A black background with white text

Description automatically generated with low confidence

MSc Data Science Project

7PAM2002-0509-2024

Department of Physics, Astronomy and Mathematics

**Data Science FINAL PROJECT REPORT**

**Project Title:**

Performance analysis of Machine Learning and Deep Learning Models for Life Expectancy Prediction

**Student Name and SRN:**

Kudumula Prasad Reddy

22083068

Supervisor: Hyungrok Kim

Date Submitted: 28/08/2025

Word Count: 5138

GitHub Link: <https://github.com/PrasadReddyKudumula/Life-Expectancy-ML-DL>

# DECLARATION STATEMENT

This report is submitted in partial fulfilment of the requirement for the degree of Master of Science **in Data Science** at the University of Hertfordshire.

I have read the detailed guidance to students on academic integrity, misconduct and plagiarism information at [Assessment Offences and Academic Misconduct](https://ask.herts.ac.uk/assessment-offences-and-academic-misconduct) and understand the University process of dealing with suspected cases of academic misconduct and the possible penalties, which could include failing the project or course.

I certify that the work submitted is my own and that any material derived or quoted from published or unpublished work of other persons has been duly acknowledged. (Ref. UPR AS/C/6.1, section 7 and UPR AS/C/5, section 3.6)

I did not use human participants in my MSc Project.

I hereby give permission for the report to be made available on module websites provided the source is acknowledged.

Student Name printed: Kudumula prasad reddy

Student Name signature

Student SRN number:22083068

UNIVERSITY OF HERTFORDSHIRE

SCHOOL OF PHYSICS, ENGINEERING AND COMPUTER SCIENCE

ACKNOWLEDGEMENT

As I come close to finishing my post-graduate studies, I would like to emphasize that it has been a wonderful learning experience, and I want to express my gratitude to all the people who have supported me along the way.

I'd like to start by expressing my gratitude to Almighty God for never ceasing to inspire me with His endless blessings and for giving me the confidence and valor to move forward with assurance and self-belief.

I would like to convey my appreciation and gratitude to Hyungrok Kim, who served as my supervisor, for his constant advice and assistance in this project. I am appreciative of his constant support and his patience towards my inquisitiveness.

I would also like to express my gratitude towards all my professors at the University of Hertfordshire who helped me gain knowledge and understanding of the subjects and helped me throughout my course.

I would also like to thank my parents, and my friends for their unwavering encouragement and support, without which this would not have been possible.

# ABSTRACT

Life expectancy is a very important indicator that identifies the overall health and wellbeing of a population and represents the remaining number of years that an individual can expect to live. Its precise forecasting is paramount in shaping public health policies, resources deployment and socio-economic planning nationally and globally. This research evaluated the predictive performance of the recent deep learning models (LSTM, GRU, Transformer) compared to traditional ML regression (Gradient Boosting, Random Forest, Linear Regression) in the task of life expectancy prediction using a WHO dataset. The models are implemented and evaluated using R², MAE, MSE, and RMSE, and hyperparameters are optimized through GridSearchCV. The results clearly show that baseline machine learning models are much better compared to the sophisticated deep learning models. The most successful is Gradient Boosting with an impressive R² = 0.9407 and low error values (MAE=1.2267, RMSE=1.5375), followed closely by Random Forest (R² = 0.9227). Although LSTM exhibited good results as a deep learning model R²=0.8681), it is not as successful as the best traditional models. GRU (R² =0.3116) and Transformer (R² =-0.8805) in particular fared worse, with the latter suggesting predictions worse than random chance due to the data-hungry nature and overfitting risk of these models. This research demonstrates that in the case of structured tabular data such as the one analyzed, ensemble machine learning-based models typically outperform deep learning frameworks in computational efficiency and performance robustness, making Gradient Boosting the model of choice in the situation as far as the model should be accurate and computationally efficient.

Table of Contents

[DECLARATION STATEMENT 2](#_Toc207140801)

[ABSTRACT 4](#_Toc207140802)

[1 Introduction 6](#_Toc207140803)

[1.1 Background and Motivation 6](#_Toc207140804)

[1.2 Research Question 6](#_Toc207140805)

[1.3 Research Aim 6](#_Toc207140806)

[1.4 Research Objectives 6](#_Toc207140807)

[1.5 Feasibility of the Research 7](#_Toc207140808)

[1.6 Ethical Considerations 7](#_Toc207140809)

[2 Literature Comparison 8](#_Toc207140810)

[2.1 Identification of Gaps 9](#_Toc207140811)

[3 Methodology 10](#_Toc207140812)

[3.2 Research Methodology 10](#_Toc207140813)

[3.3 Dataset Description 12](#_Toc207140814)

[3.4 Data Preprocessing and Preparation 12](#_Toc207140815)

[3.5 Time Series Structuring 12](#_Toc207140816)

[3.6 Train-Test Split 12](#_Toc207140817)

[3.7 Model Architectures 13](#_Toc207140818)

[3.8 Tools and Techniques 13](#_Toc207140819)

[3.9 Model Evaluation 13](#_Toc207140820)

[3.10 Ethical Considerations 13](#_Toc207140821)

[3.11 Project Management 14](#_Toc207140822)

[3.12 Summary 14](#_Toc207140823)

[4 Experiment 15](#_Toc207140824)

[4.1 Overview of the Life Expectancy Dataset 15](#_Toc207140825)

[4.2 Preprocessing Stage – Handling Missing Data 16](#_Toc207140826)

[4.3 Visualization of Life Expectancy Data 17](#_Toc207140827)

[4.4 Numerical Conversion of Categorical Features 24](#_Toc207140828)

[4.5 Data Preparation for Time-Series Forecasting 25](#_Toc207140829)

[4.6 Model Implementation and Evaluation 26](#_Toc207140830)

[4.7 Critical Comparative Analysis with Past Works 28](#_Toc207140831)

[5 Conclusion 29](#_Toc207140832)

[5.1 Future Work 29](#_Toc207140833)

[Appendix 31](#_Toc207140834)

[Scatter Plots 31](#_Toc207140835)

[Code 32](#_Toc207140836)

# Introduction

## Background and Motivation

The life expectancy has been described as a major measure of the health of a population, the level of socio-economic development as well as the general standard of living in any particular country. It summarizes the efficiency of population health measures, the availability and quality of health care services and a wide diversity of demographic and environmental variables (Chen et al, 2021). Over the decades, international health organizations like the World Health Organization (WHO) have put a lot of stress on the comprehension of the tendencies of life expectancy to base policies and resources distribution. Nevertheless, the problem of life expectancy forecasting is fundamentally complicated because such influencing factors as mortality rates, education, immunization coverage, economic indicators, and access to healthcare are non-linear and multivariate (World Health Organization, 2024).

The prediction of life expectancy has traditionally been carried out by statistical models which work well in the linear assumption and are unable to extract the hidden associations and time-dependency. More recently machine learning models have been utilized to increase accuracy of prediction by modeling more complex interactions of features (Kizito et al, 2021). Nevertheless, even these models consider each data point, to a large extent in isolation, and have difficulties with time series properties.

By using machine learning and deep learning models, especially those designed for sequential data, there is potential to learn long-range dependencies and complex patterns in multivariate time series data. Such models offer strong representational power and can handle datasets with temporal characteristics, making them a promising option for predicting life expectancy (Ahmed et al, 2022).

## Research Question

Does using advanced deep learning models (like LSTM, Transformer, and GRU) gives much better predictions compared to simpler regression models (like Gradient Boosting, Random Forest, and Linear Regression)?

## Research Aim

This project aims to assess and compare the predictive performance of traditional machine learning models and advanced deep learning models on a WHO dataset of life expectancy and related features. The study will help determine whether the added computational complexity of deep learning models yields substantial gains in accuracy and practical usability.

## Research Objectives

To fulfil this aim, the research is structured around the following objectives:

* To carry out a thorough review of previously established life expectancy prediction methods that depend on machine learning and deep learning techniques.
* To handle any missing data, change categorical factors, and scale the features in order to process and analyse the data from WHO.
* To implement baseline machine learning models such as Gradient Boosting, Random Forest, and Linear Regression for life expectancy prediction.
* To design and implement advanced deep learning models like LSTM, GRU, and Transformer architectures tailored for time-series and tabular data.
* To evaluate and compare model performance using appropriate regression measures such as Mean Absolute Error (MAE), Mean Square Error (MSE), Root Mean Square Error (RMSE), and R² Score.
* To identify as well as recommend the most efficient model for accurately predicting life expectancy based on performance analysis.

## Feasibility of the Research

The research is highly feasible due to the availability of open-access WHO data and the use of Python, a powerful and widely supported programming language. Python libraries including Pandas, Scikit-learn, TensorFlow, and Matplotlib streamline the workflow from data preprocessing to visualization and model deployment on standard academic hardware.

## Ethical Considerations

This research uses anonymized, publicly available data from the World Health Organization, ensuring no privacy violations or ethical risks. No human participants are involved, and the data contains no personal identifiers. Proper citation and license compliance are maintained, aligning the research with academic and institutional ethical standards.

# Literature Comparison

Life expectancy prediction has been explored utilizing numerous ML and DL methods using various datasets and approaches to consider the complicated interaction of socioeconomic, environmental factors and health. Lipesa et al, (2023) sought to detect life expectancy based on supervised ML and applied the XGBoost algorithm to the data on 193 UN member states. With their model, which included behavioral, socioeconomic, and health characteristics, they outperformed RF and ANN with a MAE of 1.554 and a RMSE of 2.402. Their study was however limited by the fact that it did not include quality-adjusted life years (QALYs) as well as environmental factors hence there is need to have wider datasets. Equally, Ahamad et al, (2025) applied XGBoost to a WHO and UN dataset and got an impressive 97.73% accuracy, with an MSE of 2.45 and MAE of 1.03. Their technique preferred the use of feature selection via Randomized Search cross-validation and declared HIV/AIDS and income composition as important predictors, although the possibility of biases in the dataset was mentioned as a weakness.

Conversely, Dawoud et al, (2023) used a multilayer perceptron neural network on a Kaggle dataset that had 22 features and got 99.27% accuracy and an average error of 0.0034. SHAP-based feature importance indicated HIV/AIDS prevalence and socioeconomic status as essential predictors, and the research took notice of possible biases and low generalizability. Pisal et al, (2022) concentrated on Asian populations and used tree-based classifiers such as Random Forest, which obtained an accuracy of 84% using 10-fold cross-validation. They prioritized socioeconomic and health-related qualities in their research and recommended further investigation of the attributes correlation and the consideration of new datasets containing the effects of COVID-19. Rubi et al, (2021) investigated the life expectancy in Bangladesh based on the MLR and ANN, where MLR provided an accuracy of 98 percent. They managed to prove the impact of GDP and the population size, yet the research was limited by the size of datasets and the necessity to use sophisticated algorithms.

