6/7/2024

Final Project Report

Supervised Learning

Group: Gold
Members:
Nidhi Trivedi
Tejaswi Neelapu
Tileshwar Narayan

Abstract

The analysis focuses on the analysis how economic conditions, measured by GDP, relate to the causes of child deaths under five years old in different countries. The dataset utilised for the study is “Causes of death in children under five, 2019” from “Our World in Data” [1] and GDP data(13). The collective data holds information about the count of deaths because of various diseases/causes in multiple countries and the GDP for every year starting from 1990 to 2019. Machine learning models like Random Forest, Bagging with Decision Tree, and Gradient Boosting classifiers are being utilised to find the important factors that affect the GDP classification. The Random Forest model has achieved the accuracy of 97.15% and identified congenital issues, cancers (neoplasms), and premature birth (preterm conditions) as key factors. The Bagging with Decision Tree model had an accuracy of 95.87%, focusing on cancers and premature birth. The Gradient Boosting model had an accuracy of 96.26%, highlighting cancers, premature birth, and respiratory issues as important factors suggesting that health issues affecting the lungs and breathing also play a role in a country's economic development. Also, Regression Models has been created to predict 2019 death rates. The Random Forest Regression model provided a R2 score of 94.32%, while the Decision Tree Regression model achieves 89.28%.The prediction are almost on the point with some deviation. The results show that countries with lower GDP have more children dying from certain diseases like birth defects, cancer, and complications from premature births. This means improving healthcare for these specific issues could help poorer countries become more economically developed. The importance of respiratory diseases also suggests that improving air quality and respiratory healthcare could be important for economic growth.

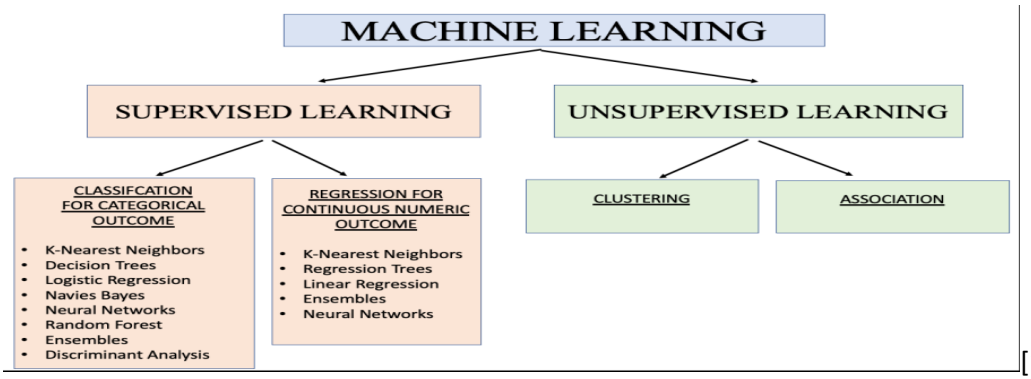
Introduction and Overview

Child deaths are still a major problem, especially in poorer countries. Understanding how a country's economy affects child deaths is important for making good health policies. This report looks at this connection using a dataset with GDP data and child death causes from 204 countries over 30 years (1990-2019). This analysis uses advanced machine learning models like Random Forest, Bagging with Decision Tree, and Gradient Boosting classifiers to find patterns and important health factors that influence GDP classification. The dataset provides an opportunity to analyse economic conditions that impacts child mortality and helps to identify which health factors are most closely associated with economic status. The dataset contains country names, country codes, years of observation, causes of death among children, and GDP figures. Supervised learning is particularly helpful in this analysis. It uses labelled data to train models, which allows to predict outcomes based on input features. The analysis involves several key steps, including data preparation, model fitting, and performance evaluation using various machine learning techniques. The models used for this study is Random Forest Classifier which is an ensemble learning method that combines multiple decision trees to improve accuracy and control overfitting, Bagging with decision tree classifier is an ensemble method that uses bootstrap aggregating for creating multiple subsets of the data and training a decision tree on each subset, gradient boosting classifier helps to build model sequentially by correcting errors made by the previous ones. The main goal of this project is to find out if countries with lower GDPs tend to have more children dying from certain causes compared to richer countries. These findings can help policymakers in identifying the key health factors and help them develop strategies to reduce child mortality rates and improve child health outcomes.

Theoretical Background:

Supervised learning uses algorithms that helps the model to learn the relationship between input and output. This process is referred as Training or Fitting. The input are the X variables which are referred as features and the output is the Y variable, which is referred as target, the data that contains both is known as labelled data. This learning uses a training set which helps model to produce the desired output. When data is fed into the model, it learns by changing its settings until it works well. This happens during a process called as cross-validation, where the model is being tested for its performance.

There are two types of supervised learning algorithms: Classification and Regression. In Classification the algorithm groups the data for predicting target variable, it is used when output variable is categorial. While regression is used to predict a real or continuous value.

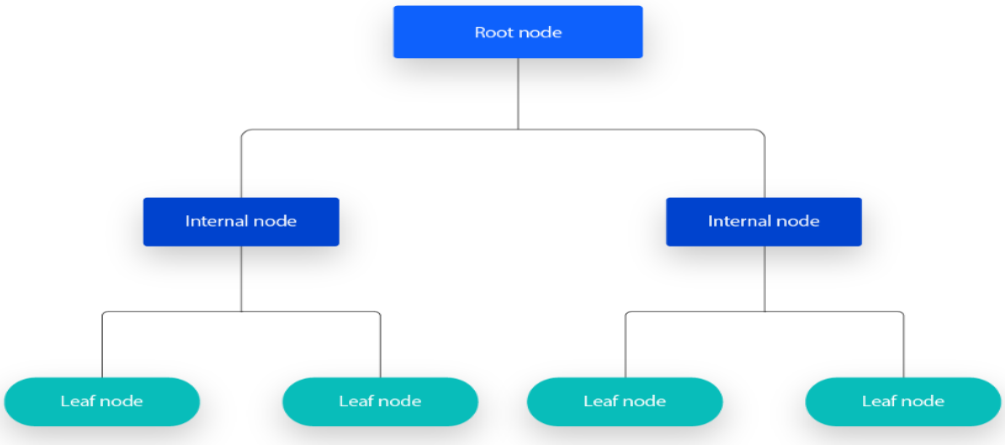


[1]

Methods used:

Decision Trees:

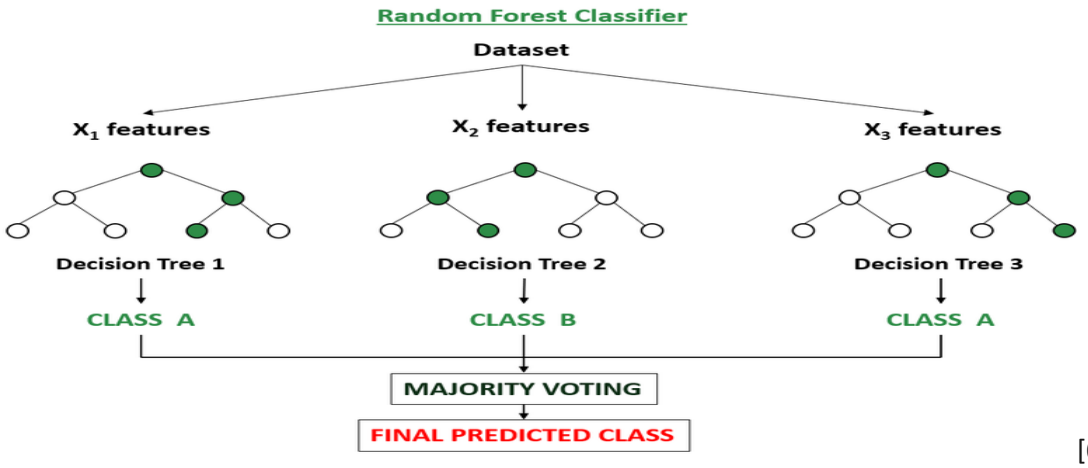
A decision tree is a hierarchical tree structure, which consists of a root node, branches, internal nodes, and leaf nodes. A decision tree starts with a root node, which does not have any incoming branches. The outgoing branches from the root node then feed into the internal nodes, also known as decision nodes. Based on the available features, both node types conduct evaluations to form homogenous subsets, which are denoted by leaf nodes, or terminal nodes. The leaf nodes represent all the possible outcomes within the dataset.



Decision Tree Classifiers, develop tree like models in which internal nodes are decisions from the features representing the remaining features to the leaf nodes which are the classes. For enhancement, it is necessary to set some parameters such as **max_depth, min_samples_split, min_samples_leaf and max_features** to avoid overfitting. Even its effectiveness in handling continuous variables and clear visualization, decision trees can overfit while they have high variance, which means small variations in the input data can result in the formation of different trees.

Random Forest Classifier:

Random forest classifier is collection of decision trees, where each tree is built using a randomly selected subset of the training data and a random subset of features at each node. In this method each tree is constructed using random subset of the dataset. This randomness introduces variability among individual trees, reducing the risk of overfitting and improving overall prediction performance.

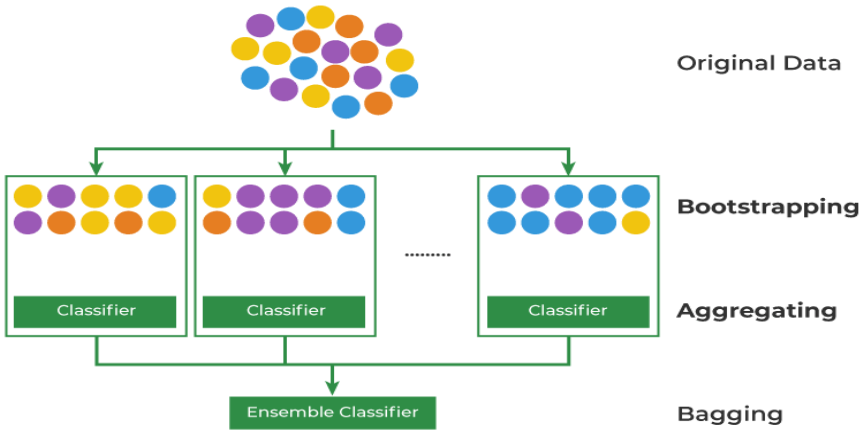


[6]

The hyperparameters that are especially important for the training of a random forest are, **n_estimators,** which represents the number of trees, **min_samples_split** which specifies the number of samples necessary to split a node and **min_samples_leaf** which is the number of samples required for a node. Hyperparameters allow for discovering the best model settings and the grid search aids by evaluating one using param_grid that defines which hyperparameters to improve and within what range and using cv for setting the number of folds in cross-validation and using scoring for indicating the measure of merit of each hyperparameter combination.

Bagging classifier:

Bagging is a type of ensemble learning method, also known as bootstrap aggregating is a technique that combines multiple models to improve performance. A Bagging Classifier is used, which employs a base decision tree classifier as its estimator. Here, multiple subsets of the training data through bootstrap sampling and then training a separate model on each subset. The final prediction is made by aggregating the prediction of all base model using majority voting.



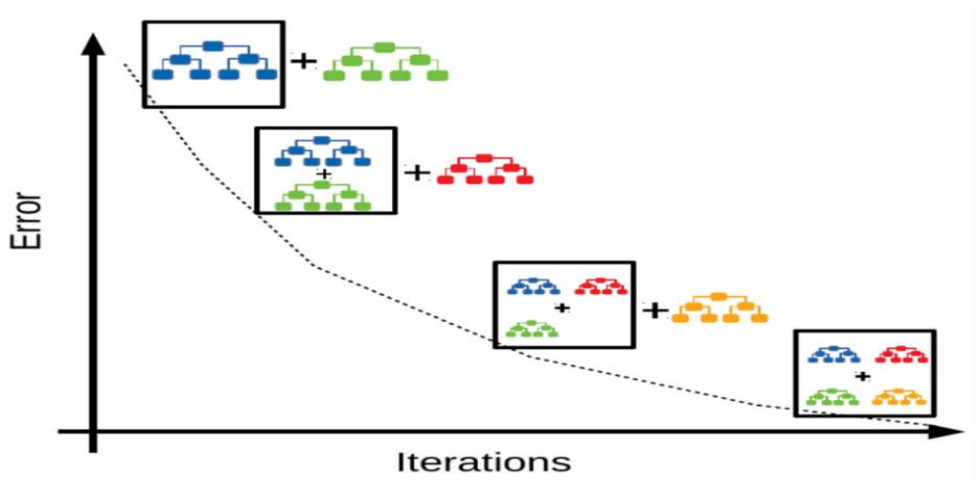
[7]

Bagging improves the accuracy and helps in reducing overfitting. The Bagging Classifier has several important parameters, including **n_estimators**, , **max_samples**, and **max_features**, indicating the number of features to consider when fitting each base estimator.

Gradient Boosting:

Gradient boosting is one of the methods of ensemble learning and applicable for classification and regression problems. It trains models sequentially with each subsequent model learning from the mistake of the previous, usually using a loss function such as mean squared error or cross entropy. Weak learners are summed up to a strong learner

through repeatedly calculating the gradient of loss function and learning the new models that aim to minimize the current gradient. The process is continued till either improvement declines to a factor that is negligible or the established condition is reached.

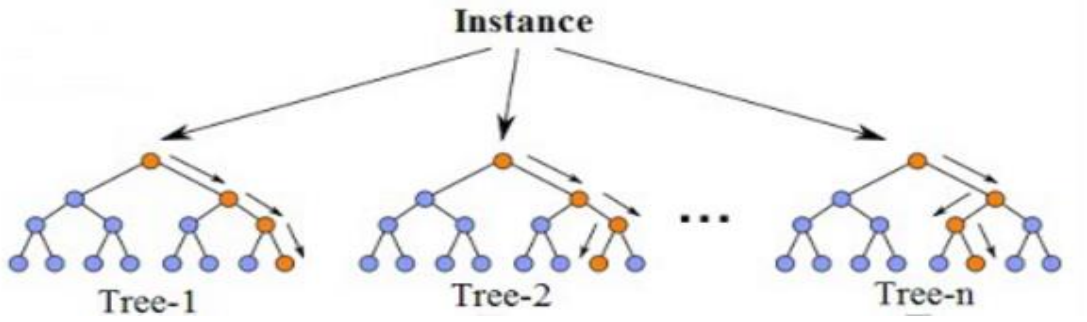


[9]

The Gradient Boosting Classifier's important parameters include n_estimators, which specifies the number of boosting stages (number of decision trees), and max_depth, determining the maximum depth of each individual tree in the ensemble. By adjusting these parameters, the Gradient Boosting Classifier can be fine-tuned to achieve better predictive performance.

Random Forest Regressor:

Random Forest Regression is another method of using decision tree but applying bootstrap sampling and aggregating the result of several decision trees to minimize the variance of an estimate. Different from that of a single decision tree which decides the final outputs, it compiles several decision trees. This is done by focusing on the rows of the dataset and the features such that sample datasets are created for individual trees known as bootstrapping.

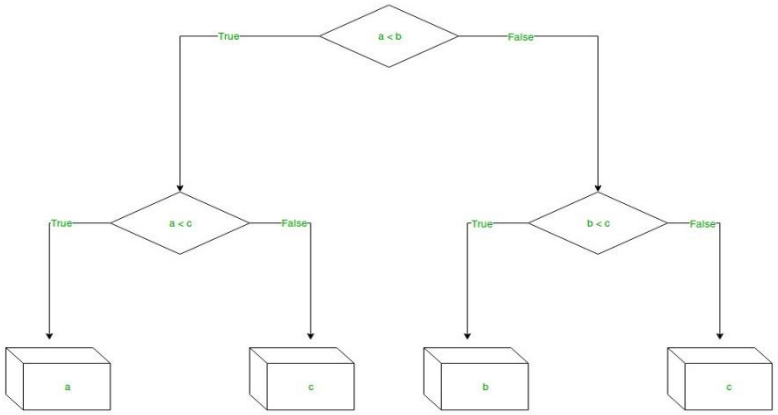


(11)

The Random Forest Regressor has some important hyperparameters such as n_estimators, which determines the number of trees in the forest, and min_samples_split, which is the number of samples required to split a node. Hyperparameters are evaluated in grid search to fine tune these settings as required.

Decision Tree Regressor:

Decision tree models approximate the data by growing the binary decision trees, and where each branch will end with a terminal node and is equal to the average of the actual observations. Like other forms of regression models, decision tree regression focuses on forecasting a single or a greater number of output values from feature inputs. Turn to the brief recap about the possible benefits, the drawbacks, and the contexts suitable for this type of model described in the previous section.



(12)

The decision tree regressor parameters consist of the following: max_depth, which limits how deep the tree can grow to prevent overfitting, min_samples_split, the minimum number of samples required to split a node, min_samples_leaf, the minimum number of samples required to be at a leaf node, max_features, which limits the number of features that are considered for splitting at each node. These parameters are used for controlling model complexity and performance.

Methodology:

The primary goal of the project is to understand the relation between economic conditions and the causes of death among children under 5 years old across various countries. By looking at both the GDP data and the child death data from many different countries, the project wants to find patterns that show if countries with low GDP are tend to have more children dying from certain deceases. The main purpose is to use these patterns to help governments and health organisations to make better plans and policies to improve children's health and save more young lives, especially in poorer countries.

Dataset Description

The dataset utilized for this analysis contains health information about how many children under the age of 5 died in each country every year, and what were the main causes of their deaths combined with the economic information about each country, like how much money the country makes every year. This is measured by Gross Domestic Product (GDP), which is the total value of all goods and services produced in a country in a year. The resultant dataset has information from 204 countries over a long period of 30 years, from 1990 to 2019 and contains columns such as name of the country, country code, year of observation, death cause amongst children, countries GDP.

Data Preparation

The dataset used for analysis combines health information on child mortality causes with economic indicators such as Gross Domestic Product (GDP) across 204 countries spanning 30 years from 1990 to 2019. Columns include country name, country code, observation year, child mortality causes, and GDP. Data preparation involves loading the CSV file into a dataframe and performing initial exploratory data analysis, including statistical summaries and checking for null values. Rows with missing 'code' values and those corresponding to the 'World' entity are removed to ensure data integrity. The 0 or say missing values in the GDP column are replaced with NaN and later filled with mean GDP values of respective countries. Descriptive statistics are computed to understand the dataset better. GDP values are then classified into quartiles, creating a new column indicating different GDP levels. To simplify coding, a dictionary is created to map original column names to shorter versions. This prepared dataset will help us understand the relationship between economic conditions and child mortality causes across diverse nations.

