# Life Expectancy Prediction Using Decision Tree and Random Forest Algorithm

**Author: Adel.Ahmadi**

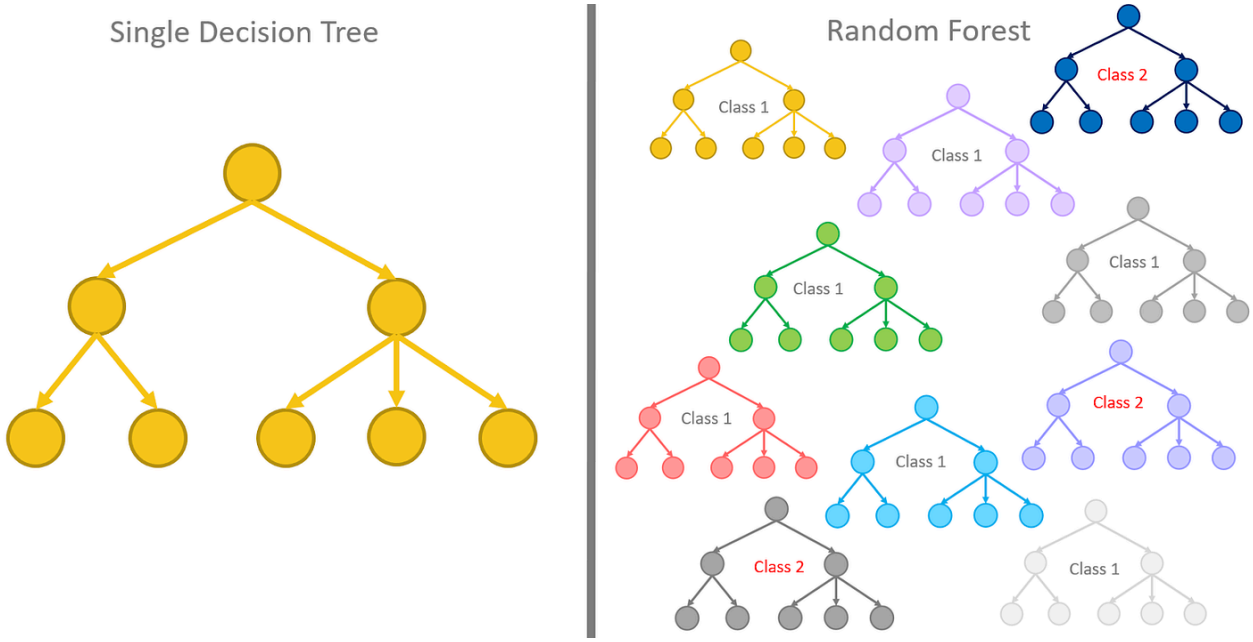
## 1. Decision Tree Algorithm Overview

Decision Tree (DT) is one of most used supervised learning algorithms for classification and regression tasks. We surely can say DT is one of most used algorithms in real world scenarios.

DT structures the dataset by splitting features into regions based on feature values. Each internal node in the tree represents a decision point and each leaf node represents a class label in classification. DTs are popular because of their ability to handle numerical and categorical values but sometimes using a single DT may cause overfitting the model so it is suggested to tune hyper parameters like maximum depth, minimum samples, etc…

## 2. Random Forest Algorithm Overview

Random Forest (RF) is an ensemble learning method built on a collection of decision trees. Unlike a single decision tree, a random forest aggregates the predictions from multiple trees, each of which is trained on a different random subset of the data (using bagging, bootstrap, etc..). This ensemble approach improves robustness and reduces overfitting. Additionally, Random Forests introduce feature randomness at each split, ensuring diverse trees in the forest and improving performance.




## 3. Decision Tree and Random Forest Hyper-parameters

In this section, we review DT and RF hyper-parameters that affect model accuracy, later on in the next chapters, we try to use some algorithms to find optimal hyper-parameters and optimize our model.

- Criterion

  This parameter specifies the metric used for measuring splitting quality. Some choices are gini (measuring split impurity) and entropy (measuring information gain).

- Splitter

  This parameter specifies how we want to split the node. There are some options including best (optimal split) and random split.

- Max Depth

  This parameter controls how specific a model can become. This parameter influences complexity and generalization.

- Min Samples Split

  This parameter determines required samples to split an internal node.

- Min Samples Leaf

  This parameter determines minimum samples needed to be at a leaf node.

## 4. Code Implementation

a. Data Loading and Preprocessing

First the dataset is loaded. The dataset is already cleaned so we just do some tiny data preprocessing like, categorizing target column and one-hot encoding “country” column.

b. Feature Scaling

We center and scale the data to have a mean of 0 and standard deviation of 1. This step causes the model to train faster and have better performance.

c. Model Definition

We define DT and RF models with some hyper-parameters to tune.

d. Hyper-parameter tuning

In this step, GridSearchCV is used to perform cross-validation across a range of defined hyper-parameter combinations.

e. Training best model

After tuning, the best hyper parameters are extracted based on cross-validation. Now we train our best model.

f. Model evaluation

We use classification_report to evaluate metrics like precision, recall, F1-score and etc… for each class of classification. We also plot the confusion matrix using the Seaborn and Matplotlib library.

## 5. Results

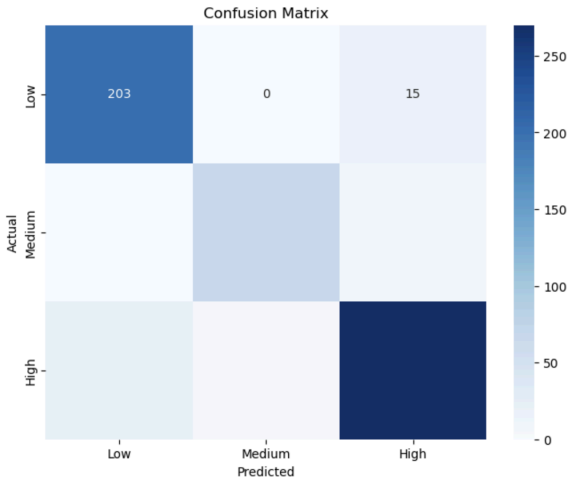
The Decision Tree (DT) and Random Forest (RF) classifiers were evaluated on their performance in predicting life expectancy categories: Low, Medium, and High. Performance metrics such as precision, recall, f1-score, and support were used to assess classification quality. Here are the detailed results:

### Decision Tree Results

The Decision Tree classifier achieved an overall accuracy of 92%, showing balanced precision and recall across all classes.

- High life expectancy class had a precision of 91%, recall of 93%, and an f1-score of 92%, indicating strong predictive power in this class.
- Low life expectancy class achieved a precision of 94%, recall of 88%, and an f1-score of 91%.
- Medium life expectancy class also showed balanced performance with a precision and recall of 92%, yielding an f1-score of 92%.

The macro average f1-score was 92%, reflecting that the model performed consistently well across all classes, while the weighted average accuracy was also 92%, suggesting the model was not biased toward any particular class.



## Random Forest Results

The Random Forest model also reached an overall accuracy of 92%, with the following breakdown:

- High class predictions yielded a precision of 93%, recall of 92%, and f1-score of 93%.
- Low class precision was high at 95%, but with slightly lower recall at 81%, resulting in an f1-score of 88%.
- Medium class achieved 90% precision, 94% recall, and an f1-score of 92%.

The macro average for Random Forest was 91%, while the weighted average was 92%. The difference in recall for the Low class suggests that the model could potentially benefit from further tuning or data balancing.