A Comprehensive Study of Machine Learning Models for Predicting Life Expectancy

Adel Ahmadi

January 20, 2025

Abstract

Predicting life expectancy is a critical task with applications in public health planning and policy-making. This study explores machine learning approaches, including Support Vector Machines (SVM), Decision Trees (DT), Random Forests (RF), and ensemble methods like Bagging and Boosting, for classifying life expectancy into low, medium, and high categories based on socio-economic and health indicators. The dataset comprises key features such as GDP, schooling, and mortality rates. Comprehensive evaluations using hyperparameter tuning, classification reports, and confusion matrices identify optimal models. The findings contribute to advancing predictive modeling in healthcare analytics.

1 Introduction

1.1 Problem Statement

Life expectancy is a critical indicator of a population's overall health. Accurately predicting life expectancy allows policymakers and healthcare organizations to allocate resources effectively and track the impact of health-related initiatives. However, the complex interplay of socio-economic, environmental, and healthcare factors makes this prediction task challenging. This study aims to leverage machine learning techniques to classify life expectancy into low, medium, and high categories using a cleaned dataset of global health and socio-economic indicators.

1.2 Importance and Relevance

Predicting life expectancy is a practical solution for addressing global health disparities. Countries with lower life expectancy often face challenges such as poor healthcare infrastructure, inadequate education, and economic instability. Machine learning models can offer a data-driven approach to identifying key factors influencing life expectancy and predicting accurate outcomes.

1.3 Dataset Overview

The dataset used in this study is sourced from the publicly available Cleaned Life Expectancy Dataset, which contains observations from multiple countries over several years. Key features include Gross Domestic Product (GDP), adult mortality rate, schooling, and other socio-economic and healthcare indicators. Countries with a ton of missing values have been completely removed from the dataset, and countries with a small amount of missing values have been predicted by a machine learning model. As such, there are no missing values. The dataset has already been scaled and cleaned.

Table 1: Summary of Features in the Dataset

Feature	**Description**	**Type**
Country	Name of the country	Categorical
Year	Year of the observation	Discrete
Life Expectancy	Average number of years a person is expected to live	Continuous
Adult Mortality	Probability of dying between ages 15 and 60	Continuous
Infant Deaths	Number of infant deaths per 1,000 live births	Continuous
Alcohol Consumption	Average alcohol consumption per capita (liters)	Continuous
Percentage Expenditure	Percentage of GDP spent on health	Continuous
Hepatitis B Immunization	Percentage of 1-year-olds immunized against Hepatitis B	Continuous
Measles Immunization	Number of reported measles cases per 1,000 live births	Continuous
BMI	Average Body Mass Index	Continuous
Under-five Deaths	Number of deaths of children under five per 1,000 live births	Continuous
Polio Immunization	Percentage of 1-year-olds immunized against polio	Continuous
Total Expenditure	Total expenditure on health as percentage of GDP	Continuous
Diphtheria Immunization	Percentage of 1-year-olds immunized against diphtheria	Continuous
HIV/AIDS Deaths	Number of deaths due to HIV/AIDS per 1,000	Continuous
GDP	Gross Domestic Product per capita (USD)	Continuous
Population	Population of the country	Continuous
Schooling	Average years of schooling	Continuous

1.4 Research Objectives

The primary objectives of this study are as follows:

- To evaluate the performance of various machine learning algorithms, including Support Vector Machines, Decision Trees, Random Forests, and ensemble methods such as Bagging and Boosting, for predicting life expectancy categories.
- To identify key features that contribute most significantly to the prediction task.
- To compare model performance using metrics such as accuracy, precision, recall, and F1-score, while also analyzing confusion matrices for deeper insights.
- To propose the best-performing model for predicting life expectancy based on extensive hyper-parameter tuning and validation.

2 Methodology

This section outlines the methodologies employed in this study to classify life expectancy into three (Low, Medium and High) categories. We experimented with various machine learning models, including both individual classifiers and ensemble techniques. To enhance model performance. In this section, we provide a mathematical overview of each machine learning model used, explain their significance in classification tasks.

