Health and Development Indicators: Global Insights

Sushruthi Panakanti[1, *], Meghana Darla[1], Sai Mahidhar Reddy Byreddy[1], Sudharani Yeruva[1], and Shamily Reddy Muddam[1]

[1] Luddy School of Informatics, Indiana University-Purdue University Indianapolis, IN 46202, USA

spanakan@iu.edu, mdarla@iu.edu, saibyred@iu.edu, syeruva@iu.edu, shmuddam@iu.edu

Abstract: Life expectancy varies widely between developed and developing nations. This study utilizes the "Health & Development Indicators: Global Insights" dataset to identify factors influencing life expectancy globally. Correlation and regression analyses will examine relationships between life expectancy and indicators including GDP, education, healthcare spending, and environment. Predictive modeling will estimate potential lifespan gains from improving key factors. Visualizations and statistical analyses will reveal the strongest drivers of life expectancy and highlight priorities for investment to promote longevity worldwide.

Keywords: Life expectancy · Public health · Global health · Social determinants of health · Health indicators

1 Project Scope

1.1 Introduction

Life expectancy, as we all know, is the average life span of an individual. In the pre-modern era, life expectancy was approximately 30 years regardless of global location (Khan et al., 2016).
Although life expectancy started to climb in the early 19th century, it remained low in various regions, highlighting disparities in health standards. However, during the 20th century, these global disparities in life expectancy began to diminish. According to Khan et al. (2016), current life expectancy is on a consistent upward trajectory, with many nations nearing an average of 70 to 75 years. In today's world, there exists a stark contrast in life expectancies between developed and developing countries. People in developed nations tend to live longer, healthier lives compared to their counterparts in developing regions. This global inequality in life expectancy has drawn significant attention, yet a comprehensive analysis of the factors contributing to this phenomenon remains elusive. Khan et al. (2016), based on their study, concluded that financial resources, healthcare quality and availability, and everyday living conditions are the key contributors to variations in life expectancy among developed and developing countries across the globe. To identify this disparity, we are using the dataset by Jangir (2023) to develop a predictive model. With data spanning multiple years and countries, this dataset provides a comprehensive view of how different nations fare in terms of public health and development status.

1.2 Aim

Our aim is to identify the factors that have the greatest impact on life expectancy in different countries. This can be achieved by analyzing the correlations between life expectancy and various factors such as GDP, education, healthcare spending, and environmental factors. The results of this analysis can be used to inform public policy and healthcare interventions aimed at improving life expectancy in different countries.

Our Research Hypothesis include:

Null Hypothesis (H0)
There is no significant linear relationship between any of the independent variables (adult mortality, alcohol, income composition of resources, hepatitis B, country, HIV/AIDS, year, schooling, BMI, GDP, status, total expenditure) and life expectancy.

Alternate Hypothesis (H1)

At least one of the independent variables has a significant linear relationship with life expectancy.

1.3 Purpose

In this era of global interconnectedness, where health crises, resource allocation, and policy decisions have far-reaching consequences, the "Health & Development Indicators: Global Insights" dataset provides a valuable resource for research, analysis, and evidence-based decision-making in the fields of public health and international development. The purpose of this study is to utilize this dataset to identify the factors that have the greatest impact on life expectancy across different countries. The study will examine the correlations between life expectancy and potential influencing factors such as GDP, education, healthcare spending, and environmental indicators. Statistical analysis will be utilized to determine the relative effects of these factors. The goal is to inform public health policy and interventions to improve life expectancy by understanding which factors show the strongest associations. The results may highlight priorities for investment and systemic changes needed to extend lifespan in different national contexts. Through this analysis of the intricate relationship between health and development indicators, the aim is to provide actionable insights to guide policies and programs to increase life expectancy globally.

2 Methodology

2.1 Steps of the Project

This study will employ a quantitative research approach, specifically using descriptive statistics. This approach is observational and analytical, meaning it will observe and analyze existing data rather than conducting experiments or interventions. Descriptive research seeks to describe the status of an identified variable, recognizing trends and patterns in data. This choice involves working with numerical data to understand patterns, relationships, and trends. It is crucial for our study because we're analyzing various numerical indicators' impact on life expectancy across different countries.

The main focus of our project will be to compare the association between factors like alcohol, adult mortality, income, hepatitis B, HIV/AIDS, year, schooling, BMI, GDP, status and total expenditure with life expectancy using database technologies such as SQL and Python. The tools which we used are Python JupyterHub and phpMyAdmin.

The methodology of our project involves four stages, namely:

1. Data Collection
2. Data Pre-processing
3. Data Analysis using machine learning models
4. Data Visualization

2.2 Original Team Members and Responsibilities

Our team comprised of individuals from diverse backgrounds, possessing unique interests and skills. We originally divided the tasks amongst ourselves. Below is the list of our team members and the responsibilities that were distributed at the start of the project:

Team Member	Background	Responsibilities
Meghana Darla	Bachelor of Dental Surgery	Data extraction using SQL Normality test using Python Binary classification model development
Sushruthi Panakanti	Bachelor of Dental Surgery	Clustering analysis Data visualization of clustering Evaluating binary classification (Logistic Regression) using Python Visualization of linear regression model using Python Matplotlib

Sai Mahidhar Reddy	Bachelor of Dental Surgery	Data visualization of normality test results using Python Seaborn and Matplotlib (seaborn pairplots, histogram, heatmap) Evaluating binary classification (Support Vector Machine) using Python Visualization of SVM using Python
Sudha Rani Yeruva	Masters in nursing	Evaluating binary classification (Naives Bayes Classifier) using Python Visualization of Naives Bayes classifier using Python Final project report
Shamily Reddy	Bachelor of Dental Surgery	Project proposal Risk prediction model development Evaluating risks by risk score using Python

Table 1. Original Responsibilities of Team Members

2.3 Actual Contributions from Individual Team Members

After we started to work on our respective assigned tasks, we realized that the above table was not completely representative of responsibilities of each team member due to unforeseen changes. We also realized that some of the machine learning models that we assumed we shall be using initially, would actually not be applicable to our project, so we had to revise the responsibilities accordingly all over again. Below is the updated table of how we actually worked:

Team Member	**Background**	**Responsibilities**
Sushruthi Panakanti	Bachelor of Dental Surgery	Project Manager, Model accuracy testing, Proofreading, Project Presentation
Meghana Darla	Bachelor of Dental Surgery	Data Collection and pre-processing, Data Analysis, Finding and Citing Sources, Project Report
Sai Mahidhar Reddy	Bachelor of Dental Surgery	Exploratory Data Analysis and Visualization, Normality testing, Machine learning integration
Sudha Rani Yeruva	Masters in Nursing	Data analysis and visualization, Machine learning integration
Shamily Reddy	Bachelor of Dental Surgery	Data analysis and visualization, Proofreading

Table 2. Revised Responsibilities of Team Members

2.4 Challenges

Initially, our team faced challenges in coordinating meetings due to the demands of hectic coursework, resulting in a lag in the progress of the project. However, this difficulty was resolved, and subsequent meetings were conducted effectively, allowing us to successfully complete the assigned tasks.

Another limitation was the outdated dataset. The dataset's time range, spanning approximately from 2000 to 2015, might not reflect the most recent global health developments, potentially affecting the relevance of findings.

A significant challenge encountered by most of us was unfamiliarity with programming. We initially encountered difficulties in acquiring or applying certain technical skills required for data analysis, machine learning, or visualization. However, we created a collaborative environment within our team, encouraging

members to share their expertise. Those who excel in technical skills are actively supporting others, offering guidance and assistance in learning and applying specific techniques.

Additionally, linear regression, which we initially thought would be a good fit for our dataset, did not provide valuable results and accuracy, because of which we had to perform hyperparameter tuning with ridge regression using GridsearchCV.

3. Data collection

The dataset was taken from Kaggle, https://www.kaggle.com/datasets/arunjangir245/life-expectancy-data/datalife- (Health & Development Indicators: Global Insights). This dataset provides a deep understanding of various aspects and to identify the factors that have the greatest impact on life expectancy in different countries.

This dataset consists of one csv file named Life_Expectancy_Data.csv.

4. Data extraction and Data storage

4.1 Data Extraction:

The first step in the data extraction and storage stage was selection of variables from our life expectancy data csv file (Life_Expectancy_Data.csv). We did a thorough review of the variables in this file and chose the variables that were most in line with our aim. Our aim was to identify the factors that have the greatest impact on life expectancy in different countries. This can be achieved by analyzing the correlations between life expectancy and various factors such as GDP, education, healthcare spending, and environmental factors. The results of this analysis can be used to inform public policy and healthcare interventions aimed at improving life expectancy in different countries.

The variables we selected are below:

1. Adult_Mortality
2. Alcohol
3. Income_composition_of_resources
4. Hepatitis_B
5. Country
6. _HIV/AIDS
7. Year
8. Schooling
9. _BMI_
10. GDP
11. Status
12. Total_expenditure

4.2 Data Cleaning:

The data cleaning process involved reviewing the attributes we selected. This was a very necessary step because our data set does not contain null and negative values. As part of this cleaning process, we reviewed each row of data in Excel to identify problematic values.

```
print(data.isnull().sum())
```

```
Country                          0
Year                             0
Status                           0
Life expectancy                  0
Adult Mortality                  0
infant deaths                    0
Alcohol                          0
percentage expenditure           0
Hepatitis B                      0
Measles                          0
 BMI                             0
under-five deaths                0
Polio                            0
Total expenditure                0
Diphtheria                       0
 HIV/AIDS                        0
GDP                              0
Population                       0
 thinness  1-19 years            0
 thinness 5-9 years              0
Income composition of resources  0
Schooling                        0
dtype: int64
```

Fig 1: It shows no null or negative values

4.3 Data Import:

After the variables in the Excel file were cleaned, we started the process to import the data into a single database called “Life_expectancy_data.csv” in phpMyAdmin. We got shared access to phpMyAdmin from the professor, and we used this shared database for the import. All our detailed queries for the data import process are located in the Appendix. We started by importing our dataset as CSV (Comma-Separated Values) files into new notebook in phpMyAdmin. Once our dataset was complete, we made sure to connect our database to Python Jupyter Notebook which allowed us to move onto our data analysis and code implementation. A screenshot of the structure for the table “Life_expectancy_data.csv” is below.