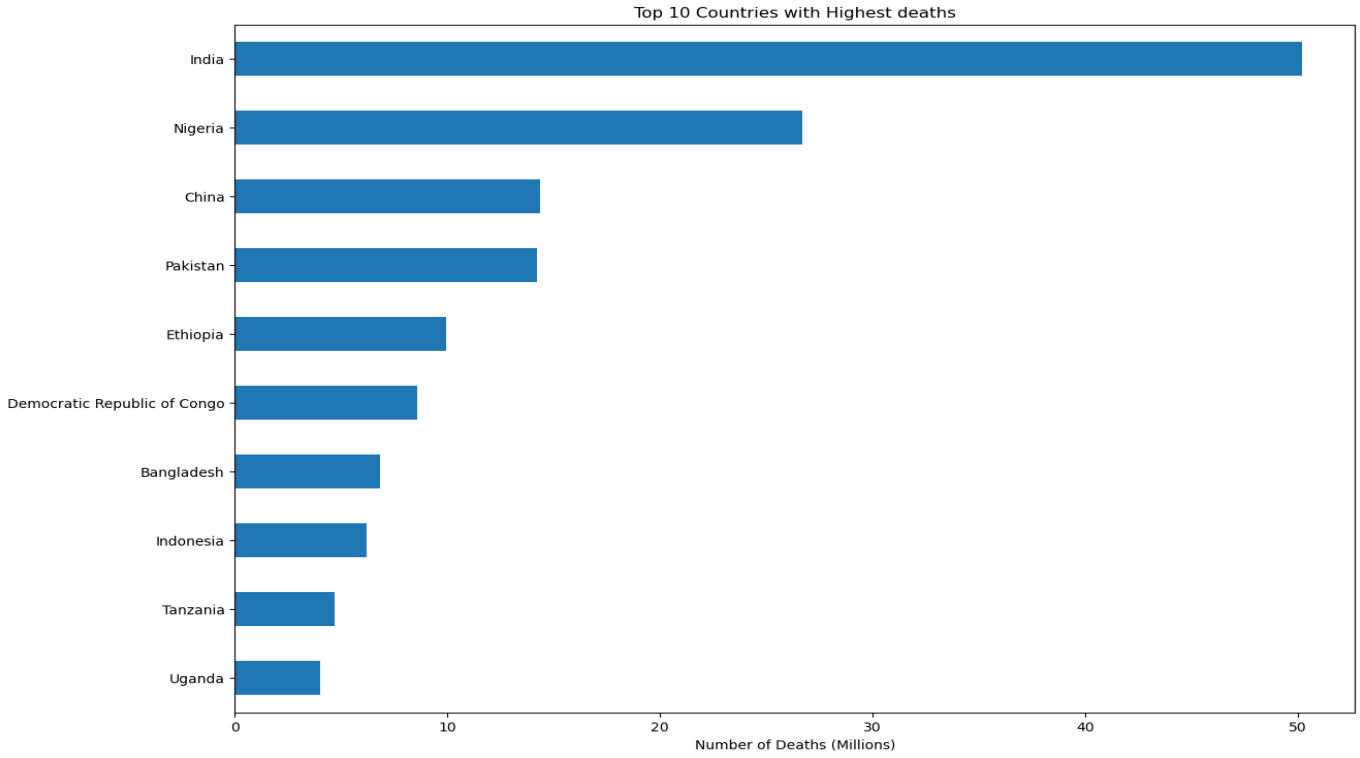
Model Fitting

Classification Models: In the project, a classification task is performed using three distinct models: Random Forest Classifier, Decision Tree Classifier with Bagging, and Gradient Boosting Classifier. Initially, the dataset is split into training and testing subsets in a ratio of 70:30, 70 as training and 30 as testing. GridSearchCV is then used to optimize parameters for each model. For the Random Forest Classifier, the parameter n_estimators (number of decision trees) are tuned. In the case of the Decision Tree Classifier with Bagging, parameters tuned include n_estimators (number of estimators) and max_features (maximum features for splitting). Lastly, for the Gradient Boosting Classifier, the parameters n_estimators (number of estimators) and max_depth (maximum depth of each tree) is tuned. After parameter tuning, each model is fitted to the training data and used to predict the GDP classes of the test data. The accuracy of predictions is assessed using metrics such as accuracy score, confusion matrix, and classification report. Feature importance analysis is also conducted to identify attributes with significant contributions to GDP class prediction. Visualizations of the top features help in understanding their importance in the classification process.

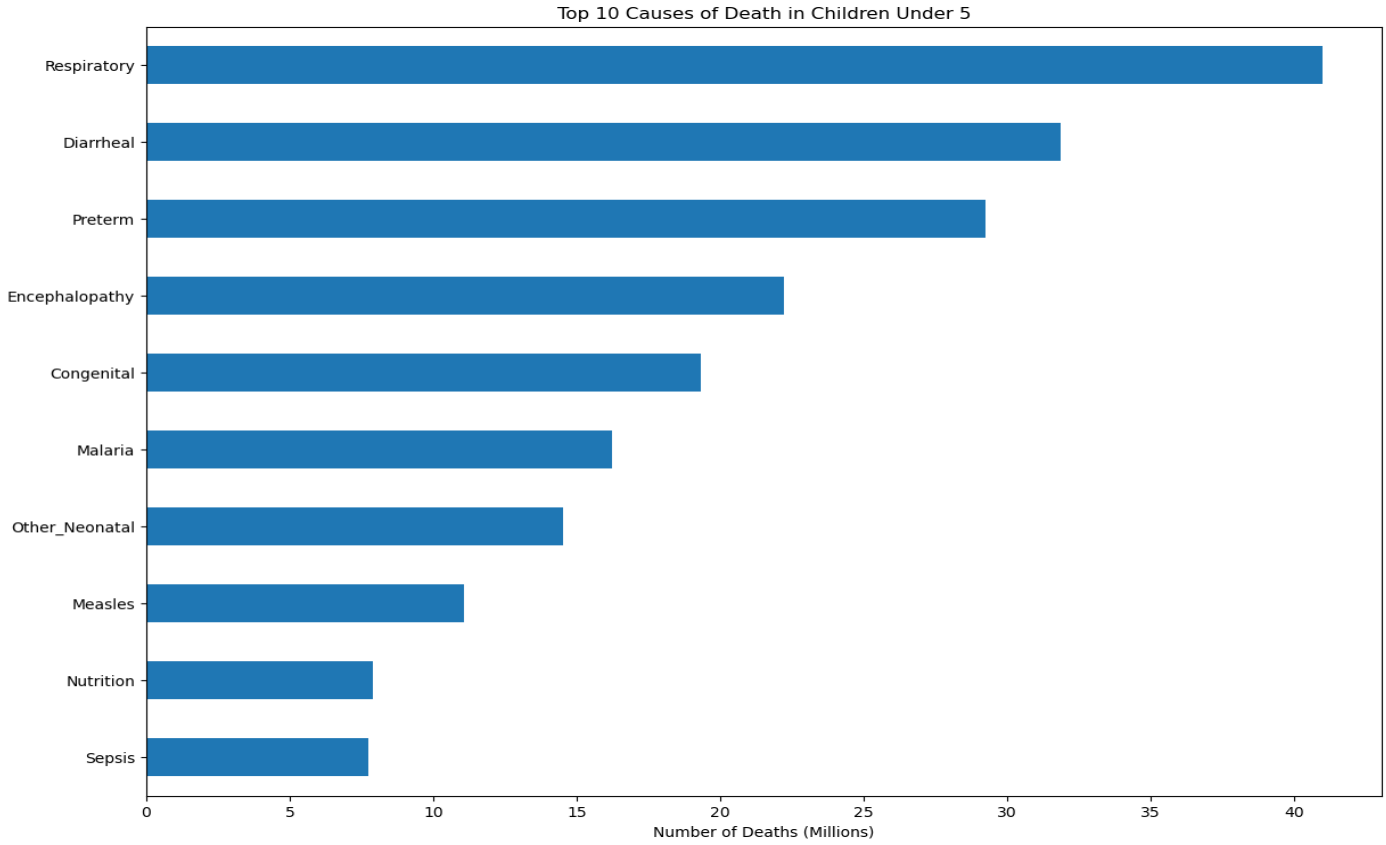
Regression Models: In the regression analysis, two models are evaluated: Random Forest Regressor and Decision Tree Regressor with Bagging. Initially, the dataset is divided into training and testing sets, and then scaled. For the Random Forest Regressor, GridSearchCV is employed to tune parameters, focusing on n_estimators (number of decision trees) and max_depth. The best parameters are determined, and the model is trained and evaluated on the test set using metrics like Mean Squared Error (MSE), Mean Absolute Error (MAE), and R-squared. Similarly, for the Decision Tree Regressor with Bagging, GridSearchCV is utilized to optimize parameters such as max_depth, min_samples_split, and min_samples_leaf. The best model is selected, trained, and evaluated. Results indicate the performance of each model, providing insights into their effectiveness in predicting GDP classes based on the given features.

Computational Results:

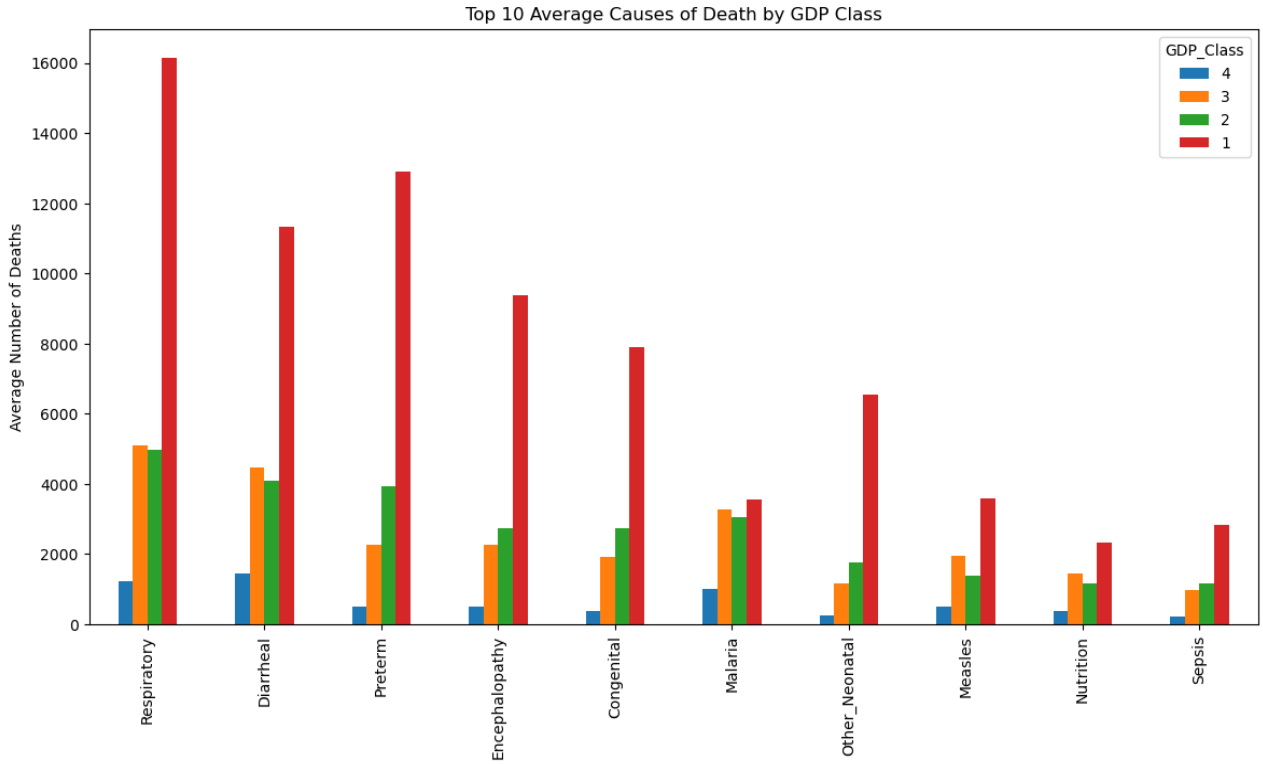
The below graph plots the top 10 Countries vs number of deaths



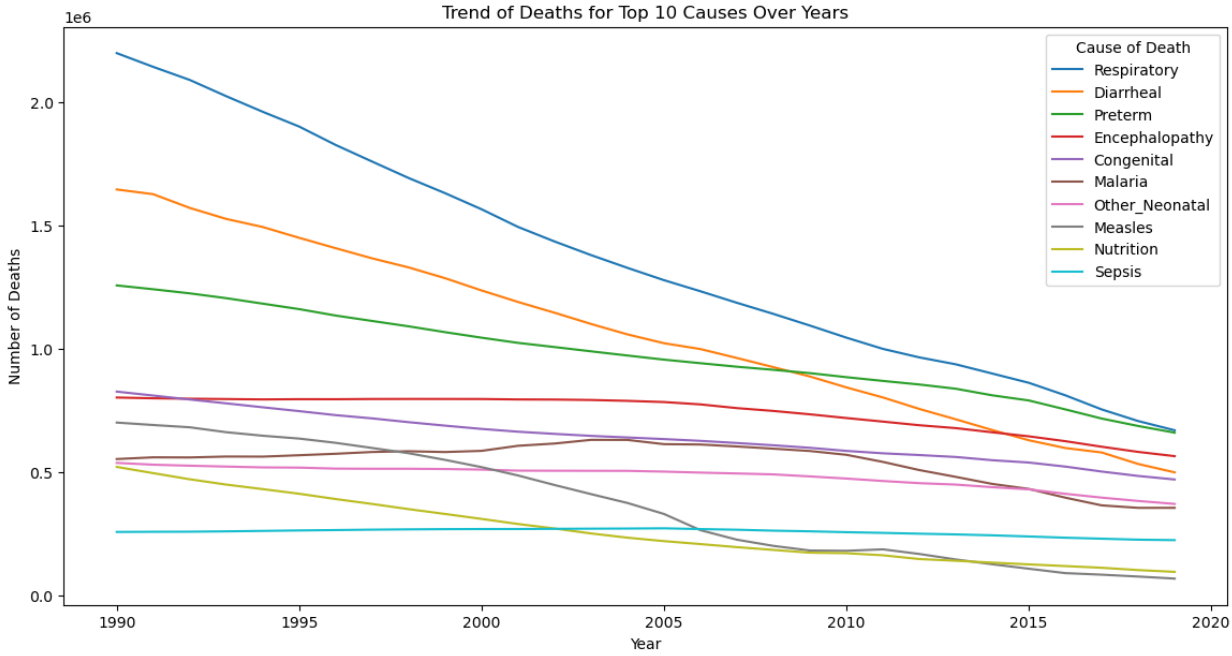
Top 10 Causes vs number of deaths



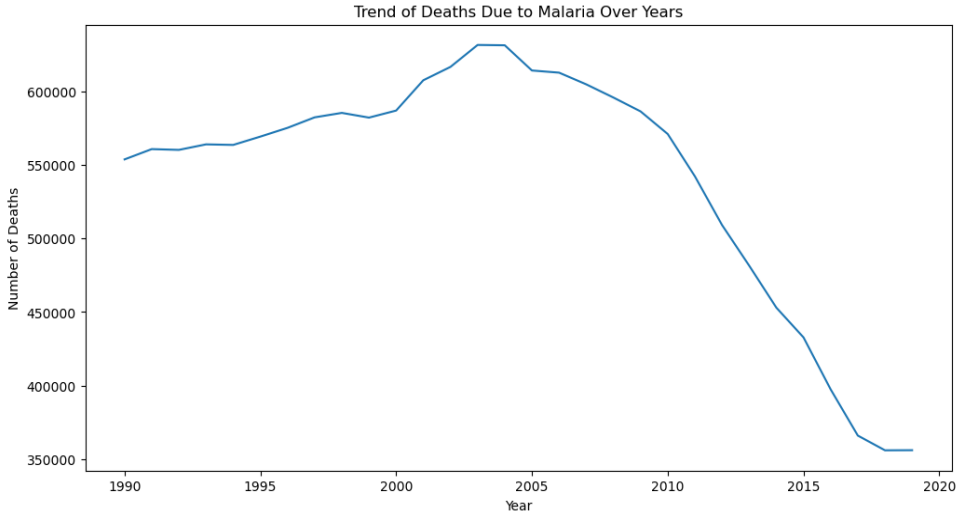
The below graph plots the average number of deaths per GDP class for top 10 deceases.



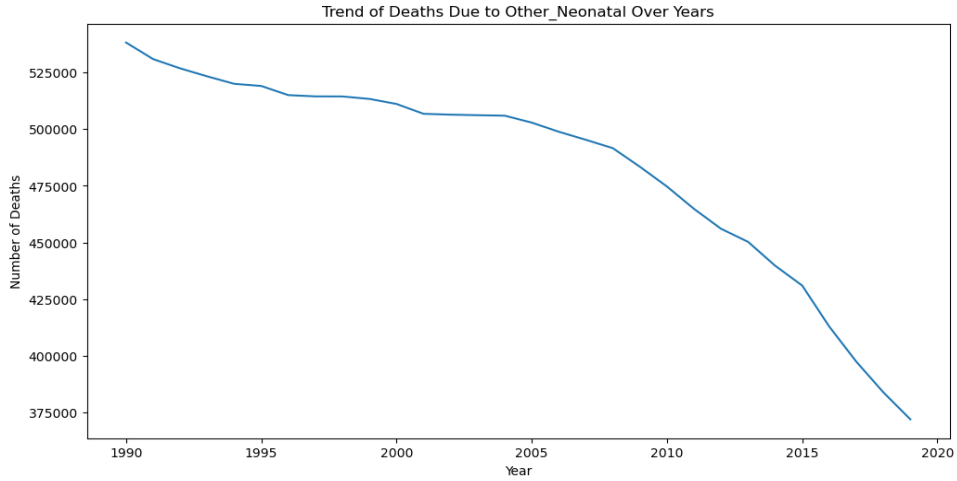
The below graph shows the trend of deaths for top 10 deceases over the years



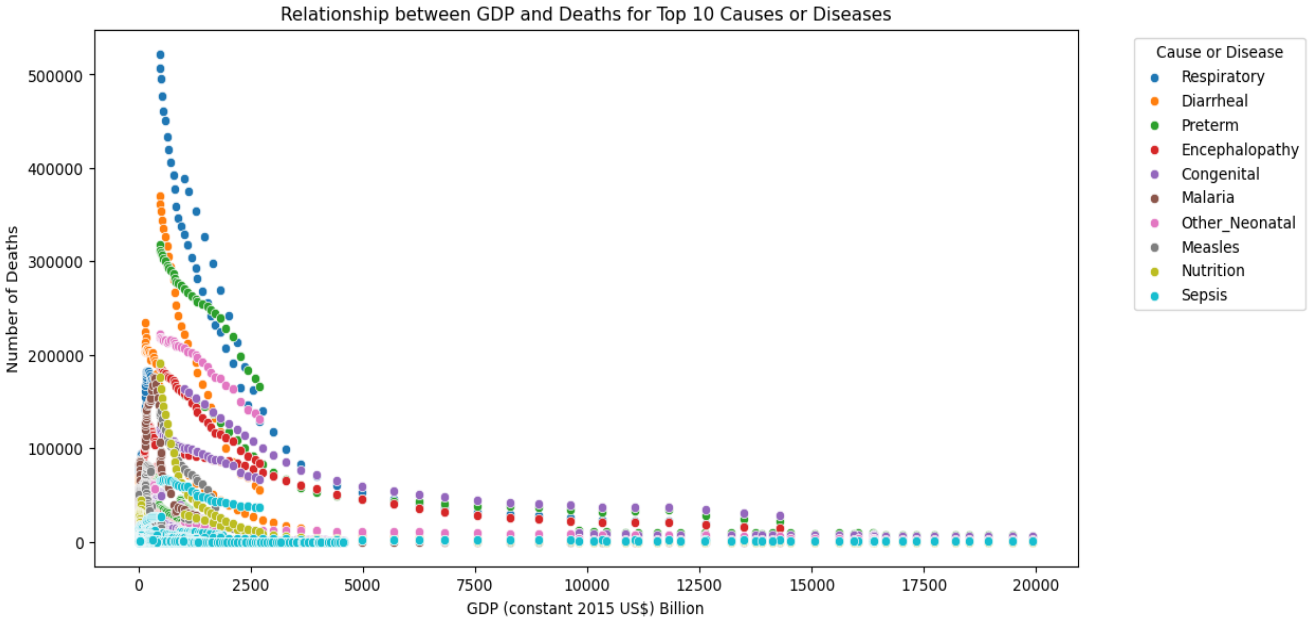
The below graph helps to visualize the deaths over the years due to Malaria.



The below graph helps to visualize the deaths over the years due to Other Neonatal disease.



The below graph helps to visualize the relationship between GDP and number of deaths.



