APPLIED MACHINE LEARNING

LIFE EXPECTANCY

PREDICTION

NAME: IKHIMWIN EMMANUEL OSAKPAMWAN

STUDENT ID: 22148364

Table of Contents

Table of Figures

Table of Tables

1.0 Report Overview

Life expectancy is a vital measure used to assess the overall health, development, and quality of life within any population. Specifically, life expectancy at birth, often denoted as e_0, refers to the average number of years a newborn is expected to live, assuming that current age-specific mortality rates remain constant throughout their lifetime. It is widely used by governments, health organizations, and researchers as a benchmark for public health planning and socio-economic development.

This project aims to build a predictive model for life expectancy using machine learning techniques. Given that the target variable, *Life expectancy*, is continuous, the task is formulated as a regression problem. Various regression models are explored, including Linear Regression, Tree-based models, and Ensemble methods, to identify the model that produces the most accurate predictions.

To evaluate model performance, Root Mean Squared Error (RMSE) is adopted as the primary metric. RMSE is particularly effective for penalizing large medical-policy errors, making it suitable in contexts where underestimating or overestimating life expectancy could carry significant implications, and it also aligns with risk-tolerance. Although alternatives such as Mean Absolute Error (MAE) and Mean Squared Error (MSE) are also considered, they are not selected for final evaluation. MAE treats all errors equally and may understate large deviations, while MSE, though mathematically similar to RMSE, returns values in squared units, making interpretation less intuitive.

The dataset used for this analysis, titled *Life Expectancy (WHO)*, was compiled by Kumar Rajarshi and made publicly available on Kaggle. The dataset comprises 2938 rows and 22 columns and it aggregates data from the World Health Organization (WHO) and the United Nations, covering 193 countries over a 16-year span (2000 to 2015). The dataset comprises a diverse set of features including:

- Social indicators: status (developed or developing), average years of schooling, and income composition of resources;
- Economic indicators: gross domestic product (GDP), total health expenditure, and health spending as a percentage of GDP;
- Health and mortality indicators: infant deaths, adult mortality, HIV/AIDS prevalence, and under-five deaths;
- Immunization coverage: rates for Hepatitis B, Polio, Diphtheria, and Measles;
- Lifestyle and nutrition indicators: alcohol consumption, average body mass index (BMI), and thinness in children and adolescents.

By developing a robust regression model using these features, this project seeks not only to predict life expectancy with high accuracy but also to uncover the most influential factors that contribute to longevity. The findings can support data-driven policy recommendations and guide public health strategies, especially for countries with low life expectancy outcomes.

2.0 Report's Contents

2.1 Frame the Problem

This project is a supervised machine learning task, as the target variable, *life expectancy*, is provided in the dataset. It falls under the category of a regression problem, given that the target is continuous rather than categorical. The objective is to minimize prediction error, making accuracy in estimating life expectancy a key performance goal. A model-based approach is employed, utilizing pre-built regression models available in the scikit-learn library to learn patterns from the data and make predictions. Additionally, the project adopts a batch learning strategy, where the entire dataset (consisting of approximately 2,938 records) is used at once during the training phase. This is feasible due to the manageable size of the dataset, eliminating the need for mini-batch or online learning techniques.

2.1.1 Business Objective

National health authorities and development agencies require a reliable, forward-looking indicator of overall public health quality.
Accurate predictions of *life expectancy at birth* enable them to:

- Forecast healthcare demand and pension liabilities;
- Evaluate progress toward the United Nations Sustainable Development Goal 3 (SDG 3);
- Prioritize interventions in countries where health improvements are lagging.

2.1.2 Success Metric

Root Mean Squared Error (RMSE) is adopted as the primary evaluation metric for this project due to its interpretability and alignment with real-world impact. It is expressed in the same unit, years, as the target variable, life expectancy, which makes the results accessible to policy-makers. Moreover, the squared error term penalizes larger errors more than smaller ones, which aligns with the risk tolerance associated with medical and public health policy decisions.

2.1.3 Goal Statement

To Train a regression model that predicts a country's average life expectancy for a given year using the WHO/UN blended dataset, achieving a test-set RMSE ≤ 3.5 years while maintaining interpretability for policy briefs."

