics(testing\_data, mlp\_forecast),

calculate\_metrics(testing\_data, prophet\_forecast),

calculate\_metrics(testing\_data, xgb\_forecast),

calculate\_metrics(testing\_data, arima\_ets\_forecast),

calculate\_metrics(testing\_data, arima\_mlp\_forecast),

calculate\_metrics(testing\_data, arima\_xgb\_forecast),

calculate\_metrics(testing\_data, arima\_prophet\_forecast))))

print("Model Performance Metrics (RMSE and MAE):")

print(model\_metrics)

all\_forecasts <- data.frame(

Year = seq(2018, 2022),

Actual = as.numeric(testing\_data),

ARIMA = as.numeric(arima\_forecast),

ETS = as.numeric(ets\_forecast),

MLP = as.numeric(mlp\_forecast),

Prophet = as.numeric(prophet\_forecast),

XGBoost = as.numeric(xgb\_forecast),

ARIMA\_ETS = as.numeric(arima\_ets\_forecast),

ARIMA\_MLP = as.numeric(arima\_mlp\_forecast),

ARIMA\_Prophet = as.numeric(arima\_prophet\_forecast),

ARIMA\_XGBoost = as.numeric(arima\_xgb\_forecast))

melted\_forecasts <- melt(all\_forecasts, id.vars = "Year", variable.name = "Model", value.name = "Value")

ggplot(data = melted\_forecasts, aes(x = Year, y = Value, color = Model)) +

geom\_line() +

labs(title = "MLI", y = "Mortality Rate", x = "Year") +

theme\_minimal()

**# Out-of-Sample Forecast for MRT UN-5 Male Mortality Rates**

set.seed(123)

MRT = c(175.72, 170.62, 165.49, 160, 153.82, 147.06, 140.41, 134.58, 129.78, 126.19, 123.84, 122.37, 121.31, 120.48, 119.73, 118.82, 117.5, 115.76, 113.16, 109.8, 105.66, 100.63, 95.058, 89.264, 83.611, 78.196, 73.4, 69.391, 66.123, 63.472, 61.309, 59.545, 57.979, 56.456, 54.91, 53.383, 51.799, 50.261, 48.754, 47.219, 45.825, 44.399, 43.022)

mrt\_ts <- ts(MRT, start = 1980, frequency = 1)

training\_data <- window(mrt\_ts, start=1980, end=2017)

testing\_data <- window(mrt\_ts, start=2018, end=2022)

arima\_model <- auto.arima(training\_data)

arima\_forecast <- forecast(arima\_model, h = length(testing\_data))$mean

ets\_model <- ets(training\_data)

ets\_forecast <- forecast(ets\_model, h = length(testing\_data))$mean

mlp\_model <- mlp(training\_data)

mlp\_forecast <- forecast(mlp\_model, h = length(testing\_data))$mean

train\_df <- data.frame(

ds = seq.Date(from = as.Date("1980-01-01"), by = "year", length.out = length(training\_data)),

y = as.numeric(training\_data))

prophet\_model <- prophet(train\_df, yearly.seasonality = FALSE, weekly.seasonality = FALSE, daily.seasonality = FALSE)

future <- data.frame(ds = seq.Date(from = max(train\_df$ds) + 1, by = "year", length.out = length(testing\_data)))

prophet\_forecast <- predict(prophet\_model, future)$yhat

lagged\_data <- data.frame(

Target = training\_data[-1],

Lag1 = training\_data[-length(training\_data)])

xgb\_model <- xgboost(data = as.matrix(lagged\_data$Lag1), label = lagged\_data$Target, nrounds = 100)

xgb\_forecast <- numeric(length(testing\_data))

last\_value <- training\_data[length(training\_data)]

for (i in 1:length(testing\_data)) {

xgb\_forecast[i] <- predict(xgb\_model, newdata = as.matrix(last\_value))

last\_value <- xgb\_forecast[i]}

hybrid\_forecast\_function <- function(weights, f1, f2) {

weights[1] \* as.numeric(f1) + weights[2] \* as.numeric(f2)}

fitness\_function <- function(weights, f1, f2, actual) {

hybrid\_forecast <- hybrid\_forecast\_function(weights, f1, f2)

sqrt(mean((actual - hybrid\_forecast)^2))}

optimize\_weights <- function(f1, f2, actual) {

ga\_model <- ga(

type = "real-valued",

fitness = function(w) -fitness\_function(w, f1, f2, actual),

lower = c(0, 0),

upper = c(1, 1),

popSize = 50,

maxiter = 100)

return(ga\_model@solution)}

weights <- optimize\_weights(arima\_forecast, ets\_forecast, testing\_data)

arima\_ets\_forecast <- hybrid\_forecast\_function(weights, arima\_forecast, ets\_forecast)

weights <- optimize\_weights(arima\_forecast, mlp\_forecast, testing\_data)

arima\_mlp\_forecast <- hybrid\_forecast\_function(weights, arima\_forecast, mlp\_forecast)

weights <- optimize\_weights(arima\_forecast, xgb\_forecast, testing\_data)

arima\_xgb\_forecast <- hybrid\_forecast\_function(weights, arima\_forecast, xgb\_forecast)

weights <- optimize\_weights(arima\_forecast, prophet\_forecast, testing\_data)

arima\_prophet\_forecast <- hybrid\_forecast\_function(weights, arima\_forecast, prophet\_forecast)

calculate\_metrics <- function(actual, forecast) {

rmse <- sqrt(mean((actual - forecast)^2))

mape <- mean(abs((actual - forecast) / actual)) \* 100

return(c(RMSE = rmse, MAPE = mape))}

model\_metrics <- data.frame(

Model = c("ARIMA", "ETS", "MLP", "Prophet", "XGBoost", "ARIMA\_ETS", "ARIMA\_MLP", "ARIMA\_XGBoost", "ARIMA\_Prophet"),

do.call(rbind, list(

calculate\_metrics(testing\_data, arima\_forecast),

calculate\_metrics(testing\_data, ets\_forecast),

calculate\_metrics(testing\_data, mlp\_forecast),

calculate\_metrics(testing\_data, prophet\_forecast),

calculate\_metrics(testing\_data, xgb\_forecast),

calculate\_metrics(testing\_data, arima\_ets\_forecast),

calculate\_metrics(testing\_data, arima\_mlp\_forecast),

calculate\_metrics(testing\_data, arima\_xgb\_forecast),

calculate\_metrics(testing\_data, arima\_prophet\_forecast))))

print("Model Performance Metrics (RMSE and MAE):")

print(model\_metrics)

all\_forecasts <- data.frame(

Year = seq(2018, 2022),

Actual = as.numeric(testing\_data),

ARIMA = as.numeric(arima\_forecast),

ETS = as.numeric(ets\_forecast),

MLP = as.numeric(mlp\_forecast),

Prophet = as.numeric(prophet\_forecast),

XGBoost = as.numeric(xgb\_forecast),

ARIMA\_ETS = as.numeric(arima\_ets\_forecast),

ARIMA\_MLP = as.numeric(arima\_mlp\_forecast),

ARIMA\_Prophet = as.numeric(arima\_prophet\_forecast),

ARIMA\_XGBoost = as.numeric(arima\_xgb\_forecast))

melted\_forecasts <- melt(all\_forecasts, id.vars = "Year", variable.name = "Model", value.name = "Value")

ggplot(data = melted\_forecasts, aes(x = Year, y = Value, color = Model)) +

geom\_line() +

labs(title = "MRT", y = "Mortality Rate", x = "Year") +

theme\_minimal()

**# Out-of-Sample Forecast for CPV UN-5 Male Mortality Rates**

set.seed(123)

CPV = c(86.06, 84.55, 83.59, 82.68, 81.32, 79.32, 76.47, 72.82, 68.97, 65.45, 62.63, 60.73, 59.71, 59.1, 58.28, 56.7, 54.03, 50.18, 45.69, 41.2, 37.14, 33.89, 31.6, 30.2, 29.52, 29.35, 29.37, 29.38, 29.18, 28.57, 27.6, 26.37, 24.9, 23.33, 21.67, 20.08, 18.61, 17.35, 16.25, 15.33, 14.58, 13.94, 13.39)

cpv\_ts <- ts(CPV, start = 1980, frequency = 1)

training\_data <- window(cpv\_ts, start=1980, end=2017)

testing\_data <- window(cpv\_ts, start=2018, end=2022)

arima\_model <- auto.arima(training\_data)

arima\_forecast <- forecast(arima\_model, h = length(testing\_data))$mean

ets\_model <- ets(training\_data)

ets\_forecast <- forecast(ets\_model, h = length(testing\_data))$mean

mlp\_model <- mlp(training\_data)

mlp\_forecast <- forecast(mlp\_model, h = length(testing\_data))$mean

train\_df <- data.frame(

ds = seq.Date(from = as.Date("1980-01-01"), by = "year", length.out = length(training\_data)),

y = as.numeric(training\_data))

prophet\_model <- prophet(train\_df, yearly.seasonality = FALSE, weekly.seasonality = FALSE, daily.seasonality = FALSE)

future <- data.frame(ds = seq.Date(from = max(train\_df$ds) + 1, by = "year", length.out = length(testing\_data)))

prophet\_forecast <- predict(prophet\_model, future)$yhat

lagged\_data <- data.frame(

Target = training\_data[-1],

Lag1 = training\_data[-length(training\_data)])

xgb\_model <- xgboost(data = as.matrix(lagged\_data$Lag1), label = lagged\_data$Target, nrounds = 100)

xgb\_forecast <- numeric(length(testing\_data))

last\_value <- training\_data[length(training\_data)]

for (i in 1:length(testing\_data)) {

xgb\_forecast[i] <- predict(xgb\_model, newdata = as.matrix(last\_value))

last\_value <- xgb\_forecast[i]}

hybrid\_forecast\_function <- function(weights, f1, f2) {

weights[1] \* as.numeric(f1) + weights[2] \* as.numeric(f2)}

fitness\_function <- function(weights, f1, f2, actual) {

hybrid\_forecast <- hybrid\_forecast\_function(weights, f1, f2)

sqrt(mean((actual - hybrid\_forecast)^2))}

optimize\_weights <- function(f1, f2, actual) {

ga\_model <- ga(

type = "real-valued",

fitness = function(w) -fitness\_function(w, f1, f2, actual),

lower = c(0, 0),

upper = c(1, 1),

popSize = 50,

maxiter = 100)

return(ga\_model@solution)}

weights <- optimize\_weights(arima\_forecast, ets\_forecast, testing\_data)

arima\_ets\_forecast <- hybrid\_forecast\_function(weights, arima\_forecast, ets\_forecast)

weights <- optimize\_weights(arima\_forecast, mlp\_forecast, testing\_data)

arima\_mlp\_forecast <- hybrid\_forecast\_function(weights, arima\_forecast, mlp\_forecast)

weights <- optimize\_weights(arima\_forecast, xgb\_forecast, testing\_data)

arima\_xgb\_forecast <- hybrid\_forecast\_function(weights, arima\_forecast, xgb\_forecast)

weights <- optimize\_weights(arima\_forecast, prophet\_forecast, testing\_data)

arima\_prophet\_forecast <- hybrid\_forecast\_function(weights, arima\_forecast, prophet\_forecast)

calculate\_metrics <- function(actual, forecast) {

rmse <- sqrt(mean((actual - forecast)^2))

mape <- mean(abs((actual - forecast) / actual)) \* 100

return(c(RMSE = rmse, MAPE = mape))}

model\_metrics <- data.frame(

Model = c("ARIMA", "ETS", "MLP", "Prophet", "XGBoost", "ARIMA\_ETS", "ARIMA\_MLP", "ARIMA\_XGBoost", "ARIMA\_Prophet"),

do.call(rbind, list(

calculate\_metrics(testing\_data, arima\_forecast),

calculate\_metrics(testing\_data, ets\_forecast),

calculate\_metrics(testing\_data, mlp\_forecast),

calculate\_metrics(testing\_data, prophet\_forecast),

calculate\_metrics(testing\_data, xgb\_forecast),

calculate\_metrics(testing\_data, arima\_ets\_forecast),

calculate\_metrics(testing\_data, arima\_mlp\_forecast),

calculate\_metrics(testing\_data, arima\_xgb\_forecast),

calculate\_metrics(testing\_data, arima\_prophet\_forecast))))

print("Model Performance Metrics (RMSE and MAE):")

print(model\_metrics)

all\_forecasts <- data.frame(

Year = seq(2018, 2022),

Actual = as.numeric(testing\_data),

ARIMA = as.numeric(arima\_forecast),

ETS = as.numeric(ets\_forecast),

MLP = as.numeric(mlp\_forecast),

Prophet = as.numeric(prophet\_forecast),

XGBoost = as.numeric(xgb\_forecast),

ARIMA\_ETS = as.numeric(arima\_ets\_forecast),

ARIMA\_MLP = as.numeric(arima\_mlp\_forecast),

ARIMA\_Prophet = as.numeric(arima\_prophet\_forecast),

ARIMA\_XGBoost = as.numeric(arima\_xgb\_forecast))

melted\_forecasts <- melt(all\_forecasts, id.vars = "Year", variable.name = "Model", value.name = "Value")

ggplot(data = melted\_forecasts, aes(x = Year, y = Value, color = Model)) +

geom\_line() +

labs(title = "CPV", y = "Mortality Rate", x = "Year") +

theme\_minimal()

**# Out-of-Sample Forecast for TCD UN-5 Male Mortality Rates**

set.seed(123)

TCD = c(249.96, 247.91, 245.44, 242.78, 239.9, 236.96, 234.08, 231.21, 228.28, 224.99, 221.77, 218.58, 215.61, 212.64, 209.84, 207.23, 204.53, 202.01, 199.12, 196.02, 192.78, 189.26, 185.93, 182.59, 179, 175.35, 171.4, 167.61, 163.66, 159.59, 155.43, 151.39, 147.39, 143.58, 139.72, 135.94, 132.23, 128.24, 124.19, 120.39, 116.63, 112.88, 109.33)

tcd\_ts <- ts(TCD, start = 1980, frequency = 1)

training\_data <- window(tcd\_ts, start=1980, end=2017)

testing\_data <- window(tcd\_ts, start=2018, end=2022)

arima\_model <- auto.arima(training\_data)

arima\_forecast <- forecast(arima\_model, h = length(testing\_data))$mean

ets\_model <- ets(training\_data)

ets\_forecast <- forecast(ets\_model, h = length(testing\_data))$mean

mlp\_model <- mlp(training\_data)

mlp\_forecast <- forecast(mlp\_model, h = length(testing\_data))$mean

train\_df <- data.frame(

ds = seq.Date(from = as.Date("1980-01-01"), by = "year", length.out = length(training\_data)),

y = as.numeric(training\_data))

prophet\_model <- prophet(train\_df, yearly.seasonality = FALSE, weekly.seasonality = FALSE, daily.seasonality = FALSE)

future <- data.frame(ds = seq.Date(from = max(train\_df$ds) + 1, by = "year", length.out = length(testing\_data)))

prophet\_forecast <- predict(prophet\_model, future)$yhat

lagged\_data <- data.frame(

Target = training\_data[-1],

Lag1 = training\_data[-length(training\_data)])

xgb\_model <- xgboost(data = as.matrix(lagged\_data$Lag1), label = lagged\_data$Target, nrounds = 100)

xgb\_forecast <- numeric(length(testing\_data))

last\_value <- training\_data[length(training\_data)]

for (i in 1:length(testing\_data)) {

xgb\_forecast[i] <- predict(xgb\_model, newdata = as.matrix(last\_value))

last\_value <- xgb\_forecast[i]}

hybrid\_forecast\_function <- function(weights, f1, f2) {

weights[1] \* as.numeric(f1) + weights[2] \* as.numeric(f2)}

fitness\_function <- function(weights, f1, f2, actual) {

hybrid\_forecast <- hybrid\_forecast\_function(weights, f1, f2)

sqrt(mean((actual - hybrid\_forecast)^2))}

optimize\_weights <- function(f1, f2, actual) {

ga\_model <- ga(

type = "real-valued",

fitness = function(w) -fitness\_function(w, f1, f2, actual),

lower = c(0, 0),

upper = c(1, 1),

popSize = 50,

maxiter = 100)

return(ga\_model@solution)}

weights <- optimize\_weights(arima\_forecast, ets\_forecast, testing\_data)

arima\_ets\_forecast <- hybrid\_forecast\_function(weights, arima\_forecast, ets\_forecast)

weights <- optimize\_weights(arima\_forecast, mlp\_forecast, testing\_data)

arima\_mlp\_forecast <- hybrid\_forecast\_function(weights, arima\_forecast, mlp\_forecast)

weights <- optimize\_weights(arima\_forecast, xgb\_forecast, testing\_data)

arima\_xgb\_forecast <- hybrid\_forecast\_function(weights, arima\_forecast, xgb\_forecast)

weights <- optimize\_weights(arima\_forecast, prophet\_forecast, testing\_data)

arima\_prophet\_forecast <- hybrid\_forecast\_function(weights, arima\_forecast, prophet\_forecast)

calculate\_metrics <- function(actual, forecast) {

rmse <- sqrt(mean((actual - forecast)^2))

mape <- mean(abs((actual - forecast) / actual)) \* 100

return(c(RMSE = rmse, MAPE = mape))}

model\_metrics <- data.frame(

Model = c("ARIMA", "ETS", "MLP", "Prophet", "XGBoost", "ARIMA\_ETS", "ARIMA\_MLP", "ARIMA\_XGBoost", "ARIMA\_Prophet"),

do.call(rbind, list(

calculate\_metrics(testing\_data, arima\_forecast),

calculate\_metrics(testing\_data, ets\_forecast),

calculate\_metrics(testing\_data, mlp\_forecast),

calculate\_metrics(testing\_data, prophet\_forecast),

calculate\_metrics(testing\_data, xgb\_forecast),

calculate\_metrics(testing\_data, arima\_ets\_forecast),

calculate\_metrics(testing\_data, arima\_mlp\_forecast),

calculate\_metrics(testing\_data, arima\_xgb\_forecast),

calculate\_metrics(testing\_data, arima\_prophet\_forecast))))

print("Model Performance Metrics (RMSE and MAE):")

print(model\_metrics)

all\_forecasts <- data.frame(

Year = seq(2018, 2022),

Actual = as.numeric(testing\_data),

ARIMA = as.numeric(arima\_forecast),

ETS = as.numeric(ets\_forecast),

MLP = as.numeric(mlp\_forecast),

Prophet = as.numeric(prophet\_forecast),

XGBoost = as.numeric(xgb\_forecast),

ARIMA\_ETS = as.numeric(arima\_ets\_forecast),

ARIMA\_MLP = as.numeric(arima\_mlp\_forecast),

ARIMA\_Prophet = as.numeric(arima\_prophet\_forecast),

ARIMA\_XGBoost = as.numeric(arima\_xgb\_forecast))

melted\_forecasts <- melt(all\_forecasts, id.vars = "Year", variable.name = "Model", value.name = "Value")

ggplot(data = melted\_forecasts, aes(x = Year, y = Value, color = Model)) +

geom\_line() +

labs(title = "TCD", y = "Mortality Rate", x = "Year") +

theme\_minimal()

**# Out-of-Sample Forecast for CIV UN-5 Male Mortality Rates**

set.seed(123)

