# Human Life Expectancy Prediction Analysis

## 1. Introduction

The goal of this project is to develop a machine learning model that accurately predicts human life expectancy based on various socio-economic, environmental, and health-related features.

This report provides a **detailed analysis** of life expectancy prediction using multiple machine learning models. The study explores:

- Different **modeling approaches**
- **Effectiveness** of various models
- **Selection** of the best model based on performance metrics

## 2. Exploratory Data Analysis

### 2.1 Data Overview

The dataset consists of:

- **2071 observations and 23 features**
- **Numerical and categorical** variables representing **economic, health, and demographic** aspects affecting life expectancy
- Some features **contain missing values** and **potential outliers**

Using *train_df.describe()*, we obtained summary statistics for numerical variables, including:

- **Count, Mean, Standard Deviation, Minimum, and Maximum** values
- Insights into feature distributions and potential **outliers**

### 2.2 Target Variable Distribution

A histogram of **Life Expectancy** reveals that the distribution of life expectancy in the dataset is slightly right-skewed, with most values concentrated between **60 and 80 years**, peaking around **70–75 years**. A few observations fall below **50 years**, indicating potential outliers, likely from countries with lower healthcare standards or economic challenges. Despite this skewness, the data follows a roughly normal pattern, suggesting that while life expectancy is generally high, disparities exist due to various socio-economic and health-related factors.

### 2.3 Correlation Analysis

A correlation heatmap highlights the relationships between features, revealing key insights:

- **Strong positive correlation** between Life Expectancy and **Income Composition of Resources, Schooling**, and **Total Expenditure**.
- **Strong negative correlation** between Life Expectancy and **Adult Mortality**.

### 2.4 Outlier Detection and IQR Analysis

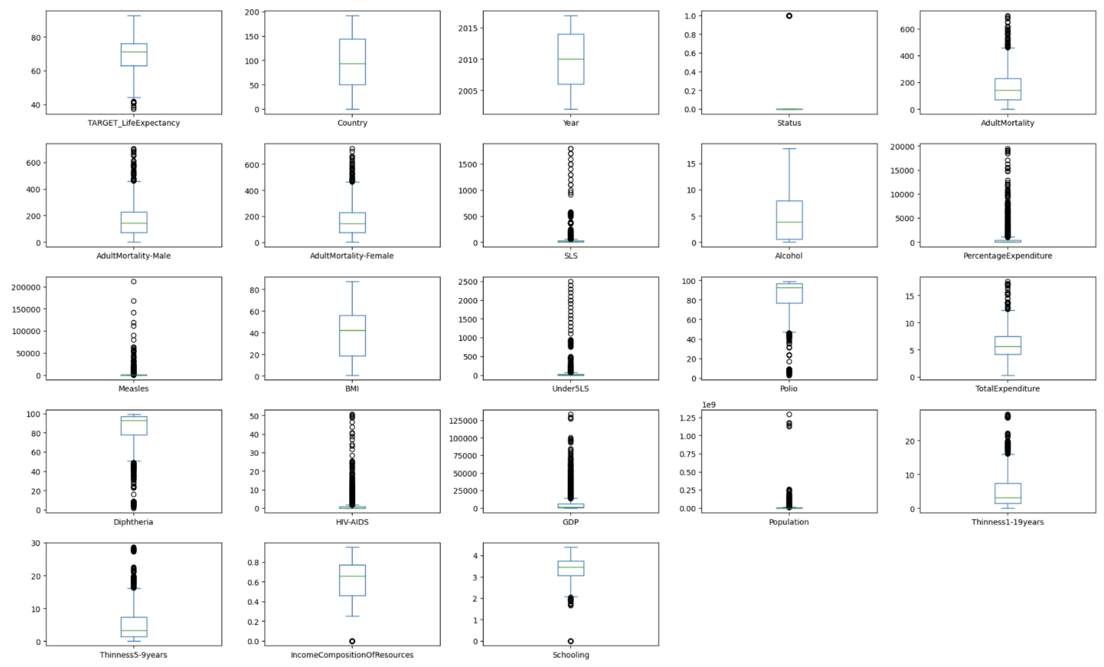


**Figure 1:** Outlier Detection with Boxplots

Box plots were generated for all numerical features to visually assess distributions and identify outliers. The analysis revealed that most features contain outliers. Using the IQR method, the number of outliers per feature was computed. Notably, **Adult Mortality**, **GDP**, and **Schooling** had significant outliers.