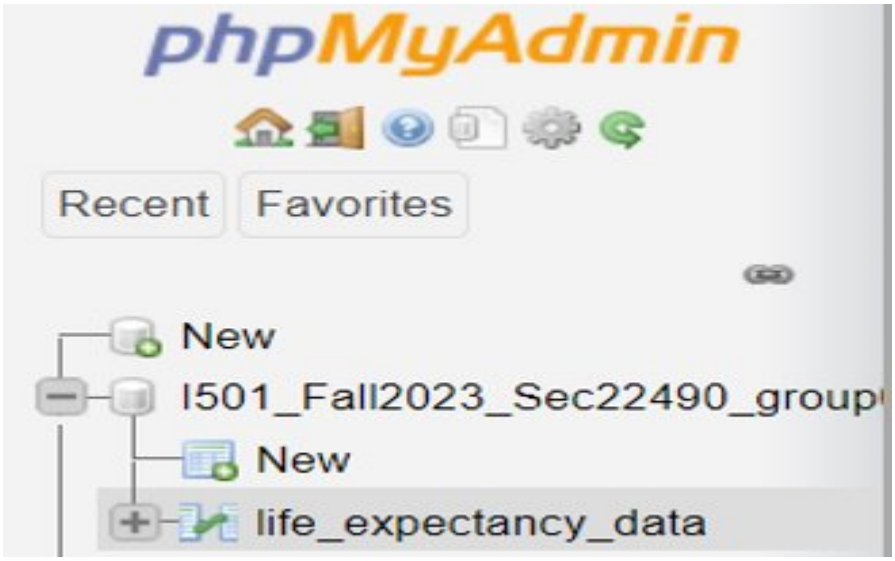

Fig 2: Uploaded life_expectancy_data.csv in phpMyAdmin

	Country	Year	Status	Life expectancy	Adult Mortality	infant deaths	Alcohol	percentage expenditure	Hepatitis B	Measles	...	Polio	Total expenditure	Diphtheria	HIV/AIDS	GDP	P
0	Afghanistan	2015	Developing	65.0	263	62	0.01	71.279624	65	1154	...	6	8.16	65	0.1	584.259210	3
1	Afghanistan	2014	Developing	59.9	271	64	0.01	73.523582	62	492	...	58	8.18	62	0.1	612.696514	
2	Afghanistan	2013	Developing	59.9	268	66	0.01	73.219243	64	430	...	62	8.13	64	0.1	631.744976	3
3	Afghanistan	2012	Developing	59.5	272	69	0.01	78.184215	67	2787	...	67	8.52	67	0.1	669.959000	
4	Afghanistan	2011	Developing	59.2	275	71	0.01	7.097109	68	3013	...	68	7.87	68	0.1	63.537231	

5 rows × 22 columns

Fig 3: Reading data using .head() function

4.4 Defining the variables:

We have defined the selected variables (adult mortality, alcohol, income composition of resources, hepatitis B, country, HIV/AIDS, year, schooling, BMI, GDP, status, total expenditure) and our target variable life expectancy.

```
selected_features = ['Adult_Mortality', 'Alcohol', 'Income_composition_of_resources',
                     'Hepatitis_B', 'Country' , '_HIV/AIDS', 'Year', 'Schooling', '_BMI_', 'GDP', 'Status', 'Total_expenditure']
target_variable = 'Life_expectancy_'
```

```
print(selected_features)
```

```
['Adult_Mortality', 'Alcohol', 'Income_composition_of_resources', 'Hepatitis_B', 'Country', '_HIV/AIDS', 'Year', 'Schooling', '_BMI_', 'GDP',
'Status', 'Total_expenditure']
```

```
data_selected = data[selected_features]
```

Fig 4: Code that defines the selected variable and target variable.

```
selected_data.describe()
```

	Adult_Mortality	Alcohol	Income_composition_of_resources	Hepatitis_B	_HIV/AIDS	Year	Schooling	_BMI_	GDP	Total_expenditur
count	1649.000000	1649.000000	1649.000000	1649.000000	1649.000000	1649.000000	1649.000000	1649.000000	1649.000000	1649.00000
mean	168.215282	4.533196	0.631551	79.217708	1.983869	2007.840509	12.119891	38.128623	5566.031887	5.95592
std	125.310417	4.029189	0.183089	25.604664	6.032360	4.087711	2.795388	19.754249	11475.900117	2.29938
min	1.000000	0.010000	0.000000	2.000000	0.100000	2000.000000	4.200000	2.000000	1.681350	0.74000
25%	77.000000	0.810000	0.509000	74.000000	0.100000	2005.000000	10.300000	19.500000	462.149650	4.41000
50%	148.000000	3.790000	0.673000	89.000000	0.100000	2008.000000	12.300000	43.700000	1592.572182	5.84000
75%	227.000000	7.340000	0.751000	96.000000	0.700000	2011.000000	14.000000	55.800000	4718.512910	7.47000
max	723.000000	17.870000	0.936000	99.000000	50.600000	2015.000000	20.700000	77.100000	119172.741800	14.39000

Fig 5: Quick quantitative review of data.

We cleaned the “Life_expectancy_data.csv” data set. Once we had the file saved, we uploaded it to phpMyAdmin. We then connected the introdataset table to Python Jupyter Notebook.

5 Data Analysis

Python was used for data analysis in which linear regression was employed to analyze the data. We implemented a predictive model Linear regression, Random Forest, and Decision tree technique for the data analysis. Below are some of the steps we took to effectively analyze the data:

5.1 Analyzing the outliers.

Step1: Detecting the outliers: We have used the statical method Interquartile range for detecting the outliers. First, we select only the numeric columns (integers and floats) from the DataFrame using the select_dtypes () method. This keeps only the columns containing quantitative data for analyzing outliers.

Next, we calculate the interquartile range (IQR) for each numeric column. The IQR gives the range between the 25th and 75th percentiles (Q1 and Q3). It represents the middle 50% of the data and provides a way to detect outlier values. We calculate Q1 and Q3 using the quantile () method. Then we take their difference to obtain the IQR.

We then set lower and upper bounds to identify outliers using the IQR. Values below Q1 - 1.5IQR are considered low outliers, while values above Q3 + 1.5IQR are considered high outliers. The factor of 1.5 is commonly used to detect outliers.

Using these thresholds, we detect outlier values in each column by checking if they are below the lower bound or above the upper bound. The sum () aggregates the total count of outliers across all numeric columns.

Finally, we print out the count of outliers detected in each numeric column to display the number of outliers identified in the dataset.

```
# Select only numeric columns for outlier detection
numeric_columns = selected_data.select_dtypes(include=['int64', 'float64'])

# Calculate the IQR for each numeric column
Q1 = numeric_columns.quantile(0.25)
Q3 = numeric_columns.quantile(0.75)
IQR = Q3 - Q1

# Detect outliers for each numeric column
lower_bound = Q1 - 1.5 * IQR
upper_bound = Q3 + 1.5 * IQR

outliers_count = ((numeric_columns < lower_bound) | (numeric_columns > upper_bound)).sum()

# Display the number of outliers in each numeric column
print("Number of outliers in each numeric column:")
print(outliers_count)
```

Fig 6: Code that performs the detection of outliers using IQR.

```
Number of outliers in each numeric column:
Adult_Mortality                   54
Alcohol                            2
Income_composition_of_resources    48
Hepatitis_B                       165
_HIV/AIDS                         299
Year                               0
Schooling                         16
_BMI_                              0
GDP                               208
Total_expenditure                  7
Life_expectancy_                  39
dtype: int64
```

Fig 7: Displaying the outliers in each selected variable.

Step2: Plotting the Outliers using the Boxplots:

We provided a visual representation of the distribution of numeric columns, particularly focusing on identifying outliers through box plots for each individual column in the dataset.

```python
import matplotlib.pyplot as plt
import seaborn as sns
# Identify numerical columns
numerical_columns = selected_data.select_dtypes(include='number')

# Calculate the number of rows and columns for subplots
num_rows = len(numerical_columns) // 2 + len(numerical_columns) % 2
num_cols = 2

# Set up subplots
fig, axes = plt.subplots(nrows=len(numerical_columns.columns) // 2 + len(numerical_columns.columns) % 2, ncols=2, figsize=(10, 3 * (len(num
fig.subplots_adjust(hspace=0.5)

# Iterate through each numerical column
for i, column in enumerate(numerical_columns.columns):
    # Box plot for numerical variables with outliers
    sns.boxplot(x=column, data=numerical_columns, ax=axes[i // 2, i % 2])
    axes[i // 2, i % 2].set_title(f'{column} Box Plot with Outliers')

# Remove empty subplots if the number of numerical columns is odd
if len(numerical_columns.columns) % 2 != 0:
    fig.delaxes(axes[-1, -1])

plt.show()
```

Fig 8: Python code creating boxplots using matplotlib and seaborn.

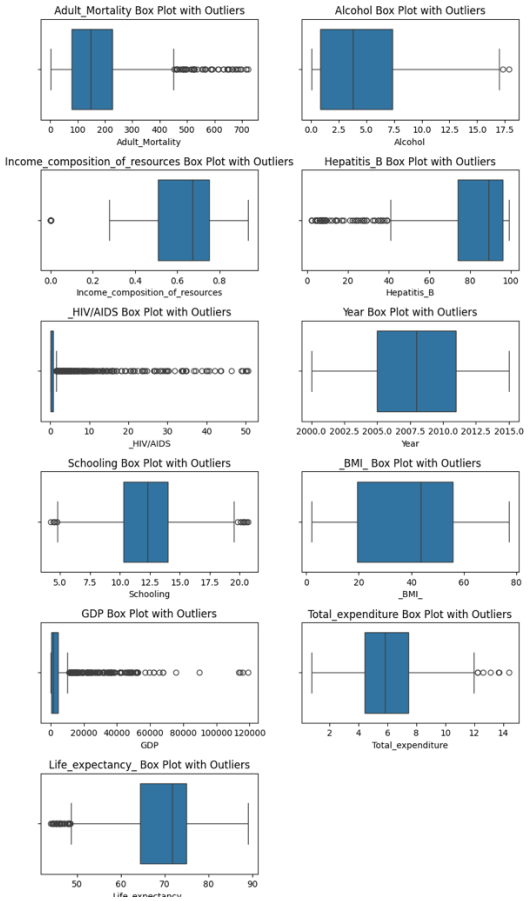

Fig 9: Boxplots of the selected variables with outliers.

Step3: Handling the outliers.

We performed outlier capping or clipping for each column in the selected_data DataFrame based on the calculated lower and upper bounds obtained using the Interquartile Range (IQR) method.

We handled the outliers by setting extreme values to the nearest reasonable boundaries based on the calculated IQR, thus mitigating the impact of outliers on subsequent analysis or modeling.

```
Q1 = numerical_columns.quantile(0.25)
Q3 = numerical_columns.quantile(0.75)
IQR = Q3 - Q1
# Define the lower and upper bounds for capping outliers
lower_bound = Q1 - 1.5 * IQR
upper_bound = Q3 + 1.5 * IQR
# Cap outliers for each column
for column in numerical_columns.columns:
    selected_data.loc[selected_data[column] < lower_bound[column], column] = lower_bound[column]
    selected_data.loc[selected_data[column] > upper_bound[column], column] = upper_bound[column]
```

Fig 10: Code that performs the capping of the Outliers using the IQR.