CIV = c(181.68, 177.17, 173.54, 170.72, 168.58, 166.68, 165.37, 164.74, 164.68, 164.99, 165.31, 165.47, 165.35, 165.32, 165, 164.22, 162.93, 161.05, 158.62, 155.67, 152.55, 149.01, 145.28, 141.51, 137.54, 133.48, 129.6, 125.59, 121.49, 116.87, 112.86, 109.34, 106, 102.43, 98.739, 95.379, 92.666, 90.033, 87.174, 84.164, 81.657, 79.277, 76.893)

civ\_ts <- ts(CIV, start = 1980, frequency = 1)

training\_data <- window(civ\_ts, start=1980, end=2017)

testing\_data <- window(civ\_ts, start=2018, end=2022)

arima\_model <- auto.arima(training\_data)

arima\_forecast <- forecast(arima\_model, h = length(testing\_data))$mean

ets\_model <- ets(training\_data)

ets\_forecast <- forecast(ets\_model, h = length(testing\_data))$mean

mlp\_model <- mlp(training\_data)

mlp\_forecast <- forecast(mlp\_model, h = length(testing\_data))$mean

train\_df <- data.frame(

ds = seq.Date(from = as.Date("1980-01-01"), by = "year", length.out = length(training\_data)),

y = as.numeric(training\_data))

prophet\_model <- prophet(train\_df, yearly.seasonality = FALSE, weekly.seasonality = FALSE, daily.seasonality = FALSE)

future <- data.frame(ds = seq.Date(from = max(train\_df$ds) + 1, by = "year", length.out = length(testing\_data)))

prophet\_forecast <- predict(prophet\_model, future)$yhat

lagged\_data <- data.frame(

Target = training\_data[-1],

Lag1 = training\_data[-length(training\_data)])

xgb\_model <- xgboost(data = as.matrix(lagged\_data$Lag1), label = lagged\_data$Target, nrounds = 100)

xgb\_forecast <- numeric(length(testing\_data))

last\_value <- training\_data[length(training\_data)]

for (i in 1:length(testing\_data)) {

xgb\_forecast[i] <- predict(xgb\_model, newdata = as.matrix(last\_value))

last\_value <- xgb\_forecast[i]}

hybrid\_forecast\_function <- function(weights, f1, f2) {

weights[1] \* as.numeric(f1) + weights[2] \* as.numeric(f2)}

fitness\_function <- function(weights, f1, f2, actual) {

hybrid\_forecast <- hybrid\_forecast\_function(weights, f1, f2)

sqrt(mean((actual - hybrid\_forecast)^2))}

optimize\_weights <- function(f1, f2, actual) {

ga\_model <- ga(

type = "real-valued",

fitness = function(w) -fitness\_function(w, f1, f2, actual),

lower = c(0, 0),

upper = c(1, 1),

popSize = 50,

maxiter = 100)

return(ga\_model@solution)}

weights <- optimize\_weights(arima\_forecast, ets\_forecast, testing\_data)

arima\_ets\_forecast <- hybrid\_forecast\_function(weights, arima\_forecast, ets\_forecast)

weights <- optimize\_weights(arima\_forecast, mlp\_forecast, testing\_data)

arima\_mlp\_forecast <- hybrid\_forecast\_function(weights, arima\_forecast, mlp\_forecast)

weights <- optimize\_weights(arima\_forecast, xgb\_forecast, testing\_data)

arima\_xgb\_forecast <- hybrid\_forecast\_function(weights, arima\_forecast, xgb\_forecast)

weights <- optimize\_weights(arima\_forecast, prophet\_forecast, testing\_data)

arima\_prophet\_forecast <- hybrid\_forecast\_function(weights, arima\_forecast, prophet\_forecast)

calculate\_metrics <- function(actual, forecast) {

rmse <- sqrt(mean((actual - forecast)^2))

mape <- mean(abs((actual - forecast) / actual)) \* 100

return(c(RMSE = rmse, MAPE = mape))}

model\_metrics <- data.frame(

Model = c("ARIMA", "ETS", "MLP", "Prophet", "XGBoost", "ARIMA\_ETS", "ARIMA\_MLP", "ARIMA\_XGBoost", "ARIMA\_Prophet"),

do.call(rbind, list(

calculate\_metrics(testing\_data, arima\_forecast),

calculate\_metrics(testing\_data, ets\_forecast),

calculate\_metrics(testing\_data, mlp\_forecast),

calculate\_metrics(testing\_data, prophet\_forecast),

calculate\_metrics(testing\_data, xgb\_forecast),

calculate\_metrics(testing\_data, arima\_ets\_forecast),

calculate\_metrics(testing\_data, arima\_mlp\_forecast),

calculate\_metrics(testing\_data, arima\_xgb\_forecast),

calculate\_metrics(testing\_data, arima\_prophet\_forecast))))

print("Model Performance Metrics (RMSE and MAE):")

print(model\_metrics)

all\_forecasts <- data.frame(

Year = seq(2018, 2022),

Actual = as.numeric(testing\_data),

ARIMA = as.numeric(arima\_forecast),

ETS = as.numeric(ets\_forecast),

MLP = as.numeric(mlp\_forecast),

Prophet = as.numeric(prophet\_forecast),

XGBoost = as.numeric(xgb\_forecast),

ARIMA\_ETS = as.numeric(arima\_ets\_forecast),

ARIMA\_MLP = as.numeric(arima\_mlp\_forecast),

ARIMA\_Prophet = as.numeric(arima\_prophet\_forecast),

ARIMA\_XGBoost = as.numeric(arima\_xgb\_forecast))

melted\_forecasts <- melt(all\_forecasts, id.vars = "Year", variable.name = "Model", value.name = "Value")

ggplot(data = melted\_forecasts, aes(x = Year, y = Value, color = Model)) +

geom\_line() +

labs(title = "CIV", y = "Mortality Rate", x = "Year") +

theme\_minimal()

**# Out-of-Sample Forecast for GNQ UN-5 Male Mortality Rates**

set.seed(123)

GNQ = c(210.40, 210.4, 210.4, 206.2, 201.85, 198.09, 195.07, 192.62, 190.47, 188.77, 187.05, 185.44, 183.9, 182.51, 181.08, 179.5, 177.39, 174.89, 171.44, 167.91, 163.84, 159.69, 155.21, 150.74, 146.19, 141.4, 136.61, 131.77, 127.04, 122.63, 118.57, 114.55, 110.62, 106.86, 103.37, 99.931, 96.594, 93.446, 90.2, 87.159, 84.438, 81.857, 79.273)

gnq\_ts <- ts(GNQ, start = 1980, frequency = 1)

training\_data <- window(gnq\_ts, start=1980, end=2017)

testing\_data <- window(gnq\_ts, start=2018, end=2022)

arima\_model <- auto.arima(training\_data)

arima\_forecast <- forecast(arima\_model, h = length(testing\_data))$mean

ets\_model <- ets(training\_data)

ets\_forecast <- forecast(ets\_model, h = length(testing\_data))$mean

mlp\_model <- mlp(training\_data)

mlp\_forecast <- forecast(mlp\_model, h = length(testing\_data))$mean

train\_df <- data.frame(

ds = seq.Date(from = as.Date("1980-01-01"), by = "year", length.out = length(training\_data)),

y = as.numeric(training\_data))

prophet\_model <- prophet(train\_df, yearly.seasonality = FALSE, weekly.seasonality = FALSE, daily.seasonality = FALSE)

future <- data.frame(ds = seq.Date(from = max(train\_df$ds) + 1, by = "year", length.out = length(testing\_data)))

prophet\_forecast <- predict(prophet\_model, future)$yhat

lagged\_data <- data.frame(

Target = training\_data[-1],

Lag1 = training\_data[-length(training\_data)])

xgb\_model <- xgboost(data = as.matrix(lagged\_data$Lag1), label = lagged\_data$Target, nrounds = 100)

xgb\_forecast <- numeric(length(testing\_data))

last\_value <- training\_data[length(training\_data)]

for (i in 1:length(testing\_data)) {

xgb\_forecast[i] <- predict(xgb\_model, newdata = as.matrix(last\_value))

last\_value <- xgb\_forecast[i]}

hybrid\_forecast\_function <- function(weights, f1, f2) {

weights[1] \* as.numeric(f1) + weights[2] \* as.numeric(f2)}

fitness\_function <- function(weights, f1, f2, actual) {

hybrid\_forecast <- hybrid\_forecast\_function(weights, f1, f2)

sqrt(mean((actual - hybrid\_forecast)^2))}

optimize\_weights <- function(f1, f2, actual) {

ga\_model <- ga(

type = "real-valued",

fitness = function(w) -fitness\_function(w, f1, f2, actual),

lower = c(0, 0),

upper = c(1, 1),

popSize = 50,

maxiter = 100)

return(ga\_model@solution)}

weights <- optimize\_weights(arima\_forecast, ets\_forecast, testing\_data)

arima\_ets\_forecast <- hybrid\_forecast\_function(weights, arima\_forecast, ets\_forecast)

weights <- optimize\_weights(arima\_forecast, mlp\_forecast, testing\_data)

arima\_mlp\_forecast <- hybrid\_forecast\_function(weights, arima\_forecast, mlp\_forecast)

weights <- optimize\_weights(arima\_forecast, xgb\_forecast, testing\_data)

arima\_xgb\_forecast <- hybrid\_forecast\_function(weights, arima\_forecast, xgb\_forecast)

weights <- optimize\_weights(arima\_forecast, prophet\_forecast, testing\_data)

arima\_prophet\_forecast <- hybrid\_forecast\_function(weights, arima\_forecast, prophet\_forecast)

calculate\_metrics <- function(actual, forecast) {

rmse <- sqrt(mean((actual - forecast)^2))

mape <- mean(abs((actual - forecast) / actual)) \* 100

return(c(RMSE = rmse, MAPE = mape))}

model\_metrics <- data.frame(

Model = c("ARIMA", "ETS", "MLP", "Prophet", "XGBoost", "ARIMA\_ETS", "ARIMA\_MLP", "ARIMA\_XGBoost", "ARIMA\_Prophet"),

do.call(rbind, list(

calculate\_metrics(testing\_data, arima\_forecast),

calculate\_metrics(testing\_data, ets\_forecast),

calculate\_metrics(testing\_data, mlp\_forecast),

calculate\_metrics(testing\_data, prophet\_forecast),

calculate\_metrics(testing\_data, xgb\_forecast),

calculate\_metrics(testing\_data, arima\_ets\_forecast),

calculate\_metrics(testing\_data, arima\_mlp\_forecast),

calculate\_metrics(testing\_data, arima\_xgb\_forecast),

calculate\_metrics(testing\_data, arima\_prophet\_forecast))))

print("Model Performance Metrics (RMSE and MAE):")

print(model\_metrics)

all\_forecasts <- data.frame(

Year = seq(2018, 2022),

Actual = as.numeric(testing\_data),

ARIMA = as.numeric(arima\_forecast),

ETS = as.numeric(ets\_forecast),

MLP = as.numeric(mlp\_forecast),

Prophet = as.numeric(prophet\_forecast),

XGBoost = as.numeric(xgb\_forecast),

ARIMA\_ETS = as.numeric(arima\_ets\_forecast),

ARIMA\_MLP = as.numeric(arima\_mlp\_forecast),

ARIMA\_Prophet = as.numeric(arima\_prophet\_forecast),

ARIMA\_XGBoost = as.numeric(arima\_xgb\_forecast))

melted\_forecasts <- melt(all\_forecasts, id.vars = "Year", variable.name = "Model", value.name = "Value")

ggplot(data = melted\_forecasts, aes(x = Year, y = Value, color = Model)) +

geom\_line() +

labs(title = "GNQ", y = "Mortality Rate", x = "Year") +

theme\_minimal()

**# Out-of-Sample Forecast for GMB UN-5 Male Mortality Rates**

set.seed(123)

GMB = c(240.36, 233.75, 227.15, 220.51, 213.83, 207.16, 200.42, 194.03, 187.83, 181.77, 175.79, 169.85, 163.82, 157.85, 151.91, 146.3, 140.8, 135.37, 130.17, 125.16, 120.2, 115.39, 110.75, 106.26, 101.89, 97.669, 93.579, 89.685, 86.001, 82.434, 79.091, 75.963, 72.905, 70.031, 67.323, 64.698, 62.246, 59.855, 57.645, 55.583, 53.628, 51.819, 50.137)

gmb\_ts <- ts(GMB, start = 1980, frequency = 1)

training\_data <- window(gmb\_ts, start=1980, end=2017)

testing\_data <- window(gmb\_ts, start=2018, end=2022)

arima\_model <- auto.arima(training\_data)

arima\_forecast <- forecast(arima\_model, h = length(testing\_data))$mean

ets\_model <- ets(training\_data)

ets\_forecast <- forecast(ets\_model, h = length(testing\_data))$mean

mlp\_model <- mlp(training\_data)

mlp\_forecast <- forecast(mlp\_model, h = length(testing\_data))$mean

train\_df <- data.frame(

ds = seq.Date(from = as.Date("1980-01-01"), by = "year", length.out = length(training\_data)),

y = as.numeric(training\_data))

prophet\_model <- prophet(train\_df, yearly.seasonality = FALSE, weekly.seasonality = FALSE, daily.seasonality = FALSE)

future <- data.frame(ds = seq.Date(from = max(train\_df$ds) + 1, by = "year", length.out = length(testing\_data)))

prophet\_forecast <- predict(prophet\_model, future)$yhat

lagged\_data <- data.frame(

Target = training\_data[-1],

Lag1 = training\_data[-length(training\_data)])

xgb\_model <- xgboost(data = as.matrix(lagged\_data$Lag1), label = lagged\_data$Target, nrounds = 100)

xgb\_forecast <- numeric(length(testing\_data))

last\_value <- training\_data[length(training\_data)]

for (i in 1:length(testing\_data)) {

xgb\_forecast[i] <- predict(xgb\_model, newdata = as.matrix(last\_value))

last\_value <- xgb\_forecast[i]}

hybrid\_forecast\_function <- function(weights, f1, f2) {

weights[1] \* as.numeric(f1) + weights[2] \* as.numeric(f2)}

fitness\_function <- function(weights, f1, f2, actual) {

hybrid\_forecast <- hybrid\_forecast\_function(weights, f1, f2)

sqrt(mean((actual - hybrid\_forecast)^2))}

optimize\_weights <- function(f1, f2, actual) {

ga\_model <- ga(

type = "real-valued",

fitness = function(w) -fitness\_function(w, f1, f2, actual),

lower = c(0, 0),

upper = c(1, 1),

popSize = 50,

maxiter = 100)

return(ga\_model@solution)}

weights <- optimize\_weights(arima\_forecast, ets\_forecast, testing\_data)

arima\_ets\_forecast <- hybrid\_forecast\_function(weights, arima\_forecast, ets\_forecast)

weights <- optimize\_weights(arima\_forecast, mlp\_forecast, testing\_data)

arima\_mlp\_forecast <- hybrid\_forecast\_function(weights, arima\_forecast, mlp\_forecast)

weights <- optimize\_weights(arima\_forecast, xgb\_forecast, testing\_data)

arima\_xgb\_forecast <- hybrid\_forecast\_function(weights, arima\_forecast, xgb\_forecast)

weights <- optimize\_weights(arima\_forecast, prophet\_forecast, testing\_data)

arima\_prophet\_forecast <- hybrid\_forecast\_function(weights, arima\_forecast, prophet\_forecast)

calculate\_metrics <- function(actual, forecast) {

rmse <- sqrt(mean((actual - forecast)^2))

mape <- mean(abs((actual - forecast) / actual)) \* 100

return(c(RMSE = rmse, MAPE = mape))}

model\_metrics <- data.frame(

Model = c("ARIMA", "ETS", "MLP", "Prophet", "XGBoost", "ARIMA\_ETS", "ARIMA\_MLP", "ARIMA\_XGBoost", "ARIMA\_Prophet"),

do.call(rbind, list(

calculate\_metrics(testing\_data, arima\_forecast),

calculate\_metrics(testing\_data, ets\_forecast),

calculate\_metrics(testing\_data, mlp\_forecast),

calculate\_metrics(testing\_data, prophet\_forecast),

calculate\_metrics(testing\_data, xgb\_forecast),

calculate\_metrics(testing\_data, arima\_ets\_forecast),

calculate\_metrics(testing\_data, arima\_mlp\_forecast),

calculate\_metrics(testing\_data, arima\_xgb\_forecast),

calculate\_metrics(testing\_data, arima\_prophet\_forecast))))

print("Model Performance Metrics (RMSE and MAE):")

print(model\_metrics)

all\_forecasts <- data.frame(

Year = seq(2018, 2022),

Actual = as.numeric(testing\_data),

ARIMA = as.numeric(arima\_forecast),

ETS = as.numeric(ets\_forecast),

MLP = as.numeric(mlp\_forecast),

Prophet = as.numeric(prophet\_forecast),

XGBoost = as.numeric(xgb\_forecast),

ARIMA\_ETS = as.numeric(arima\_ets\_forecast),

ARIMA\_MLP = as.numeric(arima\_mlp\_forecast),

ARIMA\_Prophet = as.numeric(arima\_prophet\_forecast),

ARIMA\_XGBoost = as.numeric(arima\_xgb\_forecast))

melted\_forecasts <- melt(all\_forecasts, id.vars = "Year", variable.name = "Model", value.name = "Value")

ggplot(data = melted\_forecasts, aes(x = Year, y = Value, color = Model)) +

geom\_line() +

labs(title = "GMB", y = "Mortality Rate", x = "Year") +

theme\_minimal()

**# Out-of-Sample Forecast for GHA UN-5 Male Mortality Rates**

set.seed(123)