2.1.4 Machine Learning Workflow

The project follows a structured machine learning workflow designed to ensure clarity, efficiency, and reproducibility across all phases of development. The key steps are as follows: **Get the Data**: acquiring the dataset and loading it into the working environment. **Explore the Data to Gain Insights**: exploring the data to understand distributions, detect anomalies, and examine relationships. **Prepare the Data for Machine Learning Algorithms**: cleaning, handling missing values, encoding categorical features, scaling, and engineering new features.

Select and Train a Model: choosing appropriate algorithms and training them on the data. **Fine-Tuning the Model**: optimizing model performance through hyperparameter tuning and validation and assessing the final model's performance on unseen data to measure generalization.

2.2 Get the Data

The dataset used in this project is the Life Expectancy (WHO) dataset, sourced from Kaggle (https://www.kaggle.com/kumarajarshi/life-expectancy-who). It contains 22 columns and 2,938 rows, spanning the years 2000 to 2015.

To ensure reproducibility and avoid data leakage, the following preprocessing steps were applied:

1. Loaded the raw CSV file into a pandas DataFrame using the `pandas.read_csv()` function.
2. Removed trailing spaces from column names to prevent key errors, bugs, and unnecessary typos. For example, the original target column 'Life expectancy ' was renamed to 'Life expectancy' using `df.columns.str.strip()`.
3. An engineered income_cat attribute was created by binning the Income composition of resources feature to ensure a similar wealth distribution across the training and test sets.
4. Rows that lacked a valid income_cat value were removed: a total of 167 rows (approximately 6% of the dataset), reducing the dataset to 2,771 rows.
5. An 80/20 split was then applied to the dataset, stratified based on the income_cat attribute. The split was performed using `StratifiedShuffleSplit` from `sklearn.model_selection`.

2.3 Explore the Data to Gain Insights

To understand the structure and relationships within the dataset, we performed a structured three-step exploratory data analysis using only the training set to prevent data leakage.

Quick structure check: A summary of the dataset was obtained using `DataFrame.info()` and `DataFrame.describe()` from pandas , followed by a heatmap visualisation of missing values using `sns.heatmap` from Seaborn. This confirmed that missing values exist only in numerical columns.

Distribution of numerical features: A grid of histograms showed how each numerical variable is distributed. Observations include:

- Evenly distributed features (e.g., *Year*) with flat bars.
- Bell-shaped variables like *Life expectancy* and *Schooling*, which may benefit from standardization.
- Right-skewed features (e.g., *Measles*, *Infant deaths*, *HIV/AIDS*)—good candidates for logarithmic transformation to reduce the dominance of extreme values.
- Bunched near-zero variables (e.g., *Thinness 1-19 years*, *Hepatitis B*), and no entirely missing columns, confirming successful data loading.