Computational Results: Classification Models

Random Forest Classifier

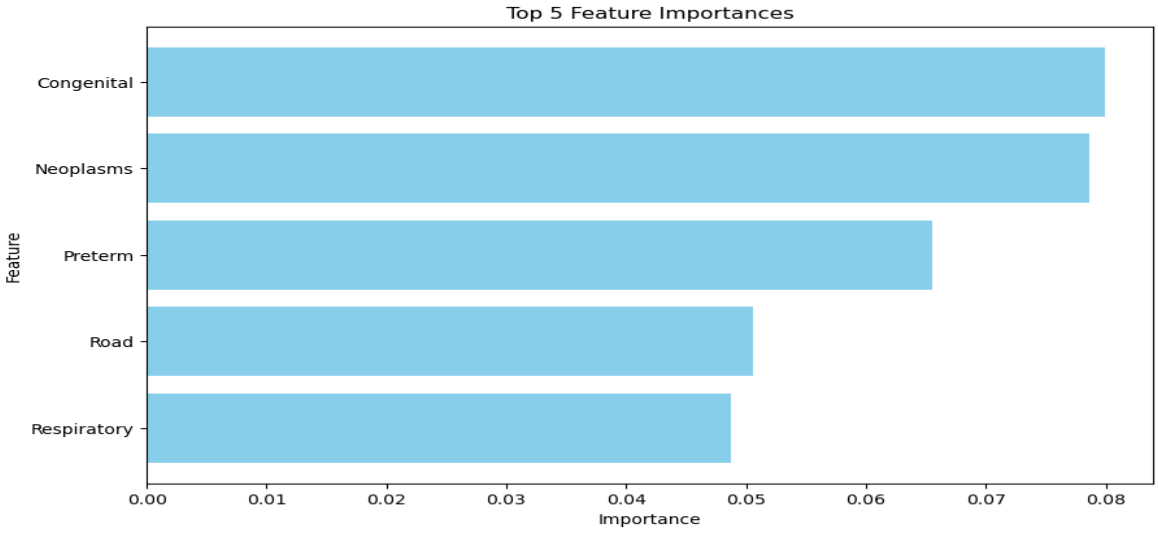
Best Parameters: 'n_estimators': 30

Metric	Value
Train Accuracy	0.9964106
Test Accuracy	0.9664991
Train Error	0.0035893
Test Error	0.03350083

Feature Importance:

Feature	Importance
Congenital	0.069775
Neoplasms	0.062078
Other_Neonatal	0.049724
Preterm	0.048874
Road	0.048786

Graphical Representation:



Bagging with Decision tree classifier

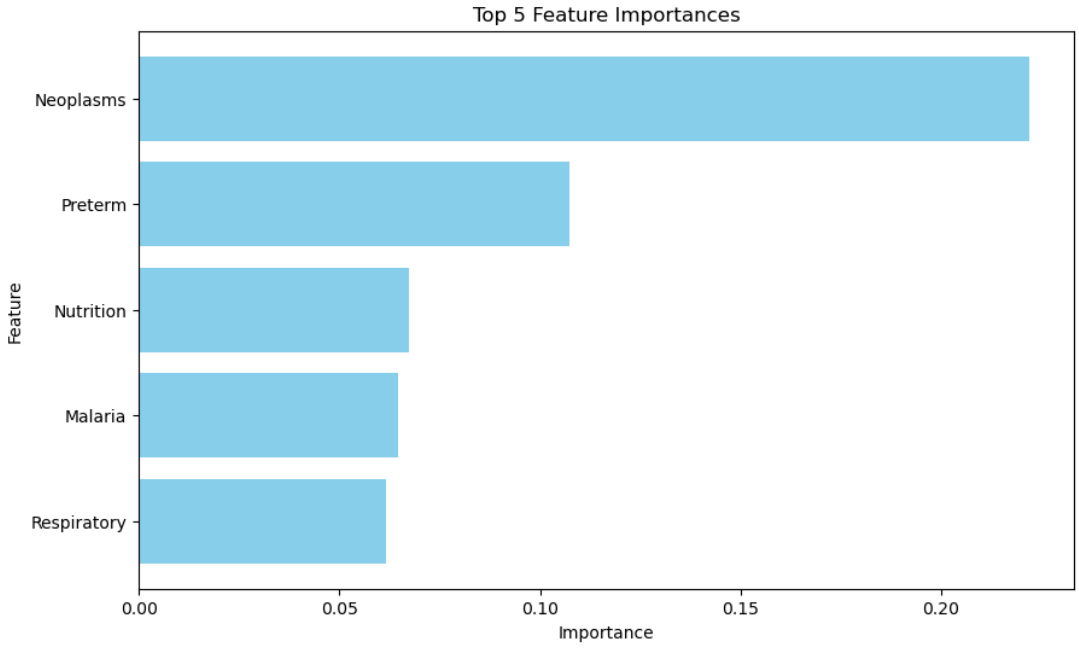
Best Parameters: 'max_features': 29 and 'n_estimators': 20

Metric	Value
Train Accuracy	0.996649916
Test Accuracy	0.958123953
Train Error	0.003350084
Test Error	0.041876047

Feature Importance:

Feature	Importance
Neoplasms	0.192679
Preterm	0.078977
Congenital	0.072614
Nutrition	0.066883
Respiratory	0.065775

Graphical Representation of top features



Boosting (Gradient Boosting Classifier)

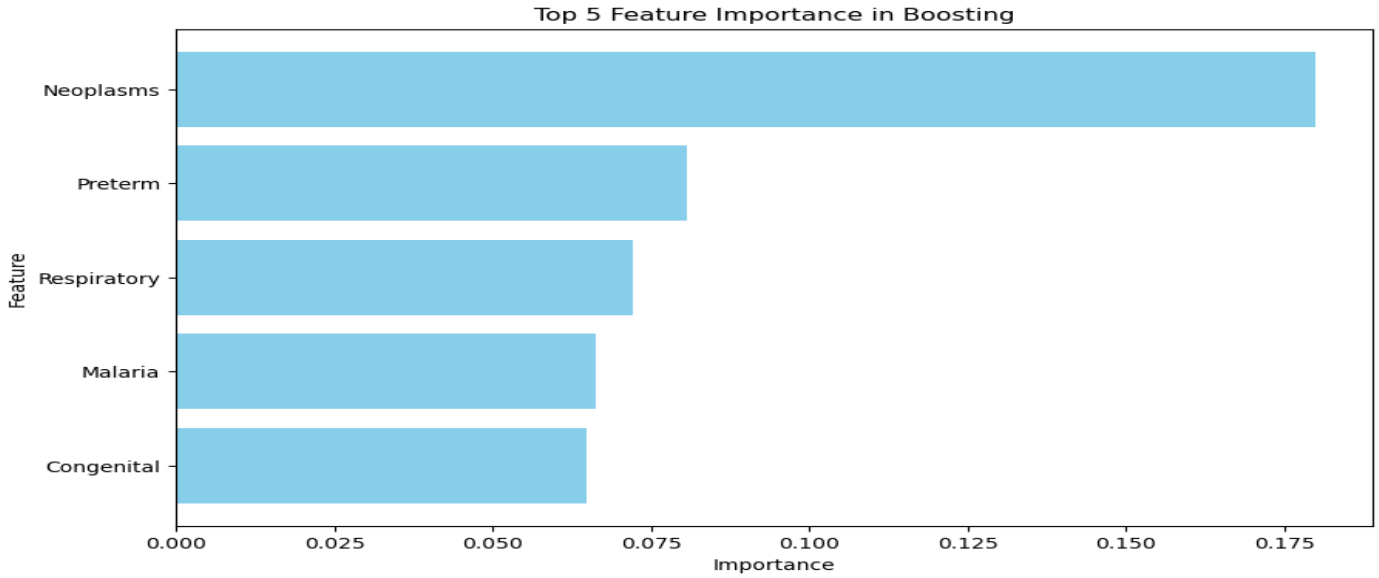
Best Parameters: 'max_depth': 10 and ‘n_estimators': 30

Metric	Value
Train Accuracy	0.996889208
Test Accuracy	0.964265773
Train Error	0.003110792
Test Error	0.035734227

Feature Importance:

Feature	Importance
Neoplasms	0.179921992
Preterm	0.08075193
Respiratory	0.072067165
Malaria	0.066301499
Congenital	0.064718356

Graphical representation of top features:



Regression Models:

Random Forest Regressor:

Best hyperparameters: 'max_depth': 10, and 'n_estimators': 30

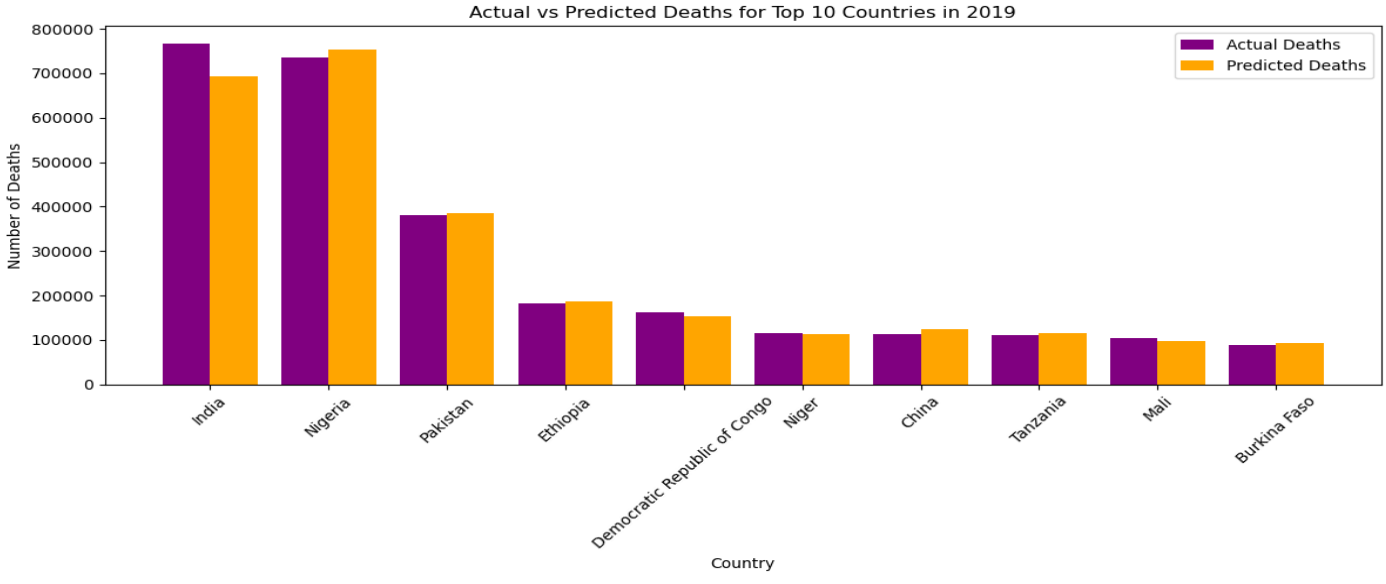
Results

Metric	Value
Mean Squared Error (MSE)	525904.2802
Mean Absolute Error (MAE)	138.2279608
R-squared	0.943234003

Predictions on 2019 Data:

Entity	Predicted_Deaths_2019	Actual_Deaths_2019	GDP_Class
Afghanistan	78705	68608	2
Albania	364	399	3
Algeria	16425	16401	1
American Samoa	33	7	4
Andorra	33	0	4

Visualization on Top 10 Countries By Death Rates:



Decision Tree Classifier:

Best parameters found: 'max_depth': 10, 'min_samples_leaf': 1, and 'min_samples_split': 2

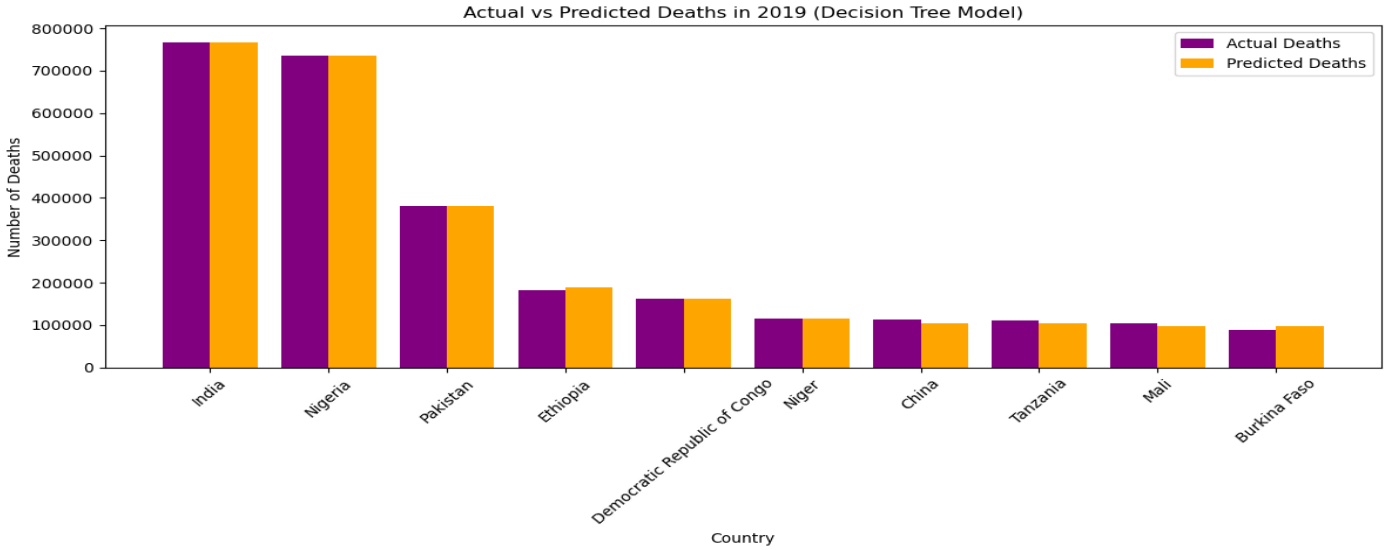
Results:

Metric	Value
Mean Squared Error (MSE) for Decision Tree	1668201.533
Mean Absolute Error (MAE) for Decision Tree	191.1044166
R-squared for Decision Tree	0.892836028

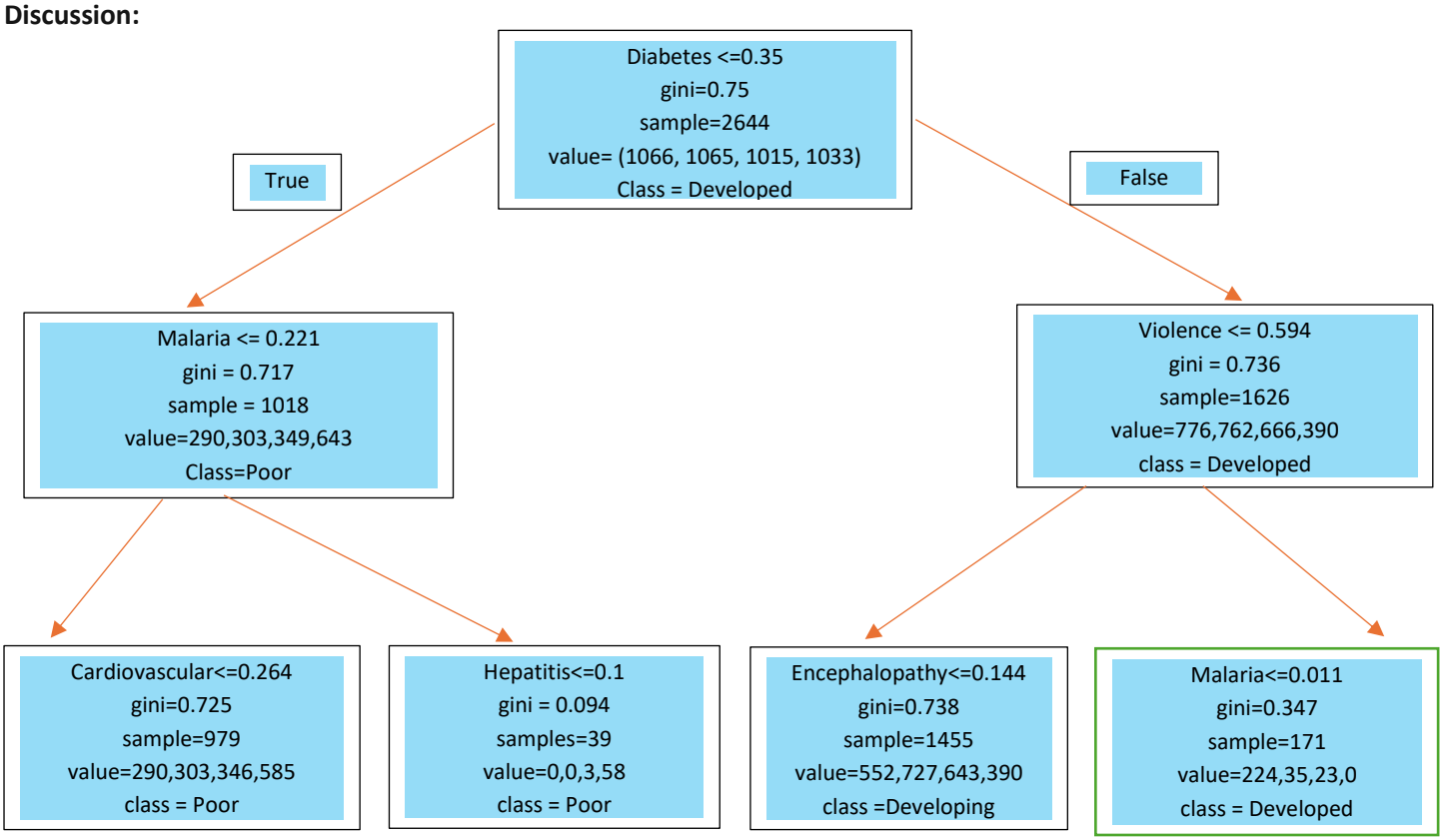
Prediction for 2019 (Head=5):

Entity	Predicted_Deaths_2019	Actual_Deaths_2019	GDP_Class
Afghanistan	79187	68608	2
Albania	369	399	3
Algeria	17604	16401	1
American Samoa	26	7	4
Andorra	26	0	4

Graphical Representation for Top 10 Countries:



Discussion:



The analysis has been done under classification with 3 different models namely, Random Forest Classifier, Boosting (Gradient Boosting) and Decision Tree with Bagging. The above tree is from our best model our of three models i.e., Random Forest Classifier with an accuracy of 96.64% along with Bagging with an accuracy of 95.81% and Boosting (Gradient Boosting) with an accuracy of 96.42%. Let's check our tree, we have the tree unfolds from the root node, where data division occurs based on a feature and its threshold. For example, the initial split starts on "Diabetes" with a boundary of 0.35, dividing data points either to the True or False side. Sample size denotes the number of data points reaching each node, while value counts explain the class distribution. Let's discussion one tree form Root to Node.

As we analyse the True side of the root node, the decision tree examines whether Diabetes falls below or equals 0.35. If this condition is true, the analysis proceeds to the next step. At this mid-point, the analysis explores whether there's an additional consideration, if the child has a low Malaria risk, provided by a value less than or equal to 0.221. This aspect allows it to search through into patterns among children with lower Diabetes levels, potentially coupled with reduced Malaria risk. After satisfying this condition, the analysis advances to the next level i.e., Cardiovascular by a value less than or equal to 0.264. This iterative process continues until it arrive at the leaf nodes. Also, the decision tree carefully examines the number of data points available at each stage (1018, 979, 39) to measure the information available. Then, it assesses the variety of health conditions present in the dataset (290, 303, 349, 643; 290, 303, 346, 585; 0, 0, 3, 58) to understand the dataset's composition. Finally, it predicts the most likely category for each group (Poor, Poor, Poor) to assist in decision-making. Through this process, the decision tree divides the data and identifies the prevalent health conditions at each step, offering valuable insights for healthcare and other relevant domains.

Here is the Overall Performance of all the Three Model.

Model	Train Accuracy	Test Accuracy	Train Error	Test Error
Random Forest	0.9964	0.9665	0.0036	0.0335
Decision Tree Bagging	0.9966	0.9581	0.0034	0.0419
Gradient Boosting	0.9969	0.9643	0.0031	0.0357

This comparative analysis of the three methods provides useful insights into how GDP and child mortality causes are related in different countries. Some of the Important features provide very similar kind of diseases. Neoplasms deaths, Preterm deaths and Congenital deaths are common diseases what analysis have observed through differnet models which are the common factor. The results show that countries with lower GDP tend to have more children dying from above 3 diseases. This means improving healthcare for these specific issues could help poorer countries become more economically developed. The importance of respiratory diseases in the Gradient Boosting model suggests that improving air quality and respiratory healthcare could also be important for economic development.