GHA = c(175.08, 173.71, 172.37, 170.44, 167.51, 163.42, 158.24, 152.45, 146.45, 140.61, 135.5, 131.35, 128.09, 125.85, 124.09, 122.43, 120.47, 118.1, 115.07, 111.55, 107.61, 103.46, 99.584, 96.236, 93.358, 90.743, 88.311, 85.723, 82.855, 79.624, 76.175, 72.679, 69.287, 66.103, 63.205, 60.61, 58.142, 55.832, 53.737, 51.854, 49.954, 48.205, 46.67)

gha\_ts <- ts(GHA, start = 1980, frequency = 1)

training\_data <- window(gha\_ts, start=1980, end=2017)

testing\_data <- window(gha\_ts, start=2018, end=2022)

arima\_model <- auto.arima(training\_data)

arima\_forecast <- forecast(arima\_model, h = length(testing\_data))$mean

ets\_model <- ets(training\_data)

ets\_forecast <- forecast(ets\_model, h = length(testing\_data))$mean

mlp\_model <- mlp(training\_data)

mlp\_forecast <- forecast(mlp\_model, h = length(testing\_data))$mean

train\_df <- data.frame(

ds = seq.Date(from = as.Date("1980-01-01"), by = "year", length.out = length(training\_data)),

y = as.numeric(training\_data))

prophet\_model <- prophet(train\_df, yearly.seasonality = FALSE, weekly.seasonality = FALSE, daily.seasonality = FALSE)

future <- data.frame(ds = seq.Date(from = max(train\_df$ds) + 1, by = "year", length.out = length(testing\_data)))

prophet\_forecast <- predict(prophet\_model, future)$yhat

lagged\_data <- data.frame(

Target = training\_data[-1],

Lag1 = training\_data[-length(training\_data)])

xgb\_model <- xgboost(data = as.matrix(lagged\_data$Lag1), label = lagged\_data$Target, nrounds = 100)

xgb\_forecast <- numeric(length(testing\_data))

last\_value <- training\_data[length(training\_data)]

for (i in 1:length(testing\_data)) {

xgb\_forecast[i] <- predict(xgb\_model, newdata = as.matrix(last\_value))

last\_value <- xgb\_forecast[i]}

hybrid\_forecast\_function <- function(weights, f1, f2) {

weights[1] \* as.numeric(f1) + weights[2] \* as.numeric(f2)}

fitness\_function <- function(weights, f1, f2, actual) {

hybrid\_forecast <- hybrid\_forecast\_function(weights, f1, f2)

sqrt(mean((actual - hybrid\_forecast)^2))}

optimize\_weights <- function(f1, f2, actual) {

ga\_model <- ga(

type = "real-valued",

fitness = function(w) -fitness\_function(w, f1, f2, actual),

lower = c(0, 0),

upper = c(1, 1),

popSize = 50,

maxiter = 100)

return(ga\_model@solution)}

weights <- optimize\_weights(arima\_forecast, ets\_forecast, testing\_data)

arima\_ets\_forecast <- hybrid\_forecast\_function(weights, arima\_forecast, ets\_forecast)

weights <- optimize\_weights(arima\_forecast, mlp\_forecast, testing\_data)

arima\_mlp\_forecast <- hybrid\_forecast\_function(weights, arima\_forecast, mlp\_forecast)

weights <- optimize\_weights(arima\_forecast, xgb\_forecast, testing\_data)

arima\_xgb\_forecast <- hybrid\_forecast\_function(weights, arima\_forecast, xgb\_forecast)

weights <- optimize\_weights(arima\_forecast, prophet\_forecast, testing\_data)

arima\_prophet\_forecast <- hybrid\_forecast\_function(weights, arima\_forecast, prophet\_forecast)

calculate\_metrics <- function(actual, forecast) {

rmse <- sqrt(mean((actual - forecast)^2))

mape <- mean(abs((actual - forecast) / actual)) \* 100

return(c(RMSE = rmse, MAPE = mape))}

model\_metrics <- data.frame(

Model = c("ARIMA", "ETS", "MLP", "Prophet", "XGBoost", "ARIMA\_ETS", "ARIMA\_MLP", "ARIMA\_XGBoost", "ARIMA\_Prophet"),

do.call(rbind, list(

calculate\_metrics(testing\_data, arima\_forecast),

calculate\_metrics(testing\_data, ets\_forecast),

calculate\_metrics(testing\_data, mlp\_forecast),

calculate\_metrics(testing\_data, prophet\_forecast),

calculate\_metrics(testing\_data, xgb\_forecast),

calculate\_metrics(testing\_data, arima\_ets\_forecast),

calculate\_metrics(testing\_data, arima\_mlp\_forecast),

calculate\_metrics(testing\_data, arima\_xgb\_forecast),

calculate\_metrics(testing\_data, arima\_prophet\_forecast))))

print("Model Performance Metrics (RMSE and MAE):")

print(model\_metrics)

all\_forecasts <- data.frame(

Year = seq(2018, 2022),

Actual = as.numeric(testing\_data),

ARIMA = as.numeric(arima\_forecast),

ETS = as.numeric(ets\_forecast),

MLP = as.numeric(mlp\_forecast),

Prophet = as.numeric(prophet\_forecast),

XGBoost = as.numeric(xgb\_forecast),

ARIMA\_ETS = as.numeric(arima\_ets\_forecast),

ARIMA\_MLP = as.numeric(arima\_mlp\_forecast),

ARIMA\_Prophet = as.numeric(arima\_prophet\_forecast),

ARIMA\_XGBoost = as.numeric(arima\_xgb\_forecast))

melted\_forecasts <- melt(all\_forecasts, id.vars = "Year", variable.name = "Model", value.name = "Value")

ggplot(data = melted\_forecasts, aes(x = Year, y = Value, color = Model)) +

geom\_line() +

labs(title = "GHA", y = "Mortality Rate", x = "Year") +

theme\_minimal()

**# Out-of-Sample Forecast for NER UN-5 Male Mortality Rates**

set.seed(123)

NER = c(324.44, 324.26, 326.35, 330.26, 334.87, 339.29, 342.28, 343.1, 341.85, 338.55, 333.72, 327.93, 319.82, 309.77, 297.67, 284.64, 272.09, 260.7, 250.34, 240.77, 231.36, 221.43, 210.15, 198.28, 186.55, 175.33, 165.25, 156.58, 149.47, 143.91, 139.66, 136.56, 134.29, 132.54, 131.04, 129.71, 128.7, 127.5, 126.42, 125.42, 124.27, 122.71, 120.54)

ner\_ts <- ts(NER, start = 1980, frequency = 1)

training\_data <- window(ner\_ts, start=1980, end=2017)

testing\_data <- window(ner\_ts, start=2018, end=2022)

arima\_model <- auto.arima(training\_data)

arima\_forecast <- forecast(arima\_model, h = length(testing\_data))$mean

ets\_model <- ets(training\_data)

ets\_forecast <- forecast(ets\_model, h = length(testing\_data))$mean

mlp\_model <- mlp(training\_data)

mlp\_forecast <- forecast(mlp\_model, h = length(testing\_data))$mean

train\_df <- data.frame(

ds = seq.Date(from = as.Date("1980-01-01"), by = "year", length.out = length(training\_data)),

y = as.numeric(training\_data))

prophet\_model <- prophet(train\_df, yearly.seasonality = FALSE, weekly.seasonality = FALSE, daily.seasonality = FALSE)

future <- data.frame(ds = seq.Date(from = max(train\_df$ds) + 1, by = "year", length.out = length(testing\_data)))

prophet\_forecast <- predict(prophet\_model, future)$yhat

lagged\_data <- data.frame(

Target = training\_data[-1],

Lag1 = training\_data[-length(training\_data)])

xgb\_model <- xgboost(data = as.matrix(lagged\_data$Lag1), label = lagged\_data$Target, nrounds = 100)

xgb\_forecast <- numeric(length(testing\_data))

last\_value <- training\_data[length(training\_data)]

for (i in 1:length(testing\_data)) {

xgb\_forecast[i] <- predict(xgb\_model, newdata = as.matrix(last\_value))

last\_value <- xgb\_forecast[i]}

hybrid\_forecast\_function <- function(weights, f1, f2) {

weights[1] \* as.numeric(f1) + weights[2] \* as.numeric(f2)}

fitness\_function <- function(weights, f1, f2, actual) {

hybrid\_forecast <- hybrid\_forecast\_function(weights, f1, f2)

sqrt(mean((actual - hybrid\_forecast)^2))}

optimize\_weights <- function(f1, f2, actual) {

ga\_model <- ga(

type = "real-valued",

fitness = function(w) -fitness\_function(w, f1, f2, actual),

lower = c(0, 0),

upper = c(1, 1),

popSize = 50,

maxiter = 100)

return(ga\_model@solution)}

weights <- optimize\_weights(arima\_forecast, ets\_forecast, testing\_data)

arima\_ets\_forecast <- hybrid\_forecast\_function(weights, arima\_forecast, ets\_forecast)

weights <- optimize\_weights(arima\_forecast, mlp\_forecast, testing\_data)

arima\_mlp\_forecast <- hybrid\_forecast\_function(weights, arima\_forecast, mlp\_forecast)

weights <- optimize\_weights(arima\_forecast, xgb\_forecast, testing\_data)

arima\_xgb\_forecast <- hybrid\_forecast\_function(weights, arima\_forecast, xgb\_forecast)

weights <- optimize\_weights(arima\_forecast, prophet\_forecast, testing\_data)

arima\_prophet\_forecast <- hybrid\_forecast\_function(weights, arima\_forecast, prophet\_forecast)

calculate\_metrics <- function(actual, forecast) {

rmse <- sqrt(mean((actual - forecast)^2))

mape <- mean(abs((actual - forecast) / actual)) \* 100

return(c(RMSE = rmse, MAPE = mape))}

model\_metrics <- data.frame(

Model = c("ARIMA", "ETS", "MLP", "Prophet", "XGBoost", "ARIMA\_ETS", "ARIMA\_MLP", "ARIMA\_XGBoost", "ARIMA\_Prophet"),

do.call(rbind, list(

calculate\_metrics(testing\_data, arima\_forecast),

calculate\_metrics(testing\_data, ets\_forecast),

calculate\_metrics(testing\_data, mlp\_forecast),

calculate\_metrics(testing\_data, prophet\_forecast),

calculate\_metrics(testing\_data, xgb\_forecast),

calculate\_metrics(testing\_data, arima\_ets\_forecast),

calculate\_metrics(testing\_data, arima\_mlp\_forecast),

calculate\_metrics(testing\_data, arima\_xgb\_forecast),

calculate\_metrics(testing\_data, arima\_prophet\_forecast))))

print("Model Performance Metrics (RMSE and MAE):")

print(model\_metrics)

all\_forecasts <- data.frame(

Year = seq(2018, 2022),

Actual = as.numeric(testing\_data),

ARIMA = as.numeric(arima\_forecast),

ETS = as.numeric(ets\_forecast),

MLP = as.numeric(mlp\_forecast),

Prophet = as.numeric(prophet\_forecast),

XGBoost = as.numeric(xgb\_forecast),

ARIMA\_ETS = as.numeric(arima\_ets\_forecast),

ARIMA\_MLP = as.numeric(arima\_mlp\_forecast),

ARIMA\_Prophet = as.numeric(arima\_prophet\_forecast),

ARIMA\_XGBoost = as.numeric(arima\_xgb\_forecast))

melted\_forecasts <- melt(all\_forecasts, id.vars = "Year", variable.name = "Model", value.name = "Value")

ggplot(data = melted\_forecasts, aes(x = Year, y = Value, color = Model)) +

geom\_line() +

labs(title = "NER", y = "Mortality Rate", x = "Year") +

theme\_minimal()

**# Out-of-Sample Forecast for NGA UN-5 Male Mortality Rates**

set.seed(123)

NGA = c(223.14, 219.95, 217.96, 217.03, 217.09, 217.75, 218.78, 219.94, 220.46, 220.63, 220.41, 219.69, 218.6, 217.13, 215.43, 213.24, 210.34, 206.43, 202.09, 196.74, 190.96, 185.39, 179.54, 173.98, 168.45, 162.98, 158.07, 153.31, 149.1, 145.54, 142.56, 140.04, 137.88, 136.07, 134.46, 132.85, 131.03, 128.7, 126.25, 123.14, 120.11, 116.75, 113.24)

nga\_ts <- ts(NGA, start = 1980, frequency = 1)

training\_data <- window(nga\_ts, start=1980, end=2017)

testing\_data <- window(nga\_ts, start=2018, end=2022)

arima\_model <- auto.arima(training\_data)

arima\_forecast <- forecast(arima\_model, h = length(testing\_data))$mean

ets\_model <- ets(training\_data)

ets\_forecast <- forecast(ets\_model, h = length(testing\_data))$mean

mlp\_model <- mlp(training\_data)

mlp\_forecast <- forecast(mlp\_model, h = length(testing\_data))$mean

train\_df <- data.frame(

ds = seq.Date(from = as.Date("1980-01-01"), by = "year", length.out = length(training\_data)),

y = as.numeric(training\_data))

prophet\_model <- prophet(train\_df, yearly.seasonality = FALSE, weekly.seasonality = FALSE, daily.seasonality = FALSE)

future <- data.frame(ds = seq.Date(from = max(train\_df$ds) + 1, by = "year", length.out = length(testing\_data)))

prophet\_forecast <- predict(prophet\_model, future)$yhat

lagged\_data <- data.frame(

Target = training\_data[-1],

Lag1 = training\_data[-length(training\_data)])

xgb\_model <- xgboost(data = as.matrix(lagged\_data$Lag1), label = lagged\_data$Target, nrounds = 100)

xgb\_forecast <- numeric(length(testing\_data))

last\_value <- training\_data[length(training\_data)]

for (i in 1:length(testing\_data)) {

xgb\_forecast[i] <- predict(xgb\_model, newdata = as.matrix(last\_value))

last\_value <- xgb\_forecast[i]}

hybrid\_forecast\_function <- function(weights, f1, f2) {

weights[1] \* as.numeric(f1) + weights[2] \* as.numeric(f2)}

fitness\_function <- function(weights, f1, f2, actual) {

hybrid\_forecast <- hybrid\_forecast\_function(weights, f1, f2)

sqrt(mean((actual - hybrid\_forecast)^2))}

optimize\_weights <- function(f1, f2, actual) {

ga\_model <- ga(

type = "real-valued",

fitness = function(w) -fitness\_function(w, f1, f2, actual),

lower = c(0, 0),

upper = c(1, 1),

popSize = 50,

maxiter = 100)

return(ga\_model@solution)}

weights <- optimize\_weights(arima\_forecast, ets\_forecast, testing\_data)

arima\_ets\_forecast <- hybrid\_forecast\_function(weights, arima\_forecast, ets\_forecast)

weights <- optimize\_weights(arima\_forecast, mlp\_forecast, testing\_data)

arima\_mlp\_forecast <- hybrid\_forecast\_function(weights, arima\_forecast, mlp\_forecast)

weights <- optimize\_weights(arima\_forecast, xgb\_forecast, testing\_data)

arima\_xgb\_forecast <- hybrid\_forecast\_function(weights, arima\_forecast, xgb\_forecast)

weights <- optimize\_weights(arima\_forecast, prophet\_forecast, testing\_data)

arima\_prophet\_forecast <- hybrid\_forecast\_function(weights, arima\_forecast, prophet\_forecast)

calculate\_metrics <- function(actual, forecast) {

rmse <- sqrt(mean((actual - forecast)^2))

mape <- mean(abs((actual - forecast) / actual)) \* 100

return(c(RMSE = rmse, MAPE = mape))}

model\_metrics <- data.frame(

Model = c("ARIMA", "ETS", "MLP", "Prophet", "XGBoost", "ARIMA\_ETS", "ARIMA\_MLP", "ARIMA\_XGBoost", "ARIMA\_Prophet"),

do.call(rbind, list(

calculate\_metrics(testing\_data, arima\_forecast),

calculate\_metrics(testing\_data, ets\_forecast),

calculate\_metrics(testing\_data, mlp\_forecast),

calculate\_metrics(testing\_data, prophet\_forecast),

calculate\_metrics(testing\_data, xgb\_forecast),

calculate\_metrics(testing\_data, arima\_ets\_forecast),

calculate\_metrics(testing\_data, arima\_mlp\_forecast),

calculate\_metrics(testing\_data, arima\_xgb\_forecast),

calculate\_metrics(testing\_data, arima\_prophet\_forecast))))

print("Model Performance Metrics (RMSE and MAE):")

print(model\_metrics)

all\_forecasts <- data.frame(

Year = seq(2018, 2022),

Actual = as.numeric(testing\_data),

ARIMA = as.numeric(arima\_forecast),

ETS = as.numeric(ets\_forecast),

MLP = as.numeric(mlp\_forecast),

Prophet = as.numeric(prophet\_forecast),

XGBoost = as.numeric(xgb\_forecast),

ARIMA\_ETS = as.numeric(arima\_ets\_forecast),

ARIMA\_MLP = as.numeric(arima\_mlp\_forecast),

ARIMA\_Prophet = as.numeric(arima\_prophet\_forecast),

ARIMA\_XGBoost = as.numeric(arima\_xgb\_forecast))

melted\_forecasts <- melt(all\_forecasts, id.vars = "Year", variable.name = "Model", value.name = "Value")

ggplot(data = melted\_forecasts, aes(x = Year, y = Value, color = Model)) +

geom\_line() +

labs(title = "NGA", y = "Mortality Rate", x = "Year") +

theme\_minimal()

**# Out-of-Sample Forecast for SEN UN-5 Male Mortality Rates**

set.seed(123)

SEN = c(211.62, 205.06, 198.96, 193.02, 186.56, 179.07, 170.65, 162.2, 154.98, 149.37, 145.41, 143.37, 142.75, 143.09, 143.95, 145.06, 145.67, 145.46, 144.02, 140.99, 136, 129.15, 121.23, 112.89, 104.73, 97.138, 90.494, 84.651, 79.607, 75.247, 71.265, 67.703, 64.387, 61.287, 58.381, 55.462, 52.681, 50.216, 47.873, 45.804, 43.966, 42.293, 40.775)

sen\_ts <- ts(SEN, start = 1980, frequency = 1)

training\_data <- window(sen\_ts, start=1980, end=2017)

testing\_data <- window(sen\_ts, start=2018, end=2022)

arima\_model <- auto.arima(training\_data)

arima\_forecast <- forecast(arima\_model, h = length(testing\_data))$mean

ets\_model <- ets(training\_data)

ets\_forecast <- forecast(ets\_model, h = length(testing\_data))$mean

mlp\_model <- mlp(training\_data)

mlp\_forecast <- forecast(mlp\_model, h = length(testing\_data))$mean

train\_df <- data.frame(

ds = seq.Date(from = as.Date("1980-01-01"), by = "year", length.out = length(training\_data)),

y = as.numeric(training\_data))

prophet\_model <- prophet(train\_df, yearly.seasonality = FALSE, weekly.seasonality = FALSE, daily.seasonality = FALSE)

future <- data.frame(ds = seq.Date(from = max(train\_df$ds) + 1, by = "year", length.out = length(testing\_data)))

prophet\_forecast <- predict(prophet\_model, future)$yhat

lagged\_data <- data.frame(

Target = training\_data[-1],

Lag1 = training\_data[-length(training\_data)])

xgb\_model <- xgboost(data = as.matrix(lagged\_data$Lag1), label = lagged\_data$Target, nrounds = 100)

xgb\_forecast <- numeric(length(testing\_data))

last\_value <- training\_data[length(training\_data)]

for (i in 1:length(testing\_data)) {

xgb\_forecast[i] <- predict(xgb\_model, newdata = as.matrix(last\_value))

last\_value <- xgb\_forecast[i]}

hybrid\_forecast\_function <- function(weights, f1, f2) {

weights[1] \* as.numeric(f1) + weights[2] \* as.numeric(f2)}

fitness\_function <- function(weights, f1, f2, actual) {

hybrid\_forecast <- hybrid\_forecast\_function(weights, f1, f2)

sqrt(mean((actual - hybrid\_forecast)^2))}

optimize\_weights <- function(f1, f2, actual) {

ga\_model <- ga(

type = "real-valued",

fitness = function(w) -fitness\_function(w, f1, f2, actual),

lower = c(0, 0),

upper = c(1, 1),

popSize = 50,

maxiter = 100)

return(ga\_model@solution)}

weights <- optimize\_weights(arima\_forecast, ets\_forecast, testing\_data)

arima\_ets\_forecast <- hybrid\_forecast\_function(weights, arima\_forecast, ets\_forecast)

weights <- optimize\_weights(arima\_forecast, mlp\_forecast, testing\_data)

arima\_mlp\_forecast <- hybrid\_forecast\_function(weights, arima\_forecast, mlp\_forecast)

weights <- optimize\_weights(arima\_forecast, xgb\_forecast, testing\_data)

arima\_xgb\_forecast <- hybrid\_forecast\_function(weights, arima\_forecast, xgb\_forecast)

weights <- optimize\_weights(arima\_forecast, prophet\_forecast, testing\_data)

arima\_prophet\_forecast <- hybrid\_forecast\_function(weights, arima\_forecast, prophet\_forecast)

calculate\_metrics <- function(actual, forecast) {

rmse <- sqrt(mean((actual - forecast)^2))

mape <- mean(abs((actual - forecast) / actual)) \* 100

return(c(RMSE = rmse, MAPE = mape))}

model\_metrics <- data.frame(

Model = c("ARIMA", "ETS", "MLP", "Prophet", "XGBoost", "ARIMA\_ETS", "ARIMA\_MLP", "ARIMA\_XGBoost", "ARIMA\_Prophet"),

do.call(rbind, list(

calculate\_metrics(testing\_data, arima\_forecast),

calculate\_metrics(testing\_data, ets\_forecast),

calculate\_metrics(testing\_data, mlp\_forecast),

calculate\_metrics(testing\_data, prophet\_forecast),

calculate\_metrics(testing\_data, xgb\_forecast),

calculate\_metrics(testing\_data, arima\_ets\_forecast),

calculate\_metrics(testing\_data, arima\_mlp\_forecast),

calculate\_metrics(testing\_data, arima\_xgb\_forecast),

calculate\_metrics(testing\_data, arima\_prophet\_forecast))))

print("Model Performance Metrics (RMSE and MAE):")

print(model\_metrics)

all\_forecasts <- data.frame(

Year = seq(2018, 2022),

Actual = as.numeric(testing\_data),

ARIMA = as.numeric(arima\_forecast),

ETS = as.numeric(ets\_forecast),

MLP = as.numeric(mlp\_forecast),

Prophet = as.numeric(prophet\_forecast),

XGBoost = as.numeric(xgb\_forecast),

ARIMA\_ETS = as.numeric(arima\_ets\_forecast),

ARIMA\_MLP = as.numeric(arima\_mlp\_forecast),

ARIMA\_Prophet = as.numeric(arima\_prophet\_forecast),

ARIMA\_XGBoost = as.numeric(arima\_xgb\_forecast))

melted\_forecasts <- melt(all\_forecasts, id.vars = "Year", variable.name = "Model", value.name = "Value")

ggplot(data = melted\_forecasts, aes(x = Year, y = Value, color = Model)) +

geom\_line() +

labs(title = "SEN", y = "Mortality Rate", x = "Year") +

theme\_minimal()