The DL methods have also become popular. Beeksma et al, (2019) applied an LSTM model to electronic medical records and obtained an accuracy of 29%, which was higher than the baseline models and physicians prognosed. Their text features allowed reducing the overestimation errors in their keyword model, but the model had issues with the specificity and interpretability of the dataset. El-Rashidy et al, (2025) proposed a multitask LSTM-GRU model to detect transformer health index and life expectancy and got an MSE of 2.543 and R 2 of 0.985. Their solution was quite promising but showed the problem of scalability and the lack of feature completeness. Das et al, (2025) evaluated 8 ML models on WHO Cleaned-Life-Exp data, and the best performance belonged to Random Forest (R 2 =0.969, RMSE = 0.179). They applied Boruta and Regularized Random Forest to feature selection, which highlighted the significance of HIV/AIDS and adult mortality, but they used only one training-test split, which was a weakness.

All of these studies prove the effectiveness of both ML and DL methods in detecting life expectancy, with more advanced models such as XGBoost, Random Forest, and LSTM performing better compared to more traditional models like Linear Regression. Hyperparameter optimization and feature selection, which are observed in Ahamad et al, (2025) and Das et al, (2025), are important to increase the accuracy of the model. Nonetheless, numerous experiments are restricted by the biases in the datasets, the absence of environmental or time dimensions, as well as the incomplete consideration of hybrid or ensemble approaches, which unite the advantages of ML and DL.

## Identification of Gaps

The existing literature does not provide extensive comparisons of ML and DL models. Such works as Lipesa et al, (2023) concentrate on XGBoost without trying such DL architecture as LSTM or Transformers, which are good at temporal patterns. Beeksma et al, (2019) and El-Rashidy et al, (2025) show the potential of LSTM and GRU but do not provide extensive comparisons to ML. Most of the studies, such as those by Pisal et al, (2022), rely on small amounts of features, ignoring environmental or lifestyle aspects. Dynamic aspects on the temporal dimension are under-researched, and the majority consider data as being stationary. The lack of reliability is caused by single training-test splits, like in Das et al, (2025), and by poor interpretability, especially in the case of DL. The given gaps in the research are bridged in this study, as LSTM, GRU, and Transformer are compared to Gradient Boosting, RF, and Linear Regression on a WHO dataset with a strong preprocessing, temporal analysis, and interpretability considered to apply the study findings to practical healthcare settings.

# Methodology

The present research proposes a clear procedure of designing a deep learning based and conventional ML based model for predicting life expectancy. The data represents 2000–2021 and covers ten countries. The procedure flow involves data preprocessing, exploratory analysis, sequence generation, model implementation and evaluation. LSTM, GRU, Transformer networks, Gradient Boosting, Random Forest, and Linear Regression were utilized as predictive models. The models were selected so some capture time trends and others provide interpretability. The chapter also indicates the methods, equipment employed, ethics followed, and how the project was carried out efficiently.

## 3.2 Research Methodology

This research was conducted in a systematic way with literature review as an initial step to get the current methods of life expectancy prediction. Once the gaps had been identified, the WHO Global Health Observatory was used to provide data on life expectancy. Preprocessing of data was done to include missing values, as well as categorical encoding of the features.

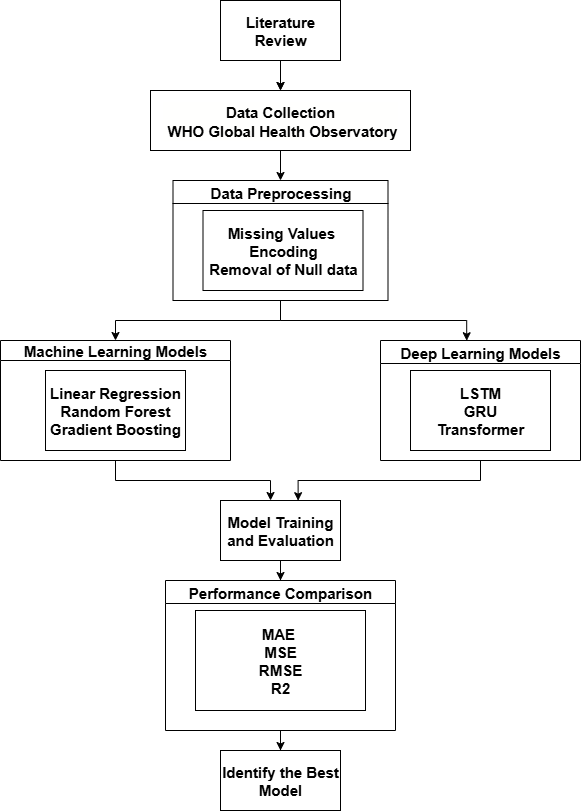


Figure Research Workflow

The modeling phase was split into two: ML and DL. In the ML stream, models were built using Scikit-learn, and hyperparameters were tuned with grid search. For deep learning models, TensorFlow/Keras was used. The dataset was converted into sequences using a sliding window approach to make it suitable for models like LSTM and GRU. Attention-based architectures such as the Transformer were implemented with encoder layers and multi-head self-attention mechanisms to learn dynamic feature importance.

Each model was trained using the same training and test data splits for consistency. Evaluation was conducted using standard regression metrics. The models were compared based on prediction accuracy and ability to generalize to unseen data.

## 3.3 Dataset Description

The dataset was retrieved from the WHO Global Health Observatory and compiled into an Excel file for analysis. It includes annual life expectancy data from ten countries spanning from 2000 to 2021. Key features in the dataset include country, year, mortality rates, immunization coverage, literacy rate, healthcare expenditure, and income levels. These variables represent a combination of categorical and numerical data, necessitating a mix of preprocessing strategies. The dataset’s chronological structure makes it particularly suitable for time-series forecasting, where past trends can inform future life expectancy predictions.

## 3.4 Data Preprocessing and Preparation

Data preprocessing started by loading dataset and examining the structure for missing values and inconsistencies. Non-numeric entries, especially in the column 'Hospital beds (per 10,000 population)', were converted to numeric using error coercion. This step converted unprocessable entries into NaN values. The missing values were then handled through statistical imputation, using the mean for numerical columns. Label encoding was used to make categorical features like name of countries into numbers that could be used in modeling. Data were also arranged according to the country and year to preserve the chronological character needed in time series analysis.

## 3.5 Time Series Structuring

The sliding window was employed to prepare data on temporal models. The input samples were based on the features of three successive years (with t-2, t-1 and t), whereas the target value was the life expectancy in the calendar year t+1. Such conversion formed patterns of data that could be studied by models in order to realize time-dependent patterns. This method was also useful in the creation of additional samples using low sample data over the years using overlapping sequence of years.

## 3.6 Train-Test Split

The temporal characteristic of the data was preserved and chronological train-test division was used. The training was done using records between the year 2000 and 2018 with the years 2019 to 2021 left to test. This would mean that the models would be tested on new information keeping the problems of future leakage to the past out of context and would also have resembled real life forecasting conditions.

## 3.7 Model Architectures

The LSTM model with 50 hidden units and the stable training was utilized with SELU activation, and the SELU activation within the output layer to output continuous values. It was trained on Adam optimizer and MSE loss, and due to the generation accuracy and consistency, the model was trained with running 50 epoch over shuffle=False with batch size 8. The GRU model adopted the same structure with the exception of the use of GRU units replacing LSTM cells to produce even better convergence rates and reduced parameters. The Transformer-like model used a simplified attention mechanism that had three attention heads and input sequence attention. It included dropout regularization (0.1), residual connections, layer normalization, a feed-forward network with 64 hidden units, and final SELU activation for prediction output.

For traditional models, Gradient Boosting was tuned using GridSearchCV with parameters like criterion, learning rates, number of estimators, and loss functions. Random Forest was tuned with variations in min\_samples\_split, max\_depth, n\_estimators, and criteria. Linear Regression was adjusted based on fit\_intercept, copy\_X, and n\_jobs.

## 3.8 Tools and Techniques

The project was developed entirely in Python using Google Colab as the main platform. Pandas and NumPy were used for data manipulation, while Matplotlib and Seaborn supported visualization. Machine learning modeling was done using Scikit-learn, and deep learning models were built using TensorFlow and Keras. Hyperparameter tuning was carried out with GridSearchCV. Evaluation metrics included MAE, MSE, RMSE, and R² to ensure a comprehensive understanding of model performance.

## 3.9 Model Evaluation

Model evaluation relied on standard regression metrics. MAE provided average deviation from actual values, MSE penalized large deviations, RMSE offered interpretability by converting errors back to original units, and R² measured the proportion of variance explained. Predicted versus actual values were visualized using line plots and scatter plots, with a focus on predictions for the year 2021 to evaluate generalization capabilities.

## 3.10 Ethical Considerations

All data utilized in the research were publicly available and anonymized, ensuring no privacy concerns. There was no personally identifying information present. The dataset was used purely for academic and research purposes. All preprocessing, modeling, and evaluation steps were performed transparently to maintain research integrity and prevent misleading claims about model accuracy or applicability.

## 3.11 Project Management

The project was managed in phases, starting from data acquisition, preprocessing, and modeling, to evaluation and documentation. Weekly milestones were set to track progress, and Google Drive and GitHub were used for version control and collaboration. Google Colab facilitated cloud-based experimentation, enabling resource-efficient development. The project was finalized over six weeks in a structured and timely manner.

## 3.12 Summary

This chapter outlined the complete methodology for predicting life expectancy, from WHO data acquisition to preprocessing, sequence generation, model development, and evaluation. Both DL and traditional ML models were used for comparison. Ethical practices and structured project management ensured reliability. The next chapter presents the experimental results and performance analysis of these models.

# Experiment

## Overview of the Life Expectancy Dataset

The selected data set, which has a total row count of 220 rows and 10 columns, is loaded as Life expectancy.xlsx. This data set includes the following columns: Country, Year, Life Expectancy at Birth (years), and additional demographic and health variables, such as hospital beds, Incidence of tuberculosis, infant mortality rate, population utilizing securely managed sanitation facilities, population using basic drinking water services, current health expenditure (CHE) as a percentage of GDP (%), and fertility rate. The existence of missing information (NaN), especially in columns, such as, Population utilizing at least basic drinking-water services (%) and Hospital beds (per 10 000 population), highlights the necessity of data preprocessing steps for handling missing data and encoding categorical variables prior to implementing a model.