2.1 Machine Learning Models

2.1.1 Support Vector Machine (SVM)

Support Vector Machines (SVM) is a powerful supervised learning algorithm primarily used for classification tasks, similar to this paper's idea. The core idea of SVM is to find an optimal hyperplane that separates data points of different classes in a high-dimensional space, maximizing the margin between them. This margin ensures better generalization for unseen data in test data of the dataset.

Mathematics of SVM:

1. Hyperplane Equation: The decision boundary in SVM is defined as:

$$w \cdot x + b = 0$$

   where:

   - w: Weight vector (determines the orientation of the hyperplane).
   - x: Input feature vector.
   - b: Bias term (shifts the hyperplane).

2. Optimization for Linearly Separable Data: SVM aims to maximize the margin $\frac{1}{\|w\|}$ while satisfying:

$$y_i(w \cdot x_i + b) \geq 1, \quad \forall i$$

   Here, $y_i \in \{-1, 1\}$ are the class labels. The optimization problem becomes:

$$\text{Minimize: } \frac{1}{2}\|w\|^2$$

3. Soft Margin SVM: For non-linearly separable data, SVM introduces slack variables ξ_i to allow some misclassifications:

$$\text{Minimize: } \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{n}\xi_i$$

   subject to:

$$y_i(w \cdot x_i + b) \geq 1 - \xi_i, \quad \xi_i \geq 0$$

   C is a regularization parameter that balances margin maximization and misclassification penalties.

4. Kernel Trick: SVM can handle non-linear decision boundaries using kernel functions, which map the input data into higher-dimensional spaces:

$$K(x_i, x_j) = \phi(x_i) \cdot \phi(x_j)$$

   Common kernels include linear, polynomial, and Radial Basis Function (RBF).

5. Decision Function: The decision function of SVM is expressed as:

$$f(x) = \text{sign}\left(\sum_{i=1}^{n}\alpha_i y_i K(x_i, x) + b\right)$$

   where α_i are the support vector coefficients.

SVM Hyperparameters:

1. C: Controls the trade-off between margin maximization and misclassification. A small C emphasizes a wider margin, while a large C focuses on classifying all training points correctly.

2. Kernel: Determines the type of decision boundary. Common options are linear, polynomial, and RBF.

3. Gamma: Defines how far the influence of a single data point extends in RBF and polynomial kernels.

Implementation Steps:

1. Data Preprocessing:
   - The dataset was cleaned, scaled, and preprocessed.
   - The continuous target variable (life expectancy) was binned into three categories: low, medium, and high.
   - Categorical variables such as country names were encoded using one-hot encoding.

2. Model Training:
   - The SVM classifier was trained using the Scikit-learn library in Python.
   - A grid search with cross-validation was conducted to tune hyperparameters such as C, gamma, and kernel.

3. Hyper Parameter Tuning:
   - A Grid Search Algorithm was applied to identify optimal value of each hyper parameter, improving model interpretability and performance.

4. Model Evaluation:
   - Classification metrics such as precision, recall, and F1-score were used to assess model performance. - Confusion matrices were analyzed to understand the classification accuracy across all life expectancy categories.

2.1.2 Decision Tree (DT)

Decision Tree (DT) is a widely used supervised learning algorithm for both classification and regression tasks. Its simplicity and interpretability make it a popular choice in real-world applications. A decision tree structures the data by recursively splitting features into subsets based on feature values. Each internal node represents a decision rule, while each leaf node represents a class label or a continuous value in regression tasks.

Mathematics of Decision Tree:

- Splitting Criterion:
  Decision trees use a splitting criterion to decide how to partition the dataset at each node. Common criteria include:
  - **Gini Index**: Measures the impurity of a node. A lower Gini Index indicates purer splits.

    $$Gini = 1 - \sum_{i=1}^{C} p_i^2$$

    where p_i is the proportion of samples belonging to class i.
  - **Entropy**: Measures information gain for splitting:

    $$Entropy = -\sum_{i=1}^{C} p_i \log_2(p_i)$$

- Tree Construction:
  A decision tree is built by recursively splitting the data based on the splitting criterion, aiming to minimize impurity at each step. The process terminates when:
  - A maximum tree depth is reached.
  - A minimum number of samples per node is satisfied.
  - Further splitting does not reduce impurity.