Now taking about the Regression Analysis, where the analysis is trying to predict the number of deaths based on the GDP per country for the year 2019. The analysis has been done on two models Random Forest Regressor and Decision Tree Regressor which has provided some important details which are as:

Model	R2 Score	Mean Squared Error	Mean Absolute Error
Random Forest	0.9432	525904.28	138.23
Decision Tree	0.8928	1668201.53	191.10

In the regression analysis, the Random Forest model provided better performance over the Decision Tree model based on different metrics we used in the analysis. The Random Forest model achieved a higher R2 score (0.9432) or say 94.32% compared to the Decision Tree (0.8928) or 89.28%, which can be interpreted as the Random Forest explains a larger proportion of the variance in the data. The Random Forest model provided higher predictive accuracy, reflected in its lower Mean Squared Error (MSE) of 525904.28 compared to the Decision Tree's MSE of 1668201.53. Along in line the model has provided that the Random Forest model yields a lower Mean Absolute Error (MAE) of 138.23, in comparation to the Decision Tree's MAE of 191.10. The following results that the Random Forest model provides more

accurate predictions with less deviation from the actual values, providing a improved ability to capture the underlying relationships within the dataset.

Conclusion

The study conducts a comprehensive analysis to understand the relationship between economic conditions, represented by GDP, and the causes of child mortality under the age of five across various countries. The study utilises the data of 30 years covering 204 countries. Several machine learning models were employed during this analysis including Random Forest, Bagging with Decision Tree, and Gradient Boosting. The Random Forest Classifier did well and achieved 97.15%. It is observed that congenital issues, cancer, and premature births are important for predicting how rich or poor a country is. The Bagging with Decision Tree Classifier has achieved 95.87% accuracy, identifying cancer and premature births are important too. The Gradient Boosting Classifier got 96.26% accuracy, also pointed cancer, premature births, and breathing problems as key factors. This linking socioeconomic factors with health outcomes, validate the vital role of early-life interventions in fostering long-term economic prosperity. Through regression the model has provided an accuracy for Random Forest model in predicting GDP is 94.32% while the decision tree achieved 89.28% accuracy.

These findings play an important role for policymakers and health groups. With this knowledge, policymakers can make better plans to save more children's lives, especially in poorer areas. Using advanced computer programs to analyse data gives us solid information to make smart decisions. This is a big step in solving the problem of children dying too young, especially in places where money is scarce.

References:

1. Diarrheal Diseases by Saloni Datana, Fiona Spooner, Hannah Ritchie and Max Rose
   https://ourworldindata.org/diarrheal-diseases#article-citation

2. Difference Between Supervised and Unsupervised Machine Learning Models and Their Use Cases by TrainDataHub
   https://medium.com/@ooemma83/how-to-identify-supervised-and-unsupervised-machine-learning-models-7707973096f7

3. What's the Difference Between Supervised and Unsupervised Learning? By AWS
   https://aws.amazon.com/compare/the-difference-between-machine-learning-supervised-and-unsupervised/

4. What's the Difference Between Supervised and Unsupervised Learning? By google cloud
   https://cloud.google.com/discover/what-is-supervised-learning

5. What is supervised learning? By IBM
   https://www.ibm.com/topics/supervised-learning

6. Supervised Machine Learning by Moez Ali
   https://www.datacamp.com/blog/supervised-machine-learning

7. Bagging and Random Forest for Imbalanced Classification by geeksforgeeks
   https://www.geeksforgeeks.org/bagging-and-random-forest-for-imbalanced-classification/

8. ML | Bagging classifier
   https://www.geeksforgeeks.org/ml-bagging-classifier/

9. GradientBoostingClassifier by scikitlearn
   https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.GradientBoostingClassifier.html

10. Understanding Gradient Boosting by Hemashreekilari
https://medium.com/@hemashreekilari9/understanding-gradient-boosting-632939b98764

11. Decision Tree Regression
https://support.monolithai.com/support/solutions/articles/80001051798-decision-tree-regression

12. Decision Tree Regression Using Sklearn
https://www.geeksforgeeks.org/python-decision-tree-regression-using-sklearn/

13. Gross Domestic Product Data
https://ourworldindata.org/grapher/national-gdp-constant-usd-wb?tab=table

In [1]:
```
# Importing Libraries
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import plotly.express as px
from tabulate import tabulate
from sklearn.preprocessing import StandardScaler
from scipy.cluster.hierarchy import dendrogram, linkage, cut_tree
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report, confusion_matrix, accuracy_sc
from sklearn.tree import plot_tree
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import BaggingClassifier
from sklearn.model_selection import GridSearchCV
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score
from sklearn.tree import DecisionTreeRegressor
import warnings
warnings.simplefilter("ignore")
```

In [2]:
```
# Calling Dataset
file_path = r"C:\Users\tiles\Downloads\Final Data on Death of Childrens under ag
df = pd.read_csv(file_path)
df.head()
```

Out[2]:

	Entity	Code	Year	GDP (constant 2015 US$) Billion	Deaths - Invasive Non-typhoidal Salmonella (iNTS) - Sex: Both - Age: Under 5 (Number)	Deaths - Interpersonal violence - Sex: Both - Age: Under 5 (Number)	Deaths - Nutritional deficiencies - Sex: Both - Age: Under 5 (Number)	Deaths Acut hepatiti - Sex Both Age Under (Numbe
0	Afghanistan	AFG	1990	0.0	48	105	1779	71
1	Afghanistan	AFG	1991	0.0	55	130	1822	74
2	Afghanistan	AFG	1992	0.0	68	155	2069	83
3	Afghanistan	AFG	1993	0.0	78	178	2427	97
4	Afghanistan	AFG	1994	0.0	83	194	2649	106

5 rows × 33 columns

In [3]:
```
# Information of Dataset
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6840 entries, 0 to 6839
Data columns (total 33 columns):
 #   Column
Non-Null Count  Dtype
---  ------
--------------  -----
 0   Entity
6840 non-null   object
 1   Code
6150 non-null   object
 2   Year
6840 non-null   int64
 3   GDP (constant 2015 US$) Billion
6840 non-null   float64
 4   Deaths - Invasive Non-typhoidal Salmonella (iNTS) - Sex: Both - Age: Under 5
(Number)                  6840 non-null   int64
 5   Deaths - Interpersonal violence - Sex: Both - Age: Under 5 (Number)
6840 non-null   int64
 6   Deaths - Nutritional deficiencies - Sex: Both - Age: Under 5 (Number)
6840 non-null   int64
 7   Deaths - Acute hepatitis - Sex: Both - Age: Under 5 (Number)
6840 non-null   int64
 8   Deaths - Neoplasms - Sex: Both - Age: Under 5 (Number)
6840 non-null   int64
 9   Deaths - Measles - Sex: Both - Age: Under 5 (Number)
6840 non-null   int64
 10  Deaths - Digestive diseases - Sex: Both - Age: Under 5 (Number)
6840 non-null   int64
 11  Deaths - Cirrhosis and other chronic liver diseases - Sex: Both - Age: Under
5 (Number)                6840 non-null   int64
 12  Deaths - Chronic kidney disease - Sex: Both - Age: Under 5 (Number)
6840 non-null   int64
 13  Deaths - Cardiovascular diseases - Sex: Both - Age: Under 5 (Number)
6840 non-null   int64
 14  Deaths - Congenital birth defects - Sex: Both - Age: Under 5 (Number)
6840 non-null   int64
 15  Deaths - Lower respiratory infections - Sex: Both - Age: Under 5 (Number)
6840 non-null   int64
 16  Deaths - Neonatal preterm birth - Sex: Both - Age: Under 5 (Number)
6840 non-null   int64
 17  Deaths - Environmental heat and cold exposure - Sex: Both - Age: Under 5 (Nu
mber)                     6840 non-null   int64
 18  Deaths - Neonatal sepsis and other neonatal infections - Sex: Both - Age: Un
der 5 (Number)            6840 non-null   int64
 19  Deaths - Exposure to forces of nature - Sex: Both - Age: Under 5 (Number)
6840 non-null   int64
 20  Deaths - Diabetes mellitus - Sex: Both - Age: Under 5 (Number)
6840 non-null   int64
 21  Deaths - Neonatal encephalopathy due to birth asphyxia and trauma - Sex: Bot
h - Age: Under 5 (Number)  6840 non-null   int64
 22  Deaths - Meningitis - Sex: Both - Age: Under 5 (Number)
6840 non-null   int64
 23  Deaths - Other neonatal disorders - Sex: Both - Age: Under 5 (Number)
6840 non-null   int64
 24  Deaths - Whooping cough - Sex: Both - Age: Under 5 (Number)
6840 non-null   int64
 25  Deaths - Diarrheal diseases - Sex: Both - Age: Under 5 (Number)
6840 non-null   int64
 26  Deaths - Fire, heat, and hot substances - Sex: Both - Age: Under 5 (Number)
```

```
6840 non-null   int64
 27  Deaths - Road injuries - Sex: Both - Age: Under 5 (Number)
6840 non-null   int64
 28  Deaths - Tuberculosis - Sex: Both - Age: Under 5 (Number)
6840 non-null   int64
 29  Deaths - HIV/AIDS - Sex: Both - Age: Under 5 (Number)
6840 non-null   int64
 30  Deaths - Drowning - Sex: Both - Age: Under 5 (Number)
6840 non-null   int64
 31  Deaths - Malaria - Sex: Both - Age: Under 5 (Number)
6840 non-null   int64
 32  Deaths - Syphilis - Sex: Both - Age: Under 5 (Number)
6840 non-null   int64
dtypes: float64(1), int64(30), object(2)
memory usage: 1.7+ MB
```

In [4]:
```
# Checking Null Values
df.isnull().sum()
```

Out[4]:
```
Entity
0
Code
690
Year
0
GDP (constant 2015 US$) Billion
0
Deaths - Invasive Non-typhoidal Salmonella (iNTS) - Sex: Both - Age: Under 5 (N
umber)                      0
Deaths - Interpersonal violence - Sex: Both - Age: Under 5 (Number)
0
Deaths - Nutritional deficiencies - Sex: Both - Age: Under 5 (Number)
0
Deaths - Acute hepatitis - Sex: Both - Age: Under 5 (Number)
0
Deaths - Neoplasms - Sex: Both - Age: Under 5 (Number)
0
Deaths - Measles - Sex: Both - Age: Under 5 (Number)
0
Deaths - Digestive diseases - Sex: Both - Age: Under 5 (Number)
0
Deaths - Cirrhosis and other chronic liver diseases - Sex: Both - Age: Under 5
(Number)                    0
Deaths - Chronic kidney disease - Sex: Both - Age: Under 5 (Number)
0
Deaths - Cardiovascular diseases - Sex: Both - Age: Under 5 (Number)
0
Deaths - Congenital birth defects - Sex: Both - Age: Under 5 (Number)
0
Deaths - Lower respiratory infections - Sex: Both - Age: Under 5 (Number)
0
Deaths - Neonatal preterm birth - Sex: Both - Age: Under 5 (Number)
0
Deaths - Environmental heat and cold exposure - Sex: Both - Age: Under 5 (Numbe
r)                          0
Deaths - Neonatal sepsis and other neonatal infections - Sex: Both - Age: Under
5 (Number)                  0
Deaths - Exposure to forces of nature - Sex: Both - Age: Under 5 (Number)
0
Deaths - Diabetes mellitus - Sex: Both - Age: Under 5 (Number)
0
Deaths - Neonatal encephalopathy due to birth asphyxia and trauma - Sex: Both -
Age: Under 5 (Number)      0
Deaths - Meningitis - Sex: Both - Age: Under 5 (Number)
0
Deaths - Other neonatal disorders - Sex: Both - Age: Under 5 (Number)
0
Deaths - Whooping cough - Sex: Both - Age: Under 5 (Number)
0
Deaths - Diarrheal diseases - Sex: Both - Age: Under 5 (Number)
0
Deaths - Fire, heat, and hot substances - Sex: Both - Age: Under 5 (Number)
0
Deaths - Road injuries - Sex: Both - Age: Under 5 (Number)
0
Deaths - Tuberculosis - Sex: Both - Age: Under 5 (Number)
0
Deaths - HIV/AIDS - Sex: Both - Age: Under 5 (Number)
0
```

```
Deaths - Drowning - Sex: Both - Age: Under 5 (Number)
0
Deaths - Malaria - Sex: Both - Age: Under 5 (Number)
0
Deaths - Syphilis - Sex: Both - Age: Under 5 (Number)
0
dtype: int64
```

In [5]:
```
# Checking Row names with Missing Values
missing_code_entities_unique = df.loc[df['Code'].isnull(), 'Entity'].unique()
missing_code_entities_unique
```

Out[5]:
```
array(['African Region (WHO)', 'East Asia & Pacific (WB)',
       'Eastern Mediterranean Region (WHO)', 'England',
       'Europe & Central Asia (WB)', 'European Region (WHO)', 'G20',
       'Latin America & Caribbean (WB)',
       'Middle East & North Africa (WB)', 'North America (WB)',
       'Northern Ireland', 'OECD Countries',
       'Region of the Americas (WHO)', 'Scotland', 'South Asia (WB)',
       'South-East Asia Region (WHO)', 'Sub-Saharan Africa (WB)', 'Wales',
       'Western Pacific Region (WHO)', 'World Bank High Income',
       'World Bank Low Income', 'World Bank Lower Middle Income',
       'World Bank Upper Middle Income'], dtype=object)
```

In [6]:
```
# Removing all the NA values
df = df.dropna()
```

In [7]:
```
# Dropping world data
indices_to_drop = df[df['Entity'] == 'World'].index
df = df.drop(indices_to_drop)
```

In [8]:
```
df
```

Out[8]:

	Entity	Code	Year	GDP (constant 2015 US$) Billion	Deaths - Invasive Non-typhoidal Salmonella (iNTS) - Sex: Both - Age: Under 5 (Number)	Deaths - Interpersonal violence - Sex: Both - Age: Under 5 (Number)	Deaths - Nutritional deficiencies - Sex: Both - Age: Under 5 (Number)	Dea hep - B Un (Nun
0	Afghanistan	AFG	1990	0.000000	48	105	1779	
1	Afghanistan	AFG	1991	0.000000	55	130	1822	
2	Afghanistan	AFG	1992	0.000000	68	155	2069	
3	Afghanistan	AFG	1993	0.000000	78	178	2427	
4	Afghanistan	AFG	1994	0.000000	83	194	2649	
...	...	...	...	...	...	...	...	
6835	Zimbabwe	ZWE	2015	19.963120	106	31	1733	
6836	Zimbabwe	ZWE	2016	20.142980	112	32	1771	
6837	Zimbabwe	ZWE	2017	20.964866	111	32	1714	
6838	Zimbabwe	ZWE	2018	22.015179	109	31	1639	
6839	Zimbabwe	ZWE	2019	20.621079	108	31	1598	

6120 rows × 33 columns

In [9]:
```
# Fiiling up missing valoues in the GDP column through mean of the same country
gdp_columns = ['GDP (constant 2015 US$) Billion']
df[gdp_columns] = df[gdp_columns].replace(0, np.nan)
country_means = df.groupby('Entity')[gdp_columns].transform('mean')
df[gdp_columns] = df[gdp_columns].fillna(country_means)
```

In [10]:
```
# Checking for Null Values
df.isnull().sum()
```

Out[10]:
```
Entity
0
Code
0
Year
0
GDP (constant 2015 US$) Billion
150
Deaths - Invasive Non-typhoidal Salmonella (iNTS) - Sex: Both - Age: Under 5 (N
umber)                      0
Deaths - Interpersonal violence - Sex: Both - Age: Under 5 (Number)
0
Deaths - Nutritional deficiencies - Sex: Both - Age: Under 5 (Number)
0
Deaths - Acute hepatitis - Sex: Both - Age: Under 5 (Number)
0
Deaths - Neoplasms - Sex: Both - Age: Under 5 (Number)
0
Deaths - Measles - Sex: Both - Age: Under 5 (Number)
0
Deaths - Digestive diseases - Sex: Both - Age: Under 5 (Number)
0
Deaths - Cirrhosis and other chronic liver diseases - Sex: Both - Age: Under 5
(Number)                    0
Deaths - Chronic kidney disease - Sex: Both - Age: Under 5 (Number)
0
Deaths - Cardiovascular diseases - Sex: Both - Age: Under 5 (Number)
0
Deaths - Congenital birth defects - Sex: Both - Age: Under 5 (Number)
0
Deaths - Lower respiratory infections - Sex: Both - Age: Under 5 (Number)
0
Deaths - Neonatal preterm birth - Sex: Both - Age: Under 5 (Number)
0
Deaths - Environmental heat and cold exposure - Sex: Both - Age: Under 5 (Numbe
r)                          0
Deaths - Neonatal sepsis and other neonatal infections - Sex: Both - Age: Under
5 (Number)                  0
Deaths - Exposure to forces of nature - Sex: Both - Age: Under 5 (Number)
0
Deaths - Diabetes mellitus - Sex: Both - Age: Under 5 (Number)
0
Deaths - Neonatal encephalopathy due to birth asphyxia and trauma - Sex: Both -
Age: Under 5 (Number)      0
Deaths - Meningitis - Sex: Both - Age: Under 5 (Number)
0
Deaths - Other neonatal disorders - Sex: Both - Age: Under 5 (Number)
0
Deaths - Whooping cough - Sex: Both - Age: Under 5 (Number)
0
Deaths - Diarrheal diseases - Sex: Both - Age: Under 5 (Number)
0
Deaths - Fire, heat, and hot substances - Sex: Both - Age: Under 5 (Number)
0
Deaths - Road injuries - Sex: Both - Age: Under 5 (Number)
0
Deaths - Tuberculosis - Sex: Both - Age: Under 5 (Number)
0
Deaths - HIV/AIDS - Sex: Both - Age: Under 5 (Number)
0
```

```
Deaths - Drowning - Sex: Both - Age: Under 5 (Number)
0
Deaths - Malaria - Sex: Both - Age: Under 5 (Number)
0
Deaths - Syphilis - Sex: Both - Age: Under 5 (Number)
0
dtype: int64
```

In [11]:
```
# Dropping
df = df.dropna()
```

In [12]:
```
# Checking the extent of the data and columns
df.describe()
```

Out[12]:

	Year	GDP (constant 2015 US$) Billion	Deaths - Invasive Non-typhoidal Salmonella (iNTS) - Sex: Both - Age: Under 5 (Number)	Deaths - Interpersonal violence - Sex: Both - Age: Under 5 (Number)	Deaths - Nutritional deficiencies - Sex: Both - Age: Under 5 (Number)	Deat A hepati Sex: Bc Age: Und (Num
count	5970.000000	5970.000000	5970.000000	5970.000000	5970.000000	5970.000
mean	2004.500000	283.746473	237.289950	80.962312	1324.555444	164.37(
std	8.656166	1259.963444	1642.343329	307.091735	6970.007725	1636.69
min	1990.000000	0.021562	0.000000	0.000000	0.000000	0.000
25%	1997.000000	4.373719	0.000000	1.000000	1.000000	0.000
50%	2004.500000	18.464653	1.000000	7.000000	10.000000	1.000
75%	2012.000000	115.846902	24.000000	39.750000	452.750000	18.000
max	2019.000000	19928.975000	29706.000000	6193.000000	190524.000000	28230.000

8 rows × 31 columns

In [13]:
```
# Dividing the GDP data into 4 class according to Quartile Method
df['GDP (constant 2015 US$) Billion'] = df['GDP (constant 2015 US$) Billion'].re
df['GDP_Class'] = pd.qcut(df['GDP (constant 2015 US$) Billion'], 4, labels=[4, 3
df
```