**# Out-of-Sample Forecast for SLE UN-5 Male Mortality Rates**

set.seed(123)

SLE = c(300.39, 296.73, 293.46, 290.18, 286.98, 284.2, 281.51, 278.99, 276.47, 274.32, 272.19, 270.38, 268.39, 265.99, 263.04, 259.49, 255.39, 250.9, 245.93, 240.48, 235.21, 229.3, 223.42, 217.22, 210.66, 204.11, 197.26, 190.27, 183.01, 175.55, 168.57, 161.66, 155.12, 148.96, 147.04, 147.99, 133.07, 128.21, 123.65, 119.3, 115.07, 111.22, 107.51)

sle\_ts <- ts(SLE, start = 1980, frequency = 1)

training\_data <- window(sle\_ts, start=1980, end=2017)

testing\_data <- window(sle\_ts, start=2018, end=2022)

arima\_model <- auto.arima(training\_data)

arima\_forecast <- forecast(arima\_model, h = length(testing\_data))$mean

ets\_model <- ets(training\_data)

ets\_forecast <- forecast(ets\_model, h = length(testing\_data))$mean

mlp\_model <- mlp(training\_data)

mlp\_forecast <- forecast(mlp\_model, h = length(testing\_data))$mean

train\_df <- data.frame(

ds = seq.Date(from = as.Date("1980-01-01"), by = "year", length.out = length(training\_data)),

y = as.numeric(training\_data))

prophet\_model <- prophet(train\_df, yearly.seasonality = FALSE, weekly.seasonality = FALSE, daily.seasonality = FALSE)

future <- data.frame(ds = seq.Date(from = max(train\_df$ds) + 1, by = "year", length.out = length(testing\_data)))

prophet\_forecast <- predict(prophet\_model, future)$yhat

lagged\_data <- data.frame(

Target = training\_data[-1],

Lag1 = training\_data[-length(training\_data)])

xgb\_model <- xgboost(data = as.matrix(lagged\_data$Lag1), label = lagged\_data$Target, nrounds = 100)

xgb\_forecast <- numeric(length(testing\_data))

last\_value <- training\_data[length(training\_data)]

for (i in 1:length(testing\_data)) {

xgb\_forecast[i] <- predict(xgb\_model, newdata = as.matrix(last\_value))

last\_value <- xgb\_forecast[i]}

hybrid\_forecast\_function <- function(weights, f1, f2) {

weights[1] \* as.numeric(f1) + weights[2] \* as.numeric(f2)}

fitness\_function <- function(weights, f1, f2, actual) {

hybrid\_forecast <- hybrid\_forecast\_function(weights, f1, f2)

sqrt(mean((actual - hybrid\_forecast)^2))}

optimize\_weights <- function(f1, f2, actual) {

ga\_model <- ga(

type = "real-valued",

fitness = function(w) -fitness\_function(w, f1, f2, actual),

lower = c(0, 0),

upper = c(1, 1),

popSize = 50,

maxiter = 100)

return(ga\_model@solution)}

weights <- optimize\_weights(arima\_forecast, ets\_forecast, testing\_data)

arima\_ets\_forecast <- hybrid\_forecast\_function(weights, arima\_forecast, ets\_forecast)

weights <- optimize\_weights(arima\_forecast, mlp\_forecast, testing\_data)

arima\_mlp\_forecast <- hybrid\_forecast\_function(weights, arima\_forecast, mlp\_forecast)

weights <- optimize\_weights(arima\_forecast, xgb\_forecast, testing\_data)

arima\_xgb\_forecast <- hybrid\_forecast\_function(weights, arima\_forecast, xgb\_forecast)

weights <- optimize\_weights(arima\_forecast, prophet\_forecast, testing\_data)

arima\_prophet\_forecast <- hybrid\_forecast\_function(weights, arima\_forecast, prophet\_forecast)

calculate\_metrics <- function(actual, forecast) {

rmse <- sqrt(mean((actual - forecast)^2))

mape <- mean(abs((actual - forecast) / actual)) \* 100

return(c(RMSE = rmse, MAPE = mape))}

model\_metrics <- data.frame(

Model = c("ARIMA", "ETS", "MLP", "Prophet", "XGBoost", "ARIMA\_ETS", "ARIMA\_MLP", "ARIMA\_XGBoost", "ARIMA\_Prophet"),

do.call(rbind, list(

calculate\_metrics(testing\_data, arima\_forecast),

calculate\_metrics(testing\_data, ets\_forecast),

calculate\_metrics(testing\_data, mlp\_forecast),

calculate\_metrics(testing\_data, prophet\_forecast),

calculate\_metrics(testing\_data, xgb\_forecast),

calculate\_metrics(testing\_data, arima\_ets\_forecast),

calculate\_metrics(testing\_data, arima\_mlp\_forecast),

calculate\_metrics(testing\_data, arima\_xgb\_forecast),

calculate\_metrics(testing\_data, arima\_prophet\_forecast))))

print("Model Performance Metrics (RMSE and MAE):")

print(model\_metrics)

all\_forecasts <- data.frame(

Year = seq(2018, 2022),

Actual = as.numeric(testing\_data),

ARIMA = as.numeric(arima\_forecast),

ETS = as.numeric(ets\_forecast),

MLP = as.numeric(mlp\_forecast),

Prophet = as.numeric(prophet\_forecast),

XGBoost = as.numeric(xgb\_forecast),

ARIMA\_ETS = as.numeric(arima\_ets\_forecast),

ARIMA\_MLP = as.numeric(arima\_mlp\_forecast),

ARIMA\_Prophet = as.numeric(arima\_prophet\_forecast),

ARIMA\_XGBoost = as.numeric(arima\_xgb\_forecast))

melted\_forecasts <- melt(all\_forecasts, id.vars = "Year", variable.name = "Model", value.name = "Value")

ggplot(data = melted\_forecasts, aes(x = Year, y = Value, color = Model)) +

geom\_line() +

labs(title = "SLE", y = "Mortality Rate", x = "Year") +

theme\_minimal()

**# Out-of-Sample Forecast for TGO UN-5 Male Mortality Rates**

set.seed(123)

TGO = c(187.53, 183.2, 179.06, 175.14, 171.48, 168.16, 165.26, 162.69, 160.21, 158.02, 155.99, 153.96, 152, 149.99, 147.7, 145.07, 142.2, 138.9, 135.33, 131.82, 128.14, 124.36, 120.6, 116.95, 113.45, 110.11, 106.82, 103.76, 100.73, 97.691, 94.78, 92, 89.156, 86.428, 83.783, 81.165, 78.681, 76.289, 73.919, 71.605, 69.329, 67.244, 65.13)

tgo\_ts <- ts(TGO, start = 1980, frequency = 1)

training\_data <- window(tgo\_ts, start=1980, end=2017)

testing\_data <- window(tgo\_ts, start=2018, end=2022)

arima\_model <- auto.arima(training\_data)

arima\_forecast <- forecast(arima\_model, h = length(testing\_data))$mean

ets\_model <- ets(training\_data)

ets\_forecast <- forecast(ets\_model, h = length(testing\_data))$mean

mlp\_model <- mlp(training\_data)

mlp\_forecast <- forecast(mlp\_model, h = length(testing\_data))$mean

train\_df <- data.frame(

ds = seq.Date(from = as.Date("1980-01-01"), by = "year", length.out = length(training\_data)),

y = as.numeric(training\_data))

prophet\_model <- prophet(train\_df, yearly.seasonality = FALSE, weekly.seasonality = FALSE, daily.seasonality = FALSE)

future <- data.frame(ds = seq.Date(from = max(train\_df$ds) + 1, by = "year", length.out = length(testing\_data)))

prophet\_forecast <- predict(prophet\_model, future)$yhat

lagged\_data <- data.frame(

Target = training\_data[-1],

Lag1 = training\_data[-length(training\_data)])

xgb\_model <- xgboost(data = as.matrix(lagged\_data$Lag1), label = lagged\_data$Target, nrounds = 100)

xgb\_forecast <- numeric(length(testing\_data))

last\_value <- training\_data[length(training\_data)]

for (i in 1:length(testing\_data)) {

xgb\_forecast[i] <- predict(xgb\_model, newdata = as.matrix(last\_value))

last\_value <- xgb\_forecast[i]}

hybrid\_forecast\_function <- function(weights, f1, f2) {

weights[1] \* as.numeric(f1) + weights[2] \* as.numeric(f2)}

fitness\_function <- function(weights, f1, f2, actual) {

hybrid\_forecast <- hybrid\_forecast\_function(weights, f1, f2)

sqrt(mean((actual - hybrid\_forecast)^2))}

optimize\_weights <- function(f1, f2, actual) {

ga\_model <- ga(

type = "real-valued",

fitness = function(w) -fitness\_function(w, f1, f2, actual),

lower = c(0, 0),

upper = c(1, 1),

popSize = 50,

maxiter = 100)

return(ga\_model@solution)}

weights <- optimize\_weights(arima\_forecast, ets\_forecast, testing\_data)

arima\_ets\_forecast <- hybrid\_forecast\_function(weights, arima\_forecast, ets\_forecast)

weights <- optimize\_weights(arima\_forecast, mlp\_forecast, testing\_data)

arima\_mlp\_forecast <- hybrid\_forecast\_function(weights, arima\_forecast, mlp\_forecast)

weights <- optimize\_weights(arima\_forecast, xgb\_forecast, testing\_data)

arima\_xgb\_forecast <- hybrid\_forecast\_function(weights, arima\_forecast, xgb\_forecast)

weights <- optimize\_weights(arima\_forecast, prophet\_forecast, testing\_data)

arima\_prophet\_forecast <- hybrid\_forecast\_function(weights, arima\_forecast, prophet\_forecast)

calculate\_metrics <- function(actual, forecast) {

rmse <- sqrt(mean((actual - forecast)^2))

mape <- mean(abs((actual - forecast) / actual)) \* 100

return(c(RMSE = rmse, MAPE = mape))}

model\_metrics <- data.frame(

Model = c("ARIMA", "ETS", "MLP", "Prophet", "XGBoost", "ARIMA\_ETS", "ARIMA\_MLP", "ARIMA\_XGBoost", "ARIMA\_Prophet"),

do.call(rbind, list(

calculate\_metrics(testing\_data, arima\_forecast),

calculate\_metrics(testing\_data, ets\_forecast),

calculate\_metrics(testing\_data, mlp\_forecast),

calculate\_metrics(testing\_data, prophet\_forecast),

calculate\_metrics(testing\_data, xgb\_forecast),

calculate\_metrics(testing\_data, arima\_ets\_forecast),

calculate\_metrics(testing\_data, arima\_mlp\_forecast),

calculate\_metrics(testing\_data, arima\_xgb\_forecast),

calculate\_metrics(testing\_data, arima\_prophet\_forecast))))

print("Model Performance Metrics (RMSE and MAE):")

print(model\_metrics)

all\_forecasts <- data.frame(

Year = seq(2018, 2022),

Actual = as.numeric(testing\_data),

ARIMA = as.numeric(arima\_forecast),

ETS = as.numeric(ets\_forecast),

MLP = as.numeric(mlp\_forecast),

Prophet = as.numeric(prophet\_forecast),

XGBoost = as.numeric(xgb\_forecast),

ARIMA\_ETS = as.numeric(arima\_ets\_forecast),

ARIMA\_MLP = as.numeric(arima\_mlp\_forecast),

ARIMA\_Prophet = as.numeric(arima\_prophet\_forecast),

ARIMA\_XGBoost = as.numeric(arima\_xgb\_forecast))

melted\_forecasts <- melt(all\_forecasts, id.vars = "Year", variable.name = "Model", value.name = "Value")

ggplot(data = melted\_forecasts, aes(x = Year, y = Value, color = Model)) +

geom\_line() +

labs(title = "TGO", y = "Mortality Rate", x = "Year") +

theme\_minimal()

**# Sensitivity Analysis for BEN UN-5 Male Mortality using Time Horizon: 3, 5, and 7 years**

set.seed(123)

BEN = c(222.61, 218.26, 214.02, 209.88, 205.56, 201.31, 196.89, 192.56, 188.19, 183.71, 179.31, 174.92, 170.57, 166.25, 162.28, 158.55, 155.01, 151.5, 148.16, 145.25, 142.29, 139.35, 136.26, 133.13, 130.12, 127.33, 124.65, 122.14, 119.87, 117.63, 115.36, 113.17, 110.87, 108.57, 106.29, 104.03, 101.72, 99.419, 96.985, 94.427, 91.807, 89.172, 86.516)

ben\_ts <- ts(BEN, start = 1980, frequency = 1)

horizons <- c(3, 5, 7)

results <- data.frame(Horizon = integer(), Model = character(), RMSE = numeric())

for (h in horizons) {

training\_data <- window(ben\_ts, start = 1980, end = 2017)

testing\_data <- window(ben\_ts, start = 2018, end = 2017 + h)

arima\_model <- auto.arima(training\_data)

arima\_forecast <- forecast(arima\_model, h = length(testing\_data))$mean

ets\_model <- ets(training\_data)

ets\_forecast <- forecast(ets\_model, h = length(testing\_data))$mean

mlp\_model <- mlp(training\_data)

mlp\_forecast <- forecast(mlp\_model, h = length(testing\_data))$mean

train\_df <- data.frame(

ds = seq.Date(from = as.Date("1980-01-01"), by = "year", length.out = length(training\_data)),

y = as.numeric(training\_data))

prophet\_model <- prophet(train\_df, yearly.seasonality = FALSE)

future <- data.frame(ds = seq.Date(from = max(train\_df$ds) + 1, by = "year", length.out = length(testing\_data)))

prophet\_forecast <- predict(prophet\_model, future)$yhat

lagged\_data <- data.frame(Target = training\_data[-1], Lag1 = training\_data[-length(training\_data)])

xgb\_model <- xgboost(data = as.matrix(lagged\_data$Lag1), label = lagged\_data$Target, nrounds = 100)

xgb\_forecast <- numeric(length(testing\_data))

last\_value <- training\_data[length(training\_data)]

for (i in 1:length(testing\_data)) {

xgb\_forecast[i] <- predict(xgb\_model, newdata = as.matrix(last\_value))

last\_value <- xgb\_forecast[i]}

hybrid\_forecast\_function <- function(weights, f1, f2) {

weights[1] \* as.numeric(f1) + weights[2] \* as.numeric(f2)}

optimize\_weights <- function(f1, f2, actual) {ga\_model <- ga(type = "real-valued", fitness = function(w) -sqrt(mean((actual - hybrid\_forecast\_function(w, f1, f2))^2)), lower = c(0, 0), upper = c(1, 1), popSize = 50, maxiter = 100)

return(ga\_model@solution)}

weights <- optimize\_weights(arima\_forecast, ets\_forecast, testing\_data)

arima\_ets\_forecast <- hybrid\_forecast\_function(weights, arima\_forecast, ets\_forecast)

weights <- optimize\_weights(arima\_forecast, mlp\_forecast, testing\_data)

arima\_mlp\_forecast <- hybrid\_forecast\_function(weights, arima\_forecast, mlp\_forecast)

weights <- optimize\_weights(arima\_forecast, prophet\_forecast, testing\_data)

arima\_prophet\_forecast <- hybrid\_forecast\_function(weights, arima\_forecast, prophet\_forecast)

weights <- optimize\_weights(arima\_forecast, xgb\_forecast, testing\_data)

arima\_xgb\_forecast <- hybrid\_forecast\_function(weights, arima\_forecast, xgb\_forecast)

calculate\_rmse <- function(actual, forecast) {

sqrt(mean((actual - forecast)^2))}

models <- list(

ARIMA = arima\_forecast,

ETS = ets\_forecast,

MLP = mlp\_forecast,

Prophet = prophet\_forecast,

XGBoost = xgb\_forecast,

ARIMA\_ETS = arima\_ets\_forecast,

ARIMA\_MLP = arima\_mlp\_forecast,

ARIMA\_Prophet = arima\_prophet\_forecast,

ARIMA\_XGBoost = arima\_xgb\_forecast)

for (model\_name in names(models)) {

rmse <- calculate\_rmse(testing\_data, models[[model\_name]])

results <- rbind(results, data.frame(Horizon = h, Model = model\_name, RMSE = rmse))}}

print("Sensitivity Analysis Results:")

print(results)

ggplot(results, aes(x = Horizon, y = RMSE, color = Model, group = Model)) +geom\_line() +geom\_point() +labs(title = "BEN", y = "RMSE", x = "Forecast Horizon (Years)") +theme\_minimal()

**# Sensitivity Analysis for BFA UN-5 Male Mortality using Time Horizon: 3, 5, and 7 years**

set.seed(123)