- Pruning:
  To prevent overfitting, trees can be pruned by removing branches that have little significance in improving classification accuracy. Pruning reduces model complexity and improves generalization.

Hyperparameters in Decision Trees:

1. Criterion: Defines the splitting criterion (e.g., *gini* or *entropy*)

2. Max Depth: Limits the maximum depth of the tree, controlling the model's complexity.

3. Min Samples Split: Specifies the minimum number of samples required to split an internal node.

4. Splitter: Determines the strategy for choosing the split at each node (*best* or *random*).

Implementation Steps:

1. Data Preprocessing:
   - The dataset was cleaned and preprocessed as described earlier.
   - The continuous target variable (life expectancy) was categorized into three classes: low, medium, and high.
   - Categorical features such as *Country* were encoded using one-hot encoding.

2. Feature Scaling:
   While decision trees are not sensitive to feature scaling, scaling was applied during preprocessing to maintain consistency with other models.

3. Model Definition:
   A decision tree classifier was defined with initial hyperparameters, including *criterion*, *max depth*, and *min samples leaf.*

4. Hyperparameter Tuning:
   GridSearchCV was used to perform cross-validation across a range of hyperparameter combinations. The optimal values for parameters such as *max depth* and *min samples split* were identified.

5. Training and Evaluation:
   After selecting the best hyperparameters, the model was trained on the preprocessed dataset. Classification metrics such as precision, recall, Confusion matrix and F1-score were computed to assess performance. This ensures that the classifier effectively distinguishes between the three life expectancy categories without overfitting.

2.1.3 Random Forest (RF)

Random Forest (RF) is an ensemble learning method that combines the predictions of multiple decision trees to improve accuracy and reduce overfitting. (Unlike a single decision tree, which can be prone to high variance)

Key Concepts of Random Forest:

1. Bagging:
   Random Forest uses the bagging technique, where each decision tree is trained on a random subset of the data sampled with replacement. This ensures that each tree is exposed to slightly different data, improving model robustness.

2. Feature Randomness:
   At each split, Random Forest selects a random subset of features instead of considering all features. This introduces diversity among the trees and reduces the risk of overfitting.

3. Aggregation:
   For classification tasks, the final prediction is made using majority voting across all decision trees. For regression, the average of predictions is used.

Hyperparameters in Random Forest:
While Random Forest shares many hyperparameters with Decision Trees (e.g., *max depth*, *min samples leaf*), it introduces additional parameters:

1. Number of Trees:
   Specifies the number of decision trees in the forest.

2. Max Features:
   Limits the number of features considered for splitting at each node.

Implementation Steps:

1. Data Preprocessing:
   Similar to Decision Trees, the dataset was preprocessed, with the target variable categorized into three classes (low, medium, high) and categorical variables encoded.

2. Model Definition:
   A Random Forest classifier was defined, specifying the number of trees, maximum depth, and other hyperparameters.

3. Hyperparameter Tuning: GridSearchCV was used to tune parameters such as *n estimators* (number of trees), *max features*, and *max depth* to identify the best configuration.

4. Training and Evaluation:
   The model was trained on the preprocessed dataset, and its performance was evaluated using metrics such as precision, recall, and F1-score.

Random Forest's ensemble approach makes it a powerful and versatile model for predicting life expectancy categories.

2.1.4 Ensemble Learning: Bagging

Bagging, short for (Bootstrap Aggregating), is an ensemble learning method designed to enhance model stability and accuracy by reducing variance and preventing overfitting. The core idea of bagging is to train multiple base models on different bootstrapped subsets of the dataset (i.e., sampling with replacement). Each model produces a prediction, and the final output is determined by majority voting or averaging.

Bagging Implementation:

1. Base Learner:
   For this study, we used the (BaggingClassifier) with Support Vector Machines (SVM) as the base learners. Six SVM models were trained in parallel, each on a unique subset of the dataset.

2. Model Aggregation:
   After training, predictions from all six SVM models were aggregated using majority voting to classify life expectancy into the three categories.

