Recent trends in life expectancy across countries: retrospective observational study using Machine learning algorithms

A MINI PROJECT REPORT

Submitted by

OM TIWARI(RA2011026010341)
SUHANI JAIN(RA2011026010242)
AISHWARYA(RA2011026010187)

Under the guidance of
Dr. Uma M
(Associate Professor / Dept.of Computational Intelligence)

In partial satisfaction of the requirements for the degree of

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE & ENGINEERING
With specialization in <AIML>

SCHOOL OF COMPUTING
COLLEGE OF ENGINEERING AND TECHNOLOGY
SRM INSTITUTE OF SCIENCE AND TECHNOLOGY
KATTANKULATHUR - 603203

MAY 2023

COLLEGE OF ENGINEERING & TECHNOLOGY
SRM INSTITUTE OF SCIENCE & TECHNOLOGY
S.R.M. NAGAR, KATTANKULATHUR – 603 203

BONAFIDE CERTIFICATE

Certified that this project report**" Recent trends in life expectancy across countries: retrospective observational study using Machine learning algorithms "** is the bonafide work of **" Om Tiwari (RA2011026010341), Suhani Jain (RA2011026010242)and Aishwarya(RA2011026010187)"** of III Year/VI Sem B.tech(CSE) who carried out the mini project work under my supervision for the course 18CSE479E - STATISTICAL MACHINE LEARNING in SRM Institute of Science and Technology during the academic year 2022-2023(Even sem).

SIGNATURE

Dr.Uma M
Associate Professor
Department of Computational Intelligence.
SRM Institute of Science and Technology

ABSTRACT

Health and social-services planning and investments require consideration of possible future trends in health and corresponding drivers. Many choices have long lag periods between initial investments and their effects, which can unfold over several decades; examples include training in different medical specialties, research and development for new drugs and vaccines, health system infrastructure construction, fiscal solvency of social security or health insurance, and policy implementation. Forecasts and alternative scenarios can help to frame these choices and to identify areas of more or less uncertainty. For instance, determining whether a country is likely to see more deaths from diabetes amid decreases in deaths from tuberculosis can serve as a crucial input to inform national policy dialogues and resource allocation. Quantitative forecasts of mortality and causes of death might be useful in this respect, particularly if they are linked to forecast and posited changes or alternative scenarios derived from the main independent drivers of population health. Such drivers include risk factors (eg, tobacco use, hypertension, air pollution, diet, and sanitation), interventions (eg, HIV treatment and vaccinations), and broader sociodemographic and health system factors.

Health forecasts, sometimes known as reference scenarios, aim to delineate the most likely future health trends. Forecast performance or accuracy can be empirically assessed by withholding data from recent periods and comparing forecasts generated without these data with what actually happened. Nonetheless, even forecasts based on models with a good out-of-time performance cannot anticipate all future drivers of health change. For example, no model from the 1970s would have forecast the HIV epidemic or, more recently, a cure for hepatitis C. Health forecasts are fundamentally imperfect, grounded only in what we know of the past and present; such forecasts can and should be supplemented with alternative scenarios that examine the universe of plausible futures. Alternative scenarios are particularly useful when they are linked to actionable choices that can affect different health drivers. A comprehensive forecast and scenarios framework for mortality and causes of death should possess two key attributes: good out-of-time forecast performance; and preservation of the causal effects for independent drivers and health outcomes that are consistent with known evidence from randomized controlled trials, and cohort or other observational studies.

Some country-specific forecasts show the alarming potential of rising obesity and subsequent declines in life expectancy, particularly driven by increased diabetes-related morbidity and mortality.13 As health and social service agencies face an increasing set of complex challenges (eg, population aging, escalating expenses, and shortages in health-care providers), the value of robust forecasts and alternative scenarios that can chart probable health futures and options for modifying these trajectories rises in tandem.

ACKNOWLEDGEMENT

We express our heartfelt thanks to our honorable **Vice Chancellor, Dr. C. MUTHAMIZHCHELVAN**, for being the beacon in all our endeavors.

We would like to express my warm gratitude to our **Registrar, Dr. S. Ponnusamy,** for his encouragement.

We express our profound gratitude to our **Dean (College of Engineering and Technology), Dr. T. V.Gopal,** for bringing out novelty in all executions.

We would like to express our heartfelt thanks to Chairperson, School of Computing, **Dr. Revathi Venkataraman,** for imparting confidence to complete my course project

We wish to express my sincere thanks to **Course Audit Professor Dr.Annapurani Panaiyappan, Professor and Head, Department of Computational Intelligence** and **Course Coordinators** for their constant encouragement and support.

We are highly thankful to our Course project Faculty **Dr. Uma M, Assistant Professor, CINTEL,** for her assistance, timely suggestion, and guidance throughout the duration of this course project.

We extend my gratitude to our **HoD, Dr. R. Annie Uthra, Professor and Head of the CINTEL** department, and my Departmental colleagues for their Support.

Finally, we thank our parents and friends, near and dear ones, who directly and indirectly contributed to the successful completion of our project. Above all, I thank the almighty for showering his blessings on me to complete my Course project.

TABLE OF CONTENTS

1.INTRODUCTION

Life expectancy is a key summary measure of the health and wellbeing of a population. A nation's life expectancy reflects its social and economic conditions and the quality of its public health and healthcare infrastructure, among other factors. Monumental improvements in life expectancy have been the predominant trend for high income, developed countries over the course of the 20th and 21st centuries.

In the absence of wars, new epidemics, or substantial economic reforms, lack of improvement or stagnation in life expectancy gains are viewed as a cause for concern, and actual declines in life expectancy are particularly alarming. Stagnation or declines in life expectancy may signal a decline in the health profile of the population driven by adverse socio economic trends, a deterioration in the provision or quality of healthcare services, or worsening behavioral factors.

Recent trends in the United States suggest that it has experienced a break from the trajectory of continued gains in life expectancy. In December 2017, the US National Center for Health Statistics reported that the country experienced a decline in life expectancy for two consecutive years.**3** Between 2014 and 2016, overall life expectancy in the USA declined by 0.3 years.This decline was most pronounced among men, who experienced a decline of 0.2 years in each of the two consecutive years. American women experienced a decline of 0.2 years during 2014-15 and no appreciable change in life expectancy during 2015-16.

These declines are particularly troublesome in light of the US's poor performance in international rankings of life expectancy. The USA now has the lowest life expectancy levels among high income developed countries, and Americans fare poorly across a broad set of ages, health conditions, and causes of death compared with their counterparts in these countries.

The USA may not be alone in experiencing declines in life expectancy—a recent study documented an increase in the age standardized death rate in England and Wales during 2014-15. However, whether this decline was driven by factors similar to those in the USA and whether any other high income countries experienced similar declines is unknown.

We assessed whether declines in life expectancy occurred in other high income countries during 2014-16, the main age groups and causes of death contributing to these declines, and the extent to which these declines were driven by shared or differing factors across countries. This study also provides an evaluation of how recent adverse trends in life expectancy have affected the US's life expectancy standing compared with other high income countries.

We constructed life tables by sex for each country in 1990, 1995, 2000, and 2005, and for each year between 2010 and 2016 using standard life table methods and graduation to parameterize the life table. A life table is a demographic tool used to compute life expectancy, and graduation is a recursive smoothing technique to produce estimates of the average years lived by decedents within an age group. The supplementary methodological appendix provides additional information on life table methods.

Data on deaths and person years of exposure come from the Human Mortality Database through the

latest year available. For the remaining years, we used data from the official vital statistics agencies of individual countries (see supplementary table A1). The following quantities of interest are drawn from these life tables: life expectancy at birth—the number of years newborns who experience the life table death rates throughout their lifetime could expect to live; life expectancy between ages 0 and 65 years—the number of years newborns who experience the life table death rates throughout their lifetime could expect to live between the ages of 0 and 65; and life expectancy at ages 65 or more years—the number of years individuals who have survived to age 65 and who then experience the life table death rates throughout the remainder of their lifetime could expect to live.

Cause of death data

We consider 22 mutually exclusive and exhaustive cause of death categories (see supplementary table A2 for the corresponding ICD-10 codes) and their contribution to changes in life expectancy during 2014-15. Causes of death data are primarily drawn from the World Health Organization mortality databaseand supplemented with data from Statistics Canada and Statistics Portugal(see supplementary table A1). France's 2015 cause of death data are not yet available. We obtained the proportions of total deaths due to each of these 22 categories and applied them to all cause death rates to obtain specific death rates by age, sex, year, and country.

Two hypotheses have been suggested to explain increasing mortality in the USA and the UK: particularly severe influenza seasonsand ongoing opioid epidemics. In a subset of analyses testing these hypotheses we focused on four cause of death categories: influenza and pneumonia (ICD-10 codes J09-J18), respiratory diseases (J00-J99), drug overdose (X40-X44, X60-X64, X85, and Y10-Y14), and external causes (V01-Y98). These categories are non-mutually exclusive: influenza and pneumonia are a subcategory of respiratory diseases, and drug overdose is a subcategory of external causes. We combined influenza and pneumonia into a single category because deaths due to influenza are commonly coded as pneumonia on death certificates. The two broader categories are employed in order to include countries that have less detailed cause of death data and because deaths from influenza are often under-detected and end up being coded as deaths from other respiratory illnesses, particularly among older adults.

Cause of death analyses

We used Arriaga's decomposition to determine which causes of death are primarily responsible for the 2014-2015 change in life expectancy for men and women in each country.Arriaga's decomposition is a method that partitions changes in life expectancy into cause of death contributions. We computed five cause of death contributions for each country (further broken down into 22 specific categories in supplementary figures A2 and A3). Negative contributions indicate that the cause tended to reduce life expectancy, whereas positive contributions indicate that the cause tended to increase life expectancy. For a given country-sex combination, the cause specific contributions sum to the total change in life expectancy during 2014-15. The supplementary methodological appendix provides additional information on Arriaga's decomposition.

Corresponding to the two hypotheses described previously, we examined four specific causes of death that represent the impact of influenza and pneumonia and drug overdose. To determine whether these causes are responsible for recent declines in life expectancy, we computed cause deleted life tables using Chiang's method for each country and for each of the four cause of death

categories of interest in 2014 and 2015.Cause deleted life tables are counterfactuals that answer the question, “What would life expectancy be in a given country if all deaths from a specific cause of death were eliminated?” This in turn allows us to determine whether, for example, life expectancy in the USA would still have declined during 2014-15 in the absence of drug overdose.

We determined the contribution of these four causes of death to life expectancy trends by comparing the observed life tables with the cause of deleted life tables. The contribution of a specific cause of death to a decline in life expectancy during 2014-15 can be characterized in three ways. If, in the absence of the cause of death: life expectancy would have increased during 2014-15, then that cause is responsible for all of the decline; life expectancy would have declined by a smaller magnitude than observed during 2014-15, then that cause is partly responsible for the decline and we quantified this contribution in both absolute and percentage terms; and life expectancy would have declined even more than observed during 2014-15, then that cause is not responsible for the decline. The supplementary methodological appendix provides additional information on cause deleted life table methods.

Declines in life expectancy have particular salience for the USA because it has lagged behind its peer countries.While the levels of life expectancy in the USA have been ranked near the bottom of high income countries since 1990, the rate of increase in life expectancy was not substantially different from that of other countries until 2010. For example, between 1990 and 2010, the pace of increase in life expectancy for American men was fairly comparable to that of countries with the highest life expectancy in each year—the world leaders . American women had slower rates of improvement in the 1990s but stronger gains in the 2000s . From 2010 onwards, however, life expectancy essentially plateaued for American men and women, resulting in the other high income countries pulling far above the USA.

2.LITERATURE SURVEY

Health forecasts and alternative future scenarios can serve as vital inputs into long-term planning and investments in health, particularly in terms of framing different choices, their potential effects, and the relative certainty associated with each option. Past work to generate health-focused forecasts includes that from the UN Population Division, which routinely produces all-cause mortality forecasts through year 2100, and the Austrian Wittgenstein Center, which produces life expectancy forecasts with different scenarios to the end of the 21st century. Longer-range forecasts have also been developed to assess the potential effects of climate change on mortality. Furthermore, various national agencies produce country-level mortality forecasts, and forecasts for individual causes of death have been produced periodically as well. Comprehensive forecasts of cause-specific and all-cause mortality were developed as part of the Global Burden of Disease Study 1990 (GBD 1990); those methods were then applied for the period from 2002–30. The primary purpose of these past modelling efforts was to generate reference or baseline forecasts of what was likely to occur on the basis of past trends; however, few—if any—offered insights into a range of future scenarios while accounting for independent drivers of potential health changes.

Added value of this study

With this study, we provided a completely novel approach to forecasting all-cause and cause-specific mortality and scenario construction. Our modelling framework was designed to leverage the data on risk outcome relationships in the Global Burden of Diseases, Injuries, and Risk Factors Study 2016 such that the relationship between risk factors (eg, smoking) and specific disease outcomes (eg, lung cancer) were consistent with relevant cohort studies and randomised controlled trials. Because this forecasting framework is grounded in 79 independent drivers of health change, we could leverage these models to generate a full suite of alternative scenarios beyond reference forecasts. The overall model has three components: a model for risk-attributable deaths, a model for mortality not explained by risk factors that is a function of the Socio-demographic Index (SDI) and other covariates depending on the cause and time, and an autoregressive integrated moving average process for the unexplained latent trends for each location-age-sex-cause. To ensure the robustness of this framework, we assessed the overall model performance by fitting it to data from 1990–2006 and evaluating its forecasts for 2007–2016. Our model performed better than other widely used methods such as Lee-Carter. Based on our model, we generated reference scenarios for all-cause mortality, life expectancy, and 250 causes of death for 195 countries and territories from 2016–40. Additionally, we assessed better health and worse health scenarios for each metric by country, reflecting the 85th and 15th percentiles, respectively, of the annualised rates of change observed for the drivers across locations in the past.

Implications of all the available evidence

Our reference forecast predicted continued declines in global mortality and improvements in life expectancy, though at a slower rate than achieved in the past. Because of faster progress among many lower-SDI countries, absolute disparities between countries are currently projected to narrow by 2040. Nonetheless, the differences between better health and worse health scenarios for 2040 remain substantial, emphasising that the reference forecast is not inevitable and thus policy choices

made today can profoundly affect each country's future health trajectories. For most countries, prioritising non-communicable diseases (NCDs) and NCD-related risks in health planning and investment decisions has the potential to markedly reduce premature mortality by 2040. Although NCDs were projected to rise in many low-SDI countries, communicable, maternal, neonatal, and nutritional diseases are likely to remain among the leading causes of early death. Furthermore, based on our 2040 worse health scenario, rebounds in HIV mortality and thus reversals in life expectancy could occur if countries cannot maintain the gains achieved in the past. Continued technical innovation and increased health spending, both domestic and international funding targeted to countries with the most need, are crucial for a future where all people have the opportunity to live full, healthy lives.

The GBD provides a unique resource from which a new generation of health forecasts with alternative scenarios can be developed.14, 15 GBD has refined the collection and standardisation of detailed health and risk data over several recent publication cycles and now covers, from 1990–2016, 195 countries and territories with data on cause-specific and all-cause mortality, risk factors, and selected interventions.16, 17 The GBD comparative risk assessment, with its meta-analyses of published studies (ie, randomised trials and cohort and other observational data) on risk-outcome causal relationships, enables the modelling of future disease burden while accounting for drivers of health change.16 Additionally, an overall measure of development—the Socio-demographic Index (SDI)—was developed as part of GBD, providing a mechanism for assessing the effects of improved development on health. Drawing from the broader GBD study, we have developed a novel forecasting model, producing reference forecast, better health scenario, and worse health scenario for life expectancy, all-cause mortality, and cause-specific mortality from 250 causes from 2017–40 in 195 countries and territories. By leveraging the relationships between independent drivers of health captured within GBD, we provide a robust platform from which alternative scenarios can be assessed, which could be vital inputs for strategies and investments to improve population health.

3.STATISTICAL ANALYSIS

3.1 MEAN,MEDIAN,MODE

```
import pandas as pd
data = pd.read_csv('life_expectancy_dataset.csv')
data.head()
print('Mean life expectancy:', data['Life_expectancy '].mean())
print('Mode life expectancy:', data['Life_expectancy '].mode())
print('Median life expectancy:', data['Life_expectancy '].median())
```

```
Mean life expectancy: 69.22493169398908
Mode life expectancy: 0    73.0
Name: Life_expectancy , dtype: float64
Median life expectancy: 72.1
```

3.2 T TEST

```
import numpy as np
import pandas as pd
import difflib
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.metrics.pairwise import cosine_similarity
import pandas as pd
from scipy.stats import ttest_ind
```

```
#/content/led.csv
import pandas as pd
from scipy.stats import ttest_1samp

# Load the dataset
data = pd.read_csv('/content/led.csv')

# Extract the column that you want to test
sample = data['percentageexpenditure']

# Specify the null hypothesis value
null_mean = 0

# Perform the t-test
t_stat, p_val = ttest_1samp(sample, null_mean)
```

```
# Print the results
print('T-Statistic: ', t_stat)
print('P-value: ', p_val)
```

T-Statistic: 20.129469697443017
P-value: 1.5894032680437329e-84

Hypothesis

```
import pandas as pd
from scipy.stats import ttest_ind

# Load the dataset/content/HR_comma_sep.csv
df = pd.read_csv('/content/led.csv')

# Split the dataset into two samples
sample1 = df['percentageexpenditure'][df['Status'] == 'Developed']
sample2 = df['percentageexpenditure'][df['Status'] == 'Developing']

…# Compare the p-value to alpha
if p_value < alpha:
    print('Reject the null hypothesis')
else:
    print('Fail to reject the null hypothesis')
```

t-statistic: 27.629292883512814
p-value: 1.4558637811848918e-149
Reject the null hypothesis

3.3 CHI TEST

```
import pandas as pd
from scipy.stats import chi2_contingency

# Load the dataset
df = pd.read_csv('/content/led.csv')

# Create a contingency table
contingency_table = pd.crosstab(df['Alcohol'], df['AdultMortality'])
# Split the dataset into two samples
sample1 = df['percentageexpenditure'][df['Status'] == 'Developed']
sample2 = df['percentageexpenditure'][df['Status'] == 'Developing']
```

```
# Set the significance level
alpha = 0.05

# Perform two-sample t-test
t_statistic, p_value = ttest_ind(sample1, sample2)
# Perform chi-square test
chi2_statistic, p_value, dof, expected = chi2_contingency(contingency_table)

# Print the results
print('chi-square statistic:', chi2_statistic)
print('p-value:', p_value)
```

chi-square statistic: 433990.1501160751
p-value: 1.0

3.4 F TEST(ANNOVA)

```
import pandas as pd
from scipy.stats import f_oneway

# Load the dataset
df = pd.read_csv('/content/led.csv')

# Perform ANOVA
f_statistic, p_value = f_oneway(df['infantdeaths'], df['HIV/AIDS'], df['Measles'])

# Print the results
print('F-statistic:', f_statistic)
print('p-value:', p_value)
```

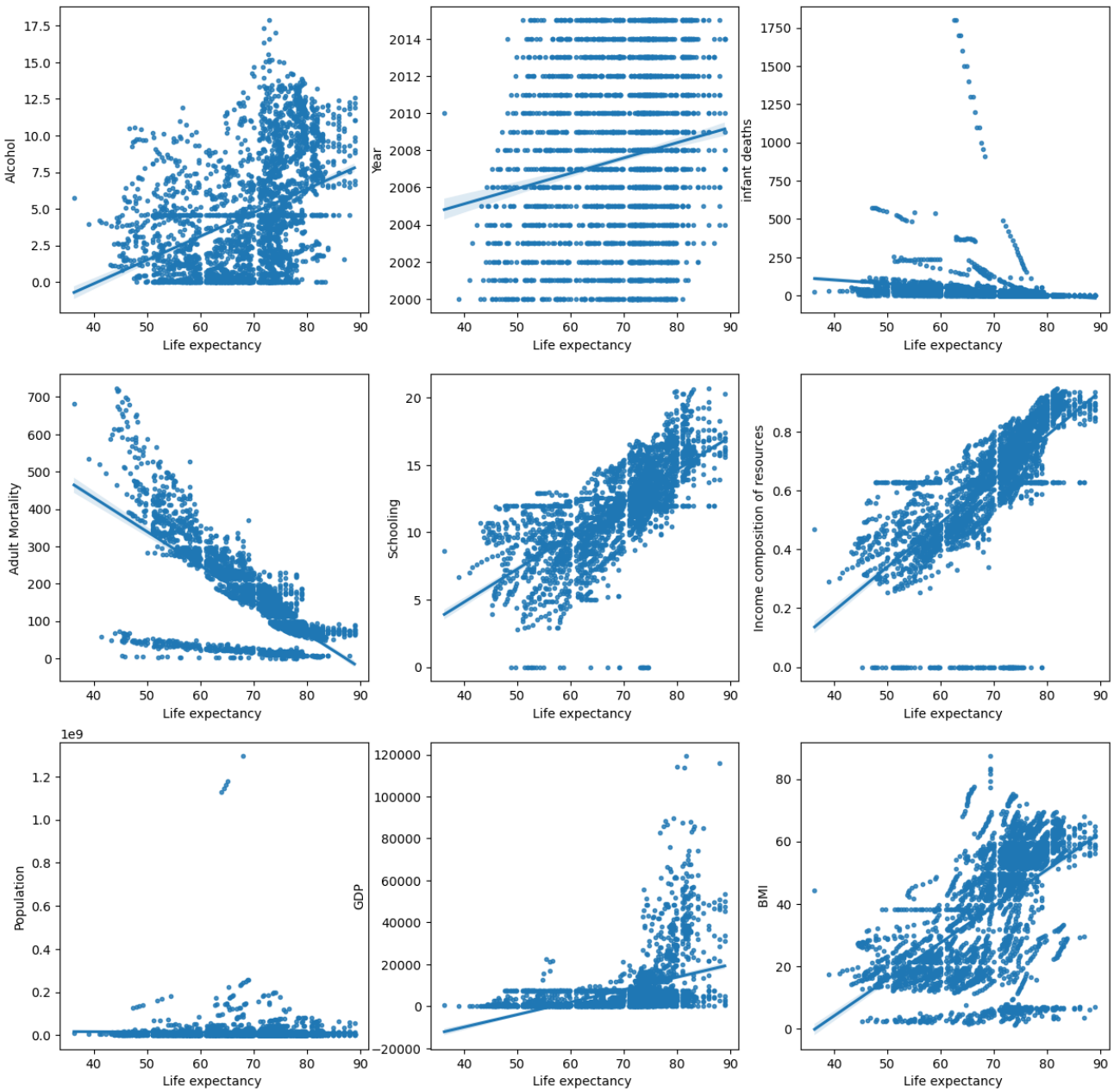
F-statistic: 129.0757841496727
p-value: 5.605352909461064e-56

4.SUPERVISED LEARNING

4.1 LINEAR REGRESSION

```
!pip install pycountry pycountry-convert

import os
import pycountry
import pycountry_convert as pc
import pandas as pd
import seaborn
import random
import plotly.express as px
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error
from sklearn.metrics import r2_score
from sklearn.linear_model import LinearRegression
data = pd.read_csv('/content/led.csv')
for column in dataset.columns:
    if dataset[column].isnull().sum() != 0:
        dataset[column] = dataset[column].fillna(value=dataset[column].mean())
dataset.isnull().sum()
#Visualization
continents = []
for country in dataset['Country']:
    try:
        alpha2 = pc.country_name_to_country_alpha2(country)
        continent_code = pc.country_alpha2_to_continent_code(alpha2)
        continent = pc.convert_continent_code_to_continent_name(continent_code)
    except:
        continent = 'Africa'
    continents.append(continent)
dataset["Continent"] = continents
dataset.head()

df = px.data.gapminder()
fig = px.scatter(dataset[["Country", "Year", "Life expectancy ", "GDP", "Population", "Continent"]],
x="GDP", y="Life expectancy ",
                 size="Population", color="Continent",
                 hover_name="Country", log_x=True, size_max=50)
fig.show()
#Now we will find those predicting factors that really affect Life expactancy. First let's look at the
heatmap as a whole:
```

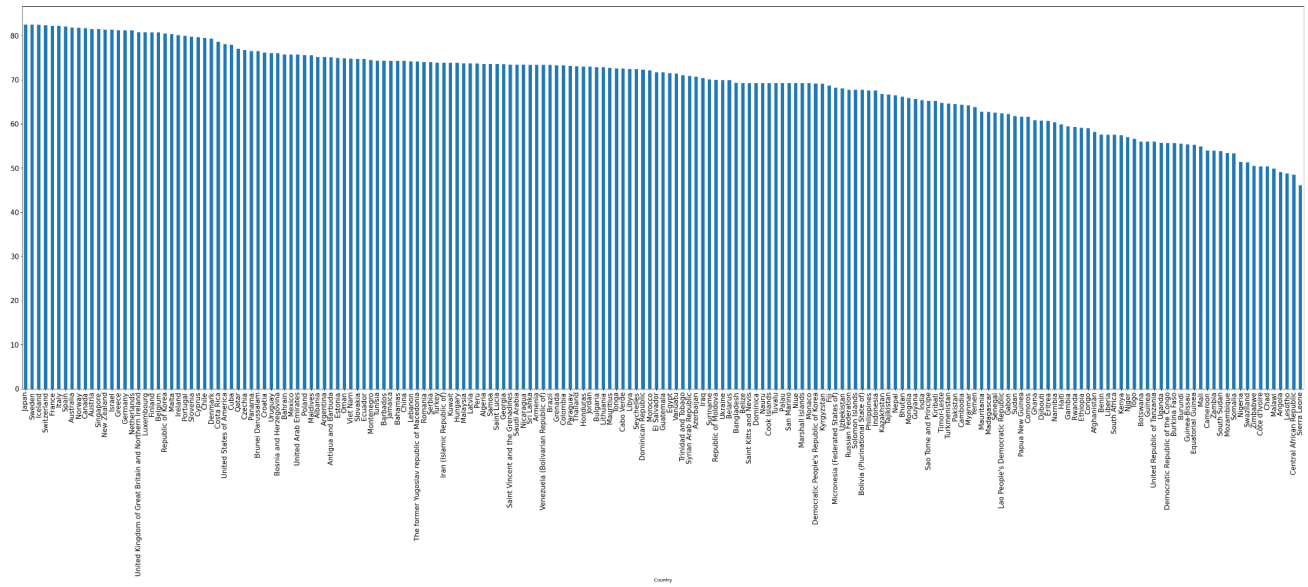
```
dropped_columns = ['Country', 'Status', 'Year']
fig, ax = plt.subplots(figsize=(15, 15))
seaborn.heatmap(dataset.drop(dropped_columns, axis=1).corr(), annot=True, square=True,
linewidths=.5)

interest = [
    'Alcohol', 'Year', 'infant deaths',
    'Adult Mortality', 'Schooling', 'Income composition of resources',
    'Population', 'GDP', ' BMI ',
]
fig, axes = plt.subplots(nrows=3, ncols=3, figsize=(15, 15))
for i, label in enumerate(interest):
    seaborn.regplot(
        x=dataset['Life expectancy '],
        y=dataset[label],
        ax=axes[i//3][i%3],
        marker='.',

    )
```



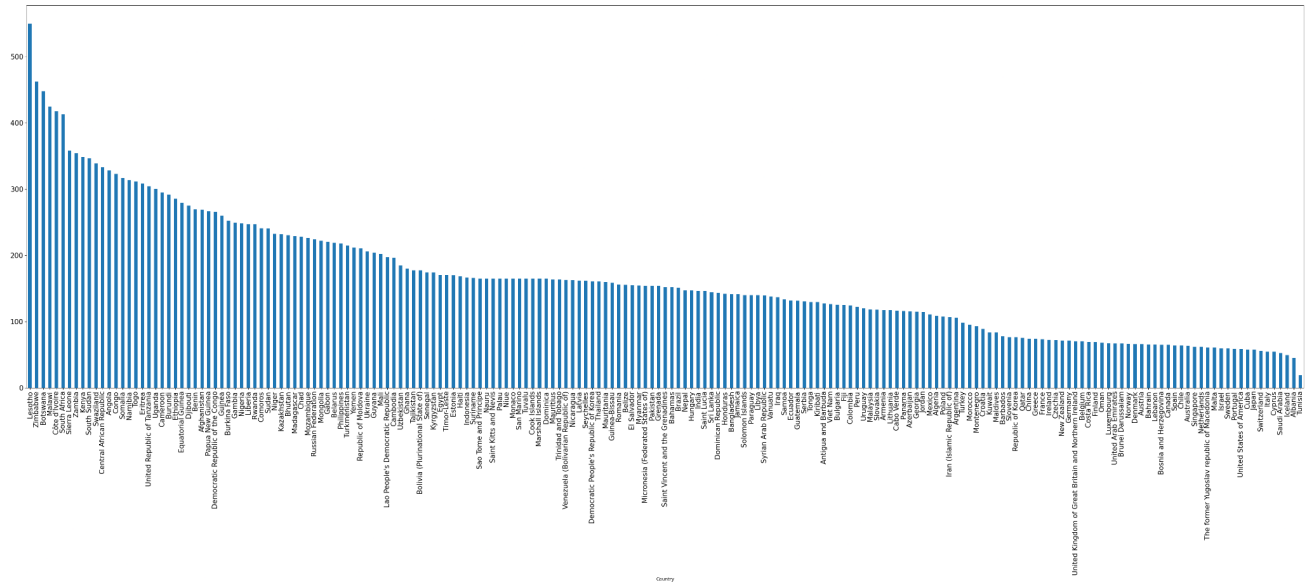
The following chart explains the following trends:

- Countries with high alcohol consumption have high life expectancy.
- Each year the average life expectancy increases.
- Countries with lower infant and adult mortality have longer life expectancy.
- !IMPORTANT! Schooling affects life expectancy.
- It also makes sense that as one's own income increases, one's life expectancy increases
- As the population grows, life expectancy decreases.
- (Except for a few) GDP and life expectancy have a strong linear relationship.
- BMI and life expectancy have a linear relationship.

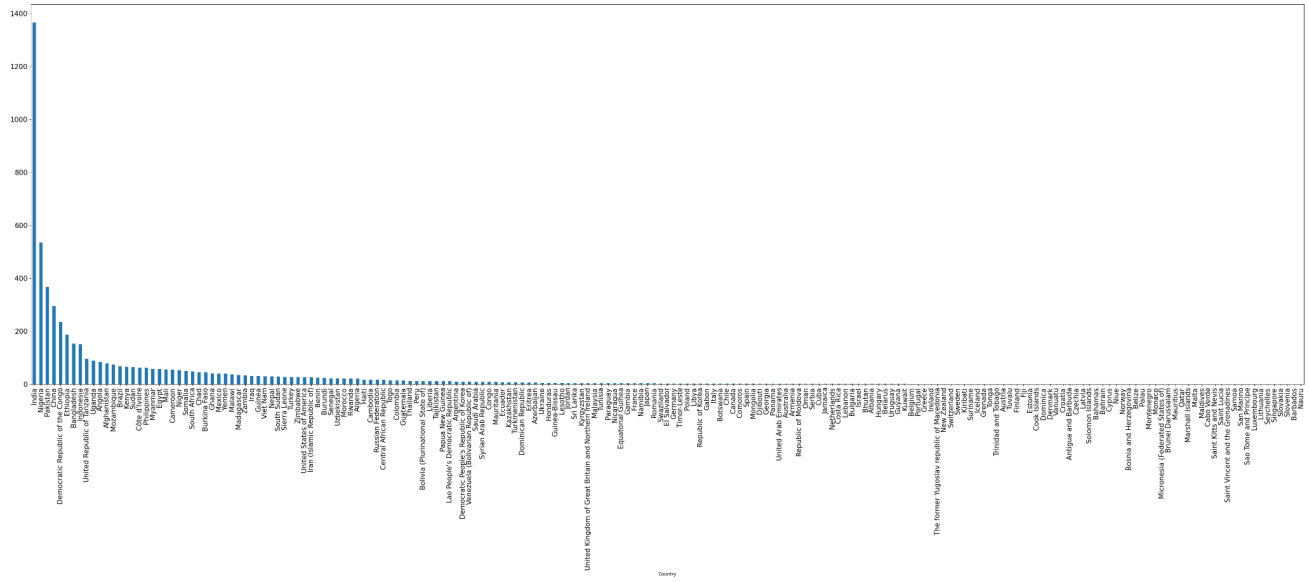
#Calculating different metrics by Country and visualizing them
plt.figure(figsize=(50,15))
val=dataset.groupby('Country')['Life expectancy
'].mean().sort_values(ascending=False).plot(kind='bar',fontsize=15)



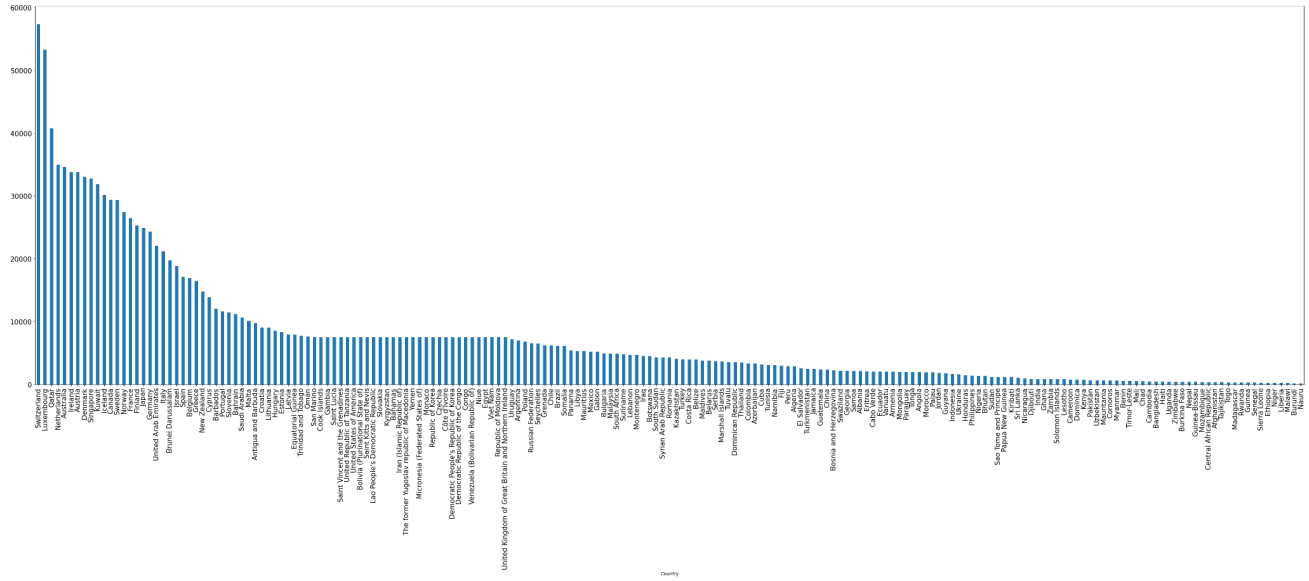
plt.figure(figsize=(50,15))
val=dataset.groupby('Country')['Adult Mortality'].mean().sort_values(ascending=False).plot(kind='bar',fontsize=15)



plt.figure(figsize=(50,15))
val=dataset.groupby('Country')['infant deaths'].mean().sort_values(ascending=False).plot(kind='bar',fontsize=15)



```
plt.figure(figsize=(50,15))
val=dataset.groupby('Country')['GDP'].mean().sort_values(ascending=False).plot(kind='bar',fontsize=15)
```



#Using linear regression to predicting the life expectancy

```
df = dataset.copy().drop('Continent', axis=1)
df.rename(columns=lambda x: x.strip(), inplace=True)
df['Status'].value_counts()
```

Developing 2426
Developed 512
Name: Status, dtype: int64

```
def encode_status(x):
    if x == 'Developed':
        return 1
    else:
```

```
        return 0
df['Status'] = df['Status'].apply(encode_status)
df = pd.concat([df, pd.get_dummies(df['Country'], prefix='Country', drop_first=True)], axis=1)
df = df.drop(['Country'], axis=1)
X = df.drop(['Life expectancy'], axis=1)
y = df['Life expectancy']
df
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.15)
#Score
lr = LinearRegression()
lr.fit(X_train, y_train)
y_pred = lr.predict(X_test)
mean_squared_error(y_test, y_pred)
r2_score(y_test, y_pred)*100
```

95.72150521606262

4.2 LOGISTIC REGRESSION

```
import numpy as np


class LogisticRegression:
    def __init__(self, alpha=0.05, iterations=1000):
        self.alpha = alpha
        self.iterations = iterations


    def fit(self, X, y):
        # Initialize the weights
        self.theta = np.zeros(X.shape[1])


        # Run gradient descent
        for i in range(self.iterations):
            z = np.dot(X, self.theta)
            h = self.sigmoid(z)
            gradient = np.dot(X.T, (h - y)) / y.size
            self.theta -= self.alpha * gradient
```

```
    def predict(self, X):
        z = np.dot(X, self.theta)
        h = self.sigmoid(z)
        return (h >= 0.5).astype(int)

    def sigmoid(self, z):
        return 1 / (1 + np.exp(-z))
import pandas as pd

# Load the CSV file into a DataFrame object
df = pd.read_csv('/content/led.csv')

# Print the first few rows of the DataFrame
print(df.head())

#Since, only a few rows have null values in them, we are only removing those rows from the dataset.
df =
df.dropna(subset=['GDP','BMI','Lifeexpectancy','AdultMortality','Alcohol','HepatitisB','Polio','Totale
xpenditure','Diphtheria'])
total_null_values = df.isnull().sum().sort_values(ascending = False)
percentage = ((df.isnull().sum()/df.isnull().count())*100).sort_values(ascending = False)
print("Total values present are ",df.shape[0])
total_missing_data = (pd.concat([total_null_values,percentage.round(2)],axis=1,keys=['Total
Missing','In precentage']))
total_missing_data
df['Population'].fillna(value = df['Population'].mean(),inplace=True)
#after removing null values
total_null_values = df.isnull().sum().sort_values(ascending = False)
percentage = ((df.isnull().sum()/df.isnull().count())*100).sort_values(ascending = False)
print("Total values present are ",df.shape[0])
total_missing_data = (pd.concat([total_null_values,percentage.round(2)],axis=1,keys=['Total
Missing','In precentage']))
```

```
total_missing_data
#rSplitting the dependent and independent variables.
x = df.drop("Schooling",axis=1)
y = df['Schooling']
y.shape #checking the output variable
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.2,random_state=42)
y_train
model=LogisticRegression()
df.drop(columns=['Country'], inplace=True)
df.shape


selected_columns = ['Measles', 'Polio', 'HIV/AIDS']
data = df[selected_columns]


X = df[selected_columns]


y = df["Status"]
# Split the data 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)


# Train a logistic regression model
model = LogisticRegression()
model.fit(X_train, y_train)
X_train.shape
X_test.shape,y_train.shape,y_test.shape
model=LogisticRegression()
lr = model.fit(X_train,y_train)
lr.score(X_train,y_train)
```

```
lr_accuracy=lr.score(X_test,y_test)
print(lr_accuracy)
# Train a logistic regression model
model = LogisticRegression()
model.fit(X_train, y_train)

# Make predictions on the test set
y_pred = model.predict(X_test)

# Compute classification metrics
report = classification_report(y_test, y_pred)

print(report)
# Train a logistic regression classifier on the training data
clf = LogisticRegression()
clf.fit(X_train, y_train)

# Predict the labels of the testing data
y_pred = clf.predict(X_test)

# Compute the precision score for the testing data
precision = precision_score(y_test, y_pred, average='micro')
print("Precision:", precision)
```

Precision: 0.8814016172506739

```
from sklearn.metrics import f1_score

# Predict the target values on the test set
```

```
y_pred = clf.predict(X_test)

# Compute classification metrics
accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred, average='weighted')
recall = recall_score(y_test, y_pred, average='weighted')
f1 = f1_score(y_test, y_pred, average='weighted')

# Print the metrics
print(f"Accuracy: {accuracy:.2f}")
print(f"Precision: {precision:.2f}")
print(f"Recall: {recall:.2f}")
print(f"F1 Score: {f1:.2f}")
```

Accuracy: 0.88

Precision: 0.78

Recall: 0.88

F1 Score: 0.83

4.3 DECISION TREE

```
# instantiate the DecisionTreeClassifier model with criterion entropy

clf_en = DecisionTreeClassifier(criterion='entropy', max_depth=3, random_state=0)


# fit the model
clf_en.fit(X_train, y_train)

y_pred_en = clf_en.predict(X_test)
```

from sklearn.metrics import accuracy_score

print('Model accuracy score with criterion entropy: {0:0.4f}'. format(accuracy_score(y_test, y_pred_en)))

y_pred_train_en = clf_en.predict(X_train)

y_pred_train_en

print('Training-set accuracy score: {0:0.4f}'. format(accuracy_score(y_train, y_pred_train_en)))

C# print the scores on training and test set

print('Training set score: {:.4f}'.format(clf_en.score(X_train, y_train)))

print('Test set score: {:.4f}'.format(clf_en.score(X_test, y_test)))

Training set score: 0.9642

Test set score: 0.9299

plt.figure(figsize=(12,8))

from sklearn import tree

tree.plot_tree(clf_en.fit(X_train, y_train))

#Data Preprocessing

```
data.info()
x = data['Life expectancy '].mean()
data['Life expectancy '].fillna(x, inplace=True)

x = data['Adult Mortality'].mean()
data['Adult Mortality'].fillna(x, inplace=True)

x = data['Alcohol'].mean()
data['Alcohol'].fillna(x, inplace=True)
```

```
x = data['Hepatitis B'].mean()
data['Hepatitis B'].fillna(x, inplace=True)

x = data[' BMI '].mean()
data[' BMI '].fillna(x, inplace=True)

x = data['Polio'].mean()
data['Polio'].fillna(x, inplace=True)

x = data['Total expenditure'].mean()
data['Total expenditure'].fillna(x, inplace=True)

x = data['Diphtheria '].mean()
data['Diphtheria '].fillna(x, inplace=True)

x = data['GDP'].mean()
data['GDP'].fillna(x, inplace=True)

x = data['Population'].mean()
data['Population'].fillna(x, inplace=True)

x = data[' thinness  1-19 years'].mean()
data[' thinness  1-19 years'].fillna(x, inplace=True)

x = data[' thinness 5-9 years'].mean()
data[' thinness 5-9 years'].fillna(x, inplace=True)

x = data['Income composition of resources'].mean()
data['Income composition of resources'].fillna(x, inplace=True)

x = data['Schooling'].mean()
data['Schooling'].fillna(x, inplace=True)
from sklearn.preprocessing import LabelEncoder

le = LabelEncoder()
data['Country'] = le.fit_transform(data['Country'])

le = LabelEncoder()
data['Status'] = le.fit_transform(data['Status'])
data.isnull().sum()
data = data.iloc[:1000, :]

data.shape
data.head()
```

4.4 Random forest

#RandomForest Regressor

```
import random
# Decision Tree Regressor
# Node Class
class Node:

    def __init__(self, feature=None, threshold=None, left=None, right=None, *, value=None):
        self.feature = feature
        self.threshold = threshold
        self.left = left
        self.right = right
        self.value = value

    def is_leaf_node(self):
        return self.value is not None


# Decision Tree Regressor Class
class RegressionTree:
    def __init__(self, n_feats = None, max_depth = 100, min_samples_split = 2, min_samples_leaf=1,
min_weight_fraction_leaf=0.0, max_features='auto',
                 random_state=None ,max_leaf_nodes=None, min_impurity_decrease=0.0,
ccp_alpha=0.0):

        self.root = None
        self.n_feats = n_feats
        self.max_depth = max_depth
        self.min_samples_split = min_samples_split
        self.max_features = max_features



    def fit(self, X, Y):
        self.n_feats = X.shape[1] if not self.n_feats else min(self.n_feats, X.shape[1])
        self.col = list(X.columns)
        self.root = self.growTree(X, Y)

    def growTree(self, X, Y, depth = 0):

        df = X.copy()
        df['y'] = Y
```

```
    ymean = np.mean(Y)

    self.mse = self.get_mse(Y, ymean)

    n_sample, n_feature = X.shape

    # stopping criteria
    if (depth >= self.max_depth or n_sample <= self.min_samples_split):
        leaf_value = np.mean(Y)
        return Node(value=leaf_value)

    data = self.feature_sampling(X, self.max_features)

    feats_idxs = list(data.columns)

    best_feat, best_thresh = self.best_criteria(X, Y, feats_idxs)

    left_df, right_df = df[df[best_feat]<=best_thresh].copy(), df[df[best_feat]>best_thresh].copy()

    left = self.growTree(left_df.drop('y', axis=1), left_df['y'].values.tolist(), depth+1)
    right = self.growTree(right_df.drop('y', axis=1), right_df['y'].values.tolist(), depth+1)

    return Node(best_feat, best_thresh, left, right)


# find out best criteria
def best_criteria(self, X, Y, feats_idxs):

    df = X.copy()

    df['y'] = Y

    mse_base = self.mse

    best_feature = None
    best_thresh = None

    for feat in feats_idxs:

        xdf = df.sort_values(feat)

        x_mean = self.moving_average(xdf[feat], 2)

        for value in x_mean:
```

```
            left_y = xdf[xdf[feat] < value]['y'].values
            right_y = xdf[xdf[feat] >= value]['y'].values

            left_mean = 0
            right_mean = 0
            if len(left_y) > 0:
                left_mean = np.mean(left_y)
            if len(right_y) > 0:
                right_mean = np.mean(right_y)

            res_left = left_y - left_mean
            res_right = right_y - right_mean

            r = np.concatenate((res_left, res_right), axis=None)

            n = len(r)

            r = r ** 2
            r = np.sum(r)
            mse_split = r / n

            if mse_split < mse_base:
                mse_base = mse_split
                best_feature = feat
                best_thresh = value

    return (best_feature, best_thresh)

def get_mse(self, y_true, y_hat):
    n = len(y_true)

    r = y_true - y_hat

    r = r ** 2

    r = np.sum(r)

    return r / n

def moving_average(self, x:np.array, window : int):
    return np.convolve(x, np.ones(window), 'valid') / window

def predict(self, X):
    X = X.to_numpy().tolist()
```

```
        return np.array([self.traverse_tree(x, self.root) for x in X])

    def traverse_tree(self, x, node):

        if node.value is not None:
            return node.value

        fr = node.feature
        index = self.col.index(fr)

        if x[index] <= node.threshold:
            return self.traverse_tree(x, node.left)

        return self.traverse_tree(x, node.right)

    def feature_sampling(self, data, val):
        if type(val) == int:
            col = random.sample(data.columns.tolist()[:], val)
            new_df = data[col]
            return new_df
        elif type(val) == float:
            col = random.sample(data.columns.tolist()[:], int(val * data.shape[1]))
            new_df = data[col]
            return new_df
        elif val == 'auto' or val == 'sqrt':
            col = random.sample(data.columns.tolist()[:], int(math.sqrt(data.shape[1])))
            new_df = data[col]
            return new_df
        elif val == 'log2':
            col = random.sample(data.columns.tolist()[:], int(math.log2(data.shape[1])))
            new_df = data[col]
            return new_df
        else:
            return data


class RandomForestRegressor:
    def __init__(self, n_estimators=100, criterion='entropy', max_depth=None, min_samples_split=2,
bootstrap=True, max_samples=None,
                 max_features='auto', oob_score=False):
        self.n_estimators = n_estimators
        self.criterion = criterion
        self.max_depth = max_depth
        self.min_samples_split = min_samples_split
        self.bootstrap = bootstrap
```

```
        self.max_samples = max_samples
        self.max_features = max_features
        self.oob_score = oob_score

    def fit(self, X_train, y_train):
        dummy_data = X_train.copy()
        dummy_data['target'] = y_train

        self.tree_list = []

        for i in range(self.n_estimators):

            if self.bootstrap == True:
                df = self.row_sampling(dummy_data, self.max_samples)
            else:
                df = dummy_data.copy()

            tree = RegressionTree(max_depth=self.max_depth,
min_samples_split=self.min_samples_split, max_features=self.max_features)

            tree.fit(df.drop('target', axis=1), df.target)
            print(tree, i)
            self.tree_list.append(tree)

    def row_sampling(self, data, val):
        if type(val) == float:
            return data.sample(int(val * data.shape[0]), replace=True)
        if type(val) == int:
            return data.sample(val, replace=True)
        if val == None:
            return data

    def predict(self, X_test):
        y_preds = np.empty((X_test.shape[0], len(self.tree_list)))
        # Let each tree make a prediction on the data
        for i, tree in enumerate(self.tree_list):
            # Indices of the features that the tree has trained on
            # idx = tree.feature_indices
            # Make a prediction based on those features
            prediction = tree.predict(X_test)
            y_preds[:, i] = prediction

        y_pred = []
        # For each sample
        for sample_predictions in y_preds:
```

```
            # Select the most common class prediction
            y_pred.append(np.mean(sample_predictions))
        return y_pred
```
#Main Function
```
if __name__ == '__main__':
    x = data.drop('Life expectancy ', axis=1)
    y = data['Life expectancy ']

    from sklearn.model_selection import train_test_split

    X_train, X_test, y_train, y_test = train_test_split(x, y, test_size=0.3)

    DRT = RandomForestRegressor(n_estimators=5, max_depth = 15,min_samples_split = 10,
max_samples=500, max_features=15)

    DRT.fit(X_train, y_train)
y_pred = DRT.predict(X_test)
```

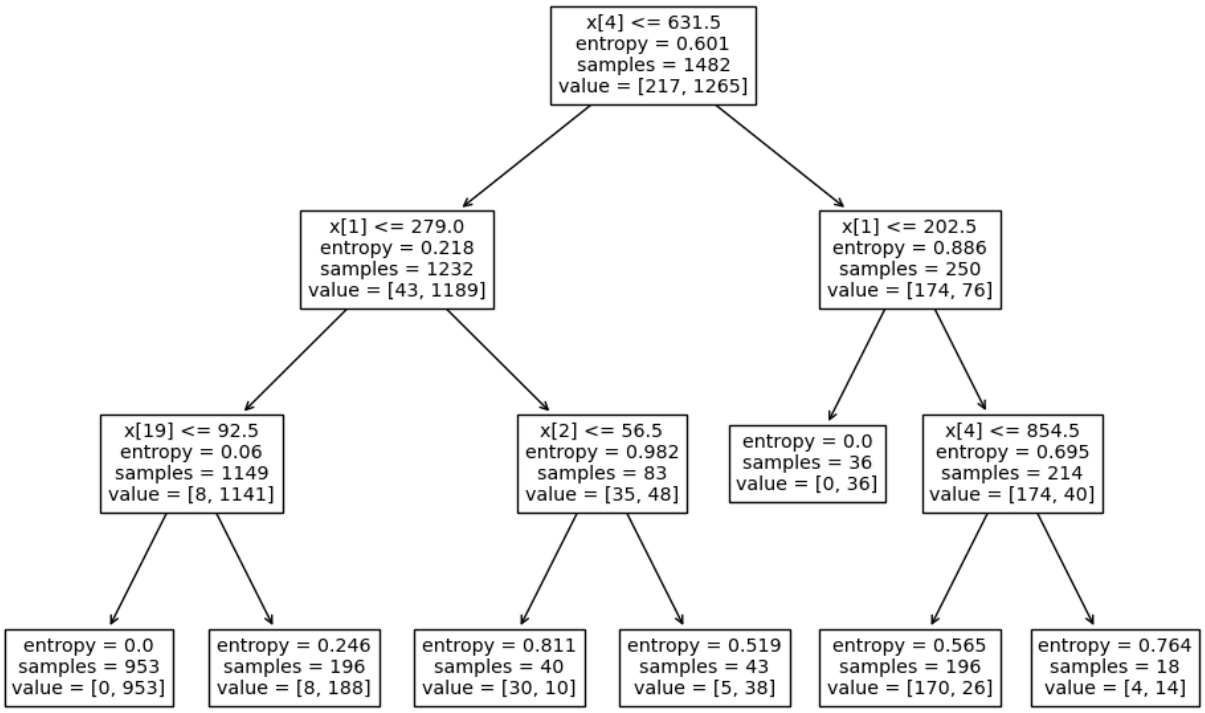
#DataFrame of y_pred and y_test values

```
finalData_1 = pd.DataFrame({'Actual': y_test, 'Predicted': y_pred})
finalData_1.head()
```

#Mean Squared Error

```
n = len(y_test)
mse = y_test - y_pred
mse = mse ** 2
mse = np.sum(mse)
mse = mse / n
print('Mean Squared Error :', mse)
```

Mean Squared Error : 5.715606463320461



4.5 K Nearest Neighbours

```
from sklearn.neighbors import KNeighborsRegressor

knn = KNeighborsRegressor()

param_grid = {
        'n_neighbors':[1,2,3,4,5,6,7,8,9],
        'weights':['uniform', 'distance'],
        'algorithm':['auto', 'ball_tree', 'kd_tree', 'brute'],
        'p':[1,2],
         }
grid = HalvingGridSearchCV(knn, param_grid, refit = True, verbose = -1, n_jobs=-1,
return_train_score = True)
```

```
knn_best_model = grid.fit(X_train, y_train)


print(grid.best_params_)


#print(grid.refit_time_)
```

n_iterations: 5

n_required_iterations: 5

n_possible_iterations: 5

min_resources_: 24

max_resources_: 1951

aggressive_elimination: False

factor: 3

----

iter: 0

n_candidates: 144

n_resources: 24

----

iter: 1

n_candidates: 48

n_resources: 72

----

iter: 2

n_candidates: 16

n_resources: 216

----

iter: 3

n_candidates: 6

n_resources: 648

----

iter: 4

n_candidates: 2

n_resources: 1944

{'algorithm': 'auto', 'n_neighbors': 8, 'p': 1, 'weights': 'distance'}

```
y_train_pred = knn_best_model.predict(X_train)

y_test_pred = knn_best_model.predict(X_test)


print("The training R^2 score for best KNN model is",r2_score(y_train, y_train_pred))

print()

print("The training RMSE score for best KNN model
is",math.sqrt(mean_squared_error(y_train_pred,y_train)))
```

The training R^2 score for best KNN model is 1.0

The training RMSE score for best KNN model is 0.0

Kernel Ridge Regression

```
from sklearn.kernel_ridge import KernelRidge


krr = KernelRidge()


param_grid = {
        "alpha": [1e0, 0.1, 1e-2, 1e-3],
        "gamma": np.logspace(-2, 2, 5),
        "degree": [3, 5, 7, 9]
         }
grid = HalvingGridSearchCV(krr, param_grid, refit = True, verbose = -1, n_jobs=-1,
return_train_score = True)


krr_best_model = grid.fit(X_train, y_train)


print(grid.best_params_)
```

#print(grid.refit_time_)

n_iterations: 4

n_required_iterations: 4

n_possible_iterations: 4

min_resources_: 72

max_resources_: 1951

aggressive_elimination: False

factor: 3

----

iter: 0

n_candidates: 80

n_resources: 72

----

iter: 1

n_candidates: 27

n_resources: 216

----

iter: 2

n_candidates: 9

n_resources: 648

----

iter: 3

n_candidates: 3

n_resources: 1944

{'alpha': 1.0, 'degree': 5, 'gamma': 1.0}

y_train_pred = krr_best_model.predict(X_train)

y_test_pred = krr_best_model.predict(X_test)

print("The training R^2 score for best Kernel Ridge model is",r2_score(y_train,

```
y_train_pred))
print()
print("The training RMSE score for best Kernel Ridge model
is",math.sqrt(mean_squared_error(y_train_pred,y_train)))
```

The training R^2 score for best Kernel Ridge model is -7.28159148691805

The training RMSE score for best Kernel Ridge model is 74989.72782867817

 # # Note: Model perform badly when shuffle is off

4.6 SVM

```
from sklearn.svm import SVR

svr = SVR()

param_grid = {
        'kernel': ['linear','poly', 'rbf', 'sigmoid'],
        'C': [10000,50000,100000,500000],
        'gamma': [1, 0.1, 0.01, 0.001, 0.0001],
        'gamma':['scale', 'auto'],
        }
grid = HalvingGridSearchCV(svr, param_grid, refit = True, verbose = -1,n_jobs=-1)

svr_best_model = grid.fit(X_train, y_train)

print(grid.best_params_)
```

#print(grid.refit_time_)

n_iterations: 4

n_required_iterations: 4

n_possible_iterations: 4

min_resources_: 72

max_resources_: 1951

aggressive_elimination: False

factor: 3

----

iter: 0

n_candidates: 32

n_resources: 72

----

iter: 1

n_candidates: 11

n_resources: 216

----

iter: 2

n_candidates: 4

n_resources: 648

----

iter: 3

n_candidates: 2

n_resources: 1944

{'C': 100000, 'gamma': 'scale', 'kernel': 'rbf'}

```
y_train_pred = svr_best_model.predict(X_train)
y_test_pred = svr_best_model.predict(X_test)


print("The training R^2 score for best SVM model is",r2_score(y_train, y_train_pred))
print()
print("The training RMSE score for best SVM model
is",math.sqrt(mean_squared_error(y_train_pred,y_train)))
```

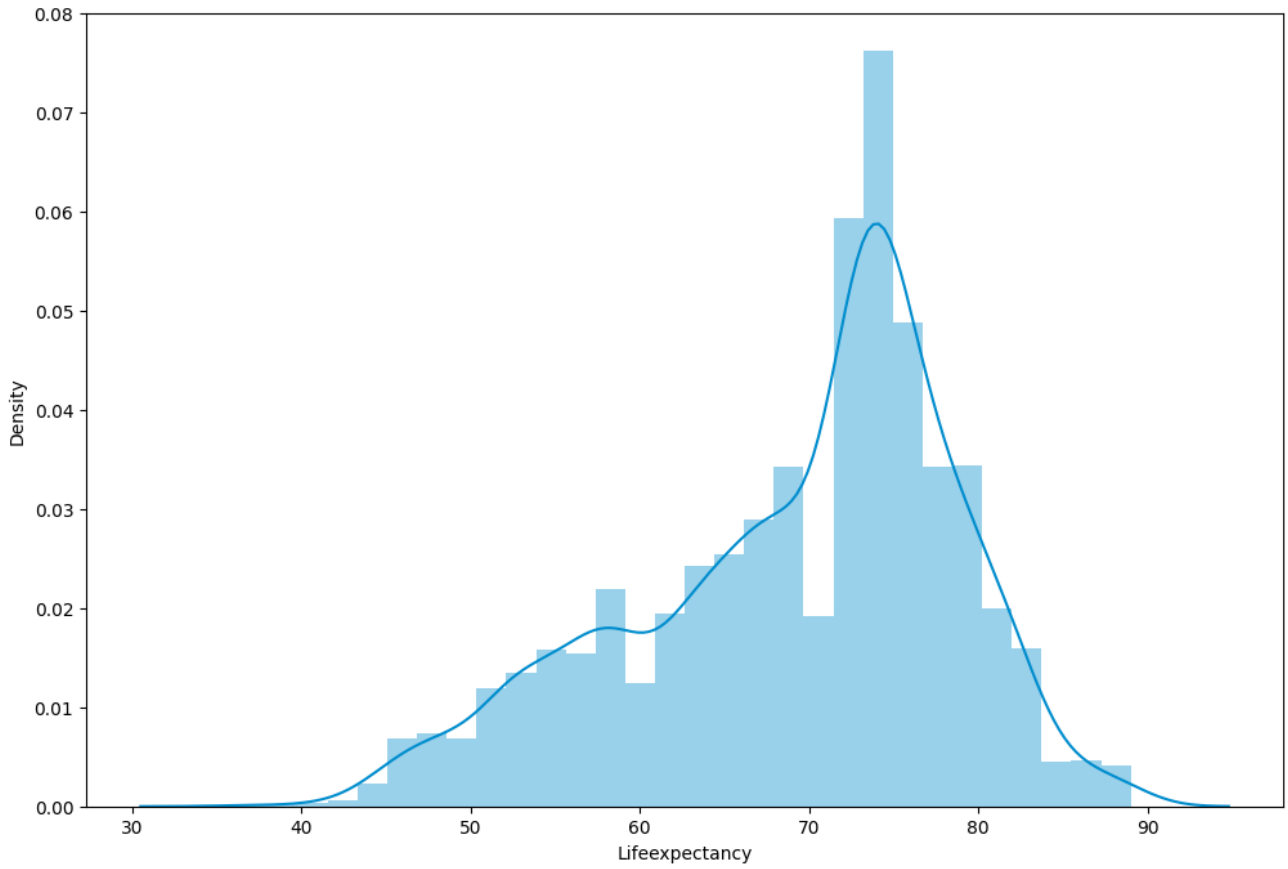
The training R^2 score for best SVM model is 0.9526227318860632

The training RMSE score for best SVM model is 5671.917205915255

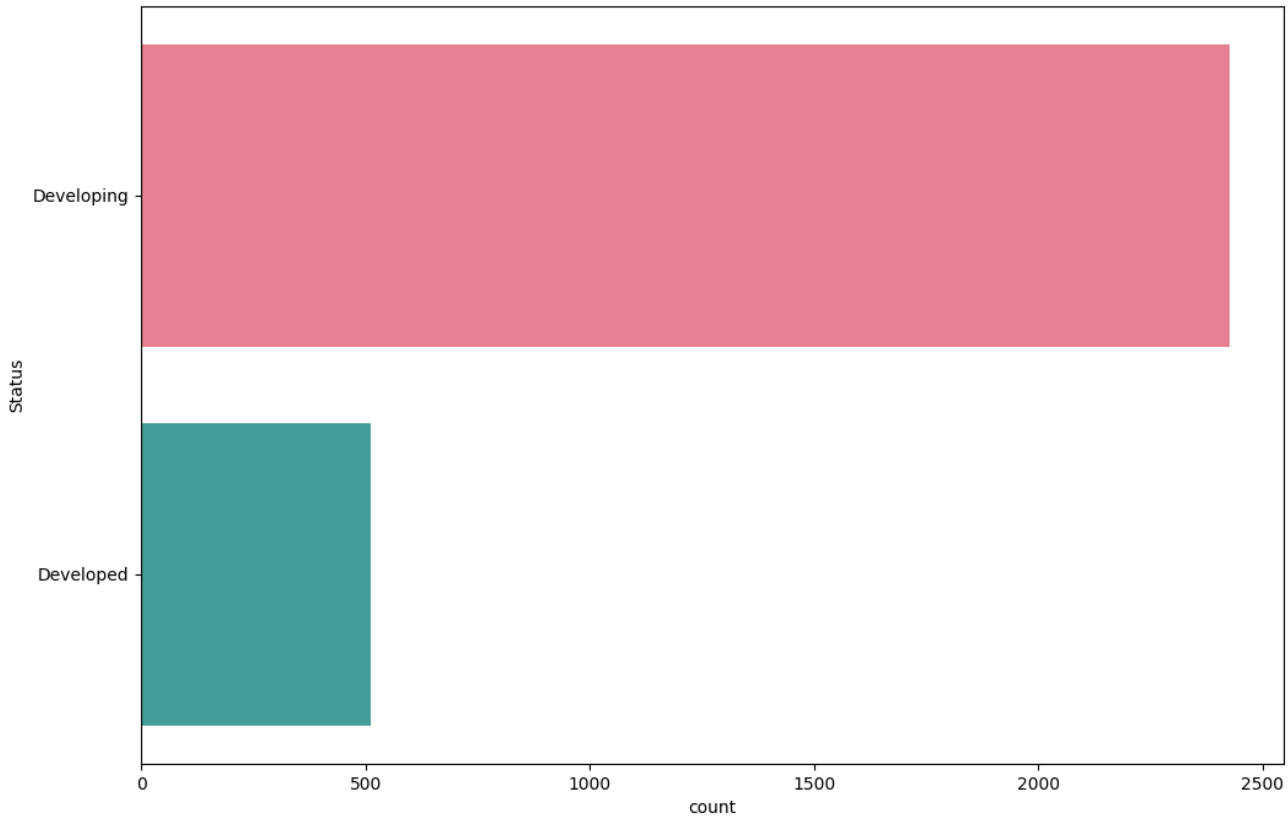
4.7 ARTIFICIAL NEURAL NETWORK

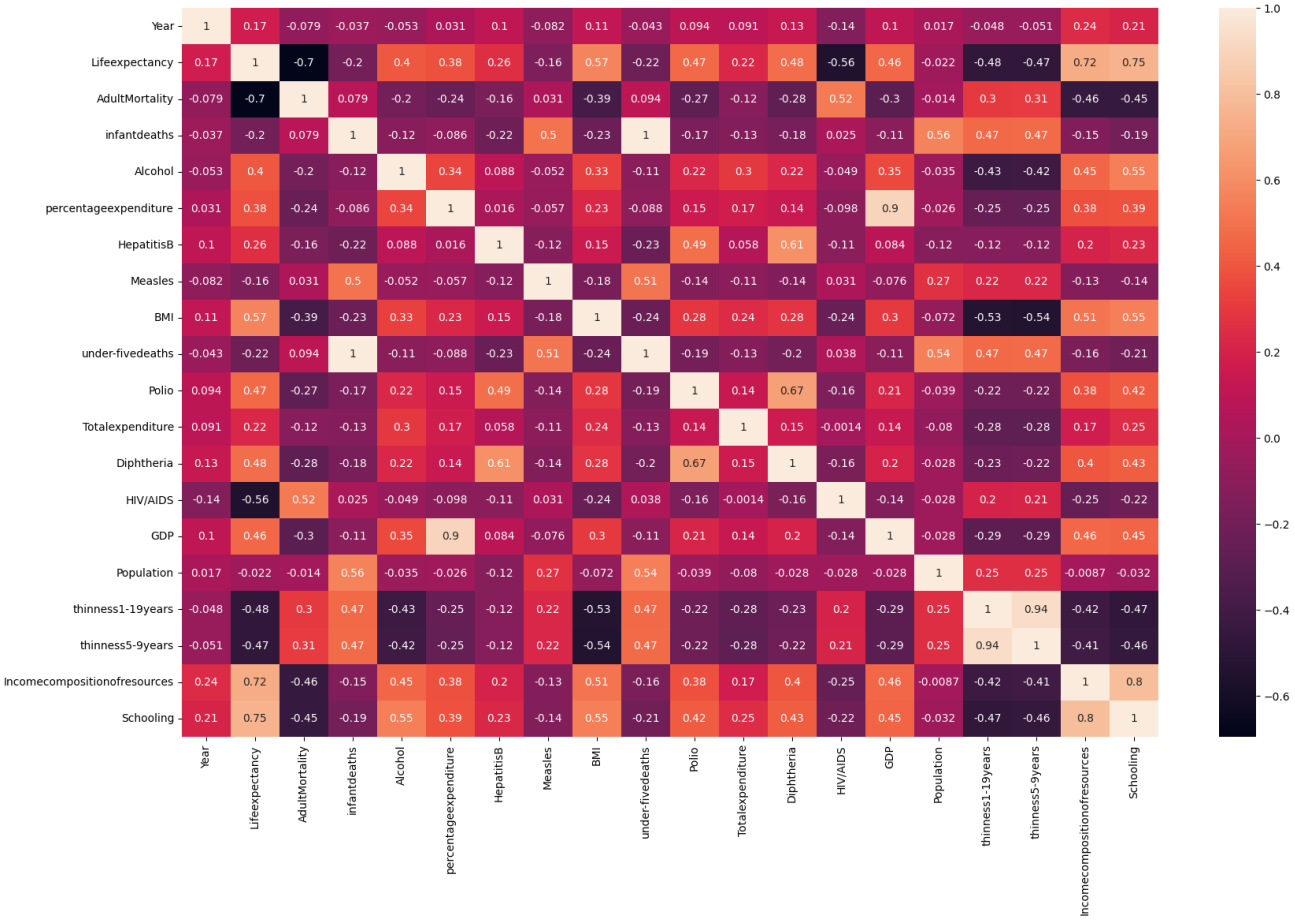
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import random
# data preprocessing
from sklearn.preprocessing import LabelEncoder
from sklearn.preprocessing import MinMaxScaler
from sklearn.model_selection import train_test_split
# data modeling
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Activation
from tensorflow.keras.optimizers import Adam
from tensorflow.keras import regularizers
# evaluation
from sklearn.metrics import mean_absolute_error,mean_squared_error

# hide warnings
import warnings
warnings.filterwarnings('ignore')
#/content/led.csv
data = pd.read_csv('/content/led.csv')
data
data.isnull().sum()/data.shape[0]*100
data.columns
```

```
data.describe().transpose()
data.describe(include="object").transpose()
def rand_color():
  c = "#"+"".join(map(hex,random.choices(range(256), k=3))).replace("0x","")
  if len(c) != 7:
    return rand_color()
  return c
obj_cols = data.select_dtypes(include='object').columns.tolist()
num_cols = data.select_dtypes(exclude='object').columns.drop("Lifeexpectancy").tolist()
plt.figure(figsize=(12,8))
sns.distplot(data['Lifeexpectancy'], color="#008dce")
```



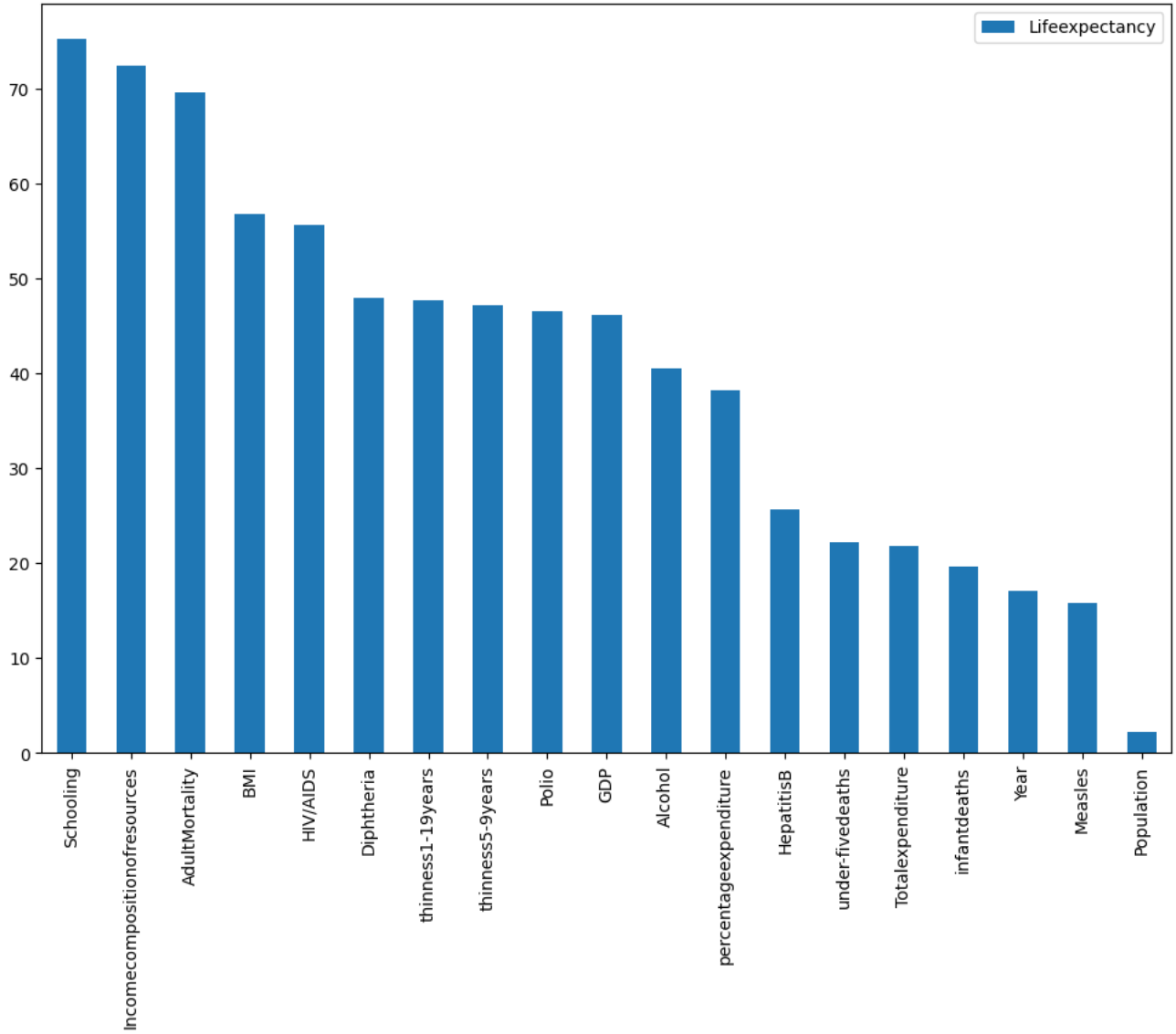

```
for i in num_cols:
  print("\n")
  plt.figure(figsize=(12,8))
  sns.distplot(data[i], color = rand_color()).set(title = i)
  plt.show()
```





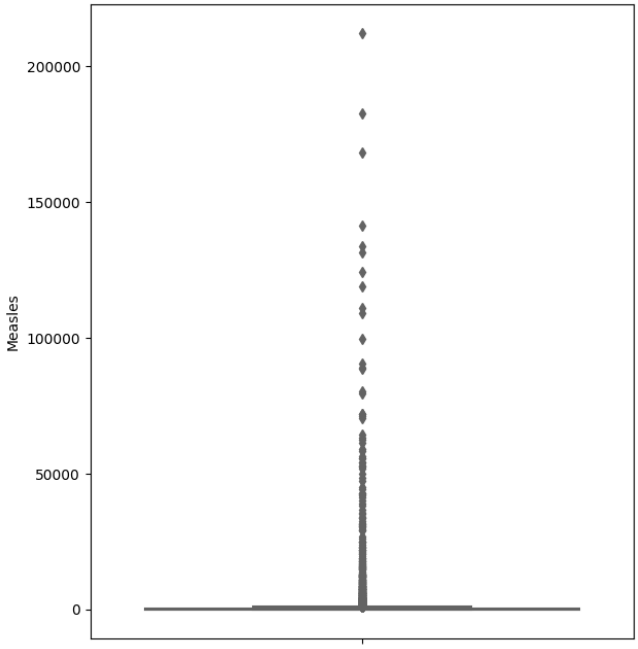
Let's answer the question: Using the heat map presented above we can see a strong negative correlation between life expectancy and disease, child and adult mortality. Strong positive correlations with life expectancy are found for factors such as hepatitis B, polio and diphtheria

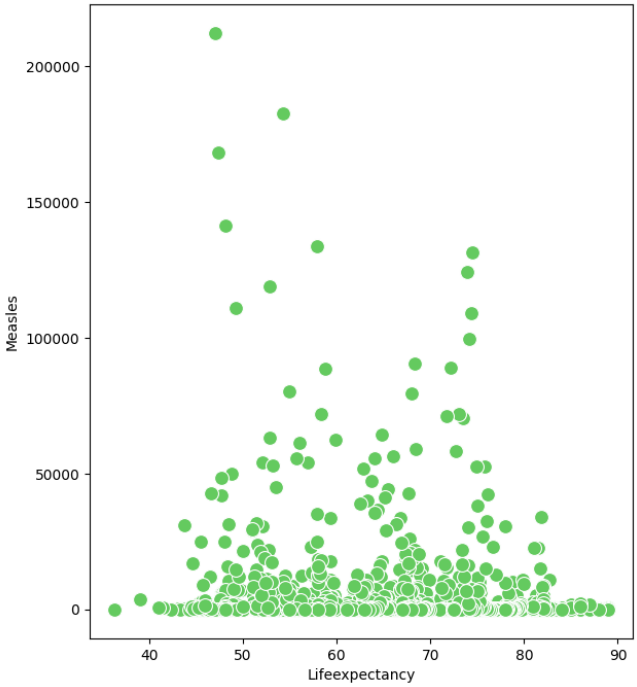
vaccination. Hence, longer life expectancy in vaccinated people.Let's look at the graphs that interest us the most: 1) Alcohol </br> 2) Year </br> 3) Infant Mortality </br> 4) Adult Mortality </br> 5) Schooling </br> 6) Total Income </br> 7) Population </br> 8) GDP </br> 9) BMI </br>



for col in num_cols:

```
    print("\n")
    fig, ax =plt.subplots(1,2, figsize=(15,8))
    sns.boxplot(data=data, y=col, ax=ax[0], color=rand_color())
    sns.scatterplot(data=data,x = 'Lifeexpectancy', s = 100, y=col, ax=ax[1], color = rand_color())
    plt.show()
```





mv_cols = ["Life expectancy ", "Adult Mortality", "Alcohol", "Hepatitis B", " BMI ", "Polio", "Total expenditure", "Diphtheria ", "GDP", "Population", " thinness 1-19 years", " thinness 5-9 years", "Income composition of resources", "Schooling"]

#Since, only a few rows have null values in them, we are only removing those rows from the dataset.

data = data.dropna(subset=['GDP','BMI','Lifeexpectancy','AdultMortality','Population','Alcohol','HepatitisB','Polio','Totalexpenditure','Diphtheria'])

RELU ACTIVATION FUNCTION

total_null_values = data.isnull().sum().sort_values(ascending = False)

percentage = ((data.isnull().sum()/data.isnull().count())*100).sort_values(ascending = False)

print("Total values present are ",data.shape[0])

total_missing_data = (pd.concat([total_null_values,percentage.round(2)],axis=1,keys=['Total Missing','In precentage']))

total_missing_data

data.isnull().sum()/data.shape[0]*100

status_le = LabelEncoder()

country_le = LabelEncoder()

```
data['Status'] = status_le.fit_transform(data['Status'])
data['Country'] = country_le.fit_transform(data['Country'])
x = data.drop('Lifeexpectancy',axis=1)
y = data['Lifeexpectancy']
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.25, random_state=42)
scaler = MinMaxScaler()
x_train_scaled = scaler.fit_transform(x_train)
x_test_scaled = scaler.transform(x_test)
x_train_scaled.shape

model = Sequential()

model.add(Dense(28,input_shape=(21,),activation='relu',
kernel_regularizer=regularizers.L1L2(l1=1e-3, l2=1e-2),
    bias_regularizer=regularizers.L2(1e-3),
    activity_regularizer=regularizers.L2(1e-4)))

#Hidden Layer
model.add(Dense(56,activation='relu'))
model.add(Dense(42,activation='relu'))
model.add(Dense(35,activation='relu'))
model.add(Dense(20,activation='relu'))

#Output layer

model.add(Dense(1))
```

REGULARIZATION(L1,L2)

Ridge Regression

```
gs_rr = GridSearchCV(Ridge(),
                     param_grid={
                        'alpha':[0.1, 0.3, 1, 3, 6, 8, 10]
```

```
        }, verbose=1)

gs_rr.fit(X_Train, y_train)
rr=gs_rr.best_estimator_
```

Fitting 5 folds for each of 7 candidates, totalling 35 fits

[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.

[Parallel(n_jobs=1)]: Done 35 out of 35 |

```
print(f"Explained Variance Score: {explained_variance_score(y_pred=rr.predict(X_Train),
y_true=y_train)}")
print(f"RMSE: {np.sqrt(metrics.mean_squared_error(y_pred=rr.predict(X_Train),
y_true=y_train))}")
```

Explained Variance Score: 0.8259764164206265

RMSE: 4.006915738152002

Lasso Regression

```
gs_lr = GridSearchCV(Lasso(),
        param_grid={
            'alpha': [0.1, 0.3, 1, 3, 6, 8, 10]
        }, verbose=1)

gs_lr.fit(X_Train, y_train)
```

[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.

Fitting 5 folds for each of 7 candidates, totalling 35 fits

[Parallel(n_jobs=1)]: Done 35 out of 35 |

```
GridSearchCV(estimator=Lasso(),
        param_grid={'alpha': [0.1, 0.3, 1, 3, 6, 8, 10]}, verbose=1)
lasso = gs_lr.best_estimator_
print(f"Explained Variance Score: {explained_variance_score(y_pred=lasso.predict(X_Train),
```

y_true=y_train)}")

print(f"RMSE: {np.sqrt(metrics.mean_squared_error(y_pred=lasso.predict(X_Train), y_true=y_train))}")

Explained Variance Score: 0.8253886864025879

RMSE: 4.013676315047004

OPTIMIZATION(ADAM)

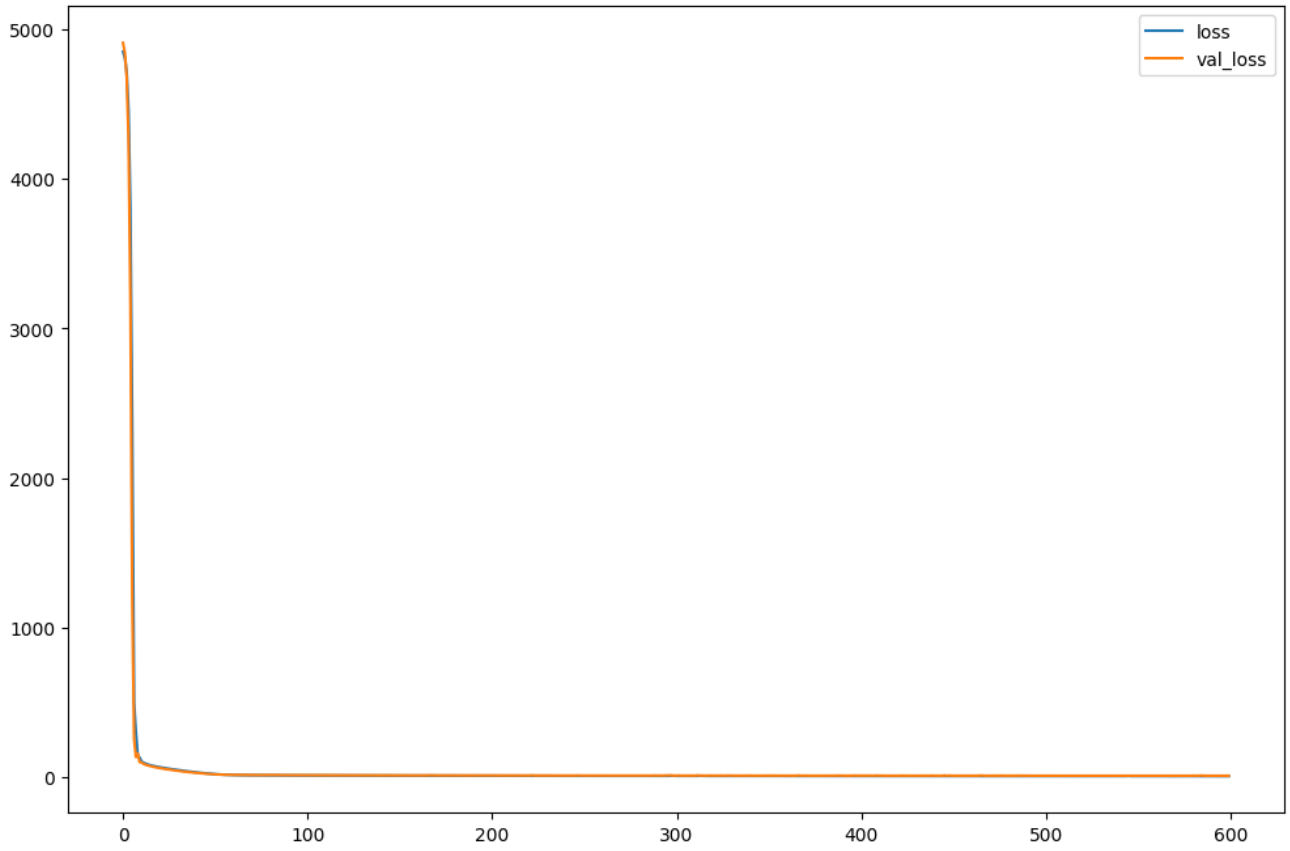
model.compile(optimizer='Adam',loss='mse',metrics=['mae'])

model.fit(x=x_train_scaled,y=y_train.values,validation_data=(x_test_scaled,y_test.values),batch_size=128,epochs=600)

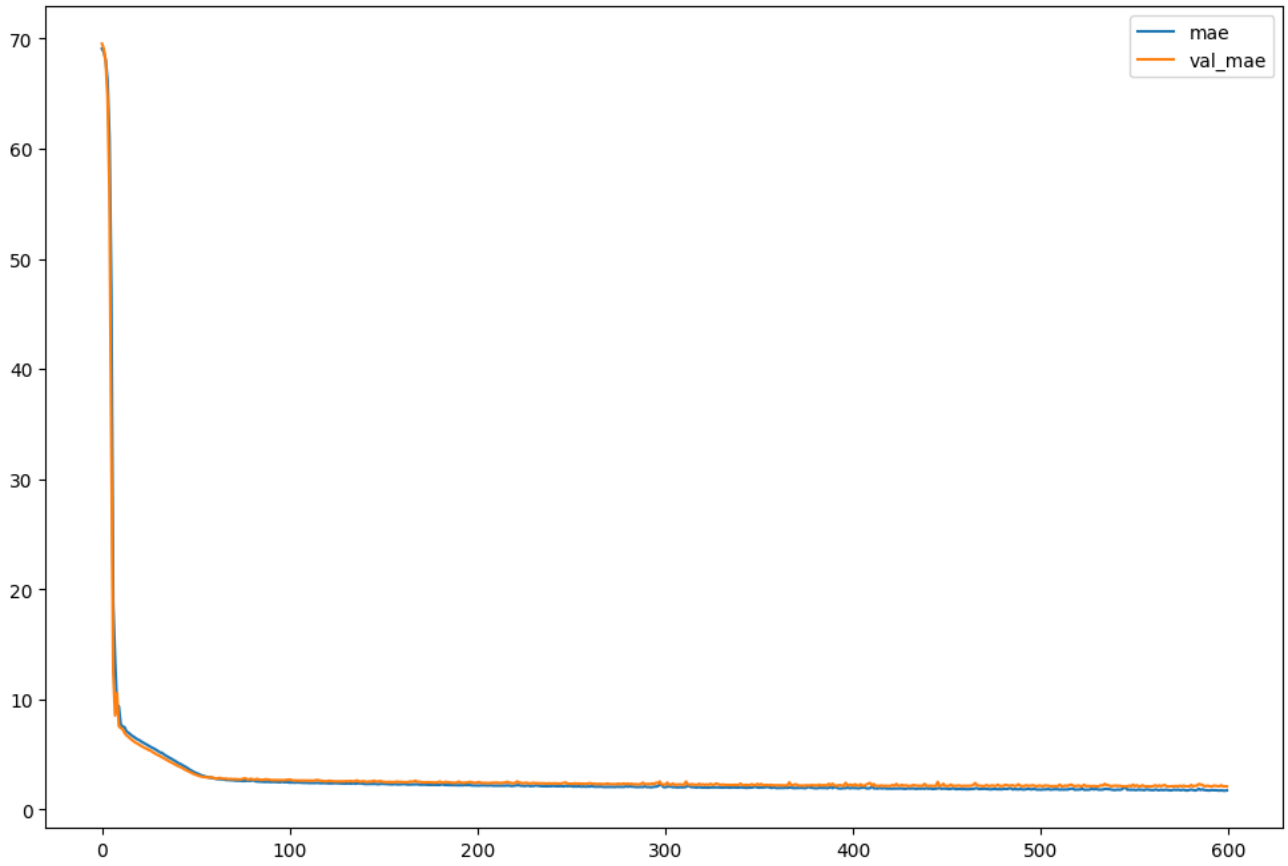
losses=pd.DataFrame(model.history.history)

losses.head()

losses[['loss','val_loss']].plot(figsize=(12,8))



losses[['mae','val_mae']].plot(figsize=(12,8))



MEAN ABSOLUTE ERROR

predictions = model.predict(x_test_scaled)

mean_abs_error = mean_absolute_error(predictions,y_test)

mean_abs_error

2.1051765663571858

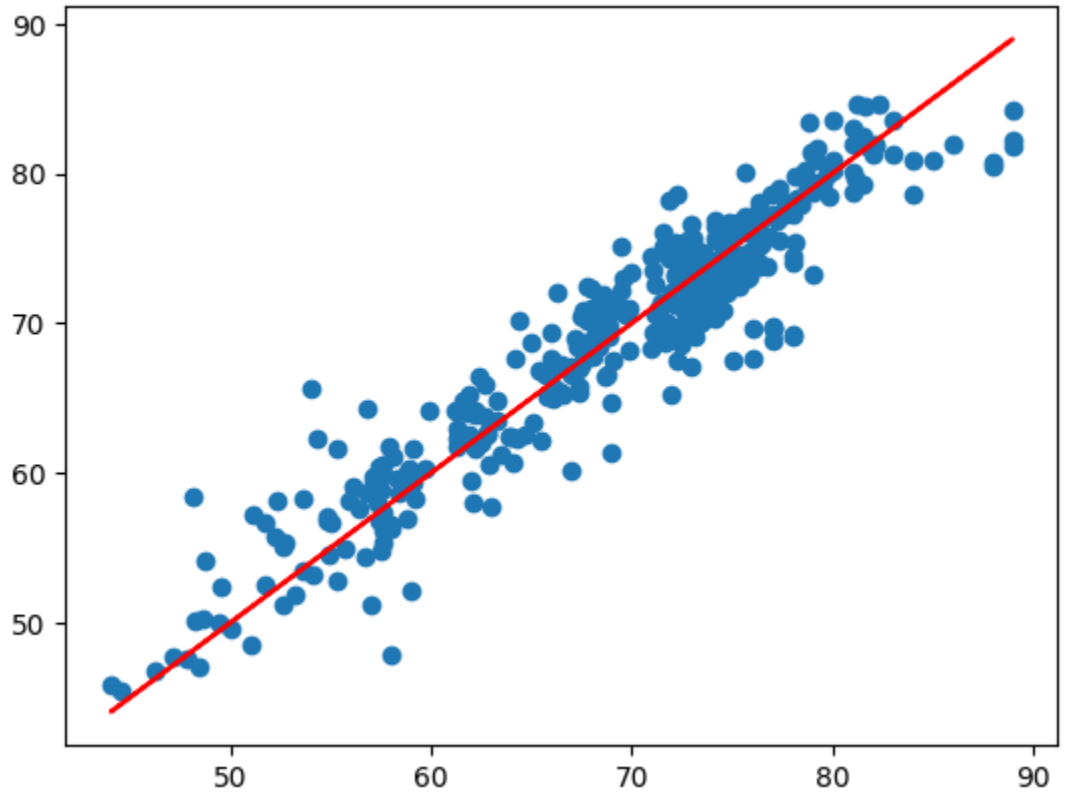
np.sqrt(mean_squared_error(y_test,predictions))

2.8662676453965132

MEAN SQUARE ERROR

model.evaluate(x_test_scaled,y_test)

```
plt.scatter(y_test,predictions)
plt.plot(y_test,y_test,'r')
```

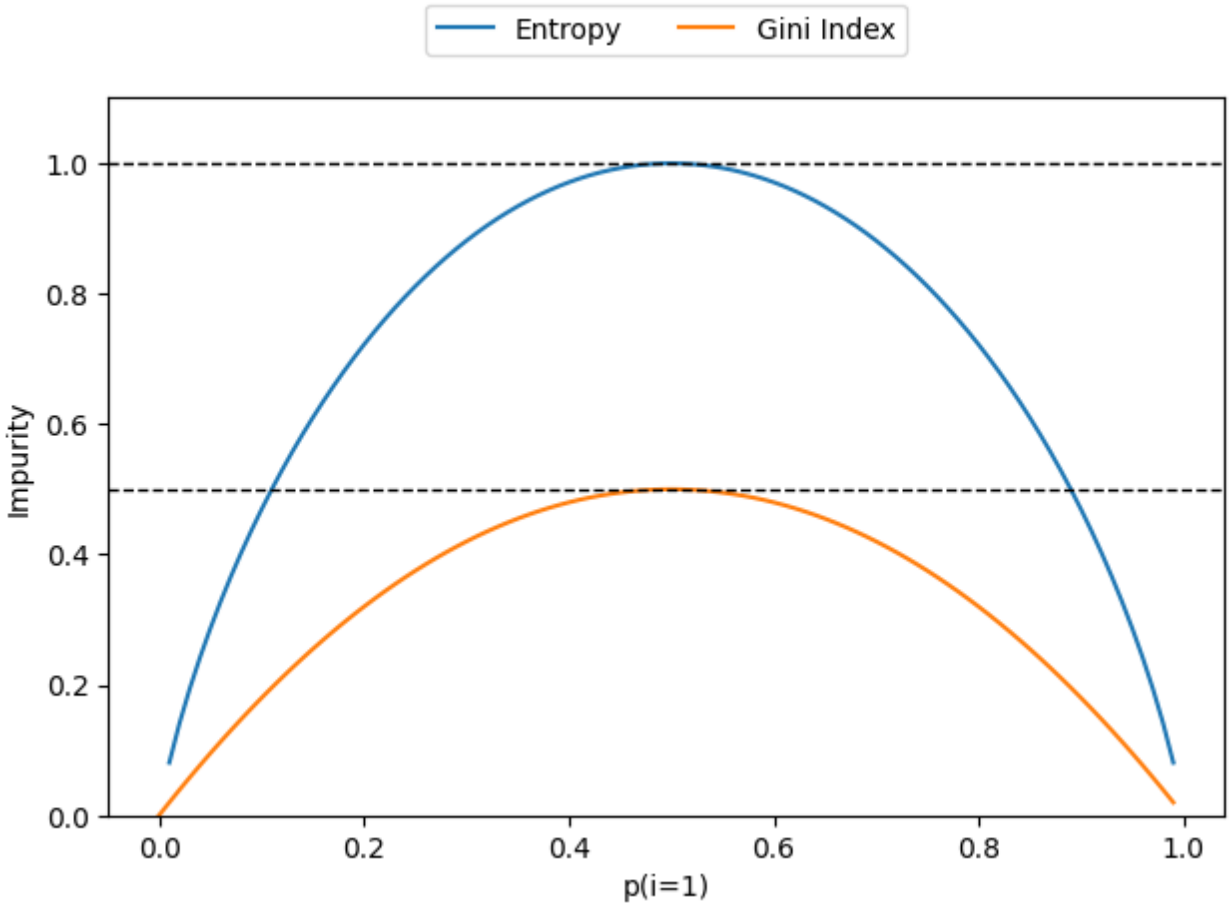


5.UNSUPERVISED LEARNING

5.1 K MEANS

```
# Visualization of Entropy and Gini index used for Node splitting
# here we implement Entropy and Gini index from scratch
import pandas as pd
import numpy as np

def entropy(p):
    return - p * np.log2(p) - (1 - p) * np.log2(1 - p)



def gini(p):
    return p * (1 - p) + (1 - p) * ( 1 - (1 - p) )



# generate some data
x = np.arange(0, 1., 0.01)
print(x)
# compute entropy and Gini index
ent = [entropy(p) if p != 0 else None for p in x]
gi = gini(x)
# plot entropy and gini
import matplotlib.pyplot as plt
plt.plot(x, ent, label = 'Entropy')
plt.plot(x, gi, label = 'Gini Index')
plt.legend(loc = 'upper center', bbox_to_anchor = (0.5, 1.15),
           ncol = 3, fancybox = True, shadow = False)
plt.axhline(y = 0.5, linewidth = 1, color = 'k', linestyle = '--')
plt.axhline(y = 1.0, linewidth = 1, color = 'k', linestyle = '--')
```

```
plt.ylim([ 0, 1.1 ])
plt.xlabel('p(i=1)')
plt.ylabel('Impurity')
plt.tight_layout()
plt.show()
```




```
# check the number of samples in each clases and plot them as bar chart
df['Status'].value_counts().sort_values().plot(kind = 'barh')
# lets seperate out the feature vectors and the class variables
X=df.drop(columns=['Status'])
Y=df.Status
# to ensure proper split:
df.shape,X.shape,Y.shape
from sklearn.preprocessing import OrdinalEncoder
from sklearn import tree
clf = tree.DecisionTreeClassifier()
clf = clf.fit(X,Y)
```

```
clf.score(X,Y)
# instantiate the DecisionTreeClassifier model with criterion gini index

clf_gini = DecisionTreeClassifier(criterion='gini', max_depth=3, random_state=0)



# fit the model
clf_gini.fit(X_train, y_train)
y_pred_gini = clf_gini.predict(X_test)
from sklearn.metrics import accuracy_score

print('Model accuracy score with criterion gini index: {0:0.4f}'. format(accuracy_score(y_test, y_pred_gini)))
```

Model accuracy score with criterion gini index: 0.927

```
y_pred_train_gini = clf_gini.predict(X_train)

y_pred_train_gini
print('Training-set accuracy score: {0:0.4f}'. format(accuracy_score(y_train, y_pred_train_gini)))
```

Training-set accuracy score: 0.9669

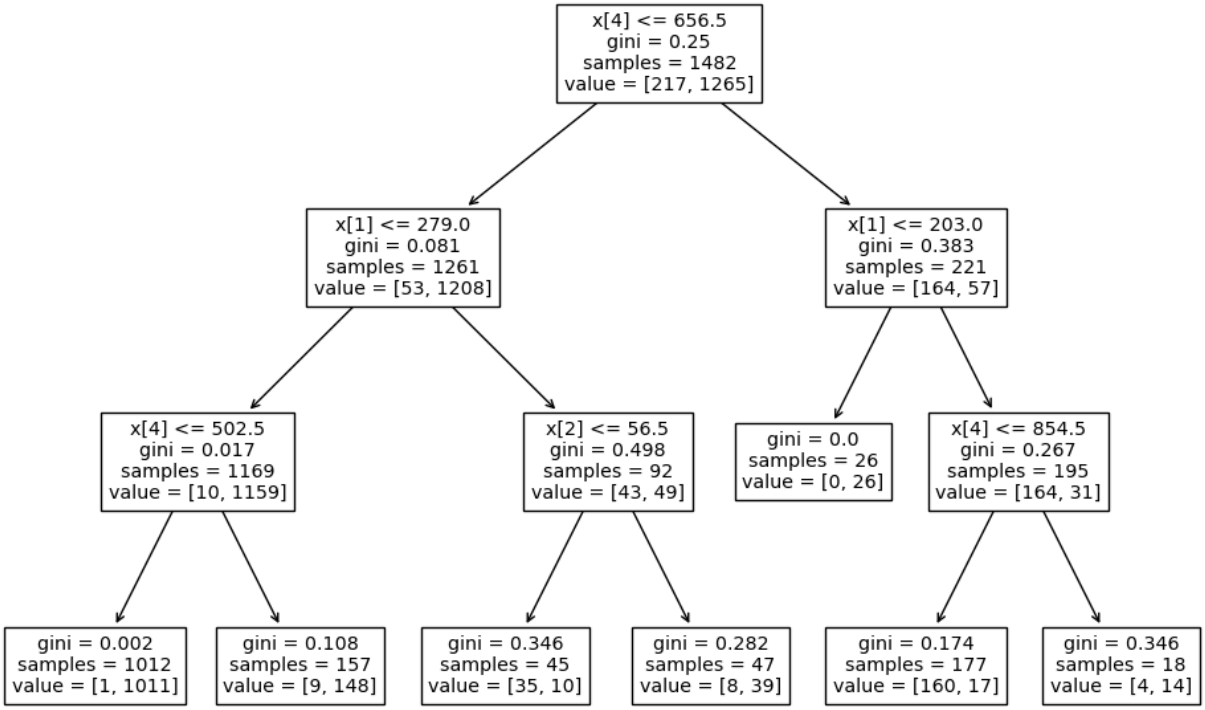
```
# print the scores on training and test set

print('Training set score: {:.4f}'.format(clf_gini.score(X_train, y_train)))

print('Test set score: {:.4f}'.format(clf_gini.score(X_test, y_test)))
```

Training set score: 0.9669

Test set score: 0.9272

```
plt.figure(figsize=(12,8))

from sklearn import tree
```

tree.plot_tree(clf_gini.fit(X_train, y_train))



#K-MEANS

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import matplotlib.ticker as ticker

import seaborn as sns

import math

from scipy import stats

from sklearn.preprocessing import StandardScaler

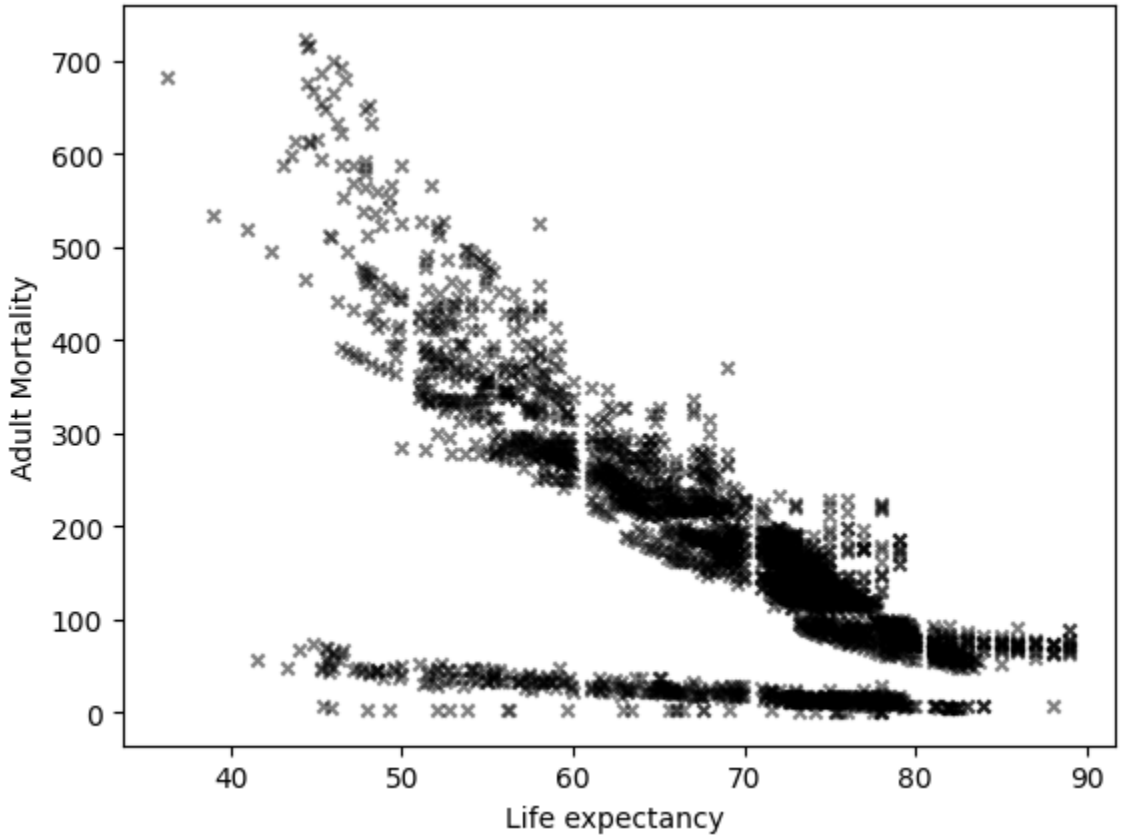
from sklearn.decomposition import PCA

from scipy.stats.mstats import winsorize

from sqlalchemy import create_engine

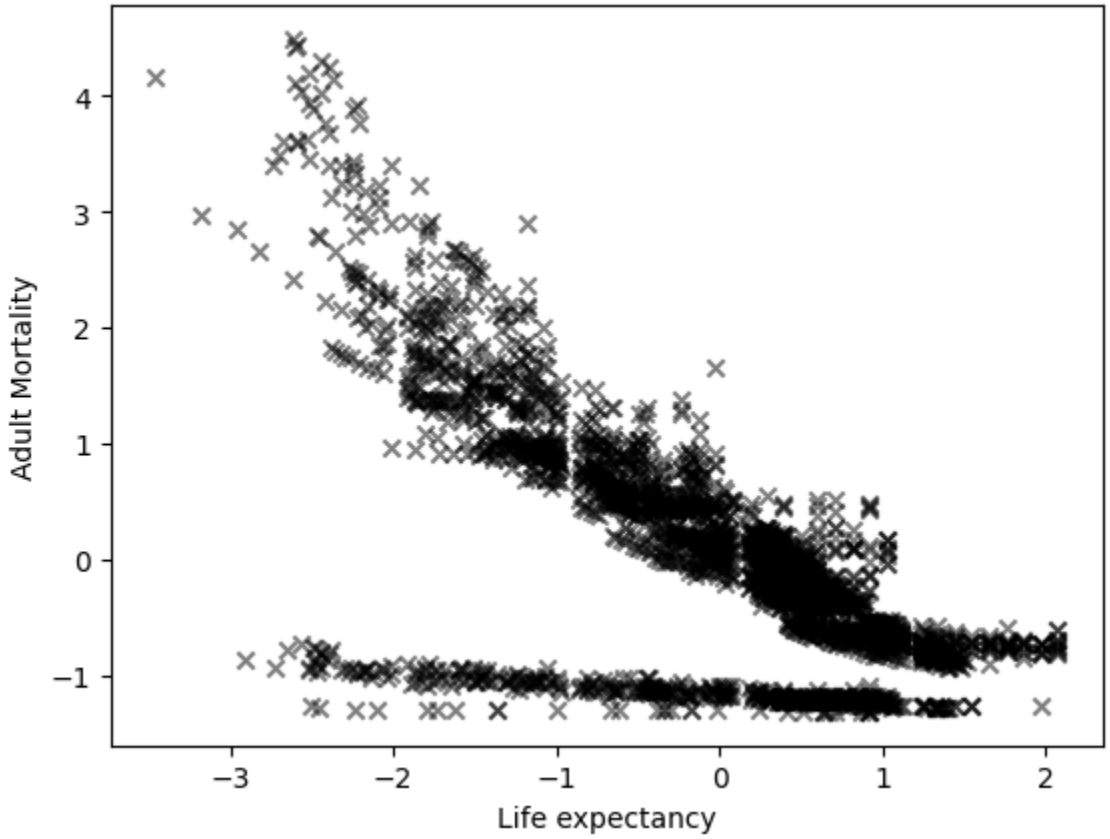
import warnings

```
import plotly.express as px
data = pd.read_csv('/content/Life Expectancy Data.csv')

X_train = data[["Life expectancy ", "Adult Mortality"]]
X_train.plot.scatter(x="Life expectancy ", y="Adult Mortality",
                     c="black", marker="x", alpha=.5)
```




```
from pandas.core.common import random_state
from sklearn.preprocessing import StandardScaler


RANDOM_STATE = 15

def data_standardization(X_train):
    scaler = StandardScaler()
    scaler.fit(X_train)
```

```
    X_train_std = scaler.transform(X_train)

    return X_train_std

X_train_std = data_standardization(X_train)

plt.scatter(X_train_std[:, 0], X_train_std[:, 1], c="black", marker="x", alpha=.5)

plt.xlabel("Life expectancy")
plt.ylabel("Adult Mortality")
plt.show()
```



5.2 PCA

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.ticker as ticker
import seaborn as sns
```

```
import math
from scipy import stats
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
from scipy.stats.mstats import winsorize
from sqlalchemy import create_engine
import warnings

import plotly.express as px
import pandas as pd
le_df = pd.read_csv("/content/Life Expectancy Data.csv")
le_df.info()

#output cut out some of the columns. hacked it.
le_df[:5].T.head(22)
# I'm going to adjust the columns names and remove extra spacing for ease of use
le_df.rename(columns = lambda x: x.strip().replace(' ', '_').lower(), inplace=True)

# one column doesn't match our underscoring convention or the Kaggle description. fixing.
le_df.rename(columns = {'thinness__1-19_years':'thinness_10-19_years'}, inplace=True)

print((f'Number of columns: {len(le_df.columns)}'))
le_df.columns
# that's better
```

Number of columns: 22

Index(['country', 'year', 'status', 'life_expectancy', 'adult_mortality',

'infant_deaths', 'alcohol', 'percentage_expenditure', 'hepatitis_b',

'measles', 'bmi', 'under-five_deaths', 'polio', 'total_expenditure',

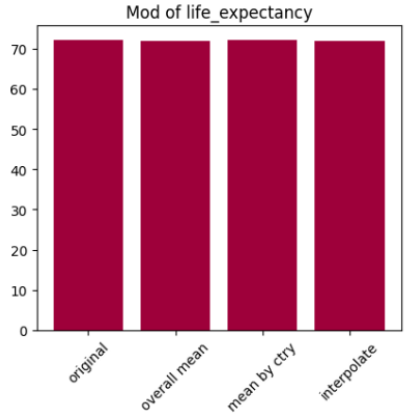
'diphtheria', 'hiv/aids', 'gdp', 'population', 'thinness_10-19_years',

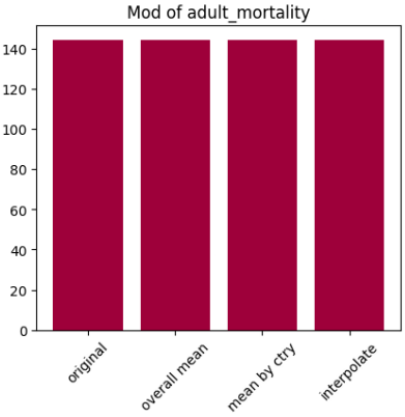
'thinness_5-9_years', 'income_composition_of_resources', 'schooling'],

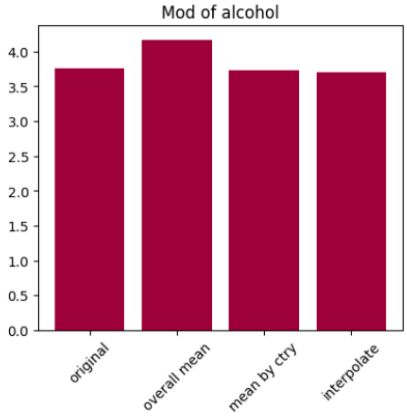
dtype='object')

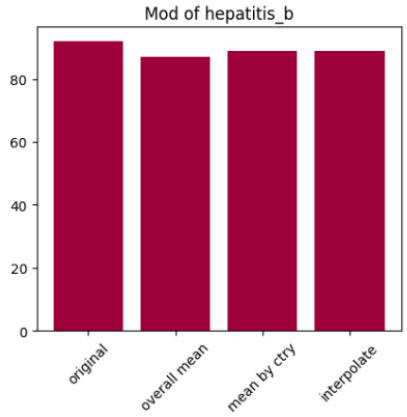
```
# checking for nans
le_df.isnull().sum()

# will fill nans multiple ways to see what approach is best
le_df2 = le_df.copy() # will fill by mean
le_df3 = le_df.copy() # will fill mean by country
le_df4 = le_df.copy() # will fill by interpolation

countries = le_df2['country'].unique()
na_cols = ['life_expectancy', 'adult_mortality', 'alcohol', 'hepatitis_b',
       'bmi', 'polio', 'total_expenditure','diphtheria', 'gdp', 'population',
        'thinness_10-19_years', 'thinness_5-9_years',
        'income_composition_of_resources', 'schooling']

# fill with overall mean
for col in na_cols:
    le_df2[col].fillna(le_df2[col].mean(), inplace=True)

# mean by country
for col in na_cols:
    for country in countries:
        le_df3.loc[le_df3['country']== country, col] = le_df3.loc[le_df3['country'] == country, col].fillna(
            le_df3[le_df3['country'] == country][col].mean())
        # interpolated by entire df
# due to missing values, I did not interpolate by country as there are too many missing values
for col in na_cols:
    le_df4.loc[:,col] = le_df4.loc[:,col].interpolate(limit_direction='both')
```

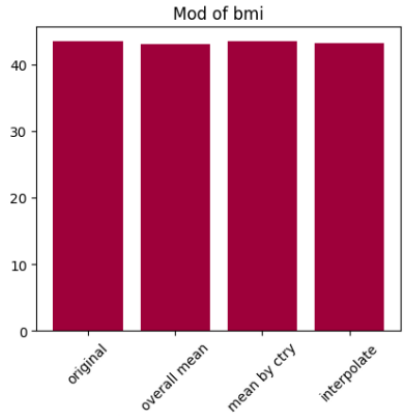
```
#printing nulls for each method
dfs = [le_df, le_df2, le_df3, le_df4]
df_names = ['le_df', 'le_df2', 'le_df3', 'le_df4']

for name, df in zip(df_names, dfs):
    print('_'*60)
    print(f'nulls for {name}')
    print('_'*60)
    print(df.isnull().sum())
#plotting each method by column
plt.figure(figsize=(13,60))

for i, col in enumerate(na_cols):
    df = pd.concat([le_df[col], le_df2[col], le_df3[col], le_df4[col]], axis=1)

    plt.subplot(len(na_cols), 3, i+1)
    plt.bar(['original', 'overall mean', 'mean by ctry', 'interpolate'],df.median(), color=('xkcd:cranberry') )
    plt.title(f'Mod of {col}')
    plt.xticks(rotation=45)

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

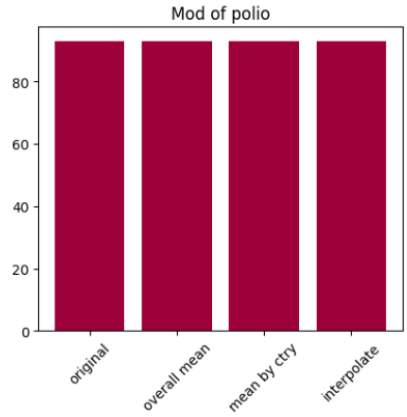

Mod of life_expectancy
original
overall mean
mean by ctry
interpolate

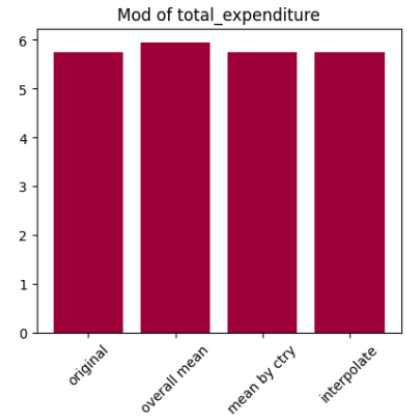

Mod of adult_mortality
original
overall mean
mean by ctry
interpolate

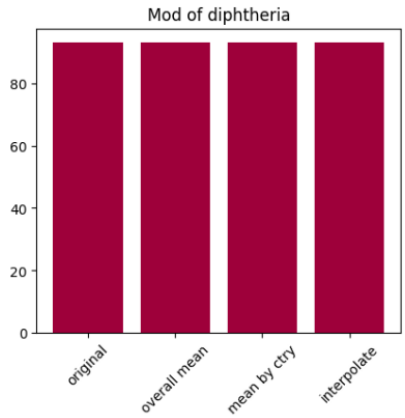

Mod of alcohol
original
overall mean
mean by ctry
interpolate

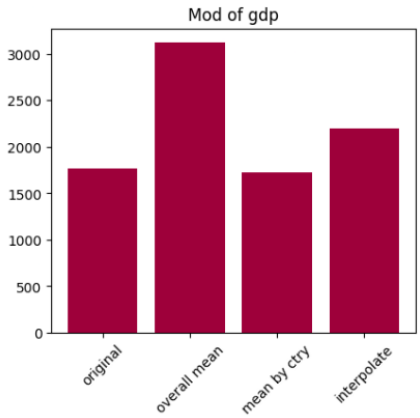

Mod of hepatitis_b
original
overall mean
mean by ctry
interpolate

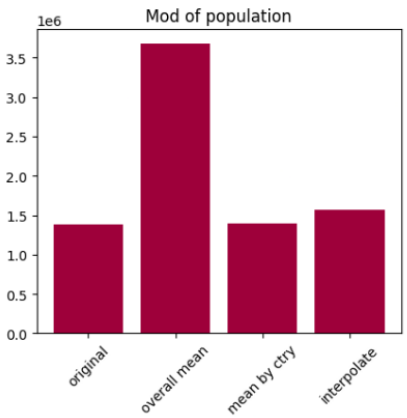

Mod of bmi
original
overall mean
mean by ctry
interpolate

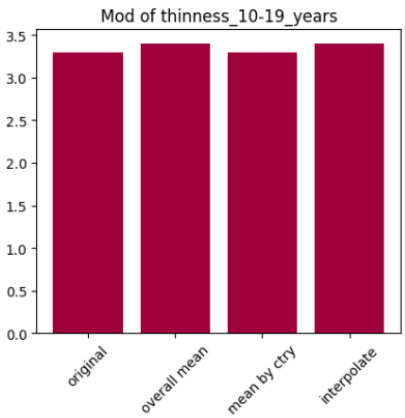

Mod of polio
original
overall mean
mean by ctry
interpolate

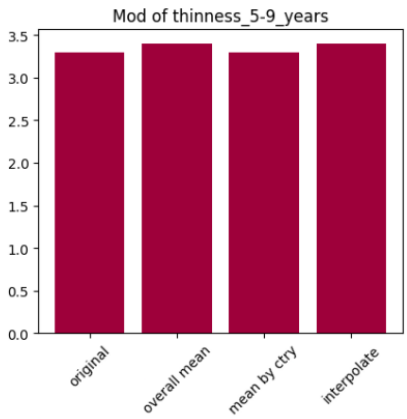

Mod of total_expenditure
original
overall mean
mean by ctry
interpolate

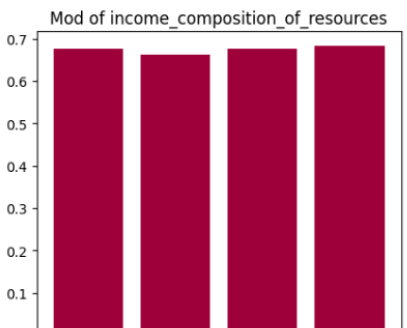

Mod of diphtheria
original
overall mean
mean by ctry
interpolate

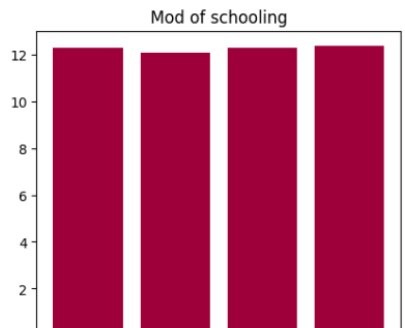

Mod of gdp
original
overall mean
mean by ctry
interpolate


1e6
Mod of population
original
overall mean
mean by ctry
interpolate

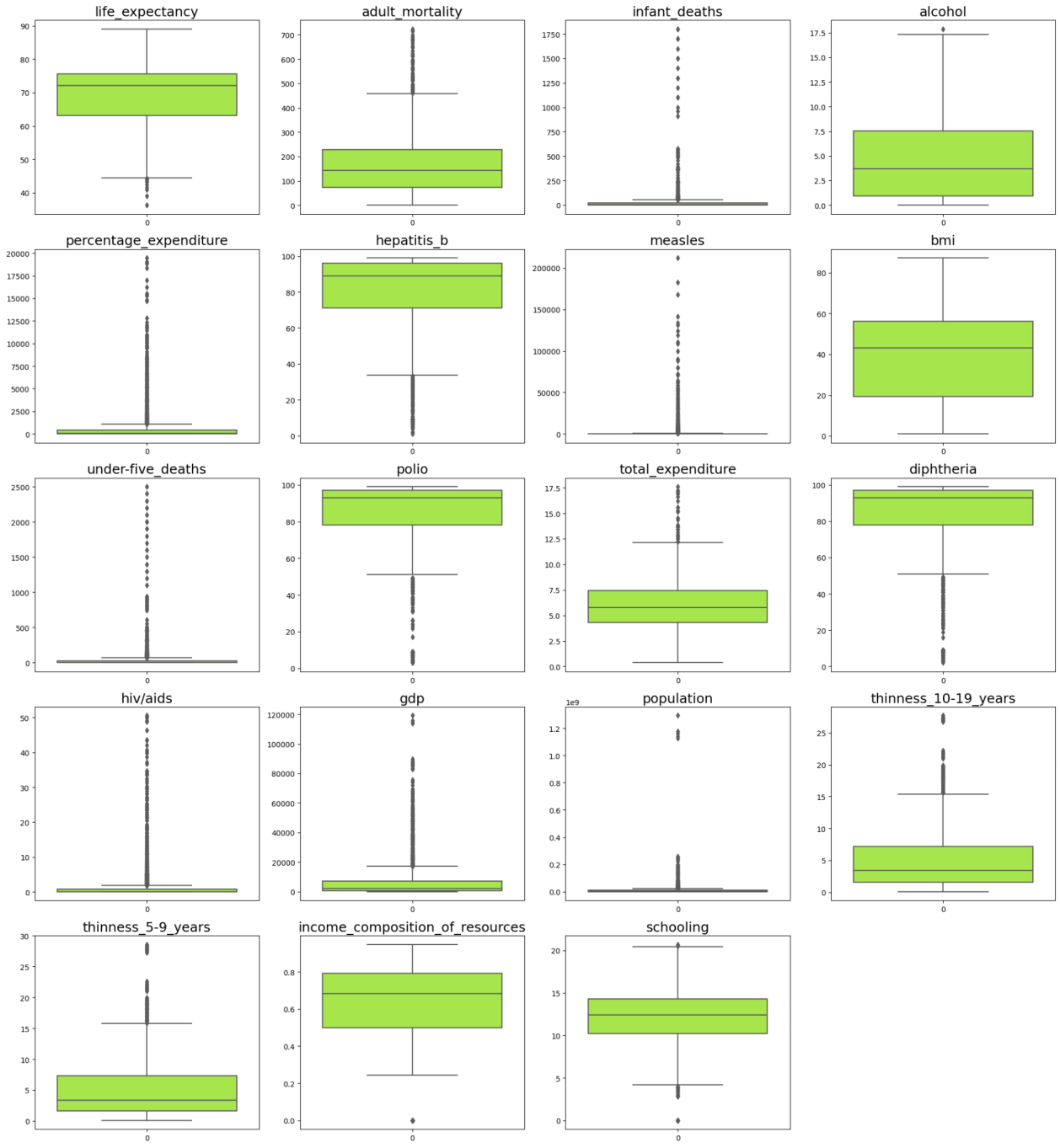

Mod of thinness_10-19_years
original
overall mean
mean by ctry
interpolate


Mod of thinness_5-9_years
original
overall mean
mean by ctry
interpolate


Mod of income_composition_of_resources
original
overall mean
mean by ctry
interpolate

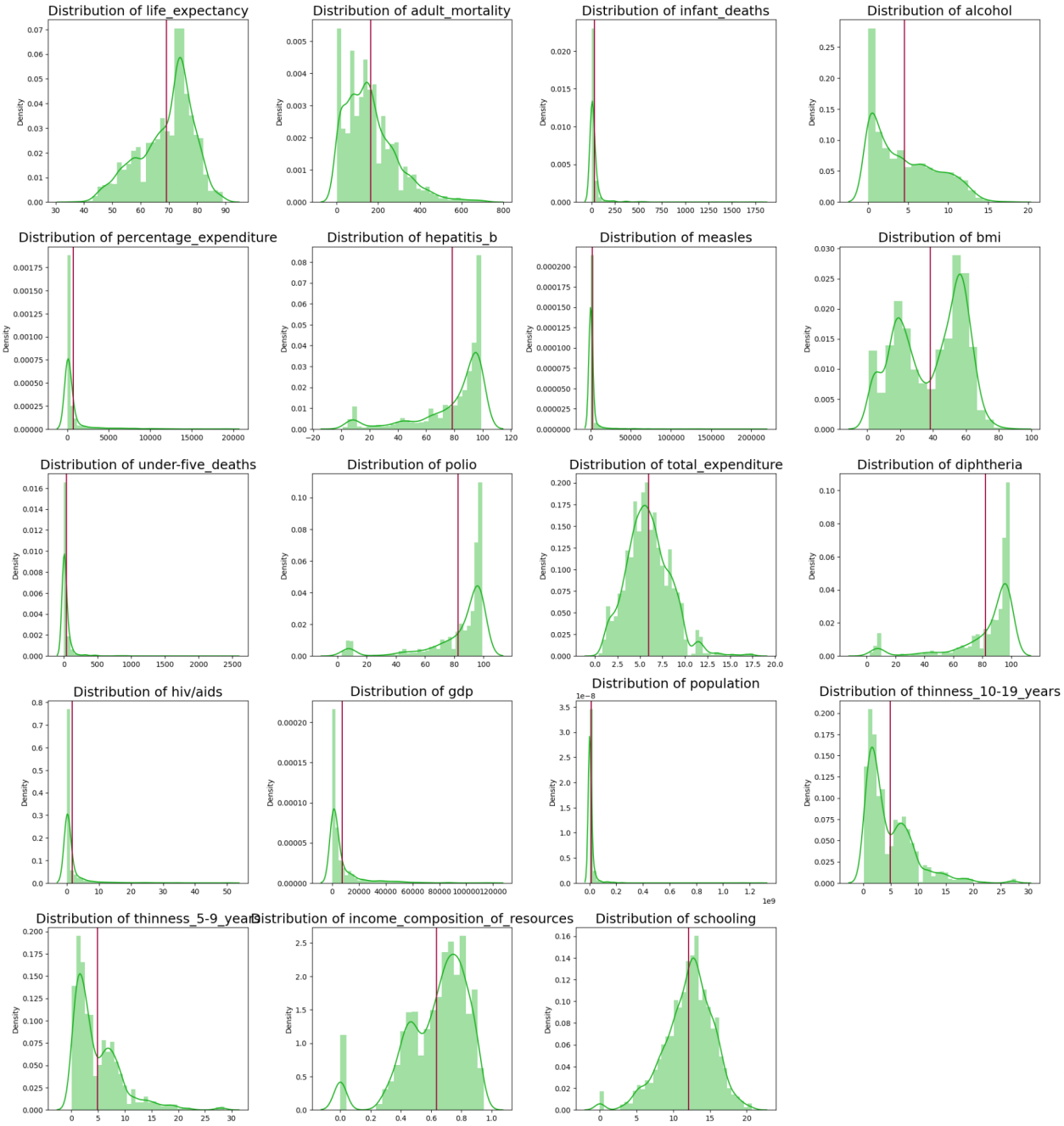

Mod of schooling
original
overall mean
mean by ctry
interpolate

```
#let's look at the distributions of our continuous variables
num_cols = ['life_expectancy', 'adult_mortality',
       'infant_deaths', 'alcohol', 'percentage_expenditure', 'hepatitis_b',
       'measles', 'bmi', 'under-five_deaths', 'polio', 'total_expenditure',
       'diphtheria', 'hiv/aids', 'gdp', 'population', 'thinness_10-19_years',
       'thinness_5-9_years', 'income_composition_of_resources', 'schooling']

# detecting outliers
plt.figure(figsize=(20,60))

for i, col in enumerate(num_cols):
    plt.subplot(len(na_cols), 4, i+1)
    sns.boxplot(le_df4[col], color=('xkcd:lime'))
    plt.title(f'{col}', fontsize=18)
    plt.xlabel('')

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




```
# detecting outliers and distribution via histograms
plt.figure(figsize=(20,60))

for i, col in enumerate(num_cols):
    plt.subplot(len(na_cols), 4, i+1)
    sns.distplot(le_df4[col], color=('xkcd:green'))
    plt.title(f'Distribution of {col}', fontsize=18)
```

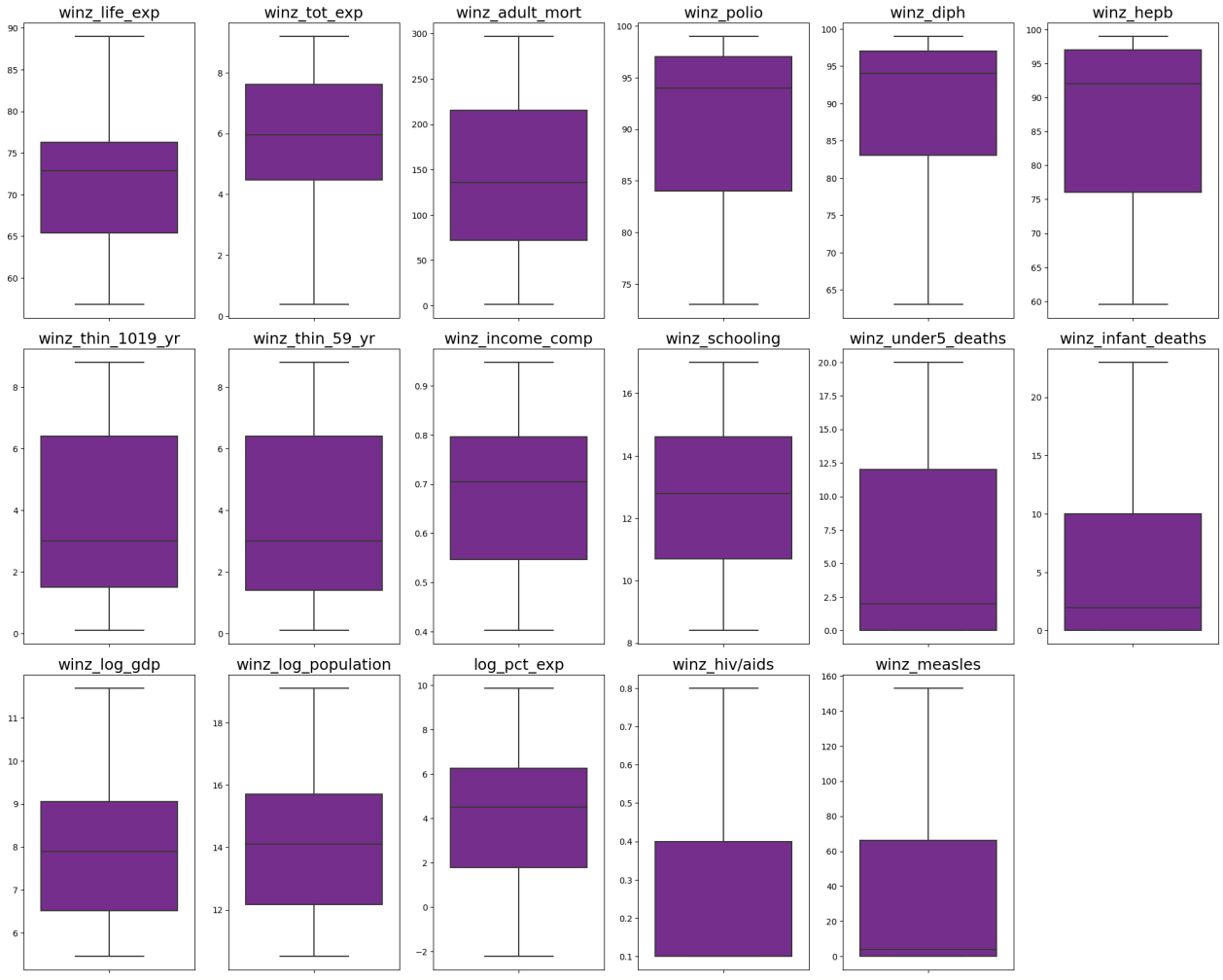
```
    plt.xlabel('')
    plt.axvline(le_df4.loc[:,col].mean(), color=('xkcd:cranberry')) #red line is the mean

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




```
# dropping mistakes in the data collection per the reasoning above
adj_le_df4 = le_df4[(le_df4.infant_deaths<1000) & (le_df4.measles<1000) & (le_df4['under-five_deaths']<1000)]

# winsorizations
```

```
adj_le_df4['winz_life_exp'] = winsorize(adj_le_df4['life_expectancy'], (0.10,0.0))
adj_le_df4['winz_tot_exp'] = winsorize(adj_le_df4['total_expenditure'], (0.0,0.10))
adj_le_df4['winz_adult_mort'] = winsorize(adj_le_df4['adult_mortality'], (0.0,0.10))
adj_le_df4['winz_polio'] = winsorize(adj_le_df4['polio'], (0.15,0.0))
adj_le_df4['winz_diph'] = winsorize(adj_le_df4['diphtheria'], (0.10,0.0))
adj_le_df4['winz_hepb'] = winsorize(adj_le_df4['hepatitis_b'], (0.15,0.0))
adj_le_df4['winz_thin_1019_yr'] = winsorize(adj_le_df4['thinness_10-19_years'], (0.0,0.10))
adj_le_df4['winz_thin_59_yr'] = winsorize(adj_le_df4['thinness_5-9_years'], (0.0,0.10))
adj_le_df4['winz_income_comp'] = winsorize(adj_le_df4['income_composition_of_resources'], (0.10,0.0))
adj_le_df4['winz_schooling'] = winsorize(adj_le_df4['schooling'], (0.10,0.05))
adj_le_df4['winz_under5_deaths'] = winsorize(adj_le_df4['under-five_deaths'], (0.0, 0.20))
adj_le_df4['winz_infant_deaths'] = winsorize(adj_le_df4['infant_deaths'], (0.0, 0.15))
adj_le_df4['winz_hiv/aids'] = winsorize(adj_le_df4['hiv/aids'], (0.0, 0.21))
adj_le_df4['winz_measles'] = winsorize(adj_le_df4['measles'], (0.0, 0.17))



# transformations
adj_le_df4['winz_log_gdp'] = winsorize(np.log(adj_le_df4['gdp']), (0.10, 0.0))
adj_le_df4['winz_log_population'] = winsorize(np.log(adj_le_df4['population']), (0.10, 0.0))
adj_le_df4['log_pct_exp'] = np.log(adj_le_df4['percentage_expenditure'])


# reinspecting to see how outliers were handled
adj_num_cols = [ 'winz_life_exp', 'winz_tot_exp',
      'winz_adult_mort', 'winz_polio', 'winz_diph', 'winz_hepb',
      'winz_thin_1019_yr', 'winz_thin_59_yr', 'winz_income_comp',
      'winz_schooling', 'winz_under5_deaths', 'winz_infant_deaths',
      'winz_log_gdp', 'winz_log_population', 'log_pct_exp', 'winz_hiv/aids',
      'winz_measles']


plt.figure(figsize=(20,90))
```

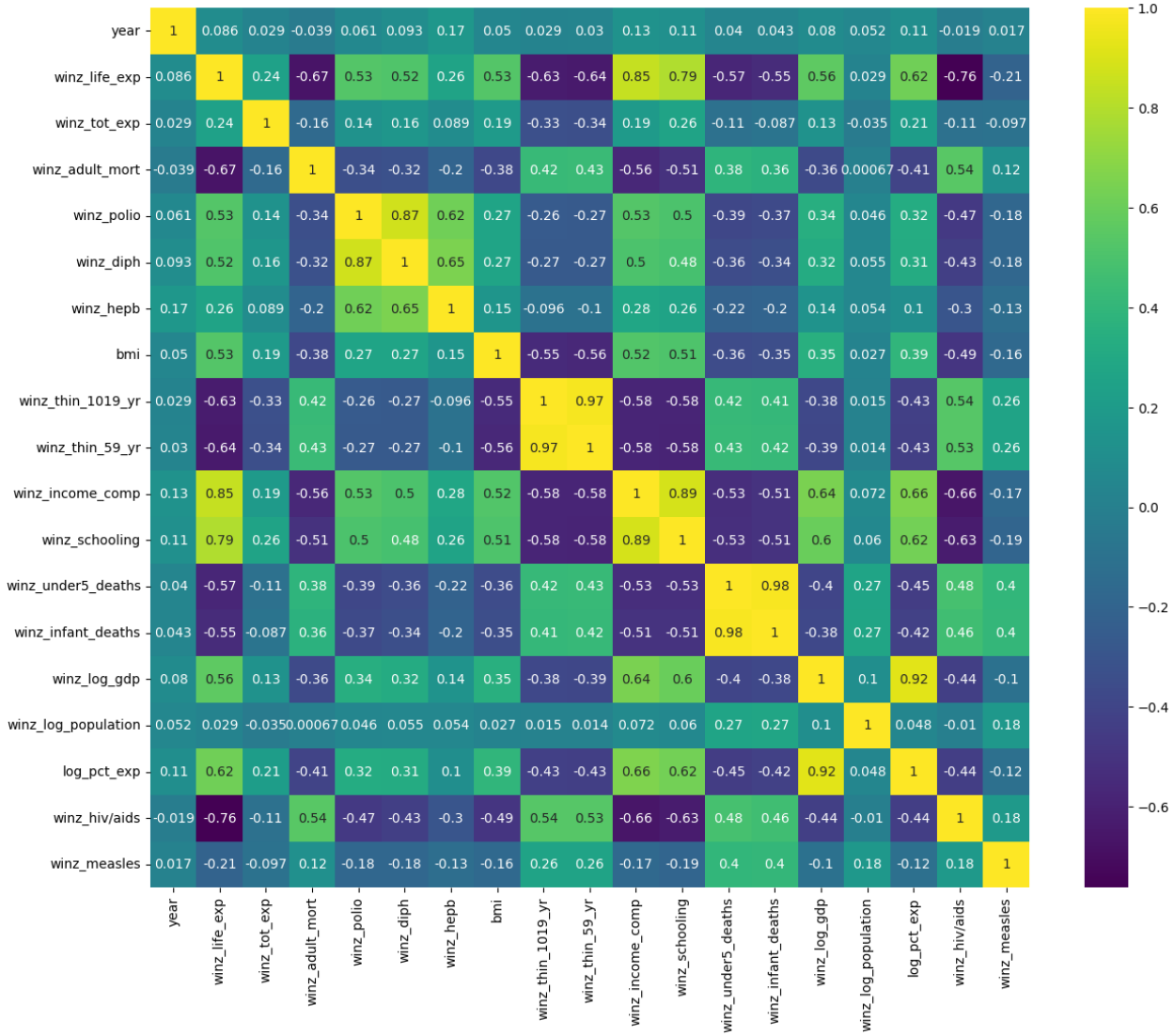
```
for i, col in enumerate(adj_num_cols):
    plt.subplot(len(adj_num_cols), 6, i+1)
    sns.boxplot(y=adj_le_df4[col], color=('xkcd:purple'))
    plt.title(f'{col}', fontsize=18)
    plt.ylabel('')

plt.tight_layout()
plt.show()
# all outliers have been dealt with
```




```
# correlation heat map
adj_corr = adj_le_df4[['year', 'status', 'country', 'winz_life_exp', 'winz_tot_exp',
       'winz_adult_mort', 'winz_polio', 'winz_diph', 'winz_hepb', 'bmi',
       'winz_thin_1019_yr', 'winz_thin_59_yr', 'winz_income_comp',
       'winz_schooling', 'winz_under5_deaths', 'winz_infant_deaths',
```

```
    'winz_log_gdp', 'winz_log_population', 'log_pct_exp', 'winz_hiv/aids',
    'winz_measles']].corr()
```

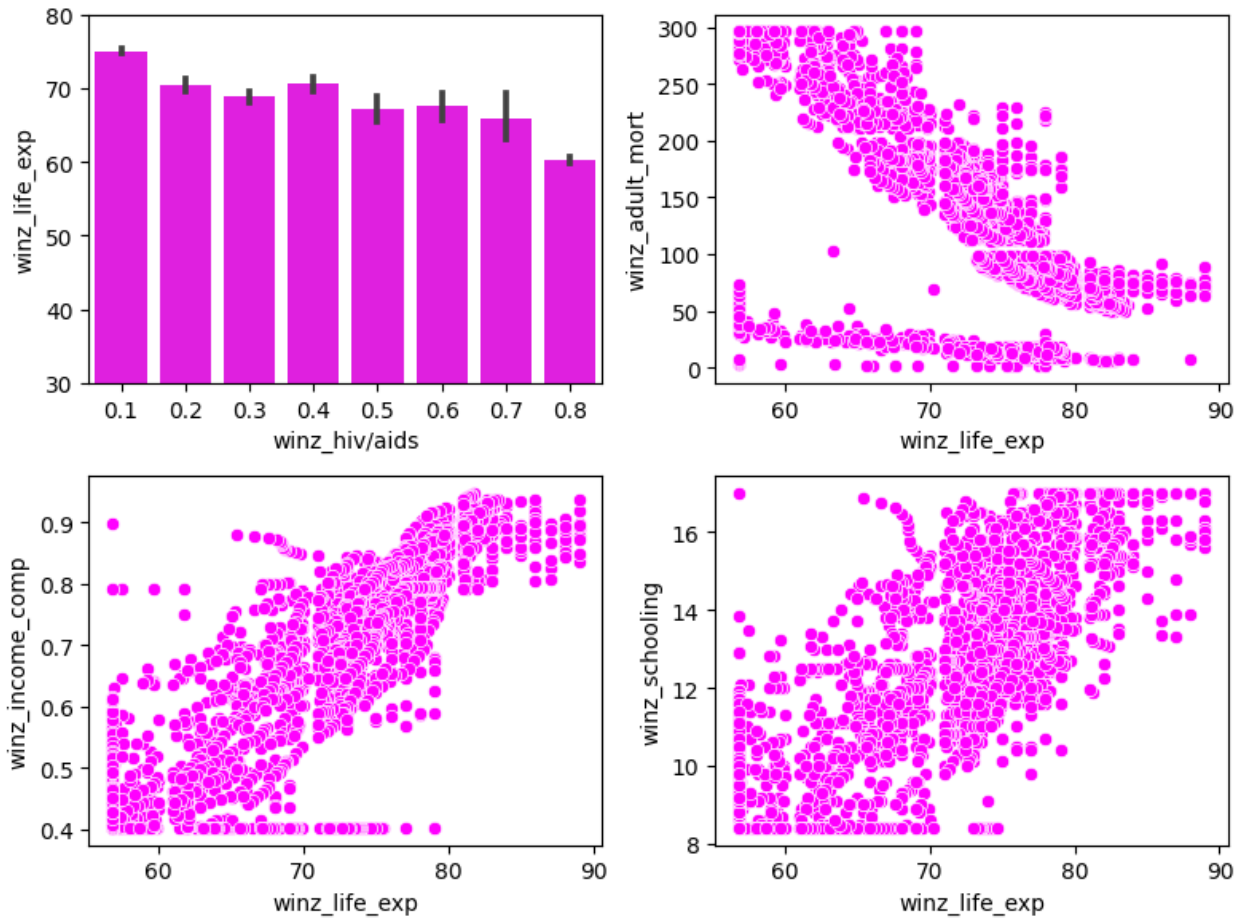
```
plt.figure(figsize=(15,12))
sns.heatmap(adj_corr, square=True, annot=True, cmap='viridis');
```




```
#Let's take a visual of these correlations
plt.figure(figsize=(8,6))
```

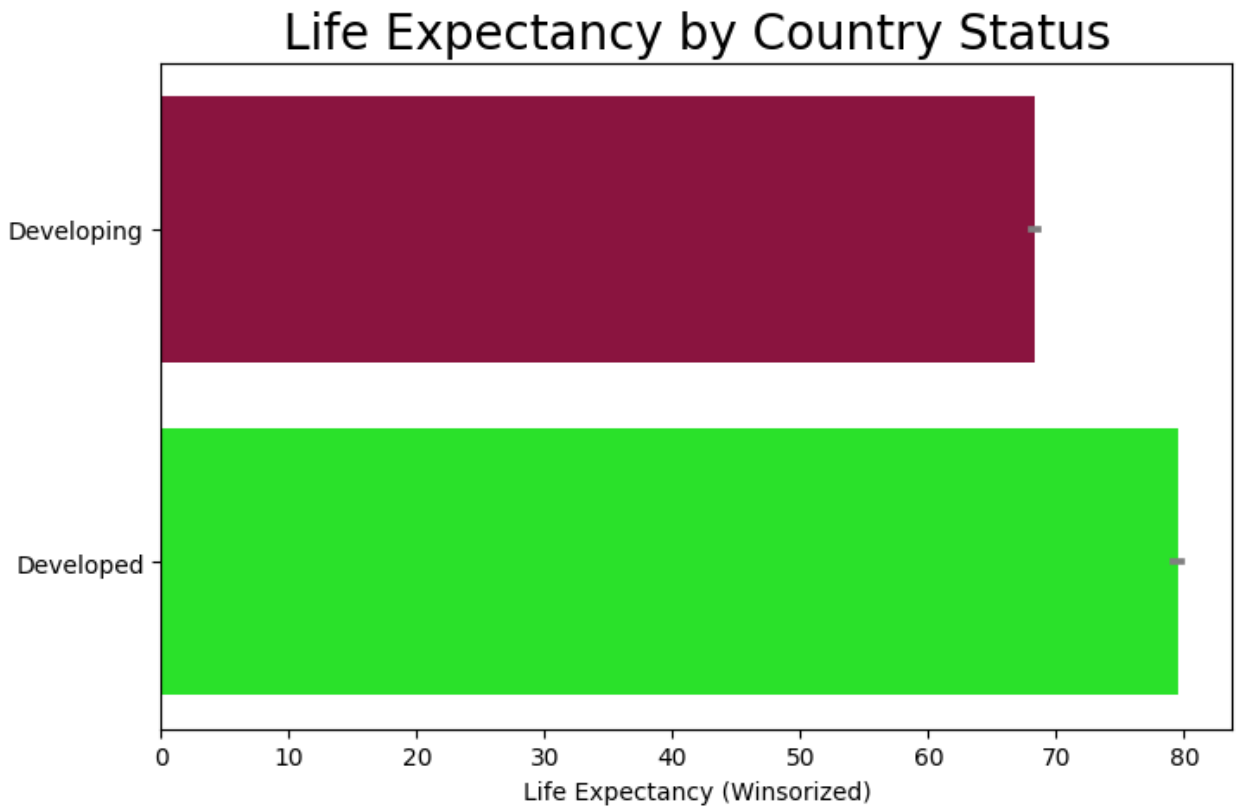
```
plt.subplot(2,2,1)
# Turns out winz_hiv/aids is actually a categorical variable, so I am using a bar plot
ax = sns.barplot(x='winz_hiv/aids', y='winz_life_exp', data=adj_le_df4, color='magenta')
ax.set_ylim(30,80)
```

```
plt.subplot(2,2,2)

ax1 = sns.scatterplot(x='winz_life_exp', y='winz_adult_mort', data=adj_le_df4, color='magenta')


plt.subplot(2,2,3)

ax2 = sns.scatterplot(x='winz_life_exp', y='winz_income_comp', data=adj_le_df4, color='magenta')


plt.subplot(2,2,4)

ax3 = sns.scatterplot(x='winz_life_exp', y='winz_schooling', data=adj_le_df4, color='magenta')


plt.tight_layout()

plt.show;
```



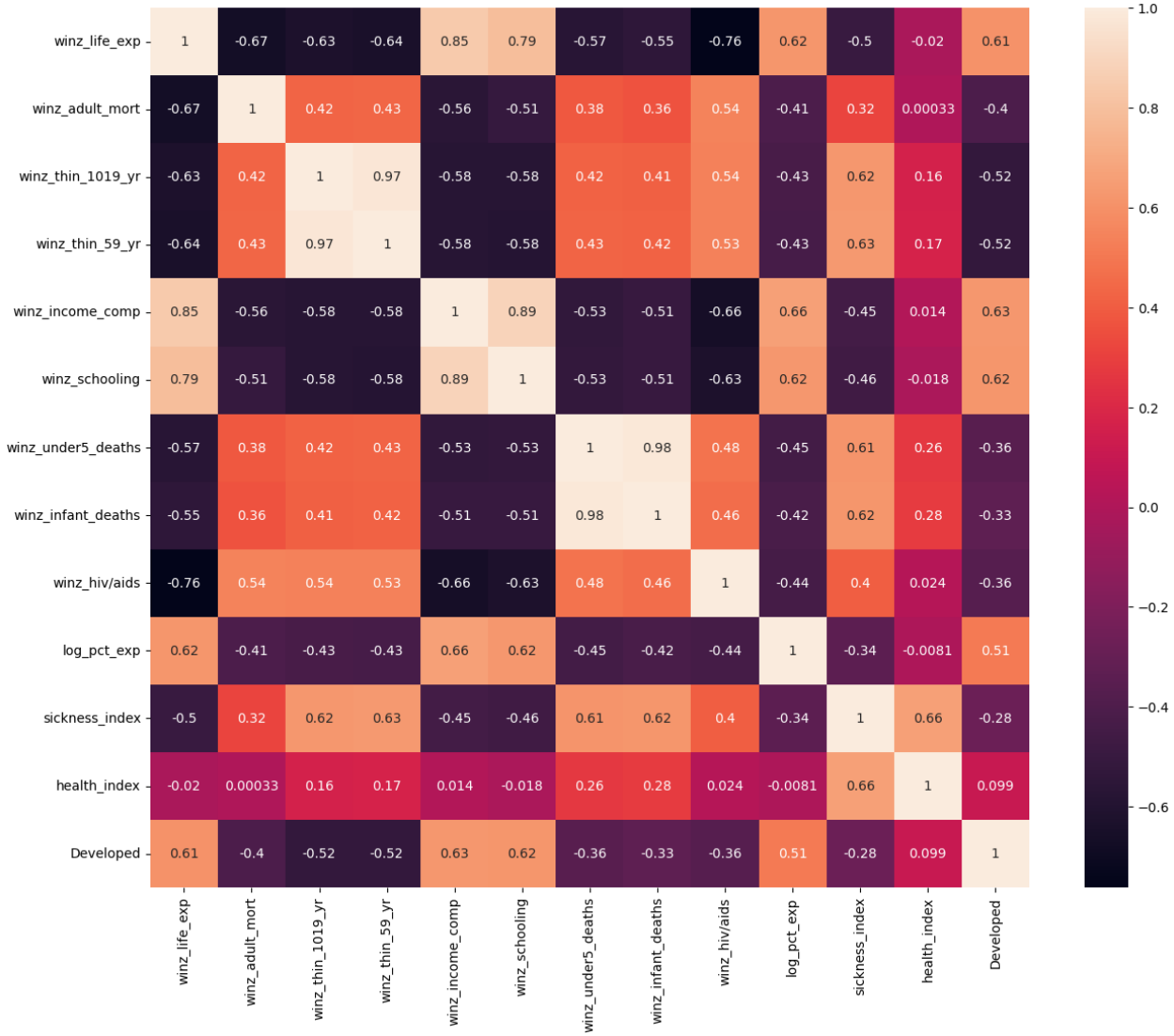

```
# Comparing Developed vs Developing

# In order to do this, we will need to correct the incorrectly labeled countries
```

```
# List of incorrectly classified countries
incorr_status = ['Canada', 'Chile', 'Greece', 'Finland', 'France', 'Israel', 'Republic of Korea']


# Loop through and change them to Developed
for country in incorr_status:
    adj_le_df4['status'].loc[adj_le_df4.country == country] = 'Developed'


# Verify
print(adj_le_df4[adj_le_df4['status']=='developed']['country'].unique())
# Verified;
# Plotting Developed v Developing
plt.figure(figsize=(8,5))

sns.barplot(y='status', x='winz_life_exp', data=adj_le_df4, orient='h',
            palette = ['xkcd:cranberry','xkcd:neon green'], errcolor='grey');
plt.title('Life Expectancy by Country Status', fontsize=20)
plt.xlabel('Life Expectancy (Winsorized)')
plt.ylabel('');
```




```
# Differences in life expectancy seem drastic

# Running an independent T Test to see if results are significant

stats.ttest_ind(adj_le_df4[adj_le_df4['status']=='Developed']['winz_life_exp'],
                adj_le_df4[adj_le_df4['status']=='Developing']['winz_life_exp'])

# trying my hand at a sort of sickness ratio and health ratio

adj_le_df4['sickness_index'] = ((adj_le_df4['winz_infant_deaths']+adj_le_df4['winz_measles']+
                                 adj_le_df4['winz_under5_deaths']+
                                 adj_le_df4['winz_hiv/aids'])/4)*((adj_le_df4['winz_thin_59_yr']+
                                                                   adj_le_df4['winz_thin_1019_yr'])/2)


adj_le_df4['health_index'] = (adj_le_df4['winz_measles']+adj_le_df4['winz_polio']+
                              adj_le_df4['winz_diph'])/3


# Creating binary variable for Developed and Developing replacing Status

adj_le_df4 = pd.concat([adj_le_df4, pd.get_dummies(adj_le_df4.status)], axis=1)

# dropping non-transformed columns and low correlation items

adj_le_df4.drop(['status', 'life_expectancy', 'adult_mortality',
```

```
        'infant_deaths', 'percentage_expenditure', 'hepatitis_b',
        'measles', 'under-five_deaths', 'polio', 'total_expenditure',
        'diphtheria', 'hiv/aids', 'gdp', 'population', 'thinness_10-19_years',
        'thinness_5-9_years', 'income_composition_of_resources', 'schooling', 'winz_tot_exp',
         'winz_hepb', 'Developing', 'winz_log_population', 'winz_measles',
         'winz_log_gdp', 'year', 'alcohol', 'bmi', 'winz_polio', 'winz_diph'],
         axis=1, inplace=True)
plt.figure(figsize=(15,12))
suite_corr = adj_le_df4.corr()

sns.heatmap(suite_corr, square=True, annot=True);
```



```
PCA_df = adj_le_df4[['winz_life_exp', 'winz_adult_mort', 'winz_thin_1019_yr',
       'winz_thin_59_yr', 'winz_income_comp', 'winz_schooling',
       'winz_under5_deaths', 'winz_infant_deaths', 'winz_hiv/aids']]


stndzd_PCA_df = StandardScaler().fit_transform(PCA_df)


sklearn_PCA = PCA(n_components=4)


PCs = sklearn_PCA.fit_transform(stndzd_PCA_df)


print(
    'The percentage of total variance in the dataset explained by each',
    'component from Sklearn PCA: \n',
    #sklearn_PCA.components_,
    sklearn_PCA.explained_variance_ratio_,
    '\n Eigenvalues of each component: \n',
    sklearn_PCA.explained_variance_

)
```

The percentage of total variance in the dataset explained by each component from Sklearn PCA:

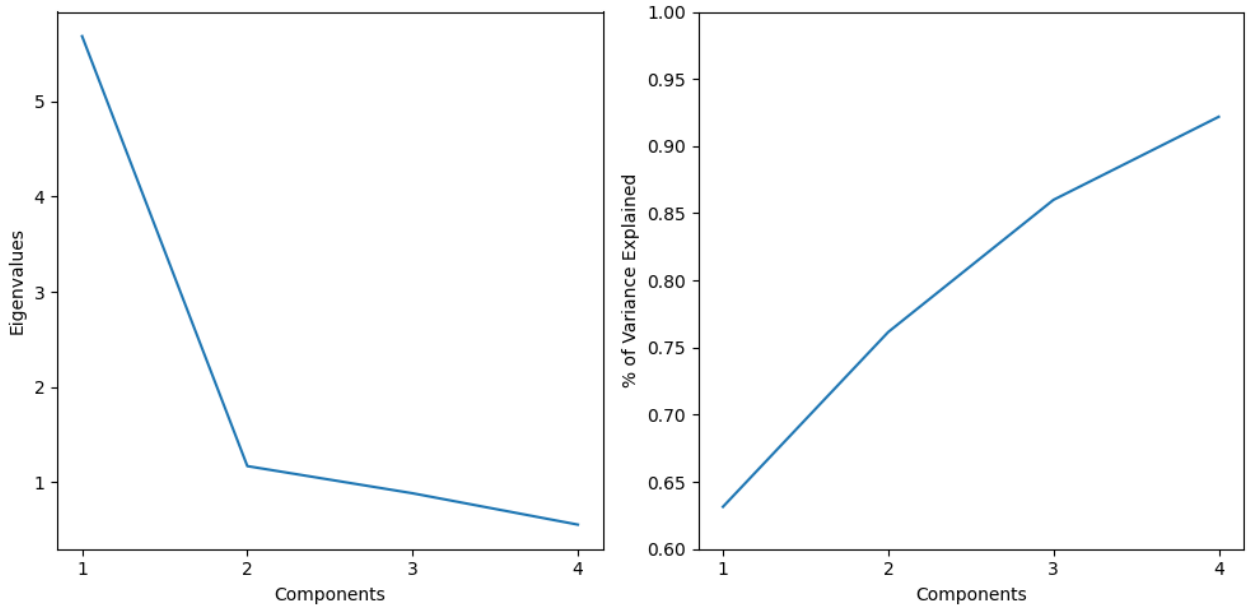
[0.63146182 0.13009528 0.09841427 0.06185948]

Eigenvalues of each component:

[5.68551258 1.17134294 0.88609561 0.55696614]