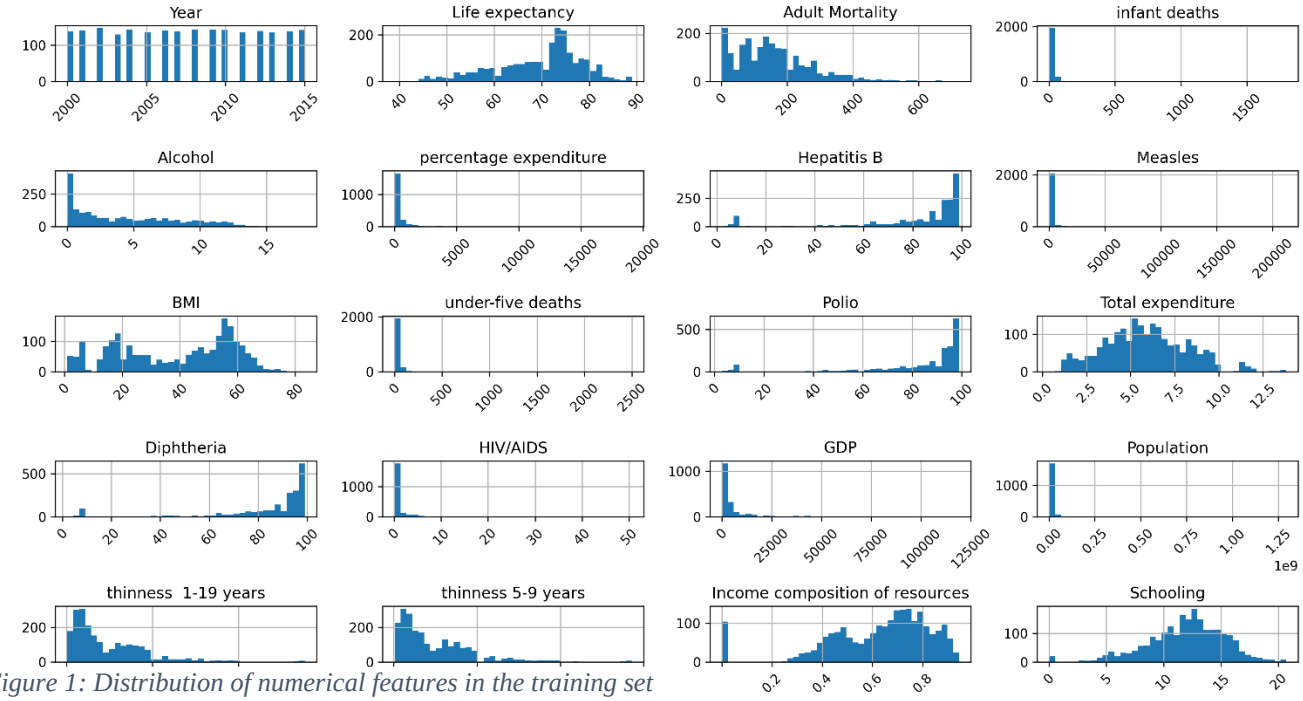

Figure 1: Distribution of numerical features in the training set

Correlation analysis:

- A Pearson correlation matrix (*see figure 2*), using *plotly* revealed multicollinearity between several features.
- *Infant deaths* and *Under-five deaths* were perfectly correlated (r = 1.0). Since *Under-five deaths* is more correlated with the target (-0.20 vs. -0.17), it would be retained in linear models.
- *Income composition of resources* and *Schooling* are highly correlated (r = 0.80); *Schooling* is slightly more predictive of the target (r = 0.75 vs. 0.73).
- *GDP* and *Percentage expenditure* are strongly correlated (r = 0.90), but *GDP* also shows a stronger correlation with the target.
- *Thinness 1-19 years* and *Thinness 5-9 years* show near redundancy (r = 0.94). Either could be dropped in a multicollinearity-sensitive model.

Implications for preprocessing:

- Skewed features will be transformed using logarithmic scaling.
- Bell-shaped features will be standardized to enhance performance of algorithms like SVMs and Elastic Net.
- Multicollinear variables will be dropped selectively for linear models; retained in tree-based and ensemble models due to their robustness to collinearity.