BFA = c(248.21, 241.81, 236.12, 230.91, 225.96, 221.39, 216.92, 212.89, 209.53, 207.06, 205.35, 204.24, 203.63, 202.75, 201.63, 199.56, 197.01, 194.02, 190.97, 187.65, 184.08, 180.11, 175.79, 170.6, 164.54, 158.01, 151.01, 144.1, 137.52, 131.45, 126.02, 121.26, 116.9, 112.83, 109.03, 105.42, 101.74, 98.319, 94.992, 92.171, 89.15, 86.099, 83.342)

bfa\_ts <- ts(BFA, start = 1980, frequency = 1)

horizons <- c(3, 5, 7)

results <- data.frame(Horizon = integer(), Model = character(), RMSE = numeric())

for (h in horizons) {

training\_data <- window(bfa\_ts, start = 1980, end = 2017)

testing\_data <- window(bfa\_ts, start = 2018, end = 2017 + h)

arima\_model <- auto.arima(training\_data)

arima\_forecast <- forecast(arima\_model, h = length(testing\_data))$mean

ets\_model <- ets(training\_data)

ets\_forecast <- forecast(ets\_model, h = length(testing\_data))$mean

mlp\_model <- mlp(training\_data)

mlp\_forecast <- forecast(mlp\_model, h = length(testing\_data))$mean

train\_df <- data.frame(

ds = seq.Date(from = as.Date("1980-01-01"), by = "year", length.out = length(training\_data)),

y = as.numeric(training\_data))

prophet\_model <- prophet(train\_df, yearly.seasonality = FALSE)

future <- data.frame(ds = seq.Date(from = max(train\_df$ds) + 1, by = "year", length.out = length(testing\_data)))

prophet\_forecast <- predict(prophet\_model, future)$yhat

lagged\_data <- data.frame(Target = training\_data[-1], Lag1 = training\_data[-length(training\_data)])

xgb\_model <- xgboost(data = as.matrix(lagged\_data$Lag1), label = lagged\_data$Target, nrounds = 100)

xgb\_forecast <- numeric(length(testing\_data))

last\_value <- training\_data[length(training\_data)]

for (i in 1:length(testing\_data)) {

xgb\_forecast[i] <- predict(xgb\_model, newdata = as.matrix(last\_value))

last\_value <- xgb\_forecast[i]}

hybrid\_forecast\_function <- function(weights, f1, f2) {

weights[1] \* as.numeric(f1) + weights[2] \* as.numeric(f2)}

optimize\_weights <- function(f1, f2, actual) {ga\_model <- ga(type = "real-valued", fitness = function(w) -sqrt(mean((actual - hybrid\_forecast\_function(w, f1, f2))^2)), lower = c(0, 0), upper = c(1, 1), popSize = 50, maxiter = 100)

return(ga\_model@solution)}

weights <- optimize\_weights(arima\_forecast, ets\_forecast, testing\_data)

arima\_ets\_forecast <- hybrid\_forecast\_function(weights, arima\_forecast, ets\_forecast)

weights <- optimize\_weights(arima\_forecast, mlp\_forecast, testing\_data)

arima\_mlp\_forecast <- hybrid\_forecast\_function(weights, arima\_forecast, mlp\_forecast)

weights <- optimize\_weights(arima\_forecast, prophet\_forecast, testing\_data)

arima\_prophet\_forecast <- hybrid\_forecast\_function(weights, arima\_forecast, prophet\_forecast)

weights <- optimize\_weights(arima\_forecast, xgb\_forecast, testing\_data)

arima\_xgb\_forecast <- hybrid\_forecast\_function(weights, arima\_forecast, xgb\_forecast)

calculate\_rmse <- function(actual, forecast) {

sqrt(mean((actual - forecast)^2))}

models <- list(

ARIMA = arima\_forecast,

ETS = ets\_forecast,

MLP = mlp\_forecast,

Prophet = prophet\_forecast,

XGBoost = xgb\_forecast,

ARIMA\_ETS = arima\_ets\_forecast,

ARIMA\_MLP = arima\_mlp\_forecast,

ARIMA\_Prophet = arima\_prophet\_forecast,

ARIMA\_XGBoost = arima\_xgb\_forecast)

for (model\_name in names(models)) {

rmse <- calculate\_rmse(testing\_data, models[[model\_name]])

results <- rbind(results, data.frame(Horizon = h, Model = model\_name, RMSE = rmse))}}

print("Sensitivity Analysis Results:")

print(results)

ggplot(results, aes(x = Horizon, y = RMSE, color = Model, group = Model)) +geom\_line() +geom\_point() +labs(title = "BFA", y = "RMSE", x = "Forecast Horizon (Years)") +theme\_minimal()

**# Sensitivity Analysis for GIN UN-5 Male Mortality using Time Horizon: 3, 5, and 7 years**

set.seed(123)

GIN = c(289.64, 284.73, 279.85, 274.71, 269.65, 264.85, 260.11, 255.59, 251.02, 246.14, 240.82, 235.09, 228.6, 222.07, 215.38, 208.5, 201.67, 194.82, 187.96, 180.99, 174.08, 167.36, 161.06, 155.22, 150.02, 145.41, 141.47, 138.11, 135.23, 132.65, 130.27, 127.9, 125.77, 123.68, 122.32, 120.47, 117.99, 116.09, 113.88, 111.45, 108.85, 106.06, 103.14)

gin\_ts <- ts(GIN, start = 1980, frequency = 1)

horizons <- c(3, 5, 7)

results <- data.frame(Horizon = integer(), Model = character(), RMSE = numeric())

for (h in horizons) {

training\_data <- window(gin\_ts, start = 1980, end = 2017)

testing\_data <- window(gin\_ts, start = 2018, end = 2017 + h)

arima\_model <- auto.arima(training\_data)

arima\_forecast <- forecast(arima\_model, h = length(testing\_data))$mean

ets\_model <- ets(training\_data)

ets\_forecast <- forecast(ets\_model, h = length(testing\_data))$mean

mlp\_model <- mlp(training\_data)

mlp\_forecast <- forecast(mlp\_model, h = length(testing\_data))$mean

train\_df <- data.frame(

ds = seq.Date(from = as.Date("1980-01-01"), by = "year", length.out = length(training\_data)),

y = as.numeric(training\_data))

prophet\_model <- prophet(train\_df, yearly.seasonality = FALSE)

future <- data.frame(ds = seq.Date(from = max(train\_df$ds) + 1, by = "year", length.out = length(testing\_data)))

prophet\_forecast <- predict(prophet\_model, future)$yhat

lagged\_data <- data.frame(Target = training\_data[-1], Lag1 = training\_data[-length(training\_data)])

xgb\_model <- xgboost(data = as.matrix(lagged\_data$Lag1), label = lagged\_data$Target, nrounds = 100)

xgb\_forecast <- numeric(length(testing\_data))

last\_value <- training\_data[length(training\_data)]

for (i in 1:length(testing\_data)) {

xgb\_forecast[i] <- predict(xgb\_model, newdata = as.matrix(last\_value))

last\_value <- xgb\_forecast[i]}

hybrid\_forecast\_function <- function(weights, f1, f2) {

weights[1] \* as.numeric(f1) + weights[2] \* as.numeric(f2)}

optimize\_weights <- function(f1, f2, actual) {ga\_model <- ga(type = "real-valued", fitness = function(w) -sqrt(mean((actual - hybrid\_forecast\_function(w, f1, f2))^2)), lower = c(0, 0), upper = c(1, 1), popSize = 50, maxiter = 100)

return(ga\_model@solution)}

weights <- optimize\_weights(arima\_forecast, ets\_forecast, testing\_data)

arima\_ets\_forecast <- hybrid\_forecast\_function(weights, arima\_forecast, ets\_forecast)

weights <- optimize\_weights(arima\_forecast, mlp\_forecast, testing\_data)

arima\_mlp\_forecast <- hybrid\_forecast\_function(weights, arima\_forecast, mlp\_forecast)

weights <- optimize\_weights(arima\_forecast, prophet\_forecast, testing\_data)

arima\_prophet\_forecast <- hybrid\_forecast\_function(weights, arima\_forecast, prophet\_forecast)

weights <- optimize\_weights(arima\_forecast, xgb\_forecast, testing\_data)

arima\_xgb\_forecast <- hybrid\_forecast\_function(weights, arima\_forecast, xgb\_forecast)

calculate\_rmse <- function(actual, forecast) {

sqrt(mean((actual - forecast)^2))}

models <- list(

ARIMA = arima\_forecast,

ETS = ets\_forecast,

MLP = mlp\_forecast,

Prophet = prophet\_forecast,

XGBoost = xgb\_forecast,

ARIMA\_ETS = arima\_ets\_forecast,

ARIMA\_MLP = arima\_mlp\_forecast,

ARIMA\_Prophet = arima\_prophet\_forecast,

ARIMA\_XGBoost = arima\_xgb\_forecast)

for (model\_name in names(models)) {

rmse <- calculate\_rmse(testing\_data, models[[model\_name]])

results <- rbind(results, data.frame(Horizon = h, Model = model\_name, RMSE = rmse))}}

print("Sensitivity Analysis Results:")

print(results)

ggplot(results, aes(x = Horizon, y = RMSE, color = Model, group = Model)) +geom\_line() +geom\_point() +labs(title = "GIN", y = "RMSE", x = "Forecast Horizon (Years)") +theme\_minimal()

**# Sensitivity Analysis for GNB UN-5 Male Mortality using Time Horizon: 3, 5, and 7 years**

set.seed(123)

GNB = c(255.03, 255.03, 255.03, 255.03, 255.03, 255.03, 250.16, 245.56, 241.3, 237.17, 232.89, 228.42, 223.74, 219.02, 214.19, 209.2, 204.12, 199.12, 194.21, 189.18, 183.82, 178.4, 172.82, 166.88, 160.78, 154.57, 148.15, 141.69, 135.22, 128.94, 122.76, 117.07, 111.81, 107.22, 102.82, 99.118, 95.589, 92.405, 89.207, 86.104, 83.296, 80.465, 77.896)

gnb\_ts <- ts(GNB, start = 1980, frequency = 1)

horizons <- c(3, 5, 7)

results <- data.frame(Horizon = integer(), Model = character(), RMSE = numeric())

for (h in horizons) {

training\_data <- window(gnb\_ts, start = 1980, end = 2017)

testing\_data <- window(gnb\_ts, start = 2018, end = 2017 + h)

arima\_model <- auto.arima(training\_data)

arima\_forecast <- forecast(arima\_model, h = length(testing\_data))$mean

ets\_model <- ets(training\_data)

ets\_forecast <- forecast(ets\_model, h = length(testing\_data))$mean

mlp\_model <- mlp(training\_data)

mlp\_forecast <- forecast(mlp\_model, h = length(testing\_data))$mean

train\_df <- data.frame(

ds = seq.Date(from = as.Date("1980-01-01"), by = "year", length.out = length(training\_data)),

y = as.numeric(training\_data))

prophet\_model <- prophet(train\_df, yearly.seasonality = FALSE)

future <- data.frame(ds = seq.Date(from = max(train\_df$ds) + 1, by = "year", length.out = length(testing\_data)))

prophet\_forecast <- predict(prophet\_model, future)$yhat

lagged\_data <- data.frame(Target = training\_data[-1], Lag1 = training\_data[-length(training\_data)])

xgb\_model <- xgboost(data = as.matrix(lagged\_data$Lag1), label = lagged\_data$Target, nrounds = 100)

xgb\_forecast <- numeric(length(testing\_data))

last\_value <- training\_data[length(training\_data)]

for (i in 1:length(testing\_data)) {

xgb\_forecast[i] <- predict(xgb\_model, newdata = as.matrix(last\_value))

last\_value <- xgb\_forecast[i]}

hybrid\_forecast\_function <- function(weights, f1, f2) {

weights[1] \* as.numeric(f1) + weights[2] \* as.numeric(f2)}

optimize\_weights <- function(f1, f2, actual) {ga\_model <- ga(type = "real-valued", fitness = function(w) -sqrt(mean((actual - hybrid\_forecast\_function(w, f1, f2))^2)), lower = c(0, 0), upper = c(1, 1), popSize = 50, maxiter = 100)

return(ga\_model@solution)}

weights <- optimize\_weights(arima\_forecast, ets\_forecast, testing\_data)

arima\_ets\_forecast <- hybrid\_forecast\_function(weights, arima\_forecast, ets\_forecast)

weights <- optimize\_weights(arima\_forecast, mlp\_forecast, testing\_data)

arima\_mlp\_forecast <- hybrid\_forecast\_function(weights, arima\_forecast, mlp\_forecast)

weights <- optimize\_weights(arima\_forecast, prophet\_forecast, testing\_data)

arima\_prophet\_forecast <- hybrid\_forecast\_function(weights, arima\_forecast, prophet\_forecast)

weights <- optimize\_weights(arima\_forecast, xgb\_forecast, testing\_data)

arima\_xgb\_forecast <- hybrid\_forecast\_function(weights, arima\_forecast, xgb\_forecast)

calculate\_rmse <- function(actual, forecast) {

sqrt(mean((actual - forecast)^2))}

models <- list(

ARIMA = arima\_forecast,

ETS = ets\_forecast,

MLP = mlp\_forecast,

Prophet = prophet\_forecast,

XGBoost = xgb\_forecast,

ARIMA\_ETS = arima\_ets\_forecast,

ARIMA\_MLP = arima\_mlp\_forecast,

ARIMA\_Prophet = arima\_prophet\_forecast,

ARIMA\_XGBoost = arima\_xgb\_forecast)

for (model\_name in names(models)) {

rmse <- calculate\_rmse(testing\_data, models[[model\_name]])

results <- rbind(results, data.frame(Horizon = h, Model = model\_name, RMSE = rmse))}}

print("Sensitivity Analysis Results:")

print(results)

ggplot(results, aes(x = Horizon, y = RMSE, color = Model, group = Model)) +geom\_line() +geom\_point() +labs(title = "GNB", y = "RMSE", x = "Forecast Horizon (Years)") +theme\_minimal()

**# Sensitivity Analysis for LBR UN-5 Male Mortality using Time Horizon: 3, 5, and 7 years**

set.seed(123)

LBR = c(257.9, 254.9, 252.63, 251.31, 251.38, 253.12, 256.79, 261.92, 267.8, 273.4, 277.82, 279.7, 278.8, 275.22, 268.86, 260.27, 249.95, 238.37, 225.77, 212.9, 199.78, 186.18, 172.89, 160.19, 148.35, 137.69, 128.73, 121.25, 115.23, 110.34, 106.43, 103.27, 100.56, 98.226, 98.232, 94.286, 92.204, 90.23, 88.193, 85.849, 83.494, 81.188, 78.972)

lbr\_ts <- ts(LBR, start = 1980, frequency = 1)

horizons <- c(3, 5, 7)

results <- data.frame(Horizon = integer(), Model = character(), RMSE = numeric())

for (h in horizons) {

training\_data <- window(lbr\_ts, start = 1980, end = 2017)

testing\_data <- window(lbr\_ts, start = 2018, end = 2017 + h)

arima\_model <- auto.arima(training\_data)

arima\_forecast <- forecast(arima\_model, h = length(testing\_data))$mean

ets\_model <- ets(training\_data)

ets\_forecast <- forecast(ets\_model, h = length(testing\_data))$mean

mlp\_model <- mlp(training\_data)

mlp\_forecast <- forecast(mlp\_model, h = length(testing\_data))$mean

train\_df <- data.frame(

ds = seq.Date(from = as.Date("1980-01-01"), by = "year", length.out = length(training\_data)),

y = as.numeric(training\_data))

prophet\_model <- prophet(train\_df, yearly.seasonality = FALSE)

future <- data.frame(ds = seq.Date(from = max(train\_df$ds) + 1, by = "year", length.out = length(testing\_data)))

prophet\_forecast <- predict(prophet\_model, future)$yhat

lagged\_data <- data.frame(Target = training\_data[-1], Lag1 = training\_data[-length(training\_data)])

xgb\_model <- xgboost(data = as.matrix(lagged\_data$Lag1), label = lagged\_data$Target, nrounds = 100)

xgb\_forecast <- numeric(length(testing\_data))

last\_value <- training\_data[length(training\_data)]

for (i in 1:length(testing\_data)) {

xgb\_forecast[i] <- predict(xgb\_model, newdata = as.matrix(last\_value))

last\_value <- xgb\_forecast[i]}

hybrid\_forecast\_function <- function(weights, f1, f2) {

weights[1] \* as.numeric(f1) + weights[2] \* as.numeric(f2)}

optimize\_weights <- function(f1, f2, actual) {ga\_model <- ga(type = "real-valued", fitness = function(w) -sqrt(mean((actual - hybrid\_forecast\_function(w, f1, f2))^2)), lower = c(0, 0), upper = c(1, 1), popSize = 50, maxiter = 100)

return(ga\_model@solution)}

weights <- optimize\_weights(arima\_forecast, ets\_forecast, testing\_data)

arima\_ets\_forecast <- hybrid\_forecast\_function(weights, arima\_forecast, ets\_forecast)

weights <- optimize\_weights(arima\_forecast, mlp\_forecast, testing\_data)

arima\_mlp\_forecast <- hybrid\_forecast\_function(weights, arima\_forecast, mlp\_forecast)

weights <- optimize\_weights(arima\_forecast, prophet\_forecast, testing\_data)

arima\_prophet\_forecast <- hybrid\_forecast\_function(weights, arima\_forecast, prophet\_forecast)

weights <- optimize\_weights(arima\_forecast, xgb\_forecast, testing\_data)

arima\_xgb\_forecast <- hybrid\_forecast\_function(weights, arima\_forecast, xgb\_forecast)

calculate\_rmse <- function(actual, forecast) {

sqrt(mean((actual - forecast)^2))}

models <- list(

ARIMA = arima\_forecast,

ETS = ets\_forecast,

MLP = mlp\_forecast,

Prophet = prophet\_forecast,

XGBoost = xgb\_forecast,

ARIMA\_ETS = arima\_ets\_forecast,

ARIMA\_MLP = arima\_mlp\_forecast,

ARIMA\_Prophet = arima\_prophet\_forecast,

ARIMA\_XGBoost = arima\_xgb\_forecast)

for (model\_name in names(models)) {

rmse <- calculate\_rmse(testing\_data, models[[model\_name]])

results <- rbind(results, data.frame(Horizon = h, Model = model\_name, RMSE = rmse))}}

print("Sensitivity Analysis Results:")

print(results)

ggplot(results, aes(x = Horizon, y = RMSE, color = Model, group = Model)) +geom\_line() +geom\_point() +labs(title = "LBR", y = "RMSE", x = "Forecast Horizon (Years)") +theme\_minimal()

**# Sensitivity Analysis for MLI UN-5 Male Mortality using Time Horizon: 3, 5, and 7 years**

set.seed(123)