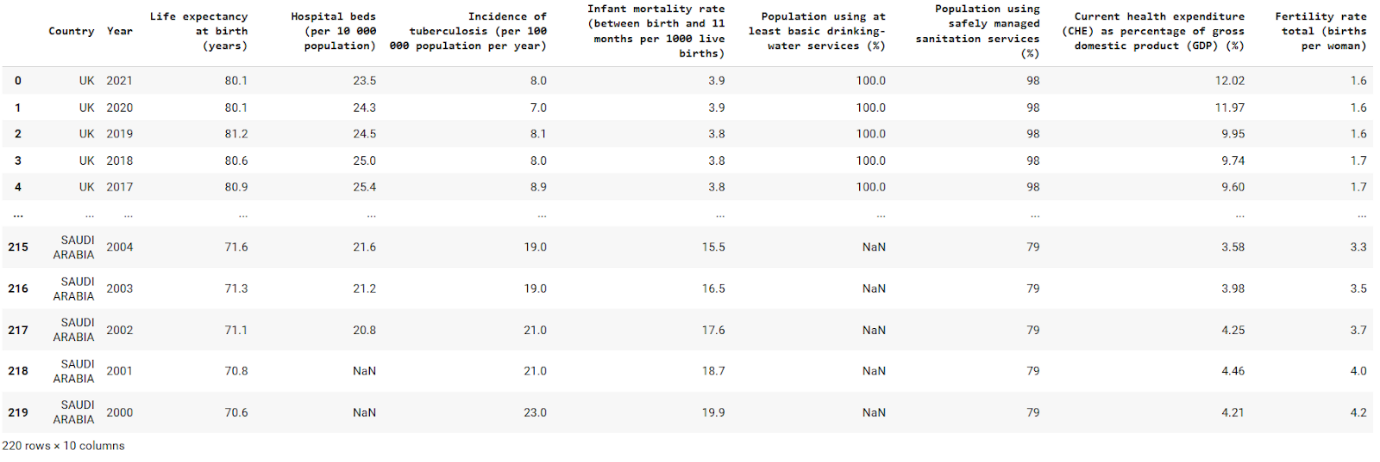


Figure Structure of Life Expectancy Dataset

The structure and data types of WHO dataset of life expectancy is shown below. The different columns have different types of data: the column named country is object (probably strings), the Year and Population using safely managed sanitation services (%) are int64, whereas the other seven, including the target, Life expectancy at birth (years), are float64. More importantly, the column row called the Non-Null Count indicates that there are missing values (NaNs) in two columns: 'Hospital beds (per 10 000 population)' column have only 190 non-null values out of 220, and Population utilizing at least basic drinking-water services (%) column has 198 non-null values, which require further processing at the preprocessing stage.

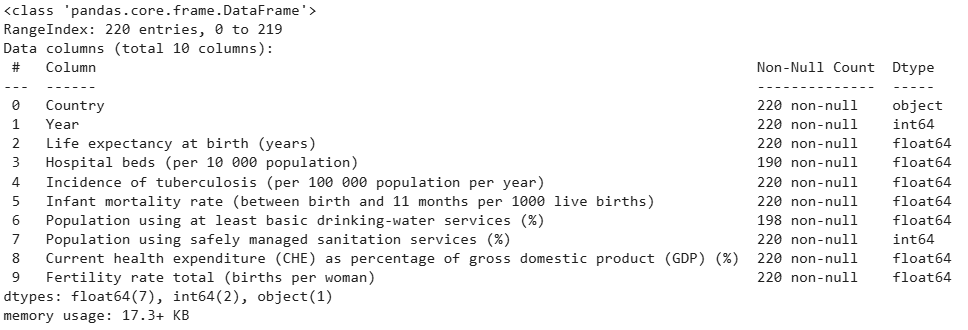


Figure Data Types of Life Expectancy Dataset

## Preprocessing Stage – Handling Missing Data

The result of life\_expect.isna().sum() gives a summary and compact overview of any missing values (NaNs) that are contained per column in life expectancy data frame.

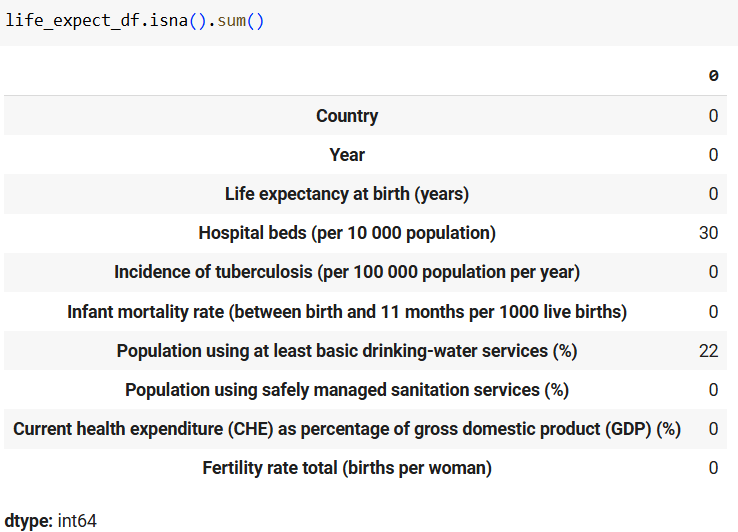


Figure Null Value Counts in Life Expectancy Dataset

The above figure clearly illustrates that the majority of columns contain no missing values (0 NaNs), whereas some columns are affected by incompleteness: there are 30 missing values in column 'Hospital beds (per 10 000 population)', and 22 in column 'Population utilizing at least basic drinking-water services (%)'.

To address these missing values, first, it ensures that the column 'Hospital beds (per 10 000 population)' is purely a numeric column by converting any non-numeric entries into NaN values. Subsequently, it computes the mean of the value of the column 'Hospital beds (per 10 000 population)' and Population utilizing at least basic drinking-water services (%) column. These computed mean values go along at solving the NaN values of their own corresponding columns. The resultant output shows 0 in all the columns demonstrates that this imputation process has successfully eliminated all missing values from the dataset, making it ready for further analysis as shown below.

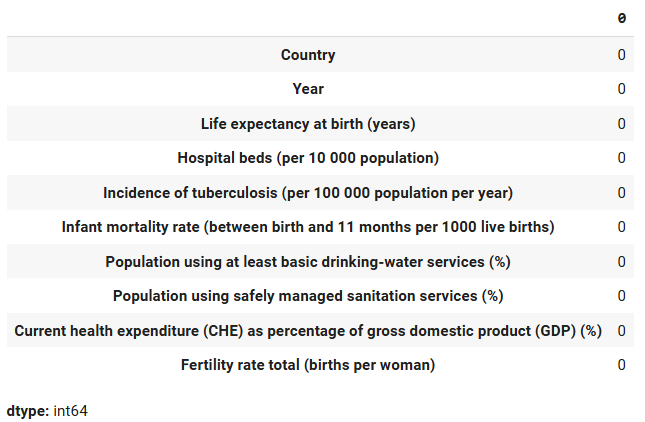


Figure Summary of Missing Values After Imputation

## Visualization of Life Expectancy Data

The below plot illustrates the visualization of how life expectancy at birth changes with time (between 2000 and 2021) in various countries where each distinct colored line represents a different country.

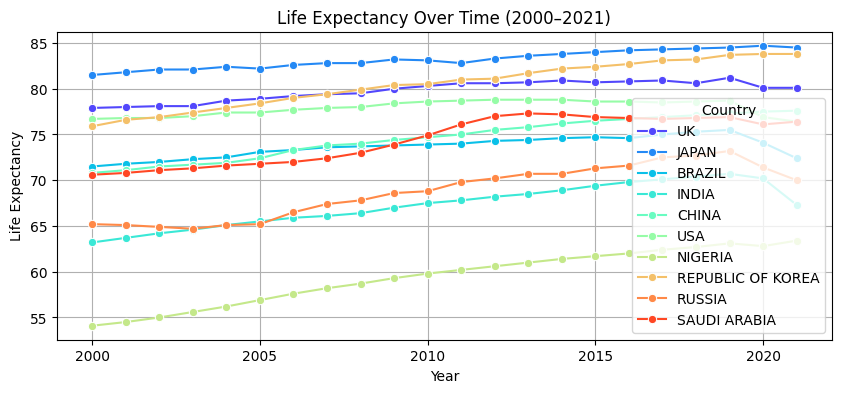


Figure Visualization of Life Expectancy Over Time

The above graph shows the overall rising tendency in most countries, meaning health and life conditions improved over time (2000–2021). Moreover, there are inequalities, with some countries having higher life expectancy (e.g., Japan) and others showing dips or plateaus. For example, Russia and Saudi Arabia had bigger drops during the COVID-19 pandemic because of higher deaths and weaker healthcare systems, while Japan and Republic of Korea had only small declines due to strong healthcare, quick actions, and fewer deaths. To understand these country-specific differences more clearly, the following figures focus on selected nations.

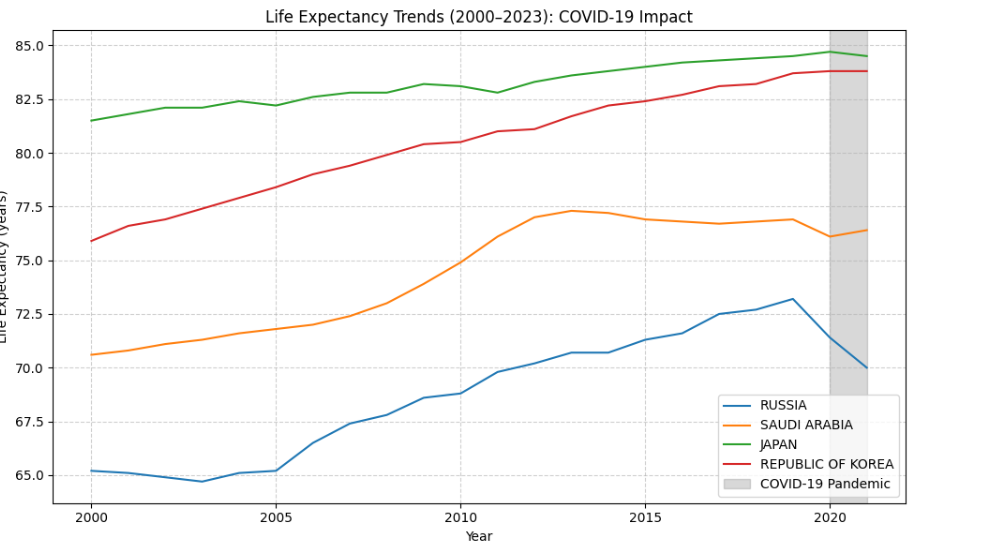


Figure Life Expectancy Trends of Russia, Saudi Arabia, Japan and Republic of Korea

These observations are supported by the correlation matrix heatmaps (below), which illustrate how life expectancy relates to key health and socio-economic factors. Russia and Saudi Arabia show strong negative correlations with hospital capacity and higher disease burdens, explaining sharper drops during COVID-19. In contrast, Japan and South Korea demonstrate strong positive correlations between health expenditure and life expectancy, reflecting resilient healthcare systems and effective public health measures. Overall, the heatmaps highlight the critical role of healthcare infrastructure, disease control, and policy responses in shaping life expectancy outcomes. Building on these insights, the next step is to examine broader correlations between life expectancy and multiple key variables across countries in 2000–2021.