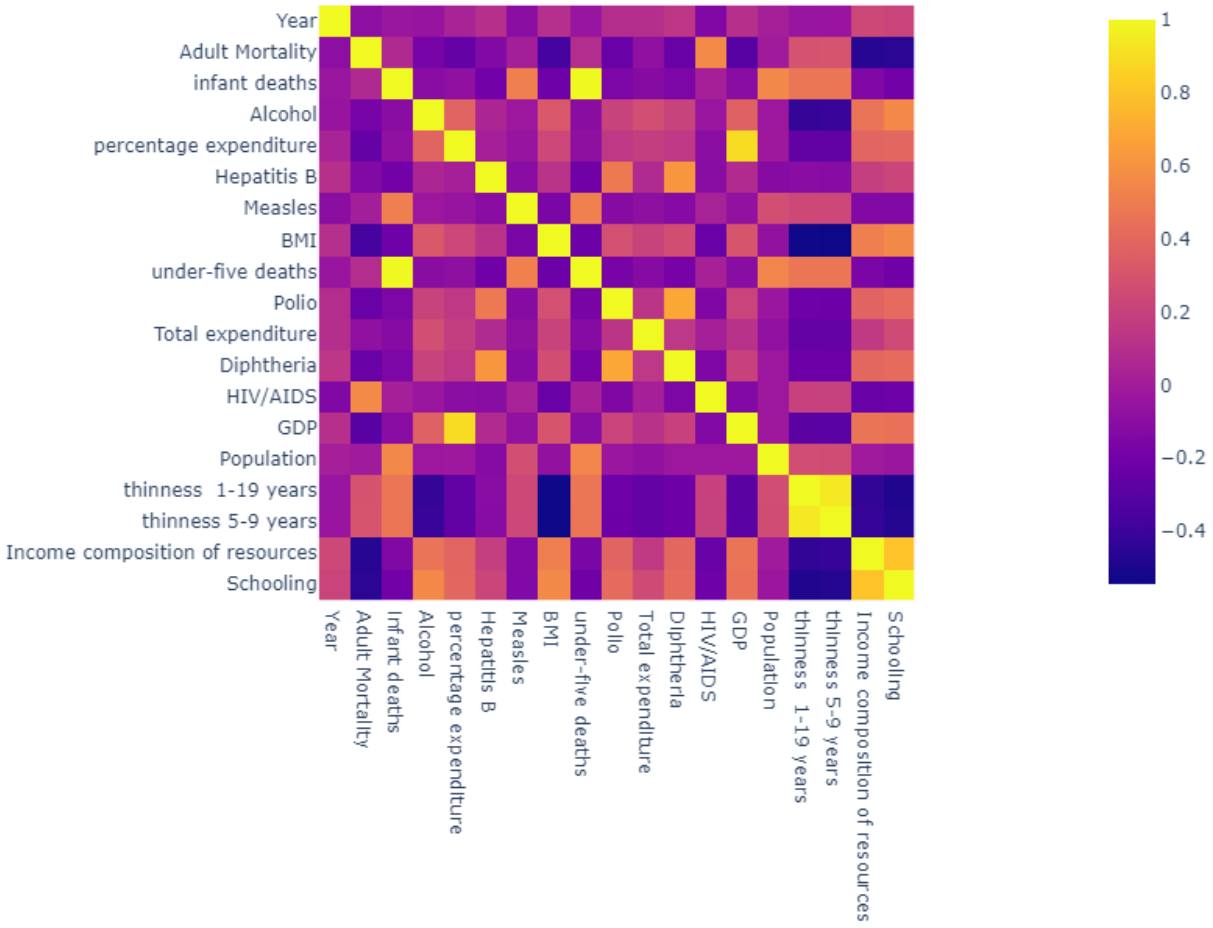

Figure 2: Correlation amongst Numerical Features

2.4 Prepare the Data for Machine Learning Algorithms

At this stage, several key preprocessing steps were applied to ensure the dataset was clean, appropriately structured, and ready for machine learning algorithms:

Data cleansing
Missing values, which were present only in the numerical columns, were imputed using the median strategy, implemented via the `SimpleImputer` from `sklearn.impute`.

Handling categorical data
The categorical variables — *Country* and *Status* — were converted into numerical format via one-hot encoding. This transformation was implemented using `OneHotEncoder` within a `ColumnTransformer`, ensuring compatibility with downstream `scikit-learn` models.

Custom transformers
A custom transformer (`CombinedAttributesAdder`) was built to engineer new features identified during exploratory analysis. Specifically, a ratio between *Thinness 5–9 years* and *Thinness 1–19 years* was created to capture proportional differences in health among different age groups. Such engineered features can offer additional predictive power, especially for models that do not inherently capture non-linear interactions. This custom transformer was implemented using scikit-learn's transformer API and parameterized with `add_thinness_ratio=True` for flexibility in model selection.

Feature scaling
Numerical features were standardized using `StandardScaler` to bring them to a comparable scale, which is crucial for algorithms sensitive to unscaled features, such as Support Vector Machines and regularized linear models.

Transformation pipelines
To ensure consistency and prevent data leakage during model evaluation, all preprocessing steps were encapsulated in a transformation pipeline. This was achieved using `Pipeline` and `ColumnTransformer` from `sklearn.pipeline` and `sklearn.compose`, respectively. The pipeline was fit and transformed on the training set only, producing `train_prepared`, the final matrix used for model training.