Step 4: Detecting the outliers after Capping:

```
# Select only numeric columns for outlier detection
numeric_columns = selected_data.select_dtypes(include=['int64', 'float64'])

# Calculate the IQR for each numeric column
Q1 = numeric_columns.quantile(0.25)
Q3 = numeric_columns.quantile(0.75)
IQR = Q3 - Q1

# Detect outliers for each numeric column
lower_bound = Q1 - 1.5 * IQR
upper_bound = Q3 + 1.5 * IQR

outliers_count = ((numeric_columns < lower_bound) | (numeric_columns > upper_bound)).sum()

# Display the number of outliers in each numeric column
print("Number of outliers in each numeric column:")
print(outliers_count)
```

Fig 11: Code that detects the outliers from the selected variables.

```
Number of outliers in each numeric column:
Adult_Mortality                  0
Alcohol                          0
Income_composition_of_resources  0
Hepatitis_B                      0
_HIV/AIDS                        0
Year                             0
Schooling                        0
_BMI_                            0
GDP                              0
Total_expenditure                0
Life_expectancy_                 0
dtype: int64
```

Fig 12: Number of outliers after capping

After capping the outliers, we have visualized the data using boxplots.

```
import matplotlib.pyplot as plt
import seaborn as sns
# Identify numerical columns
numerical_columns = selected_data.select_dtypes(include='number')
# Calculate the number of rows and columns for subplots
num_rows = len(numerical_columns) // 2 + len(numerical_columns) % 2
num_cols = 2
# Set up subplots
fig, axes = plt.subplots(nrows=len(numerical_columns.columns) // 2 + len(numerical_columns.columns) % 2, ncols=2, figsize=(10, 3 * (len(num
fig.subplots_adjust(hspace=0.5)
boxplot_color = 'orange'
# Iterate through each numerical column
for i, column in enumerate(numerical_columns.columns):
    # Box plot for numerical variables with outliers
    sns.boxplot(x=column, data=numerical_columns, ax=axes[i // 2, i % 2], color=boxplot_color)
    axes[i // 2, i % 2].set_title(f'{column} Box Plot with Outliers')

# Remove empty subplots if the number of numerical columns is odd
if len(numerical_columns.columns) % 2 != 0:
    fig.delaxes(axes[-1, -1])
plt.show()
```

Fig 13: Python code that creates the box plots using matplotlib and seaborn

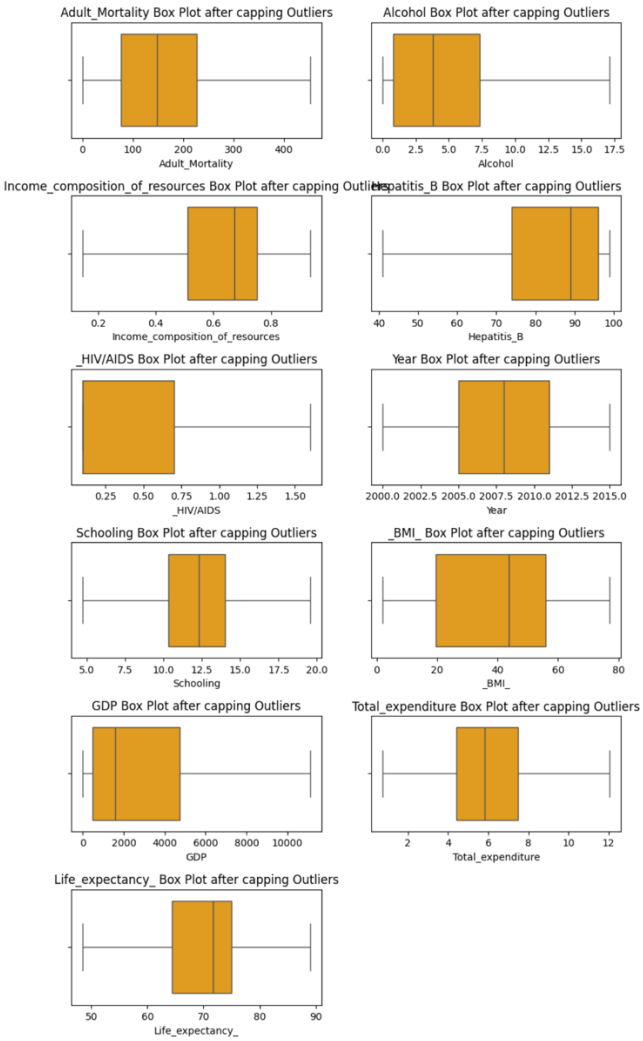

Fig 14: Boxplots of the selected variables after capping the outliers.

5.2 Descriptive Statistics

We have performed a descriptive statistics using the .describe() function that provides the count, mean, standard deviation, minimum and maximum values, 25th, 50th, 75th percentiles in each column.

```
selected_data.describe()
```

	Adult_Mortality	Alcohol	Income_composition_of_resources	Hepatitis_B	_HIV/AIDS	Year	Schooling	_BMI_	GDP	To
count	1649.000000	1649.000000	1649.000000	1649.000000	1649.000000	1649.000000	1649.000000	1649.000000	1649.000000	
mean	164.469375	4.532644	0.635801	82.018799	0.483445	2007.840509	12.116192	38.128623	3301.756961	
std	113.645940	4.027419	0.169567	18.574411	0.588738	4.087711	2.779930	19.754249	3670.000460	
min	1.000000	0.010000	0.146000	41.000000	0.100000	2000.000000	4.750000	2.000000	1.681350	
25%	77.000000	0.810000	0.509000	74.000000	0.100000	2005.000000	10.300000	19.500000	462.149650	
50%	148.000000	3.790000	0.673000	89.000000	0.100000	2008.000000	12.300000	43.700000	1592.572182	
75%	227.000000	7.340000	0.751000	96.000000	0.700000	2011.000000	14.000000	55.800000	4718.512910	
max	452.000000	17.135000	0.936000	99.000000	1.600000	2015.000000	19.550000	77.100000	11103.057800	

Fig 15: Descriptive statistics using .describe() function.

We have extracted important summary statistics like mean, median, and standard deviation for the numerical columns within the data frame.

These are the fundamental measures in statistics that provide a crucial insight into the characteristics and distribution of the dataset.

```
# Display mean, median, and standard deviation
summary_stats = selected_data.describe()
numeric_columns = selected_data.select_dtypes(include=['number'])
summary_stats = numeric_columns.describe()

print("Mean:")
print(summary_stats.loc['mean'])

print("\nMedian:")
print(numeric_columns.median())

print("\nStandard Deviation:")
print(summary_stats.loc['std'])
```

Fig 16: Python code that finds the mean, median, and standard deviation of the selected variables.

```
Mean:
Adult_Mortality                      164.469375
Alcohol                                4.532644
Income_composition_of_resources        0.635801
Hepatitis_B                           82.018799
_HIV/AIDS                              0.483445
Year                                2007.840509
Schooling                             12.116192
_BMI_                                 38.128623
GDP                                 3301.756961
Total_expenditure                      5.951340
Life_expectancy_                      69.355185
Name: mean, dtype: float64

Median:
Adult_Mortality                      148.000000
Alcohol                                3.790000
Income_composition_of_resources        0.673000
Hepatitis_B                           89.000000
_HIV/AIDS                              0.100000
Year                                2008.000000
Schooling                             12.300000
_BMI_                                 43.700000
GDP                                 1592.572182
Total_expenditure                      5.840000
Life_expectancy_                      71.700000
dtype: float64

Standard Deviation:
Adult_Mortality                      113.645940
Alcohol                                4.027419
Income_composition_of_resources        0.169567
Hepatitis_B                           18.574411
_HIV/AIDS                              0.588738
Year                                   4.087711
Schooling                              2.779930
_BMI_                                 19.754249
GDP                                 3670.000460
Total_expenditure                      2.285542
Life_expectancy_                       8.661437
Name: std, dtype: float64
```

Fig 17: Output of mean, median, and standard deviation of the selected variables.

5.3 Steps to Test the Normality of the Data

Step 1: To test whether the datapoints in the selected variables are normally distributed, we have used the Shapiro-wilk test.

```python
from scipy.stats import shapiro

numerical_features = selected_data[numerical_columns]

# Shapiro-Wilk test for each numerical feature
for column in numerical_columns:
    stat, p_value = shapiro(numerical_features[column])

    # Set a significance level (e.g., 0.05)
    alpha = 0.05

    # Print the results
    print(f'{column}:')
    print(f'Statistic: {stat}, p-value: {p_value}')

    # Check the null hypothesis
    if p_value > alpha:
        print('Result: Data looks normally distributed (fail to reject H0)\n')
    else:
        print('Result: Data does not look normally distributed (reject H0)\n')
```

Fig 18: Python that performs the Shapiro Test.

```
Adult_Mortality:
Statistic: 0.9416123032569885, p-value: 5.395622223322862e-25
Result: Data does not look normally distributed (reject H0)

Alcohol:
Statistic: 0.9099046587944031, p-value: 3.5255539417319726e-30
Result: Data does not look normally distributed (reject H0)

Income_composition_of_resources:
Statistic: 0.9559358358383179, p-value: 6.606685090112496e-22
Result: Data does not look normally distributed (reject H0)

Hepatitis_B:
Statistic: 0.7929079532623291, p-value: 1.4109674237286583e-41
Result: Data does not look normally distributed (reject H0)

_HIV/AIDS:
Statistic: 0.6428849697113037, p-value: 0.0
Result: Data does not look normally distributed (reject H0)

Year:
Statistic: 0.9531741142272949, p-value: 1.4840763217098546e-22
Result: Data does not look normally distributed (reject H0)

Schooling:
Statistic: 0.9937988519668579, p-value: 2.1570883745880565e-06
Result: Data does not look normally distributed (reject H0)

_BMI_:
Statistic: 0.9313482046127319, p-value: 7.19120593781619e-27
Result: Data does not look normally distributed (reject H0)

GDP:
Statistic: 0.787646472454071, p-value: 5.97513665188102e-42
Result: Data does not look normally distributed (reject H0)

Total_expenditure:
Statistic: 0.9913502335548401, p-value: 2.7222270659876813e-08
Result: Data does not look normally distributed (reject H0)

Life_expectancy_:
Statistic: 0.9636785984039307, p-value: 6.520848945688781e-20
Result: Data does not look normally distributed (reject H0)

In all cases, the null hypothesis (H0) is rejected, suggesting that the data is not normally distributed based on the Shapiro-Wilk test
```

Fig 19: Output of Shapiro test indicating that data is not normally distributed.

All the p-values are extremely low (far less than 0.05), indicating strong evidence against the null hypothesis of normality. Thus, based on the Shapiro-Wilk test, we have concluded that none of the selected variables seem to follow a normal distribution.

Step2: Visualizing the datapoints of the selected variables by using the Q-Q Plots and Histograms.

```python
from scipy.stats import probplot
numerical_features = selected_data[numerical_columns]
# Set up subplots for histograms and Q-Q plots
fig, axes = plt.subplots(nrows=2, ncols=len(numerical_features.columns), figsize=(70, 20))
# Plot histograms
for i, column in enumerate(numerical_features.columns):
    sns.histplot(data=numerical_features, x=column, bins=50, kde=True, ax=axes[0, i])
    axes[0, i].set_title(f'Distribution of {column}')
    axes[0, i].set_ylabel('Frequency')
# Q-Q plots
for i, column in enumerate(numerical_features.columns):
    probplot(numerical_features[column], dist="norm", plot=axes[1, i])
    axes[1, i].set_title(f'Q-Q plot for {column}')
plt.show()
```

Fig 20: Code creating the Q-Q plots and histograms using the probplot.

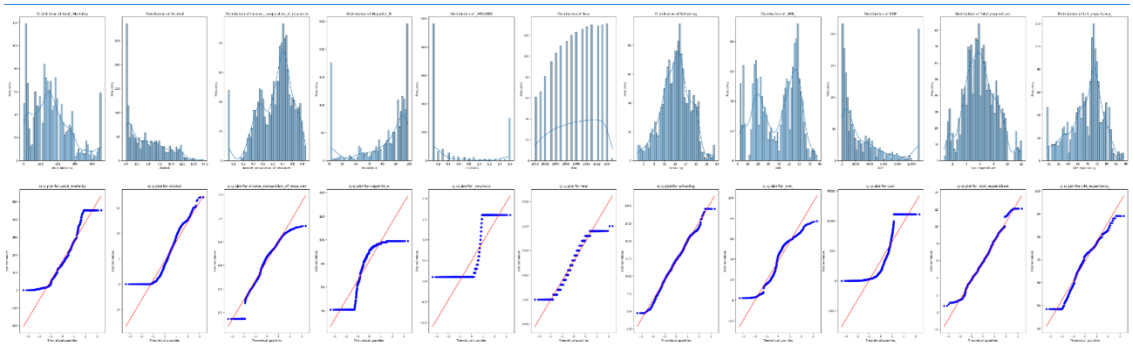

Fig 21: Q-Q plots and histograms showing the data distribution of the selected variables.

As our selected variables are not normally distributed, we have chosen the non-parametric test Mann Whitney U test to check whether the distribution of data is similar in the two groups Developed and Developing (this is based on the 'status' column which has a categorical data of developed and developing).