3. Advantages of Bagging:
   - Reduces model variance by averaging predictions across multiple models.
   - Works effectively with high-variance models, such as decision trees or SVM, by stabilizing their predictions.

Hyperparameters of Bagging:

1. Number of Base Learners: Defines the number of base models. A value of 6 was chosen for this study.

2. Max Samples: Determines the proportion of the dataset each base model is trained on. Adjusting this affects diversity among models.

3. Bootstrap Sampling: Ensures sampling with replacement to create diverse training subsets.

2.1.5 Ensemble Learning: Boosting

Boosting is an ensemble technique that converts a collection of weak learners into a strong and accurate model. Unlike bagging, which trains models independently, boosting trains models sequentially. Each subsequent model focuses on correcting errors made by its predecessor. This iterative process helps reduce bias and variance, improving overall performance.

Boosting Implementation

1. Base Learner:
   For this study, the AdaBoostClassifier was employed, with Decision Trees as base learners.

2. Model Aggregation:
   In AdaBoost, predictions from all weak learners are combined into a weighted sum, where the weights are determined by the accuracy of each model. This ensures that models with better performance have a stronger influence on the final prediction.

Hyperparameters of Boosting:

1. Number of Base Learners: Defines the number of weak learners.
2. Learning Rate: Controls the contribution of each model to the final prediction.
3. Tree Depth: Ensures that each weak learner remains simple to prevent overfitting.

Boosting effectively enhanced the predictive capability of Decision Trees, making it a robust choice for classification tasks.

2.2 Feature Engineering and Data Preprocessing

The dataset used in this study was already cleaned, with missing values imputed and inconsistencies resolved. However, additional preprocessing steps were performed to prepare the data for machine learning models and optimize performance:

1. Feature Scaling:
   To ensure consistent contributions from features, all continuous variables were scaled to have a mean of 0 and a standard deviation of 1.
2. Target Variable Transformation:
   The continuous target variable, life expectancy, was binned into three categories:
   - Low: For life expectancy values in the lower range.
   - Medium: For values in the middle range.
   - High: For values in the upper range.

   This binning process simplifies the prediction task.
3. Categorical Feature Encoding:
   Categorical columns such as `Country` were one-hot encoded. This method converts categorical variables into binary indicator variables, ensuring they are represented in a format suitable for machine learning algorithms.
4. Data Splitting:
   The dataset was split into training and testing subsets:
   - 80% for Training: Used to train and tune the models.
   - 20% for Testing: Held out for evaluating the generalization performance of the models.

These preprocessing steps ensured that the dataset was optimized for analysis while maintaining data integrity. Proper scaling, transformation, and encoding played a critical role in achieving accurate and reliable predictions for life expectancy categories.

3 Results

3.1 Hyper Parameter Tuning Results

3.1.1 Introduction to Hyperparameter Tuning

Hyperparameter tuning is a critical step in machine learning model development, as it directly impacts the model's performance and ability to generalize. Hyperparameters are configuration parameters that define the model's structure or the learning process, such as the depth of a Decision Tree, the regularization parameter in SVM, or the number of estimators in ensemble methods. Unlike model parameters, which are learned from data, hyperparameters are set prior to training and require optimization to achieve the best possible results.

3.1.2 Techniques Used

In this study, we employed two widely-used techniques for hyperparameter optimization: Grid Search and Genetic Algorithms (GA). Each technique has unique advantages and was applied based on the complexity of the hyperparameter space for different algorithms.

1. Grid Search:
   This method is a systematic approach that evaluates all possible combinations of predefined hyperparameter values. For example, in SVM, we used Grid Search to explore combinations of hyperparameters such as the regularization parameter C, kernel types, and *gamma*. Cross-validation was employed at each combination to ensure model generalization and avoid overfitting. Although computationally expensive for large hyperparameter spaces, Grid Search provides precise results and guarantees finding the optimal configuration within the specified grid.