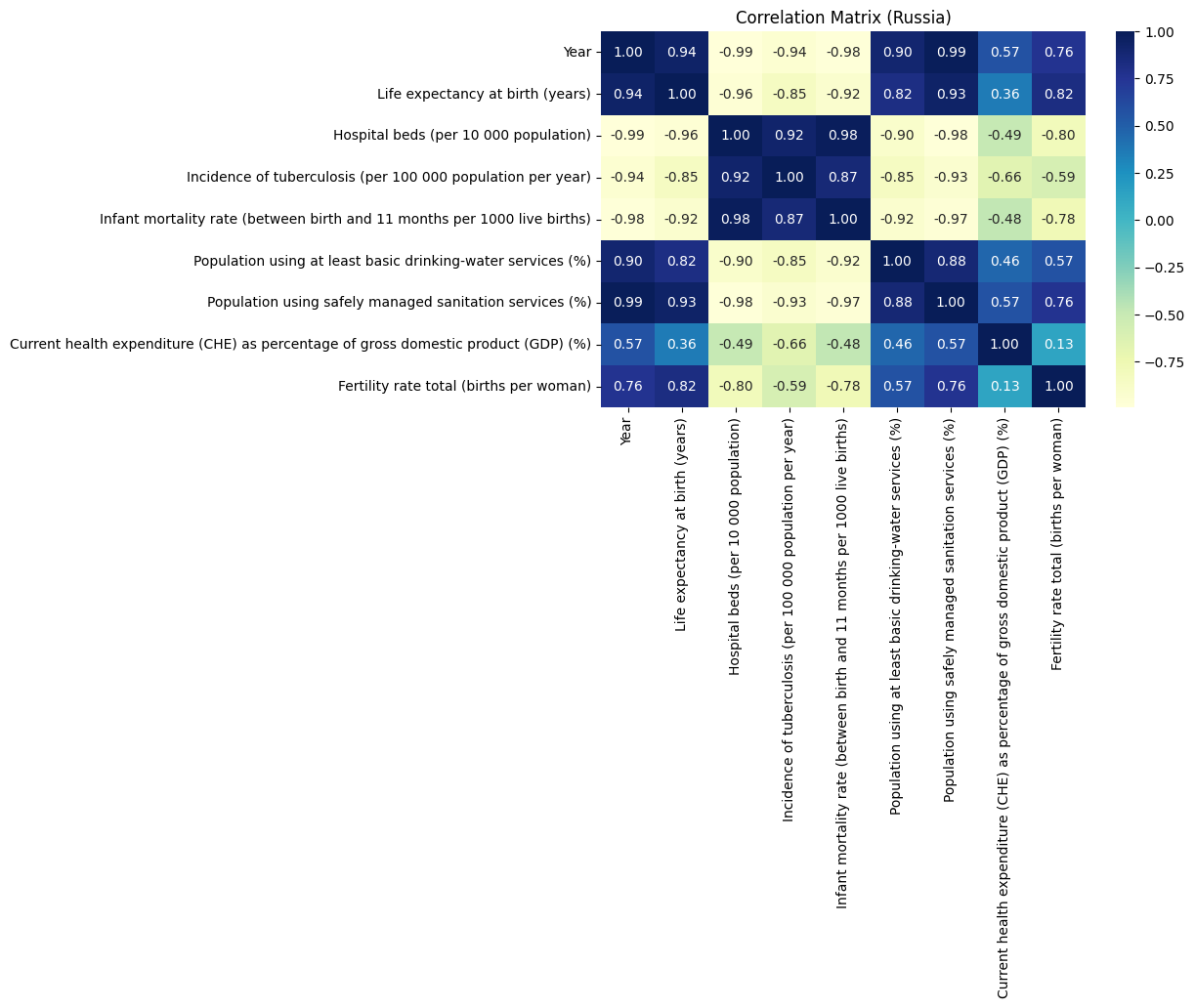


Figure Correlation Matrix Heatmap for Russia

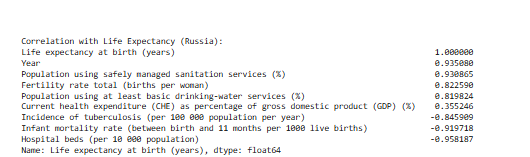


Figure Correlation of Life Expectancy with Key Variables in Russia

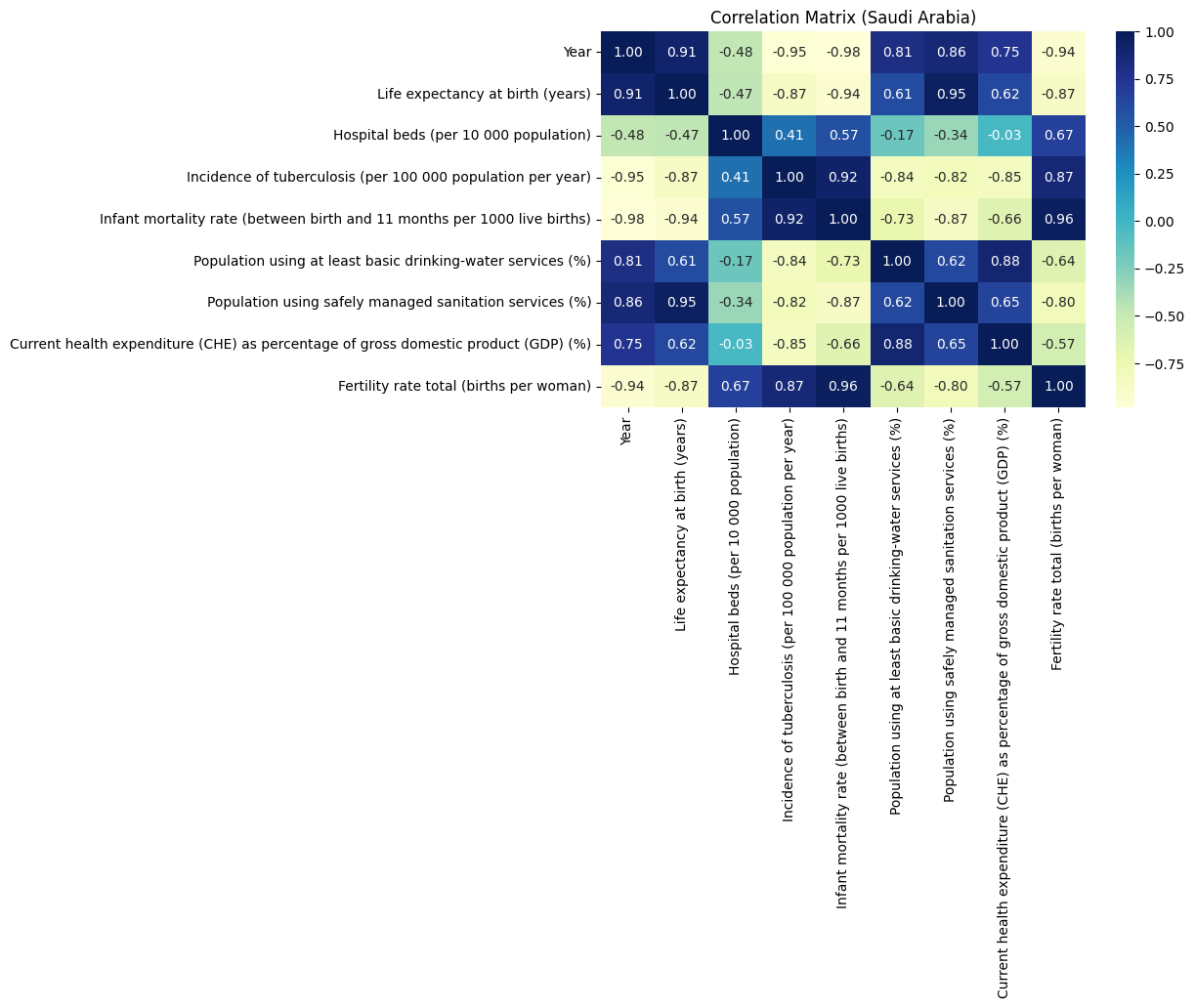


Figure Correlation Matrix Heatmap for Saudi Arabia

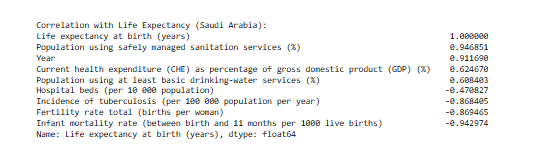


Figure Correlation of Life Expectancy with Key Variables in Saudi Arabia

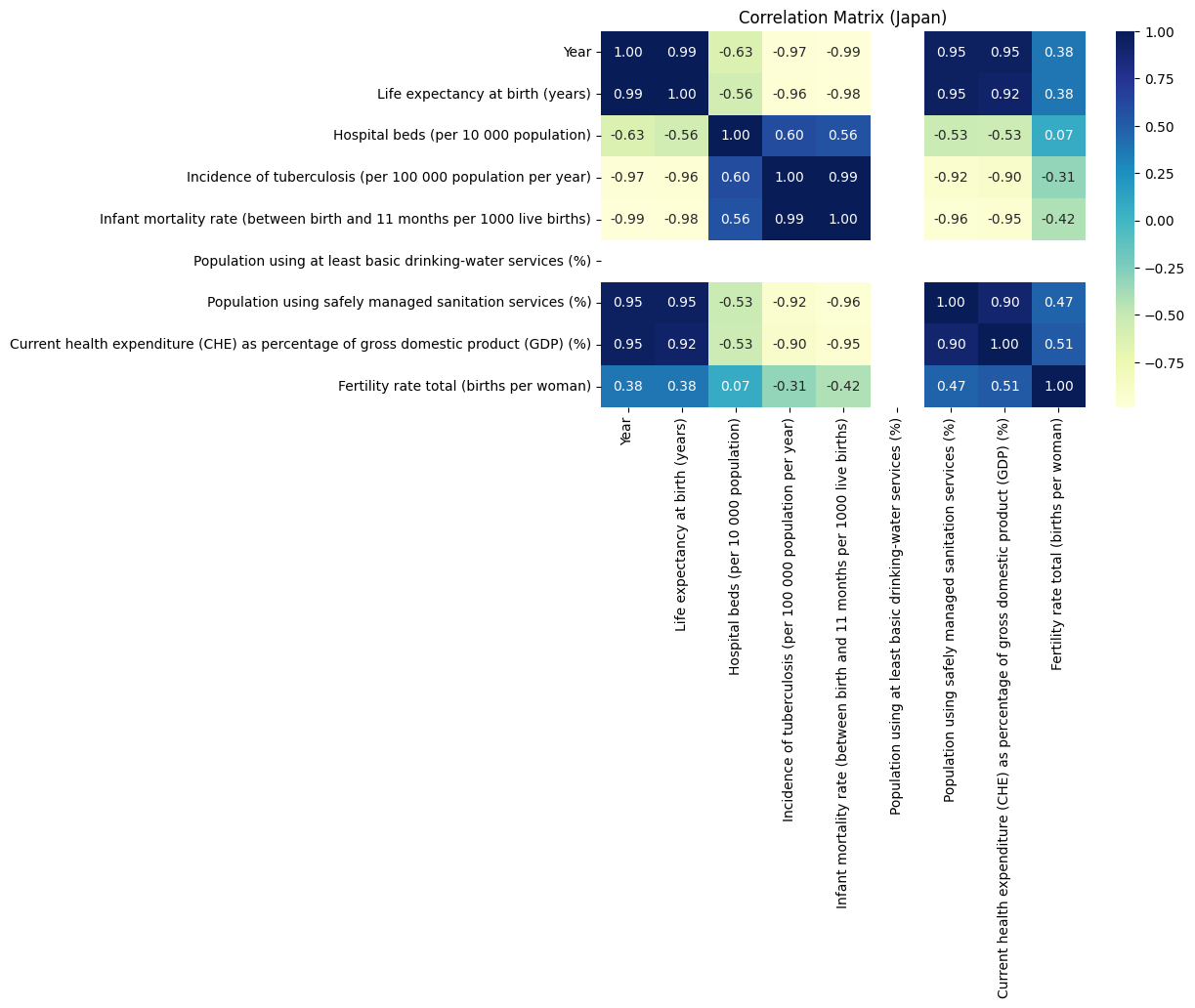


Figure Correlation Matrix Heatmap for Japan

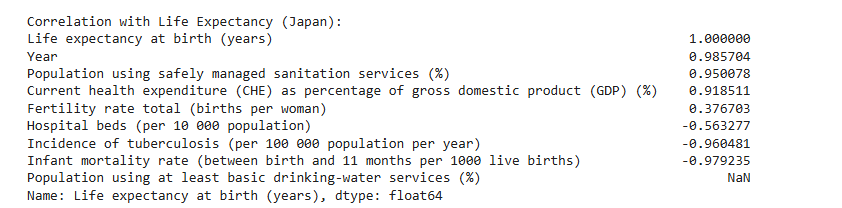


Figure Correlation of Life Expectancy with Key Variables in Japan

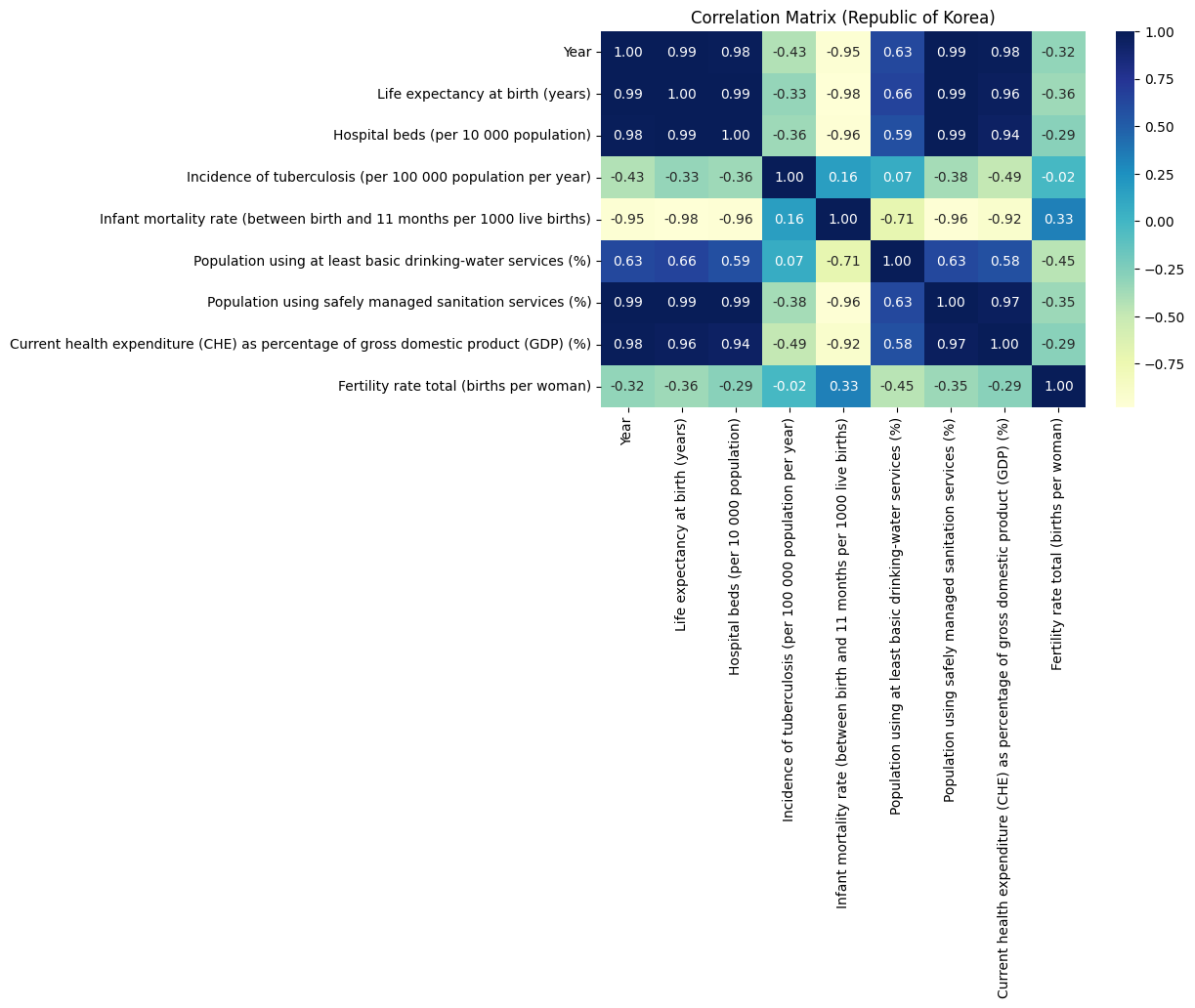


Figure Correlation Matrix Heatmap for Republic of Korea

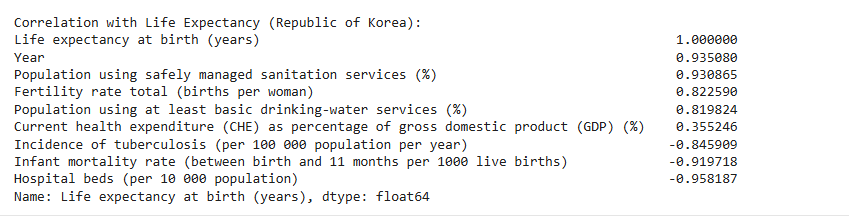


Figure Correlation of Life Expectancy with Key Variables in Republic of Korea

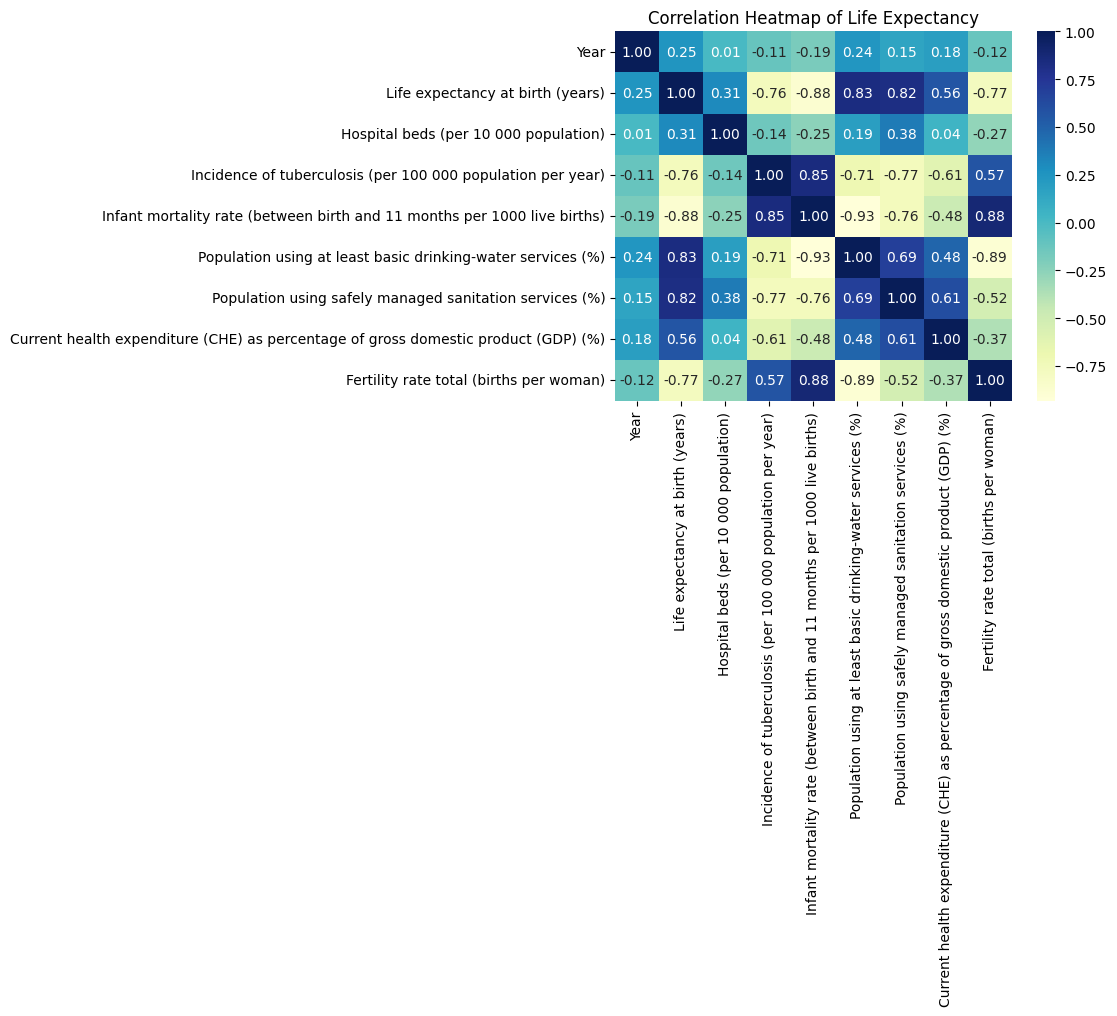


Figure Heat Map Visualization of Numeric Columns

The heatmap shows the correlation coefficient estimates of the features and the variable 'Life expectancy at birth (years)'. There exists a strong positive relation between life expectancy and parameters, such as, 'Population utilizing safely managed sanitation services (%)' (0.82) and ‘Population using at least basic drinking-water services (%)’ (0.83), showing that higher the values in these parameters, higher the life expectancy is likely to be. Conversely, there are strong negative correlations between life expectancy and infant mortality rate (-0.88), Fertility rate total (births per woman) (-0.77), and the Incidence of tuberculosis (-0.76). These negative correlations mean that the larger the infant mortality, the fertility rates, and the incidences of tuberculosis, the less the life expectancy.

Since fertility rate is one of the strongest negatively correlated factors, the following figure provides a closer look at its trend over time.