## 3. Data Cleaning

- **Missing values were imputed** using the median method to prevent data loss. This method ensures that extreme values do not skew the dataset, making it a robust choice for handling missing values.
- **Rows with missing values** that could not be reliably filled were removed.
- **Duplicates were checked and removed** to avoid redundancy.
- **Final dataset shape:** (2071, 23) with no missing values and no duplicates.

## 4. Data Preprocessing

In this phase, I optimize and evaluate both the training and test datasets to ensure consistent performance across unseen data. The preprocessing steps include:

### 4.1 Handling Categorical Variables

The dataset includes a high-cardinality categorical feature, **Country**, with 136 unique values that require encoding for effective model learning, while **Status** is already binary (0 and 1) and does not require further encoding.

| Method | Pros | Cons |
|---|---|---|
| **Target Encoding** | - Preserves relationship with target variable.<br>- Efficient for high-cardinality.<br>- Reduces memory usage. | - Requires careful handling to avoid overfitting.<br>- Slightly more complex than Label Encoding |
| **One-Hot Encoding (OHE)** | - Simple binary representation.<br>- No ordinal bias. | - Creates 136 new columns, leading to the curse of dimensionality.<br>- Increases sparsity (mostly zeros).<br>- Slows training. |
| **Label Encoding** | - Simple numerical conversion.<br>- No dimensionality increase. | - Introduces false ordinal relationships (e.g., "USA=1 > Germany=2").<br>- Misleads models (e.g., linear regression assumes order). |
| **Frequency Encoding** | - Captures category prevalence.<br>- No dimensionality increase. | - Ignore relationship with target.<br>- Less informative for prediction |

**Table 1:** Comparison of Categorical Encoding Methods [1]

**Justification:** Target encoding was chosen because it provides the best balance between interpretability and model performance by capturing the statistical relevance of each country to life expectancy. Unlike one-hot encoding, which creates too many new columns and leads to computational inefficiency and overfitting, target encoding remains efficient. Label encoding was rejected because assigning arbitrary numerical values can mislead the model, and frequency encoding was dismissed since the occurrence frequency of a country does not necessarily correlate with life expectancy. Therefore, target encoding offers a more meaningful and effective transformation for categorical data in this case.

## 4.2 Feature Scaling

To ensure fair comparisons between features and improve model performance, numerical features are standardized before training.

- **Benefits of Standardization:** Standardization improves model convergence by helping gradient-based models, such as Linear Regression, train faster [2]. It also prevents feature bias by ensuring no single feature dominates due to large magnitude differences [2]. Additionally, it enhances interpretability by making feature coefficients more comparable [2].
- **Method Used:** For consistency in model training, standardization is applied only to numerical features, excluding the target variable. The *StandardScaler()* method is used, scaling features to have a mean of 0 and a standard deviation of 1.

## 4.3 Outlier Handling

Outliers can significantly impact model performance by skewing statistical measures and distorting training [3]. To mitigate these effects, different techniques were applied based on feature characteristics. Handling outliers is crucial as extreme values can distort key statistics like mean and variance, leading to biased interpretations. Additionally, addressing outliers improves model generalization by reducing the risk of overfitting to anomalies, ensuring more reliable predictions.

| Method | Applied to Features | Reason |
|---|---|---|
| **Winsorization (Capping extremes)** | HIV-AIDS, Measles, PercentageExpenditure, GDP, Population, SLS, Diphtheria, Under5LS, Polio, IncomeCompositionOfResources, Schooling | Limits extreme values without removing data. |
| **Log Transformation** | GDP, Population, PercentageExpenditure, Measles | Reduces skewness in right-tailed distributions. |
| **Robust Scaling (Uses median/IQR)** | Thinness1-19years, Thinness5-9years, IncomeCompositionOfResources | Resistant to outliers while preserving variance. |

**Table 2:** Outlier Handling Techniques Used

## 5. Model Building

### 5.1 Data Splitting

Hold-out validation was chosen to ensure fair model evaluation by maintaining a separate validation set while balancing sufficient training data (80%) with reliable validation (20%).

### 5.2 Model Selection & Training

| Model | Type | Reason for Selection |
|---|---|---|
| **Linear Regression** | Linear | Baseline for simple relationships. |
| **Ridge Regression** | Linear (L2 Reg.) | Handles multicollinearity (e.g., GDP vs. Income). |
| **Lasso Regression** | Linear (L1 Reg.) | Performs feature selection. |
| **Decision Tree** | Non-linear | Captures complex patterns but prone to overfitting. |
| **Random Forest** | Ensemble (Bagging) | More robust, reduces overfitting. |
| **XGBoost** | Ensemble (Boosting) | Optimizes errors iteratively, known for handling structured data efficiently |

**Table 3:** Comparison of Regression Models for Life Expectancy Prediction

**Justification:** These models balance simplicity, interpretability, and accuracy for life expectancy estimation. Linear Regression serves as a baseline, while Ridge and Lasso handle multicollinearity and feature selection. Decision Trees capture complex patterns but may overfit, which Random Forest mitigates through ensemble learning. XGBoost, an advanced boosting technique, optimizes errors iteratively and is well-suited for structured data, making it highly effective for this regression task.