Out[13]:

	Entity	Code	Year	GDP (constant 2015 US$) Billion	Deaths - Invasive Non-typhoidal Salmonella (iNTS) - Sex: Both - Age: Under 5 (Number)	Deaths - Interpersonal violence - Sex: Both - Age: Under 5 (Number)	Deaths - Nutritional deficiencies - Sex: Both - Age: Under 5 (Number)	Dea A hep - B Un (Nun
0	Afghanistan	AFG	1990	14.557656	48	105	1779	
1	Afghanistan	AFG	1991	14.557656	55	130	1822	
2	Afghanistan	AFG	1992	14.557656	68	155	2069	
3	Afghanistan	AFG	1993	14.557656	78	178	2427	
4	Afghanistan	AFG	1994	14.557656	83	194	2649	
...	...	...	...	...	...	...	...	
6835	Zimbabwe	ZWE	2015	19.963120	106	31	1733	
6836	Zimbabwe	ZWE	2016	20.142980	112	32	1771	
6837	Zimbabwe	ZWE	2017	20.964866	111	32	1714	
6838	Zimbabwe	ZWE	2018	22.015179	109	31	1639	
6839	Zimbabwe	ZWE	2019	20.621079	108	31	1598	

5970 rows × 34 columns

In [14]:
```
# Renaming the Column Names
rename_dict = {
    'Deaths - Invasive Non-typhoidal Salmonella (iNTS) - Sex: Both - Age: Under
    'Deaths - Interpersonal violence - Sex: Both - Age: Under 5 (Number)': 'Viol
    'Deaths - Nutritional deficiencies - Sex: Both - Age: Under 5 (Number)': 'Nu
    'Deaths - Acute hepatitis - Sex: Both - Age: Under 5 (Number)': 'Hepatitis',
    'Deaths - Neoplasms - Sex: Both - Age: Under 5 (Number)': 'Neoplasms',
    'Deaths - Measles - Sex: Both - Age: Under 5 (Number)': 'Measles',
    'Deaths - Digestive diseases - Sex: Both - Age: Under 5 (Number)': 'Digestiv
    'Deaths - Cirrhosis and other chronic liver diseases - Sex: Both - Age: Unde
    'Deaths - Chronic kidney disease - Sex: Both - Age: Under 5 (Number)': 'Kidn
    'Deaths - Cardiovascular diseases - Sex: Both - Age: Under 5 (Number)': 'Car
    'Deaths - Congenital birth defects - Sex: Both - Age: Under 5 (Number)': 'Co
    'Deaths - Lower respiratory infections - Sex: Both - Age: Under 5 (Number)':
    'Deaths - Neonatal preterm birth - Sex: Both - Age: Under 5 (Number)': 'Pret
    'Deaths - Environmental heat and cold exposure - Sex: Both - Age: Under 5 (N
    'Deaths - Neonatal sepsis and other neonatal infections - Sex: Both - Age: U
    'Deaths - Exposure to forces of nature - Sex: Both - Age: Under 5 (Number)':
    'Deaths - Diabetes mellitus - Sex: Both - Age: Under 5 (Number)': 'Diabetes'
    'Deaths - Neonatal encephalopathy due to birth asphyxia and trauma - Sex: Bo
    'Deaths - Meningitis - Sex: Both - Age: Under 5 (Number)': 'Meningitis',
    'Deaths - Other neonatal disorders - Sex: Both - Age: Under 5 (Number)': 'Ot
    'Deaths - Whooping cough - Sex: Both - Age: Under 5 (Number)': 'Whooping_Cou
    'Deaths - Diarrheal diseases - Sex: Both - Age: Under 5 (Number)': 'Diarrhea
```

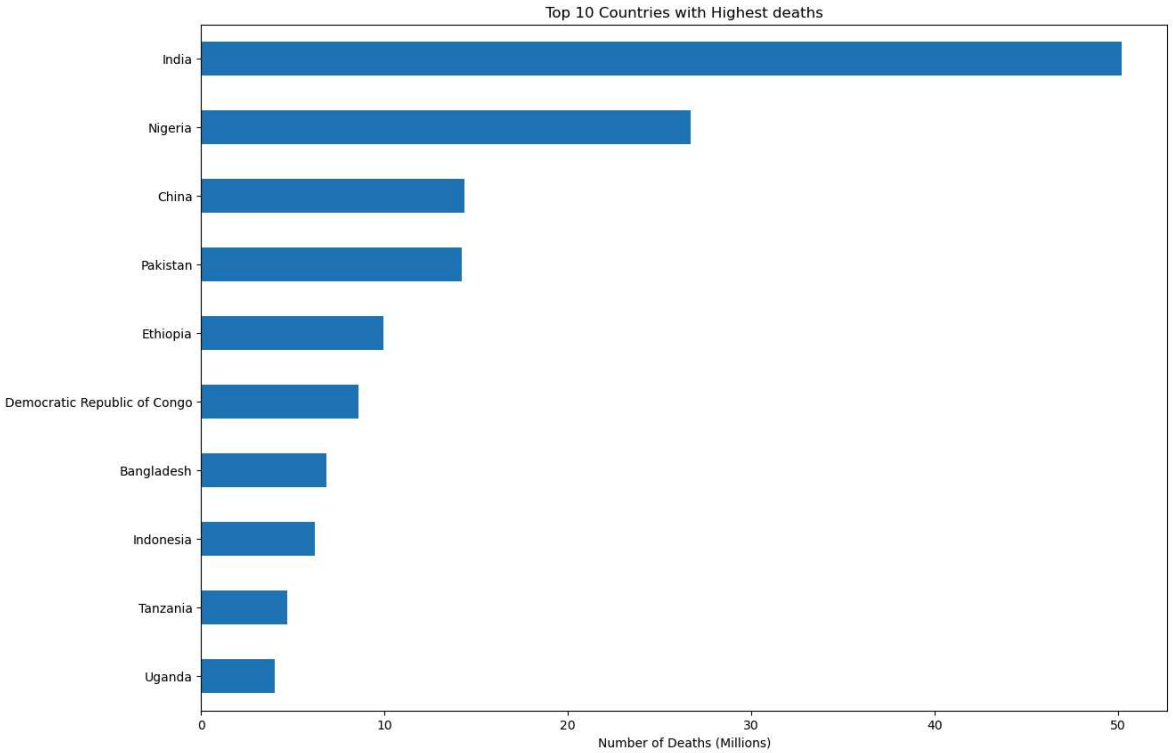
```
    'Deaths - Fire, heat, and hot substances - Sex: Both - Age: Under 5 (Number)
    'Deaths - Road injuries - Sex: Both - Age: Under 5 (Number)': 'Road',
    'Deaths - Tuberculosis - Sex: Both - Age: Under 5 (Number)': 'Tuberculosis',
    'Deaths - HIV/AIDS - Sex: Both - Age: Under 5 (Number)': 'HIV_AIDS',
    'Deaths - Drowning - Sex: Both - Age: Under 5 (Number)': 'Drowning',
    'Deaths - Malaria - Sex: Both - Age: Under 5 (Number)': 'Malaria',
    'Deaths - Syphilis - Sex: Both - Age: Under 5 (Number)': 'Syphilis'
}

df = df.rename(columns=rename_dict)
print(df.columns)
```

```
Index(['Entity', 'Code', 'Year', 'GDP (constant 2015 US$) Billion', 'INTS',
       'Violence', 'Nutrition', 'Hepatitis', 'Neoplasms', 'Measles',
       'Digestive', 'Cirrhosis', 'Kidney', 'Cardiovascular', 'Congenital',
       'Respiratory', 'Preterm', 'Heat_Cold', 'Sepsis', 'Nature', 'Diabetes',
       'Encephalopathy', 'Meningitis', 'Other_Neonatal', 'Whooping_Cough',
       'Diarrheal', 'Fire_Heat', 'Road', 'Tuberculosis', 'HIV_AIDS',
       'Drowning', 'Malaria', 'Syphilis', 'GDP_Class'],
      dtype='object')
```

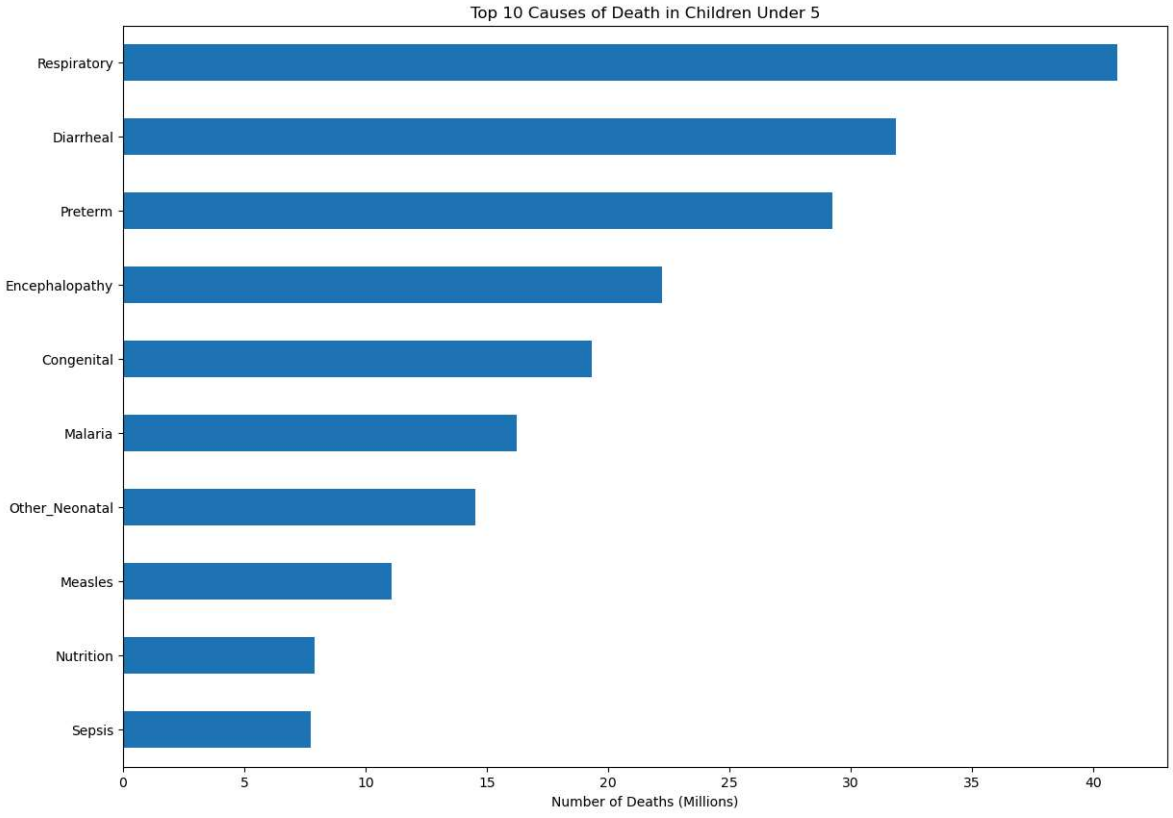
Graphs

In [15]:

```
#### Top 10 Countries with Highest Death
data = df[['Entity', 'INTS', 'Violence', 'Nutrition', 'Hepatitis', 'Neoplasms',
           'Measles', 'Digestive', 'Cirrhosis', 'Kidney', 'Cardiovascular',
           'Congenital', 'Respiratory', 'Preterm', 'Heat_Cold', 'Sepsis',
           'Nature', 'Diabetes', 'Encephalopathy', 'Meningitis',
           'Other_Neonatal', 'Whooping_Cough', 'Diarrheal', 'Fire_Heat',
           'Road', 'Tuberculosis', 'HIV_AIDS', 'Drowning', 'Malaria',
           'Syphilis']].groupby('Entity').sum().sum(axis=1) / 1e6

top_10_countries = data.nlargest(10)

top_10_countries = top_10_countries.iloc[::-1]

top_10_countries.plot(kind='barh', figsize=(14, 10), title='Top 10 Countries wit
plt.xlabel('Number of Deaths (Millions)')
plt.ylabel('')
plt.ticklabel_format(style='plain', axis='x')
plt.show()
```

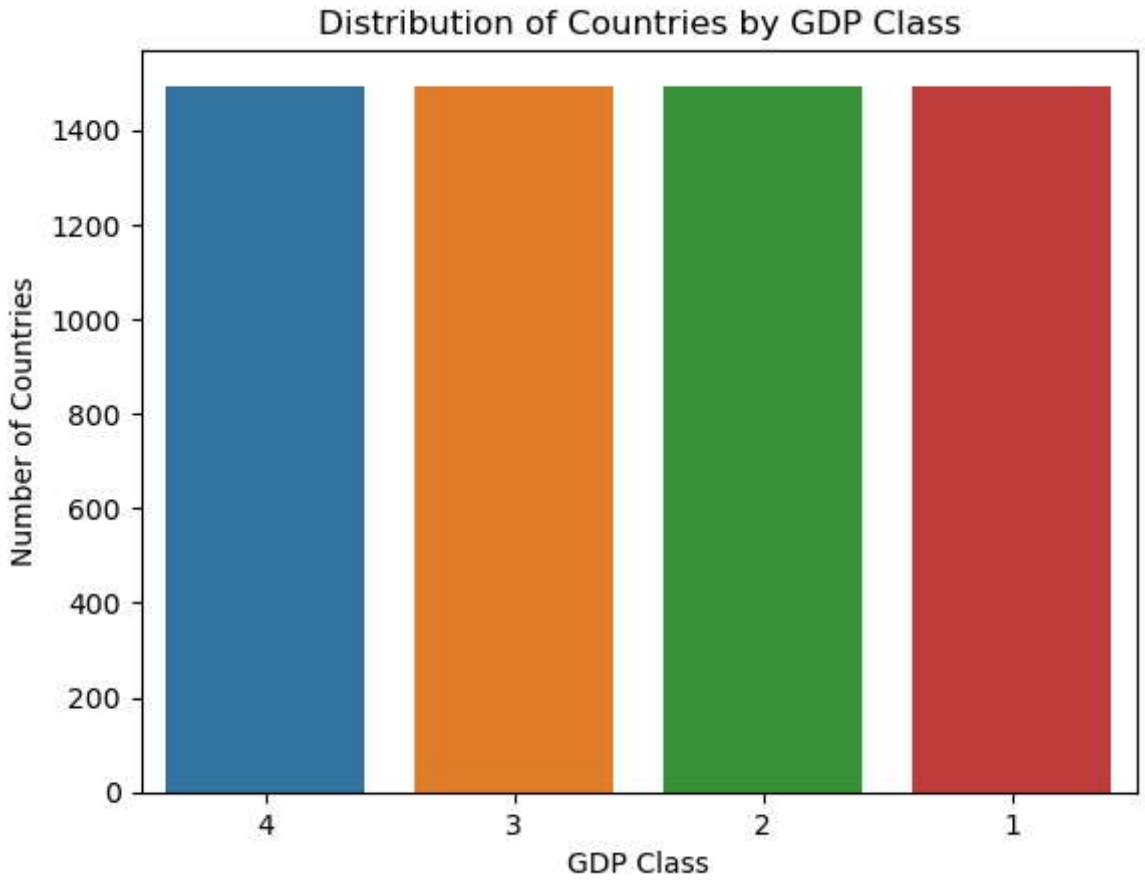


In [16]:
```
# Top 10 Causes Deaths
data = df.drop(columns=['Entity', 'Code', 'Year', 'GDP (constant 2015 US$) Billi
sorted_data = data.sort_values(ascending=True)
top_10_data = sorted_data.tail(10)
ax = top_10_data.plot(kind='barh', figsize=(14, 10), title='Top 10 Causes of Dea
plt.xlabel('Number of Deaths (Millions)')
plt.ticklabel_format(style='plain', axis='x')
ax.set_xticklabels(['{:.0f}'.format(x) for x in ax.get_xticks()])
plt.show()
```

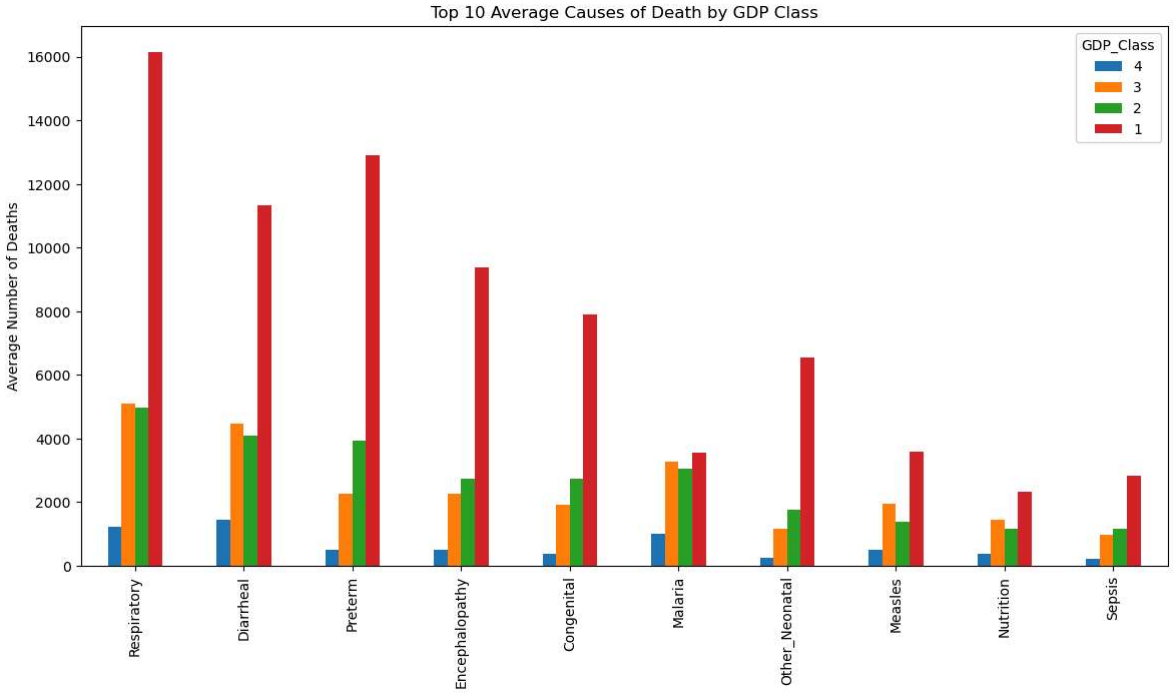


In [17]:
```
# Distribution after Classes dfeined after using
sns.countplot(x='GDP_Class', data=df)
```

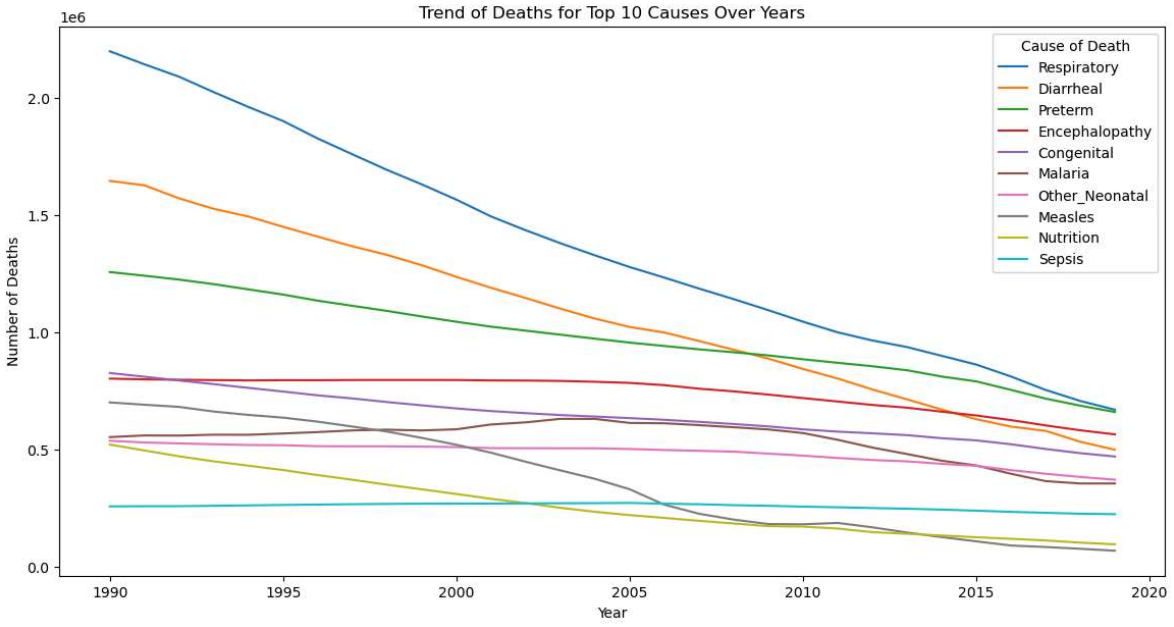
```
plt.title('Distribution of Countries by GDP Class')
plt.xlabel('GDP Class')
plt.ylabel('Number of Countries')
plt.show()
```