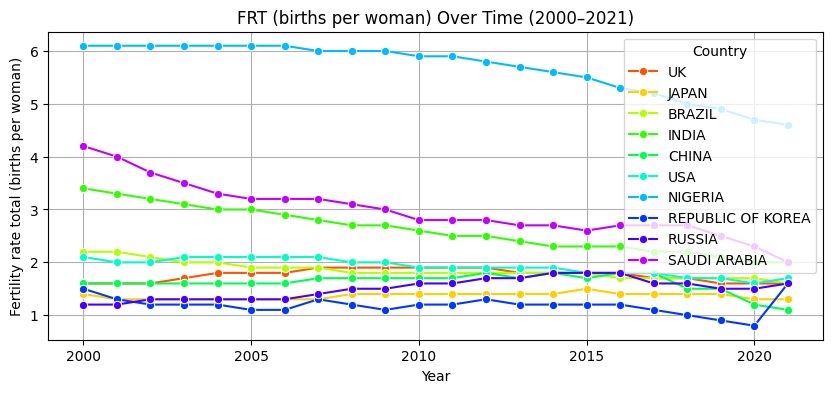


Figure Visualization of Fertility Rate Over Time

The plot of FRT (births per woman) Over Time (2000-2021) shows a general declining trend in the total fertility rates of the majority of countries observed within the discussed time range. Although there are countries where fertility rates remain high compared to others, the general trend shows that the world is moving towards fewer children born per women in these 20 years.

Overall, these visualizations and correlations highlight how improvements in healthcare, sanitation, and demographic shifts collectively shape the trends in life expectancy across nations.

## Numerical Conversion of Categorical Features

The ‘Country’ feature is of categorical type and it is preprocessed for machine learning. First, the data is arranged in the chronologically historical order per country. Then, it encodes the categorical names of countries into corresponding numbers in a new column named Country\_enc using Label Encoder. The new encoded column is visualized below.

****

Figure Verification of Encoded Country Entries

## Data Preparation for Time-Series Forecasting

Next, the life expectancy dataset is prepared for time-series predicting, by structuring the data into sequences suitable for predictive models. It identifies "Life expectancy at birth (years)" as the target variable removes identifier columns like 'Country' and 'Year' from the features. Next, a sliding window strategy with a three-year is implemented to create sequences. In order to forecast the life expectancy for the following year, it then iterates over each nation, sorting data by year and generating sliding window samples using the attributes from the previous three years. Lists containing the features, target values, years, and country labels are transformed into arrays for modeling. Then, the data is partitioned into training and testing datasets where the data till 2018 will be utilized in training and after this data shall be held back and be used in testing the model. The number of samples devoted to training the model is 160 and 30 samples are designated to check the performance of the model on unseen data as shown below.

https://lh7-rt.googleusercontent.com/docsz/AD_4nXcm0RWAxDkM7OHoYPLQ1n5RLqGC5BsO1v7Fy8CZ60Qe2lm4QvCf386M-e-_PWfuPCACKDRuE8rhavkE2UVVllMG6Fw4p_wm2axLqb7DNVICU_dbaBI6f1UAOA8cDBU6GIKrJDeTbcLC7zT1ZuYZLw?key=MHzK0k_tkPG2R7rcSun8RQ

Figure Count of Training and Testing Samples

The below image presents two representatives of the test set, indicating, in each, the country (Brazil) and the target year (2019 and 2020 respectively) and the input features generated using a window of 3 years of historical data.

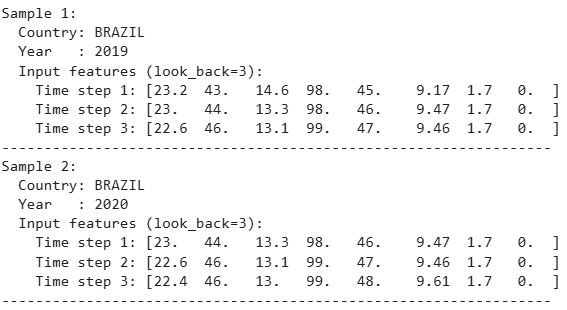


Figure Sample Test Set Inputs for Brazil (2019 and 2020)

## Model Implementation and Evaluation

Next, a comprehensive suite of regression models including LSTM, GRU, Transformer, Gradient Boosting, Random Forest, and LR to detect life expectancy based on historical data are implemented. All models are trained and tested based on applicable performance measures (i.e., R², MAE, MSE, RMSE) and hyperparameters are tuned using GridSearchCV to maximize performance.



Figure Comparison of Baseline and Advanced Regression Models for Life Expectancy Prediction

When examining performance, advanced deep learning models performed lower than baseline machine learning regression models. Gradient Boosting (GB) was the most successful, with an R² score of 0.9407 and the lowest error rates (MAE = 1.2267, MSE = 2.3638, RMSE = 1.5375). This means Gradient Boosting accounted for about 94 percent of the variance in life expectancy with little prediction error. RF also showed high performance with R² = 0.9227, just after Gradient Boosting. Even Linear Regression, with R² = 0.8031, reflected underlying relationships reasonably. Though fairly successful, linear models are constrained by assuming linear relationships and cannot incorporate complex interactions present in the data. This emphasizes the role of non-linear models in health-related predictive analytics, especially with multifaceted social-economic and health monitors.

Shifting to deep learning models, the LSTM architecture performed well with R² = 0.8681 compared to GRU (0.3116) and Transformer (0.8263). Its capacity to process sequential data and temporal dependencies seems beneficial even with this tabular dataset. The corresponding errors (MAE: 1.8062; RMSE: 2.2925) are also small, showing LSTM approximates the target variable with fairly high accuracy. However, it still lagged behind Gradient Boosting and Random Forest as its error rates are higher.

The scatter plot of LSTM visually compares actual life expectancy (years) against predicted life expectancy (years) by the LSTM model (red dots).

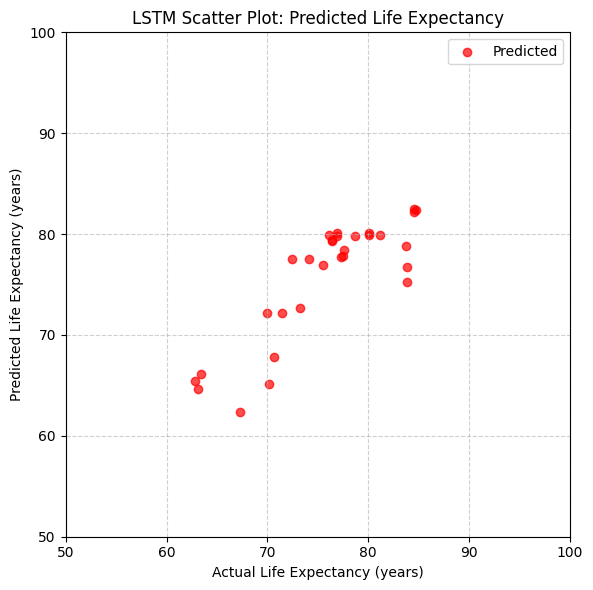


Figure Comparison of Actual vs Predicted Life Expectancy (LSTM Model)

Although at some points (mid-range values in 70s), predicted values match closely with actual values, there are deviations where the LSTM underestimates higher life expectancies and slightly overestimates lower ones, showing decent accuracy yet still with errors in some ranges. scatter plot of GRU and Transformers are attached in Appendix. However, GRU and Transformer have R² of 0.3116 and 0.8263, respectively, providing moderate performance. The low R² and large errors indicate GRU did not learn meaningful representations. The moderate R² of Transformer (0.8263) implies it captures some variance, but predictions are less accurate than Gradient Boosting and Random Forest, as they are also data-hungry. Scatter plots of GRU and Transformer are attached in the Appendix. This could result in overfitting and failing to learn useful patterns on tabular data.

Overall, these findings indicate traditional ensemble models, including GB and RF, are effective and reliable regression methods to predict life expectancy, with high accuracy and computational efficiency. Although deep learning models like LSTM can be competitive, especially with temporal or sequential data, they need careful tuning and do not always justify their complexity. Based on these findings, Gradient Boosting is recommended as the most efficient and accurate model.

## Critical Comparative Analysis with Past Works

This research compares traditional and advanced regression models more explicitly and exhaustively on one dataset, a gap in existing literature. Although studies such as Lipesa et al. (2023) and Ahamad et al. (2025) revolve around XGBoost and Random Forest, this research critically compares their performance to deep learning models LSTM, GRU, and Transformer, using the same WHO dataset with comparable measures (R², MAE, MSE, RMSE). The results are not consistent with some previous literature, as simpler models Gradient Boosting and Random Forest perform better than advanced deep learning on this specific tabular dataset. This does not agree with literature anticipation, such as El-Rashidy et al. (2025) pointing to LSTM-GRU potential, and Beeksma et al. (2019) showing LSTM effectiveness on electronic medical records. The Transformer, despite R² of 0.8263, still underperformed compared to Gradient Boosting and Random Forest, showing limitations of sequence-based architectures on smaller tabular datasets. This research also adds rigor by using a comprehensive, preprocessed WHO dataset with clear handling of missing values, addressing weaknesses cited by Dawoud et al. (2023), including bias and low generalizability, and the small dataset size noted by Rubi et al. (2021).

# Conclusion

This research compared ML and DL models to predict life expectancy shows a clear advantage for traditional ensemble methods over sophisticated deep learning models. Gradient Boosting (GB) turned out to be the best, since its R2 is 0.9407 and error rates are extremely low (MAE=1.2267, RMSE=1.5375), which is closely followed by RF with R² = 0.9227. Even a simpler model, Linear Regression had a rather good R² of 0.8031. By contrast, more complex deep learning models mostly did not perform so well: LSTM, though improving significantly with R² = 0.8681, is still ranked behind the best of the ML models with MAE=1.8062 and RMSE=2.2925. GRU performed quite poorly with R² = 0.3116 and the Transformer model only reached an R² of 0.8263 with higher error values (MAE=2.1516, RMSE=2.6308).These results indicate that when working with this tabular data, the intuitive advantages of learning complex and non-linear relationships with ensemble tree-based models effectively dominated the more complex deep learning models. Overall, the Gradient Boosting model is the most effective and precise one in this life expectancy forecasting task.

## Future Work

* Focus on advanced feature creation/selection to better suit DL models for tabular data.
* Explore combining DL with traditional ML or ensembling multiple DL models.
* Implement advanced hyperparameter tuning and architecture search for DL models to unlock their full potential.

References

Ahamad, S.S., Kumar, K.P., Ganesh, S.S., Alharbi, F., Alharby, S.A., Dendukuri, V.S.P. and Koneru, S.V., (2025). Machine learning-based life expectancy prediction in developed and developing regions. IEEE Access, 13, pp.69520–69531.

Ahmed, D.M., Hassan, M.M. and Mstafa, R.J. (2022), ‘A review on deep sequential models for forecasting time series data’, *Applied Computational Intelligence and Soft Computing*, 2022(1), p.6596397.

Beeksma, M., Verberne, S., van den Bosch, A., Das, E., Hendrickx, I. and Groenewoud, S., (2019). Predicting life expectancy with a long short-term memory recurrent neural network using electronic medical records. *BMC medical informatics and decision making*, *19*, pp.1-15.