**Core Functions:**

1. ***train_model()*** – Trains models using the training set.
2. ***evaluate_model()*** – Computes key performance metrics (MAE, MSE, RMSE, R²).
3. ***train_and_evaluate()*** – Automates model training, evaluation, and visualization.

### 5.3 Evaluation

- **Performance Metrics:** To compare model effectiveness, the following metrics are used.

| Metric | Interpretation | Goal (Better if…) |
|---|---|---|
| **MAE** | Average absolute error | Lower |
| **MSE** | Penalizes large errors more | Lower |
| **RMSE** | Interpretable version of MSE | Lower |
| **R²** | Variance explained by the model | Close to 1 |

**Table 4:** Model Evaluation Metrics and Goals

- **Residual Distribution Analysis:**

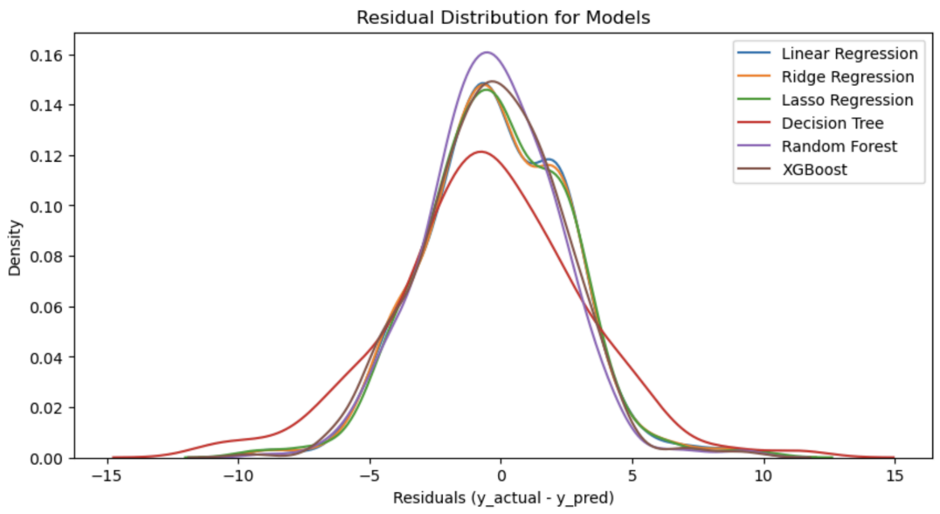


**Figure 2:** Residual Distribution Graph to assess model performance

The residuals for most models are symmetrically distributed around zero, indicating low bias. Random Forest has the narrowest spread, suggesting more consistent predictions, while the Decision Tree model shows higher variance, indicating potential overfitting. XGBoost and linear models have similar residual distributions, implying comparable performance. Overall, Random Forest appears to be the most reliable model based on residual behavior.

## 6. Hyperparameter Tuning

Hyperparameter tuning is essential for optimizing machine learning models. Since **Random Forest** and **XGBoost** demonstrated the best initial performance, they were selected for tuning.

## 6.1 Methodology

I chose **RandomizedSearchCV** over **GridSearchCV** for these reasons:

- Speeds up tuning by testing a random subset of hyperparameter combinations instead of all possible ones.
- Provides good results faster with fewer iterations (n_iter=10), making it practical for large datasets.

## 6.2 Hyperparameter Search Space

| Parameter | Values Tested | Description |
|---|---|---|
| **n_estimators** | [100, 200, 300] | Number of decision trees |
| **max_depth** | [10, 20, None] | Maximum tree depth |
| **min_samples_split** | [2, 5, 10] | Minimum samples required to split a node |

**Table 5:** Hyperparameter Search Space for Random Forest

| Parameter | Values Tested | Description |
|---|---|---|
| **n_estimators** | [100, 200, 300] | Number of boosting rounds |
| **learning_rate** | [0.01, 0.1, 0.2] | Step in size shrinkage |
| **max_depth** | [3, 6, 9] | Maximum depth of trees |

**Table 6:** Hyperparameter Search Space for XGBoost

## 6.3 Implementation

- Used **5-fold cross-validation** (cv=5) to ensure robustness.
- Optimized for **negative mean squared error (MSE)** (scoring='neg_mean_squared_error').
- Ran **10 random iterations per model** (n_iter=10) for a balance between speed and performance.