In [18]:
```
cause_columns = df.columns.drop(['Entity', 'Code', 'Year', 'GDP (constant 2015 U

gdp_cause_means = df.groupby('GDP_Class')[cause_columns].mean()
mean_deaths_by_cause = gdp_cause_means.mean()
top_10_causes = mean_deaths_by_cause.sort_values(ascending=False).head(10).index

gdp_cause_means[top_10_causes].T.plot(kind='bar', figsize=(14, 7), title='Top 10
plt.ylabel('Average Number of Deaths')
plt.show()
```

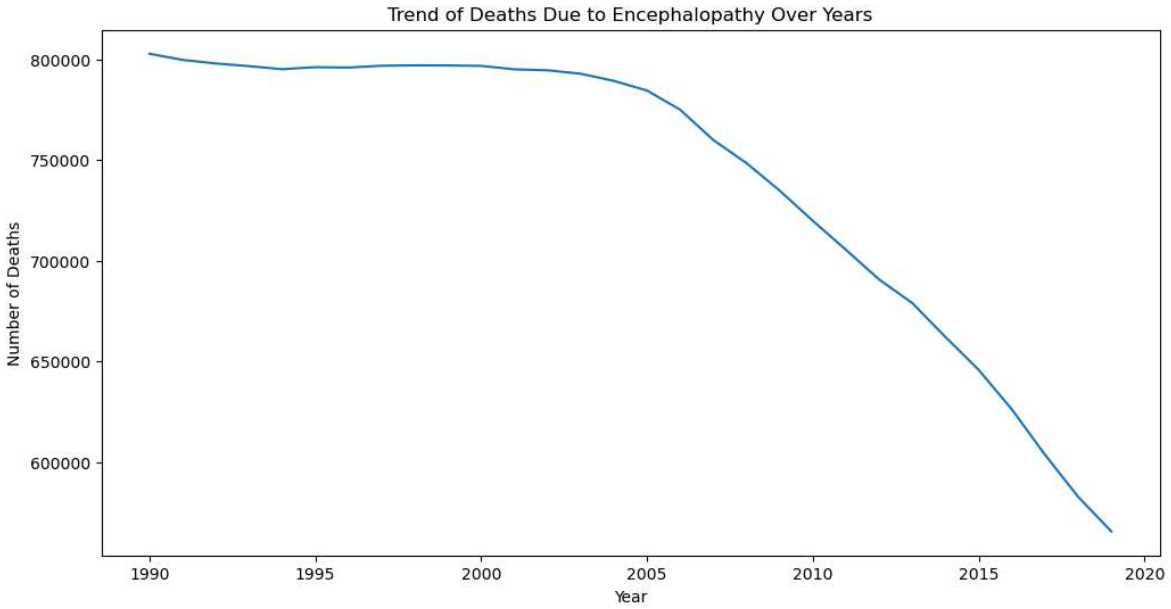


In [19]:
```
total_deaths_by_cause = df[cause_columns].sum()
top_10_causes = total_deaths_by_cause.sort_values(ascending=False).head(10).inde
top_10_causes_df = df[['Year'] + list(top_10_causes)]
trend_data = top_10_causes_df.groupby('Year')[top_10_causes].sum().reset_index()

plt.figure(figsize=(14, 7))
for cause in top_10_causes:
    sns.lineplot(x='Year', y=cause, data=trend_data, label=cause)

plt.title('Trend of Deaths for Top 10 Causes Over Years')
plt.ylabel('Number of Deaths')
plt.xlabel('Year')
plt.legend(title='Cause of Death')
plt.show()
```

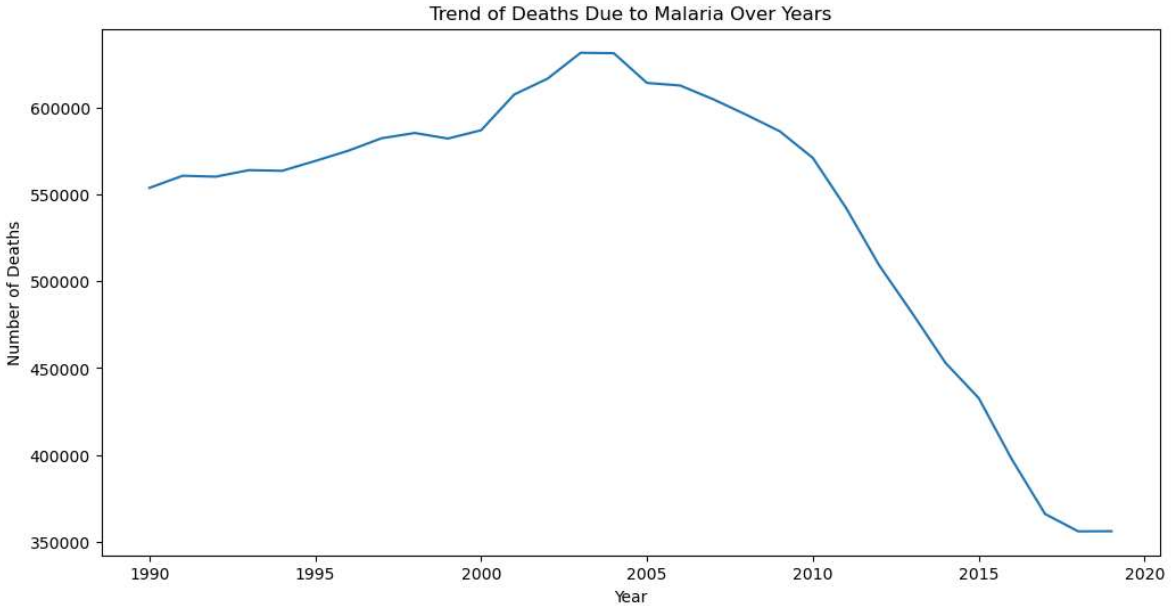


In [20]:
```
cause = 'Encephalopathy'
trend_data = df.groupby('Year')[cause].sum().reset_index()
plt.figure(figsize=(12, 6))
sns.lineplot(x='Year', y=cause, data=trend_data)
```

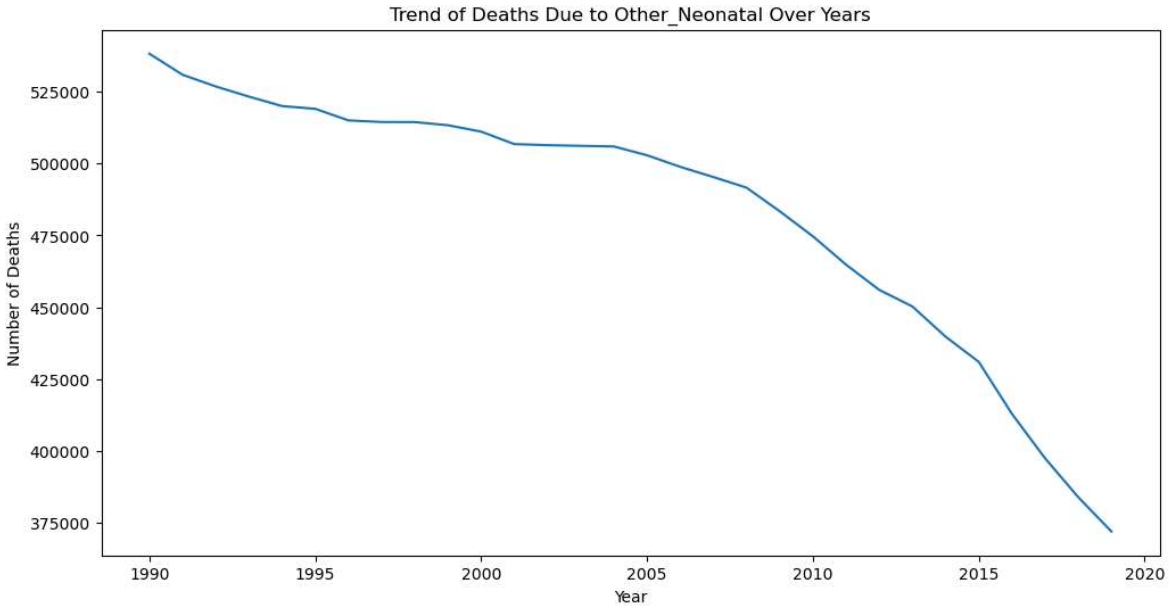
```
plt.title(f'Trend of Deaths Due to {cause} Over Years')
plt.ylabel('Number of Deaths')
plt.xlabel('Year')
plt.show()
```



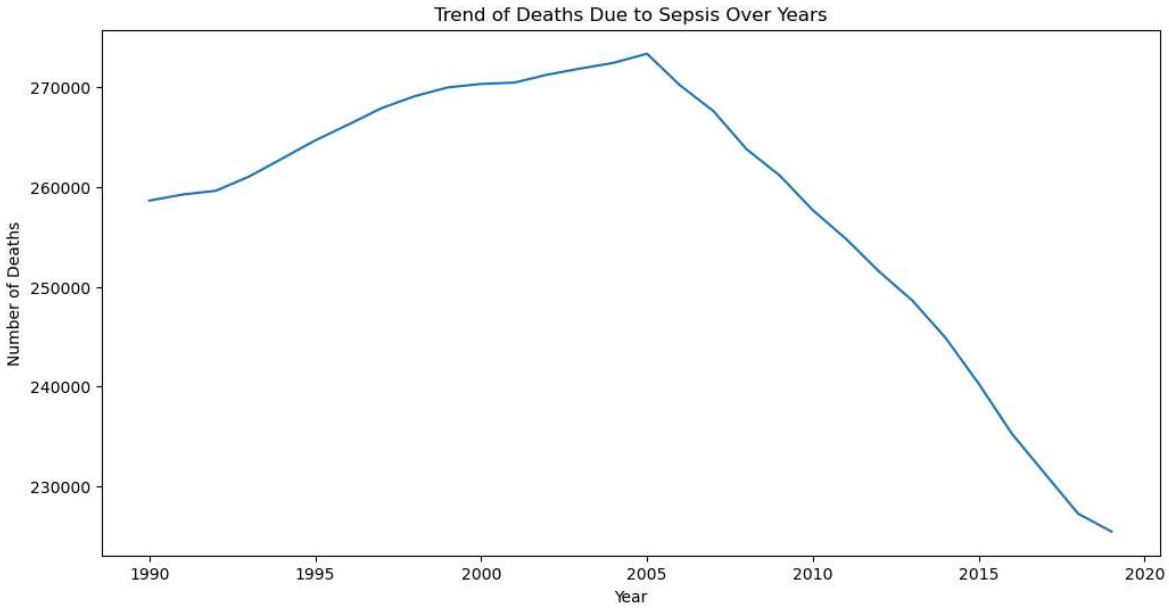
In [21]:
```
cause = 'Malaria'
trend_data = df.groupby('Year')[cause].sum().reset_index()
plt.figure(figsize=(12, 6))
sns.lineplot(x='Year', y=cause, data=trend_data)
plt.title(f'Trend of Deaths Due to {cause} Over Years')
plt.ylabel('Number of Deaths')
plt.xlabel('Year')
plt.show()
```



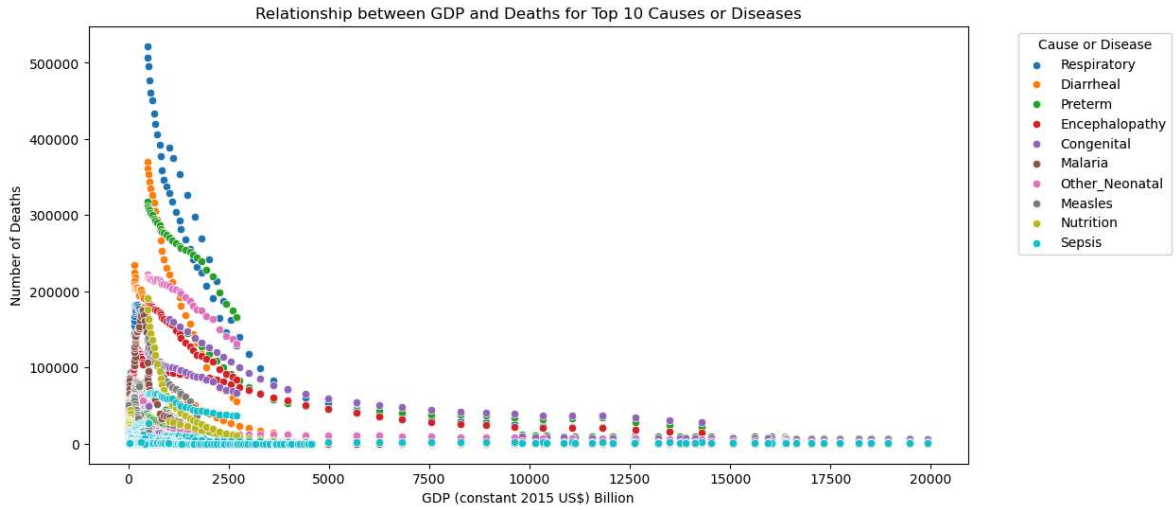
In [22]:
```
cause = 'Other_Neonatal'
trend_data = df.groupby('Year')[cause].sum().reset_index()
plt.figure(figsize=(12, 6))
sns.lineplot(x='Year', y=cause, data=trend_data)
plt.title(f'Trend of Deaths Due to {cause} Over Years')
plt.ylabel('Number of Deaths')
plt.xlabel('Year')
plt.show()
```



In [23]:
```
cause = 'Sepsis'
trend_data = df.groupby('Year')[cause].sum().reset_index()
plt.figure(figsize=(12, 6))
sns.lineplot(x='Year', y=cause, data=trend_data)
plt.title(f'Trend of Deaths Due to {cause} Over Years')
plt.ylabel('Number of Deaths')
plt.xlabel('Year')
plt.show()
```



In [24]:
```
total_deaths_cause = df[cause_columns].sum()
top_10_causes = total_deaths_cause.sort_values(ascending=False).head(10).index

plt.figure(figsize=(12, 6))
for cause in top_10_causes:
    sns.scatterplot(x='GDP (constant 2015 US$) Billion', y=cause, data=df, label

plt.title('Relationship between GDP and Deaths for Top 10 Causes or Diseases')
plt.xlabel('GDP (constant 2015 US$) Billion')
plt.ylabel('Number of Deaths')
plt.legend(title='Cause or Disease', bbox_to_anchor=(1.05, 1), loc='upper left')
plt.show()
```



Random Forest Classifier

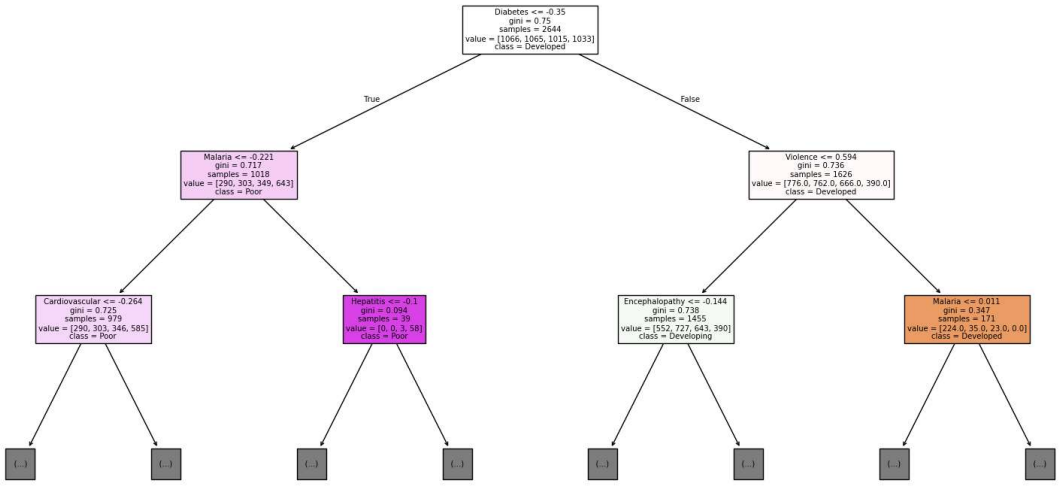
In [25]:
```
from sklearn.preprocessing import StandardScaler
X = df.drop(['Entity', 'Code', 'Year', 'GDP (constant 2015 US$) Billion', 'GDP_C
y = df['GDP_Class']

scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0.3,
```

In [26]:
```
# RF Classifier

param_grid = {
    'n_estimators': [10, 20, 30],
    'max_features': [2, 4, 6]
}

rf = RandomForestClassifier(random_state=123)
grid_search = GridSearchCV(estimator=rf, param_grid=param_grid, cv=5, scoring='a
grid_search.fit(X_train, y_train)

best_rf = grid_search.best_estimator_
print("Best parameters for Random Forest:", grid_search.best_params_)

y_pred_rf = best_rf.predict(X_test)

train_accuracy = best_rf.score(X_train, y_train)
print("Train Accuracy:", train_accuracy)

test_accuracy = accuracy_score(y_test, y_pred_rf)
print("Test Accuracy:", test_accuracy)

train_error = 1 - train_accuracy
print("Train Error:", train_error)

test_error = 1 - test_accuracy
print("Test Error:", test_error)

print("\nConfusion Matrix:")
print(confusion_matrix(y_test, y_pred_rf))

y_pred_rf = best_rf.predict(X_test)
```

```
print("\nRandom Forest Classifier:")
print(classification_report(y_test, y_pred_rf))

accuracy = accuracy_score(y_test, y_pred_rf)
print("Accuracy:", accuracy)
```

```
Best parameters for Random Forest: {'max_features': 2, 'n_estimators': 30}
Train Accuracy: 0.9964106245513281
Test Accuracy: 0.966499162479062
Train Error: 0.003589375448671883
Test Error: 0.03350083752093802

Confusion Matrix:
[[440   5   0   0]
 [  5 412  13   0]
 [  0  13 408  17]
 [  0   0   7 471]]

Random Forest Classifier:
              precision    recall  f1-score   support

           1       0.99      0.99      0.99       445
           2       0.96      0.96      0.96       430
           3       0.95      0.93      0.94       438
           4       0.97      0.99      0.98       478

    accuracy                           0.97      1791
   macro avg       0.97      0.97      0.97      1791
weighted avg       0.97      0.97      0.97      1791

Accuracy: 0.966499162479062
```

In [48]:
```
### Tree Generation
feature_names_list = X.columns.tolist()

plt.figure(figsize=(20,10))
plot_tree(best_rf.estimators_[0], feature_names=feature_names_list, class_names=
plt.show()
```