5.4 Performing Non-parametric test: Mann Whitney U test.

```
# Select 'Developed' and 'Developing' groups
group_developed = selected_data[selected_data['Status'] == 'Developed']
group_developing = selected_data[selected_data['Status'] == 'Developing']

# Perform Mann-Whitney U and Kruskal-Wallis tests for all numerical features
alpha = 0.05

for feature in selected_data.select_dtypes(include='number').columns:
    # Perform Mann-Whitney U test
    mw_statistic, mw_p_value = mannwhitneyu(group_developed[feature], group_developing[feature])

    # Null Hypothesis (H0): The distribution of the variable is the same in both groups.
    # Alternative Hypothesis (H1): The distribution of the variable is different between the groups.

    print(f'{feature}:\n'
          f'Mann-Whitney U Test - Statistic: {mw_statistic}, P-value: {mw_p_value}\n'
          f'Null Hypothesis: The distribution of {feature} is the same in both groups.\n'
          f'Alternative Hypothesis: The distribution of {feature} is different between the groups.\n')

    if mw_p_value < alpha:
        print(f'Mann-Whitney U test: There is a significant difference in {feature} between the two groups.\n')
    else:
        print(f'Mann-Whitney U test: There is insufficient evidence to suggest a significant difference in {feature} between groups.\n')
```

Fig 22: Code that performs the Mann Whitney U test

```
Adult_Mortality:
Mann-Whitney U Test - Statistic: 81481.0, P-value: 1.7334554934005212e-38
Null Hypothesis: The distribution of Adult_Mortality is the same in both groups.
Alternative Hypothesis: The distribution of Adult_Mortality is different between the groups.

Mann-Whitney U test: There is a significant difference in Adult_Mortality between the two groups.

Alcohol:
Mann-Whitney U Test - Statistic: 319261.5, P-value: 2.698977900246903e-105
Null Hypothesis: The distribution of Alcohol is the same in both groups.
Alternative Hypothesis: The distribution of Alcohol is different between the groups.

Mann-Whitney U test: There is a significant difference in Alcohol between the two groups.

Income_composition_of_resources:
Mann-Whitney U Test - Statistic: 323714.5, P-value: 2.0375127740547505e-111
Null Hypothesis: The distribution of Income_composition_of_resources is the same in both groups.
Alternative Hypothesis: The distribution of Income_composition_of_resources is different between the groups.

Mann-Whitney U test: There is a significant difference in Income_composition_of_resources between the two groups.

Hepatitis_B:
Mann-Whitney U Test - Statistic: 216305.0, P-value: 1.5641204682132877e-11
Null Hypothesis: The distribution of Hepatitis_B is the same in both groups.
Alternative Hypothesis: The distribution of Hepatitis_B is different between the groups.

Mann-Whitney U test: There is a significant difference in Hepatitis_B between the two groups.

_HIV/AIDS:
Mann-Whitney U Test - Statistic: 87362.0, P-value: 4.335585758162759e-42
Null Hypothesis: The distribution of _HIV/AIDS is the same in both groups.
Alternative Hypothesis: The distribution of _HIV/AIDS is different between the groups.

Mann-Whitney U test: There is a significant difference in _HIV/AIDS between the two groups.

Year:
Mann-Whitney U Test - Statistic: 161880.5, P-value: 0.22030165545289282
Null Hypothesis: The distribution of Year is the same in both groups.
Alternative Hypothesis: The distribution of Year is different between the groups.

Mann-Whitney U test: There is insufficient evidence to suggest a significant difference in Year between groups.

Schooling:
Mann-Whitney U Test - Statistic: 312308.0, P-value: 9.249653532933963e-96
Null Hypothesis: The distribution of Schooling is the same in both groups.
Alternative Hypothesis: The distribution of Schooling is different between the groups.

Mann-Whitney U test: There is a significant difference in Schooling between the two groups.

_BMI_:
Mann-Whitney U Test - Statistic: 265634.0, P-value: 3.587087919464716e-44
Null Hypothesis: The distribution of _BMI_ is the same in both groups.
Alternative Hypothesis: The distribution of _BMI_ is different between the groups.

Mann-Whitney U test: There is a significant difference in _BMI_ between the two groups.

GDP:
Mann-Whitney U Test - Statistic: 277195.0, P-value: 3.5427831987129405e-55
Null Hypothesis: The distribution of GDP is the same in both groups.
Alternative Hypothesis: The distribution of GDP is different between the groups.

Mann-Whitney U test: There is a significant difference in GDP between the two groups.

Total_expenditure:
Mann-Whitney U Test - Statistic: 232778.0, P-value: 6.317621064970861e-20
Null Hypothesis: The distribution of Total_expenditure is the same in both groups.
Alternative Hypothesis: The distribution of Total_expenditure is different between the groups.

Mann-Whitney U test: There is a significant difference in Total_expenditure between the two groups.

Life_expectancy_:
Mann-Whitney U Test - Statistic: 304438.5, P-value: 1.2140927560691947e-85
Null Hypothesis: The distribution of Life_expectancy_ is the same in both groups.
Alternative Hypothesis: The distribution of Life_expectancy_ is different between the groups.

Mann-Whitney U test: There is a significant difference in Life_expectancy_ between the two groups.
```

Fig 23: The Output indicates that there are significant differences in the distribution of datapoints of the selected variables between the two groups.

5.5 Kendall Tau's correlation.

Step1: Performing the Kendall Tau's correlation.

```python
from scipy.stats import kendalltau

# Select numerical columns
numerical = ['Adult_Mortality', 'Alcohol', 'Income_composition_of_resources',
             'Hepatitis_B', '_HIV/AIDS', 'Year', 'Schooling', '_BMI_', 'GDP',
             'Total_expenditure', 'Life_expectancy_']

# Perform Kendall correlation between each numerical variable and life expectancy
for feature in numerical:
    if feature != 'Life_expectancy_':
        correlation, p_value = kendalltau(selected_data[feature], selected_data['Life_expectancy_'])

        print(f'Correlation between {feature} and Life expectancy:\n'
              f'Kendall Tau: {correlation}\nP-value: {p_value}')

        alpha = 0.05
        if p_value < alpha:
            print(f'Significant correlation between {feature} and Life expectancy \n')
        else:
            print(f'No significant correlation between {feature} and Life expectancy\n')
```

Fig 24: Code that runs the Kendall Tau's Correlation

```
Correlation between Adult_Mortality and Life expectancy:
Kendall Tau: -0.554014515195799
P-value: 1.055095379553892e-246
Significant correlation between Adult_Mortality and Life expectancy

Correlation between Alcohol and Life expectancy:
Kendall Tau: 0.3110626506250147
P-value: 1.339686174493585e-78
Significant correlation between Alcohol and Life expectancy

Correlation between Income_composition_of_resources and Life expectancy:
Kendall Tau: 0.6655397014824229
P-value: 0.0
Significant correlation between Income_composition_of_resources and Life expectancy

Correlation between Hepatitis_B and Life expectancy:
Kendall Tau: 0.19951058393504498
P-value: 2.2619688866766142e-32
Significant correlation between Hepatitis_B and Life expectancy

Correlation between _HIV/AIDS and Life expectancy:
Kendall Tau: -0.5806351510871092
P-value: 1.0180601983118943e-212
Significant correlation between _HIV/AIDS and Life expectancy

Correlation between Year and Life expectancy:
Kendall Tau: 0.04225176171295366
P-value: 0.01310487390525222
Significant correlation between Year and Life expectancy

Correlation between Schooling and Life expectancy:
Kendall Tau: 0.5776991363431078
P-value: 1.5500277489375706e-266
Significant correlation between Schooling and Life expectancy

Correlation between _BMI_ and Life expectancy:
Kendall Tau: 0.42953245983770755
P-value: 1.563183332160498e-149
Significant correlation between _BMI_ and Life expectancy

Correlation between GDP and Life expectancy:
Kendall Tau: 0.40677222936619317
P-value: 9.500029386384395e-133
Significant correlation between GDP and Life expectancy

Correlation between Total_expenditure and Life expectancy:
Kendall Tau: 0.19848764419233939
P-value: 2.2279448229297584e-33
Significant correlation between Total_expenditure and Life expectancy
```

Fig 25: Output of the Kendall Tau's correlation

Based on the Kendall Tau's correlation co-efficient Adult Mortality, HIV/AIDS have the strong negative correlation. Income composition of resources, Schooling has the strong positive correlation. Alcohol, BMI, and GDP has the moderate positive correlation. Year, Total expenditure, Hepatitis-B have the weak positive correlation.

Step2: Correlation Matrix:

To check the correlation among the attributes, we plotted the heatmap for the "Lif_Expectency_Data.csv". We found that the Adult Mortality, HIV/AIDS have the strong negative correlation. Income composition of resources, Schooling has the strong positive correlation. Alcohol, BMI, and GDP has the moderate positive correlation. Year, Total expenditure, Hepatitis-B have the weak positive correlation.

```python
numeric_cols = ['Adult_Mortality', 'Alcohol', 'Income_composition_of_resources',
                'Hepatitis_B', '_HIV/AIDS', 'Year', 'Schooling', '_BMI_',
                'GDP', 'Total_expenditure', 'Life_expectancy_']
correlation_matrix = selected_data[numeric_cols].corr()
# Create heatmap
plt.figure(figsize=(12, 8))
sns.heatmap(correlation_matrix, annot=True, cmap="tab20", fmt=".2f")
plt.show()
```

Fig 26: Code that creates the heatmap for the selected variables and the target variable.

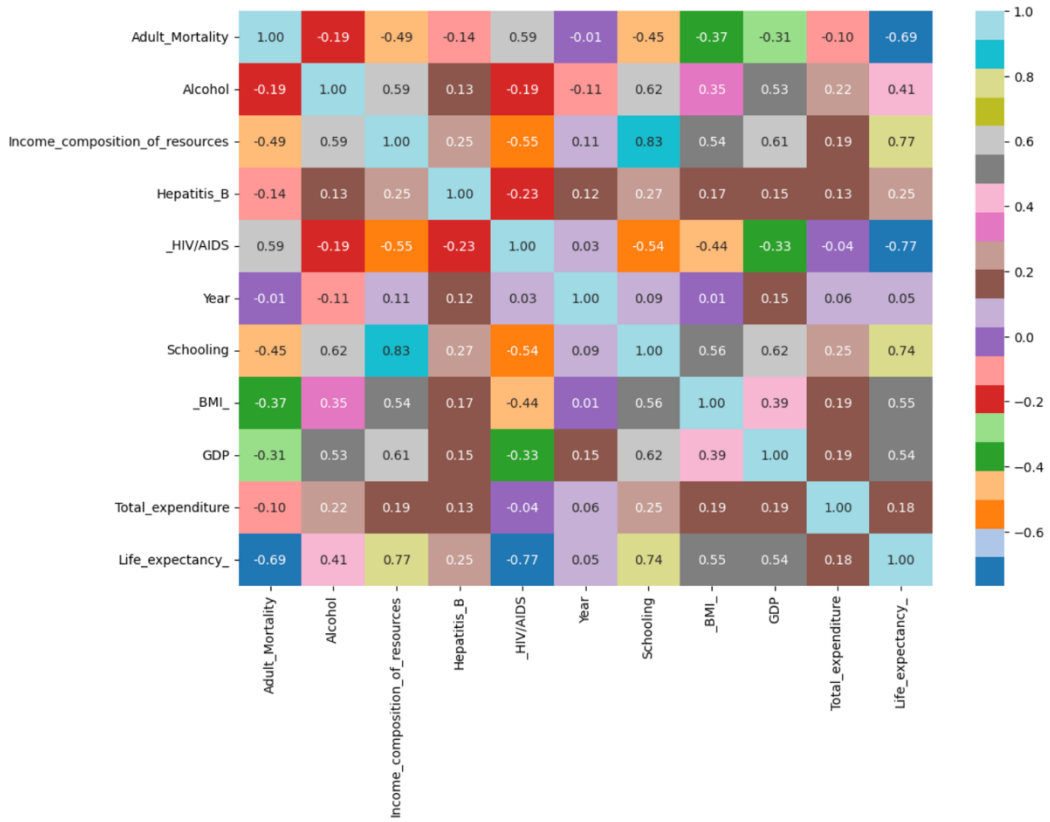

Fig 27: Heatmap showing the correlation between the selected variables and life expectancy.