## 6.4 Results

| | |
|---|---|
| Random Forest | {'n_estimators': 300, 'max_depth': 20, 'min_samples_split': 2} |
| XGBoost | {'n_estimators': 100, 'learning_rate': 0.1, 'max_depth': 3} |

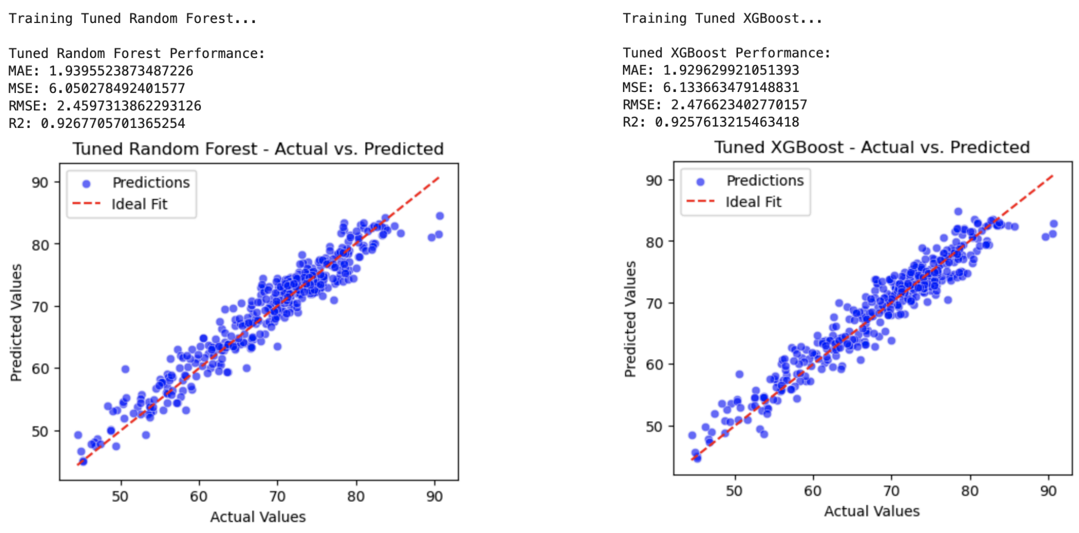
**Table 7:** Best Parameters Found for Tuned Models

## 7. Model Selection

After hyperparameter tuning, the best versions of the models were extracted using *best_estimator_*, which selects the model with the optimal parameters identified by *RandomizedSearchCV*. The tuned models, which are **Random Forest** and **XGBoost**, were stored in a dictionary for structured evaluation.

### 7.1 Performance Evaluation

The optimized models were assessed using the *train_and_evaluate* function, with performance measured through key regression metrics.



```
Training Tuned Random Forest...

Tuned Random Forest Performance:
MAE: 1.9395523873487226
MSE: 6.050278492401577
RMSE: 2.4597313862293126
R2: 0.9267705701365254
```

```
Training Tuned XGBoost...

Tuned XGBoost Performance:
MAE: 1.929629921051393
MSE: 6.133663479148831
RMSE: 2.476623402770157
R2: 0.9257613215463418
```

**Figure 3:** Performance Comparison of Tuned Models

### 7.2 Best Model Selection

To determine the best-performing model, the selection process prioritized the **highest R² score**, with **RMSE** used as a secondary criterion in case of similar R² values. The selection was carried out using a custom function that compared the models' metrics and identified the superior one.

Based on the evaluation, **Tuned Random Forest** was selected as the best model due to its **higher R² score (0.926) and lower RMSE (2.45)**, indicating strong predictive performance and better generalization.

## 8. References

[1] N.Ninja, Target Encoding: Categories Guided by Outcomes (June 14, 2023). Accessed April 10, 2025. [Blog]. Available :https://letsdatascience.com/target-encoding/
[2] mljourney, ,Importance of Feature Scaling in Machine Learning (June 15, 2014). Accessed April 10, 2025. [Blog]. Available: https://mljourney.com/importance-of-feature-scaling-in-machine-learning/
[3] F. Ododo and N. Addotey, " UNDERSTANDING THE INFLUENCE OF OUTLIERS ON MACHINE LEARNING MODEL INTERPRETABILITY", vol.7, no.2, February. 2025. doi: 10.70382/tijasdr.v07i2.019