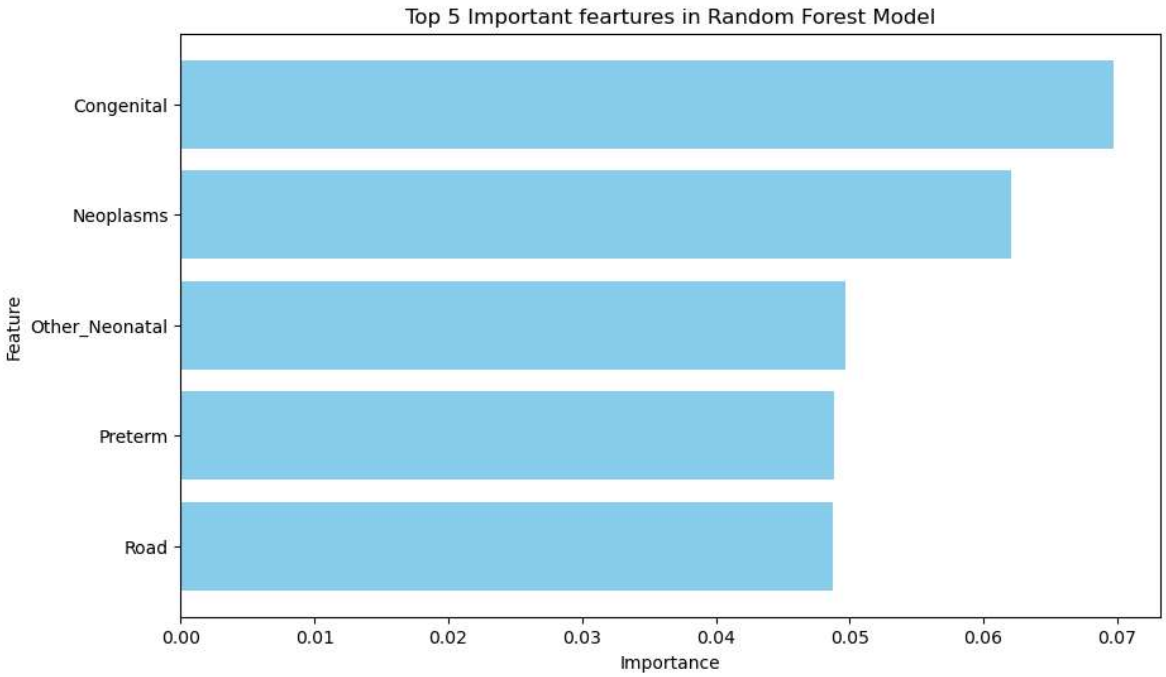


In [28]:
```
# Variable Importance
importances = best_rf.feature_importances_
feature_importance_df = pd.DataFrame({'Feature': X.columns, 'Importance': import
```

```
feature_importance_df = feature_importance_df.sort_values(by='Importance', ascen
print(feature_importance_df)
```

```
           Feature  Importance
10      Congenital    0.069775
4        Neoplasms    0.062078
19  Other_Neonatal    0.049724
12         Preterm    0.048874
23            Road    0.048786
11     Respiratory    0.047980
2        Nutrition    0.043319
21       Diarrheal    0.041974
1         Violence    0.040609
14          Sepsis    0.040272
17  Encephalopathy    0.039841
28        Syphilis    0.036030
26        Drowning    0.034981
24    Tuberculosis    0.034558
9   Cardiovascular    0.032886
18      Meningitis    0.032394
8            Kidney    0.032104
20  Whooping_Cough    0.028862
6         Digestive    0.028702
5           Measles    0.027313
7         Cirrhosis    0.026738
25         HIV_AIDS    0.024016
22        Fire_Heat    0.024001
27          Malaria    0.023921
16         Diabetes    0.022629
3         Hepatitis    0.018600
0              INTS    0.015426
13        Heat_Cold    0.014665
15           Nature    0.008940
```

In [29]:
```
# Top 5 Important Vraiables
importances = best_rf.feature_importances_
indices = np.argsort(importances)[::-1]

top_indices = indices[:5]
top_features = [feature_names_list[i] for i in top_indices]
top_importances = importances[top_indices]

plt.figure(figsize=(10, 6))
plt.barh(top_features, top_importances, color='skyblue')
plt.xlabel('Importance')
plt.ylabel('Feature')
plt.title('Top 5 Important feartures in Random Forest Model')
plt.gca().invert_yaxis()
plt.show()
```



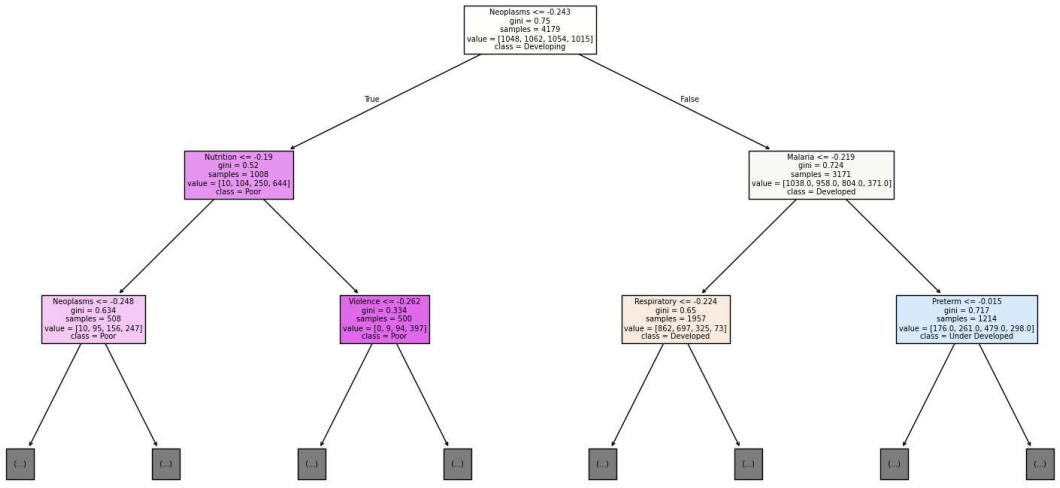
Bagging with DT

In [30]:
```
base_dt = DecisionTreeClassifier(random_state=42)

param_grid = {
    'n_estimators': [10, 20, 30],
    'max_features': [X_train.shape[1]]
}

grid_search_bagging_dt = GridSearchCV(estimator=BaggingClassifier(base_dt, randc
                                      param_grid=param_grid, cv=5, scoring='accu
grid_search_bagging_dt.fit(X_train, y_train)

best_bagging_dt = grid_search_bagging_dt.best_estimator_
print("Best parameters for Decision Tree Bagging Classifier:", grid_search_baggi

y_pred_bagging_dt_tuned = best_bagging_dt.predict(X_test)

accuracy_bagging_dt_tuned = accuracy_score(y_test, y_pred_bagging_dt_tuned)
print("Accuracy:", accuracy_bagging_dt_tuned)

train_accuracy_bagging_dt = best_bagging_dt.score(X_train, y_train)
print("Train Accuracy:", train_accuracy_bagging_dt)

test_accuracy_bagging_dt = accuracy_score(y_test, y_pred_bagging_dt_tuned)
print("Test Accuracy:", test_accuracy_bagging_dt)

train_error_bagging_dt = 1 - train_accuracy_bagging_dt
print("Train Error:", train_error_bagging_dt)

test_error_bagging_dt = 1 - test_accuracy_bagging_dt
print("Test Error:", test_error_bagging_dt)

print("\nConfusion Matrix:")
print(confusion_matrix(y_test, y_pred_bagging_dt_tuned))

print("\nClassification Report:")
print(classification_report(y_test, y_pred_bagging_dt_tuned))
```

```
accuracy_bagging_dt_tuned = accuracy_score(y_test, y_pred_bagging_dt_tuned)
print("Accuracy:", accuracy_bagging_dt_tuned)
```

```
Best parameters for Decision Tree Bagging Classifier: {'max_features': 29, 'n_est
imators': 20}
Accuracy: 0.9581239530988275
Train Accuracy: 0.9966499162479062
Test Accuracy: 0.9581239530988275
Train Error: 0.00335008375209378
Test Error: 0.041876046901172526

Confusion Matrix:
[[437   8   0   0]
 [  8 409  13   0]
 [  0  16 403  19]
 [  0   0  11 467]]

Classification Report:
              precision    recall  f1-score   support

           1       0.98      0.98      0.98       445
           2       0.94      0.95      0.95       430
           3       0.94      0.92      0.93       438
           4       0.96      0.98      0.97       478

    accuracy                           0.96      1791
   macro avg       0.96      0.96      0.96      1791
weighted avg       0.96      0.96      0.96      1791

Accuracy: 0.9581239530988275
```

In [31]:
```
# Tree model for DBagging with Decision Tree Classifier
base_decision_tree = DecisionTreeClassifier(random_state=42)
base_decision_tree.fit(X_train, y_train)
plt.figure(figsize=(20, 10))
plot_tree(base_decision_tree, feature_names=feature_names_list, class_names=['De
plt.show()
```



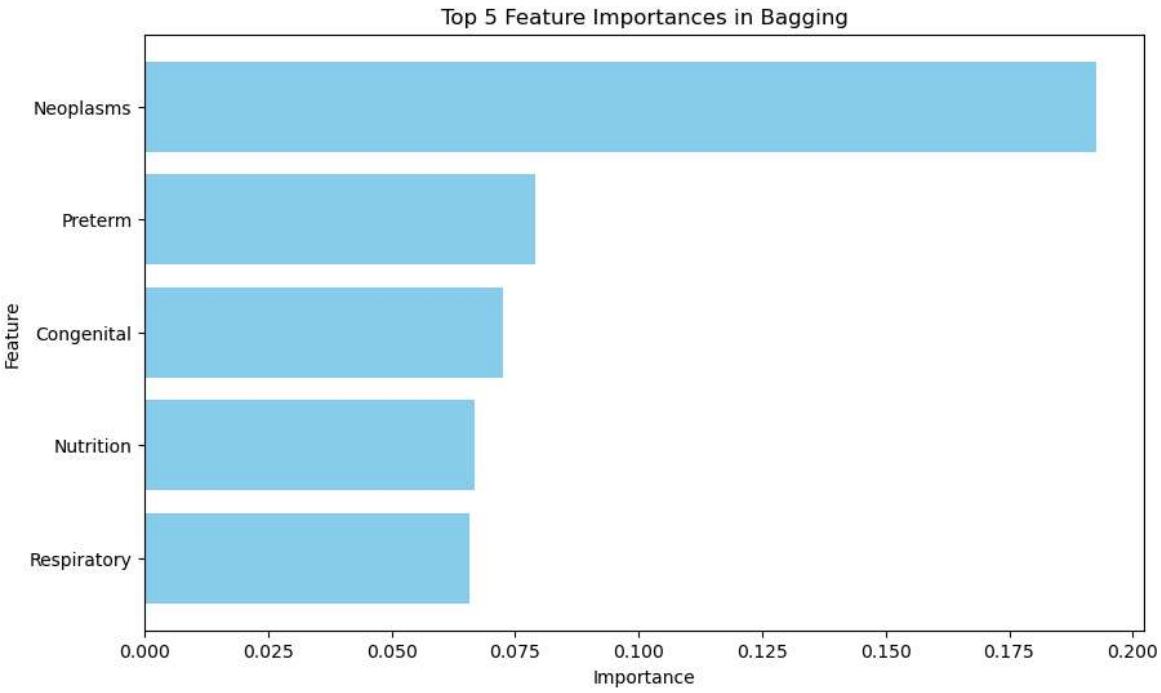
In [32]:
```
feature_importances = np.mean([
    tree.feature_importances_ for tree in best_bagging_dt.estimators_
], axis=0)
```

```
feature_importance_df = pd.DataFrame({'Feature': df.drop(['Entity', 'Code', 'Yea
feature_importance_df = feature_importance_df.sort_values(by='Importance', ascen
feature_importance_df
```

Out[32]:

	Feature	Importance
4	Neoplasms	0.192679
12	Preterm	0.078977
10	Congenital	0.072614
2	Nutrition	0.066883
11	Respiratory	0.065775
27	Malaria	0.064986
14	Sepsis	0.040569
23	Road	0.035831
28	Syphilis	0.032055
19	Other_Neonatal	0.031675
17	Encephalopathy	0.029466
24	Tuberculosis	0.027141
6	Digestive	0.023607
1	Violence	0.022334
21	Diarrheal	0.021957
3	Hepatitis	0.020395
25	HIV_AIDS	0.019921
18	Meningitis	0.019879
5	Measles	0.018076
8	Kidney	0.017858
20	Whooping_Cough	0.016745
9	Cardiovascular	0.016164
0	INTS	0.015029
26	Drowning	0.013933
16	Diabetes	0.011883
22	Fire_Heat	0.010266
7	Cirrhosis	0.006315
13	Heat_Cold	0.005440
15	Nature	0.001548

In [33]:
```
# Top 5 Features
top_features = feature_importance_df.head(5)

plt.figure(figsize=(10, 6))
plt.barh(top_features['Feature'], top_features['Importance'], color='skyblue')
plt.xlabel('Importance')
plt.ylabel('Feature')
plt.title('Top 5 Feature Importances in Bagging')
plt.gca().invert_yaxis()
plt.show()
```



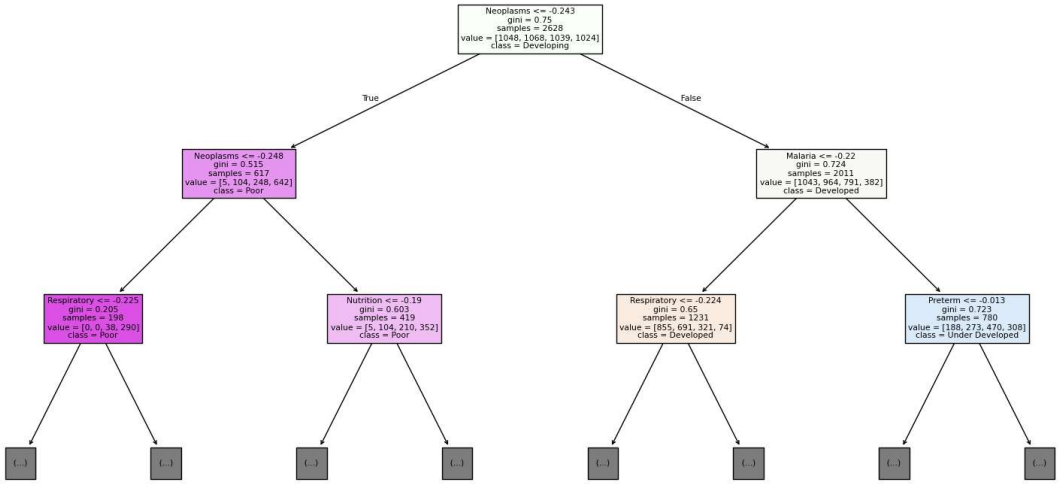
Boosting

In [34]:
```
boosting_classifier = GradientBoostingClassifier(random_state=42)

param_grid = {
    'n_estimators': [10, 20, 30],
    'max_depth': [2, 5, 10]
}

grid_search_boosting = GridSearchCV(estimator=boosting_classifier, param_grid=pa
grid_search_boosting.fit(X_train, y_train)

best_boosting = grid_search_boosting.best_estimator_
print("Best parameters for Gradient Boosting Classifier:", grid_search_boosting.

y_pred_boosting = best_boosting.predict(X_test)

train_accuracy_boosting = best_boosting.score(X_train, y_train)
print("Train Accuracy:", train_accuracy_boosting)

test_accuracy_boosting = accuracy_score(y_test, y_pred_boosting)
print("Test Accuracy:", test_accuracy_boosting)

train_error_boosting = 1 - train_accuracy_boosting
print("Train Error:", train_error_boosting)
```

```
test_error_boosting = 1 - test_accuracy_boosting
print("Test Error:", test_error_boosting)

print("\nConfusion Matrix:")
print(confusion_matrix(y_test, y_pred_boosting))

print("\nClassification Report:")
print(classification_report(y_test, y_pred_boosting))

accuracy_boosting = accuracy_score(y_test, y_pred_boosting)
print("Accuracy:", accuracy_boosting)
```

```
Best parameters for Gradient Boosting Classifier: {'max_depth': 10, 'n_estimator
s': 30}
Train Accuracy: 0.9968892079444843
Test Accuracy: 0.9642657733109995
Train Error: 0.0031107920555156765
Test Error: 0.0357342266890005

Confusion Matrix:
[[439   6   0   0]
 [  8 411  11   0]
 [  0  13 407  18]
 [  0   0   8 470]]

Classification Report:
              precision    recall  f1-score   support

           1       0.98      0.99      0.98       445
           2       0.96      0.96      0.96       430
           3       0.96      0.93      0.94       438
           4       0.96      0.98      0.97       478

    accuracy                           0.96      1791
   macro avg       0.96      0.96      0.96      1791
weighted avg       0.96      0.96      0.96      1791

Accuracy: 0.9642657733109995
```

In [35]:
```
# Tree Model for Bossting with Decision Tree
first_tree = best_bagging_dt.estimators_[0]

plt.figure(figsize=(20, 10))
plt.title("Decision Tree from Bagging Classifier")
plot_tree(first_tree, feature_names=feature_names_list, class_names=['Developed'
plt.show()
```

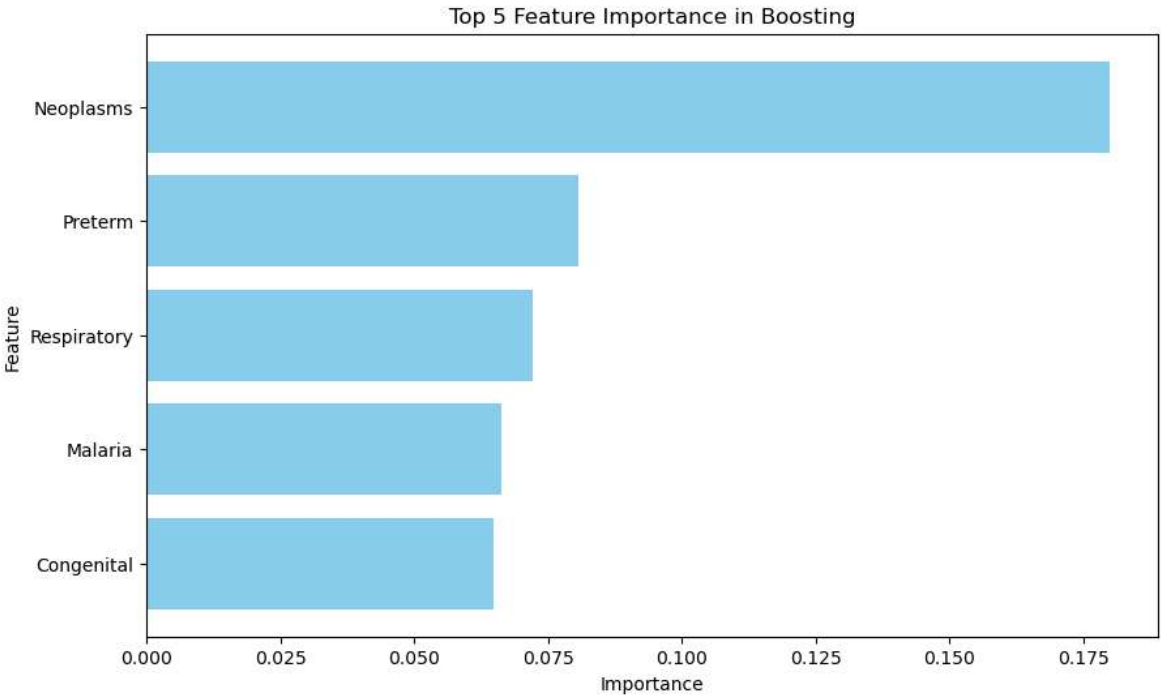


In [36]:
```
# Variable Importance
feature_importance = best_boosting.feature_importances_
sorted_indices = np.argsort(feature_importance)[::-1]
sorted_feature_names = np.array(feature_names_list)[sorted_indices]
sorted_feature_importance = feature_importance[sorted_indices]

for feature_name, importance in zip(sorted_feature_names, sorted_feature_importa
    print(f"{feature_name}: {importance}")
```

```
Neoplasms: 0.17992199227656835
Preterm: 0.0807519300583025
Respiratory: 0.07206716549280796
Malaria: 0.06630149941899859
Congenital: 0.06471835605125777
Nutrition: 0.06402706915568504
Violence: 0.052260501782867085
Sepsis: 0.04476979246066699
Other_Neonatal: 0.034296961284263676
Encephalopathy: 0.032753561806686676
Diarrheal: 0.02973225869674273
INTS: 0.02712802335708643
Road: 0.0216972537062967
Syphilis: 0.021160910311902193
Tuberculosis: 0.020325554081741724
HIV_AIDS: 0.02022936083941837
Cardiovascular: 0.019345362439555595
Hepatitis: 0.017240474709927348
Kidney: 0.016681640853162948
Drowning: 0.01622021736463195
Whooping_Cough: 0.014907898600452902
Diabetes: 0.013811817775610996
Digestive: 0.01355474901073077
Fire_Heat: 0.013248381794397898
Meningitis: 0.012690036671419063
Measles: 0.011942306048320313
Cirrhosis: 0.00986320598928833
Heat_Cold: 0.008024045176888619
Nature: 0.0003276727843204986
```