2. Genetic Algorithms (GA):
   GA is an evolutionary optimization technique inspired by the process of natural selection. GA is particularly useful for exploring large and complex hyperparameter spaces, where Grid Search may become computationally prohibitive. GA starts with a population of random hyperparameter combinations and iteratively evolves them using operations such as selection, crossover, and mutation. The fitness function evaluates each candidate based on model performance metrics such as accuracy or F1-score. This approach allowed for efficient exploration of the search space, yielding near-optimal solutions within a reasonable time frame.

3.1.3 Benefits and Challenges

Both Grid Search and Genetic Algorithms offer unique benefits and pose certain challenges during hyperparameter tuning:

- Benefits:
  - Grid Search ensures precise optimization for smaller and simpler hyperparameter spaces.
  - Genetic Algorithms are highly efficient for large, complex spaces, allowing the discovery of near-optimal solutions with fewer evaluations.
  - Both techniques significantly improve model accuracy, generalization, and robustness.
- Challenges:
  - Grid Search is computationally expensive, especially for models with many hyperparameters or large datasets.
  - GA requires careful selection of parameters such as population size, mutation rate, and crossover probability to avoid local minima or premature convergence.
  - Both methods are time-intensive and require computational resources.

In conclusion, the combination of Grid Search and Genetic Algorithms provided an effective and efficient framework for hyperparameter optimization.

3.2 Model Performance Evaluation

In this section, we evaluate the performance of each machine learning model using metrics such as precision, recall, F1-score, and accuracy. The evaluation metrics are derived from classification reports for each model.

3.2.1 Support Vector Machine (SVM)

The SVM model demonstrated excellent performance across all life expectancy categories. The overall accuracy was **92%**, with precision, recall, and F1-scores being consistently high. The macro-average F1-score was **91%**, highlighting its balanced performance across classes.

- **High Life Expectancy:** Precision of 93%, recall of 91%, and F1-score of 92%.

- **Low Life Expectancy:** Precision of 86%, recall of 90%, and F1-score of 88%.
- **Medium Life Expectancy:** Precision of 92%, recall of 93%, and F1-score of 92%.

The results suggest that SVM effectively distinguishes between life expectancy categories, particularly excelling in the medium and high categories.

3.2.2 Decision Tree (DT)

The Decision Tree classifier achieved an overall accuracy of **86%**. While it performed well in general, there was some imbalance in the recall for the low life expectancy category.

- **High Life Expectancy:** Precision of 90%, recall of 85%, and F1-score of 88%.
- **Low Life Expectancy:** Precision of 91%, recall of 65%, and F1-score of 76%.
- **Medium Life Expectancy:** Precision of 82%, recall of 92%, and F1-score of 87%.

The Decision Tree classifier demonstrated strong performance but was slightly prone to overfitting, as indicated by the disparity in recall for the low life expectancy class.

3.2.3 Random Forest (RF)

The Random Forest model outperformed the Decision Tree classifier, achieving an overall accuracy of **92%**. It exhibited robust performance across all classes with minimal variance in metrics.

- **High Life Expectancy:** Precision of 93%, recall of 92%, and F1-score of 93%.
- **Low Life Expectancy:** Precision of 95%, recall of 81%, and F1-score of 88%.
- **Medium Life Expectancy:** Precision of 90%, recall of 94%, and F1-score of 92%.

The introduction of bagging and feature randomness in the Random Forest method helped mitigate overfitting, making it a more generalized model than a single Decision Tree.

3.2.4 Bagging

The BaggingClassifier achieved an accuracy of **86%**, similar to the Decision Tree classifier. The use of multiple SVM models as base learners helped improve stability, although it still faced challenges in the low life expectancy category.

- **High Life Expectancy:** Precision of 90%, recall of 85%, and F1-score of 88%.
- **Low Life Expectancy:** Precision of 91%, recall of 65%, and F1-score of 76%.
- **Medium Life Expectancy:** Precision of 82%, recall of 92%, and F1-score of 87%.

Bagging proved effective in reducing variance but did not outperform Random Forest or SVM.

3.2.5 Boosting

The AdaBoost classifier achieved an accuracy of **88%**, improving over Bagging and Decision Tree models. Its iterative learning process allowed it to correct errors from weak learners, resulting in higher precision and recall for the low life expectancy category.