MLI = c(308.142, 299.993, 291.887, 283.921, 276.039, 268.291, 261.071, 254.887, 249.398, 244.094, 239.312, 235.079, 231.208, 227.784, 224.089, 220.369, 216.334, 211.819, 206.747, 201.047, 194.925, 188.617, 182.162, 175.681, 169.571, 163.814, 158.41, 153.433, 148.558, 143.989, 139.753, 135.552, 131.75, 128.071, 124.425, 120.894, 117.44, 114.097, 111, 107.863, 104.577, 101.615, 98.6436)

mli\_ts <- ts(MLI, start = 1980, frequency = 1)

horizons <- c(3, 5, 7)

results <- data.frame(Horizon = integer(), Model = character(), RMSE = numeric())

for (h in horizons) {

training\_data <- window(mli\_ts, start = 1980, end = 2017)

testing\_data <- window(mli\_ts, start = 2018, end = 2017 + h)

arima\_model <- auto.arima(training\_data)

arima\_forecast <- forecast(arima\_model, h = length(testing\_data))$mean

ets\_model <- ets(training\_data)

ets\_forecast <- forecast(ets\_model, h = length(testing\_data))$mean

mlp\_model <- mlp(training\_data)

mlp\_forecast <- forecast(mlp\_model, h = length(testing\_data))$mean

train\_df <- data.frame(

ds = seq.Date(from = as.Date("1980-01-01"), by = "year", length.out = length(training\_data)),

y = as.numeric(training\_data))

prophet\_model <- prophet(train\_df, yearly.seasonality = FALSE)

future <- data.frame(ds = seq.Date(from = max(train\_df$ds) + 1, by = "year", length.out = length(testing\_data)))

prophet\_forecast <- predict(prophet\_model, future)$yhat

lagged\_data <- data.frame(Target = training\_data[-1], Lag1 = training\_data[-length(training\_data)])

xgb\_model <- xgboost(data = as.matrix(lagged\_data$Lag1), label = lagged\_data$Target, nrounds = 100)

xgb\_forecast <- numeric(length(testing\_data))

last\_value <- training\_data[length(training\_data)]

for (i in 1:length(testing\_data)) {

xgb\_forecast[i] <- predict(xgb\_model, newdata = as.matrix(last\_value))

last\_value <- xgb\_forecast[i]}

hybrid\_forecast\_function <- function(weights, f1, f2) {

weights[1] \* as.numeric(f1) + weights[2] \* as.numeric(f2)}

optimize\_weights <- function(f1, f2, actual) {ga\_model <- ga(type = "real-valued", fitness = function(w) -sqrt(mean((actual - hybrid\_forecast\_function(w, f1, f2))^2)), lower = c(0, 0), upper = c(1, 1), popSize = 50, maxiter = 100)

return(ga\_model@solution)}

weights <- optimize\_weights(arima\_forecast, ets\_forecast, testing\_data)

arima\_ets\_forecast <- hybrid\_forecast\_function(weights, arima\_forecast, ets\_forecast)

weights <- optimize\_weights(arima\_forecast, mlp\_forecast, testing\_data)

arima\_mlp\_forecast <- hybrid\_forecast\_function(weights, arima\_forecast, mlp\_forecast)

weights <- optimize\_weights(arima\_forecast, prophet\_forecast, testing\_data)

arima\_prophet\_forecast <- hybrid\_forecast\_function(weights, arima\_forecast, prophet\_forecast)

weights <- optimize\_weights(arima\_forecast, xgb\_forecast, testing\_data)

arima\_xgb\_forecast <- hybrid\_forecast\_function(weights, arima\_forecast, xgb\_forecast)

calculate\_rmse <- function(actual, forecast) {

sqrt(mean((actual - forecast)^2))}

models <- list(

ARIMA = arima\_forecast,

ETS = ets\_forecast,

MLP = mlp\_forecast,

Prophet = prophet\_forecast,

XGBoost = xgb\_forecast,

ARIMA\_ETS = arima\_ets\_forecast,

ARIMA\_MLP = arima\_mlp\_forecast,

ARIMA\_Prophet = arima\_prophet\_forecast,

ARIMA\_XGBoost = arima\_xgb\_forecast)

for (model\_name in names(models)) {

rmse <- calculate\_rmse(testing\_data, models[[model\_name]])

results <- rbind(results, data.frame(Horizon = h, Model = model\_name, RMSE = rmse))}}

print("Sensitivity Analysis Results:")

print(results)

ggplot(results, aes(x = Horizon, y = RMSE, color = Model, group = Model)) +geom\_line() +geom\_point() +labs(title = "MLI", y = "RMSE", x = "Forecast Horizon (Years)") +theme\_minimal()

**# Sensitivity Analysis for MRT UN-5 Male Mortality using Time Horizon: 3, 5, and 7 years**

set.seed(123)

MRT = c(175.72, 170.62, 165.49, 160, 153.82, 147.06, 140.41, 134.58, 129.78, 126.19, 123.84, 122.37, 121.31, 120.48, 119.73, 118.82, 117.5, 115.76, 113.16, 109.8, 105.66, 100.63, 95.058, 89.264, 83.611, 78.196, 73.4, 69.391, 66.123, 63.472, 61.309, 59.545, 57.979, 56.456, 54.91, 53.383, 51.799, 50.261, 48.754, 47.219, 45.825, 44.399, 43.022)

mrt\_ts <- ts(MRT, start = 1980, frequency = 1)

horizons <- c(3, 5, 7)

results <- data.frame(Horizon = integer(), Model = character(), RMSE = numeric())

for (h in horizons) {

training\_data <- window(mrt\_ts, start = 1980, end = 2017)

testing\_data <- window(mrt\_ts, start = 2018, end = 2017 + h)

arima\_model <- auto.arima(training\_data)

arima\_forecast <- forecast(arima\_model, h = length(testing\_data))$mean

ets\_model <- ets(training\_data)

ets\_forecast <- forecast(ets\_model, h = length(testing\_data))$mean

mlp\_model <- mlp(training\_data)

mlp\_forecast <- forecast(mlp\_model, h = length(testing\_data))$mean

train\_df <- data.frame(

ds = seq.Date(from = as.Date("1980-01-01"), by = "year", length.out = length(training\_data)),

y = as.numeric(training\_data))

prophet\_model <- prophet(train\_df, yearly.seasonality = FALSE)

future <- data.frame(ds = seq.Date(from = max(train\_df$ds) + 1, by = "year", length.out = length(testing\_data)))

prophet\_forecast <- predict(prophet\_model, future)$yhat

lagged\_data <- data.frame(Target = training\_data[-1], Lag1 = training\_data[-length(training\_data)])

xgb\_model <- xgboost(data = as.matrix(lagged\_data$Lag1), label = lagged\_data$Target, nrounds = 100)

xgb\_forecast <- numeric(length(testing\_data))

last\_value <- training\_data[length(training\_data)]

for (i in 1:length(testing\_data)) {

xgb\_forecast[i] <- predict(xgb\_model, newdata = as.matrix(last\_value))

last\_value <- xgb\_forecast[i]}

hybrid\_forecast\_function <- function(weights, f1, f2) {

weights[1] \* as.numeric(f1) + weights[2] \* as.numeric(f2)}

optimize\_weights <- function(f1, f2, actual) {ga\_model <- ga(type = "real-valued", fitness = function(w) -sqrt(mean((actual - hybrid\_forecast\_function(w, f1, f2))^2)), lower = c(0, 0), upper = c(1, 1), popSize = 50, maxiter = 100)

return(ga\_model@solution)}

weights <- optimize\_weights(arima\_forecast, ets\_forecast, testing\_data)

arima\_ets\_forecast <- hybrid\_forecast\_function(weights, arima\_forecast, ets\_forecast)

weights <- optimize\_weights(arima\_forecast, mlp\_forecast, testing\_data)

arima\_mlp\_forecast <- hybrid\_forecast\_function(weights, arima\_forecast, mlp\_forecast)

weights <- optimize\_weights(arima\_forecast, prophet\_forecast, testing\_data)

arima\_prophet\_forecast <- hybrid\_forecast\_function(weights, arima\_forecast, prophet\_forecast)

weights <- optimize\_weights(arima\_forecast, xgb\_forecast, testing\_data)

arima\_xgb\_forecast <- hybrid\_forecast\_function(weights, arima\_forecast, xgb\_forecast)

calculate\_rmse <- function(actual, forecast) {

sqrt(mean((actual - forecast)^2))}

models <- list(

ARIMA = arima\_forecast,

ETS = ets\_forecast,

MLP = mlp\_forecast,

Prophet = prophet\_forecast,

XGBoost = xgb\_forecast,

ARIMA\_ETS = arima\_ets\_forecast,

ARIMA\_MLP = arima\_mlp\_forecast,

ARIMA\_Prophet = arima\_prophet\_forecast,

ARIMA\_XGBoost = arima\_xgb\_forecast)

for (model\_name in names(models)) {

rmse <- calculate\_rmse(testing\_data, models[[model\_name]])

results <- rbind(results, data.frame(Horizon = h, Model = model\_name, RMSE = rmse))}}

print("Sensitivity Analysis Results:")

print(results)

ggplot(results, aes(x = Horizon, y = RMSE, color = Model, group = Model)) +geom\_line() +geom\_point() +labs(title = "MRT", y = "RMSE", x = "Forecast Horizon (Years)") +theme\_minimal()

**# Sensitivity Analysis for CPV UN-5 Male Mortality using Time Horizon: 3, 5, and 7 years**

set.seed(123)

CPV = c(86.06, 84.55, 83.59, 82.68, 81.32, 79.32, 76.47, 72.82, 68.97, 65.45, 62.63, 60.73, 59.71, 59.1, 58.28, 56.7, 54.03, 50.18, 45.69, 41.2, 37.14, 33.89, 31.6, 30.2, 29.52, 29.35, 29.37, 29.38, 29.18, 28.57, 27.6, 26.37, 24.9, 23.33, 21.67, 20.08, 18.61, 17.35, 16.25, 15.33, 14.58, 13.94, 13.39)

cpv\_ts <- ts(CPV, start = 1980, frequency = 1)

horizons <- c(3, 5, 7)

results <- data.frame(Horizon = integer(), Model = character(), RMSE = numeric())

for (h in horizons) {

training\_data <- window(cpv\_ts, start = 1980, end = 2017)

testing\_data <- window(cpv\_ts, start = 2018, end = 2017 + h)

arima\_model <- auto.arima(training\_data)

arima\_forecast <- forecast(arima\_model, h = length(testing\_data))$mean

ets\_model <- ets(training\_data)

ets\_forecast <- forecast(ets\_model, h = length(testing\_data))$mean

mlp\_model <- mlp(training\_data)

mlp\_forecast <- forecast(mlp\_model, h = length(testing\_data))$mean

train\_df <- data.frame(

ds = seq.Date(from = as.Date("1980-01-01"), by = "year", length.out = length(training\_data)),

y = as.numeric(training\_data))

prophet\_model <- prophet(train\_df, yearly.seasonality = FALSE)

future <- data.frame(ds = seq.Date(from = max(train\_df$ds) + 1, by = "year", length.out = length(testing\_data)))

prophet\_forecast <- predict(prophet\_model, future)$yhat

lagged\_data <- data.frame(Target = training\_data[-1], Lag1 = training\_data[-length(training\_data)])

xgb\_model <- xgboost(data = as.matrix(lagged\_data$Lag1), label = lagged\_data$Target, nrounds = 100)

xgb\_forecast <- numeric(length(testing\_data))

last\_value <- training\_data[length(training\_data)]

for (i in 1:length(testing\_data)) {

xgb\_forecast[i] <- predict(xgb\_model, newdata = as.matrix(last\_value))

last\_value <- xgb\_forecast[i]}

hybrid\_forecast\_function <- function(weights, f1, f2) {

weights[1] \* as.numeric(f1) + weights[2] \* as.numeric(f2)}

optimize\_weights <- function(f1, f2, actual) {ga\_model <- ga(type = "real-valued", fitness = function(w) -sqrt(mean((actual - hybrid\_forecast\_function(w, f1, f2))^2)), lower = c(0, 0), upper = c(1, 1), popSize = 50, maxiter = 100)

return(ga\_model@solution)}

weights <- optimize\_weights(arima\_forecast, ets\_forecast, testing\_data)

arima\_ets\_forecast <- hybrid\_forecast\_function(weights, arima\_forecast, ets\_forecast)

weights <- optimize\_weights(arima\_forecast, mlp\_forecast, testing\_data)

arima\_mlp\_forecast <- hybrid\_forecast\_function(weights, arima\_forecast, mlp\_forecast)

weights <- optimize\_weights(arima\_forecast, prophet\_forecast, testing\_data)

arima\_prophet\_forecast <- hybrid\_forecast\_function(weights, arima\_forecast, prophet\_forecast)

weights <- optimize\_weights(arima\_forecast, xgb\_forecast, testing\_data)

arima\_xgb\_forecast <- hybrid\_forecast\_function(weights, arima\_forecast, xgb\_forecast)

calculate\_rmse <- function(actual, forecast) {

sqrt(mean((actual - forecast)^2))}

models <- list(

ARIMA = arima\_forecast,

ETS = ets\_forecast,

MLP = mlp\_forecast,

Prophet = prophet\_forecast,

XGBoost = xgb\_forecast,

ARIMA\_ETS = arima\_ets\_forecast,

ARIMA\_MLP = arima\_mlp\_forecast,

ARIMA\_Prophet = arima\_prophet\_forecast,

ARIMA\_XGBoost = arima\_xgb\_forecast)

for (model\_name in names(models)) {

rmse <- calculate\_rmse(testing\_data, models[[model\_name]])

results <- rbind(results, data.frame(Horizon = h, Model = model\_name, RMSE = rmse))}}

print("Sensitivity Analysis Results:")

print(results)

ggplot(results, aes(x = Horizon, y = RMSE, color = Model, group = Model)) +geom\_line() +geom\_point() +labs(title = "CPV", y = "RMSE", x = "Forecast Horizon (Years)") +theme\_minimal()

**# Sensitivity Analysis for TCD UN-5 Male Mortality using Time Horizon: 3, 5, and 7 years**

set.seed(123)

TCD = c(249.96, 247.91, 245.44, 242.78, 239.9, 236.96, 234.08, 231.21, 228.28, 224.99, 221.77, 218.58, 215.61, 212.64, 209.84, 207.23, 204.53, 202.01, 199.12, 196.02, 192.78, 189.26, 185.93, 182.59, 179, 175.35, 171.4, 167.61, 163.66, 159.59, 155.43, 151.39, 147.39, 143.58, 139.72, 135.94, 132.23, 128.24, 124.19, 120.39, 116.63, 112.88, 109.33)

tcd\_ts <- ts(TCD, start = 1980, frequency = 1)

horizons <- c(3, 5, 7)

results <- data.frame(Horizon = integer(), Model = character(), RMSE = numeric())

for (h in horizons) {

training\_data <- window(tcd\_ts, start = 1980, end = 2017)

testing\_data <- window(tcd\_ts, start = 2018, end = 2017 + h)

arima\_model <- auto.arima(training\_data)

arima\_forecast <- forecast(arima\_model, h = length(testing\_data))$mean

ets\_model <- ets(training\_data)

ets\_forecast <- forecast(ets\_model, h = length(testing\_data))$mean

mlp\_model <- mlp(training\_data)

mlp\_forecast <- forecast(mlp\_model, h = length(testing\_data))$mean

train\_df <- data.frame(

ds = seq.Date(from = as.Date("1980-01-01"), by = "year", length.out = length(training\_data)),

y = as.numeric(training\_data))

prophet\_model <- prophet(train\_df, yearly.seasonality = FALSE)

future <- data.frame(ds = seq.Date(from = max(train\_df$ds) + 1, by = "year", length.out = length(testing\_data)))

prophet\_forecast <- predict(prophet\_model, future)$yhat

lagged\_data <- data.frame(Target = training\_data[-1], Lag1 = training\_data[-length(training\_data)])

xgb\_model <- xgboost(data = as.matrix(lagged\_data$Lag1), label = lagged\_data$Target, nrounds = 100)

xgb\_forecast <- numeric(length(testing\_data))

last\_value <- training\_data[length(training\_data)]

for (i in 1:length(testing\_data)) {

xgb\_forecast[i] <- predict(xgb\_model, newdata = as.matrix(last\_value))

last\_value <- xgb\_forecast[i]}

hybrid\_forecast\_function <- function(weights, f1, f2) {

weights[1] \* as.numeric(f1) + weights[2] \* as.numeric(f2)}

optimize\_weights <- function(f1, f2, actual) {ga\_model <- ga(type = "real-valued", fitness = function(w) -sqrt(mean((actual - hybrid\_forecast\_function(w, f1, f2))^2)), lower = c(0, 0), upper = c(1, 1), popSize = 50, maxiter = 100)

return(ga\_model@solution)}

weights <- optimize\_weights(arima\_forecast, ets\_forecast, testing\_data)

arima\_ets\_forecast <- hybrid\_forecast\_function(weights, arima\_forecast, ets\_forecast)

weights <- optimize\_weights(arima\_forecast, mlp\_forecast, testing\_data)

arima\_mlp\_forecast <- hybrid\_forecast\_function(weights, arima\_forecast, mlp\_forecast)

weights <- optimize\_weights(arima\_forecast, prophet\_forecast, testing\_data)

arima\_prophet\_forecast <- hybrid\_forecast\_function(weights, arima\_forecast, prophet\_forecast)

weights <- optimize\_weights(arima\_forecast, xgb\_forecast, testing\_data)

arima\_xgb\_forecast <- hybrid\_forecast\_function(weights, arima\_forecast, xgb\_forecast)

calculate\_rmse <- function(actual, forecast) {

sqrt(mean((actual - forecast)^2))}

models <- list(

ARIMA = arima\_forecast,

ETS = ets\_forecast,

MLP = mlp\_forecast,

Prophet = prophet\_forecast,

XGBoost = xgb\_forecast,

ARIMA\_ETS = arima\_ets\_forecast,

ARIMA\_MLP = arima\_mlp\_forecast,

ARIMA\_Prophet = arima\_prophet\_forecast,

ARIMA\_XGBoost = arima\_xgb\_forecast)

for (model\_name in names(models)) {

rmse <- calculate\_rmse(testing\_data, models[[model\_name]])

results <- rbind(results, data.frame(Horizon = h, Model = model\_name, RMSE = rmse))}}

print("Sensitivity Analysis Results:")

print(results)

ggplot(results, aes(x = Horizon, y = RMSE, color = Model, group = Model)) +geom\_line() +geom\_point() +labs(title = "TCD", y = "RMSE", x = "Forecast Horizon (Years)") +theme\_minimal()

**# Sensitivity Analysis for CIV UN-5 Male Mortality using Time Horizon: 3, 5, and 7 years**

set.seed(123)

CIV = c(181.68, 177.17, 173.54, 170.72, 168.58, 166.68, 165.37, 164.74, 164.68, 164.99, 165.31, 165.47, 165.35, 165.32, 165, 164.22, 162.93, 161.05, 158.62, 155.67, 152.55, 149.01, 145.28, 141.51, 137.54, 133.48, 129.6, 125.59, 121.49, 116.87, 112.86, 109.34, 106, 102.43, 98.739, 95.379, 92.666, 90.033, 87.174, 84.164, 81.657, 79.277, 76.893)

civ\_ts <- ts(CIV, start = 1980, frequency = 1)

horizons <- c(3, 5, 7)

results <- data.frame(Horizon = integer(), Model = character(), RMSE = numeric())

for (h in horizons) {

training\_data <- window(civ\_ts, start = 1980, end = 2017)

testing\_data <- window(civ\_ts, start = 2018, end = 2017 + h)

arima\_model <- auto.arima(training\_data)

arima\_forecast <- forecast(arima\_model, h = length(testing\_data))$mean

ets\_model <- ets(training\_data)

ets\_forecast <- forecast(ets\_model, h = length(testing\_data))$mean

mlp\_model <- mlp(training\_data)

mlp\_forecast <- forecast(mlp\_model, h = length(testing\_data))$mean

train\_df <- data.frame(

ds = seq.Date(from = as.Date("1980-01-01"), by = "year", length.out = length(training\_data)),

y = as.numeric(training\_data))

prophet\_model <- prophet(train\_df, yearly.seasonality = FALSE)

future <- data.frame(ds = seq.Date(from = max(train\_df$ds) + 1, by = "year", length.out = length(testing\_data)))

prophet\_forecast <- predict(prophet\_model, future)$yhat

lagged\_data <- data.frame(Target = training\_data[-1], Lag1 = training\_data[-length(training\_data)])

xgb\_model <- xgboost(data = as.matrix(lagged\_data$Lag1), label = lagged\_data$Target, nrounds = 100)

xgb\_forecast <- numeric(length(testing\_data))

last\_value <- training\_data[length(training\_data)]

for (i in 1:length(testing\_data)) {

xgb\_forecast[i] <- predict(xgb\_model, newdata = as.matrix(last\_value))

last\_value <- xgb\_forecast[i]}

hybrid\_forecast\_function <- function(weights, f1, f2) {

weights[1] \* as.numeric(f1) + weights[2] \* as.numeric(f2)}

optimize\_weights <- function(f1, f2, actual) {ga\_model <- ga(type = "real-valued", fitness = function(w) -sqrt(mean((actual - hybrid\_forecast\_function(w, f1, f2))^2)), lower = c(0, 0), upper = c(1, 1), popSize = 50, maxiter = 100)

return(ga\_model@solution)}

weights <- optimize\_weights(arima\_forecast, ets\_forecast, testing\_data)

arima\_ets\_forecast <- hybrid\_forecast\_function(weights, arima\_forecast, ets\_forecast)

weights <- optimize\_weights(arima\_forecast, mlp\_forecast, testing\_data)

arima\_mlp\_forecast <- hybrid\_forecast\_function(weights, arima\_forecast, mlp\_forecast)

weights <- optimize\_weights(arima\_forecast, prophet\_forecast, testing\_data)

arima\_prophet\_forecast <- hybrid\_forecast\_function(weights, arima\_forecast, prophet\_forecast)

weights <- optimize\_weights(arima\_forecast, xgb\_forecast, testing\_data)

arima\_xgb\_forecast <- hybrid\_forecast\_function(weights, arima\_forecast, xgb\_forecast)

calculate\_rmse <- function(actual, forecast) {

sqrt(mean((actual - forecast)^2))}

models <- list(

ARIMA = arima\_forecast,

ETS = ets\_forecast,

MLP = mlp\_forecast,

Prophet = prophet\_forecast,

XGBoost = xgb\_forecast,

ARIMA\_ETS = arima\_ets\_forecast,

ARIMA\_MLP = arima\_mlp\_forecast,

ARIMA\_Prophet = arima\_prophet\_forecast,

ARIMA\_XGBoost = arima\_xgb\_forecast)

for (model\_name in names(models)) {

rmse <- calculate\_rmse(testing\_data, models[[model\_name]])

results <- rbind(results, data.frame(Horizon = h, Model = model\_name, RMSE = rmse))}}

print("Sensitivity Analysis Results:")

print(results)

ggplot(results, aes(x = Horizon, y = RMSE, color = Model, group = Model)) +geom\_line() +geom\_point() +labs(title = "CIV", y = "RMSE", x = "Forecast Horizon (Years)") +theme\_minimal()