After interpreting the Heatmap we found that all the selected variables are correlated with the life expectancy. The correlations of the selected variables with life expectancy from strongest to smallest are Income_composition_of_resources, Schooling, BMI, GDP, Alcohol, Hepatitis_B, Total_expenditure. Negative correlations are with HIV/AIDS, Adult_Mortality.

6. Data Visualization

Data visualization in machine learning refers to the graphical representation of data to extract insights, patterns, and relationships that aid in understanding and analyzing datasets used in machine learning tasks. It involves using various visual elements such as charts, graphs, plots, and diagrams to represent complex data in an easily understandable format.

6.1 Data Preparation

6.1a Data Collection: We first gathered datasets containing life expectancy data along with factors like GDP, education, healthcare spending, environmental factors, etc., for different countries.

6.1b Data Cleaning: We preprocessed the data to handle missing values, normalize or scale features, and ensure consistency across datasets.

6.2 Visualization Techniques:

Pie Chart
In this section, we present visualizations that provide insights into the distribution of countries based on their development status.

We employed a pie chart to illustrate the distribution of countries based on their development status.

```
[31]: plt.figure(figsize=(6, 6))
      selected_data['Status'].value_counts().plot(kind='pie', autopct='%1.1f%%', colors=['skyblue', 'red'])
      plt.title('Distribution of Development Status')
      plt.ylabel('')
      plt.show()
```

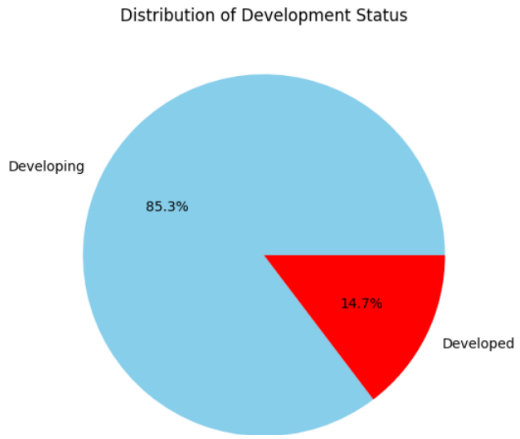

Fig 28: Pie Chart illustrating distribution of development status

The pie chart above illustrates the distribution of countries by development status. The majority of the selected countries fall under the 'Developing' category, constituting 85.3% of the dataset. Meanwhile, 'Developed' countries make up the remaining 14.7%. This visual representation provides a quick overview of the dataset's composition in terms of development status.

Pairplot of Select Features

In this section, we present a pairplot to explore the relationships and distributions between select features in the dataset.

We created a pairplot to visualize the pairwise relationships between the chosen features.

```
# Pairplot of Select Features
pairplot = selected_data[['Adult_Mortality', 'Alcohol', 'Income_composition_of_resources',
       'Hepatitis_B', 'Country', '_HIV/AIDS', 'Year', 'Schooling', '_BMI_',
       'GDP', 'Status', 'Total_expenditure', 'Life_expectancy_']]

plot_kws = {'color': 'lightcoral'}

sns.pairplot(pairplot, plot_kws=plot_kws)
plt.suptitle('Pairplot of Select Features', y=1.02)
plt.show()
```

Fig 29: Code for visualizing a pairplot

Fig 30: Pairplot of select features

The pairplot above provides a visual representation of the relationships and distributions between select features in the dataset. Each subplot in the matrix represents the relationship between two variables, and the diagonal shows the distribution of each individual variable.

Scatter Plots of Selected Independent Variables

In this section, we present scatter plots to visually explore the relationships between selected independent variables and life expectancy.

We selected specific independent variables to investigate their potential impact on life expectancy. The scatter plots below illustrate the relationships between each independent variable and life expectancy.

```
# Select independent variables
independent_vars = ['Adult_Mortality', 'Alcohol', 'Income_composition_of_resources', 'Hepatitis_B', '_HIV/AIDS', 'Year', 'Schooling', '_BMI_', 'GDP',

# Calculate the number of rows and columns for subplots
num_cols = 2
num_rows = (len(independent_vars) + 1) // num_cols

# Create subplots
fig, axes = plt.subplots(nrows=num_rows, ncols=num_cols, figsize=(10, 2 * num_rows))
fig.subplots_adjust(hspace=0.5)

# Iterate through independent variables and create scatter plots
for i, var in enumerate(independent_vars):
    row = i // num_cols
    col = i % num_cols
    axes[row, col].scatter(selected_data[var], selected_data['Life_expectancy_'], color='gold')
    axes[row, col].set_title(f'{var} vs Life Expectancy')
    axes[row, col].set_xlabel(var)
    axes[row, col].set_ylabel('Life Expectancy')
    axes[row, col].grid(True)

# Remove any empty subplots if the number of plots is odd
if len(independent_vars) % num_cols != 0:
    for ax in axes.flatten()[len(independent_vars):]:
        fig.delaxes(ax)

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

Fig 31: Code to visualize scatter plot

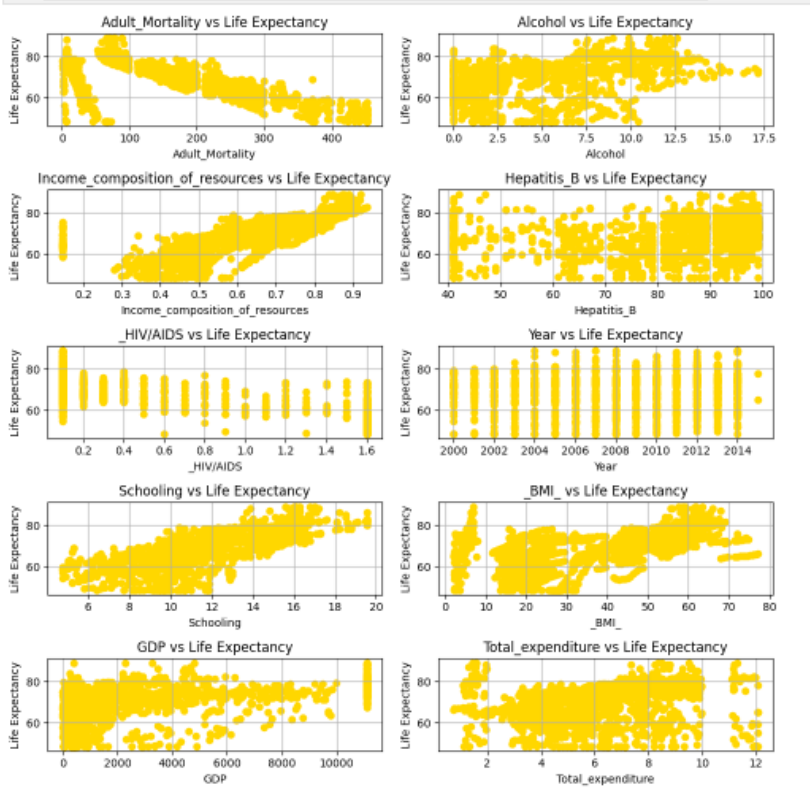

Fig 32: Scatter plot showing relationship of life expectancy with different independent variables

The scatter plots above display the relationships between selected independent variables and life expectancy. Each plot represents a specific variable, and the x-axis denotes the variable's values, while the y-axis represents life expectancy. The gold-colored points illustrate data points in the scatter plots. These visualizations provide insights into the potential associations between each independent variable and life expectancy.

Scatter Plot with Plotly

In this section, we employ Plotly Express to create an interactive scatter plot exploring the relationship between GDP, Life Expectancy, and Income Composition of Resources.

```
import plotly.express as px

# Assuming df is your DataFrame with the relevant columns
fig = px.scatter(selected_data, x='GDP', y='Life_expectancy_', color='Status', size='Income_composition_of_resources')
fig.show()
```

Fig 33: Scatter Plot with Plotly

The interactive scatter plot above visualizes the relationship between GDP, Life Expectancy, and Income Composition of Resources. Each data point represents a country, with the x-axis denoting GDP, the y-axis representing life expectancy, and color differentiating between developing and developed status. The size of each point corresponds to the income composition of resources. This dynamic visualization provides an engaging exploration of the interplay between these key factors.

Histograms

In this section, we present histograms to illustrate the distribution of selected variables in the dataset.

```
import matplotlib.pyplot as plt

selected_data.hist(bins=20, figsize=(15, 10), color='purple')

plt.show()
```

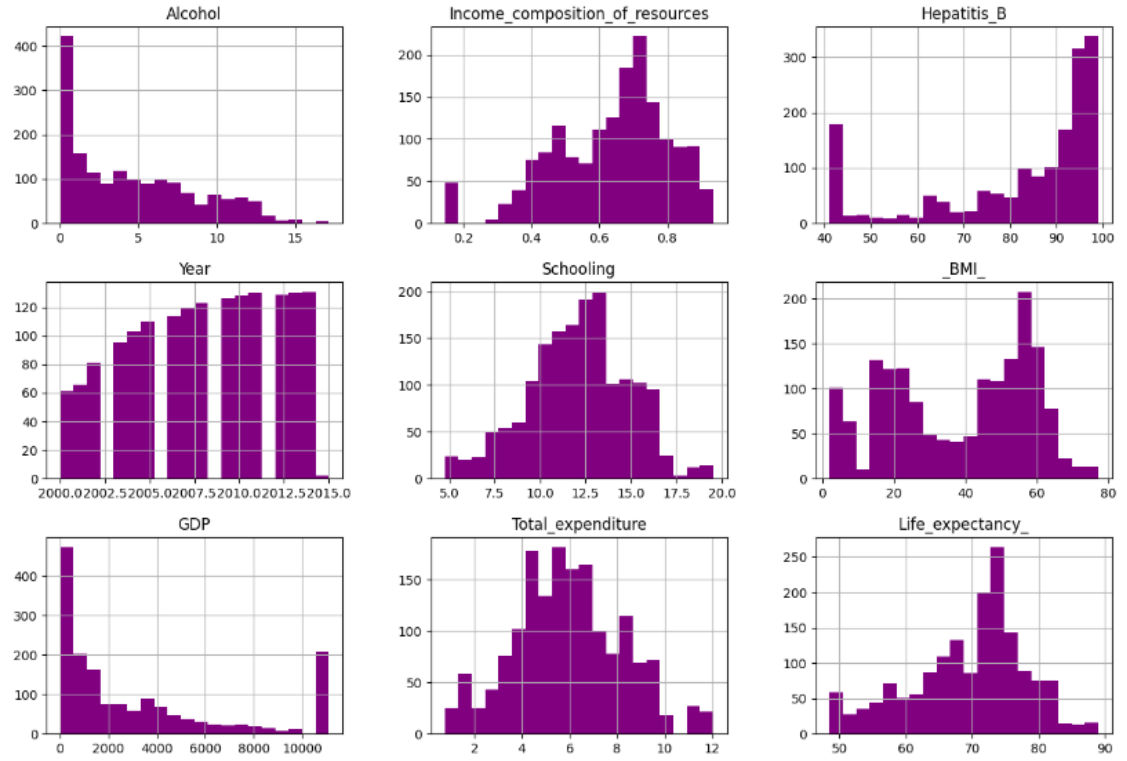

Fig 34: Code and output for Histograms

The histograms above provide a visual representation of the distribution of selected variables in the dataset. Each subplot corresponds to a specific variable, and the histograms reveal insights into the spread and concentration of values. The use of color enhances the visual appeal and aids in the interpretation of the data distribution.

Box Plots by Development Status

In this section, we utilize box plots to visualize the distribution of numerical variables across different development statuses.

```python
# Identify numerical columns
numerical_columns = selected_data.select_dtypes(include='number')

# Set up subplots
num_plots = len(numerical_columns.columns)
num_cols = 2  # You can adjust the number of columns as needed
num_rows = (num_plots // num_cols) + (num_plots % num_cols)

fig, axes = plt.subplots(num_rows, num_cols, figsize=(10, 3 * num_rows))
fig.subplots_adjust(hspace=0.5)

# Iterate through each numerical column
for i, column in enumerate(numerical_columns.columns):
    sns.boxplot(x='Status', y=column, data=selected_data, ax=axes[i // num_cols, i % num_cols])
    axes[i // num_cols, i % num_cols].set_title(f'{column} Box Plot by Status')

# Remove empty subplots if the number of numerical columns is not a multiple of num_cols
if num_plots % num_cols != 0:
    for j in range(num_plots % num_cols, num_cols):
        fig.delaxes(axes[-1, j])

plt.show()
```

Fig 35: Code to visualize box plots

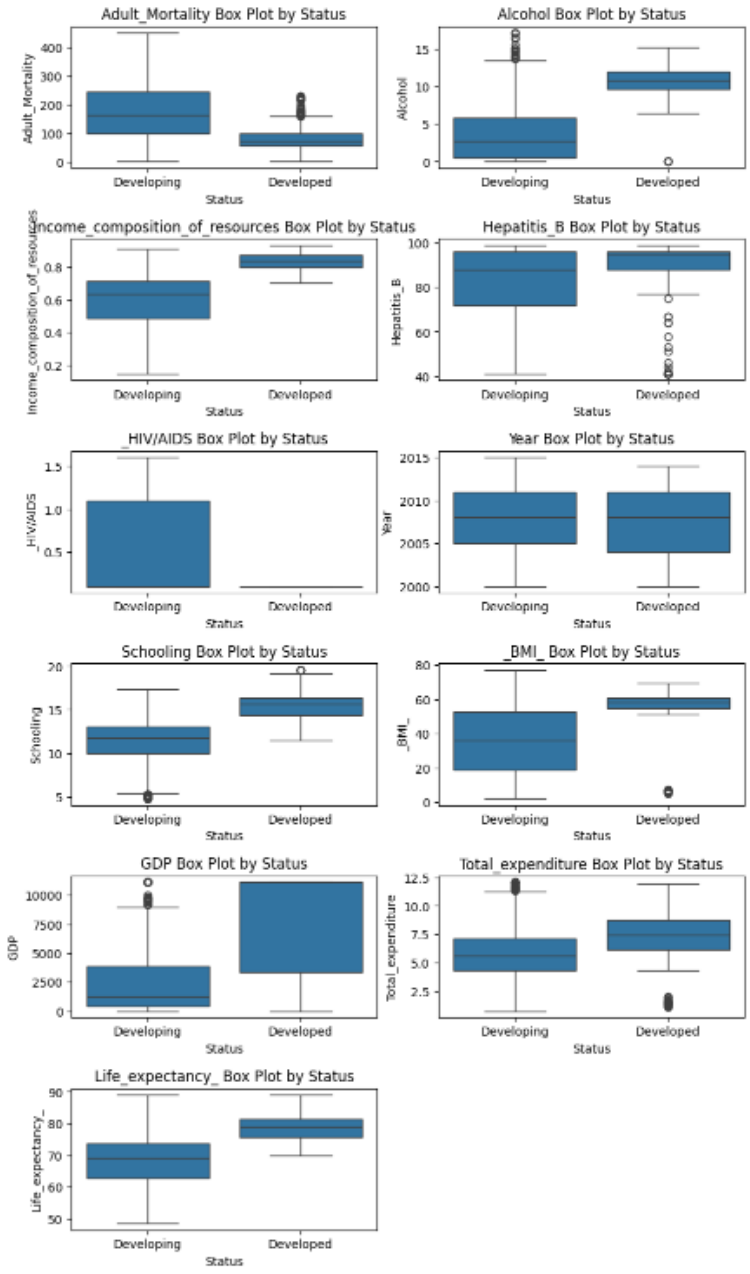

Fig 36: Visualization of box plots

The box plots above showcase the distribution of numerical variables in the dataset categorized by development status. Each subplot corresponds to a specific numerical variable, and the boxes provide insights into the central tendency, spread, and potential outliers. The use of color helps distinguish between developed and developing statuses, facilitating a comparative analysis of the variables across different categories.

Correlation Analysis with Life Expectancy

In this section, we conduct a correlation analysis to explore the relationships between various factors and life expectancy.

```
[93]: numeric_cols = ['Adult_Mortality', 'Alcohol', 'Income_composition_of_resources',
                      'Hepatitis_B', '_HIV/AIDS', 'Year', 'Schooling', '_BMI_',
                      'GDP', 'Total_expenditure', 'Life_expectancy_']

life_expectancy_corr = selected_data[numeric_cols].corr()['Life_expectancy_'].sort_values(ascending=False)

sns.barplot(x=life_expectancy_corr, y=life_expectancy_corr.index, color='orange')
plt.title('Correlation Analysis with Life Expectancy')

[93]: Text(0.5, 1.0, 'Correlation Analysis with Life Expectancy')
```

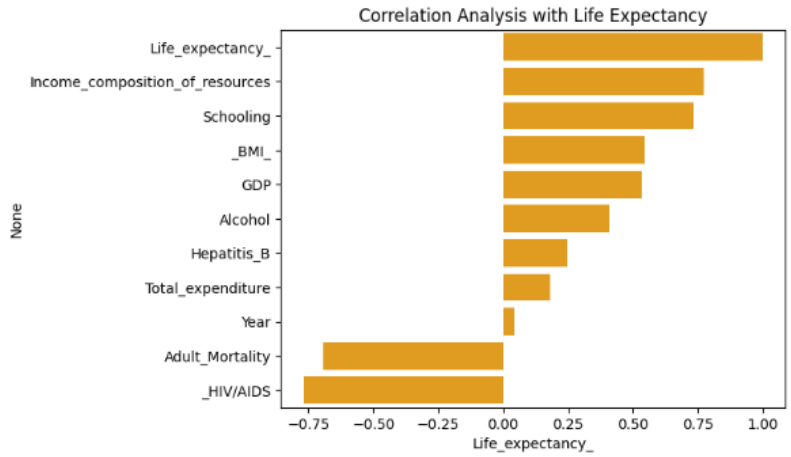

Fig 37: Code and Output for Correlation Analysis with Life Expectancy

Distribution of BMI

In this section, we examine the distribution of Body Mass Index (BMI) to gain insights into the health status of the population.

```
#Distribution of BMI:
plt.figure(figsize=(10, 6))
sns.histplot(selected_data['_BMI_'], bins=20, color='pink')
plt.xlabel('BMI')
plt.ylabel('Frequency')
plt.title('Distribution of BMI')
plt.show()
```

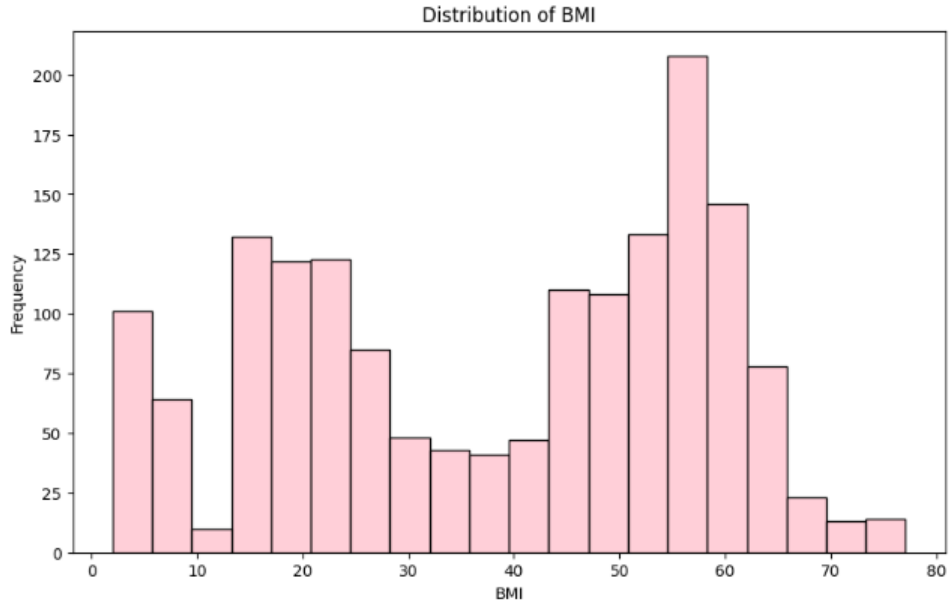

Fig 38: Distribution of BMI

The histogram above illustrates the distribution of Body Mass Index (BMI) within the population. The x-axis represents different BMI ranges, while the y-axis shows the frequency of individuals falling into each range. This visualization provides an overview of the BMI distribution, allowing for the identification of patterns and potential areas of concern related to population health.

7. Data Splitting

In this stage, we performed the essential step of splitting our dataset to facilitate the training and evaluation of our predictive models.

```
# Split the dataset into features (X) and target variable (y)
X = ready_data.drop(target_variable, axis=1)
y = ready_data[target_variable]

# Split the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Print the shapes of the resulting datasets
print("Xtrain shape:", X_train.shape)
print("Xtest shape:", X_test.shape)
print("ytrain shape:", y_train.shape)
print("ytest shape:", y_test.shape)
```

Fig 39: Code for data splitting

First, we divided the dataset into features (X) and the target variable (y). The features encompassed various risk factors, while the target variable represented the outcome we sought to predict.

Next, we proceeded to split the data into training and testing sets, utilizing the train_test_split function. This division enables us to train our models on one subset and evaluate their performance on another, ensuring a robust assessment.

We concluded this step by printing the shapes of the resulting datasets, providing insights into the size of our training and testing sets.

The shapes of our datasets are as follows:

```
Xtrain shape: (1319, 143)
Xtest shape: (330, 143)
ytrain shape: (1319,)
ytest shape: (330,)
```

Fig 40: Data Splitting Result

This step ensures that our models are trained on a sufficiently large dataset while maintaining a separate subset for evaluation, a crucial practice in machine learning model development.

7.1 Feature Scaling:

Following the data splitting stage, we proceeded with the crucial step of feature scaling to ensure uniformity and comparability across our dataset.

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

Fig 41: Code for Feature Scaling

We employed the StandardScaler to standardize our feature values, transforming them into a common scale. This is vital for machine learning models that rely on distance metrics or gradient-based optimization, as it enhances their stability and convergence.

The resulting scaled training dataset (X_train_scaled) is an array containing standardized feature values:

```
array([[-0.70940455, -0.45310766,  0.504294  , ..., -0.09977003,
        -0.08288684, -0.10357591],
       [ 0.81364778,  0.83380458, -0.83643766, ..., -0.09977003,
        -0.08288684, -0.10357591],
       [-0.05525728,  0.37005242,  0.61155254, ..., -0.09977003,
        -0.08288684, -0.10357591],
       ...,
       [ 1.94668022,  1.11785278,  0.71881107, ..., -0.09977003,
        -0.08288684, -0.10357591],
       [ 0.34710501,  0.39324003,  0.87969887, ..., -0.09977003,
        -0.08288684, -0.10357591],
       [-1.10436139, -0.64440542,  0.02163061, ..., -0.09977003,
        -0.08288684, -0.10357591]])
```

Fig 42: Feature Scaling Result

This standardized representation ensures that all features contribute uniformly to the model training process, promoting better convergence and model performance during subsequent stages.

8. Machine Learning Models

8.1 Linear Regression Model

In this phase, we delved into building a Linear Regression model to predict the outcome based on our features. Here's a breakdown of the key steps and outcomes:

```
# Linear Regression
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score, mean_absolute_error

# Train a Linear Regression model
lin_reg = LinearRegression()
lin_reg.fit(X_train, y_train)

# Make predictions on test data
y_pred = lin_reg.predict(X_test)

# Evaluate
rmse = mean_squared_error(y_test, y_pred, squared=False)
print(f'RMSE: {rmse}')
r_squared = r2_score(y_test, y_pred)
print(f'R-squared: {r_squared}')
mae = mean_absolute_error(y_test, y_pred)
print(f'MAE: {mae}')
```

Fig 43: Code for Linear Regression Model

Results:

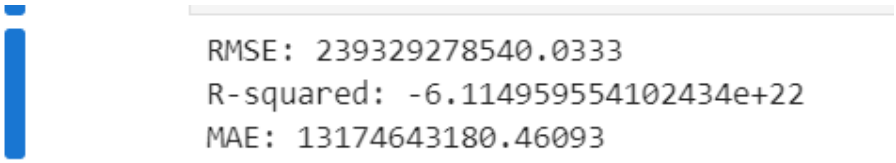

Fig 44: Linear Regression Result

The RMSE represents the square root of the average squared differences between the predicted and actual values. In our case, the high RMSE suggests significant variance between predicted and actual values.

The R-squared value indicates the proportion of the variance in the target variable explained by the model. The extremely low R-squared value suggests that the linear regression model may not be well-suited for our data or that the features may not have a linear relationship with the target.

The Mean Absolute Error (MAE) represents the average absolute differences between the predicted and actual values. The high MAE indicates substantial deviations between predicted and actual values.

8.2 Ridge Regression Model

In this stage, we implemented a Ridge Regression model, a regularization technique to improve model performance. Here's an overview of the steps and outcomes:

```python
from sklearn.linear_model import Ridge
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import mean_squared_error, r2_score

# Define the alpha values to search
alphas = [0.1, 1.0, 10.0]

# Create Ridge model
ridge_model = Ridge()

# Set up the parameter grid
param_grid = {'alpha': alphas}

# Use GridSearchCV for hyperparameter tuning
grid_search = GridSearchCV(ridge_model, param_grid, cv=5, scoring='neg_mean_squared_error')
grid_search.fit(X_train_scaled, y_train)

# Get the best hyperparameters
best_alpha = grid_search.best_params_['alpha']

# Predictions on the testing set using the best model
y_pred_ridge = grid_search.predict(X_test_scaled)

# Evaluate performance for Ridge
rmse_ridge = mean_squared_error(y_test, y_pred_ridge, squared=False)
r2_ridge = r2_score(y_test, y_pred_ridge)

# Print the best hyperparameter and evaluation metrics
print(f"Best Alpha: {best_alpha}")
print("Ridge Regression:")
print(f"Root Mean Squared Error: {rmse_ridge}")
print(f"R-squared: {r2_ridge}")
```

Fig 45: Code for Ridge Regression Model

Results:

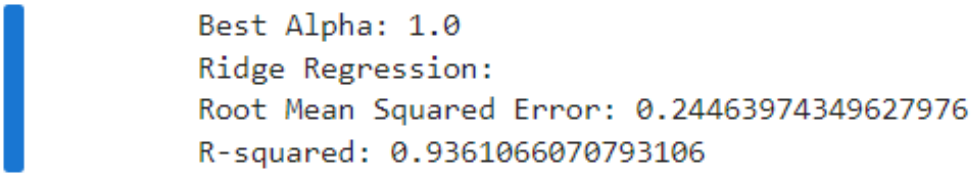

Fig 46: Ridge Regression Result

The Ridge Regression model, with hyperparameter tuning, demonstrates improved performance compared to the basic Linear Regression model. The optimal alpha value of 1.0 indicates the regularization strength applied to prevent overfitting.

The Root Mean Squared Error (RMSE) of 0.245 and the high R-squared value of 0.936 suggest that the Ridge Regression model provides a more accurate and stable prediction compared to the basic Linear Regression model. However, further analysis and comparison with other models may be conducted to enhance predictive capabilities.

8.3 Decision Tree Model

So, in this phase, we implemented a Decision Tree model for predicting outcomes based on the features. Here's a summary of the key steps and outcomes:

```
# Decision Tree
decision_tree_model = DecisionTreeRegressor(random_state=42)
decision_tree_model.fit(X_train, y_train)
y_pred_tree = decision_tree_model.predict(X_test)

# Evaluate the Decision Tree model
rmse_tree = mean_squared_error(y_test, y_pred_tree, squared=False)
r2_tree = r2_score(y_test, y_pred_tree)

print(f"Decision Tree - Root Mean Squared Error: {rmse_tree}, R-squared: {r2_tree}")
```

```
Decision Tree - Root Mean Squared Error: 0.391003708743411, R-squared: 0.8367836433744034
```

Fig 47: Code for decision tree with result

Results:

The Decision Tree model demonstrates a Root Mean Squared Error of 0.391 and an R-squared value of 0.837, suggesting good predictive performance. Decision Trees are known for capturing non-linear relationships in data, and these results indicate their effectiveness in this context.

8.4 Random Forest Model

In continuation, we extended our exploration to a Random Forest model, a more sophisticated ensemble method. Below are the steps and outcomes of this model:

```
# Random Forest
random_forest_model = RandomForestRegressor(random_state=42)
random_forest_model.fit(X_train, y_train)
y_pred_forest = random_forest_model.predict(X_test)
# Evaluate the Random Forest model
rmse_forest = mean_squared_error(y_test, y_pred_forest, squared=False)
r2_forest = r2_score(y_test, y_pred_forest)

print(f"Random Forest - Root Mean Squared Error: {rmse_forest}, R-squared: {r2_forest}")
```

```
Random Forest - Root Mean Squared Error: 0.2609216566655097, R-squared: 0.9273187847490747
```

Fig 48: Code for random forest with result

Results:

The Random Forest model outperformed both the Linear Regression and Decision Tree models, showcasing superior predictive capabilities. The lower RMSE and higher R-squared value indicate that the Random Forest model provides more accurate and reliable predictions.

9. Summary of Findings

Our investigation, utilizing the "Health & Development Indicators: Global Insights" dataset, aimed to uncover the key determinants influencing life expectancy worldwide. Through comprehensive data analysis and visualization, we identified crucial factors contributing to disparities in life expectancy between developed and developing nations.

9.1 Hypotheses:

Null Hypothesis: There is no significant linear relationship between any of the independent variables and life expectancy.

Alternate Hypothesis: At least one of the independent variables has a significant linear relationship with life expectancy.

Findings:

Our analysis focused on various factors, including adult mortality, alcohol consumption, income composition of resources, hepatitis B prevalence, country-specific attributes, HIV/AIDS rates, yearly trends, schooling levels, BMI, GDP, socio-economic status, and total expenditure. The results rejected the null hypothesis, indicating that at least one of the independent variables has a significant linear relationship with life expectancy.

9.2 Aim:

Our aim was to identify the factors with the greatest impact on life expectancy in different countries. The correlations between life expectancy and various factors were explored.

Key Findings:

High-Impact Factors

Factors such as Adult Mortality, Income composition of resources, and Schooling demonstrated robust correlations with life expectancy. These variables emerged as pivotal influencers, signifying their significant impact on life expectancy rates.

Moderate Influencers

Variables including Alcohol consumption, Hepatitis B prevalence, GDP, and Total expenditure showcased moderate correlations with life expectancy. While not as pronounced as high-impact factors, these variables still suggest a noteworthy potential to influence life expectancy to a lesser extent.

Negative Correlations

Adult Mortality and HIV/AIDS exhibited substantial negative correlations. This implies that as these factors increase, life expectancy tends to decrease. The presence of such negative correlations emphasizes the critical role of addressing these factors to enhance life expectancy globally.

These insights provide a nuanced understanding of the intricate relationship between specific factors and life expectancy, offering valuable guidance for targeted interventions and policy decisions to improve global public health outcomes.

A Appendix

A.1 Importing necessary libraries:

```
import pandas as pd
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score
import matplotlib.pyplot as plt
from scipy.stats import ttest_ind, f_oneway
from scipy.stats import kruskal, mannwhitneyu
from sklearn.preprocessing import RobustScaler
from sklearn.preprocessing import StandardScaler
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import RandomForestRegressor
from sklearn.svm import SVR
from sklearn.model_selection import GridSearchCV
```

A.2 Reading the dataset:

```
data = pd.read_csv('Life_Expectancy_Data.csv' )
```

A.3 Checking for and dropping the null values:

```
print(data.isnull().sum())
data = data.dropna()
```

A.4 To estimate the descriptive statistics of this dataset:

```
data.describe()
```

A.5 Defining the selected and target variables:

```
selected_features = ['Adult_Mortality', 'Alcohol', 'Income_composition_of_resources',
                'Hepatitis_B', 'Country' , '_HIV/AIDS', 'Year', 'Schooling', '_BMI_', 'GDP', 'Status', 'Total_expenditure']
target_variable = 'Life_expectancy_'
```

A.6 Filtering the DataFrame to include only selected features and the target variable:

```
selected_data = data[selected_features + [target_variable]]
```

A.7 Detecting Outliers:

```
numeric_columns = selected_data.select_dtypes(include=['int64', 'float64'])
Q1 = numeric_columns.quantile(0.25)
Q3 = numeric_columns.quantile(0.75)
IQR = Q3 - Q1
lower_bound = Q1 - 1.5 * IQR
upper_bound = Q3 + 1.5 * IQR
outliers_count = ((numeric_columns < lower_bound) | (numeric_columns > upper_bound)).sum()
print("Number of outliers in each numeric column:")
print(outliers_count)
```

A.8 Visualizing outliers using boxplots:

```
numerical_columns = selected_data.select_dtypes(include='number')
num_rows = len(numerical_columns) // 2 + len(numerical_columns) % 2
num_cols = 2
fig, axes = plt.subplots(nrows=len(numerical_columns.columns) // 2 + len(numerical_columns.columns) % 2,
ncols=2, figsize=(10, 3 * (len(numerical_columns.columns) // 2 + len(numerical_columns.columns) % 2)))
fig.subplots_adjust(hspace=0.5)
for i, column in enumerate(numerical_columns.columns):
   # Box plot for numerical variables with outliers
   sns.boxplot(x=column, data=numerical_columns, ax=axes[i // 2, i % 2])
   axes[i // 2, i % 2].set_title(f'{column} Box Plot with Outliers')
if len(numerical_columns.columns) % 2 != 0:
   fig.delaxes(axes[-1, -1])
plt.show()
```

A.8 Capping Outliers:

```
Q1 = numerical_columns.quantile(0.25)
Q3 = numerical_columns.quantile(0.75)
IQR = Q3 - Q1
lower_bound = Q1 - 1.5 * IQR
upper_bound = Q3 + 1.5 * IQR
for column in numerical_columns.columns:
   selected_data.loc[selected_data[column] < lower_bound[column], column] = lower_bound[column]
```

```
    selected_data.loc[selected_data[column] > upper_bound[column], column] = upper_bound[column]
```

A.9 Display mean, median, and standard deviation:

```
numeric_columns = selected_data.select_dtypes(include=['number'])
summary_stats = numeric_columns.describe()
print("Mean:")
print(summary_stats.loc['mean'])
print("\nMedian:")
print(numeric_columns.median())
print("\nStandard Deviation:")
print(summary_stats.loc['std'])
```

A.10 Visualizing normality distribution via histograms and Q-Q plots:

```
from scipy.stats import probplot
numerical_features = selected_data[numerical_columns]
fig, axes = plt.subplots(nrows=2, ncols=len(numerical_features.columns), figsize=(70, 20))
for i, column in enumerate(numerical_features.columns):
    sns.histplot(data=numerical_features, x=column, bins=50, kde=True, ax=axes[0, i])
    axes[0, i].set_title(f'Distribution of {column}')
    axes[0, i].set_ylabel('Frequency')
for i, column in enumerate(numerical_features.columns):
    probplot(numerical_features[column], dist="norm", plot=axes[1, i])
    axes[1, i].set_title(f'Q-Q plot for {column}')
plt.show()
```

A.11 Performing Shapiro wilk testing for normality:

```
from scipy.stats import shapiro
numerical_features = selected_data[numerical_columns]
for column in numerical_columns:
    stat, p_value = shapiro(numerical_features[column])
    alpha = 0.05
    print(f'{column}:')
    print(f'Statistic: {stat}, p-value: {p_value}')
    if p_value > alpha:
        print('Result: Data looks normally distributed (fail to reject H0)\n')
    else:
        print('Result: Data does not look normally distributed (reject H0)\n')
```

A.12 Performing non-parametric testing (Mann-Whitney U and Kruskal Wallis):

```
group_developed = selected_data[selected_data['Status'] == 'Developed']
group_developing = selected_data[selected_data['Status'] == 'Developing']
alpha = 0.05
for feature in selected_data.select_dtypes(include='number').columns:
    mw_statistic, mw_p_value = mannwhitneyu(group_developed[feature], group_developing[feature])
    print(f'{feature}:\n'
          f'Mann-Whitney U Test - Statistic: {mw_statistic}, P-value: {mw_p_value}\n'
          f'Null Hypothesis: The distribution of {feature} is the same in both groups.\n'
          f'Alternative Hypothesis: The distribution of {feature} is different between the groups.\n')
    if mw_p_value < alpha:
        print(f'Mann-Whitney U test: There is a significant difference in {feature} between the two groups.\n')
    else:
        print(f'Mann-Whitney U test: There is insufficient evidence to suggest a significant difference in {feature}
between groups.\n')
```

```
    kw_statistic, kw_p_value = kruskal(group_developed[feature], group_developing[feature])
    print(f'Kruskal-Wallis Test - Statistic: {kw_statistic}, P-value: {kw_p_value}\n'
          f'Null Hypothesis: The distribution of {feature} is the same in both groups.\n'
          f'Alternative Hypothesis: The distribution of {feature} is different between the groups.\n')
    if kw_p_value < alpha:
        print(f'Kruskal-Wallis test: There are significant differences in {feature} between the two groups.\n')
    else:
        print(f'Kruskal-Wallis test: There is insufficient evidence to suggest significant differences in {feature} between
groups.\n')
```

A.13 Performing Correlation for hypothesis testing (Kendell Tau):

```
from scipy.stats import kendalltau
numerical = ['Adult_Mortality', 'Alcohol', 'Income_composition_of_resources',
             'Hepatitis_B', '_HIV/AIDS', 'Year', 'Schooling', '_BMI_', 'GDP',
             'Total_expenditure', 'Life_expectancy_']
    if feature != 'Life_expectancy_':
        correlation, p_value = kendalltau(selected_data[feature], selected_data['Life_expectancy_'])
        print(f'Correlation between {feature} and Life expectancy:\n'
              f'Kendall Tau: {correlation}\nP-value: {p_value}')
        alpha = 0.05
        if p_value < alpha:
            print(f'Significant correlation between {feature} and Life expectancy \n')
        else:
            print(f'No significant correlation between {feature} and Life expectancy\n')
```

A.14 Linear regression summary:

```
import statsmodels.api as sm
constant_vars = ['Adult_Mortality', 'Alcohol', 'Income_composition_of_resources', 'Hepatitis_B', '_HIV/AIDS',
'Year', 'Schooling', '_BMI_', 'GDP', 'Total_expenditure']
target_variable = 'Life_expectancy_'
selected_data_with_constant = sm.add_constant(selected_data[constant_vars])
model = sm.OLS(selected_data[target_variable], selected_data_with_constant).fit()
print(model.summary())
```

A.15 Pairplot of Select Features:

```
pairplot = selected_data[['Adult_Mortality', 'Alcohol', 'Income_composition_of_resources',
      'Hepatitis_B', 'Country', '_HIV/AIDS', 'Year', 'Schooling', '_BMI_',
      'GDP', 'Status', 'Total_expenditure', 'Life_expectancy_']]
plot_kws = {'color': 'lightcoral'}
sns.pairplot(pairplot, plot_kws=plot_kws)
plt.suptitle('Pairplot of Select Features', y=1.02)
plt.show()
```

A.16 Scatter plots:

```
independent_vars = ['Adult_Mortality', 'Alcohol', 'Income_composition_of_resources', 'Hepatitis_B', '_HIV/AIDS',
'Year', 'Schooling', '_BMI_', 'GDP', 'Total_expenditure']
num_cols = 2
num_rows = (len(independent_vars) + 1) // num_cols
fig, axes = plt.subplots(nrows=num_rows, ncols=num_cols, figsize=(12, 6 * num_rows))
fig.subplots_adjust(hspace=0.5)
for i, var in enumerate(independent_vars):
    row = i // num_cols
    col = i % num_cols
```

```
    axes[row, col].scatter(selected_data[var], selected_data['Life_expectancy_'])
    axes[row, col].set_title(f'{var} vs Life Expectancy')
    axes[row, col].set_xlabel(var)
    axes[row, col].set_ylabel('Life Expectancy')
    axes[row, col].grid(True)
if len(independent_vars) % num_cols != 0:
    for ax in axes.flatten()[len(independent_vars):]:
        fig.delaxes(ax)
plt.tight_layout()
plt.show()
```

A.17 Histograms:

```
import matplotlib.pyplot as plt
selected_data.hist(bins=20, figsize=(15, 10), color='purple')
plt.show()
```

A.18 Correlation analysis with life expectancy:

```
numeric_cols = ['Adult_Mortality', 'Alcohol', 'Income_composition_of_resources',
            'Hepatitis_B', '_HIV/AIDS', 'Year', 'Schooling', '_BMI_',
            'GDP', 'Total_expenditure', 'Life_expectancy_']
life_expectancy_corr = selected_data[numeric_cols].corr()['Life_expectancy_'].sort_values(ascending=False)
sns.barplot(x=life_expectancy_corr, y=life_expectancy_corr.index, color='orange')
plt.title('Correlation Analysis with Life Expectancy')
```

A.19 Heatmap:

```
numeric_cols = ['Adult_Mortality', 'Alcohol', 'Income_composition_of_resources',
            'Hepatitis_B', '_HIV/AIDS', 'Year', 'Schooling', '_BMI_',
            'GDP', 'Total_expenditure', 'Life_expectancy_']
correlation_matrix = selected_data[numeric_cols].corr()
plt.figure(figsize=(12, 8))
sns.heatmap(correlation_matrix, annot=True, cmap="coolwarm", fmt=".2f")
plt.show()
```

A.20 Dropping HIV/AIDS, and adult mortality columns due to strong negative correlation:

```
features_to_drop = [ "Adult_Mortality", "_HIV/AIDS"]
selected_data_copy = selected_data.copy()
selected_data_copy.drop(features_to_drop, axis=1, inplace=True)
selected_data = selected_data_copy
```

A.21 One-hot encode categorical columns:

```
import pandas as pd
categorical_cols = ['Status', 'Country']
one_hot_cols = pd.get_dummies(selected_data[categorical_cols])
numeric_columns = selected_data.select_dtypes(include=['int64', 'float64'])
normalized_numeric_cols = (numeric_columns - numeric_columns.mean()) / numeric_columns.std()
unchanged_cols = [c for c in selected_data.columns if c not in numeric_columns.columns and c not in
categorical_cols]
other_cols = selected_data[unchanged_cols]
ready_data = pd.concat([normalized_numeric_cols, one_hot_cols, other_cols], axis=1)
ready_data
```

A.22 Splitting the data into testing and training datasets:

```
X = ready_data.drop(target_variable, axis=1)
```

```
y = ready_data[target_variable]
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
print("Xtrain shape:", X_train.shape)
print("Xtest shape:", X_test.shape)
print("ytrain shape:", y_train.shape)
print("ytest shape:", y_test.shape)
```

A.23 Linear Regression:

```
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score, mean_absolute_error
lin_reg = LinearRegression()
lin_reg.fit(X_train, y_train)
y_pred = lin_reg.predict(X_test)
rmse = mean_squared_error(y_test, y_pred, squared=False)
print(f'RMSE: {rmse}')
r_squared = r2_score(y_test, y_pred)
print(f'R-squared: {r_squared}')
mae = mean_absolute_error(y_test, y_pred)
print(f'MAE: {mae}')
```

A.24 Ridge regression using GridSearchCV for hyperparameter tuning:

```
from sklearn.linear_model import Ridge
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import mean_squared_error, r2_score
alphas = [0.1, 1.0, 10.0]  # You can adjust the range of alpha values
ridge_model = Ridge()
param_grid = {'alpha': alphas}
grid_search = GridSearchCV(ridge_model, param_grid, cv=5, scoring='neg_mean_squared_error')
grid_search.fit(X_train_scaled, y_train)
best_alpha = grid_search.best_params_['alpha']
y_pred_ridge = grid_search.predict(X_test_scaled)
rmse_ridge = mean_squared_error(y_test, y_pred_ridge, squared=False)  # Calculate RMSE by setting
squared=False
r2_ridge = r2_score(y_test, y_pred_ridge)
print(f"Best Alpha: {best_alpha}")
print("Ridge Regression:")
print(f"Root Mean Squared Error: {rmse_ridge}")
print(f"R-squared: {r2_ridge}")
```

A.25 Decision Tree

```
decision_tree_model = DecisionTreeRegressor(random_state=42)
decision_tree_model.fit(X_train, y_train)
y_pred_tree = decision_tree_model.predict(X_test)
rmse_tree = mean_squared_error(y_test, y_pred_tree, squared=False)
r2_tree = r2_score(y_test, y_pred_tree)
print(f"Decision Tree - Root Mean Squared Error: {rmse_tree}, R-squared: {r2_tree}")
```

A.26 Random Forest

```
random_forest_model = RandomForestRegressor(random_state=42)
random_forest_model.fit(X_train, y_train)
y_pred_forest = random_forest_model.predict(X_test)
rmse_forest = mean_squared_error(y_test, y_pred_forest, squared=False)  # Calculate RMSE by setting
squared=False
```

```
r2_forest = r2_score(y_test, y_pred_forest)
print(f"Random Forest - Root Mean Squared Error: {rmse_forest}, R-squared: {r2_forest}")
```

References

1. Jangir, A. (2023). *Health & development indicators: Global insights* [Data set].

2. Khan, A., Khan, S., & Khan, M. (2016). Factors affecting life expectancy in developed and developing countries of the world: An approach to available literature. *International Journal of Yoga, Physiotherapy and Physical Education, 1(1), 4-6.*