- **High Life Expectancy:** Precision of 86%, recall of 90%, and F1-score of 88%.
- **Low Life Expectancy:** Precision of 93%, recall of 83%, and F1-score of 87%.
- **Medium Life Expectancy:** Precision of 88%, recall of 87%, and F1-score of 88%.

The results demonstrate the ability of Boosting to address class imbalances and enhance model performance over a single Decision Tree.

4 Comparative Analysis and Final Model Selection

In this section, we compare the performance of all the models evaluated in the previous section and select the best-performing model based on accuracy, precision, recall, and overall robustness. A comprehensive analysis highlights the relative strengths and weaknesses of each approach, culminating in the selection of the most suitable model for predicting life expectancy categories.

4.1 Comparative Analysis of Models

- **Support Vector Machine (SVM):** The SVM model achieved an overall accuracy of **92%** and demonstrated consistently high precision, recall, and F1-scores across all categories. Its macro-average F1-score of **91%** reflects its balanced performance. SVM excelled particularly in distinguishing between medium and high life expectancy classes, with minimal misclassifications.
- **Random Forest (RF):** The Random Forest model also achieved an overall accuracy of **92%**, matching the performance of SVM. Its macro-average F1-score was slightly lower (**91%**), and while it handled class imbalances effectively, its recall for the low life expectancy category (**81%**) was slightly lower compared to its performance on other classes.
- **Boosting (AdaBoost):** The Boosting model achieved an accuracy of **88%**, with strong performance in the low life expectancy category (**Precision: 93%**). Its ability to iteratively correct errors from weak learners makes it a robust model, particularly for imbalanced datasets. However, it fell short of SVM and Random Forest in overall accuracy and consistency across classes.
- **Bagging:** The BaggingClassifier achieved an accuracy of **86%**, similar to the Decision Tree model. Its strength lies in reducing variance through the ensemble of multiple SVM models, but it did not significantly outperform other ensemble methods, particularly for the low life expectancy class.
- **Decision Tree (DT):** The Decision Tree classifier achieved an accuracy of **86%**. While it is interpretable and straightforward, its susceptibility to overfitting resulted in imbalanced performance, especially in the low life expectancy category, where recall was only **65%**.

4.1.1 Final Model Selection

Based on the comparative analysis, the **Support Vector Machine (SVM)** model is selected as the final model for life expectancy prediction. While the Random Forest model performed comparably in terms of overall accuracy, SVM demonstrated superior consistency in macro-average and weighted-average scores, reflecting its robustness across all classes. Additionally, SVM's ability to handle high-dimensional data and produce a well-defined decision boundary makes it an ideal choice for this dataset.

4.1.2 Strengths of the Selected Model

The strengths of the SVM model are as follows:

- **Balanced Performance:** SVM achieved high precision, recall, and F1-scores for all life expectancy categories, ensuring fairness across classes.
- **Robustness:** The model is less prone to overfitting compared to Decision Trees and Bagging, making it reliable for both training and testing datasets.
- **Suitability for High-Dimensional Data:** SVM excels in scenarios where the dataset contains numerous features, as it effectively maximizes the margin between classes.
- **Flexibility:** The use of kernel functions allows SVM to model complex, non-linear relationships, making it adaptable to diverse datasets.

5 Conclusion

In this study, we investigated the prediction of life expectancy categories using various machine learning algorithms, including SVM, Decision Trees, Random Forests, and ensemble methods like Bagging and Boosting. The dataset underwent preprocessing steps such as feature scaling, binning the target variable, and one-hot encoding, ensuring readiness for model training. Hyperparameter tuning through Genetic Algorithms and Grid Search significantly enhanced model performance. Among the models, SVM and Random Forest achieved the highest accuracies of 92%, demonstrating their ability to generalize well across all life expectancy categories. Ensemble methods, while effective, showed slightly lower performance, with Boosting achieving an accuracy of 88%. Decision Trees, while interpretable, faced challenges with overfitting. SVM emerged as the best-performing model due to its robust handling of non-linear data and consistent results. This study demonstrates the potential of machine learning in predicting life expectancy and provides a foundation for future work, which could explore additional features or advanced techniques for improved predictions.