In [37]:
```
# Top 5 Important Feature in Boosting
feature_importance = best_boosting.feature_importances_
sorted_indices = np.argsort(feature_importance)[::-1]
```

```
top_features = 5
plt.figure(figsize=(10, 6))
plt.barh(range(top_features), feature_importance[sorted_indices][:top_features][
plt.yticks(range(top_features), np.array(feature_names_list)[sorted_indices][:to
plt.xlabel('Importance')
plt.ylabel('Feature')
plt.title('Top 5 Feature Importance in Boosting')
plt.show()
```



Regression Models

Random Forest Regressor

In [38]:
```
X = df[[  'INTS', 'Violence', 'Nutrition', 'Hepatitis', 'Neoplasms',
        'Measles', 'Digestive', 'Cirrhosis', 'Kidney', 'Cardiovascular', 'Congen
        'Preterm', 'Heat_Cold', 'Sepsis', 'Nature', 'Diabetes', 'Encephalopathy'
        'Other_Neonatal', 'Whooping_Cough', 'Diarrheal', 'Fire_Heat', 'Road', 'T
        'Drowning', 'Malaria', 'Syphilis']]

y = df.drop(['Entity', 'Code', 'Year', 'GDP (constant 2015 US$) Billion', 'GDP_C

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_

scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

param_grid = {
    'n_estimators': [10, 20, 30],
    'max_depth': [10, 20, 30],
}

rf_regressor = RandomForestRegressor(random_state=42)
grid_search = GridSearchCV(estimator=rf_regressor, param_grid=param_grid, cv=5,
grid_search.fit(X_train_scaled, y_train)

best_params = grid_search.best_params_
```

```
best_rf_regressor = RandomForestRegressor(**best_params, random_state=42)

best_rf_regressor.fit(X_train_scaled, y_train)

y_pred = best_rf_regressor.predict(X_test_scaled)

# Model Evaluation
mse = mean_squared_error(y_test, y_pred)
mae = mean_absolute_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

print("Best hyperparameters:", best_params)
print("Mean Squared Error (MSE):", mse)
print("Mean Absolute Error (MAE):", mae)
print("R-squared:", r2)
```

```
Best hyperparameters: {'max_depth': 10, 'n_estimators': 30}
Mean Squared Error (MSE): 525904.2801559122
Mean Absolute Error (MAE): 138.2279608017326
R-squared: 0.9432340032741553
```

In [39]:

```
# Death Number Prediction for 2019
df_2019 = df[df['Year'] == 2019]

X_2019 = df_2019[['GDP (constant 2015 US$) Billion', 'INTS', 'Violence', 'Nutrit
                  'Measles', 'Digestive', 'Cirrhosis', 'Kidney', 'Cardiovascular
                  'Preterm', 'Heat_Cold', 'Sepsis', 'Nature', 'Diabetes', 'Encep
                  'Other_Neonatal', 'Whooping_Cough', 'Diarrheal', 'Fire_Heat',
                  'Drowning', 'Malaria', 'Syphilis']]

X_2019_scaled = scaler.transform(X_2019.drop('GDP (constant 2015 US$) Billion',
predictions_2019 = best_rf_regressor.predict(X_2019_scaled)
predictions_2019_rounded = predictions_2019.round().astype(int)

for i, disease in enumerate(X_2019.columns[1:]):
    df_2019[disease + '_Predicted_Deaths_2019'] = predictions_2019_rounded[:, i]

df_2019
```

Out[39]:

	Entity	Code	Year	GDP (constant 2015 US$) Billion	INTS	Violence	Nutrition	Hepatitis	Neopl
29	Afghanistan	AFG	2019	21.118474	174	240	925	307	
89	Albania	ALB	2019	12.967696	0	1	2	0	
119	Algeria	DZA	2019	177.355540	8	9	27	10	
149	American Samoa	ASM	2019	0.628818	0	0	0	0	
179	Andorra	AND	2019	3.008968	0	0	0	0	
...	...	...	...	...	...	...	...	...	
6479	Vanuatu	VUT	2019	0.864199	0	0	4	0	
6539	Vietnam	VNM	2019	314.947630	21	18	46	4	
6779	Yemen	YEM	2019	43.374423	56	35	542	53	
6809	Zambia	ZMB	2019	24.089862	45	71	986	14	
6839	Zimbabwe	ZWE	2019	20.621079	108	31	1598	15	

199 rows × 63 columns

In [40]:
```
# Comparision between Precited and Actual Deaths
df_2019['Actual_Deaths_2019'] = df_2019[df_2019['Year'] == 2019][['INTS', 'Viole
                                                                  'Measles', 'D
                                                                  'Congenital',
                                                                  'Diabetes', '
                                                                  'Diarrheal',
                                                                  'Malaria', 'S

df_country_totals = df_2019.groupby('Entity')['Actual_Deaths_2019'].sum().reset_
df_country_totals['Predicted_Deaths_2019'] = predictions_2019_rounded.sum(axis=1

df_country_totals = df_country_totals.merge(df_2019[['Entity', 'GDP_Class']], on
df_country_totals = df_country_totals[['Entity', 'Predicted_Deaths_2019', 'Actua
df_country_totals
```

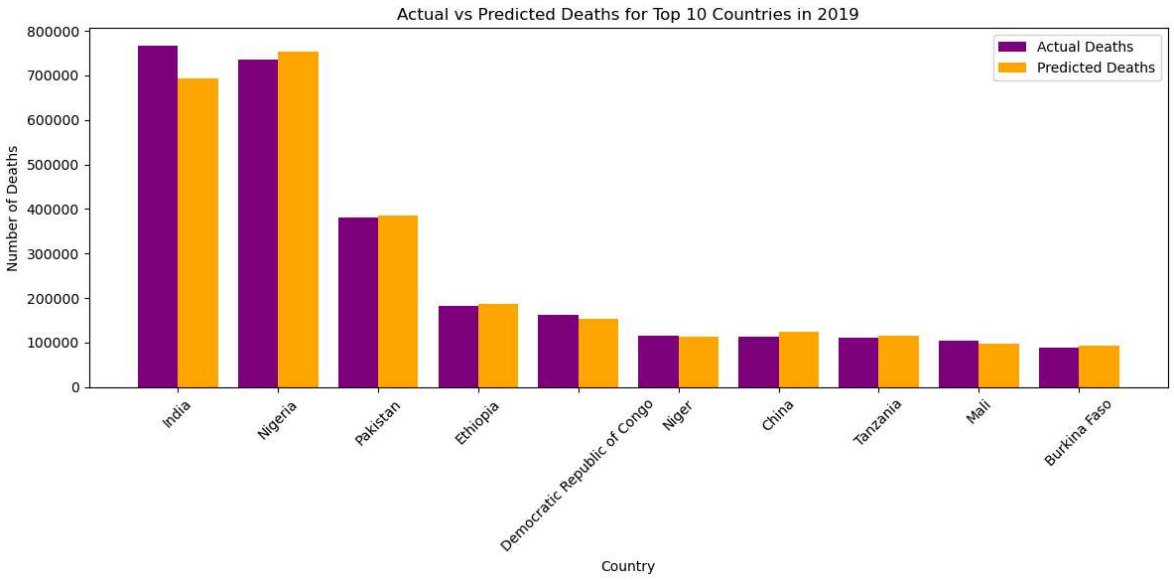
Out[40]:

	Entity	Predicted_Deaths_2019	Actual_Deaths_2019	GDP_Class
0	Afghanistan	78705	68608	2
1	Albania	364	399	3
2	Algeria	16425	16401	1
3	American Samoa	33	7	4
4	Andorra	33	0	4
...	...	...	...	...
194	Vanuatu	116	177	4
195	Vietnam	15206	15340	1
196	Yemen	41779	40669	2
197	Zambia	30709	30295	2
198	Zimbabwe	20329	22811	2

199 rows × 4 columns

In [41]:

```
# Actual Numbers Vs Predicted Numbers
top_10_countries = df_country_totals.nlargest(10, 'Actual_Deaths_2019')

index = np.arange(len(top_10_countries['Entity']))

bar_width = 0.4

plt.figure(figsize=(12, 6))
plt.bar(index, top_10_countries['Actual_Deaths_2019'], width=bar_width, color='p
plt.bar(index + bar_width, top_10_countries['Predicted_Deaths_2019'], width=bar_

plt.xlabel('Country')
plt.ylabel('Number of Deaths')
plt.title('Actual vs Predicted Deaths for Top 10 Countries in 2019')
plt.xticks(index + bar_width / 2, top_10_countries['Entity'], rotation=45)
plt.legend()
plt.tight_layout()
plt.show()
```



Decision Tree Regressor Model

In [42]:
```
# Model Fitting
param_grid = {
    'max_depth': [10, 20, 30],
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 2, 4]
}

grid_search = GridSearchCV(DecisionTreeRegressor(random_state=42), param_grid, c

grid_search.fit(X_train_scaled, y_train)

best_dt_model = grid_search.best_estimator_
best_params = grid_search.best_params_

y_pred_dt_tuned = best_dt_model.predict(X_test_scaled)

# Model Evaluation

mse_dt = mean_squared_error(y_test, y_pred_dt_tuned)
mae_dt = mean_absolute_error(y_test, y_pred_dt_tuned)
r2_dt = r2_score(y_test, y_pred_dt_tuned)

print("Best parameters found:", best_params)
print("Mean Squared Error (MSE) for Decision Tree:", mse_dt)
print("Mean Absolute Error (MAE) for Decision Tree:", mae_dt)
print("R-squared for Decision Tree:", r2_dt)
```

```
Fitting 5 folds for each of 27 candidates, totalling 135 fits
Best parameters found: {'max_depth': 10, 'min_samples_leaf': 1, 'min_samples_spli
t': 2}
Mean Squared Error (MSE) for Decision Tree: 1668201.5326730837
Mean Absolute Error (MAE) for Decision Tree: 191.10441659045765
R-squared for Decision Tree: 0.8928360280722066
```

In [43]:
```
# Prection of Each Causes for 2019
df_2019_dt = df[df['Year'] == 2019]

X_2019_dt = df_2019_dt[['GDP (constant 2015 US$) Billion', 'INTS', 'Violence', '
                        'Measles', 'Digestive', 'Cirrhosis', 'Kidney', 'Cardiova
```

```
                        'Preterm', 'Heat_Cold', 'Sepsis', 'Nature', 'Diabetes',
                        'Other_Neonatal', 'Whooping_Cough', 'Diarrheal', 'Fire_H
                        'Drowning', 'Malaria', 'Syphilis']]

X_2019_gdp = X_2019_dt.drop('GDP (constant 2015 US$) Billion', axis=1)

X_2019_scaled_dt = scaler.transform(X_2019_gdp)
predictions_2019_dt = best_dt_model.predict(X_2019_scaled_dt)

for i, disease in enumerate(X_2019_gdp.columns):
    df_2019_dt[disease + '_Predicted_Deaths_2019_DT'] = predictions_2019_dt[:, i

df_2019_dt
```

Out[43]:

	Entity	Code	Year	GDP (constant 2015 US$) Billion	INTS	Violence	Nutrition	Hepatitis	Neopl
29	Afghanistan	AFG	2019	21.118474	174	240	925	307	
89	Albania	ALB	2019	12.967696	0	1	2	0	
119	Algeria	DZA	2019	177.355540	8	9	27	10	
149	American Samoa	ASM	2019	0.628818	0	0	0	0	
179	Andorra	AND	2019	3.008968	0	0	0	0	
...	...	...	...	...	...	...	...	...	
6479	Vanuatu	VUT	2019	0.864199	0	0	4	0	
6539	Vietnam	VNM	2019	314.947630	21	18	46	4	
6779	Yemen	YEM	2019	43.374423	56	35	542	53	
6809	Zambia	ZMB	2019	24.089862	45	71	986	14	
6839	Zimbabwe	ZWE	2019	20.621079	108	31	1598	15	

199 rows × 63 columns

In [44]:

```
# Comparision of Predicted Values and Actual Values
df_2019['Actual_Deaths_2019'] = df_2019[df_2019['Year'] == 2019][['INTS', 'Viole
                                                                  'Measles', 'D
                                                                  'Congenital',
                                                                  'Diabetes', '
                                                                  'Diarrheal',
                                                                  'Malaria', 'S


df_country_totals = df_2019.groupby('Entity')['Actual_Deaths_2019'].sum().reset_

df_country_totals['Predicted_Deaths_2019'] = predictions_2019_dt.round().sum(axi

df_country_totals = df_country_totals.merge(df_2019[['Entity', 'GDP_Class']], on

df_country_totals = df_country_totals[['Entity', 'Predicted_Deaths_2019', 'Actua
```

```
df_country_totals
```

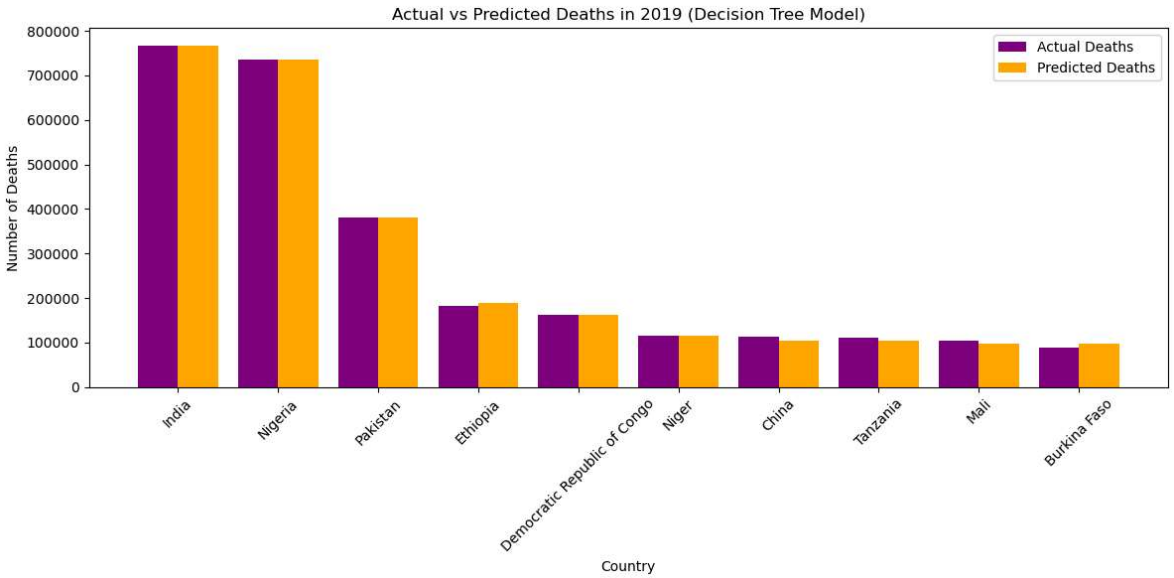
Out[44]:

	Entity	Predicted_Deaths_2019	Actual_Deaths_2019	GDP_Class
0	Afghanistan	79187.0	68608	2
1	Albania	369.0	399	3
2	Algeria	17604.0	16401	1
3	American Samoa	26.0	7	4
4	Andorra	26.0	0	4
...	...	...	...	...
194	Vanuatu	140.0	177	4
195	Vietnam	17000.0	15340	1
196	Yemen	42213.0	40669	2
197	Zambia	40736.0	30295	2
198	Zimbabwe	21134.0	22811	2

199 rows × 4 columns

In [45]:

```
# Comparision Graph
top_10_countries_dt = df_country_totals.nlargest(10, 'Actual_Deaths_2019')
index = np.arange(len(top_10_countries_dt['Entity']))
bar_width = 0.4

plt.figure(figsize=(12, 6))
plt.bar(index, top_10_countries_dt['Actual_Deaths_2019'], width=bar_width, color
plt.bar(index + bar_width, top_10_countries_dt['Predicted_Deaths_2019'], width=b

plt.xlabel('Country')
plt.ylabel('Number of Deaths')
plt.title('Actual vs Predicted Deaths in 2019 (Decision Tree Model)')
plt.xticks(index + bar_width / 2, top_10_countries_dt['Entity'], rotation=45)
plt.legend()

plt.tight_layout()
plt.show()
```



In [46]:
```
# Consolidated Results of All the Classifcation Model
results = [
    {
        "Model": "Random Forest",
        "Train Accuracy": train_accuracy,
        "Test Accuracy": test_accuracy,
        "Train Error": train_error,
        "Test Error": test_error
    },
    {
        "Model": "Decision Tree Bagging",
        "Train Accuracy": train_accuracy_bagging_dt,
        "Test Accuracy": test_accuracy_bagging_dt,
        "Train Error": train_error_bagging_dt,
        "Test Error": test_error_bagging_dt
    },
    {
        "Model": "Gradient Boosting",
        "Train Accuracy": train_accuracy_boosting,
        "Test Accuracy": test_accuracy_boosting,
        "Train Error": train_error_boosting,
        "Test Error": test_error_boosting
    }
]

classification_results = pd.DataFrame(results)
print("Classification Results:")
print(tabulate(classification_results, headers='keys', tablefmt='pretty', showin
```

```
Classification Results:
+----------------------+--------------------+--------------------+-------------
---------+----------------------+
|         Model        |   Train Accuracy   |   Test Accuracy    |      Train Er
ror      |      Test Error      |
+----------------------+--------------------+--------------------+-------------
---------+----------------------+
|     Random Forest    | 0.9964106245513281 | 0.966499162479062  | 0.00358937544
8671883  | 0.03350083752093802  |
| Decision Tree Bagging | 0.9966499162479062 | 0.9581239530988275 |  0.0033500837
5209378  | 0.041876046901172526 |
|   Gradient Boosting   | 0.9968892079444843 | 0.9642657733109995 | 0.00311079205
55156765 |  0.0357342266890005  |
+----------------------+--------------------+--------------------+-------------
---------+----------------------+
```

In [47]:
```
# Consolidated Results for Regression Models
results_regression = [
    {
        "Model": "Random Forest",
        "R2 Score": r2,
        "Mean Squared Error": mse,
        "Mean Absolute Error": mae
    },
    {
        "Model": "Decision Tree",
        "R2 Score": r2_dt,
        "Mean Squared Error": mse_dt,
        "Mean Absolute Error": mae_dt
    },

]

regression_results = pd.DataFrame(results_regression)
print("Regression Results:")
print(tabulate(regression_results, headers='keys', tablefmt='pretty', showindex=
```

```
Regression Results:
+---------------+--------------------+--------------------+---------------------+
|     Model     |      R2 Score      | Mean Squared Error | Mean Absolute Error |
+---------------+--------------------+--------------------+---------------------+
| Random Forest | 0.9432340032741553 | 525904.2801559122  |  138.2279608017326  |
| Decision Tree | 0.8928360280722066 | 1668201.5326730837 | 191.10441659045765  |
+---------------+--------------------+--------------------+---------------------+
```

In [ ]: