# DATA AND MODELING INSIGHTS FOR LIFE EXPECTANCY PREDICTORS

SOPHIE HARMAN, MICHELLE MUMFORD, KAYSHA O'BRIEN

Abstract. Life expectancy is a critical measure of societal well-being, reflecting health outcomes, socioeconomic development, and quality of life. Our research investigates the question, "What are the most significant factors that influence life expectancy across countries?" We employed linear regression, feature importance analysis, and Principal Component Analysis, to identify key predictors of life expectancy. Our analysis revealed that socioeconomic variables, such as income composition and years of schooling, along with health factors, are strong predictors of life expectancy.

## 1. Research question and overview of the data

Our research seeks to identify key factors influencing life expectancy across countries, focusing on differences between developed and developing nations. Life expectancy reflects health outcomes and quality of life, and understanding its drivers can help allocate resources and guide interventions. Previous studies have successfully employed neural networks to predict life expectancy [1]. Other studies [14] have shown that life expectancy has generally improved worldwide, though significant disparities persist between developed and developing nations despite the overall positive trend. By analyzing trends separately for these groups, we aim to uncover factors that explain this disparity, which neural networks may miss due to their lack of interpretability.

The dataset, provided by the World Health Organization, includes global life expectancy data alongside socioeconomic, health, and demographic variables, covering 182 countries from 2000 to 2015. Although it has missing entries and some out-of-range values, the dataset offers valuable insights once cleaned.

We hypothesize that socioeconomic factors like income and education will be the strongest predictors, followed by health variables such as immunization rates and access to healthcare. By using interpretable machine learning techniques, we aim to quantify these factors and examine their impact on global health outcomes, with a focus on disparities between developed and developing nations to inform interventions.

*Date*: December 10, 2024.

## 2. Data Cleaning / Feature Engineering

In our data cleaning process, we aimed to preserve relevant information while minimizing excessive imputation by dropping countries with substantial missing data, sourcing reliable replacements[1], and using linear interpolation and KNN for imputation. The dataset did not include data for South Sudan, so we dropped the country entirely. Most columns contained float values, requiring only the "Status" column—indicating whether a country is "Developed" or "Developing"—to be converted to a binary format. The population, BMI, under-5 mortality, and infant deaths, columns had large portions of data falling outside reasonable ranges and were entirely replaced with more reliable data. Scattered missing values including GDP were filled using supplementary data from the internet. For any remaining nulls after applying linear interpolation, we referenced data from other years for the same country to fill gaps where possible. When this approach was insufficient, we employed KNN imputation, which identifies similar instances in the dataset based on feature similarity to estimate missing values. These extensive replacements raised questions about the original dataset's quality, akin to the Ship of Theseus paradox.

Using our existing data, we engineered features like population density, population growth, and a health coverage index to enhance the dataset. We transformed our BMI and income features into categorical columns (e.g., underweight, overweight) for clearer trends. It was challenging to find reliable data that aligned with the scope of our dataset, so our new features are primarily derived from readily available sources. To be more specific, we incorporated features from OurWorldInData [13], such as a human rights index, a democracy index, and suicide rates to provide additional contextual information.

## 3. Data Visualization and Basic Analysis

To identify key factors influencing life expectancy, we used a multi-step approach combining data visualization, feature importance analysis, principal component analysis (PCA), and multiple linear regression. Our goal was to identify significant predictors and evaluate model effectiveness. The dataset, including health-related and socioeconomic variables like HIV/AIDS prevalence, income composition, and education, is well-suited for this analysis. We chose multiple linear regression because many features show linear relationships, and the model offers clear interpretability of how individual predictors affect life expectancy, in contrast to more complex models.

We started with exploratory data analysis and feature importance analysis using Random Forest to identify key predictors of life expectancy, excluding

[1]Missing or unreasonable values in "Population," "Under-five Deaths," and "Life Expectancy" were supplemented with data from OurWorldInData. For "GDP", missing values were filled using data from data.worldbank.org, with additional values for Venezuela [16] and North Korea [7]. All sources are reputable and widely recognized.

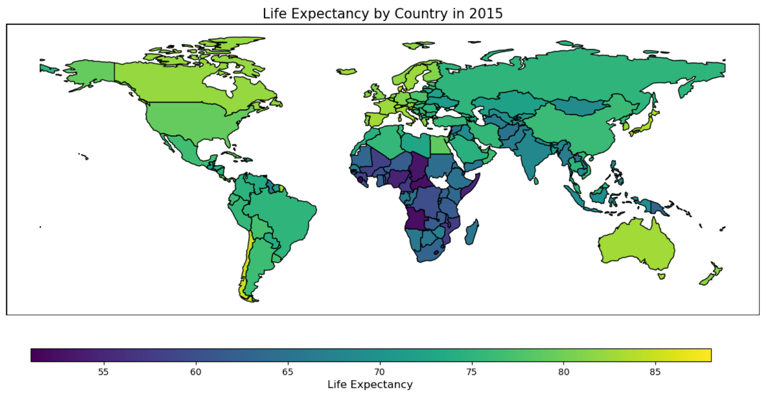
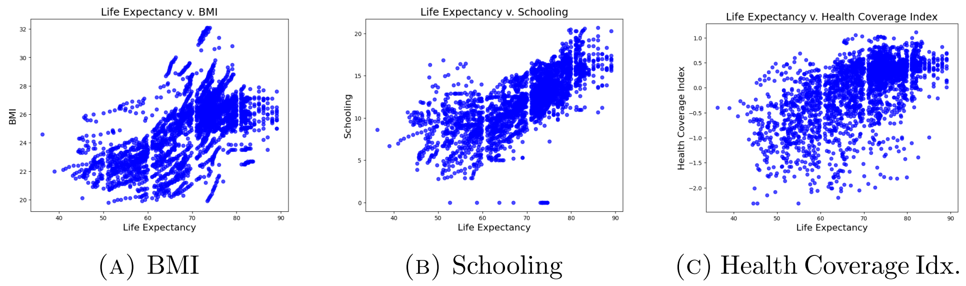


FIGURE 1. Average life expectancy per country in 2015. To observe changes in life expectancy from 2000 to 2015, click here.



(A) BMI (B) Schooling (C) Health Coverage Idx.

FIGURE 2. Several features in our dataset exhibit linear trends, as shown in the following examples above.

highly correlated factors like adult mortality and infant deaths. We applied PCA to reduce dimensionality and capture data variance. Multiple linear regression showed strong positive correlations between income composition, HIV/AIDS, high income, human rights, schooling, and life expectancy. Despite low Variance Inflation Factor (VIF) scores, we detected potential multicollinearity, which we addressed using Ridge Regression to penalize large coefficients and reduce its impact.

## 4. LEARNING ALGORITHMS AND IN-DEPTH ANALYSIS

We applied a multiple linear regression model to three country groups: all countries, developing countries, and developed countries. The overall model had an R-squared of 0.807, indicating a good fit, though the Mean Absolute Error (3.10 years) and Root Mean Squared Error (4.33 years) suggest predictions deviate by several years, likely due to unmeasured variables. Key predictors included HIV/AIDS and BMI (negative impact) and income composition, schooling, and human rights (positive impact; see Table 1).

Feature importance analysis confirmed HIV/AIDS and income composition as critical factors, while variables like agricultural employment had minimal influence.

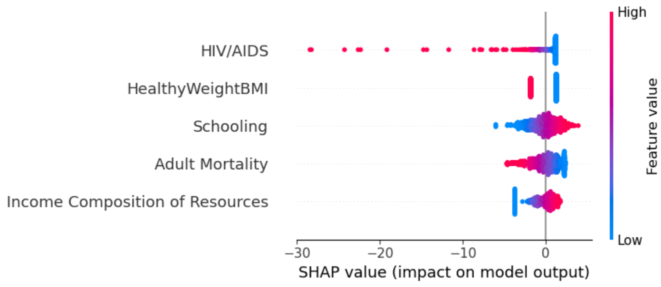
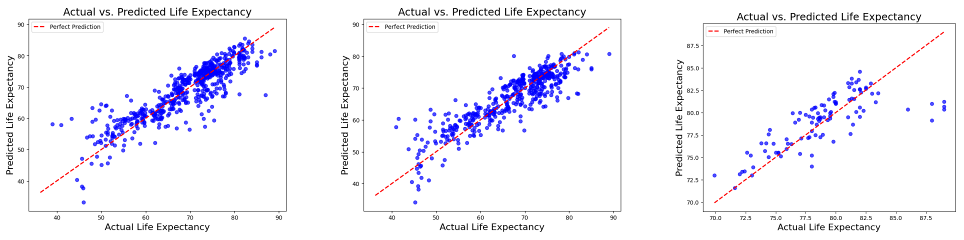


FIGURE 3. We use SHAP analysis, a feature importance technique, to visualize model predictions, where blue indicates factors that lower life expectancy, red indicates factors that increase it, and purple highlights overlapping contributions.

| **Variable** | **All Countries** | **Developing** | **Developed** |
|---|---|---|---|
| Alcohol | 0.0563*** | 0.0243** | -0.1221** |
| Diphtheria Immunizations | 0.0555* | 0.0561* | 0.0226** |
| HIV/AIDS Deaths | -0.6456* | -0.6426* | 4.5072* |
| Income Composition of Resources | 6.4301* | 4.5853* | 21.8626* |
| High Income | 1.7282* | 2.2670* | 0.1993 |
| Healthy Weight BMI | -4.6401* | -5.4095* | 3.0085* |
| Thinness 1-19 Years | -0.0515 | -0.0013 | -1.3058* |
| Human Rights | 2.4229* | 1.8488* | 17.4744** |
| Suicide Rate | -0.1553* | -0.1397* | -0.1162* |
| Schooling | 0.6419* | 0.5793* | -0.0160 |
| $R^2$ | 0.807 | 0.763 | 0.613 |

TABLE 1. Regression Coefficients by Country Type.
To analyze statistical significance, we use the following scale for the p-stat: *$p<0.01$, **$p<0.05$, ***$p<0.1$

For developing countries, percentage expenditure and high income had a stronger impact on life expectancy than in developed countries, where thinness showed a stronger correlation. Notably, the developing countries dataset (2400 rows) is significantly larger than the developed countries dataset (512 rows), which may affect result significance and explain the lower R-squared for developed countries.



(A) All Countries (B) Developing (C) Developed

FIGURE 4. Actual vs. predicted life expectancy by country type.

## 5. Ethical Implications and Conclusions

Having demonstrated our model's capabilities, we address ethical considerations. The WHO data is public and poses no privacy risks, making it a valuable tool for charities and nonprofits in resource allocation. However, using a simple algorithm for aid distribution could have unintended consequences, such as discouraging struggling nations from improving living conditions to maintain aid eligibility or prompting governments to prioritize short-term gains over long-term development. Life expectancy should not be the sole metric; we recommend complementing this model with additional factors like political stability, economic conditions, and conflict to avoid conflating correlation with causation.

Our project identified key factors influencing life expectancy, revealing that HIV/AIDS deaths negatively impact it, while income composition and schooling are strong positive predictors. Early-life mortality is strongly correlated with life expectancy, highlighting the importance of health interventions and education. We found notable differences between developed and developing countries: in developing countries, factors like expenditure and income had more influence, while in developed countries, thinness was more strongly correlated. Linear regression and PCA provided valuable insights, though unexplained variance suggests the need for additional factors. Future work will explore non-linear models and external data to deepen our understanding of life expectancy across different socioeconomic contexts.

## References

[1] Amos, B. K., & Smirnov, I. V. (n.d.). Determinants factors in predicting life expectancy using machine learning. *Advanced Engineering Research (Rostov-on-Don)*, 22(4), 373–383. `https://doi.org/10.23947/2687-1653-2022-22-4-373-383`.

[2] "Child Mortality Rate." Our World in Data, `ourworldindata.org/grapher/child-mortality?time=1986..latest.` Accessed 2 Dec. 2024.

[3] Dattani, Saloni, et al. "Life Expectancy." Our World in Data, 28 Dec. 2023, `ourworldindata.org/life-expectancy`.

[4] Dattani, Saloni, Lucas Rodés-Guirao, Hannah Ritchie, Max Roser, et al. "Suicides." Our World in Data, 22 Feb. 2024, `ourworldindata.org/suicide`.

[5] "Democracy Data Explorer." Our World in Data, `ourworldindata.org/explorers/democracy`. Accessed 2 Dec. 2024.

[6] "GDP per Capita (Constant 2015 US$)." World Bank Open Data, `data.worldbank.org/indicator/NY.GDP.PCAP.KD`. Accessed 2 Dec. 2024.

[7] "GDP per Capita (Current US$) - Korea, Rep." World Bank Open Data, `data.worldbank.org/indicator/NY.GDP.PCAP.CD?locations=KR`. Accessed 2 Dec. 2024.

[8] Hannah Ritchie, Lucas Rodés-Guirao, Edouard Mathieu, Marcel Gerber, Esteban Ortiz-Ospina, Joe Hasell and Max Roser (2023) - "Population Growth" Published online at OurWorldinData.org. Retrieved 25 Nov 2024 from `https://ourworldindata.org/population-growth`.

[9] Herre, Bastian, et al. "Human Rights." Our World in Data, 28 Dec. 2023, `ourworldindata.org/human-rights`.

[10] Herre, Bastian, et al. "Military Personnel and Spending." Our World in Data, 28 Dec. 2023, `ourworldindata.org/military-personnel-spending`.

[11] KumarRajarshi. 2017. "Life Expectancy (WHO)". Retrieved November 25 2024 from `https://www.kaggle.com/datasets/kumarajarshi/life-expectancy-who`.

[12] "North Korea Population 1950-2024." MacroTrends, www.macrotrends.net/global-metrics/countries/PRK/north-korea/population. Accessed 25 Nov. 2024.

[13] Our World in Data, and Max Roser. "Our World in Data." Our World in Data, 25 Mar. 2024, ourworldindata.org/.

[14] Phyo, A.Z.Z., Freak-Poli, R., Craig, H. et al. Quality of life and mortality in the general population: a systematic review and meta-analysis. BMC Public Health 20, 1596 (2020).

[15] Roser, Max. "Employment in Agriculture." Our World in Data, 28 Dec. 2023, `ourworldindata.org/employment-in-agriculture`.

[16] "Venezuela GDP per Capita." Trading Economics, `tradingeconomics.com/venezuela/gdp-percapita#:~:text=The%20Gross%20Domestic%20Product%20per,source:%20World%20Bank`. Accessed 2 Dec. 2024.

[17] "Who Immunization Data Portal - Detail Page." *World Health Organization*, World Health Organization, `https://immunizationdata.who.int/global/wiise-detail-page/measles-vaccination-coverage?GROUP=Countries&ANTIGEN=MCV1&YEAR=&CODE=`. Accessed 25 Nov 2024.

## Import libraries and tools

```
import numpy as np
import pandas as pd
from sklearn.impute import KNNImputer
```

# Data Cleaning

## Load Raw Data

```
# Load raw data
df = pd.read_csv("LifeExpectancyRaw.csv")
```

## Drop Rows

```
# Drop rows where life expectancy is missing (10 countries where the
country only has one year of data)
df = df.dropna(subset=['Life expectancy '])

# Drop South Sudan due to large portion of missing data
df = df[df['Country'] != 'South Sudan']
```

## Convert Columns to Binary

```
# Convert status column to binary
df['Status'].replace({'Developed': 1, 'Developing': 0}, inplace=True)
```

## Manually Insert Data

```
# Manually insert population
pop_df = pd.read_csv("population-and-demography.csv")

# Rename values and columns for merging
pop_df['Entity'] = pop_df['Entity'].replace({
    'Bolivia': 'Bolivia (Plurinational State of)',
    'Brunei': 'Brunei Darussalam',
    "Cote d'Ivoire": "Côte d'Ivoire",
    'Cape Verde': 'Cabo Verde',
    'Democratic Republic of Congo': 'Democratic Republic of the
Congo',
    'North Korea': "Democratic People's Republic of Korea",
    'Iran': 'Iran (Islamic Republic of)',
    'South Korea': 'Republic of Korea',
    'Laos': "Lao People's Democratic Republic",
    'Micronesia (country)': 'Micronesia (Federated States of)',
    'Moldova': 'Republic of Moldova',
    'North Macedonia': 'The former Yugoslav republic of Macedonia',
```

```python
    'Tanzania': 'United Republic of Tanzania',
    'United Kingdom': 'United Kingdom of Great Britain and Northern
Ireland',
    'United States': 'United States of America',
    'Venezuela': 'Venezuela (Bolivarian Republic of)',
    'Vietnam': 'Viet Nam',
    'Russia': 'Russian Federation',
    'Eswatini': 'Swaziland',
    'Syria': 'Syrian Arab Republic',
    'East Timor': 'Timor-Leste'
})

pop_df = pop_df.rename(columns={
    'Entity': 'Country',
    "Population - Sex: all - Age: all - Variant: estimates":
"Population"})

# Merge df with pop_df on Country and Year
df = df.merge(pop_df[['Country', 'Year', 'Population']],
on=['Country', 'Year'], how='left', suffixes=('', '_pop_df'))

# Fill missing values in df's Population column with values from
pop_df
df['Population'] = df['Population_pop_df']

# Drop the additional Population column from pop_df after filling in
missing values
df = df.drop(columns=['Population_pop_df'])

# Load measles data
measles_df = pd.read_csv('MeaslesCoverage.csv', encoding='latin1')

# Replace original values with NaN
df['Measles '] = np.nan

# Merge df with measles_df on 'Country' and 'Year'
measles_df = measles_df.rename(columns={'NAME': 'Country', 'YEAR':
'Year'})
merged_df = df.merge(measles_df[['Country', 'Year', 'COVERAGE']],
on=['Country', 'Year'], how='left')

# Fill NaN values in the 'Measles' column with corresponding
'COVERAGE' values from measles_df
df['Measles '] = merged_df['Measles '].fillna(merged_df['COVERAGE'])

# Replace values in 'Measles' column that fall outside the expected
percentage range with NaN
df.loc[df['Measles '] > 100, 'Measles '] = np.nan
```

```
# Manually insert under-five deaths
mortality_df = pd.read_csv("child-mortality.csv")

# Rename values and columns for merging
mortality_df['Entity'] = mortality_df['Entity'].replace({
    'Bolivia': 'Bolivia (Plurinational State of)',
    'Brunei': 'Brunei Darussalam',
    "Cote d'Ivoire": "Côte d'Ivoire",
    'Cape Verde': 'Cabo Verde',
    'Democratic Republic of Congo': 'Democratic Republic of the
Congo',
    'North Korea': "Democratic People's Republic of Korea",
    'Iran': 'Iran (Islamic Republic of)',
    'South Korea': 'Republic of Korea',
    'Laos': "Lao People's Democratic Republic",
    'Micronesia (country)': 'Micronesia (Federated States of)',
    'Moldova': 'Republic of Moldova',
    'North Macedonia': 'The former Yugoslav republic of Macedonia',
    'Tanzania': 'United Republic of Tanzania',
    'United Kingdom': 'United Kingdom of Great Britain and Northern
Ireland',
    'United States': 'United States of America',
    'Venezuela': 'Venezuela (Bolivarian Republic of)',
    'Vietnam': 'Viet Nam',
    'Russia': 'Russian Federation',
    'Eswatini': 'Swaziland',
    'Syria': 'Syrian Arab Republic',
    'East Timor': 'Timor-Leste'
})

mortality_df = mortality_df.rename(columns={
    'Entity': 'Country',
    'Under-five mortality rate': 'under-five deaths '})

# Merge df with pop_df on Country and Year
df = df.merge(mortality_df[['Country', 'Year', 'under-five deaths ']],
on=['Country', 'Year'], how='left', suffixes=('', '_mortality_df'))

# Replace under-five deaths with values from mortality_df
df['under-five deaths '] = df['under-five deaths '].astype(float) #
Make columns the same dtype to avoid errors
df['under-five deaths '] = df['under-five deaths _mortality_df'] * 10
# Adjust from percentage to rate per 1000

# Drop the additional column from mortality_df after filling in values
df = df.drop(columns=['under-five deaths _mortality_df'])

# Manually insert GDP data
gdp_df = pd.read_csv("GDP.csv", skiprows=4)
gdp_df.drop(columns=['Country Code', 'Indicator Code', '1960', '1961',
'1962', '1963', '1964', '1965', '1966', '1967', '1968',
       '1969', '1970', '1971', '1972', '1973', '1974', '1975', '1976',
'1977',
       '1978', '1979', '1980', '1981', '1982', '1983', '1984', '1985',
'1986',
       '1987', '1988', '1989', '1990', '1991', '1992', '1993', '1994',
'1995',
       '1996', '1997', '1998', '1999', '2016', '2017', '2018', '2019',
'2020', '2021', '2022',
       '2023', 'Unnamed: 68'], inplace=True)            # drop
unnecessary columns

# Rename column names to match our data
gdp_df['Country Name'] = gdp_df['Country Name'].replace({
    'Bahamas, The': 'Bahamas',
    'Bolivia': 'Bolivia (Plurinational State of)',
    "Cote d'Ivoire": "Côte d'Ivoire",
    'Congo, Dem. Rep.':'Democratic Republic of the Congo',
    'Congo, Rep.':'Congo',
    'Egypt, Arab Rep.':'Egypt',
    'Gambia, The': "Gambia",
    'Iran, Islamic Rep.': 'Iran (Islamic Republic of)',
    'Korea, Dem. People\'s Rep.':"Democratic People's Republic of
Korea",
    'Korea, Rep.': 'Republic of Korea',
    'Kyrgyz Republic': 'Kyrgyzstan',
    "Lao PDR": "Lao People's Democratic Republic",
    'Micronesia, Fed. Sts.': 'Micronesia (Federated States of)',
    'Moldova': 'Republic of Moldova',
    'North Macedonia': 'The former Yugoslav republic of Macedonia',
    'Slovak Republic': 'Slovakia',
    'St. Lucia': 'Saint Lucia',
    'St. Vincent and the Grenadines': 'Saint Vincent and the
Grenadines',
    'Tanzania': 'United Republic of Tanzania',
    'United Kingdom': 'United Kingdom of Great Britain and Northern
Ireland',
    'United States': 'United States of America',
    'Venezuela, RB': 'Venezuela (Bolivarian Republic of)',
    'Vietnam': 'Viet Nam',
    'Yemen, Rep.': 'Yemen'
})
# Turn year columns into seperate rows
gdp_df = pd.melt(gdp_df,
                   id_vars=['Country Name', 'Indicator Name'],
                   var_name='Year',
                   value_name='GDP')

# Change year to int to be able to merge
```

```python
gdp_df['Year'] = gdp_df['Year'].astype(int)

# Merge df with gdp_df on 'Country' and 'Year'
gdp_df_merged = df.merge(gdp_df[['Country Name', 'Year', 'GDP']],
left_on=['Country', 'Year'],
                         right_on=['Country Name', 'Year'],
                         how='left',
                         suffixes=('', '_new'))

# Add Venezuela data
years = list(range(2000,2016))
gdp_ven = [11.9e3, 12.1e3, 10.8e3, 9.83e3, 11.4e3, 12.4e3, 13.4e3,
14.4e3, 14.9e3, 14.2e3, 13.8e3, 14.2e3, 14.7e3, 14.7e3, 1.4e4, 14.1e3]
venezuela_gdp = pd.DataFrame({
    'Country': 'Venezuela (Bolivarian Republic of)',
    'Year': years,
    'GDP_v': gdp_ven
})

# Add North Korea data
gdp_nk = [10.61e9, 11.02e9, 10.91e9, 11.05e9, 11.17e9, 13.03e9,
13.76e9, 14.37e9, 13.34e9, 12.04e9, 13.95e9, 15.69e9, 15.91e9,
16.57e9, 17.4e9, 16.28e9]
nk_gdp = pd.DataFrame({
    'Country': "Democratic People's Republic of Korea",
    'Year':years,
    'GDP_nk':gdp_nk
})

# Merge GDP data back into the main DataFrame on Country and Year
gdp_df_merged = gdp_df_merged.merge(venezuela_gdp, on=['Country',
'Year'], how='outer')
gdp_df_merged = gdp_df_merged.merge(nk_gdp, on=['Country', 'Year'],
how='outer')

# Fill missing GDP values
gdp_df_merged['GDP'] =
gdp_df_merged['GDP'].fillna(gdp_df_merged['GDP_new'])
gdp_df_merged['GDP'] =
gdp_df_merged['GDP'].fillna(gdp_df_merged['GDP_v'])
gdp_df_merged['GDP'] =
gdp_df_merged['GDP'].fillna(gdp_df_merged['GDP_nk']/gdp_df_merged['Pop
ulation'])

# Drop joined columns after filling in missing values
df = gdp_df_merged.drop(columns=['Country Name', 'GDP_v', 'GDP_new',
'GDP_nk'])

# Manually insert infant mortality
infant_df = pd.read_csv("infant-mortality.csv")
```

```
# Rename values and columns for merging
infant_df['Entity'] = infant_df['Entity'].replace({
    'Bolivia': 'Bolivia (Plurinational State of)',
    'Brunei': 'Brunei Darussalam',
    "Cote d'Ivoire": "Côte d'Ivoire",
    'Cape Verde': 'Cabo Verde',
    'Democratic Republic of Congo': 'Democratic Republic of the
Congo',
    'North Korea': "Democratic People's Republic of Korea",
    'Iran': 'Iran (Islamic Republic of)',
    'South Korea': 'Republic of Korea',
    'Laos': "Lao People's Democratic Republic",
    'Micronesia (country)': 'Micronesia (Federated States of)',
    'Moldova': 'Republic of Moldova',
    'North Macedonia': 'The former Yugoslav republic of Macedonia',
    'Tanzania': 'United Republic of Tanzania',
    'United Kingdom': 'United Kingdom of Great Britain and Northern
Ireland',
    'United States': 'United States of America',
    'Venezuela': 'Venezuela (Bolivarian Republic of)',
    'Vietnam': 'Viet Nam',
    'Russia': 'Russian Federation',
    'Eswatini': 'Swaziland',
    'Syria': 'Syrian Arab Republic',
    'East Timor': 'Timor-Leste'
})
infant_df = infant_df.rename(columns={'Entity': 'Country'})

# Merge df with pop_df on Country and Year
df = df.merge(infant_df[['Country', 'Year', 'infant deaths']],
on=['Country', 'Year'], how='left', suffixes=('', '_infant_df'))

# Replace infant deaths with values from infant_df
df['infant deaths'] = df['infant deaths_infant_df'] * 10 # Adjust from
percentage to rate per 1000

# Drop the additional column from infant_df after filling in values
df = df.drop(columns=['infant deaths_infant_df'])
```

## Forward/Backward Fill and Linear Interpolation by Country

```
def fill_missing_values(group, col):
    # Interpolate values for years 2001-2014
    group.loc[(group['Year'] > 2000) & (group['Year'] < 2015), col] =
group[col].interpolate(method='linear')

    # For year 2000, fill with the next year's value if available
    group.loc[group['Year'] == 2000, col] = group.loc[group['Year'] ==
2000, col].fillna(group.loc[group['Year'] == 2001, col])
```

```
    # For year 2015, fill with the previous year's value if available
    group.loc[group['Year'] == 2015, col] = group.loc[group['Year'] ==
2015, col].fillna(group.loc[group['Year'] == 2014, col])

    return group

# Fill missing values by interpolation
start_col, end_col = df.columns.get_loc('Status'),
df.columns.get_loc('Schooling')
for col in df.columns[start_col:end_col + 1]:
  df = df.groupby('Country', group_keys=False).apply(lambda group:
fill_missing_values(group, col))
  df = df.reset_index(drop=True)
```

## Fix Values that Fall Outside the Reasonable Range

```
# Replace values outside the reasonable adult mortality range with NaN
df.loc[~df['Adult Mortality'].between(1, 500), 'Adult Mortality'] =
np.nan

# Load sample data
fixed_df = pd.read_csv("LifeExpectancyCleanSample.csv")

# Rename columns in fixed_df
fixed_df['Country'] = fixed_df['Country'].replace({'Bahamas,
The':'Bahamas', 'Bolivia':'Bolivia (Plurinational State of)',
        "Cote d'Ivoire":"Côte d'Ivoire", 'Congo, Rep.':'Congo',
"Egypt, Arab Rep.":"Egypt",
        'Gambia, The':'Gambia', 'Iran, Islamic Rep.':'Iran (Islamic
Republic of)', 'Kyrgyz Republic':'Kyrgyzstan',
        "Lao PDR":"Lao People's Democratic Republic", 'Micronesia,
Fed. Sts.':'Micronesia (Federated States of)',
        'Moldova':'Republic of Moldova', 'St. Lucia':'Saint Lucia',
"Congo, Dem. Rep.":"Democratic Republic of the Congo",
        'St. Vincent and the Grenadines':'Saint Vincent and the
Grenadines', 'Slovak Republic':'Slovakia',
        'Eswatini':'Swaziland', 'North Macedonia':'The former Yugoslav
republic of Macedonia',
        'Turkiye':'Turkey', 'United Kingdom':'United Kingdom of Great
Britain and Northern Ireland',
        'Tanzania':'United Republic of Tanzania', 'United
States':'United States of America',
        'Venezuela, RB':'Venezuela (Bolivarian Republic of)',
'Vietnam':'Viet Nam', 'Yemen, Rep.':'Yemen'})

# Replace BMI values with BMI values from fixed_df
df.rename(columns={' BMI ': 'BMI'}, inplace=True)
df = df.merge(fixed_df[['Country', 'Year', 'BMI']], on=['Country',
'Year'], how='left', suffixes=('', '_new'))
```

```
df['BMI'] = df['BMI_new'].combine_first(df['BMI'])
df.drop(columns=['BMI_new'], inplace=True)

# For countries not found in fixed_df, we drop the BMI values in our
df to impute by knn
df.loc[df['Country'].isin(["Democratic People's Republic of Korea",
'Republic of Korea', 'Sudan']), 'BMI'] = np.nan

# Replace 0 values in 'Total expenditure' with NaN
df['percentage expenditure'] = df['percentage expenditure'].replace(0,
np.nan)
```

## KNN

```
# Store the 'Country' column temporarily in order to perform KNN on
Numeric Data
country_col = df['Country']
df = df.drop(columns=['Country'])

# Apply KNN Imputer for Remaining NaN Values
knn_imputer = KNNImputer(n_neighbors=2)
df = pd.DataFrame(knn_imputer.fit_transform(df), columns=df.columns)

# Re-add 'Country' to the DataFrame
df['Country'] = country_col.values
```

## Formatting

```
# Strip columns of leading and trailing whitespace
df.columns = df.columns.str.strip()

# Rename columns to create consistent capitalization
df.rename(columns={'Life expectancy':'Life Expectancy', 'infant
deaths':'Infant Deaths', 'percentage expenditure':'Percentage
Expenditure',
                   'under-five deaths':'Under-five Deaths', 'Total
expenditure':'Total Expenditure', 'thinness  1-19 years':'Thinness 1-
19 Years',
                   'thinness 5-9 years':'Thinness 5-9 Years', 'Income
composition of resources':'Income Composition of Resources'},
inplace=True)

# Sort by countries
cols = ['Country'] + [col for col in df.columns if col != 'Country']
df = df[cols]
df = df.sort_values(by=['Country', 'Year'])

# Rename lengthy country names
df['Country'] = df['Country'].replace({'Bolivia (Plurinational State
of)':'Bolivia', "Côte d'Ivoire":"Cote d'Ivoire", 'Democratic Republic
of the Congo':'DR of the Congo',
                   'Iran (Islamic Republic of)':'Iran', "Lao People's
Democratic Republic":"Lao People's DR",
                   'Micronesia (Federated States of)':'Micronesia',
'Saint Vincent and the Grenadines':'St Vincent and the Grenadines',
                   'Russian Federation':'Russia', 'The former Yugoslav
republic of Macedonia':'North Macedonia',
                   'United Kingdom of Great Britain and Northern
Ireland': 'United Kingdom',
                   'United States of America':'United States',
'Venezuela (Bolivarian Republic of)':'Venezuela',
                   'Viet Nam':'Vietnam'})
```

## Assertions

```
assert df.duplicated().sum() == 0, "The DataFrame contains duplicate
rows."
assert df.isnull().sum().sum() == 0, "There are null values in the
DataFrame."
assert df.groupby('Country')['Year'].apply(lambda x: set(range(2000,
2016)).issubset(x)).all(), "Not all countries have entries for each
year from 2000 to 2015."
assert df['Life Expectancy'].between(35, 100).all(), "Life expectancy
values are out of the specified range"
assert df['Adult Mortality'].between(1, 500).all(), "Adult mortality
values are out of the specified range"
assert df['Infant Deaths'].between(1, 500).all(), "Infant mortality
values are out of the specified range"
assert df['Alcohol'].between(0, 18).all(), "Alcohol consumption values
are out of the specified range"
assert df['Percentage Expenditure'].between(0, 30000).all(),
"Percentage expenditure values are out of the specified range"
assert df['Hepatitis B'].between(0, 100).all(), "Hepatitis B values
are out of the specified range"
assert df['Measles'].between(0, 100).all(), "Measles values are out of
the specified range"
assert df['BMI'].between(18, 33).all(), "BMI values are out of the
specified range"
assert df['Under-five Deaths'].between(1, 1000).all(), "Under-five
mortality values are out of the specified range"
assert df['Polio'].between(0, 100).all(), "Polio values are out of the
specified range"
assert df['Total Expenditure'].between(0, 30000).all(), "Total
expenditure values are out of the specified range"
assert df['Diphtheria'].between(0, 100).all(), "Diptheria values are
out of the specified range"
assert df['HIV/AIDS'].between(0, 300).all(), "HIV/AIDS values are out
of the specified range"
assert df['GDP'].between(0, 300000).all(), "GDP values are out of the
```

```
specified range"
assert df['Population'].between(100, 2000000000).all(), "Population
values are out of the specified range"
assert df['Thinness 1-19 Years'].between(0, 60).all(), "Thinness (1-
19) values are out of the specified range"
assert df['Thinness 5-9 Years'].between(0, 60).all(), "Thinness (5-9)
values are out of the specified range"
assert df['Income Composition of Resources'].between(0, 1).all(),
"Income composition resources values are out of the specified range"
assert df['Schooling'].between(0, 25).all(), "Schooling values are out
of the specified range"
```

### Export Clean Data to CSV

```
df.to_csv('LifeExpectancyDraft.csv', index=False)
```

# Feature Engineering

```
# Read in cleaned data
df = pd.read_csv('LifeExpectancyDraft.csv')
```

### Population Density

```
# Load Area Data
area_df = pd.read_csv('LandAreakm.csv')

# Rename Columns for clarity
area_df = area_df.rename(columns={'Entity': 'Country', 'Land area (sq.
km)': 'Area'})

# Add 'Area' Column to df
area_df = area_df.groupby('Country', as_index=False).agg({'Area':
'first'})
df = df.merge(area_df[['Country', 'Area']], on='Country', how='left')

# Drop Duplicate Rows
df = df.drop_duplicates()

# Create Population Density Column
df['PopulationDensity'] = df['Population'] / df['Area']

# Remove Area Column
df = df.drop('Area', axis=1)
```

### Population Growth Rate

```
# Load population data
pop_df = pd.read_csv('population-and-demography.csv')
```

```python
# rename population column
pop_df.rename(columns={'Population - Sex: all - Age: all - Variant: 
estimates':'Population_1999'}, inplace=True)

pop_df['Entity'] = pop_df['Entity'].replace({
    'Brunei': 'Brunei Darussalam',
    'Cape Verde': 'Cabo Verde',
    'Democratic Republic of Congo': 'DR of the Congo',
    'Laos': "Lao People's DR",
    'Micronesia (country)': 'Micronesia',
    'Moldova': 'Republic of Moldova',
    'Tanzania': 'United Republic of Tanzania',
    'Eswatini': 'Swaziland',
    'Saint Vincent and the Grenadines': 'St Vincent and the 
Grenadines',
    'South Korea': 'Republic of Korea',
    'Syria': 'Syrian Arab Republic',
    'East Timor': 'Timor-Leste'
})

# Calculate population growth rate
df["Population Growth Rate"] = (
    df.groupby("Country")["Population"]
    .apply(lambda x: x.diff() / x.shift()).reset_index(level=0, 
drop=True)
)

# Filter datasets for years 1999 and 2000
df_1999 = pop_df[pop_df["Year"] == 1999]
df_2000 = df[df["Year"] == 2000]

# Manually insert North Korea Population data
nk_row = pd.DataFrame({"Entity": ["Democratic People's Republic of 
Korea"], "Code": ["CODE"], "Year": [1999], "Population_1999": 
[23204498]})
df_1999 = pd.concat([df_1999, nk_row], ignore_index=True)

# Merge the 2000 data with the 1999 poopulation data
df_merged = pd.merge(df_2000, df_1999, left_on="Country", 
right_on="Entity", suffixes=("_2000", "_1999"))

# Calculate the population growth rate for the year 2000
df_merged["Population Growth Rate"] = (
    (df_merged["Population"] - df_merged["Population_1999"])
    / df_merged["Population_1999"]
)

# Create dictionary of population growth rates
growth_rate_dict = dict(zip(df_merged["Country"], 
df_merged["Population Growth Rate"]))
```

```
# Add the Growth Rate from 2000 into the main df
df.loc[df["Year"] == 2000, "Population Growth Rate"] =
df.loc[df["Country"].map(growth_rate_dict).notna(),
"Country"].map(growth_rate_dict)
```

## BMI Classification

```
# Create columns for each BMI classification based on the BMI range
df['UnderweightBMI'] = (df['BMI'] < 18.5).astype(int)
df['HealthyWeightBMI'] = ((df['BMI'] >= 18.5) & (df['BMI'] <
25)).astype(int)
df['OverweightBMI'] = ((df['BMI'] >= 25) & (df['BMI'] <
30)).astype(int)
df['ObesityBMI'] = (df['BMI'] >= 30).astype(int)
```

## Income Classification

```
# Low-income economies: GNI per capita of $1,135 or less
# Lower-middle-income economies: GNI per capita between $1,136 and
$4,465
# Upper-middle-income economies: GNI per capita between $4,466 and
$13,845
# High-income economies: GNI per capita of $13,846 or more
df['LowIncome'] = (df['GDP'] < 1136).astype(int)
df['LowerMiddleIncome'] = ((df['GDP'] >= 1136) & (df['GDP'] <
4466)).astype(int)
df['UpperMiddleIncome'] = ((df['GDP'] >= 4466) & (df['GDP'] <
13846)).astype(int)
df['HighIncome'] = (df['GDP'] >= 13846).astype(int)
```

## Health Coverage Index

- Positive index means above average
- Negative index means below average
- Zero index means exactly average

```
# Reverse necessary columns so a higher number means better health
for col in ['Measles', 'HIV/AIDS', 'Thinness 1-19 Years', 'Thinness 5-
9 Years']:
    df[f'{col}_adjusted'] = df[col].max() - df[col]

# Standardize all columns using z-score normalization
for col in ['Hepatitis B', 'Polio', 'Diphtheria', 'Measles_adjusted',
'HIV/AIDS_adjusted', 'Thinness 1-19 Years_adjusted', 'Thinness 5-9
Years_adjusted']:
    df[f'{col}_z'] = (df[col] - df[col].mean()) / df[col].std()

# Combine into a health index by taking the weighted mean of z-scores
z_cols = [f'{col}_z' for col in ['Hepatitis B', 'Polio', 'Diphtheria',
'Measles_adjusted', 'HIV/AIDS_adjusted', 'Thinness 1-19
Years_adjusted', 'Thinness 5-9 Years_adjusted']]
weights = {
    'Hepatitis B_z': 0.2,
    'Polio_z': 0.2,
    'Diphtheria_z': 0.2,
    'Measles_adjusted_z': 0.15,
    'HIV/AIDS_adjusted_z': 0.1,
    'Thinness 1-19 Years_adjusted_z': 0.075,
    'Thinness 5-9 Years_adjusted_z': 0.075,
}

# Normalize weights
total_weight = sum(weights.values())
weights = {k: v / total_weight for k, v in weights.items()}

# Calculate the health index (handles missing data)
df['Health Coverage Index'] = df[z_cols].apply(
    lambda row: np.average(
        row.dropna(),
        weights=[weights[col] for col in row.index if col in weights]
    ) if not row.isnull().all() else np.nan,
    axis=1
)

# Drop unnecessary rows
df = df.drop(columns={'Measles_adjusted',
       'HIV/AIDS_adjusted', 'Thinness 1-19 Years_adjusted',
       'Thinness 5-9 Years_adjusted', 'Hepatitis B_z', 'Polio_z',
       'Diphtheria_z', 'Measles_adjusted_z', 'HIV/AIDS_adjusted_z',
       'Thinness 1-19 Years_adjusted_z', 'Thinness 5-9
Years_adjusted_z'})
```

## Human Rights

```
rights_df = pd.read_csv('human-rights-index-vdem.csv')

# Rename Columns for clarity
rights_df = rights_df.rename(columns={'Entity': 'Country'})

# Create the 'Human Rights' column in df using the mapping
rights_mapping = rights_df.set_index(['Country', 'Year'])['Civil
liberties index (best estimate, aggregate: average)'].to_dict()
df['Human Rights'] = df.apply(lambda row:
rights_mapping.get((row['Country'], row['Year']), None), axis=1)
```

## Electoral Democracy

```
democracy_df = pd.read_csv('electoral-democracy-index.csv')
```

```python
# Rename Columns for clarity
democracy_df = democracy_df.rename(columns={'Entity': 'Country'})

# Create 'Democracy' column
democracy_mapping = democracy_df.set_index(['Country', 'Year'])
['Electoral democracy index (best estimate, aggregate:
average)'].to_dict()
df['Democracy'] = df.apply(lambda row:
democracy_mapping.get((row['Country'], row['Year']), None), axis=1)
```

## Armed Personnel

```python
army_df = pd.read_csv('armed-forces-personnel-percent.csv')

# Rename Columns for clarity
army_df = army_df.rename(columns={'Entity': 'Country'})

# Create 'Army Personnel' column
army_mapping = army_df.set_index(['Country', 'Year'])['Armed forces
personnel (% of total population)'].to_dict()
df['Armed Personnel'] = df.apply(lambda row:
army_mapping.get((row['Country'], row['Year']), None), axis=1)
```

## Suicide Rate

```python
suicide_df = pd.read_csv('death-rate-from-suicides-gho.csv')

# Rename Columns for clarity
suicide_df = suicide_df.rename(columns={'Entity': 'Country'})

# Create 'Suicide Rate' column
suicide_mapping = suicide_df.set_index(['Country', 'Year'])['Age-
standardized death rate from self-harm amongboth sexes'].to_dict()
df['Suicide Rate'] = df.apply(lambda row:
suicide_mapping.get((row['Country'], row['Year']), None), axis=1)
```

## Agriculture Employment

```python
agr_df = pd.read_csv('agriculture-value-added-per-worker-wdi.csv')

# Rename Columns for clarity
agr_df = agr_df.rename(columns={'Entity': 'Country'})

# Create 'Emigration' column
agr_mapping = agr_df.set_index(['Country', 'Year'])["Agriculture,
forestry, and fishing, value added per worker (constant 2015
US$)"].to_dict()
df['Agriculture Employment'] = df.apply(lambda row:
agr_mapping.get((row['Country'], row['Year']), None), axis=1)
```

## KNN

```
# Store the 'Country' column temporarily in order to perform KNN
country_col = df['Country']
df = df.drop(columns=['Country'])

# Apply KNN Imputer for Remaining NaN Values
knn_imputer = KNNImputer(n_neighbors=2)
df = pd.DataFrame(knn_imputer.fit_transform(df), columns=df.columns)

# Re-add 'Country' to the DataFrame
df['Country'] = country_col.values
df = df[['Country'] + [col for col in df.columns if col != 'Country']]

df.to_csv('LifeExpectancyClean.csv')
```

## Import libraries and tools

```
import pandas as pd
import matplotlib.pyplot as plt
import shap
from sklearn.ensemble import RandomForestRegressor
from sklearn.model_selection import train_test_split
import geopandas as gpd
from matplotlib.patches import Rectangle
from matplotlib.colorbar import Colorbar
from matplotlib.colors import LogNorm, Normalize
from matplotlib.cm import ScalarMappable
import matplotlib.animation as animation
from matplotlib.ticker import FuncFormatter
```

## World map

```
# Load Data
world = gpd.read_file("worldmap.gpkg")
df = pd.read_csv('LifeExpectancyClean.csv', index_col=0)

# Rename 'ADMIN' Column to 'Country'
world = world.rename(columns={'ADMIN': 'Country'})

# Rename Countries in World
world['Country'] = world['Country'].replace('The Bahamas', 'Bahamas')
world['Country'] = world['Country'].replace('Brunei', 'Brunei
Darussalam')
world['Country'] = world['Country'].replace('Democratic Republic of
the Congo', 'DR of the Congo')
world['Country'] = world['Country'].replace("North Korea", "Democratic
People's Republic of Korea")
world['Country'] = world['Country'].replace('Republic of the Congo',
'Congo')
world['Country'] = world['Country'].replace("Republic of Cote
D'Ivoire", "Cote d'Ivoire")
world['Country'] = world['Country'].replace('Laos', "Lao People's DR")
world['Country'] = world['Country'].replace('South Korea', 'Republic
of Korea')
world['Country'] = world['Country'].replace('Moldova', 'Republic of
Moldova')
world['Country'] = world['Country'].replace('Republic of Serbia',
'Serbia')
world['Country'] = world['Country'].replace('eSwatini', 'Swaziland')
world['Country'] = world['Country'].replace('Syria', 'Syrian Arab
Republic')
world['Country'] = world['Country'].replace('East Timor', 'Timor-
Leste')
world['Country'] = world['Country'].replace('United States of
America', 'United States')


# Merge World and Life Expectancy Data on 'Country'
combined = world.merge(df, on='Country', how='left')
```

World map animation

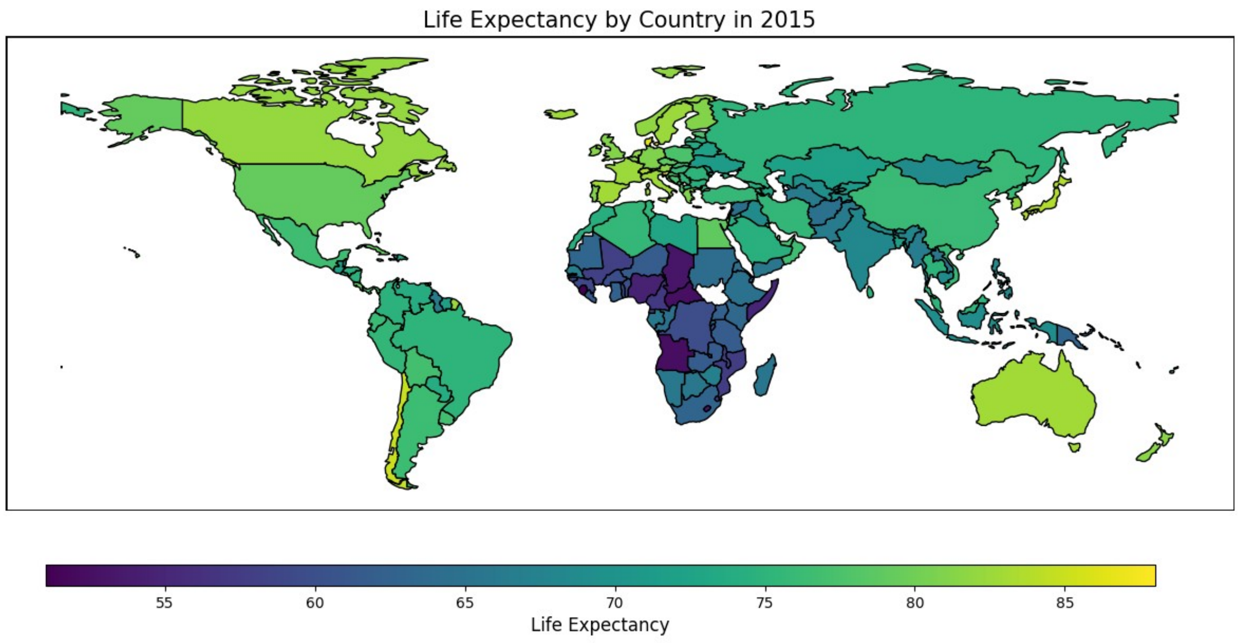
```
# Create figure and axis
fig, ax = plt.subplots(1, figsize=(10, 7))

# Normalize color scale for Life Expectancy
norm = Normalize(vmin=50, vmax=85)  # Manually set min and max values
for the color scale

def update(date):
    ax.clear()

    # Filter the data for the current year
    merged = combined[combined['Year'] == date]

    # Plot the Life Expectancy by country
    merged.plot(column="Life Expectancy", cmap="viridis", ax=ax,
edgecolor="gray", norm=norm)

    # Plot the World Country Outlines
    world.boundary.plot(ax=ax, linewidth=1, color="black")

    # Add Year Label
    ax.text(0.02, 0.95, str(int(date)), transform=ax.transAxes,
fontsize=12,
            verticalalignment='top', bbox=dict(facecolor='white',
alpha=0.7))

    # Add Title
    ax.set_title("Life Expectancy by Country from 2000-2015")
    ax.axis('off')

    # Add a black border around the plot
    rect = Rectangle(
        (0, 0), 1, 1, transform=ax.transAxes,  # (x, y, width, height)
in normalized axes coordinates
        color='black',
        linewidth=2,
        fill=False
    )
    ax.add_patch(rect)

# Set up the colorbar
sm = ScalarMappable(norm=norm, cmap="viridis")
```

```
cbar = fig.colorbar(sm, ax=ax, orientation="horizontal", pad=0)

# Manually define the colorbar ticks
ticks = [55, 60, 65, 70, 75, 80, 85]
cbar.set_ticks(ticks)
cbar.ax.xaxis.set_major_formatter(FuncFormatter(lambda x, _:
f'{x:.0f}'))  # Format as integers
cbar.set_label("Life Expectancy")

# Create the animation object
ani = animation.FuncAnimation(fig, update,
        frames=sorted(set(df['Year'])),  # Use years from df
        interval=300)

plt.close(fig)

# Save/Embed Animation
ani.save('LifeExpectancyWorld.mp4', writer='ffmpeg')
```

```
<Figure size 640x480 with 0 Axes>
```

World Life Expectancy 2015

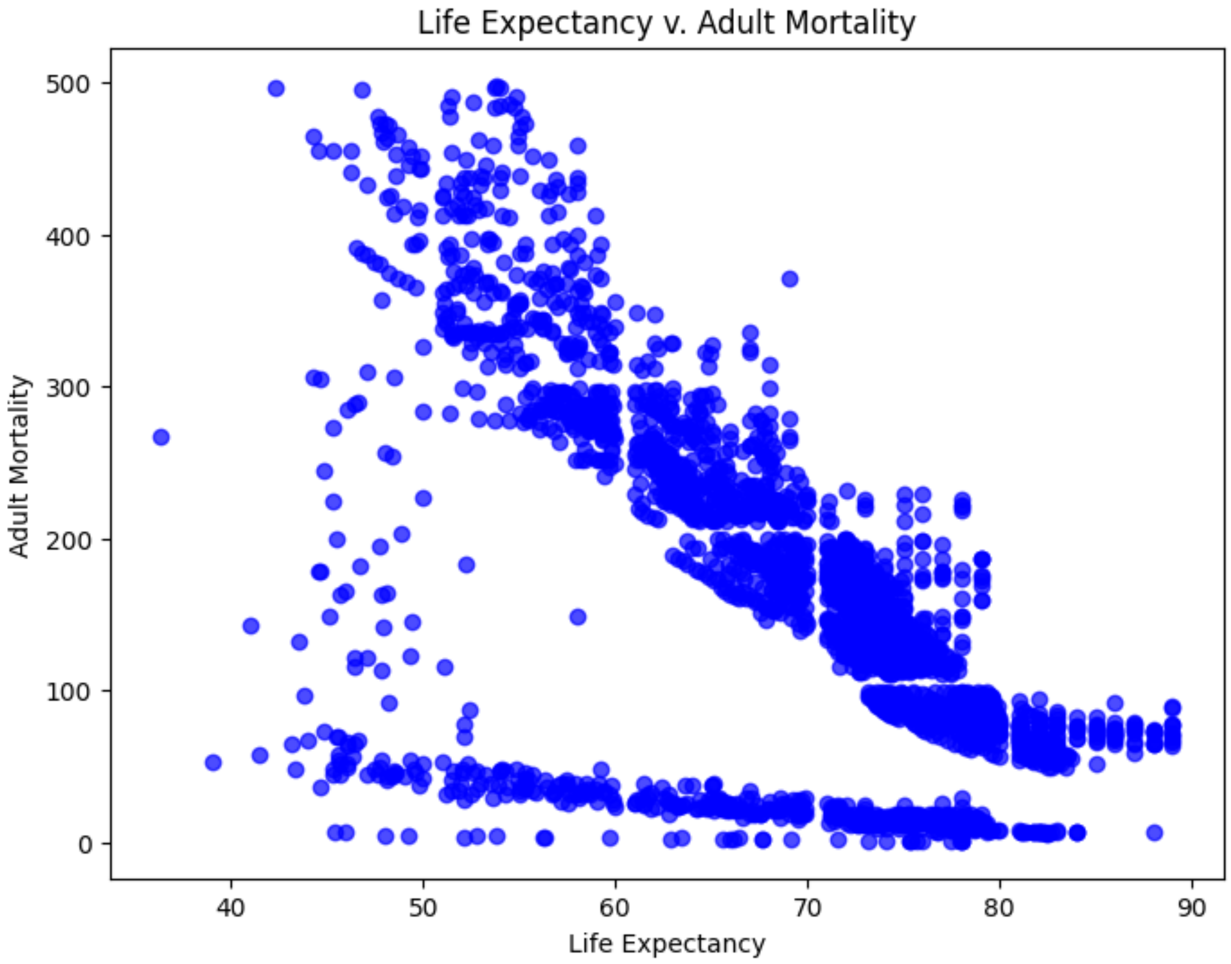
```
# Load and prepare data
df = pd.read_csv('LifeExpectancyClean.csv', index_col=0)
df = df[df['Year'] == 2015]

# Merge World and Life Expectancy Data on 'Country'
combined = world.merge(df, on='Country', how='left')

# Plot life expectancy data
fig, ax = plt.subplots(1, 1, figsize=(15, 10))

# Plot the data
combined.plot(
    column='Life Expectancy',
    cmap='viridis',
    legend=False,
    ax=ax,
    edgecolor='black'
)

# Turn off the axes
ax.axis('off')

# Add a black border around the plot
rect = Rectangle(
    (0, 0), 1, 1, transform=ax.transAxes,  # (x, y, width, height) in
normalized axes coordinates
    color='black',
    linewidth=2,
    fill=False
)
ax.add_patch(rect)

# Create a colorbar that matches the plot size
norm = Normalize(vmin=combined['Life Expectancy'].min(),
vmax=combined['Life Expectancy'].max())
sm = ScalarMappable(cmap='viridis', norm=norm)

# Add the colorbar
cbar = Colorbar(ax=ax.figure.add_axes([0.15, 0.2, 0.7, 0.02]),  #
Adjust [left, bottom, width, height]
                mappable=sm,
                orientation='horizontal')
cbar.set_label("Life Expectancy", fontsize=12)

# Set plot title
ax.set_title("Life Expectancy by Country in 2015", fontsize=15)

plt.show()
```

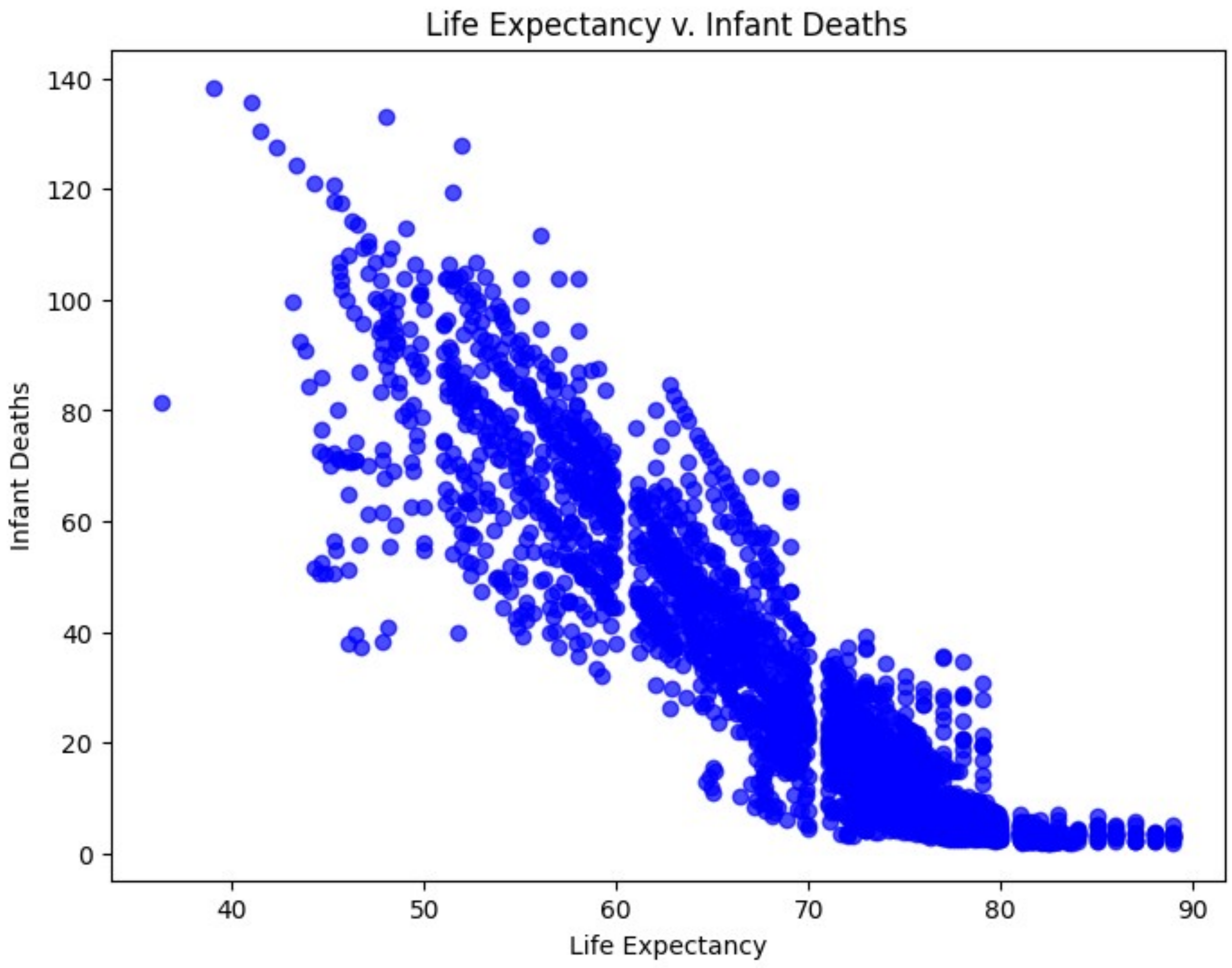


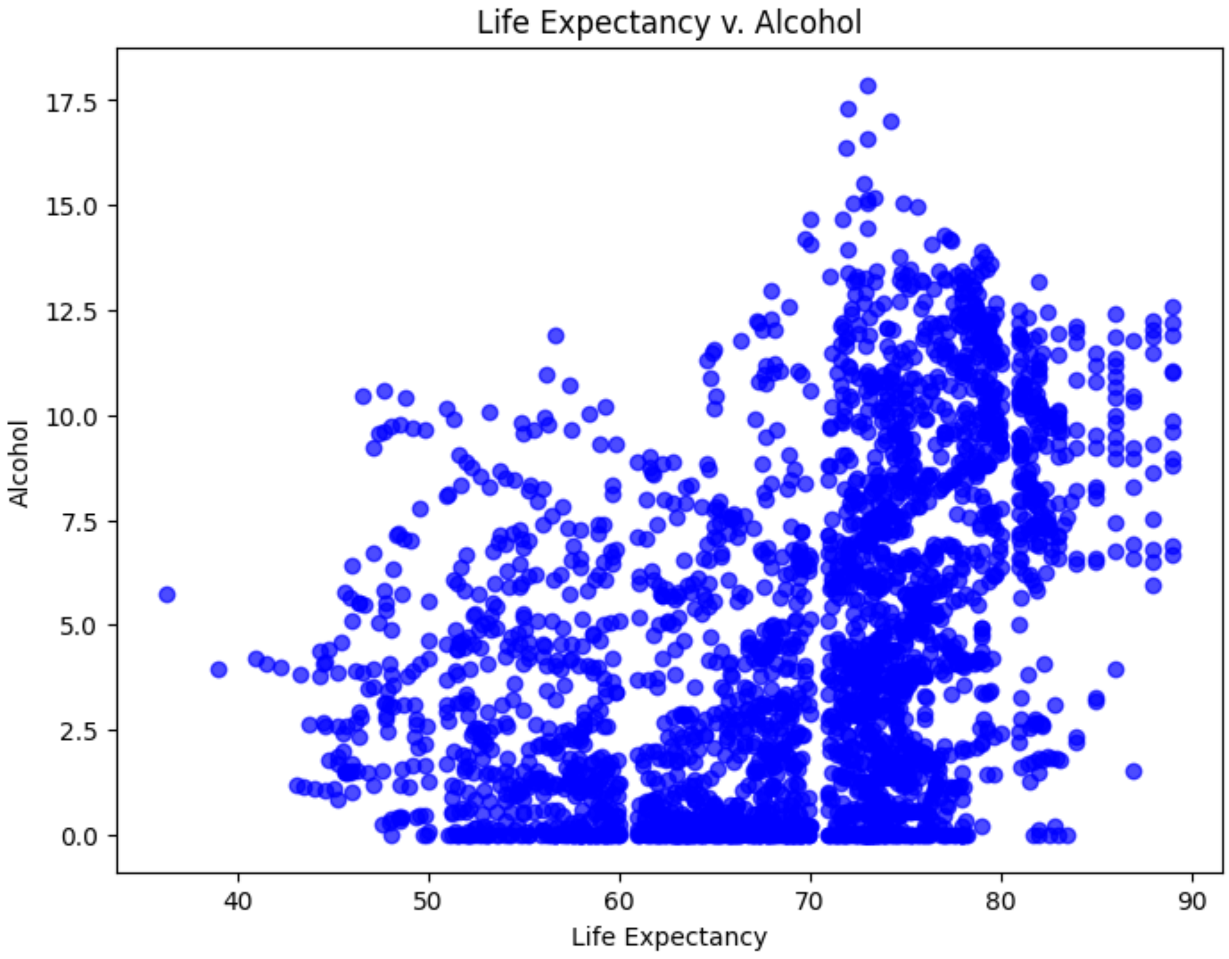
## Scatter plots

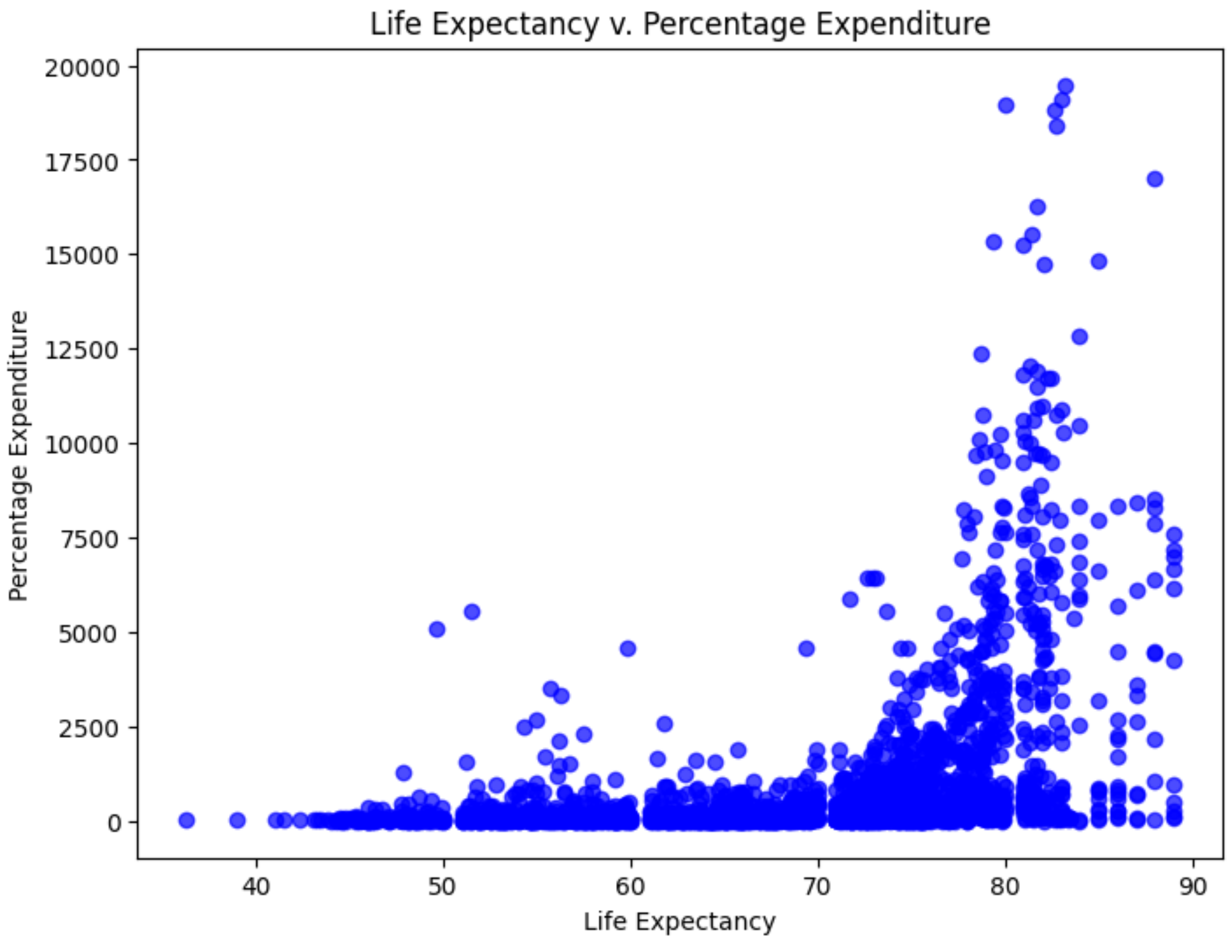
```
df = pd.read_csv('LifeExpectancyClean.csv')

continuous = ['Adult Mortality', 'Infant Deaths', 'Alcohol',
'Percentage Expenditure', 'Hepatitis B',\
              'Measles', 'BMI', 'Under-five Deaths', 'Polio', 'Total
Expenditure', 'Diphtheria',\
              'HIV/AIDS', 'GDP', 'Population', 'Thinness 1-19 Years',
```

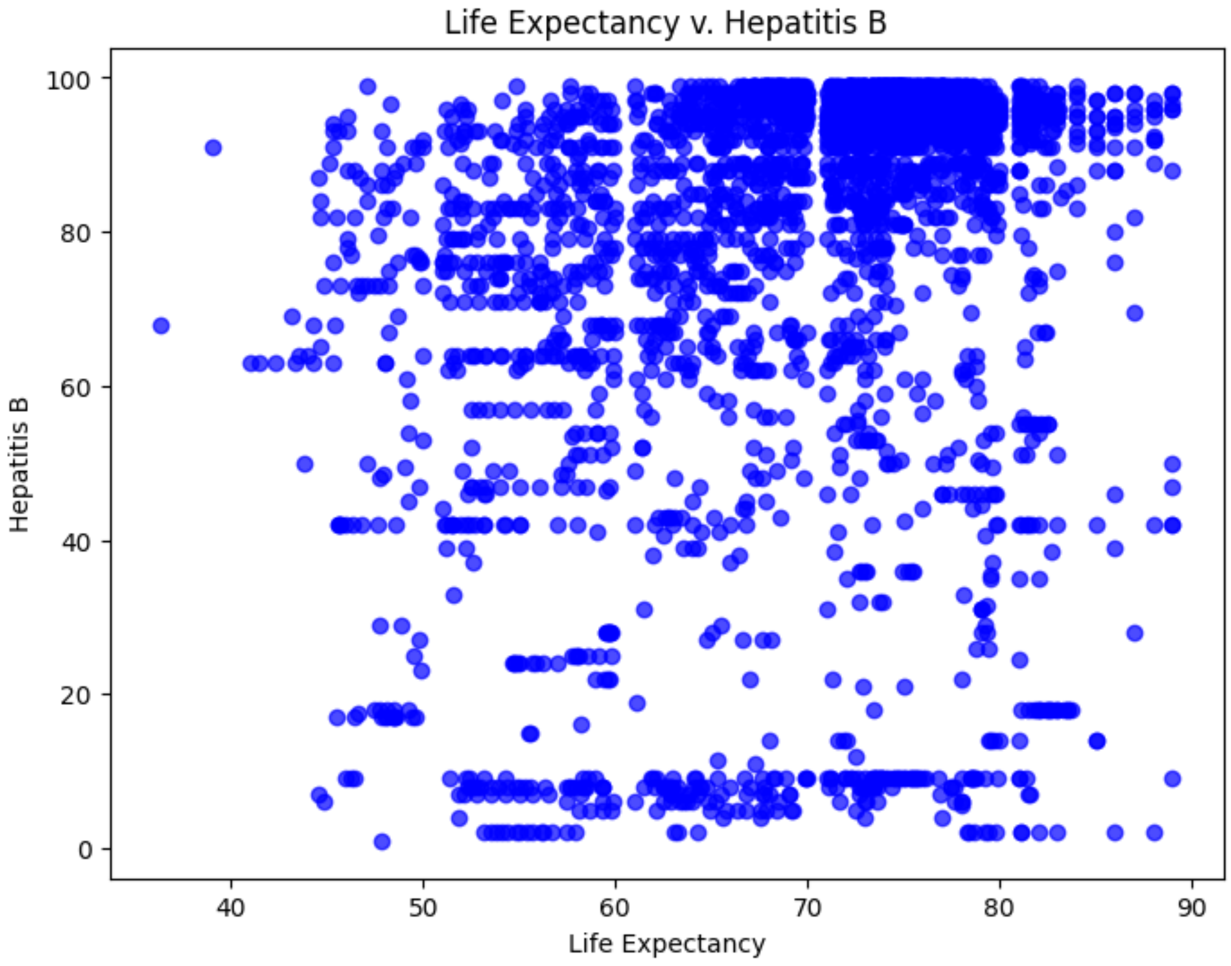
```python
'Thinness 5-9 Years',\
              'Income Composition of Resources', 'Schooling',
'PopulationDensity', 'Population Growth Rate',\
              'Health Coverage Index', 'Human Rights', 'Democracy',
'Armed Personnel', 'Suicide Rate',\
              'Agriculture Employment']


binary = ['UnderweightBMI', 'HealthyWeightBMI', 'OverweightBMI',
'ObesityBMI', 'LowIncome', 'LowerMiddleIncome',\
          'UpperMiddleIncome', 'HighIncome']

def scatter(feature):
    """ Plots a scatter plot of 'Life Expectancy' against a specified
feature.

    Parameters:
        feature (str): The name of the column from `df` to plot on the
y-axis.

    Returns:
        None. Displays the scatter plot.
    """
    # Specify the features to plot
    x_feature = 'Life Expectancy'
    y_feature = feature

    # Scatter plot
    plt.figure(figsize=(8, 6))
    plt.scatter(df[x_feature], df[y_feature], color='blue', alpha=0.7)
    plt.title(f"{x_feature} v. {y_feature}")
    plt.xlabel(x_feature)
    plt.ylabel(y_feature)
    plt.show()

for feature in continuous:
    scatter(feature)
```

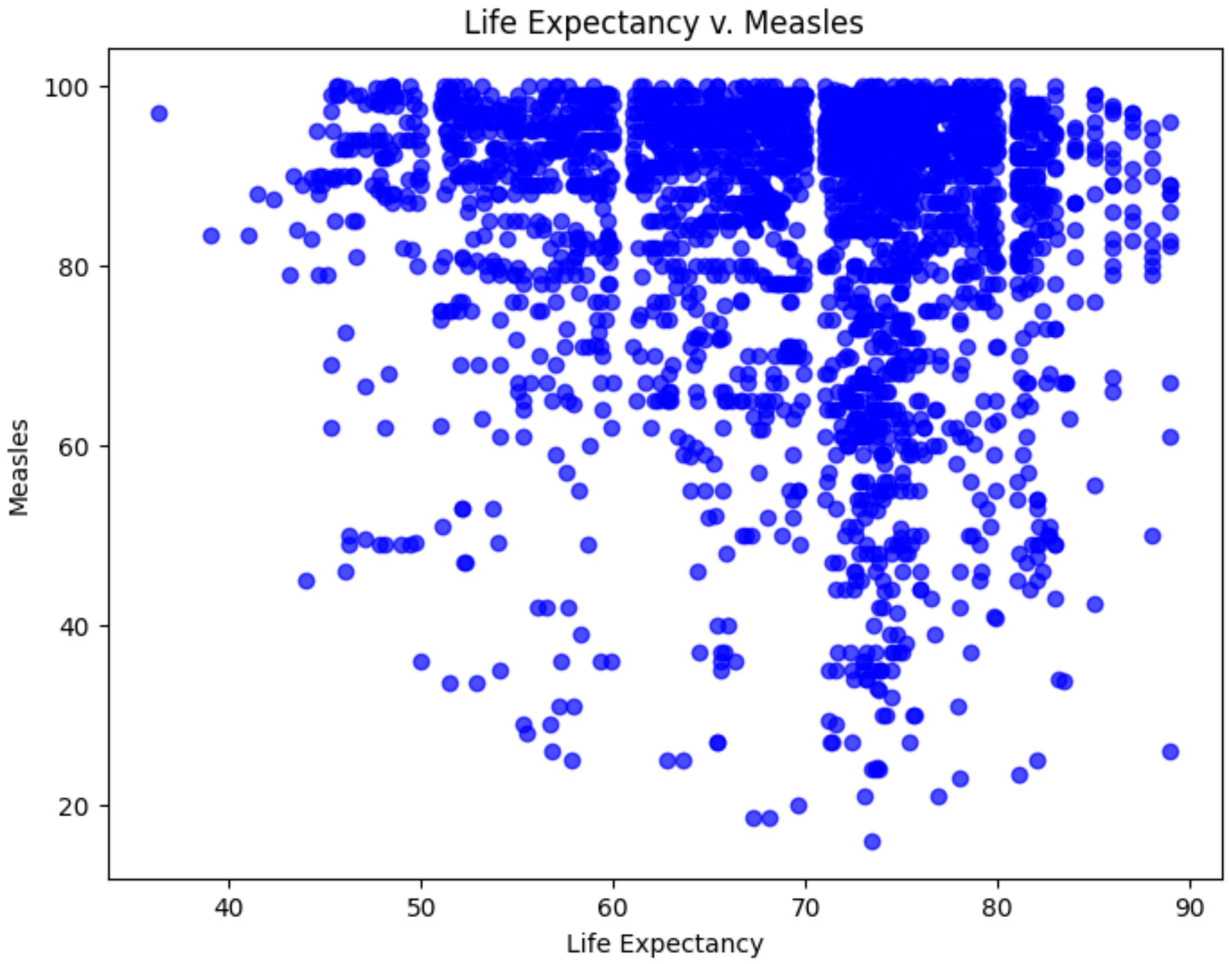

Life Expectancy v. Adult Mortality
Adult Mortality
Life Expectancy
0
100
200
300
400
500
40
50
60
70
80
90

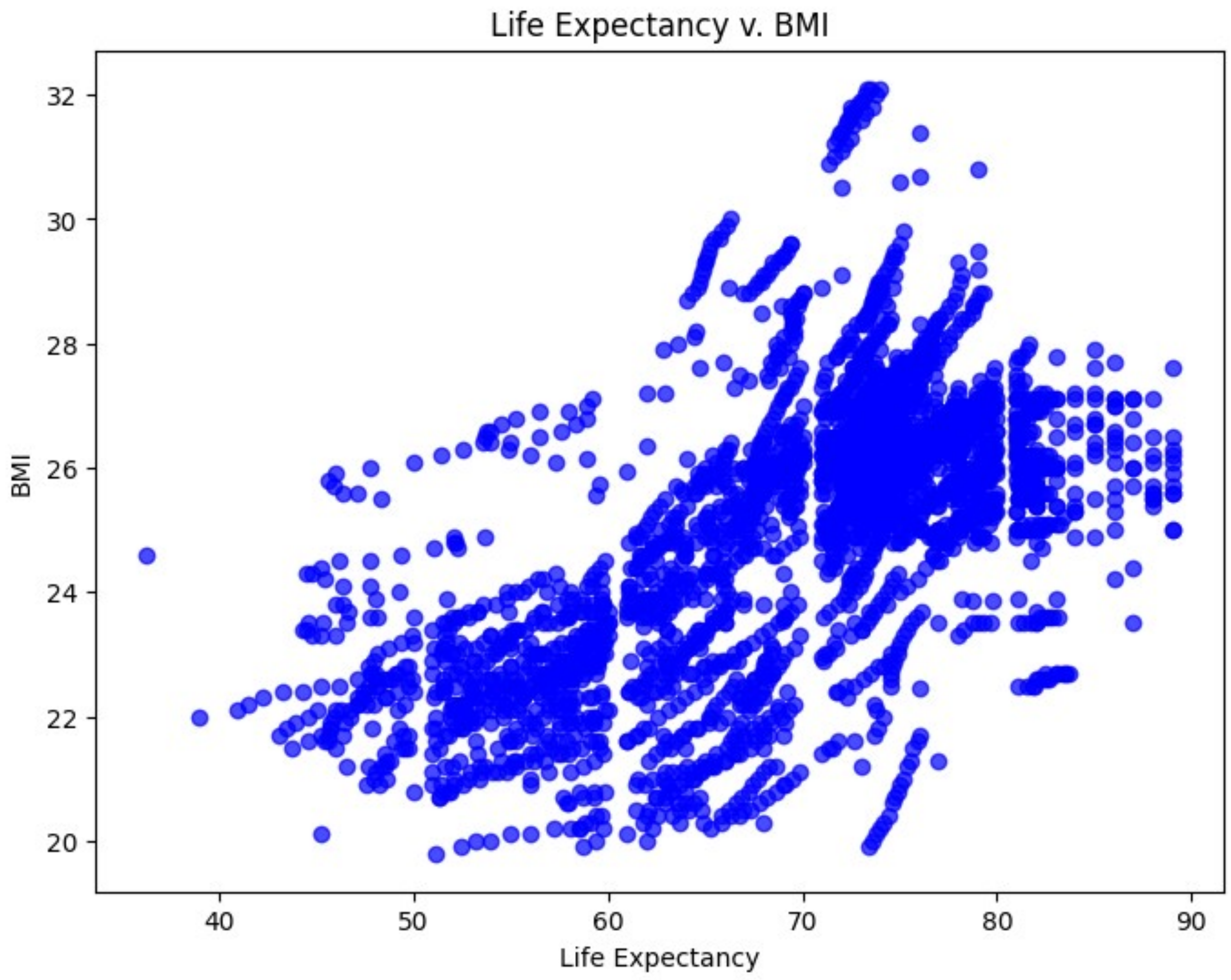

Life Expectancy v. Infant Deaths
Infant Deaths
Life Expectancy
0
20
40
60
80
100
120
140
40
50
60
70
80
90

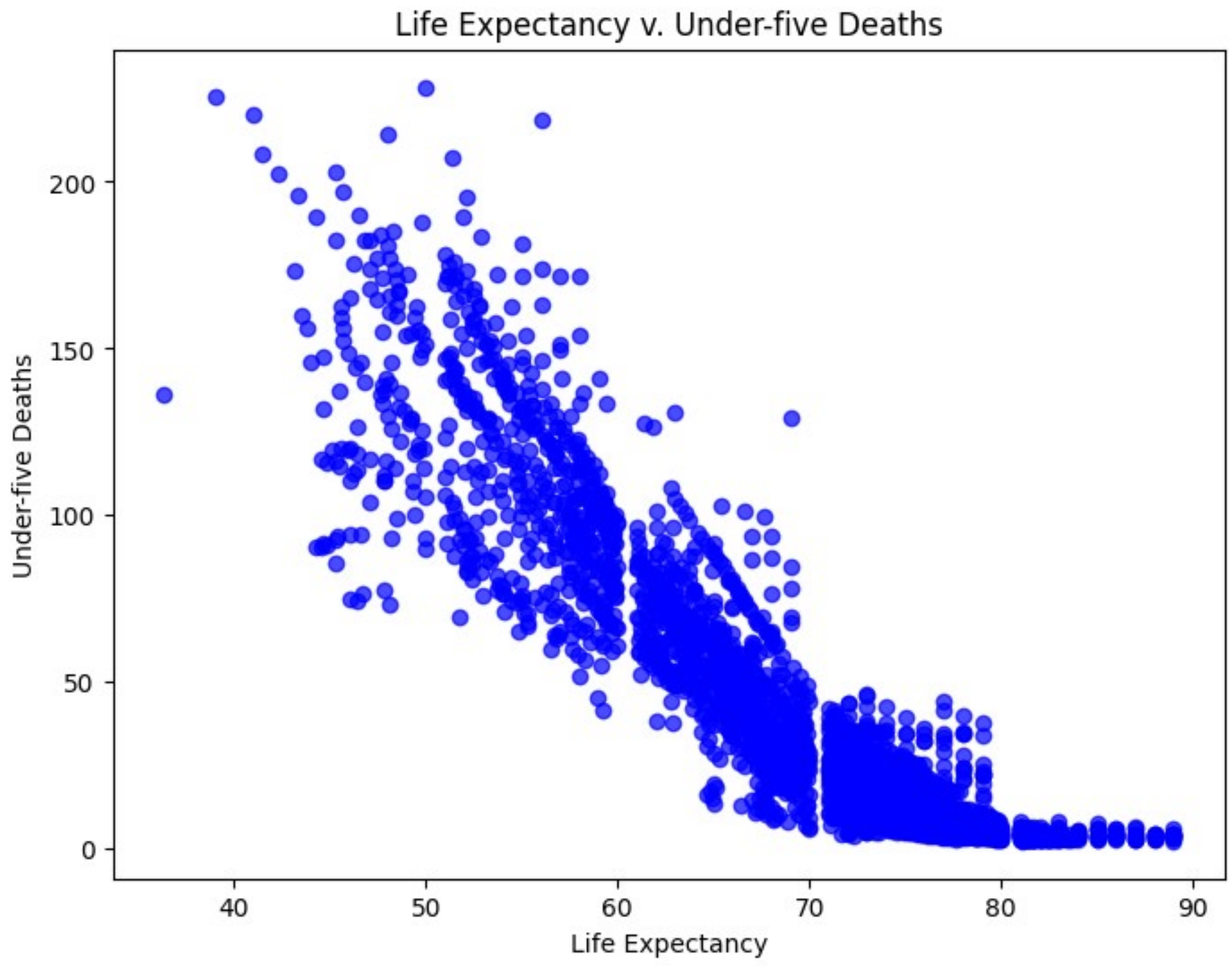

Life Expectancy v. Alcohol
Alcohol
Life Expectancy
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
40
50
60
70
80
90

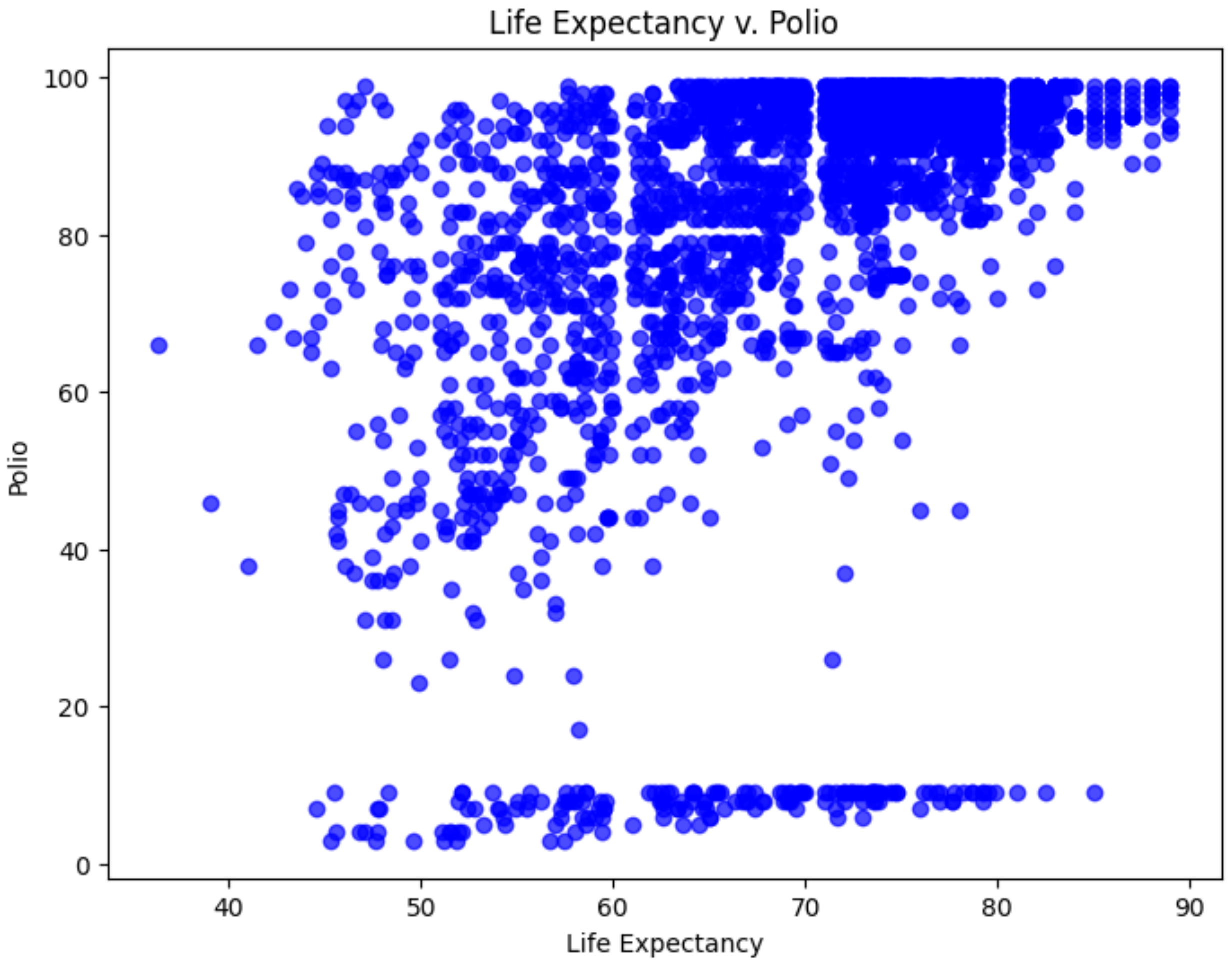

Life Expectancy v. Percentage Expenditure
Percentage Expenditure
Life Expectancy
0
2500
5000
7500
10000
12500
15000
17500
20000
40
50
60
70
80
90

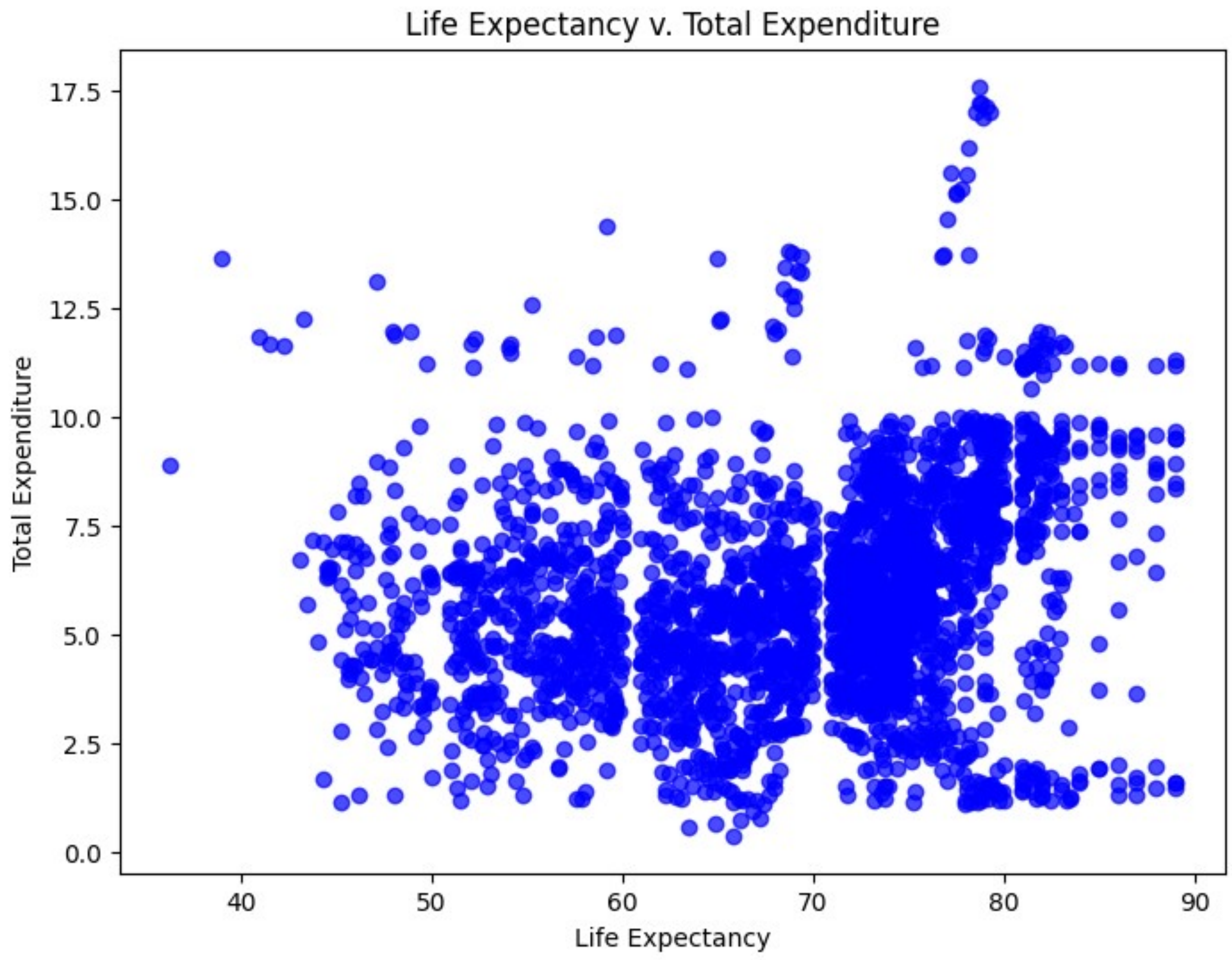

Life Expectancy v. Hepatitis B
Hepatitis B
Life Expectancy
0
20
40
60
80
100
40
50
60
70
80
90

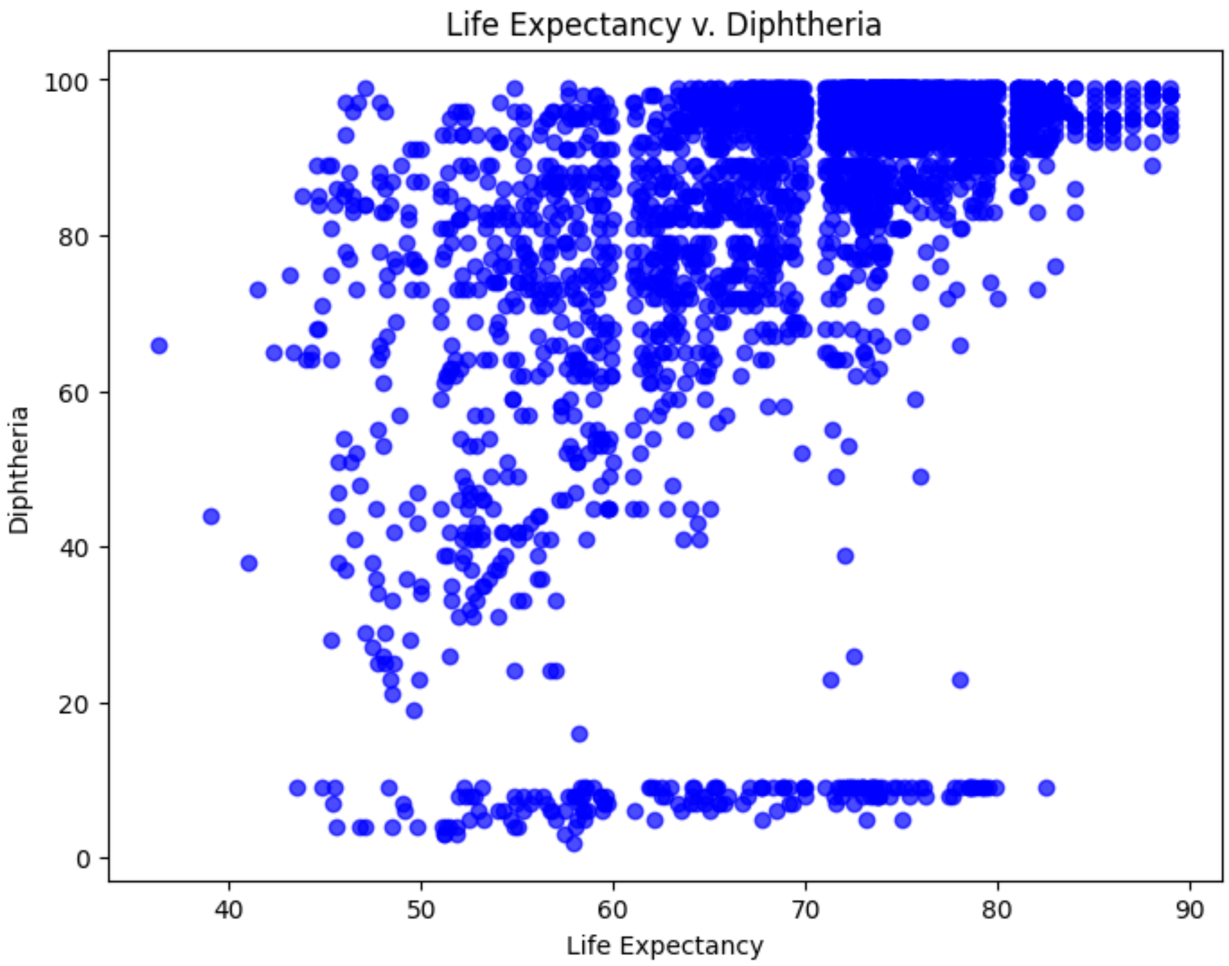

Life Expectancy v. Measles
Measles
Life Expectancy
100
80
60
40
20
40
50
60
70
80
90

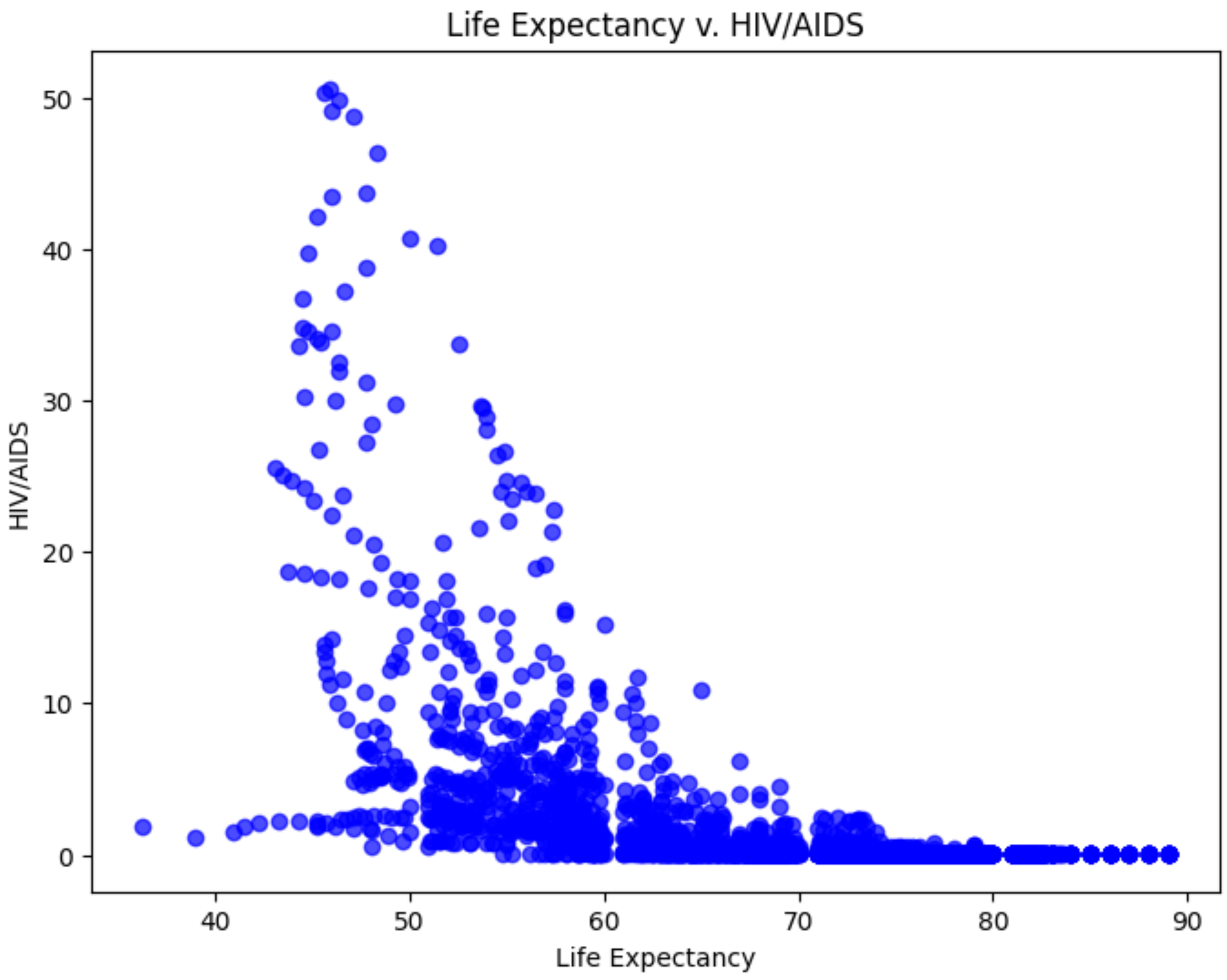

Life Expectancy v. BMI
BMI
20
22
24
26
28
30
32
Life Expectancy
40
50
60
70
80
90

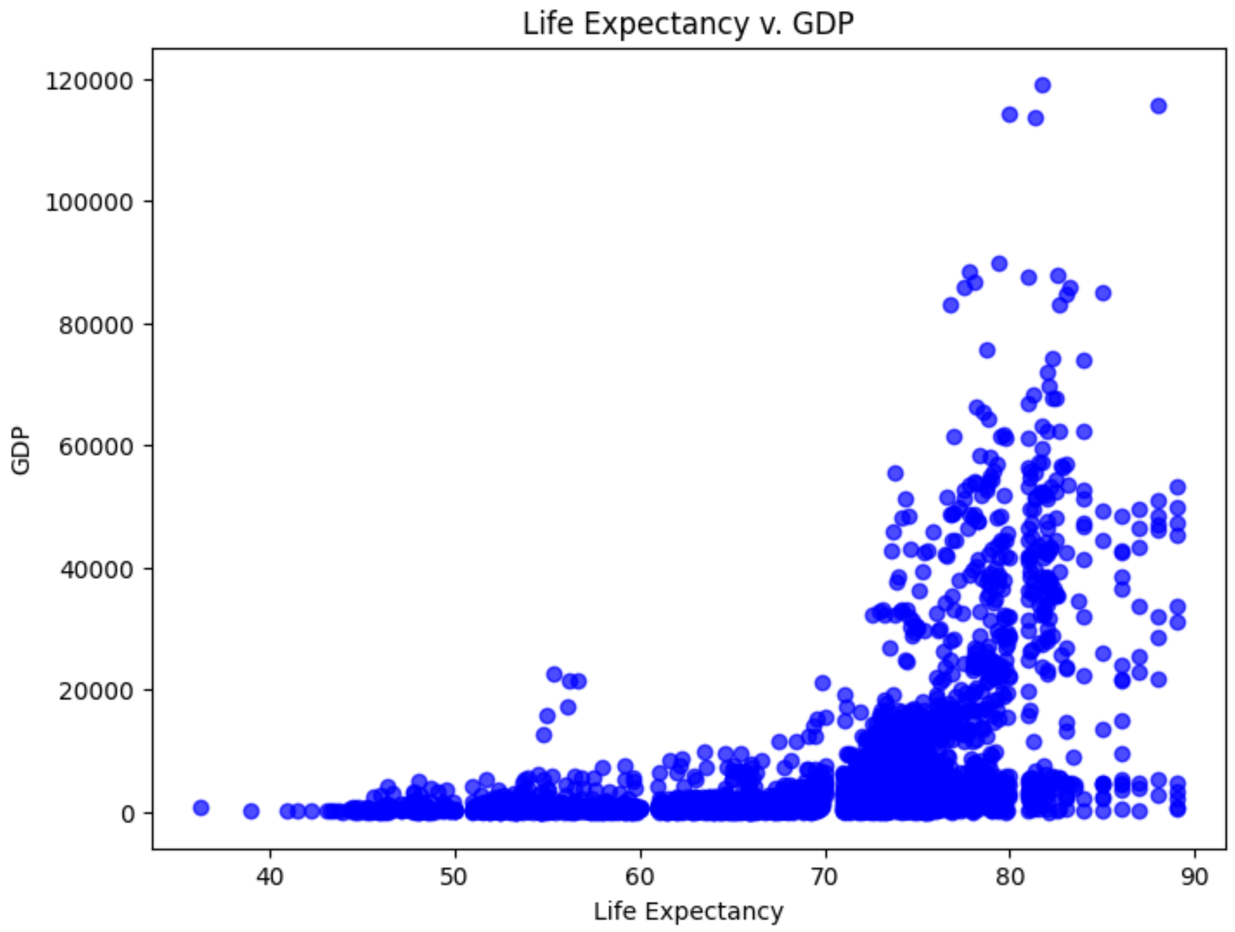

Life Expectancy v. Under-five Deaths
Under-five Deaths
Life Expectancy
0
50
100
150
200
40
50
60
70
80
90

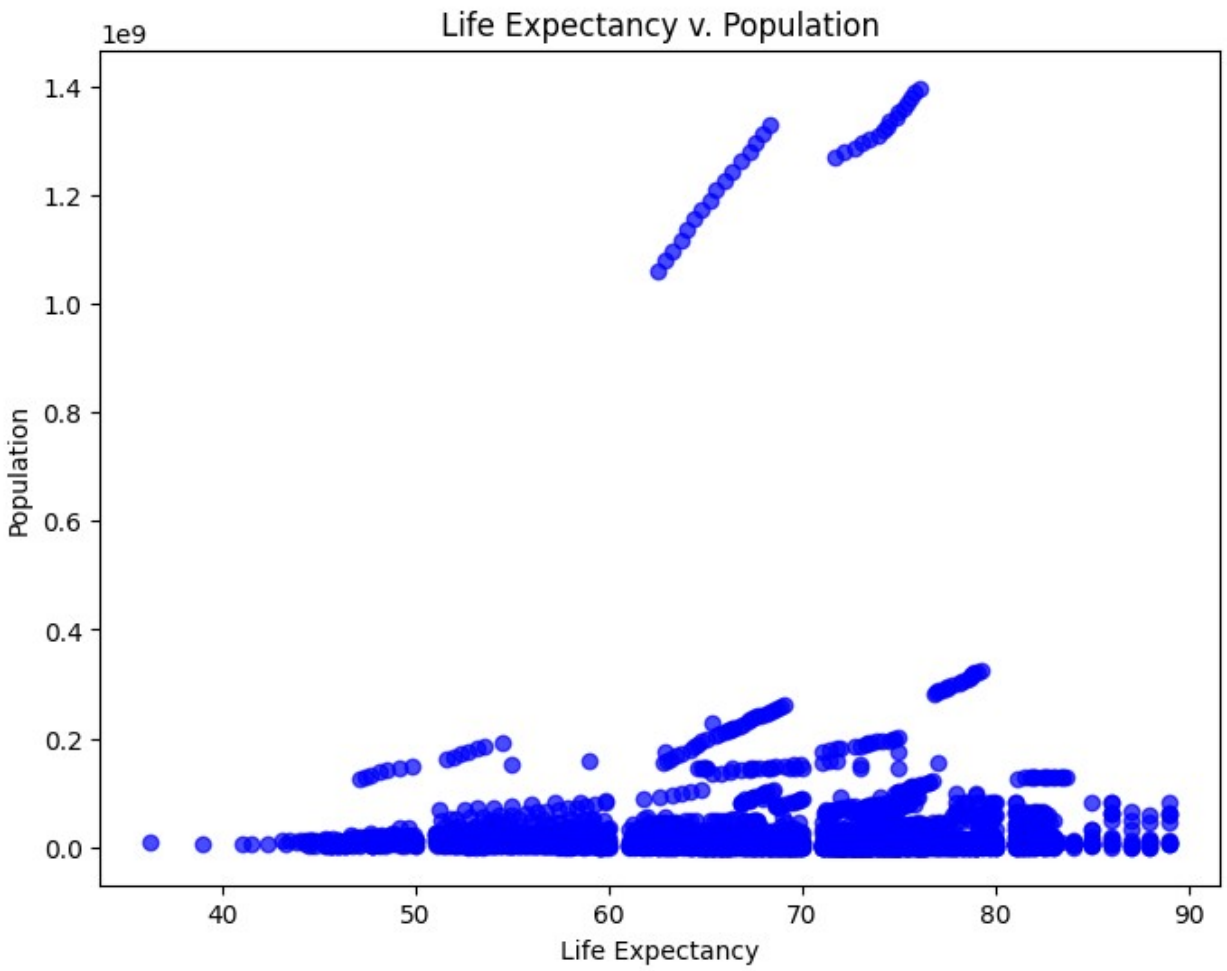

Life Expectancy v. Polio
Polio
Life Expectancy
0
20
40
60
80
100
40
50
60
70
80
90

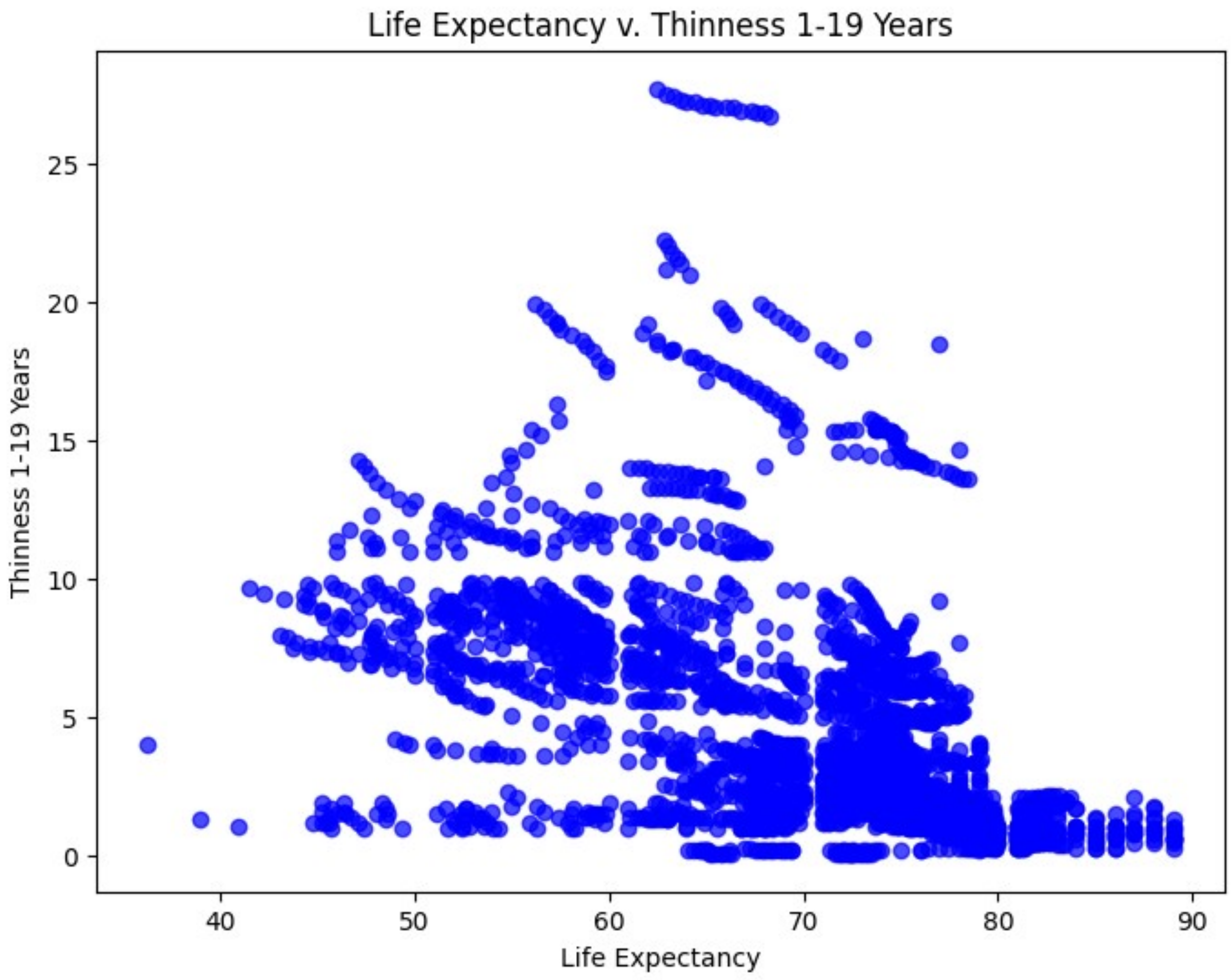

Life Expectancy v. Total Expenditure
Total Expenditure
Life Expectancy
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
40
50
60
70
80
90

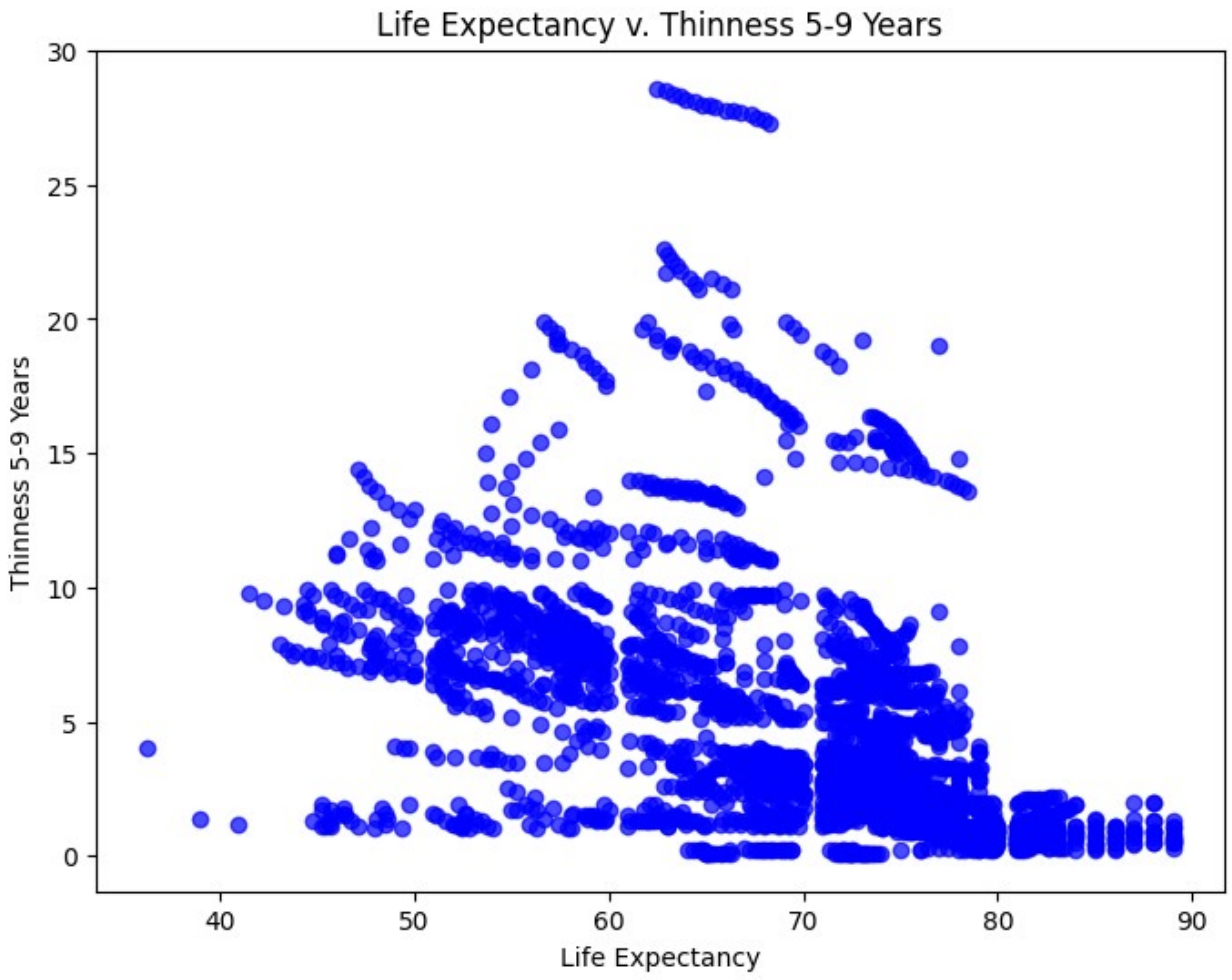

Life Expectancy v. Diphtheria
Diphtheria
Life Expectancy
0
20
40
60
80
100
40
50
60
70
80
90

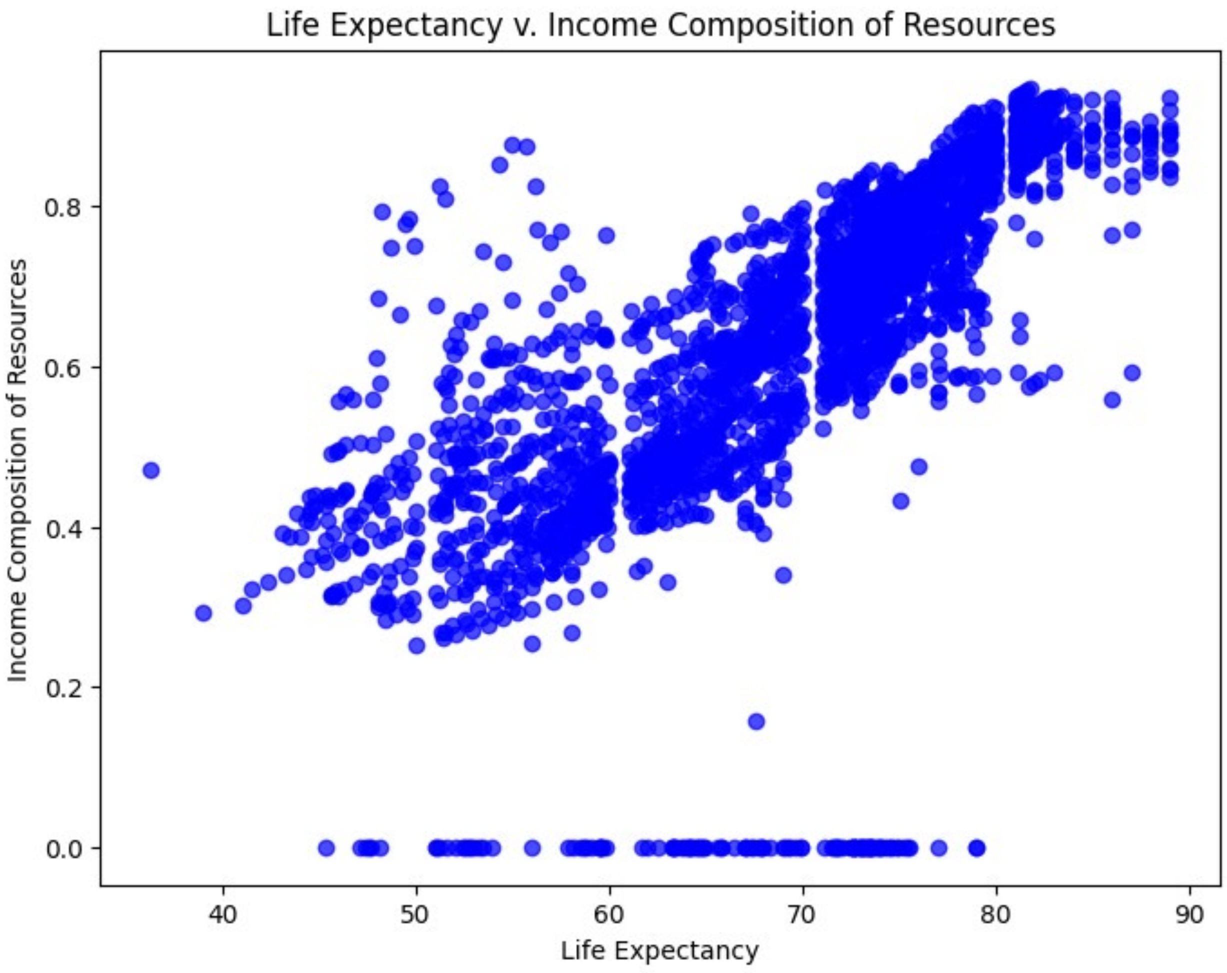

Life Expectancy v. HIV/AIDS
HIV/AIDS
Life Expectancy
0
10
20
30
40
50
40
50
60
70
80
90

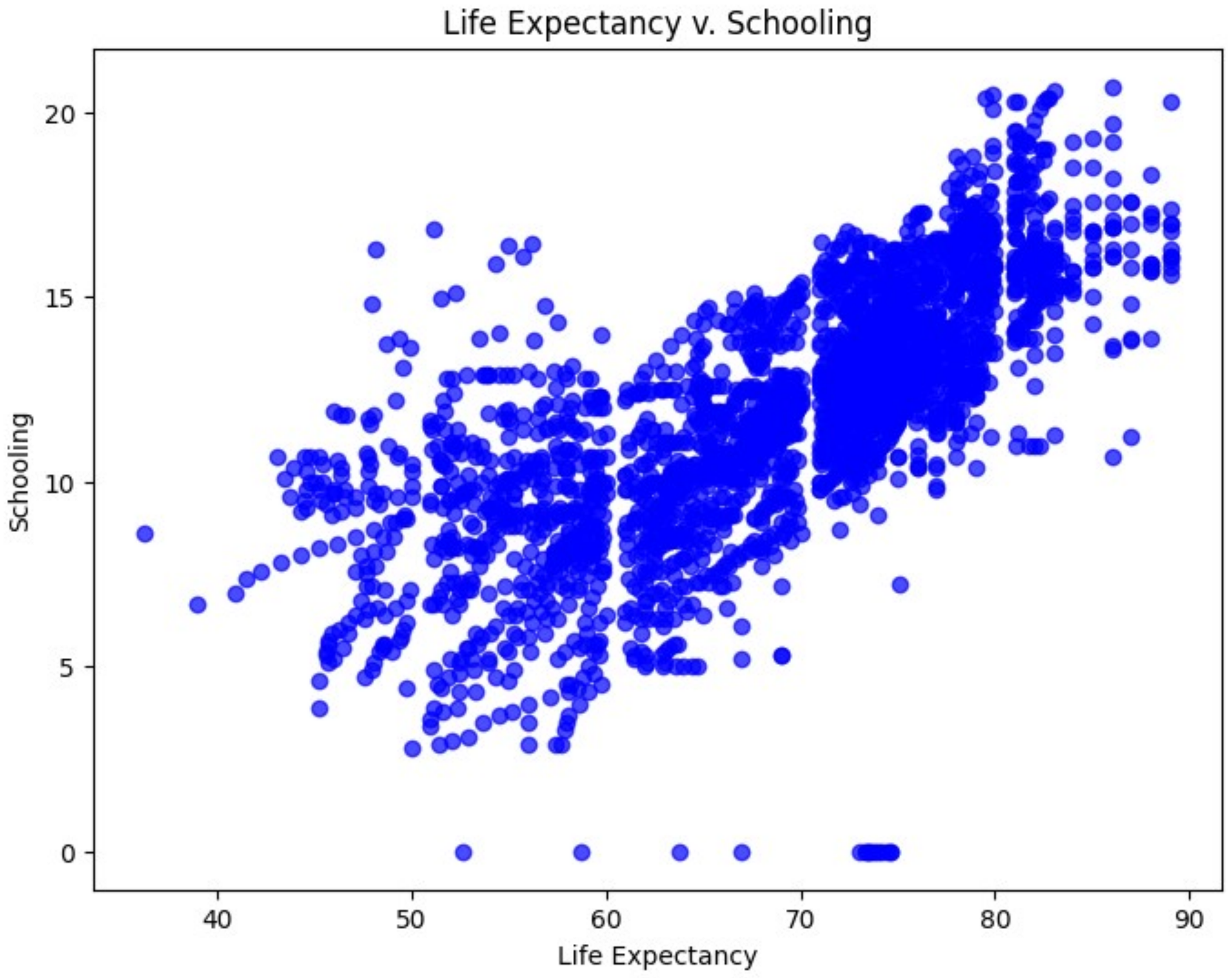

Life Expectancy v. GDP
GDP
Life Expectancy
120000
100000
80000
60000
40000
20000
0
40
50
60
70
80
90

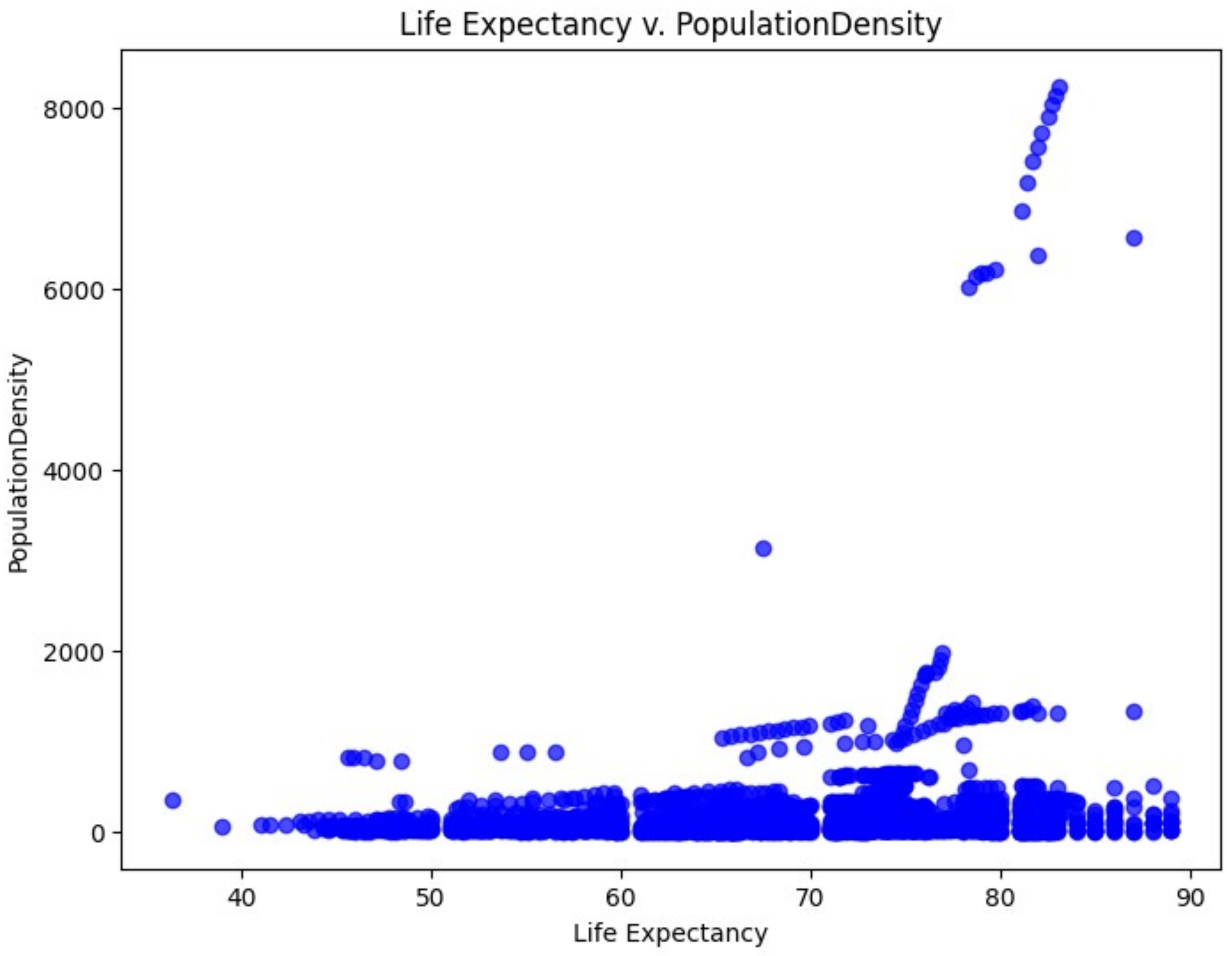

Life Expectancy v. Population
1e9
1.4
1.2
1.0
0.8
0.6
0.4
0.2
0.0
Population
40
50
60
70
80
90
Life Expectancy

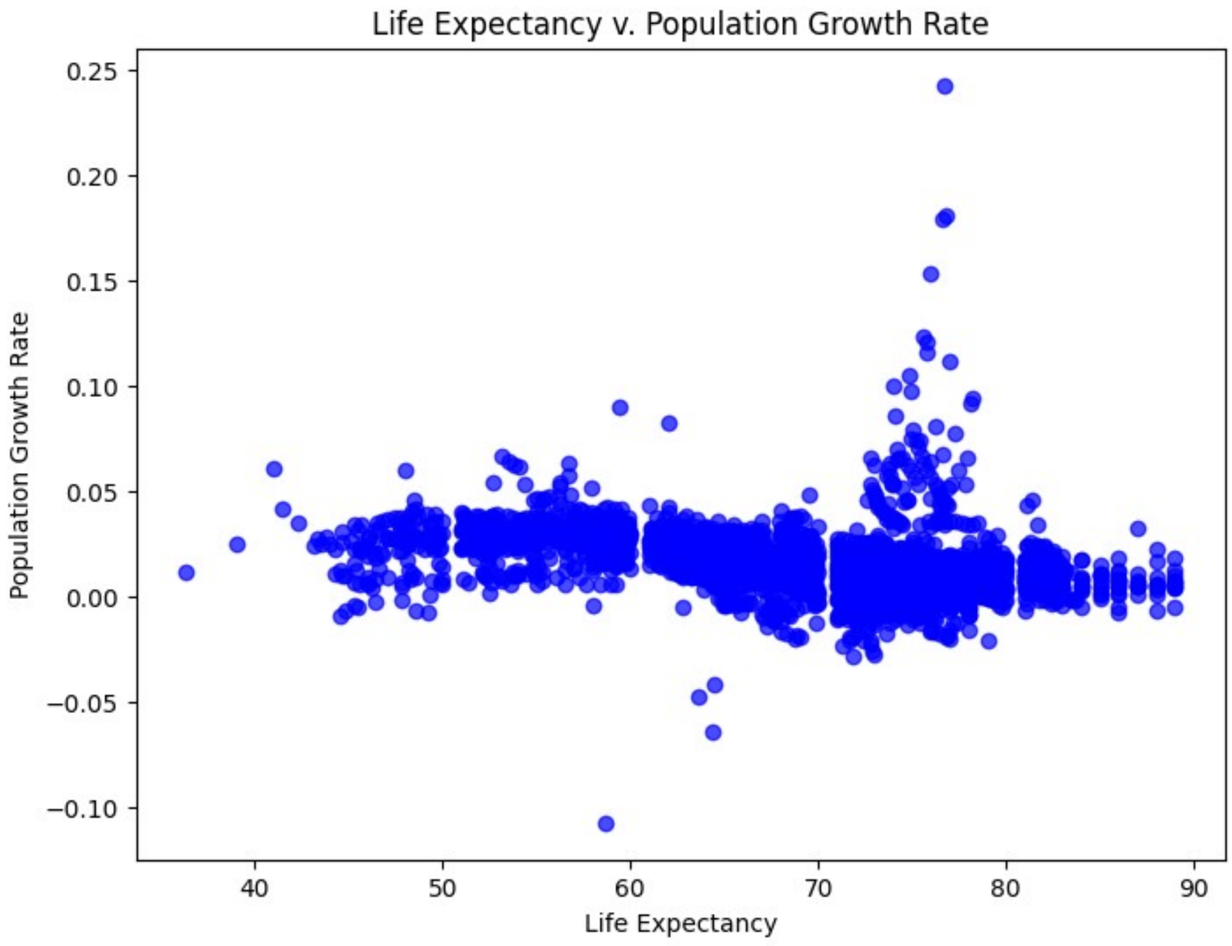

Life Expectancy v. Thinness 1-19 Years
Thinness 1-19 Years
Life Expectancy
0
5
10
15
20
25
40
50
60
70
80
90

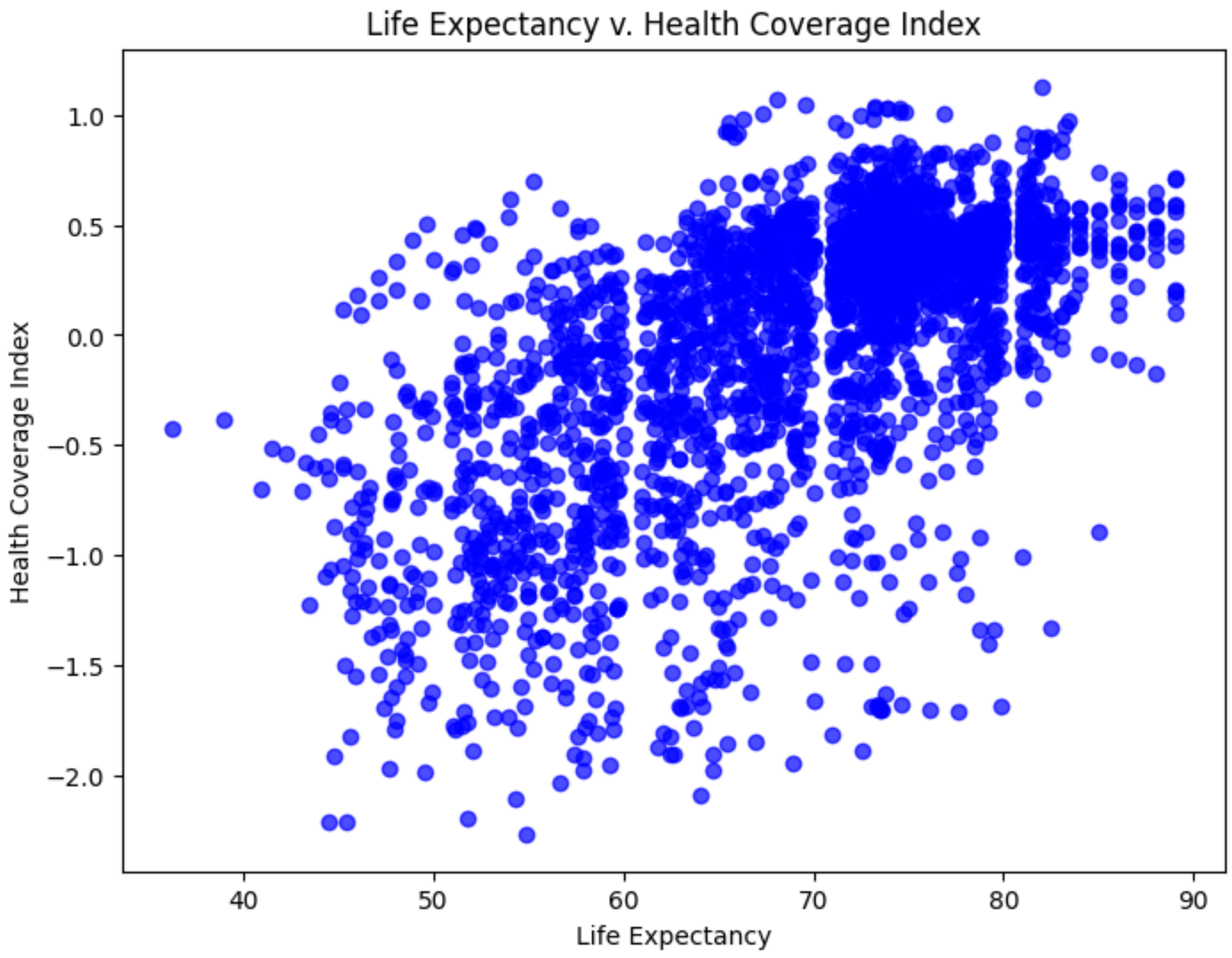

Life Expectancy v. Thinness 5-9 Years
Thinness 5-9 Years
Life Expectancy
0
5
10
15
20
25
30
40
50
60
70
80
90

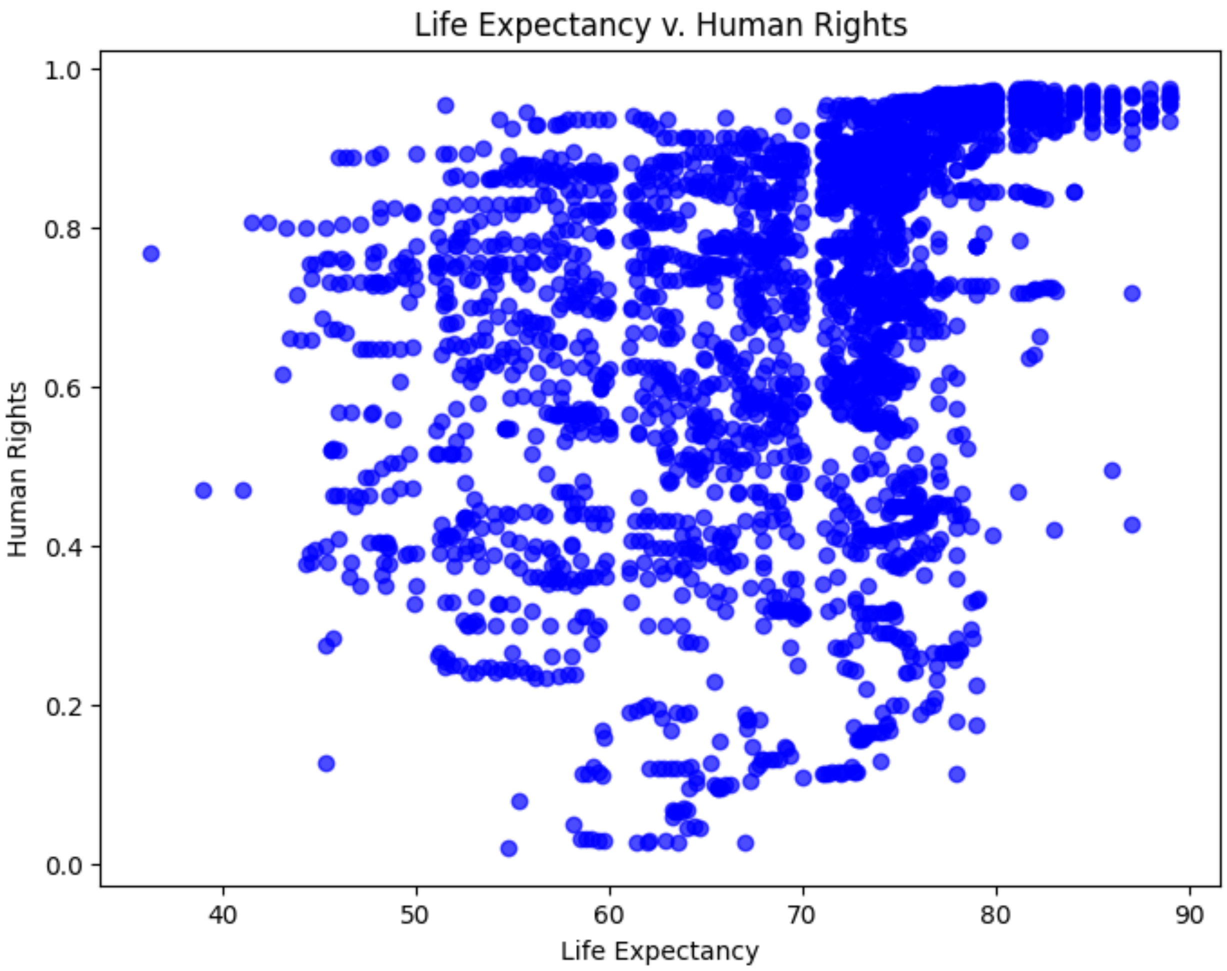

Life Expectancy v. Income Composition of Resources
Income Composition of Resources
0.0
0.2
0.4
0.6
0.8
40
50
60
70
80
90
Life Expectancy

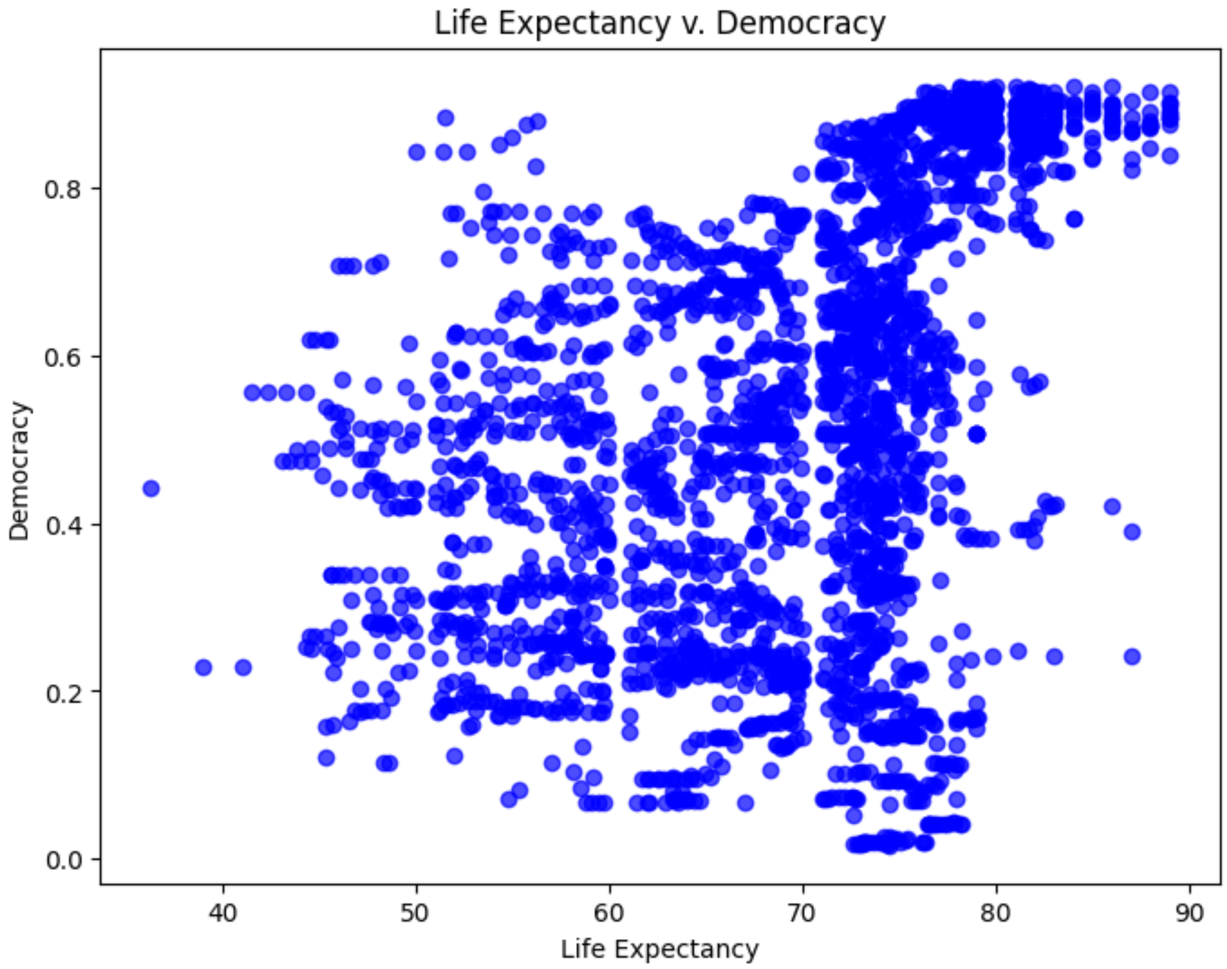

Life Expectancy v. Schooling
Schooling
Life Expectancy
0
5
10
15
20
40
50
60
70
80
90

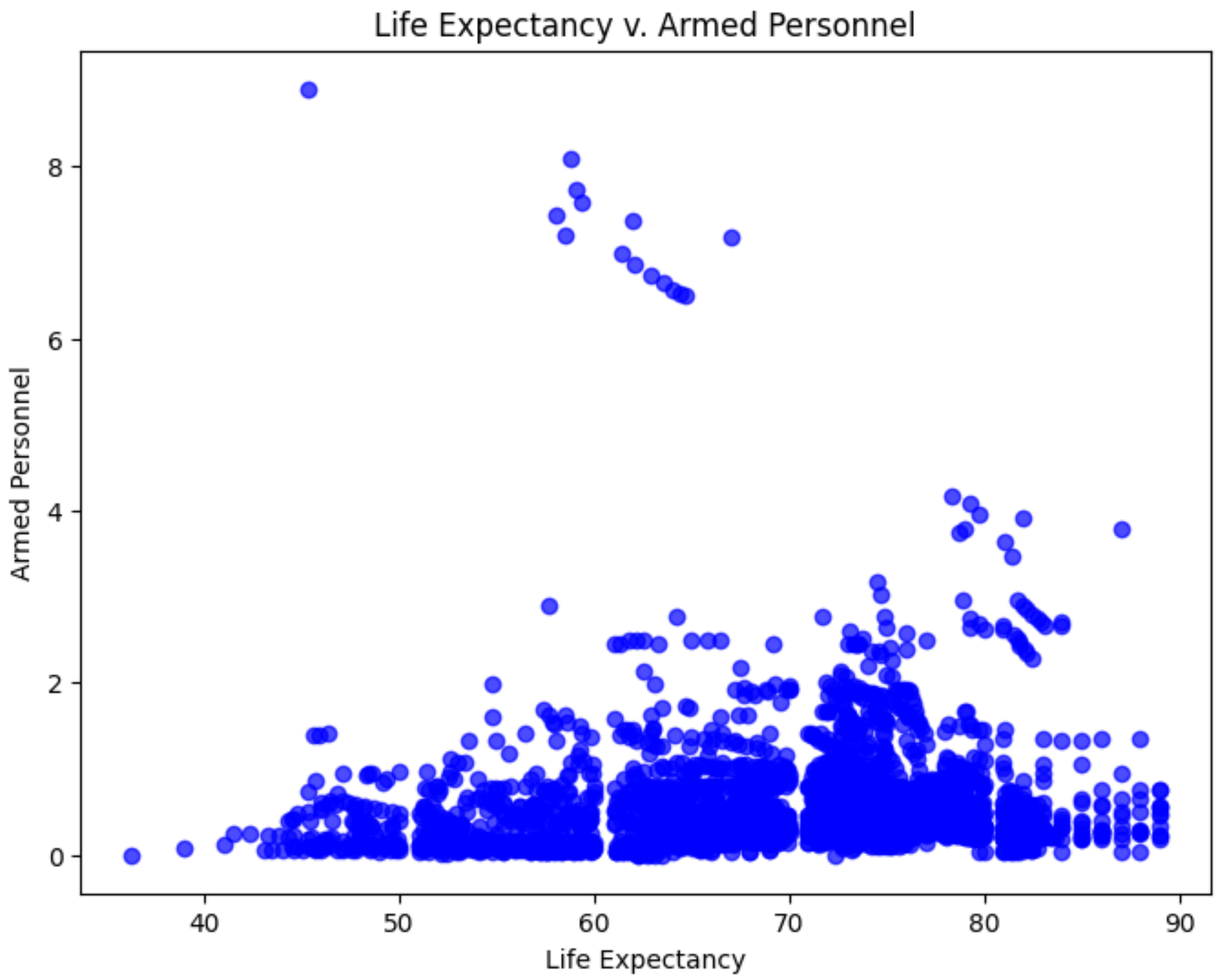

Life Expectancy v. PopulationDensity
PopulationDensity
Life Expectancy
0
2000
4000
6000
8000
40
50
60
70
80
90

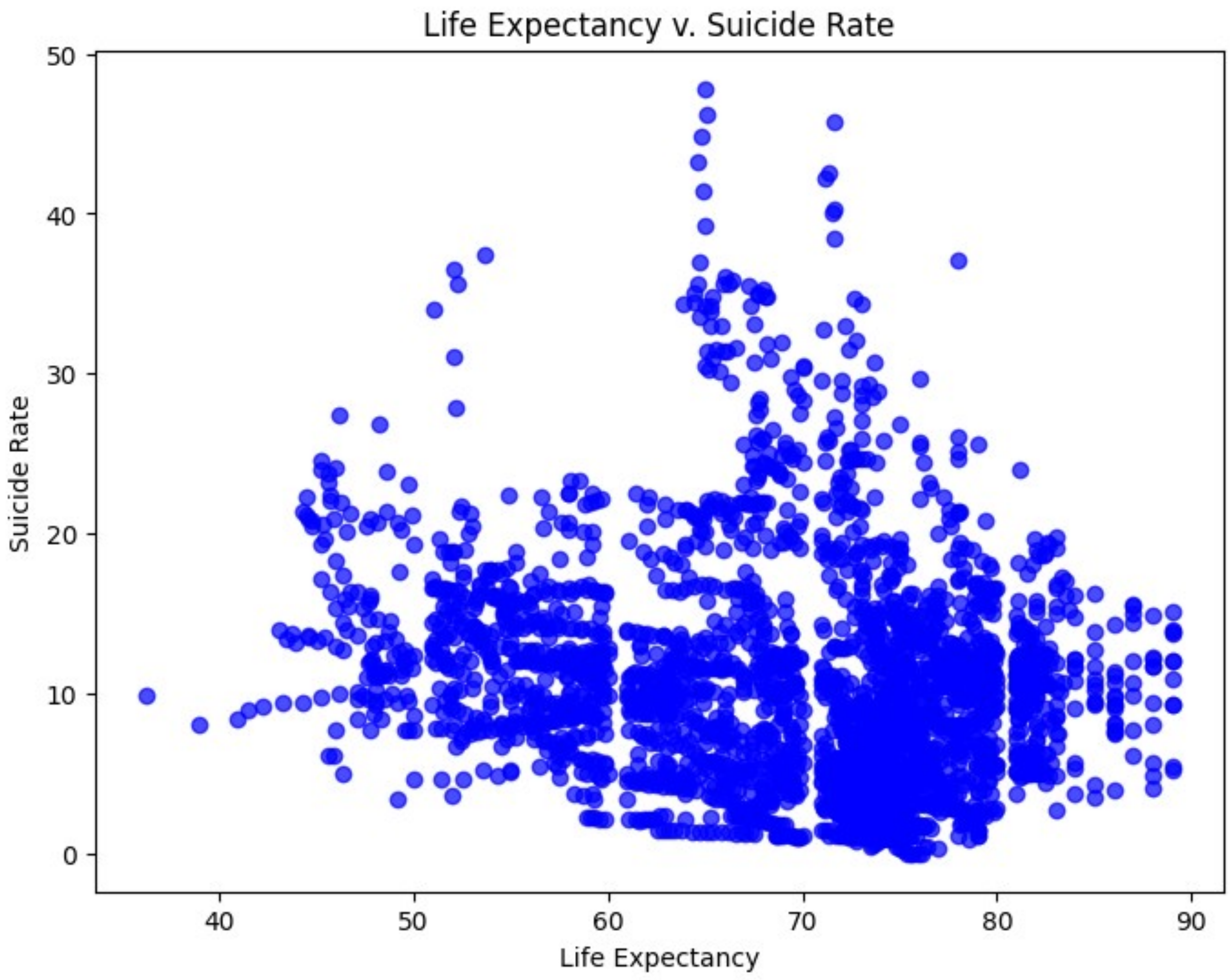

Life Expectancy v. Population Growth Rate
Population Growth Rate
0.25
0.20
0.15
0.10
0.05
0.00
−0.05
−0.10
40
50
60
70
80
90
Life Expectancy

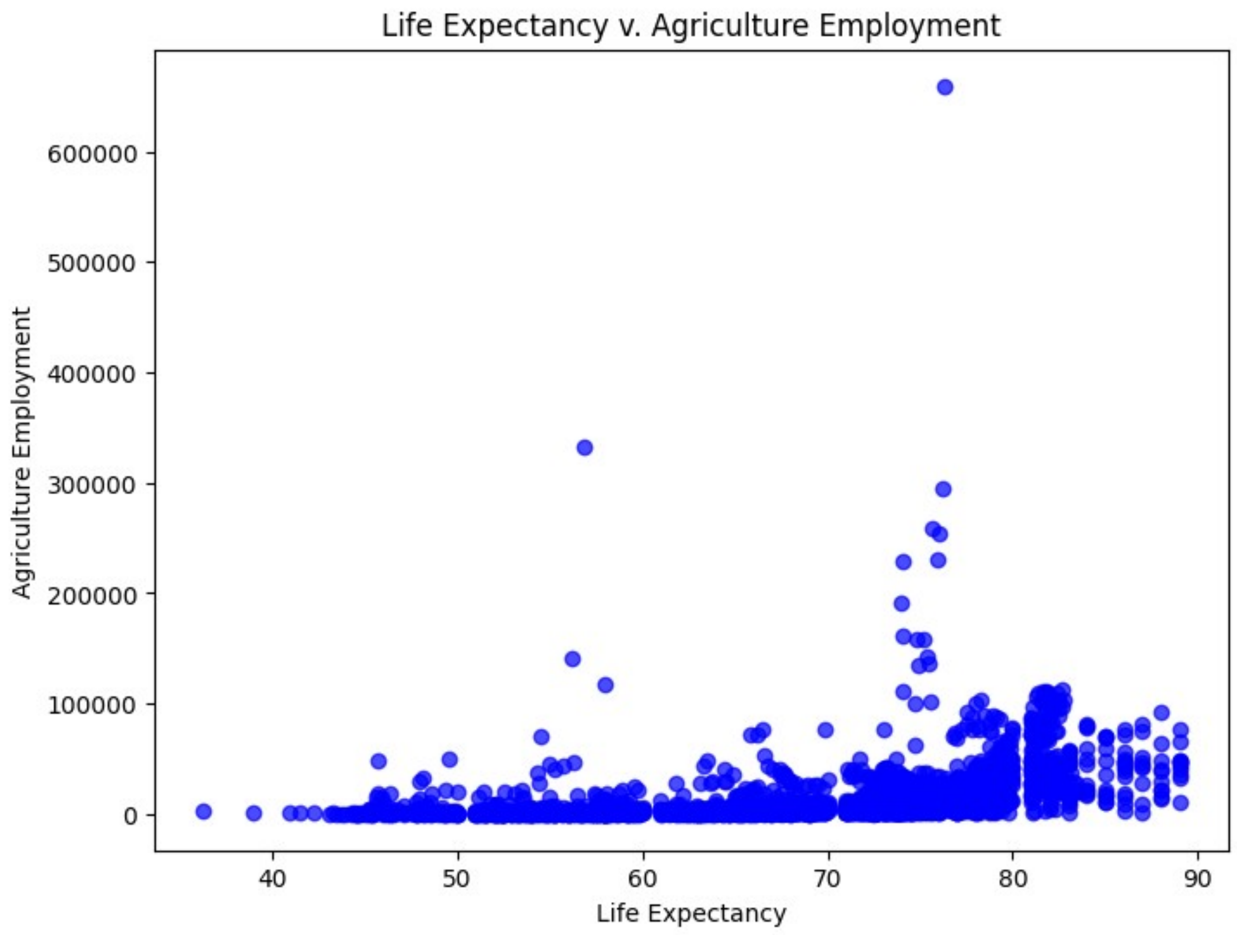

Life Expectancy v. Health Coverage Index
Health Coverage Index
Life Expectancy
1.0
0.5
0.0
−0.5
−1.0
−1.5
−2.0
40
50
60
70
80
90


Life Expectancy v. Human Rights
Human Rights
Life Expectancy
0.0
0.2
0.4
0.6
0.8
1.0
40
50
60
70
80
90


Life Expectancy v. Democracy
Democracy
0.0
0.2
0.4
0.6
0.8
Life Expectancy
40
50
60
70
80
90


Life Expectancy v. Armed Personnel
Armed Personnel
Life Expectancy
0
2
4
6
8
40
50
60
70
80
90

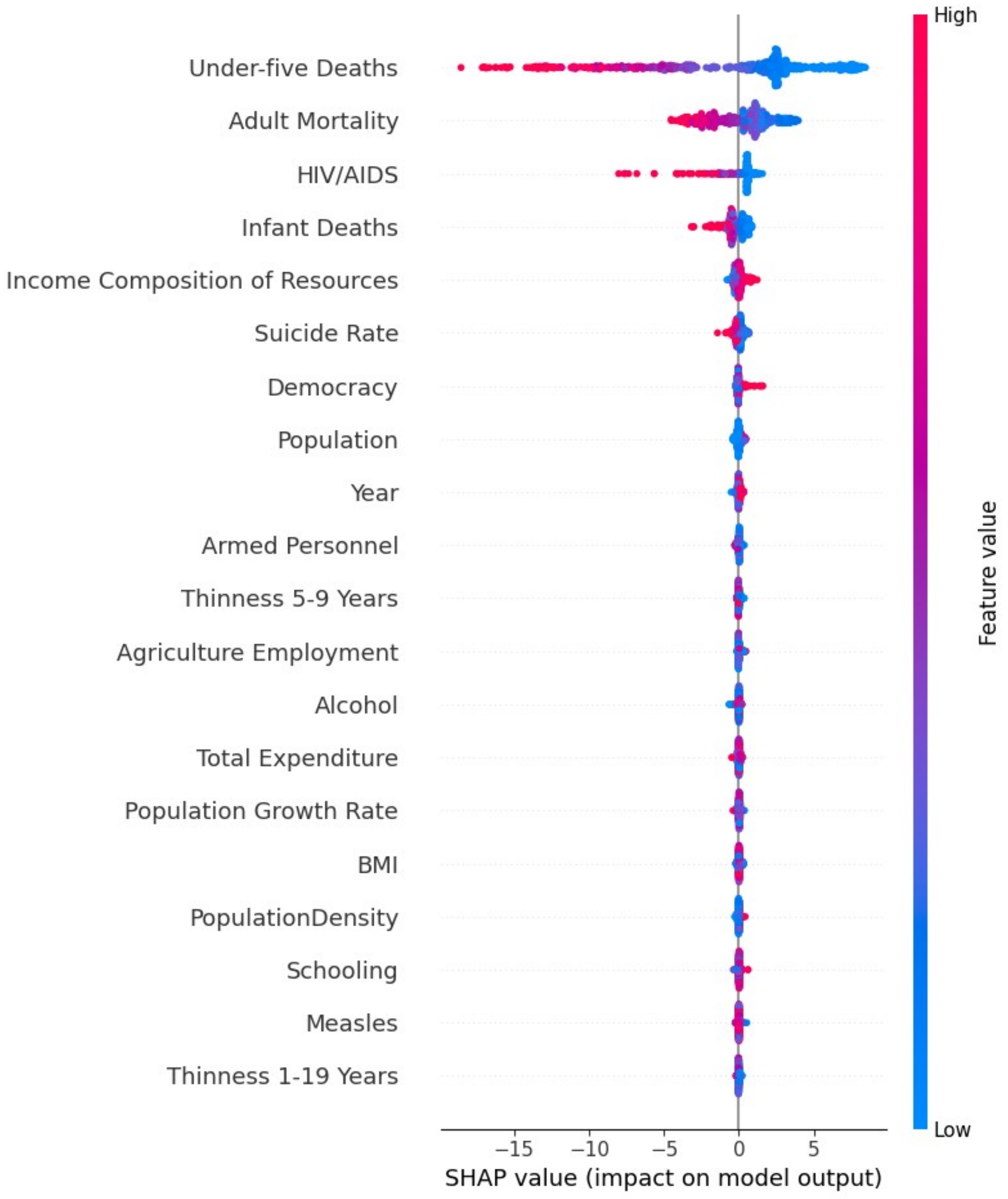

Life Expectancy v. Suicide Rate
Suicide Rate
Life Expectancy
0
10
20
30
40
50
40
50
60
70
80
90



## SHAP

```
df = pd.read_csv('LifeExpectancyClean.csv', index_col=0)
df = df.drop(columns=['Country'])

# Identify target variable
X = df.drop(["Life Expectancy"], axis=1)
y = df["Life Expectancy"]

# Split the data
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)

# Train model
model = RandomForestRegressor()
model.fit(X_train, y_train)

# Explain the model predictions using SHAP
explainer = shap.Explainer(model, X_train)  # TreeExplainer for tree-
based models
shap_values = explainer(X_test)
```

```
# Generate SHAP summary plot
shap.summary_plot(shap_values, X_test, feature_names=X_train.columns)

 97%|=================== | 567/583 [00:25<00:00]
```



## Import libraries and tools

```
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_absolute_error, mean_squared_error,
r2_score
import matplotlib.pyplot as plt
import seaborn as sns
from statsmodels.stats.outliers_influence import
variance_inflation_factor
import statsmodels.api as sm
from sklearn.decomposition import PCA
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import Ridge
```

## Read in data

```
df = pd.read_csv('LifeExpectancyClean.csv', index_col=0)
# df = df[df['Status'] == 0]

cols = ['Alcohol', 'Percentage Expenditure',
       'Diphtheria', 'HIV/AIDS', 'Population', 'PopulationDensity',
       'Income Composition of Resources', 'HighIncome',
       'HealthyWeightBMI', 'Thinness 1-19 Years',
       'Human Rights', 'Suicide Rate', 'Agriculture Employment',
'Schooling']

cols_ext = ['Life Expectancy',
       'Alcohol', 'Percentage Expenditure',
       'Diphtheria', 'HIV/AIDS', 'Population', 'PopulationDensity',
       'Income Composition of Resources', 'HighIncome',
       'HealthyWeightBMI', 'Thinness 1-19 Years',
       'Human Rights', 'Suicide Rate', 'Agriculture Employment',
'Schooling']
```

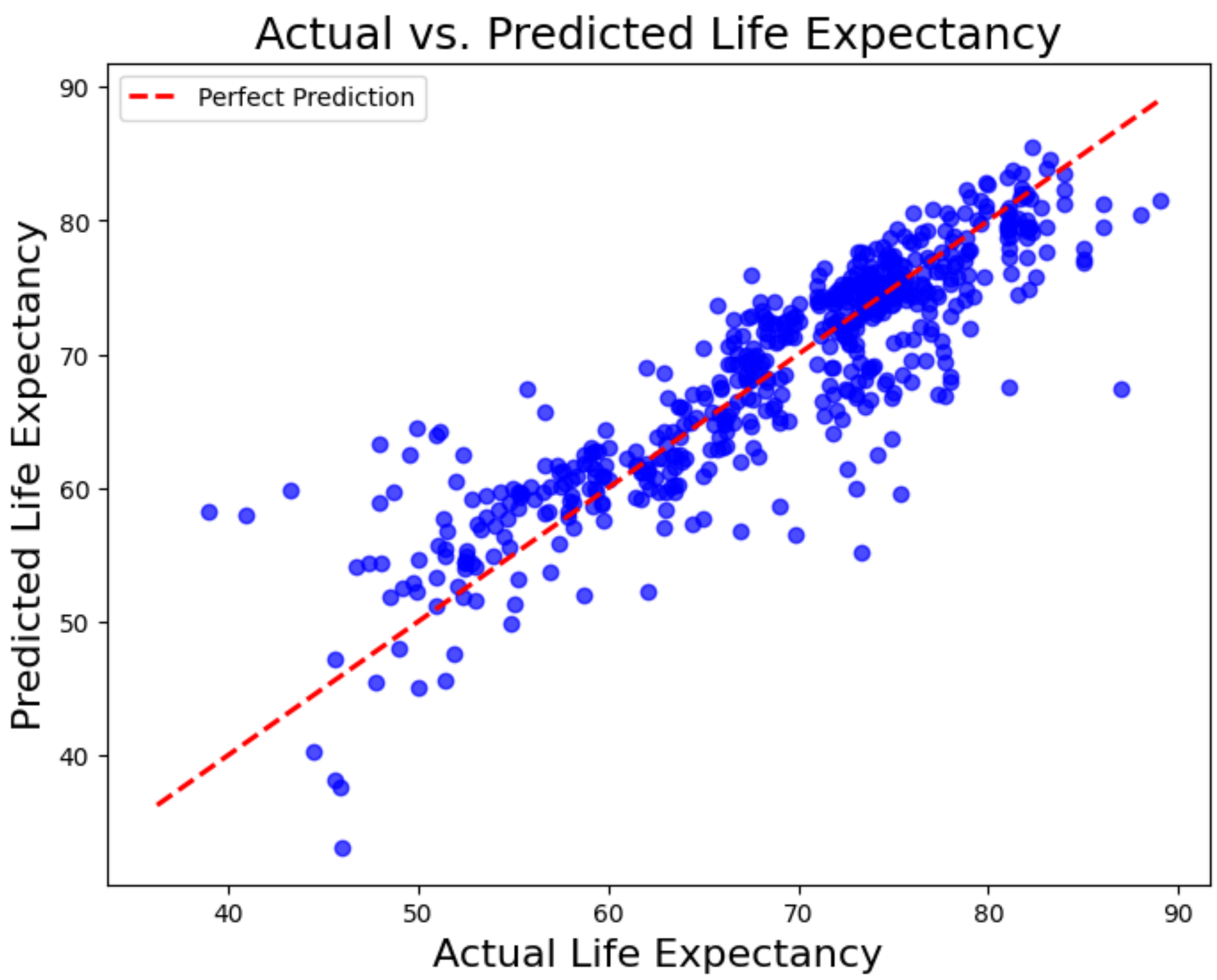
## Multiple Linear Regression Model

```
def run_regression(X, y):
    """ Fits a Ridge regression model, evaluates its performance,
displays key metrics and OLS summary
    statistics, and creates a scatter plot of predicted vs. actual
values.

    Parameters:
        X : Array of independent variables.
        y : Array of the dependent variable.
    Returns:
        None. Prints model performance metrics and OLS summary.
    """

    # Split the data
    X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)

    # Initialize and fit Ridge regression model
    ridge_model = Ridge(alpha=1.0)  # You can adjust the alpha
(regularization strength)
    ridge_model.fit(X_train, y_train)

    # Predictions
    y_pred = ridge_model.predict(X_test)

    # Evaluate the model
    mae = mean_absolute_error(y_test, y_pred)
    mse = mean_squared_error(y_test, y_pred)
    r2 = r2_score(y_test, y_pred)

    print(f'MAE: {mae}')
    print(f'MSE: {mse}')
    print(f'RMSE: {np.sqrt(mse)}')
    print(f'R^2: {r2}')

    # Coefficients and intercept
    print("Intercept:", ridge_model.intercept_)
    print("Coefficients:", ridge_model.coef_)

    # Approximate p-values using statsmodels OLS
    ols_model = sm.OLS(y_train, X_train).fit()
    print(ols_model.summary())  # Will show coefficients, p-values,
and more

    # Scatter plot of actual vs. predicted values
    plt.figure(figsize=(8, 6))
    plt.scatter(y_test, y_pred, color='blue', alpha=0.7)
    plt.plot([y.min(), y.max()], [y.min(), y.max()], 'r--', lw=2,
label='Perfect Prediction')  # Diagonal line
    plt.xlabel('Actual Life Expectancy', fontsize=16)
    plt.ylabel('Predicted Life Expectancy', fontsize=16)
    plt.title('Actual vs. Predicted Life Expectancy', fontsize=18)
    plt.legend()
    plt.show()

# Prepare the data
X = df[cols]  # Independent variables
y = df['Life Expectancy']  # Dependent variable
```

```
# Add constant to the independent variables (intercept term)
X = sm.add_constant(X, has_constant='add')

run_regression(X, y)
```

```
/opt/homebrew/lib/python3.10/site-packages/sklearn/linear_model/
_ridge.py:216: LinAlgWarning: Ill-conditioned matrix (rcond=2.36694e-
20): result may not be accurate.
  return linalg.solve(A, Xy, assume_a="pos", overwrite_a=True).T

MAE: 3.1054546843671975
MSE: 18.952501388896334
RMSE: 4.3534470697249015
R^2: 0.7826229288375897
Intercept: 54.99665966279319
Coefficients: [ 0.00000000e+00  5.75096645e-02  2.42908311e-04
5.31951431e-02
 -6.46893141e-01  4.93424380e-09  1.44485344e-03  6.26262327e+00
  1.72102807e+00 -4.67156028e+00 -5.31598821e-02  2.41652161e+00
 -1.55709579e-01  1.00043000e-05  6.47107458e-01]
                            OLS Regression Results

======================================================================
========
Dep. Variable:        Life Expectancy   R-squared:
0.806
Model:                            OLS   Adj. R-squared:
0.805
Method:                 Least Squares   F-statistic:
686.2
Date:                Mon, 09 Dec 2024   Prob (F-statistic):
0.00
Time:                        17:53:55   Log-Likelihood:
-6641.9
No. Observations:                2329   AIC:
1.331e+04
Df Residuals:                    2314   BIC:
1.340e+04
Df Model:                          14

Covariance Type:            nonrobust

======================================================================
=============================
                                      coef    std err          t
P>|t|      [0.025      0.975]
----------------------------------------------------------------------
-----------------------------
const                              54.9812      0.613     89.732
0.000      53.780      56.183
```

```
Alcohol                                  0.0565      0.030      1.916
0.055      -0.001       0.114
Percentage Expenditure                   0.0002   6.06e-05      3.979
0.000       0.000       0.000
Diphtheria                               0.0531      0.004     12.788
0.000       0.045       0.061
HIV/AIDS                                -0.6464      0.018    -34.969
0.000      -0.683      -0.610
Population                            4.923e-09   6.84e-10      7.195
0.000    3.58e-09    6.26e-09
PopulationDensity                        0.0014      0.000      9.212
0.000       0.001       0.002
Income Composition of Resources          6.4615      0.762      8.480
0.000       4.967       7.956
HighIncome                               1.7270      0.337      5.126
0.000       1.066       2.388
HealthyWeightBMI                        -4.6853      0.262    -17.880
0.000      -5.199      -4.171
Thinness 1-19 Years                     -0.0519      0.027     -1.927
0.054      -0.105       0.001
Human Rights                             2.4475      0.467      5.236
0.000       1.531       3.364
Suicide Rate                            -0.1552      0.013    -11.747
0.000      -0.181      -0.129
Agriculture Employment                9.986e-06   3.71e-06      2.694
0.007    2.72e-06    1.73e-05
Schooling                                0.6369      0.054     11.847
0.000       0.531       0.742
======================================================================
========
Omnibus:                     141.498   Durbin-Watson:
2.041
Prob(Omnibus):                 0.000   Jarque-Bera (JB):
608.018
Skew:                          0.023   Prob(JB):
9.35e-133
Kurtosis:                      5.503   Cond. No.
1.24e+09
======================================================================
========

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is
correctly specified.
[2] The condition number is large, 1.24e+09. This might indicate that
there are
strong multicollinearity or other numerical problems.
```



## Test Multicollinearity and Correlation

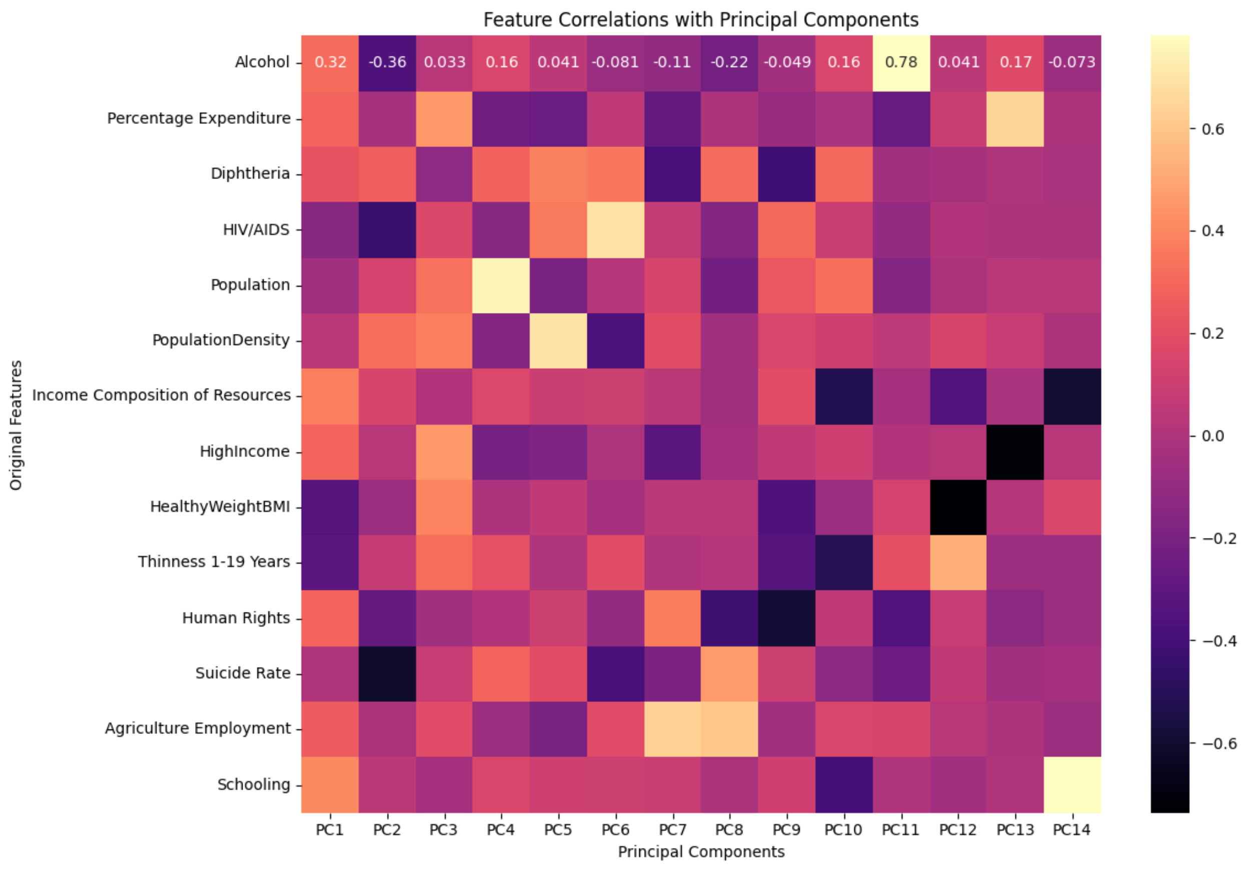
### VIF

```
def calculate_vif(X):
    """ Calculates the Variance Inflation Factor (VIF) for each
feature in the dataset
    to detect multicollinearity among independent variables.

    Parameters:
        X : X : Array of independent variables.

    Returns:
        None. Prints a DataFrame showing each feature's VIF value.
    """
    # Calculate VIF for each independent variable
    vif_data = pd.DataFrame()
    vif_data['Variable'] = X.columns
    vif_data['VIF'] = [variance_inflation_factor(X.values, i) for i in
range(X.shape[1])]

    print(vif_data)

calculate_vif(X)
```

```
                              Variable        VIF
0                                const  50.664338
1                              Alcohol   1.844292
2               Percentage Expenditure   1.912356
3                           Diphtheria   1.253468
4                             HIV/AIDS   1.140730
5                           Population   1.115808
6                    PopulationDensity   1.091584
7    Income Composition of Resources   2.945715
8                           HighIncome   1.991131
9                     HealthyWeightBMI   2.165912
10                Thinness 1-19 Years   1.841865
11                        Human Rights   1.507866
12                         Suicide Rate   1.149728
13             Agriculture Employment   1.289851
14                           Schooling   3.723695
```

## PCA

```
def calculate_pca():
    """ Performs Principal Component Analysis (PCA) on a set of
features, visualizes feature correlations
    with principal components, and prints the explained variance ratio
for each component.

    Returns:
        None. Displays a heatmap of feature-PC correlations and prints
the explained variance for each principal component.
    """
    X_PCA = df[cols]

    # Standardize the data
    scaler = StandardScaler()
    X_scaled = scaler.fit_transform(X_PCA)

    # Perform PCA
    pca = PCA()
    X_pca = pca.fit_transform(X_scaled)

    # Get the loadings
    loadings = pd.DataFrame(
        pca.components_.T,
        columns=[f"PC{i+1}" for i in range(X_pca.shape[1])],
        index=cols
    )

    # Calculate the explained variance ratio for each PC
    explained_variance = pca.explained_variance_ratio_

    # Plot the correlation matrix
    plt.figure(figsize=(12, 8))
    sns.heatmap(loadings, annot=True, cmap="magma", cbar=True)
    plt.title("Feature Correlations with Principal Components")
    plt.xlabel("Principal Components")
    plt.ylabel("Original Features")
    plt.tight_layout()
    plt.show()

    # Print explained variance ratios
    for i, var in enumerate(explained_variance, 1):
        print(f"PC{i} explains {var:.2%} of the variance.")

calculate_pca()
```



```
PC1 explains 32.24% of the variance.
PC2 explains 9.89% of the variance.
PC3 explains 9.59% of the variance.
```

```
PC4 explains 7.83% of the variance.
PC5 explains 6.82% of the variance.
PC6 explains 5.74% of the variance.
PC7 explains 5.60% of the variance.
PC8 explains 5.09% of the variance.
PC9 explains 4.28% of the variance.
PC10 explains 3.84% of the variance.
PC11 explains 3.03% of the variance.
PC12 explains 2.42% of the variance.
PC13 explains 2.31% of the variance.
PC14 explains 1.30% of the variance.
```

## Correlation Matrix

```
def calculate_corr_matrix():
    """Computes and visualizes the correlation matrix for a set of
features.

    Returns:
        None. Displays the correlation matrix and the heatmap
visualization.
    """
    df_corr = df[cols_ext]

    # Compute the correlation matrix
    corr_matrix = df_corr.corr()

    # Display the correlation matrix
    display(corr_matrix)

    # Visualize the correlation matrix using a heatmap
    plt.figure(figsize=(12, 8))
    sns.heatmap(corr_matrix, annot=True, cmap='Blues', vmin=-1,
vmax=1, fmt='.2f')
    plt.title("Correlation Matrix")
    plt.show()

calculate_corr_matrix()
```

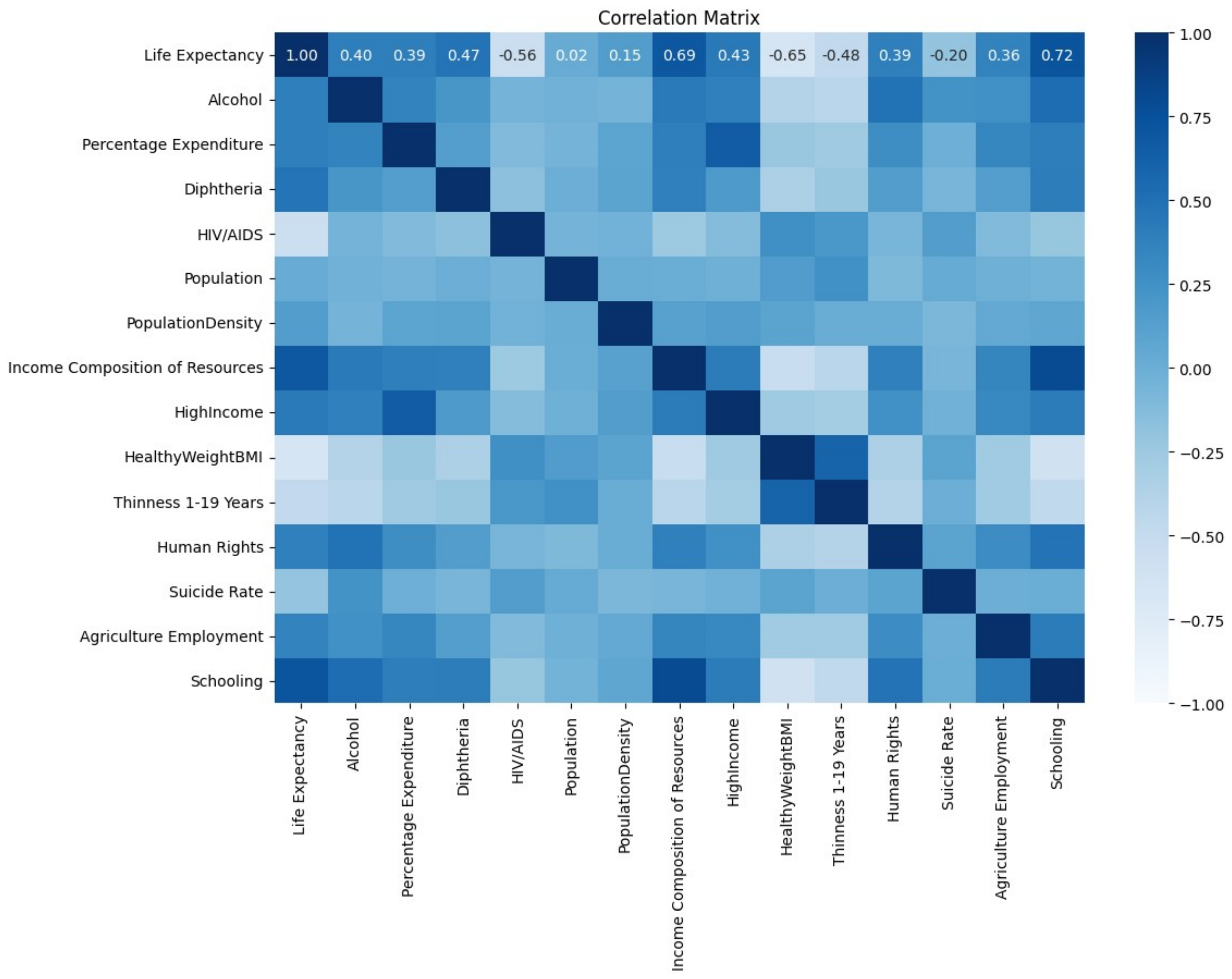
```
                                 Life Expectancy   Alcohol  \
Life Expectancy                         1.000000  0.395782
Alcohol                                 0.395782  1.000000
Percentage Expenditure                  0.392870  0.361026
Diphtheria                              0.470657  0.218709
HIV/AIDS                               -0.557549 -0.048415
Population                              0.020565 -0.024166
PopulationDensity                       0.149266 -0.047897
Income Composition of Resources         0.690552  0.436655
HighIncome                              0.434693  0.389933
HealthyWeightBMI                       -0.652780 -0.389465
```

```
Thinness 1-19 Years                                -0.476540 -0.418701
Human Rights                                        0.386159  0.489927
Suicide Rate                                       -0.202044  0.234407
Agriculture Employment                              0.361705  0.255498
Schooling                                           0.717427  0.526801

                                 Percentage Expenditure  Diphtheria
HIV/AIDS  \
Life Expectancy                                0.392870    0.470657 -
0.557549
Alcohol                                        0.361026    0.218709 -
0.048415
Percentage Expenditure                         1.000000    0.147277 -
0.106917
Diphtheria                                     0.147277    1.000000 -
0.160751
HIV/AIDS                                      -0.106917   -0.160751
1.000000
Population                                    -0.046905   -0.000952 -
0.049091
PopulationDensity                              0.085930    0.090455 -
0.036280
Income Composition of Resources                0.396316    0.389195 -
0.243533
HighIncome                                     0.662711    0.178091 -
0.131954
HealthyWeightBMI                              -0.225208   -0.336773
0.265416
Thinness 1-19 Years                           -0.256365   -0.224631
0.203049
Human Rights                                   0.281092    0.153014 -
0.067349
Suicide Rate                                  -0.008401   -0.065066
0.154562
Agriculture Employment                         0.342013    0.142392 -
0.105067
Schooling                                      0.404376    0.409614 -
0.217555

                                 Population  PopulationDensity  \
Life Expectancy                    0.020565           0.149266
Alcohol                           -0.024166          -0.047897
Percentage Expenditure            -0.046905           0.085930
Diphtheria                        -0.000952           0.090455
HIV/AIDS                          -0.049091          -0.036280
Population                         1.000000           0.010537
PopulationDensity                  0.010537           1.000000
Income Composition of Resources    0.004605           0.116358
HighIncome                        -0.022556           0.151868
```

```
HealthyWeightBMI                      0.156914           0.086919
Thinness 1-19 Years                   0.256959           0.012518
Human Rights                         -0.099423           0.008503
Suicide Rate                          0.028308          -0.085217
Agriculture Employment               -0.022485           0.044078
Schooling                            -0.043605           0.077303

                                 Income Composition of Resources
HighIncome  \
Life Expectancy                                         0.690552
0.434693
Alcohol                                                 0.436655
0.389933
Percentage Expenditure                                  0.396316
0.662711
Diphtheria                                              0.389195
0.178091
HIV/AIDS                                               -0.243533   -
0.131954
Population                                              0.004605   -
0.022556
PopulationDensity                                       0.116358
0.151868
Income Composition of Resources                         1.000000
0.420543
HighIncome                                              0.420543
1.000000
HealthyWeightBMI                                       -0.530666   -
0.253594
Thinness 1-19 Years                                    -0.417869   -
0.288875
Human Rights                                            0.384839
0.250808
Suicide Rate                                           -0.067840   -
0.026974
Agriculture Employment                                  0.350654
0.324867
Schooling                                               0.794735
0.416127

                                 HealthyWeightBMI  Thinness 1-19 Years
\
Life Expectancy                         -0.652780            -0.476540

Alcohol                                 -0.389465            -0.418701

Percentage Expenditure                  -0.225208            -0.256365

Diphtheria                              -0.336773            -0.224631
```

```
HIV/AIDS                                   0.265416             0.203049

Population                                 0.156914             0.256959

PopulationDensity                          0.086919             0.012518

Income Composition of Resources           -0.530666            -0.417869

HighIncome                                -0.253594            -0.288875

HealthyWeightBMI                           1.000000             0.599444

Thinness 1-19 Years                        0.599444             1.000000

Human Rights                              -0.338707            -0.388758

Suicide Rate                               0.088899            -0.005964

Agriculture Employment                    -0.263869            -0.271680

Schooling                                 -0.604840            -0.468108


                                 Human Rights  Suicide Rate  \
Life Expectancy                      0.386159     -0.202044
Alcohol                              0.489927      0.234407
Percentage Expenditure               0.281092     -0.008401
Diphtheria                           0.153014     -0.065066
HIV/AIDS                            -0.067349      0.154562
Population                          -0.099423      0.028308
PopulationDensity                    0.008503     -0.085217
Income Composition of Resources      0.384839     -0.067840
HighIncome                           0.250808     -0.026974
HealthyWeightBMI                    -0.338707      0.088899
Thinness 1-19 Years                 -0.388758     -0.005964
Human Rights                         1.000000      0.089517
Suicide Rate                         0.089517      1.000000
Agriculture Employment               0.284028     -0.003705
Schooling                            0.479620      0.003806

                                 Agriculture Employment  Schooling
Life Expectancy                                0.361705   0.717427
Alcohol                                        0.255498   0.526801
Percentage Expenditure                         0.342013   0.404376
Diphtheria                                     0.142392   0.409614
HIV/AIDS                                      -0.105067  -0.217555
Population                                    -0.022485  -0.043605
PopulationDensity                              0.044078   0.077303
Income Composition of Resources                0.350654   0.794735
HighIncome                                     0.324867   0.416127
```

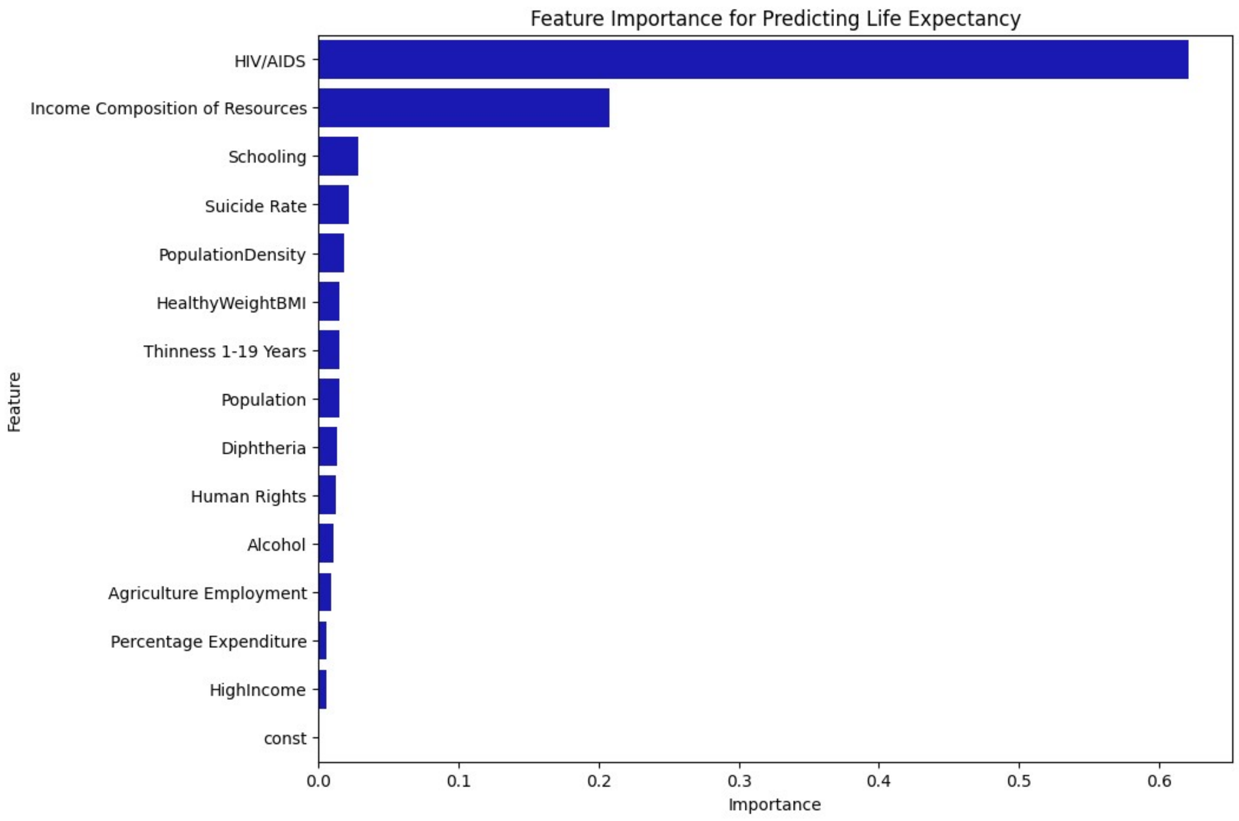
```
HealthyWeightBMI                                     -0.263869  -0.604840
Thinness 1-19 Years                                  -0.271680  -0.468108
Human Rights                                          0.284028   0.479620
Suicide Rate                                         -0.003705   0.003806
Agriculture Employment                                1.000000   0.415885
Schooling                                             0.415885   1.000000
```



## Feature Importance - Random Forest

```
def random_forest(X, y):
    """ Trains a Random Forest Regressor, evaluates feature
importance,
    and visualizes the most important features.

    Parameters:
        X : Array of independent variables.
        y : Array of the dependent variable.

    Returns:
        None. Prints the feature importance table and displays the bar
plot of feature importance.
    """
    # Split the data
    X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)

    # Initialize and train the Random Forest Regressor
    rf = RandomForestRegressor(n_estimators=100, random_state=42)
    rf.fit(X_train, y_train)

    # Predict on the test data
    y_pred = rf.predict(X_test)

    # Feature Importance
    feature_importance = pd.DataFrame({
        'Feature': X.columns,
        'Importance': rf.feature_importances_
    }).sort_values(by='Importance', ascending=False)

    print("\nFeature Importance:")
    print(feature_importance)

    # Visualize Feature Importance
    plt.figure(figsize=(10, 8))
    sns.barplot(x='Importance', y='Feature', color='mediumblue',
data=feature_importance)
    plt.title('Feature Importance for Predicting Life Expectancy')
    plt.show()

random_forest(X, y)
```

```
Feature Importance:
                             Feature  Importance
4                           HIV/AIDS    0.621173
7   Income Composition of Resources    0.207394
14                         Schooling    0.028217
12                      Suicide Rate    0.021548
6                  PopulationDensity    0.018420
9                   HealthyWeightBMI    0.015424
10               Thinness 1-19 Years    0.015423
5                         Population    0.014670
3                         Diphtheria    0.013067
11                      Human Rights    0.012430
1                            Alcohol    0.011083
13            Agriculture Employment    0.009423
2             Percentage Expenditure    0.005997
8                         HighIncome    0.005732
0                              const    0.000000
```


Feature Importance for Predicting Life Expectancy
HIV/AIDS
Income Composition of Resources
Schooling
Suicide Rate
PopulationDensity
HealthyWeightBMI
Thinness 1-19 Years
Population
Diphtheria
Human Rights
Alcohol
Agriculture Employment
Percentage Expenditure
HighIncome
const
Feature
0.0
0.1
0.2
0.3
0.4
0.5
0.6
Importance