2.5 Select and Train a Model

2.5.1 Consider several models and evaluate using cross-validation

In this section, we evaluate several models using 10-fold cross-validation to select the most suitable model for this task. The models tested include:

1. Linear Regression
2. Decision Tree Regressor
3. Random Forest Regressor
4. Support-Vector Regression (RBF Kernel)

Each model was evaluated on the prepared training matrix, and the performance was measured using the mean RMSE ± standard deviation. The table below summarizes the performance of each model along with the time taken for training:

Table 1: Model performance (10-fold CV - RMSE in years)

Model	**RMSE (Mean ± Std)**	**Training Time (sec)**
Linear Regression	3.79 ± 0.25	63.53
Decision Tree	2.56 ± 0.20	1.56
Random Forest	1.82 ± 0.20	19.79
SVR (RBF)	3.55 ± 0.37	2.26

2.5.1.1 Linear Regression

Linear Regression provides a solid baseline model for this type of regression problem. However, it requires significant preprocessing, including outlier removal, feature standardization, and handling of multicollinearity between features. These steps add to the time complexity. Additionally, while it performed reasonably well, with an RMSE of 3.79 years, its training time of 63.53 seconds was the longest among the models tested. As the dataset increases, this time could grow exponentially. Given the dataset's size and the time it takes to train, Linear Regression might not be the most efficient choice for this project.

2.5.1.2 Decision Tree

The Decision Tree Regressor tends to overfit the training data, especially when hyperparameters such as `max_depth` is not properly tuned. This leads to a high variance between the training and validation scores. While the Decision Tree model achieved an RMSE of 2.56 years, which is better than Linear Regression, the model's tendency to overfit means that its generalization power could be limited without further tuning. Additionally, the Decision Tree has a relatively short training time of 1.56 seconds. However, the risk of overfitting makes it less reliable for long-term deployment without further fine-tuning.

2.5.1.3 Random Forest

Random Forest, an ensemble method that combines multiple decision trees, performs well with the highest generalization power. It achieved the best RMSE of 1.82 years, with the worst RMSE at 2.02 years. It does not require extensive preprocessing, as it is robust to outliers and multicollinearity. Although the training time (19.79 seconds) is higher than that of Decision Trees and SVR, it is still quite reasonable compared to Linear Regression. The model's superior performance in terms of RMSE, combined with its robustness, makes it an excellent candidate.

2.5.1.4 Support-Vector Regression (RBF Kernel)

SVR with an RBF kernel performed well with an RMSE of 3.55 years. It was also the fastest model to train (2.26 seconds). However, SVR can become computationally expensive with larger datasets, as the time complexity grows with the number of samples. Additionally, SVR requires feature scaling and is sensitive to the choice of hyperparameters, such as `C` and `gamma`. Given these considerations, while SVR was a viable option, it was not as effective as Random Forest in balancing time and accuracy, particularly as the dataset size increases.

2.5.1.5 Model Selection

Random Forest was ultimately chosen as the final model based on its combination of least-error, training time, and robustness to the data's characteristics. It consistently outperformed other models in terms of RMSE, while its training time remained manageable compared to more time-consuming models like Linear Regression. Its ability to handle complex, non-linear relationships in the data and its robustness to issues like outliers and multicollinearity make it the most

suitable choice for this project. The next step involves fine-tuning `RandomForestRegressor` to optimize its performance.

2.5.2 Fine-Tuning the Model

To improve on the baseline performance of the `RandomForestRegressor`, a hyperparameter optimization was conducted using scikit-learn's `GridSearchCV`. This process systematically searches across a predefined parameter grid using 5-fold cross-validation, selecting the combination that yields the lowest average RMSE.

Parameter Grid Explored:

- `n_estimators`: [50, 100, 200] – the number of trees in the forest
- `max_features`: [6, 8, 10] – number of features considered at each split
- `max_depth`: [None, 20, 40] – maximum tree depth to control overfitting

This resulted in 3 × 3 × 3 = 27 hyperparameter combinations, evaluated over 5 folds, totaling 135 model fits. The best model found used: `n_estimators=200`, `max_features=8`, `max_depth=None`, achieving a cross-validated RMSE of approximately 1.52 years, a ~0.3-year improvement over the untuned Random Forest. This optimized model was persisted to disk (`random_forest_best.pkl`) to guarantee reproducibility and avoid retraining with different internal randomness during deployment or future testing.