**# Sensitivity Analysis for GNQ UN-5 Male Mortality using Time Horizon: 3, 5, and 7 years**

set.seed(123)

GNQ = c(210.40, 210.4, 210.4, 206.2, 201.85, 198.09, 195.07, 192.62, 190.47, 188.77, 187.05, 185.44, 183.9, 182.51, 181.08, 179.5, 177.39, 174.89, 171.44, 167.91, 163.84, 159.69, 155.21, 150.74, 146.19, 141.4, 136.61, 131.77, 127.04, 122.63, 118.57, 114.55, 110.62, 106.86, 103.37, 99.931, 96.594, 93.446, 90.2, 87.159, 84.438, 81.857, 79.273)

gnq\_ts <- ts(GNQ, start = 1980, frequency = 1)

horizons <- c(3, 5, 7)

results <- data.frame(Horizon = integer(), Model = character(), RMSE = numeric())

for (h in horizons) {

training\_data <- window(gnq\_ts, start = 1980, end = 2017)

testing\_data <- window(gnq\_ts, start = 2018, end = 2017 + h)

arima\_model <- auto.arima(training\_data)

arima\_forecast <- forecast(arima\_model, h = length(testing\_data))$mean

ets\_model <- ets(training\_data)

ets\_forecast <- forecast(ets\_model, h = length(testing\_data))$mean

mlp\_model <- mlp(training\_data)

mlp\_forecast <- forecast(mlp\_model, h = length(testing\_data))$mean

train\_df <- data.frame(

ds = seq.Date(from = as.Date("1980-01-01"), by = "year", length.out = length(training\_data)),

y = as.numeric(training\_data))

prophet\_model <- prophet(train\_df, yearly.seasonality = FALSE)

future <- data.frame(ds = seq.Date(from = max(train\_df$ds) + 1, by = "year", length.out = length(testing\_data)))

prophet\_forecast <- predict(prophet\_model, future)$yhat

lagged\_data <- data.frame(Target = training\_data[-1], Lag1 = training\_data[-length(training\_data)])

xgb\_model <- xgboost(data = as.matrix(lagged\_data$Lag1), label = lagged\_data$Target, nrounds = 100)

xgb\_forecast <- numeric(length(testing\_data))

last\_value <- training\_data[length(training\_data)]

for (i in 1:length(testing\_data)) {

xgb\_forecast[i] <- predict(xgb\_model, newdata = as.matrix(last\_value))

last\_value <- xgb\_forecast[i]}

hybrid\_forecast\_function <- function(weights, f1, f2) {

weights[1] \* as.numeric(f1) + weights[2] \* as.numeric(f2)}

optimize\_weights <- function(f1, f2, actual) {ga\_model <- ga(type = "real-valued", fitness = function(w) -sqrt(mean((actual - hybrid\_forecast\_function(w, f1, f2))^2)), lower = c(0, 0), upper = c(1, 1), popSize = 50, maxiter = 100)

return(ga\_model@solution)}

weights <- optimize\_weights(arima\_forecast, ets\_forecast, testing\_data)

arima\_ets\_forecast <- hybrid\_forecast\_function(weights, arima\_forecast, ets\_forecast)

weights <- optimize\_weights(arima\_forecast, mlp\_forecast, testing\_data)

arima\_mlp\_forecast <- hybrid\_forecast\_function(weights, arima\_forecast, mlp\_forecast)

weights <- optimize\_weights(arima\_forecast, prophet\_forecast, testing\_data)

arima\_prophet\_forecast <- hybrid\_forecast\_function(weights, arima\_forecast, prophet\_forecast)

weights <- optimize\_weights(arima\_forecast, xgb\_forecast, testing\_data)

arima\_xgb\_forecast <- hybrid\_forecast\_function(weights, arima\_forecast, xgb\_forecast)

calculate\_rmse <- function(actual, forecast) {

sqrt(mean((actual - forecast)^2))}

models <- list(

ARIMA = arima\_forecast,

ETS = ets\_forecast,

MLP = mlp\_forecast,

Prophet = prophet\_forecast,

XGBoost = xgb\_forecast,

ARIMA\_ETS = arima\_ets\_forecast,

ARIMA\_MLP = arima\_mlp\_forecast,

ARIMA\_Prophet = arima\_prophet\_forecast,

ARIMA\_XGBoost = arima\_xgb\_forecast)

for (model\_name in names(models)) {

rmse <- calculate\_rmse(testing\_data, models[[model\_name]])

results <- rbind(results, data.frame(Horizon = h, Model = model\_name, RMSE = rmse))}}

print("Sensitivity Analysis Results:")

print(results)

ggplot(results, aes(x = Horizon, y = RMSE, color = Model, group = Model)) +geom\_line() +geom\_point() +labs(title = "GNQ", y = "RMSE", x = "Forecast Horizon (Years)") +theme\_minimal()

**# Sensitivity Analysis for GMB UN-5 Male Mortality using Time Horizon: 3, 5, and 7 years**

set.seed(123)

GMB = c(240.36, 233.75, 227.15, 220.51, 213.83, 207.16, 200.42, 194.03, 187.83, 181.77, 175.79, 169.85, 163.82, 157.85, 151.91, 146.3, 140.8, 135.37, 130.17, 125.16, 120.2, 115.39, 110.75, 106.26, 101.89, 97.669, 93.579, 89.685, 86.001, 82.434, 79.091, 75.963, 72.905, 70.031, 67.323, 64.698, 62.246, 59.855, 57.645, 55.583, 53.628, 51.819, 50.137)

gmb\_ts <- ts(GMB, start = 1980, frequency = 1)

horizons <- c(3, 5, 7)

results <- data.frame(Horizon = integer(), Model = character(), RMSE = numeric())

for (h in horizons) {

training\_data <- window(gmb\_ts, start = 1980, end = 2017)

testing\_data <- window(gmb\_ts, start = 2018, end = 2017 + h)

arima\_model <- auto.arima(training\_data)

arima\_forecast <- forecast(arima\_model, h = length(testing\_data))$mean

ets\_model <- ets(training\_data)

ets\_forecast <- forecast(ets\_model, h = length(testing\_data))$mean

mlp\_model <- mlp(training\_data)

mlp\_forecast <- forecast(mlp\_model, h = length(testing\_data))$mean

train\_df <- data.frame(

ds = seq.Date(from = as.Date("1980-01-01"), by = "year", length.out = length(training\_data)),

y = as.numeric(training\_data))

prophet\_model <- prophet(train\_df, yearly.seasonality = FALSE)

future <- data.frame(ds = seq.Date(from = max(train\_df$ds) + 1, by = "year", length.out = length(testing\_data)))

prophet\_forecast <- predict(prophet\_model, future)$yhat

lagged\_data <- data.frame(Target = training\_data[-1], Lag1 = training\_data[-length(training\_data)])

xgb\_model <- xgboost(data = as.matrix(lagged\_data$Lag1), label = lagged\_data$Target, nrounds = 100)

xgb\_forecast <- numeric(length(testing\_data))

last\_value <- training\_data[length(training\_data)]

for (i in 1:length(testing\_data)) {

xgb\_forecast[i] <- predict(xgb\_model, newdata = as.matrix(last\_value))

last\_value <- xgb\_forecast[i]}

hybrid\_forecast\_function <- function(weights, f1, f2) {

weights[1] \* as.numeric(f1) + weights[2] \* as.numeric(f2)}

optimize\_weights <- function(f1, f2, actual) {ga\_model <- ga(type = "real-valued", fitness = function(w) -sqrt(mean((actual - hybrid\_forecast\_function(w, f1, f2))^2)), lower = c(0, 0), upper = c(1, 1), popSize = 50, maxiter = 100)

return(ga\_model@solution)}

weights <- optimize\_weights(arima\_forecast, ets\_forecast, testing\_data)

arima\_ets\_forecast <- hybrid\_forecast\_function(weights, arima\_forecast, ets\_forecast)

weights <- optimize\_weights(arima\_forecast, mlp\_forecast, testing\_data)

arima\_mlp\_forecast <- hybrid\_forecast\_function(weights, arima\_forecast, mlp\_forecast)

weights <- optimize\_weights(arima\_forecast, prophet\_forecast, testing\_data)

arima\_prophet\_forecast <- hybrid\_forecast\_function(weights, arima\_forecast, prophet\_forecast)

weights <- optimize\_weights(arima\_forecast, xgb\_forecast, testing\_data)

arima\_xgb\_forecast <- hybrid\_forecast\_function(weights, arima\_forecast, xgb\_forecast)

calculate\_rmse <- function(actual, forecast) {

sqrt(mean((actual - forecast)^2))}

models <- list(

ARIMA = arima\_forecast,

ETS = ets\_forecast,

MLP = mlp\_forecast,

Prophet = prophet\_forecast,

XGBoost = xgb\_forecast,

ARIMA\_ETS = arima\_ets\_forecast,

ARIMA\_MLP = arima\_mlp\_forecast,

ARIMA\_Prophet = arima\_prophet\_forecast,

ARIMA\_XGBoost = arima\_xgb\_forecast)

for (model\_name in names(models)) {

rmse <- calculate\_rmse(testing\_data, models[[model\_name]])

results <- rbind(results, data.frame(Horizon = h, Model = model\_name, RMSE = rmse))}}

print("Sensitivity Analysis Results:")

print(results)

ggplot(results, aes(x = Horizon, y = RMSE, color = Model, group = Model)) +geom\_line() +geom\_point() +labs(title = "GMB", y = "RMSE", x = "Forecast Horizon (Years)") +theme\_minimal()

**# Sensitivity Analysis for GHA UN-5 Male Mortality using Time Horizon: 3, 5, and 7 years**

set.seed(123)

GHA = c(175.08, 173.71, 172.37, 170.44, 167.51, 163.42, 158.24, 152.45, 146.45, 140.61, 135.5, 131.35, 128.09, 125.85, 124.09, 122.43, 120.47, 118.1, 115.07, 111.55, 107.61, 103.46, 99.584, 96.236, 93.358, 90.743, 88.311, 85.723, 82.855, 79.624, 76.175, 72.679, 69.287, 66.103, 63.205, 60.61, 58.142, 55.832, 53.737, 51.854, 49.954, 48.205, 46.67)

gha\_ts <- ts(GHA, start = 1980, frequency = 1)

horizons <- c(3, 5, 7)

results <- data.frame(Horizon = integer(), Model = character(), RMSE = numeric())

for (h in horizons) {

training\_data <- window(gha\_ts, start = 1980, end = 2017)

testing\_data <- window(gha\_ts, start = 2018, end = 2017 + h)

arima\_model <- auto.arima(training\_data)

arima\_forecast <- forecast(arima\_model, h = length(testing\_data))$mean

ets\_model <- ets(training\_data)

ets\_forecast <- forecast(ets\_model, h = length(testing\_data))$mean

mlp\_model <- mlp(training\_data)

mlp\_forecast <- forecast(mlp\_model, h = length(testing\_data))$mean

train\_df <- data.frame(

ds = seq.Date(from = as.Date("1980-01-01"), by = "year", length.out = length(training\_data)),

y = as.numeric(training\_data))

prophet\_model <- prophet(train\_df, yearly.seasonality = FALSE)

future <- data.frame(ds = seq.Date(from = max(train\_df$ds) + 1, by = "year", length.out = length(testing\_data)))

prophet\_forecast <- predict(prophet\_model, future)$yhat

lagged\_data <- data.frame(Target = training\_data[-1], Lag1 = training\_data[-length(training\_data)])

xgb\_model <- xgboost(data = as.matrix(lagged\_data$Lag1), label = lagged\_data$Target, nrounds = 100)

xgb\_forecast <- numeric(length(testing\_data))

last\_value <- training\_data[length(training\_data)]

for (i in 1:length(testing\_data)) {

xgb\_forecast[i] <- predict(xgb\_model, newdata = as.matrix(last\_value))

last\_value <- xgb\_forecast[i]}

hybrid\_forecast\_function <- function(weights, f1, f2) {

weights[1] \* as.numeric(f1) + weights[2] \* as.numeric(f2)}

optimize\_weights <- function(f1, f2, actual) {ga\_model <- ga(type = "real-valued", fitness = function(w) -sqrt(mean((actual - hybrid\_forecast\_function(w, f1, f2))^2)), lower = c(0, 0), upper = c(1, 1), popSize = 50, maxiter = 100)

return(ga\_model@solution)}

weights <- optimize\_weights(arima\_forecast, ets\_forecast, testing\_data)

arima\_ets\_forecast <- hybrid\_forecast\_function(weights, arima\_forecast, ets\_forecast)

weights <- optimize\_weights(arima\_forecast, mlp\_forecast, testing\_data)

arima\_mlp\_forecast <- hybrid\_forecast\_function(weights, arima\_forecast, mlp\_forecast)

weights <- optimize\_weights(arima\_forecast, prophet\_forecast, testing\_data)

arima\_prophet\_forecast <- hybrid\_forecast\_function(weights, arima\_forecast, prophet\_forecast)

weights <- optimize\_weights(arima\_forecast, xgb\_forecast, testing\_data)

arima\_xgb\_forecast <- hybrid\_forecast\_function(weights, arima\_forecast, xgb\_forecast)

calculate\_rmse <- function(actual, forecast) {

sqrt(mean((actual - forecast)^2))}

models <- list(

ARIMA = arima\_forecast,

ETS = ets\_forecast,

MLP = mlp\_forecast,

Prophet = prophet\_forecast,

XGBoost = xgb\_forecast,

ARIMA\_ETS = arima\_ets\_forecast,

ARIMA\_MLP = arima\_mlp\_forecast,

ARIMA\_Prophet = arima\_prophet\_forecast,

ARIMA\_XGBoost = arima\_xgb\_forecast)

for (model\_name in names(models)) {

rmse <- calculate\_rmse(testing\_data, models[[model\_name]])

results <- rbind(results, data.frame(Horizon = h, Model = model\_name, RMSE = rmse))}}

print("Sensitivity Analysis Results:")

print(results)

ggplot(results, aes(x = Horizon, y = RMSE, color = Model, group = Model)) +geom\_line() +geom\_point() +labs(title = "GHA", y = "RMSE", x = "Forecast Horizon (Years)") +theme\_minimal()

**# Sensitivity Analysis for NER UN-5 Male Mortality using Time Horizon: 3, 5, and 7 years**

set.seed(123)

NER = c(324.44, 324.26, 326.35, 330.26, 334.87, 339.29, 342.28, 343.1, 341.85, 338.55, 333.72, 327.93, 319.82, 309.77, 297.67, 284.64, 272.09, 260.7, 250.34, 240.77, 231.36, 221.43, 210.15, 198.28, 186.55, 175.33, 165.25, 156.58, 149.47, 143.91, 139.66, 136.56, 134.29, 132.54, 131.04, 129.71, 128.7, 127.5, 126.42, 125.42, 124.27, 122.71, 120.54)

ner\_ts <- ts(NER, start = 1980, frequency = 1)

horizons <- c(3, 5, 7)

results <- data.frame(Horizon = integer(), Model = character(), RMSE = numeric())

for (h in horizons) {

training\_data <- window(ner\_ts, start = 1980, end = 2017)

testing\_data <- window(ner\_ts, start = 2018, end = 2017 + h)

arima\_model <- auto.arima(training\_data)

arima\_forecast <- forecast(arima\_model, h = length(testing\_data))$mean

ets\_model <- ets(training\_data)

ets\_forecast <- forecast(ets\_model, h = length(testing\_data))$mean

mlp\_model <- mlp(training\_data)

mlp\_forecast <- forecast(mlp\_model, h = length(testing\_data))$mean

train\_df <- data.frame(

ds = seq.Date(from = as.Date("1980-01-01"), by = "year", length.out = length(training\_data)),

y = as.numeric(training\_data))

prophet\_model <- prophet(train\_df, yearly.seasonality = FALSE)

future <- data.frame(ds = seq.Date(from = max(train\_df$ds) + 1, by = "year", length.out = length(testing\_data)))

prophet\_forecast <- predict(prophet\_model, future)$yhat

lagged\_data <- data.frame(Target = training\_data[-1], Lag1 = training\_data[-length(training\_data)])

xgb\_model <- xgboost(data = as.matrix(lagged\_data$Lag1), label = lagged\_data$Target, nrounds = 100)

xgb\_forecast <- numeric(length(testing\_data))

last\_value <- training\_data[length(training\_data)]

for (i in 1:length(testing\_data)) {

xgb\_forecast[i] <- predict(xgb\_model, newdata = as.matrix(last\_value))

last\_value <- xgb\_forecast[i]}

hybrid\_forecast\_function <- function(weights, f1, f2) {

weights[1] \* as.numeric(f1) + weights[2] \* as.numeric(f2)}

optimize\_weights <- function(f1, f2, actual) {ga\_model <- ga(type = "real-valued", fitness = function(w) -sqrt(mean((actual - hybrid\_forecast\_function(w, f1, f2))^2)), lower = c(0, 0), upper = c(1, 1), popSize = 50, maxiter = 100)

return(ga\_model@solution)}

weights <- optimize\_weights(arima\_forecast, ets\_forecast, testing\_data)

arima\_ets\_forecast <- hybrid\_forecast\_function(weights, arima\_forecast, ets\_forecast)

weights <- optimize\_weights(arima\_forecast, mlp\_forecast, testing\_data)

arima\_mlp\_forecast <- hybrid\_forecast\_function(weights, arima\_forecast, mlp\_forecast)

weights <- optimize\_weights(arima\_forecast, prophet\_forecast, testing\_data)

arima\_prophet\_forecast <- hybrid\_forecast\_function(weights, arima\_forecast, prophet\_forecast)

weights <- optimize\_weights(arima\_forecast, xgb\_forecast, testing\_data)

arima\_xgb\_forecast <- hybrid\_forecast\_function(weights, arima\_forecast, xgb\_forecast)

calculate\_rmse <- function(actual, forecast) {

sqrt(mean((actual - forecast)^2))}

models <- list(

ARIMA = arima\_forecast,

ETS = ets\_forecast,

MLP = mlp\_forecast,

Prophet = prophet\_forecast,

XGBoost = xgb\_forecast,

ARIMA\_ETS = arima\_ets\_forecast,

ARIMA\_MLP = arima\_mlp\_forecast,

ARIMA\_Prophet = arima\_prophet\_forecast,

ARIMA\_XGBoost = arima\_xgb\_forecast)

for (model\_name in names(models)) {

rmse <- calculate\_rmse(testing\_data, models[[model\_name]])

results <- rbind(results, data.frame(Horizon = h, Model = model\_name, RMSE = rmse))}}

print("Sensitivity Analysis Results:")

print(results)

ggplot(results, aes(x = Horizon, y = RMSE, color = Model, group = Model)) +geom\_line() +geom\_point() +labs(title = "NER", y = "RMSE", x = "Forecast Horizon (Years)") +theme\_minimal()