Chen, Z., Ma, Y., Hua, J., Wang, Y. and Guo, H. (2021), ‘Impacts from economic development and environmental factors on life expectancy: A comparative study based on data from both developed and developing countries from 2004 to 2016’, *International Journal of Environmental Research and Public Health*, 18(16), p.8559.

Das, A., Uddin, M.M. and Karim, M.R., (2025). Predicting Life Expectancy using Machine Learning Techniques. *International Journal of Statistical Sciences*, *25*(1), pp.55-70.

Dawoud, A.M. and Abu-Naser, S.S., (2023). Predicting life expectancy in diverse countries using neural networks: insights and implications. International Journal of Academic Engineering Research (IJAER), 7(9), pp.46–54.

El-Rashidy, N., Sultan, Y.A. and Ali, Z.H., (2025). Predecting power transformer health index and life expectation based on digital twins and multitask LSTM-GRU model. *Scientific Reports*, *15*(1), p.1359.

Kizito, R., Scruggs, P., Li, X., Devinney, M., Jansen, J. and Kress, R. (2021), ‘Long short-term memory networks for facility infrastructure failure and remaining useful life prediction’, *IEEE Access*, 9, pp.67585–67594.

Lipesa, B.A., Okango, E., Omolo, B.O. and Omondi, E.O., (2023). An application of a supervised machine learning model for predicting life expectancy. *SN Applied Sciences*, *5*(7), p.189.

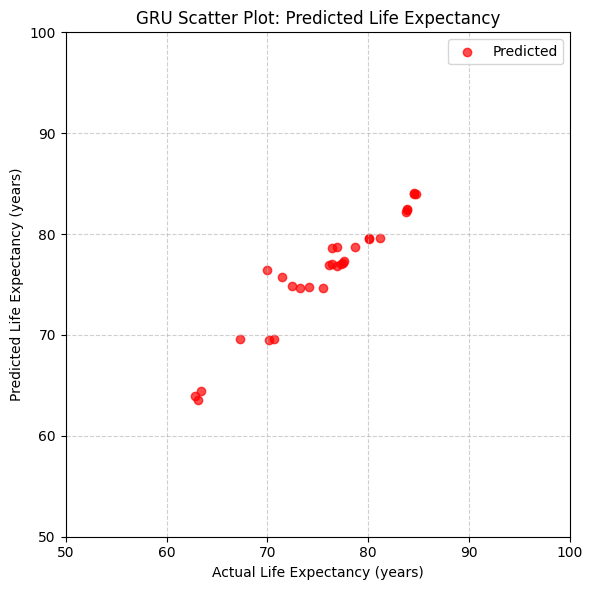
Pisal, N.S., Abdul-Rahman, S., Hanafiah, M. and Kamarudin, S.I., (2022). Prediction of life expectancy for Asian population using machine learning algorithms. *Malaysian Journal of Computing*, *7*(2), pp.1150-1161.

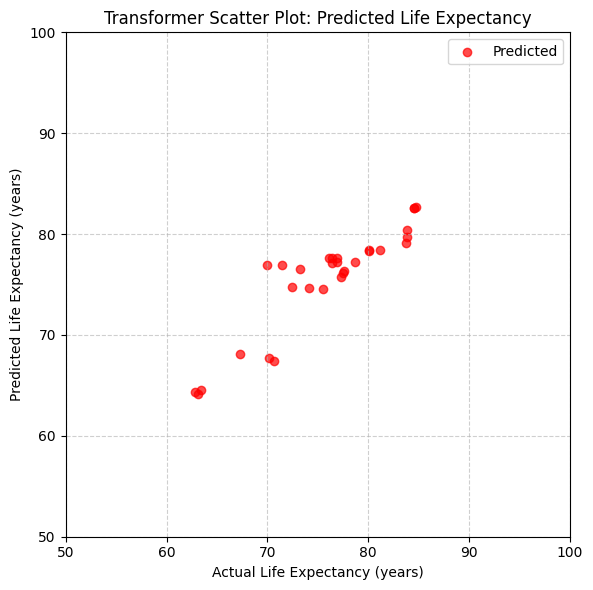
Rubi, M.A., Bijoy, H.I. and Bitto, A.K., (2021), July. Life expectancy prediction based on GDP and population size of Bangladesh using multiple linear regression and ANN model. In *2021 12th international conference on computing communication and networking technologies (ICCCNT)* (pp. 1-6). IEEE.

World Health Organization (2024), *World health statistics 2024: monitoring health for the SDGs, sustainable development goals*. World Health Organization.

# Appendix

## Scatter Plots





## Code

from google.colab import drive

drive.mount('/content/drive')

import pandas as pd

import warnings

warnings.filterwarnings('ignore')

import matplotlib.pyplot as plt

import seaborn as sns

life\_expect\_df = pd.read\_excel('Life expectancy.xlsx')

life\_expect\_df

life\_expect\_df.info()

life\_expect\_df.isna().sum()

Convert the column to numeric, coercing errors to Nan

life\_expect\_df['Hospital beds (per 10 000 population)'] = pd.to\_numeric(life\_expect\_df['Hospital beds (per 10 000 population)'], errors='coerce')

# filling NaN values by calculating mean value

mean\_value\_bed = life\_expect\_df['Hospital beds (per 10 000 population)'].mean()

mean\_value\_water = life\_expect\_df['Population using at least basic drinking-water services (%)'].mean()

life\_expect\_df['Hospital beds (per 10 000 population)'].fillna(mean\_value\_bed, inplace=True)

life\_expect\_df['Population using at least basic drinking-water services (%)'].fillna(mean\_value\_water, inplace=True)

life\_expect\_df.isna().sum()

Plot for Target (Life Expectancy) by Country

plt.figure(figsize=(10, 4))

sns.lineplot(data=life\_expect\_df, x='Year', y='Life expectancy at birth (years)', hue='Country', marker='o', palette='rainbow')

plt.title("Life Expectancy Over Time (2000–2021)")

plt.ylabel("Life Expectancy")

plt.grid(True)

plt.show()

life\_expect\_df.nunique()

life\_expect\_df.describe()

Heat Map for Numeric Columns

numeric\_data = life\_expect\_df.select\_dtypes(include=['float64', 'int64'])

# correlation between columns

sns.heatmap(numeric\_data.corr(), annot=True, cmap='YlGnBu', fmt=".2f")

plt.title("Correlation Heatmap of Life Expectancy Data")

plt.show()

Plot for Fertility rate total

plt.figure(figsize=(10, 4))

sns.lineplot(data=life\_expect\_df, x='Year', y='Fertility rate total (births per woman)', hue='Country', marker='o',palette='gist\_rainbow')

plt.title("FRT (births per woman) Over Time (2000–2021)")

plt.ylabel("Fertility rate total (births per woman)")

plt.grid(True)

plt.show()

import numpy as np

from sklearn.preprocessing import LabelEncoder

from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import LSTM, Dense

life\_expect\_df = life\_expect\_df.sort\_values(['Country', 'Year']).reset\_index(drop=True)

life\_le = LabelEncoder()

life\_expect\_df['Country\_enc'] = life\_le.fit\_transform(life\_expect\_df['Country'])

life\_expect\_df['Country\_enc'].value\_counts()

life\_expect\_df

# Define target and exclude identifier columns from features

target = 'Life expectancy at birth (years)'

exclude\_cols = ['Country', 'Year', target]

features = [col for col in life\_expect\_df.columns if col not in exclude\_cols]

look\_back = 3 # Number of past years to use as features

life\_X, life\_y, year\_list, country\_list = [], [], [], []

for country in life\_expect\_df['Country'].unique():

# Get data for this country and sort by year

country\_data = life\_expect\_df[life\_expect\_df['Country'] == country].sort\_values('Year')

features\_data = country\_data[features].values

target\_data = country\_data[target].values

years\_data = country\_data['Year'].values

# Create samples using sliding window

for i in range(len(country\_data) - look\_back):

life\_X.append(features\_data[i:i+look\_back])

life\_y.append(target\_data[i+look\_back])

year\_list.append(years\_data[i+look\_back])

country\_list.append(country)

# Convert to numpy arrays

life\_X = np.array(life\_X)

life\_y = np.array(life\_y)

year\_list = np.array(year\_list)

country\_list = np.array(country\_list)

# Split into train and test sets

train\_idx = year\_list <= 2018 # Train: up to 2018

test\_idx = year\_list > 2018 # Test: 2019, 2020, 2021

X\_train, y\_train = life\_X[train\_idx], life\_y[train\_idx]

X\_test, y\_test = life\_X[test\_idx], life\_y[test\_idx]

test\_countries = country\_list[test\_idx]

test\_years = year\_list[test\_idx]

print("Training Samples for Life Expectancy:", X\_train.shape[0])

print("Testing Samples for Life Expectancy :", X\_test.shape[0])

X\_train

X\_test

# Display each test sample's year, country, and its corresponding input features

for i in range(len(X\_test)):

print(f"Sample {i + 1}:")

print(f" Country: {test\_countries[i]}")

print(f" Year: {test\_years[i]}")

print(f" Input features (look\_back={look\_back}):")

for t in range(look\_back):

print(f" Time step {t + 1}: {X\_test[i][t]}")

print("-" \* 60)

### LSTM Regression Model

import numpy as np

import tensorflow as tf

import random

import os

from sklearn.metrics import mean\_absolute\_error

from sklearn.metrics import mean\_squared\_error

from sklearn.metrics import r2\_score

# Setting the seed for reproducibility

seed = 42

np.random.seed(seed)

tf.random.set\_seed(seed)

random.seed(seed)

os.environ['PYTHONHASHSEED'] = str(seed)

life\_model = Sequential()

life\_model.add(LSTM(50, activation='selu', input\_shape=(look\_back, len(features))))

life\_model.add(Dense(1, activation='selu'))

life\_model.compile(optimizer='adam', loss='mse')

life\_model.fit(X\_train, y\_train, epochs=50, batch\_size=8, validation\_data=(X\_test, y\_test), shuffle=False)

y\_pred = life\_model.predict(X\_test)

# Regression Metrics

mae = mean\_absolute\_error(y\_test, y\_pred)

mse = mean\_squared\_error(y\_test, y\_pred)

rmse = np.sqrt(mse)

r2 = r2\_score(y\_test, y\_pred)

print(f"\nLife Expectancy in LSTM R²: {r2:.4f}")

print(f"Life Expectancy in LSTM MAE: {mae:.4f}")

print(f"Life Expectancy in LSTM MSE: {mse:.4f}")

print(f"Life Expectancy in LSTM RMSE: {rmse:.4f}")

# Convert predictions to 1D array for easier display

y\_pred\_flat = y\_pred.flatten()

y\_test\_flat = y\_test.flatten()

print("\nFirst 10 Actual vs Predicted Life Expectancy:")

for actual, predicted in list(zip(y\_test\_flat, y\_pred\_flat))[:10]:

print(f"Actual: {actual:.2f}, Predicted: {predicted:.2f}")