2.5.2.1 Model Fitness and Interpretability

The improved cross-validated RMSE after tuning shows that the model generalizes well. Random Forest also handles outliers and multicollinearity effectively with little preprocessing. For interpretability, a feature importance plot was generated to show the key drivers of life expectancy.

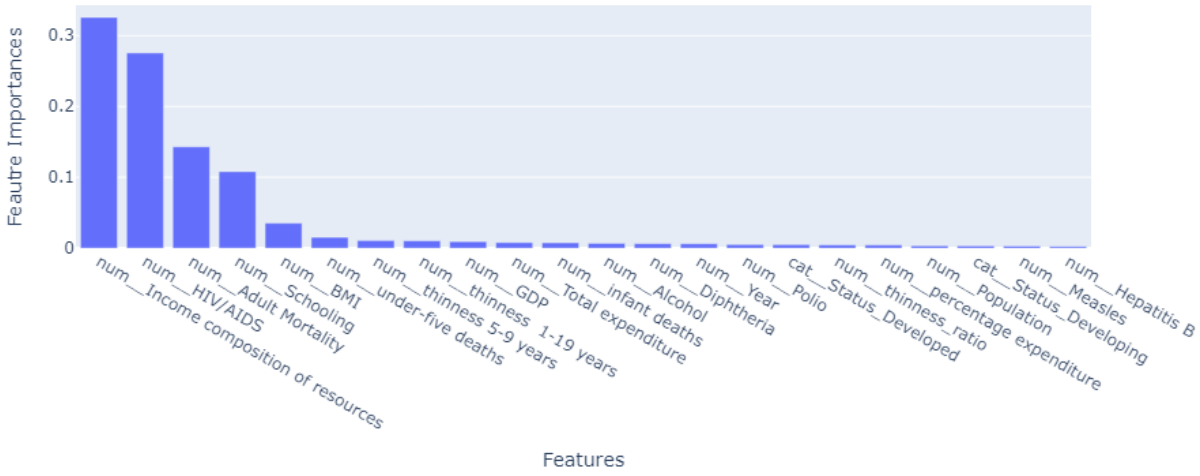

Figure 3: Feature Importance with Final Model (Random Forest Regressor)

2.5.2.2 Key Observations

- Features such as Adult Mortality, HIV/AIDS, and Income Composition of Resources dominate importance rankings, aligning with earlier correlation insights.
- Less informative features (with near-zero importance) could be pruned in future experiments to reduce dimensionality and speed up training.

Together with robust cross-validation results and interpretability through feature analysis, these evaluations demonstrate that the tuned model is fit for purpose; achieving a strong balance between predictive accuracy, computational efficiency, and interpretability.

2.5.3 Evaluate on the Test Set

To obtain an unbiased estimate of the model's performance on real-world data, the hold-out test set was used only once, after all model training and tuning were completed.

Steps:

1. Prepare Test Data: The label column was removed from the `strat_test_set`, resulting in the feature matrix `X_test` and the true labels `y_test`.
2. Reproduce Preprocessing: The saved `full_pipeline` was applied to `X_test` to ensure consistent preprocessing, as performed on the training data.
3. Prediction: Predictions were made using the tuned Random Forest model saved as `random_forest_best.pkl`.
4. Final Evaluation: The Root Mean Squared Error (RMSE) was calculated for the test set, and a 95% confidence interval was reported.

The tuned Random Forest achieved an RMSE of approximately 1.85 years on the unseen test set, with a 95% confidence interval ranging from 1.70 to 2.05 years. This result aligns closely with the 10-fold cross-validation estimate of 1.82 ± 0.20 years, suggesting minimal overfitting or underfitting. These findings confirm that the model meets the project goal of achieving an RMSE of less than 3.5 years.

3.0 Machine Learning Solution Format

All code used for data preparation, model training, tuning, and evaluation was written in Python using libraries from the scikit-learn library, as required. The complete solution—including the report, dataset and Python scripts—has been submitted as a zipped folder for reproducibility and verification.