**# Sensitivity Analysis for NGA UN-5 Male Mortality using Time Horizon: 3, 5, and 7 years**

set.seed(123)

NGA = c(223.14, 219.95, 217.96, 217.03, 217.09, 217.75, 218.78, 219.94, 220.46, 220.63, 220.41, 219.69, 218.6, 217.13, 215.43, 213.24, 210.34, 206.43, 202.09, 196.74, 190.96, 185.39, 179.54, 173.98, 168.45, 162.98, 158.07, 153.31, 149.1, 145.54, 142.56, 140.04, 137.88, 136.07, 134.46, 132.85, 131.03, 128.7, 126.25, 123.14, 120.11, 116.75, 113.24)

nga\_ts <- ts(NGA, start = 1980, frequency = 1)

horizons <- c(3, 5, 7)

results <- data.frame(Horizon = integer(), Model = character(), RMSE = numeric())

for (h in horizons) {

training\_data <- window(nga\_ts, start = 1980, end = 2017)

testing\_data <- window(nga\_ts, start = 2018, end = 2017 + h)

arima\_model <- auto.arima(training\_data)

arima\_forecast <- forecast(arima\_model, h = length(testing\_data))$mean

ets\_model <- ets(training\_data)

ets\_forecast <- forecast(ets\_model, h = length(testing\_data))$mean

mlp\_model <- mlp(training\_data)

mlp\_forecast <- forecast(mlp\_model, h = length(testing\_data))$mean

train\_df <- data.frame(

ds = seq.Date(from = as.Date("1980-01-01"), by = "year", length.out = length(training\_data)),

y = as.numeric(training\_data))

prophet\_model <- prophet(train\_df, yearly.seasonality = FALSE)

future <- data.frame(ds = seq.Date(from = max(train\_df$ds) + 1, by = "year", length.out = length(testing\_data)))

prophet\_forecast <- predict(prophet\_model, future)$yhat

lagged\_data <- data.frame(Target = training\_data[-1], Lag1 = training\_data[-length(training\_data)])

xgb\_model <- xgboost(data = as.matrix(lagged\_data$Lag1), label = lagged\_data$Target, nrounds = 100)

xgb\_forecast <- numeric(length(testing\_data))

last\_value <- training\_data[length(training\_data)]

for (i in 1:length(testing\_data)) {

xgb\_forecast[i] <- predict(xgb\_model, newdata = as.matrix(last\_value))

last\_value <- xgb\_forecast[i]}

hybrid\_forecast\_function <- function(weights, f1, f2) {

weights[1] \* as.numeric(f1) + weights[2] \* as.numeric(f2)}

optimize\_weights <- function(f1, f2, actual) {ga\_model <- ga(type = "real-valued", fitness = function(w) -sqrt(mean((actual - hybrid\_forecast\_function(w, f1, f2))^2)), lower = c(0, 0), upper = c(1, 1), popSize = 50, maxiter = 100)

return(ga\_model@solution)}

weights <- optimize\_weights(arima\_forecast, ets\_forecast, testing\_data)

arima\_ets\_forecast <- hybrid\_forecast\_function(weights, arima\_forecast, ets\_forecast)

weights <- optimize\_weights(arima\_forecast, mlp\_forecast, testing\_data)

arima\_mlp\_forecast <- hybrid\_forecast\_function(weights, arima\_forecast, mlp\_forecast)

weights <- optimize\_weights(arima\_forecast, prophet\_forecast, testing\_data)

arima\_prophet\_forecast <- hybrid\_forecast\_function(weights, arima\_forecast, prophet\_forecast)

weights <- optimize\_weights(arima\_forecast, xgb\_forecast, testing\_data)

arima\_xgb\_forecast <- hybrid\_forecast\_function(weights, arima\_forecast, xgb\_forecast)

calculate\_rmse <- function(actual, forecast) {

sqrt(mean((actual - forecast)^2))}

models <- list(

ARIMA = arima\_forecast,

ETS = ets\_forecast,

MLP = mlp\_forecast,

Prophet = prophet\_forecast,

XGBoost = xgb\_forecast,

ARIMA\_ETS = arima\_ets\_forecast,

ARIMA\_MLP = arima\_mlp\_forecast,

ARIMA\_Prophet = arima\_prophet\_forecast,

ARIMA\_XGBoost = arima\_xgb\_forecast)

for (model\_name in names(models)) {

rmse <- calculate\_rmse(testing\_data, models[[model\_name]])

results <- rbind(results, data.frame(Horizon = h, Model = model\_name, RMSE = rmse))}}

print("Sensitivity Analysis Results:")

print(results)

ggplot(results, aes(x = Horizon, y = RMSE, color = Model, group = Model)) +geom\_line() +geom\_point() +labs(title = "NGA", y = "RMSE", x = "Forecast Horizon (Years)") +theme\_minimal()

**# Sensitivity Analysis for SEN UN-5 Male Mortality using Time Horizon: 3, 5, and 7 years**

set.seed(123)

SEN = c(211.62, 205.06, 198.96, 193.02, 186.56, 179.07, 170.65, 162.2, 154.98, 149.37, 145.41, 143.37, 142.75, 143.09, 143.95, 145.06, 145.67, 145.46, 144.02, 140.99, 136, 129.15, 121.23, 112.89, 104.73, 97.138, 90.494, 84.651, 79.607, 75.247, 71.265, 67.703, 64.387, 61.287, 58.381, 55.462, 52.681, 50.216, 47.873, 45.804, 43.966, 42.293, 40.775)

sen\_ts <- ts(SEN, start = 1980, frequency = 1)

horizons <- c(3, 5, 7)

results <- data.frame(Horizon = integer(), Model = character(), RMSE = numeric())

for (h in horizons) {

training\_data <- window(sen\_ts, start = 1980, end = 2017)

testing\_data <- window(sen\_ts, start = 2018, end = 2017 + h)

arima\_model <- auto.arima(training\_data)

arima\_forecast <- forecast(arima\_model, h = length(testing\_data))$mean

ets\_model <- ets(training\_data)

ets\_forecast <- forecast(ets\_model, h = length(testing\_data))$mean

mlp\_model <- mlp(training\_data)

mlp\_forecast <- forecast(mlp\_model, h = length(testing\_data))$mean

train\_df <- data.frame(

ds = seq.Date(from = as.Date("1980-01-01"), by = "year", length.out = length(training\_data)),

y = as.numeric(training\_data))

prophet\_model <- prophet(train\_df, yearly.seasonality = FALSE)

future <- data.frame(ds = seq.Date(from = max(train\_df$ds) + 1, by = "year", length.out = length(testing\_data)))

prophet\_forecast <- predict(prophet\_model, future)$yhat

lagged\_data <- data.frame(Target = training\_data[-1], Lag1 = training\_data[-length(training\_data)])

xgb\_model <- xgboost(data = as.matrix(lagged\_data$Lag1), label = lagged\_data$Target, nrounds = 100)

xgb\_forecast <- numeric(length(testing\_data))

last\_value <- training\_data[length(training\_data)]

for (i in 1:length(testing\_data)) {

xgb\_forecast[i] <- predict(xgb\_model, newdata = as.matrix(last\_value))

last\_value <- xgb\_forecast[i]}

hybrid\_forecast\_function <- function(weights, f1, f2) {

weights[1] \* as.numeric(f1) + weights[2] \* as.numeric(f2)}

optimize\_weights <- function(f1, f2, actual) {ga\_model <- ga(type = "real-valued", fitness = function(w) -sqrt(mean((actual - hybrid\_forecast\_function(w, f1, f2))^2)), lower = c(0, 0), upper = c(1, 1), popSize = 50, maxiter = 100)

return(ga\_model@solution)}

weights <- optimize\_weights(arima\_forecast, ets\_forecast, testing\_data)

arima\_ets\_forecast <- hybrid\_forecast\_function(weights, arima\_forecast, ets\_forecast)

weights <- optimize\_weights(arima\_forecast, mlp\_forecast, testing\_data)

arima\_mlp\_forecast <- hybrid\_forecast\_function(weights, arima\_forecast, mlp\_forecast)

weights <- optimize\_weights(arima\_forecast, prophet\_forecast, testing\_data)

arima\_prophet\_forecast <- hybrid\_forecast\_function(weights, arima\_forecast, prophet\_forecast)

weights <- optimize\_weights(arima\_forecast, xgb\_forecast, testing\_data)

arima\_xgb\_forecast <- hybrid\_forecast\_function(weights, arima\_forecast, xgb\_forecast)

calculate\_rmse <- function(actual, forecast) {

sqrt(mean((actual - forecast)^2))}

models <- list(

ARIMA = arima\_forecast,

ETS = ets\_forecast,

MLP = mlp\_forecast,

Prophet = prophet\_forecast,

XGBoost = xgb\_forecast,

ARIMA\_ETS = arima\_ets\_forecast,

ARIMA\_MLP = arima\_mlp\_forecast,

ARIMA\_Prophet = arima\_prophet\_forecast,

ARIMA\_XGBoost = arima\_xgb\_forecast)

for (model\_name in names(models)) {

rmse <- calculate\_rmse(testing\_data, models[[model\_name]])

results <- rbind(results, data.frame(Horizon = h, Model = model\_name, RMSE = rmse))}}

print("Sensitivity Analysis Results:")

print(results)

ggplot(results, aes(x = Horizon, y = RMSE, color = Model, group = Model)) +geom\_line() +geom\_point() +labs(title = "SEN", y = "RMSE", x = "Forecast Horizon (Years)") +theme\_minimal()

**# Sensitivity Analysis for SLE UN-5 Male Mortality using Time Horizon: 3, 5, and 7 years**

set.seed(123)

SLE = c(300.39, 296.73, 293.46, 290.18, 286.98, 284.2, 281.51, 278.99, 276.47, 274.32, 272.19, 270.38, 268.39, 265.99, 263.04, 259.49, 255.39, 250.9, 245.93, 240.48, 235.21, 229.3, 223.42, 217.22, 210.66, 204.11, 197.26, 190.27, 183.01, 175.55, 168.57, 161.66, 155.12, 148.96, 147.04, 147.99, 133.07, 128.21, 123.65, 119.3, 115.07, 111.22, 107.51)

sle\_ts <- ts(SLE, start = 1980, frequency = 1)

horizons <- c(3, 5, 7)

results <- data.frame(Horizon = integer(), Model = character(), RMSE = numeric())

for (h in horizons) {

training\_data <- window(sle\_ts, start = 1980, end = 2017)

testing\_data <- window(sle\_ts, start = 2018, end = 2017 + h)

arima\_model <- auto.arima(training\_data)

arima\_forecast <- forecast(arima\_model, h = length(testing\_data))$mean

ets\_model <- ets(training\_data)

ets\_forecast <- forecast(ets\_model, h = length(testing\_data))$mean

mlp\_model <- mlp(training\_data)

mlp\_forecast <- forecast(mlp\_model, h = length(testing\_data))$mean

train\_df <- data.frame(

ds = seq.Date(from = as.Date("1980-01-01"), by = "year", length.out = length(training\_data)),

y = as.numeric(training\_data))

prophet\_model <- prophet(train\_df, yearly.seasonality = FALSE)

future <- data.frame(ds = seq.Date(from = max(train\_df$ds) + 1, by = "year", length.out = length(testing\_data)))

prophet\_forecast <- predict(prophet\_model, future)$yhat

lagged\_data <- data.frame(Target = training\_data[-1], Lag1 = training\_data[-length(training\_data)])

xgb\_model <- xgboost(data = as.matrix(lagged\_data$Lag1), label = lagged\_data$Target, nrounds = 100)

xgb\_forecast <- numeric(length(testing\_data))

last\_value <- training\_data[length(training\_data)]

for (i in 1:length(testing\_data)) {

xgb\_forecast[i] <- predict(xgb\_model, newdata = as.matrix(last\_value))

last\_value <- xgb\_forecast[i]}

hybrid\_forecast\_function <- function(weights, f1, f2) {

weights[1] \* as.numeric(f1) + weights[2] \* as.numeric(f2)}

optimize\_weights <- function(f1, f2, actual) {ga\_model <- ga(type = "real-valued", fitness = function(w) -sqrt(mean((actual - hybrid\_forecast\_function(w, f1, f2))^2)), lower = c(0, 0), upper = c(1, 1), popSize = 50, maxiter = 100)

return(ga\_model@solution)}

weights <- optimize\_weights(arima\_forecast, ets\_forecast, testing\_data)

arima\_ets\_forecast <- hybrid\_forecast\_function(weights, arima\_forecast, ets\_forecast)

weights <- optimize\_weights(arima\_forecast, mlp\_forecast, testing\_data)

arima\_mlp\_forecast <- hybrid\_forecast\_function(weights, arima\_forecast, mlp\_forecast)

weights <- optimize\_weights(arima\_forecast, prophet\_forecast, testing\_data)

arima\_prophet\_forecast <- hybrid\_forecast\_function(weights, arima\_forecast, prophet\_forecast)

weights <- optimize\_weights(arima\_forecast, xgb\_forecast, testing\_data)

arima\_xgb\_forecast <- hybrid\_forecast\_function(weights, arima\_forecast, xgb\_forecast)

calculate\_rmse <- function(actual, forecast) {

sqrt(mean((actual - forecast)^2))}

models <- list(

ARIMA = arima\_forecast,

ETS = ets\_forecast,

MLP = mlp\_forecast,

Prophet = prophet\_forecast,

XGBoost = xgb\_forecast,

ARIMA\_ETS = arima\_ets\_forecast,

ARIMA\_MLP = arima\_mlp\_forecast,

ARIMA\_Prophet = arima\_prophet\_forecast,

ARIMA\_XGBoost = arima\_xgb\_forecast)

for (model\_name in names(models)) {

rmse <- calculate\_rmse(testing\_data, models[[model\_name]])

results <- rbind(results, data.frame(Horizon = h, Model = model\_name, RMSE = rmse))}}

print("Sensitivity Analysis Results:")

print(results)

ggplot(results, aes(x = Horizon, y = RMSE, color = Model, group = Model)) +geom\_line() +geom\_point() +labs(title = "SLE", y = "RMSE", x = "Forecast Horizon (Years)") +theme\_minimal()

**# Sensitivity Analysis for TGO UN-5 Male Mortality using Time Horizon: 3, 5, and 7 years**

set.seed(123)

TGO = c(187.53, 183.2, 179.06, 175.14, 171.48, 168.16, 165.26, 162.69, 160.21, 158.02, 155.99, 153.96, 152, 149.99, 147.7, 145.07, 142.2, 138.9, 135.33, 131.82, 128.14, 124.36, 120.6, 116.95, 113.45, 110.11, 106.82, 103.76, 100.73, 97.691, 94.78, 92, 89.156, 86.428, 83.783, 81.165, 78.681, 76.289, 73.919, 71.605, 69.329, 67.244, 65.13)

tgo\_ts <- ts(TGO, start = 1980, frequency = 1)

horizons <- c(3, 5, 7)

results <- data.frame(Horizon = integer(), Model = character(), RMSE = numeric())

for (h in horizons) {

training\_data <- window(tgo\_ts, start = 1980, end = 2017)

testing\_data <- window(tgo\_ts, start = 2018, end = 2017 + h)

arima\_model <- auto.arima(training\_data)

arima\_forecast <- forecast(arima\_model, h = length(testing\_data))$mean

ets\_model <- ets(training\_data)

ets\_forecast <- forecast(ets\_model, h = length(testing\_data))$mean

mlp\_model <- mlp(training\_data)

mlp\_forecast <- forecast(mlp\_model, h = length(testing\_data))$mean

train\_df <- data.frame(

ds = seq.Date(from = as.Date("1980-01-01"), by = "year", length.out = length(training\_data)),

y = as.numeric(training\_data))

prophet\_model <- prophet(train\_df, yearly.seasonality = FALSE)

future <- data.frame(ds = seq.Date(from = max(train\_df$ds) + 1, by = "year", length.out = length(testing\_data)))

prophet\_forecast <- predict(prophet\_model, future)$yhat

lagged\_data <- data.frame(Target = training\_data[-1], Lag1 = training\_data[-length(training\_data)])

xgb\_model <- xgboost(data = as.matrix(lagged\_data$Lag1), label = lagged\_data$Target, nrounds = 100)

xgb\_forecast <- numeric(length(testing\_data))

last\_value <- training\_data[length(training\_data)]

for (i in 1:length(testing\_data)) {

xgb\_forecast[i] <- predict(xgb\_model, newdata = as.matrix(last\_value))

last\_value <- xgb\_forecast[i]}

hybrid\_forecast\_function <- function(weights, f1, f2) {

weights[1] \* as.numeric(f1) + weights[2] \* as.numeric(f2)}

optimize\_weights <- function(f1, f2, actual) {ga\_model <- ga(type = "real-valued", fitness = function(w) -sqrt(mean((actual - hybrid\_forecast\_function(w, f1, f2))^2)), lower = c(0, 0), upper = c(1, 1), popSize = 50, maxiter = 100)

return(ga\_model@solution)}

weights <- optimize\_weights(arima\_forecast, ets\_forecast, testing\_data)

arima\_ets\_forecast <- hybrid\_forecast\_function(weights, arima\_forecast, ets\_forecast)

weights <- optimize\_weights(arima\_forecast, mlp\_forecast, testing\_data)

arima\_mlp\_forecast <- hybrid\_forecast\_function(weights, arima\_forecast, mlp\_forecast)

weights <- optimize\_weights(arima\_forecast, prophet\_forecast, testing\_data)

arima\_prophet\_forecast <- hybrid\_forecast\_function(weights, arima\_forecast, prophet\_forecast)

weights <- optimize\_weights(arima\_forecast, xgb\_forecast, testing\_data)

arima\_xgb\_forecast <- hybrid\_forecast\_function(weights, arima\_forecast, xgb\_forecast)

calculate\_rmse <- function(actual, forecast) {

sqrt(mean((actual - forecast)^2))}

models <- list(

ARIMA = arima\_forecast,

ETS = ets\_forecast,

MLP = mlp\_forecast,

Prophet = prophet\_forecast,

XGBoost = xgb\_forecast,

ARIMA\_ETS = arima\_ets\_forecast,

ARIMA\_MLP = arima\_mlp\_forecast,

ARIMA\_Prophet = arima\_prophet\_forecast,

ARIMA\_XGBoost = arima\_xgb\_forecast)

for (model\_name in names(models)) {

rmse <- calculate\_rmse(testing\_data, models[[model\_name]])

results <- rbind(results, data.frame(Horizon = h, Model = model\_name, RMSE = rmse))}}

print("Sensitivity Analysis Results:")

print(results)

ggplot(results, aes(x = Horizon, y = RMSE, color = Model, group = Model)) +geom\_line() +geom\_point() +labs(title = "TGO", y = "RMSE", x = "Forecast Horizon (Years)") +theme\_minimal()