Plot actual vs predicted values

# Scatter Plot: Actuals & Predictions

plt.figure(figsize=(8, 6))

plt.scatter(y\_test, y\_test, color='royalblue', alpha=0.7, label="Actual")

plt.scatter(y\_test, y\_pred, color='red', alpha=0.7, label="Predicted")

plt.xlabel("Actual Life Expectancy (years)")

plt.ylabel("Predicted Life Expectancy (years)")

plt.title("LSTM Scatter Plot: Actual vs Predicted Life Expectancy")

plt.legend()

plt.grid(True, linestyle='--', alpha=0.6)

plt.tight\_layout()

plt.show()

print("\nLife Expectancy Inputs (2018-2020) used for the Prediction of 2021\n")

for country in life\_expect\_df['Country'].unique():

country\_data = life\_expect\_df[life\_expect\_df['Country'] == country].sort\_values('Year')

# Printing Country wise life expectancy for the years 2018, 2019, 2020

print(f"--- {country} ---")

for year in [2018, 2019, 2020]:

value = country\_data[country\_data['Year'] == year]['Life expectancy at birth (years)']

if not value.empty:

print(f"Year: {year}, Life Expectancy: {value.values[0]:.2f}")

# Actual 2021 life expectancy

actual\_2021 = country\_data[country\_data['Year'] == 2021]['Life expectancy at birth (years)']

if not actual\_2021.empty:

print(f"Actual 2021 Life Expectancy: {actual\_2021.values[0]:.2f}")

else:

print("Actual 2021 not available")

# Predicted 2021 life expectancy from model

idx = np.where((test\_years == 2021) & (test\_countries == country))[0]

if len(idx) > 0:

print(f"Predicted 2021 Life Expectancy: {y\_pred[idx[0]][0]:.2f}")

else:

print("Prediction not available")

print()

### GRU Regression Model

from tensorflow.keras.layers import GRU

life\_model = Sequential()

life\_model.add(GRU(50, activation='selu', input\_shape=(look\_back, len(features))))

life\_model.add(Dense(1, activation='selu'))

life\_model.compile(optimizer='adam', loss='mse')

life\_model.fit(X\_train, y\_train, epochs=50, batch\_size=8, validation\_data=(X\_test, y\_test), shuffle=False)

y\_pred = life\_model.predict(X\_test)

# Regression Metrics

mae = mean\_absolute\_error(y\_test, y\_pred)

mse = mean\_squared\_error(y\_test, y\_pred)

rmse = np.sqrt(mse)

r2 = r2\_score(y\_test, y\_pred)

print(f"\nLife Expectancy in GRU R²: {r2:.4f}")

print(f"Life Expectancy in GRU MAE: {mae:.4f}")

print(f"Life Expectancy in GRU MSE: {mse:.4f}")

print(f"Life Expectancy in GRU RMSE: {rmse:.4f}")

Plot actual vs predicted values

# Scatter Plot: Actuals & Predictions

plt.figure(figsize=(8, 6))

plt.scatter(y\_test, y\_test, color='royalblue', alpha=0.7, label="Actual")

plt.scatter(y\_test, y\_pred, color='red', alpha=0.7, label="Predicted")

plt.xlabel("Actual Life Expectancy (years)")

plt.ylabel("Predicted Life Expectancy (years)")

plt.title("GRU Scatter Plot: Actual vs Predicted Life Expectancy")

plt.legend()

plt.grid(True, linestyle='--', alpha=0.6)

plt.tight\_layout()

plt.show()

### Transformer Regression Model

from tensorflow.keras.models import Model

from tensorflow.keras.layers import Input, Dense, LayerNormalization

from tensorflow.keras.layers import MultiHeadAttention, Dropout, Flatten, Add

from tensorflow.keras.optimizers import Adam

num\_heads = 3

embed\_dim = len(features)

ff\_dim = 64 # Feed-forward layer dimension

life\_inputs = Input(shape=(look\_back, len(features))) # Input layer

# Attention layers

life\_atten\_output = MultiHeadAttention(num\_heads=num\_heads, key\_dim=embed\_dim)(life\_inputs, life\_inputs)

life\_atten\_output = Dropout(0.1)(life\_atten\_output)

life\_atten\_output = Add()([life\_inputs, life\_atten\_output]) # Residual connection

life\_atten\_output = LayerNormalization(epsilon=1e-6)(life\_atten\_output)

# Feed Forward Network

life\_ff\_output = Dense(ff\_dim, activation='selu')(life\_atten\_output)

life\_ff\_output = Dense(embed\_dim, activation='selu')(life\_ff\_output)

life\_ff\_output = Add()([life\_atten\_output, life\_ff\_output]) # Residual connection

life\_ff\_output = LayerNormalization(epsilon=1e-6)(life\_ff\_output)

flatten = Flatten()(life\_ff\_output)

life\_outputs = Dense(1, activation='selu')(flatten) # Output layer

# Transformer Model

transformer\_model = Model(inputs=life\_inputs, outputs=life\_outputs)

transformer\_model.compile(optimizer=Adam(learning\_rate=0.001), loss='mse')

# Evaluation

transformer\_model.fit(X\_train, y\_train, epochs=50, batch\_size=8, validation\_data=(X\_test, y\_test), shuffle=False)

y\_pred = transformer\_model.predict(X\_test)

mae = mean\_absolute\_error(y\_test, y\_pred)

mse = mean\_squared\_error(y\_test, y\_pred)

rmse = np.sqrt(mse)

r2 = r2\_score(y\_test, y\_pred)

print(f"\nLife Expectancy in Transformer R²: {r2:.4f}")

print(f"Life Expectancy in Transformer MAE: {mae:.4f}")

print(f"Life Expectancy in Transformer MSE: {mse:.4f}")

print(f"Life Expectancy in Transformer RMSE: {rmse:.4f}")

Plot actual vs predicted values

# Scatter Plot: Actuals & Predictions

plt.figure(figsize=(8, 6))

plt.scatter(y\_test, y\_test, color='royalblue', alpha=0.7, label="Actual")

plt.scatter(y\_test, y\_pred, color='red', alpha=0.7, label="Predicted")

plt.xlabel("Actual Life Expectancy (years)")

plt.ylabel("Predicted Life Expectancy (years)")

plt.title("Transformer Scatter Plot: Actual vs Predicted Life Expectancy")

plt.legend()

plt.grid(True, linestyle='--', alpha=0.6)

plt.tight\_layout()

plt.show()

### Gradient Boosting Regression Model

from sklearn.model\_selection import GridSearchCV as life\_cv\_grid

from sklearn.ensemble import GradientBoostingRegressor as life\_gb\_regressor

from sklearn.metrics import mean\_absolute\_error

from sklearn.metrics import mean\_squared\_error

from sklearn.metrics import r2\_score

import numpy as np

reg\_params\_life = {

'loss': ['squared\_error','absolute\_error','quantile','huber'],

'learning\_rate': [0.1, 0.001, 0.0001, 0.3],

'n\_estimators': [30, 60, 100],

'criterion': ['friedman\_mse','squared\_error']

}

life\_ml\_mod = life\_gb\_regressor(random\_state=41)

# Reshape X\_train and X\_test for Gradient Boosting

X\_train\_reshaped = X\_train.reshape(X\_train.shape[0], -1)

X\_test\_reshaped = X\_test.reshape(X\_test.shape[0], -1)

life\_ml\_mod = life\_cv\_grid(life\_ml\_mod, reg\_params\_life, cv=2)

life\_ml\_mod.fit(X\_train\_reshaped, y\_train)

print("Selected Regression Parameters GBR:\n", life\_ml\_mod.best\_params\_)

life\_ml\_mod = life\_ml\_mod.best\_estimator\_

life\_ml\_mod.fit(X\_train\_reshaped, y\_train)

y\_pred = life\_ml\_mod.predict(X\_test\_reshaped)

mae = mean\_absolute\_error(y\_test, y\_pred)

mse = mean\_squared\_error(y\_test, y\_pred)

rmse = np.sqrt(mse)

r2 = r2\_score(y\_test, y\_pred)

print(f"\nLife Expectancy in GBR R²: {r2:.4f}")

print(f"Life Expectancy in GBR MAE : {mae:.4f}")

print(f"Life Expectancy in GBR MSE : {mse:.4f}")

print(f"Life Expectancy in GBR RMSE: {rmse:.4f}")

### Random Forest Regressor

from sklearn.ensemble import RandomForestRegressor as life\_rf\_regressor

reg\_params\_life = {

'min\_samples\_split': [1, 2, 3, 6],

'max\_depth': [3, 6, 10],

'n\_estimators': [30, 60, 100],

'criterion': ['friedman\_mse','squared\_error','absolute\_error','poisson']

}

life\_ml\_mod = life\_rf\_regressor(random\_state=41)

life\_ml\_mod = life\_cv\_grid(life\_ml\_mod, reg\_params\_life, cv=2)

life\_ml\_mod.fit(X\_train\_reshaped, y\_train)

print("Selected Regression Parameters RFR:\n", life\_ml\_mod.best\_params\_)

life\_ml\_mod = life\_ml\_mod.best\_estimator\_

life\_ml\_mod.fit(X\_train\_reshaped, y\_train)

y\_pred = life\_ml\_mod.predict(X\_test\_reshaped)

mae = mean\_absolute\_error(y\_test, y\_pred)

mse = mean\_squared\_error(y\_test, y\_pred)

rmse = np.sqrt(mse)

r2 = r2\_score(y\_test, y\_pred)

print(f"\nLife Expectancy in RFR R²: {r2:.4f}")

print(f"Life Expectancy in RFR MAE : {mae:.4f}")

print(f"Life Expectancy in RFR MSE : {mse:.4f}")

print(f"Life Expectancy in RFR RMSE: {rmse:.4f}")

### Linear Regression Regressor

from sklearn.linear\_model import LinearRegression as life\_line\_regressor

reg\_params\_life = {

'fit\_intercept': [True, False],

'copy\_X': [True, False],

'n\_jobs': [3, 6, 10]

}

life\_ml\_mod = life\_line\_regressor()

life\_ml\_mod = life\_cv\_grid(life\_ml\_mod, reg\_params\_life, cv=2)

life\_ml\_mod.fit(X\_train\_reshaped, y\_train)

print("Selected Regression Parameters LRR:\n", life\_ml\_mod.best\_params\_)

life\_ml\_mod = life\_ml\_mod.best\_estimator\_

life\_ml\_mod.fit(X\_train\_reshaped, y\_train)

y\_pred = life\_ml\_mod.predict(X\_test\_reshaped)

mae = mean\_absolute\_error(y\_test, y\_pred)

mse = mean\_squared\_error(y\_test, y\_pred)

rmse = np.sqrt(mse)

r2 = r2\_score(y\_test, y\_pred)

print(f"\nLife Expectancy in LRR R²: {r2:.4f}")

print(f"Life Expectancy in LRR MAE : {mae:.4f}")

print(f"Life Expectancy in LRR MSE : {mse:.4f}")

print(f"Life Expectancy in LRR RMSE: {rmse:.4f}")