```
# Let's visualize the above values
fig, ax = plt.subplots(figsize=(10,5))


ax1 = plt.subplot(121)
plt.plot(sklearn_PCA.explained_variance_)
```

```
ax1.set_xticks([0,1,2,3])
ax1.set_xticklabels([1,2,3,4])
ax1.set_xlabel('Components')
ax1.set_ylabel('Eigenvalues')


ax2 = fig.add_subplot(122)
plt.plot(np.cumsum(sklearn_PCA.explained_variance_ratio_))
ax2.set_ylabel('% of Variance Explained')
ax2.set_xlabel('Components')
ax2.set_xticks([0,1,2,3])
ax2.set_xticklabels([1,2,3,4])
ax2.set_ylim(.6,1);

plt.tight_layout()
```




```
# Bear with me as I create a dictionary of ISO country codes for plotly
ISOs_Country = {
'ABW':'Aruba',
```

```
'AFG':'Afghanistan','AGO':'Angola','AIA':'Anguilla','ALA':'Åland
Islands','ALB':'Albania',

'AND':'Andorra','ARE':'United Arab
Emirates','ARG':'Argentina','ARM':'Armenia','ASM':'American Samoa',

'ATA':'Antarctica','ATF':'French Southern Territories','ATG':'Antigua and
Barbuda','AUS':'Australia',

'AUT':'Austria','AZE':'Azerbaijan','BDI':'Burundi','BEL':'Belgium','BEN':'Ben
in',

'BES':'Bonaire, Sint Eustatius and Saba','BFA':'Burkina
Faso','BGD':'Bangladesh','BGR':'Bulgaria',

'BHR':'Bahrain','BHS':'Bahamas','BIH':'Bosnia and Herzegovina','BLM':'Saint
Barthélemy','BLR':'Belarus',

'BLZ':'Belize','BMU':'Bermuda','BOL':'Bolivia (Plurinational State
of)','BRA':'Brazil','BRB':'Barbados',

'BRN':'Brunei Darussalam','BTN':'Bhutan','BVT':'Bouvet
Island','BWA':'Botswana','CAF':'Central African Republic',

'CAN':'Canada','CCK':'Cocos (Keeling)
Islands','CHE':'Switzerland','CHL':'Chile','CHN':'China',

'CIV':'Côte d\'Ivoire','CMR':'Cameroon','COD':'Democratic Republic of the
Congo','COG':'Congo','COK':'Cook
Islands','COL':'Colombia','COM':'Comoros','CPV':'Cabo Verde','CRI':'Costa
Rica','CUB':'Cuba',

'CUW':'Curaçao','CXR':'Christmas Island','CYM':'Cayman
Islands','CYP':'Cyprus','CZE':'Czechia',

'DEU':'Germany','DJI':'Djibouti','DMA':'Dominica','DNK':'Denmark','DOM':'Domi
nican Republic','DZA':'Algeria',

'ECU':'Ecuador','EGY':'Egypt','ERI':'Eritrea','ESH':'Western
Sahara','ESP':'Spain','EST':'Estonia',

'ETH':'Ethiopia','FIN':'Finland','FJI':'Fiji','FLK':'Falkland Islands
(Malvinas)','FRA':'France',

'FRO':'Faroe Islands','FSM':'Micronesia (Federated States of)','GAB':'Gabon',

'GBR':'United Kingdom of Great Britain and Northern
Ireland','GEO':'Georgia','GGY':'Guernsey','GHA':'Ghana',

'GIB':'Gibraltar','GIN':'Guinea','GLP':'Guadeloupe','GMB':'Gambia','GNB':'Gui
nea-Bissau',

'GNQ':'Equatorial
Guinea','GRC':'Greece','GRD':'Grenada','GRL':'Greenland','GTM':'Guatemala',
```

'GUF':'French Guiana','GUM':'Guam','GUY':'Guyana','HKG':'Hong Kong','HMD':'Heard Island and McDonald Islands',

'HND':'Honduras','HRV':'Croatia','HTI':'Haiti','HUN':'Hungary','IDN':'Indonesia','IMN':'Isle of Man',

'IND':'India','IOT':'British Indian Ocean Territory','IRL':'Ireland','IRN':'Iran (Islamic Republic of)',

'IRQ':'Iraq','ISL':'Iceland','ISR':'Israel','ITA':'Italy','JAM':'Jamaica','JEY':'Jersey','JOR':'Jordan',

'JPN':'Japan','KAZ':'Kazakhstan','KEN':'Kenya','KGZ':'Kyrgyzstan','KHM':'Cambodia','KIR':'Kiribati',

'KNA':'Saint Kitts and Nevis','KOR':'Republic of Korea','KWT':'Kuwait','LAO':'Lao People\'s Democratic Republic',

'LBN':'Lebanon','LBR':'Liberia','LBY':'Libya','LCA':'Saint Lucia', 'LIE':'Liechtenstein','LKA':'Sri Lanka',

'LSO':'Lesotho','LTU':'Lithuania','LUX':'Luxembourg','LVA':'Latvia','MAC':'Macao',

'MAF':'Saint Martin (French part)','MAR':'Morocco','MCO':'Monaco','MDA':'Republic of Moldova',

'MDG':'Madagascar','MDV':'Maldives','MEX':'Mexico','MHL':'Marshall Islands','MKD':'North Macedonia',

'MLI':'Mali','MLT':'Malta','MMR':'Myanmar','MNE':'Montenegro','MNG':'Mongolia','MNP':'Northern Mariana Islands',

'MOZ':'Mozambique','MRT':'Mauritania','MSR':'Montserrat','MTQ':'Martinique','MUS':'Mauritius','MWI':'Malawi',

'MYS':'Malaysia','MYT':'Mayotte','NAM':'Namibia','NCL':'New Caledonia','NER':'Niger','NFK':'Norfolk Island',

'NGA':'Nigeria','NIC':'Nicaragua','NIU':'Niue','NLD':'Netherlands','NOR':'Norway','NPL':'Nepal','NRU':'Nauru',

'NZL':'New Zealand','OMN':'Oman','PAK':'Pakistan','PAN':'Panama','PCN':'Pitcairn','PER':'Peru','PHL':'Philippines',

'PLW':'Palau','PNG':'Papua New Guinea','POL':'Poland','PRI':'Puerto Rico','PRK':'Democratic People\'s Republic of Korea',

'PRT':'Portugal','PRY':'Paraguay','PSE':'Palestine, State of','PYF':'French Polynesia','QAT':'Qatar','REU':'Réunion',

'ROU':'Romania','RUS':'Russian Federation','RWA':'Rwanda','SAU':'Saudi Arabia','SDN':'Sudan','SEN':'Senegal',

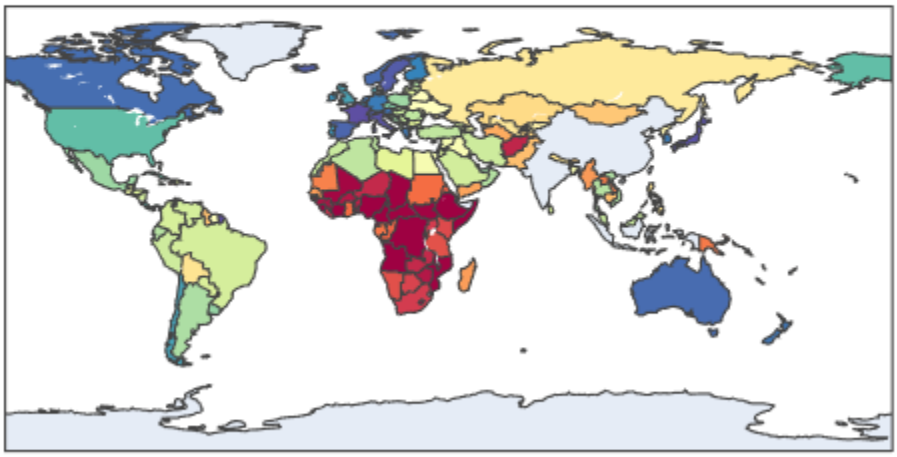
```
'SGP':'Singapore','SGS':'South Georgia and the South Sandwich
Islands','SHN':'Saint Helena, Ascension and Tristan da Cunha',

'SJM':'Svalbard and Jan Mayen','SLB':'Solomon Islands','SLE':'Sierra
Leone','SLV':'El Salvador','SMR':'San Marino',

'SOM':'Somalia','SPM':'Saint Pierre and Miquelon','SRB':'Serbia','SSD':'South
Sudan','STP':'Sao Tome and Principe',

'SUR':'Suriname','SVK':'Slovakia','SVN':'Slovenia','SWE':'Sweden','SWZ':'Eswa
tini','SXM':'Sint Maarten (Dutch part)',

'SYC':'Seychelles','SYR':'Syrian Arab Republic','TCA':'Turks and Caicos
Islands','TCD':'Chad','TGO':'Togo',

'THA':'Thailand','TJK':'Tajikistan','TKL':'Tokelau','TKM':'Turkmenistan','TLS
':'Timor-Leste','TON':'Tonga','TTO':'Trinidad and
Tobago','TUN':'Tunisia','TUR':'Turkey','TUV':'Tuvalu','TWN':'Taiwan, Province
of China',

'TZA':'United Republic of
Tanzania','UGA':'Uganda','UKR':'Ukraine','UMI':'United States Minor Outlying
Islands',

'URY':'Uruguay','USA':'United States of
America','UZB':'Uzbekistan','VAT':'Holy See','VCT':'Saint Vincent and the
Grenadines',

'VEN':'Venezuela (Bolivarian Republic of)','VGB':'Virgin Islands
(British)','VIR':'Virgin Islands (U.S.)','VNM':'Viet Nam',

'VUT':'Vanuatu','WLF':'Wallis and
Futuna','WSM':'Samoa','YEM':'Yemen','ZAF':'South Africa','ZMB':'Zambia',

'ZWE':'Zimbabwe'}
```

```
# Turning above dict into df

ISO_df = pd.DataFrame.from_dict(ISOs_Country, orient='index', columns=['country'])

# Grouping WHO df by Country and aggregating the continuous variables we want to plot

reduced_adj_df = adj_le_df4.groupby('country')['winz_life_exp', 'Developed',
                                               'winz_income_comp', 'winz_schooling'].agg(
    {'winz_life_exp':'mean', 'Developed':'min', 'winz_income_comp':'mean', 'winz_schooling':'mean'})

# Merging ISO and WHO dfs together

merged = pd.merge(reduced_adj_df, ISO_df, left_index=True, right_on='country')

# Plotting via the Plotly library
```

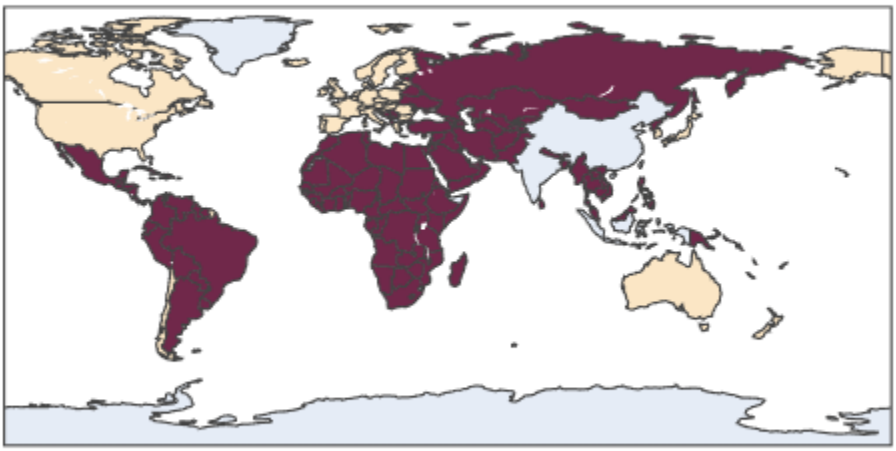
```
import plotly.graph_objects as go
from plotly.subplots import make_subplots

fig = make_subplots(
    rows=4, cols=1,
    row_heights=[0.25, 0.25, 0.25, 0.25],
    vertical_spacing=0.025,
    subplot_titles=("World Life Expectancy", "Status of Countries",
                    "Income Composition of Resources", "Highest Average Age of Schooling"),
    specs=[[{"type": "Choropleth", "rowspan": 1}],
           [{"type": "Choropleth", "rowspan": 1}],
          [{"type": "Choropleth", "rowspan": 1}],
           [{"type": "Choropleth", "rowspan": 1}]])

fig.add_trace( # Life Expectancy
    go.Choropleth(locations = merged.index,
                  z= merged['winz_life_exp'],
                  text=merged['country'],
                  name='Life Expectancy',
                  colorbar={'title':'Life<br>Expectancy', 'len':.25, 'x':.99,'y':.896},
                  colorscale='spectral',),
    row=1,col=1
)

fig.add_trace( #Developed v Developing
    go.Choropleth(locations = merged.index,
                  z= merged['Developed'],
                  text=merged['country'],
                  name='Status of Countries',
                  colorbar={'len':.227, 'x':.99,'y':.629, 'tickmode':'array','nticks':2,
                        'tickvals':[0,1], 'ticktext':('Developing', 'Developed')},
```

```
            colorscale='burgyl_r'),
    row=2,col=1
)

fig.add_trace( # Income Comp
    go.Choropleth(locations = merged.index,
            z= merged['winz_income_comp'],
            text=merged['country'],
            name='Income Composition of Resources',
            colorbar={'title':'Index', 'len':.24, 'x':.99,'y':.378},
            colorscale='bluered',),row=3,col=1
)

fig.add_trace( #Schooling
    go.Choropleth(locations = merged.index,
            z= merged['winz_schooling'],
            text=merged['country'],
            name='Highest Average Age of Schooling',
            colorbar={'len':.248, 'x':.99,'y':.1275, 'title':'Schooling<br>Age'},
            colorscale='burgyl_r'),
    row=4,col=1
)

fig.update_layout(
    margin=dict(r=1, t=30, b=10, l=30),
    width=700,
    height=1400,
)

fig.show()
```

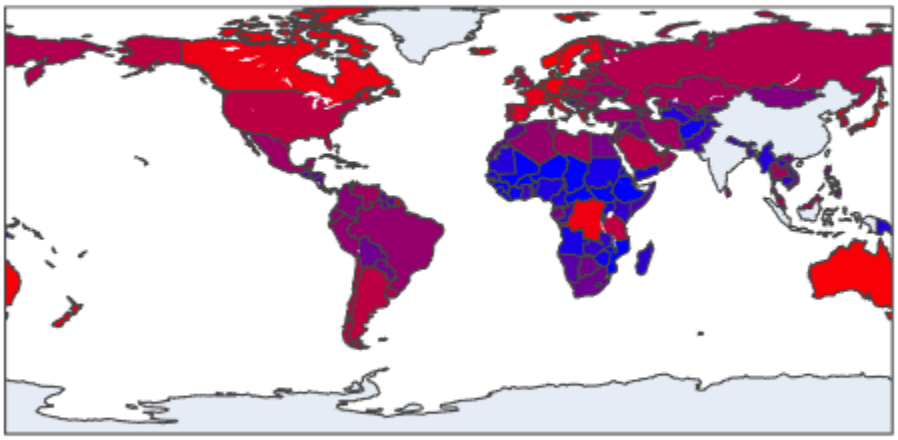
World Life Expectancy



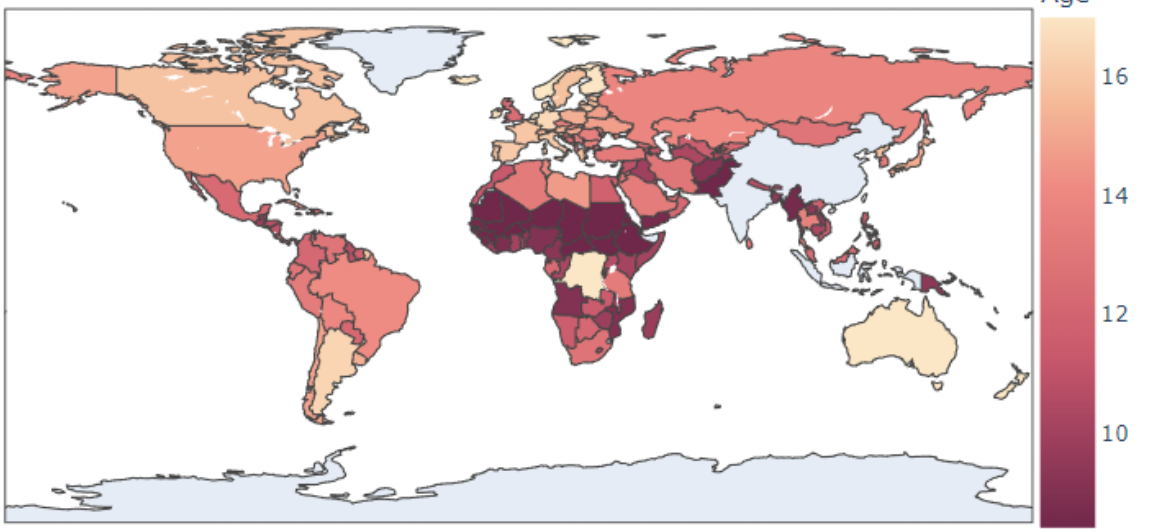
Status of Countries



Income Composition of Resources



Highest Average Age of Schooling



6.PERFORMANCE ANALYSIS

6.1COMPARISON ANALYSIS OF MACHINE LEARNING ALGORITHM

Machine learning is an application of artificial intelligence (AI) that provides systems the ability to automatically learn and improve from experience without being explicitly programmed.Machine learning algorithms are often categorized as supervised or unsupervised.In Supervised learning, you train the machine using data which is well "labeled." It means some data is already tagged with the correct answer. It can be compared to learning which takes place in the presence of a supervisor or a teacher.A supervised learning algorithm learns from labeled training data, helps you to predict outcomes for unforeseen data.In supervised learning, an algorithm is employed to learn the mapping function from the input variable (x) to the output variable (y); that is y = f(X).
The objective of such a problem is to approximate the mapping function (f) as accurately as possible such that whenever there is a new input data (x), the output variable (y) for the dataset can be predicted.Supervised Learning can further be divided into 2 categories: Regression and Classification

Regression
In machine learning, regression algorithms attempt to estimate the mapping function (f) from the input variables (x) to numerical or continuous output variables (y).

For example, when provided with a dataset about houses, and you are asked to predict their prices, that is a regression task because price will be a continuous output.

Classification
On the other hand, classification algorithms attempt to estimate the mapping function (f) from the input variables (x) to discrete or categorical output variables (y).

In case of housing prices dataset, the houses will be classified whether their prices fall into two discrete categories: above or below the said price.

Regression vs. Classification
The main difference between them is that the output variable in regression is numerical (or continuous) while that for classification is categorical (or discrete).

What is Unsupervised Learning?
Unsupervised learning is a machine learning technique, where you do not need to supervise the model. Instead, you need to allow the model to work on its own to discover information. It mainly deals with the unlabelled data.

Unsupervised learning algorithms allows you to perform more complex processing tasks compared to supervised learning.

A tool which is used for both Data mining and Machine Learning is WEKA. It was first implemented by The University of Waikato, New Zealand, in 1997 [4]. It is a collection of an enormous number of

Machine Learning and Data Mining algorithms. One drawback of this software is that it supports data files only written in ARFF (attribute relation file format) and CSV (comma separated values) format. Initially, it was written in C but later on it was rewritten in JAVA language. It comprises of a GUI interface for interaction with the data files. It possesses 49 data pre-processing tools, 15 attribute evaluators, 76 classification algorithms and 10 search algorithms for the purpose of feature selection. It comprises of three different types of graphical user interfaces (GUI'‘s):- “The Explorer”, “The Experimenter”, and “The Knowledge Flow”. WEKA provides the opportunity for the development of any new Machine Learning algorithm. It contains visualization tools and a set of panels to execute the desired tasks.

Classification algorithms or classifiers are used to basically sort out the network traffic into normal and anomaly categories. The objective behind classification techniques is to achieve high accuracy and precision and to classify the objects. Classifiers can be broadly classified into eight types in WEKA, where various machine learning algorithms reside in each category. The classifiers have been described in brief below.

Bayes Classifier-It originates from previous works in pattern recognition and is linked to the family of probabilistic Graphical Models. For each class, a probabilistic summary is stored. The conditional probability of each attribute and the probability of the class are stored in this summary. The graphical models are used to display knowledge about domains which are uncertain in nature. In the graphs [5], nodes depict random variables and the edges which connect corresponding random variable nodes are assigned weights which represent probabilistic dependencies. On encountering a new instance, the algorithm just creates an update of the probabilities stored along with the specific class [6]. The sequence of training instances and the existence of classification errors do not have any role in this process. Thus basically it has to predict the class depending on the value of the members of the class. This category consists of 13 classifiers, but only 3 of those are compatible with our chosen dataset.

Function classifier- It deploys the concept of regression and neural network. Input data is mapped to the output. It employs the iterative parameter estimation scheme. Overall there are 18 classifiers under this category, out of which only 2 are compatible with our dataset.

Lazy classifier- This requires the storage of the entire training instances and supports inclusion of new data only after classification time. The prime advantage of this classification scheme is the local approximation of the target function [5]. For each query to the system, the objective function is approximated locally thereby enabling lazy learning systems to solve multiple problems concurrently. But, the disadvantage is that it consumes a huge amount of storage space to store the entire training instance at once. It is time consuming also. Five classifiers are available under this category, but only two of those are compatible with our data set.

Meta Classifier- These sets of classifiers are essential to find the optimal set of attributes which can be used for training the base classifier [7]. New adaptive machine learning algorithms can be constructed using these classifiers and those new models can be further used for making predictions. 26 classifiers reside in this category, out of which 21 of them are compatible with our dataset.

Mi Classifier- Mi represents Multi-Instance Classifiers [8]. It consists of multiple instances in an example, but observation of one class is possible only for all the instances. Thus, it is an improvised learning technique. There are 12 classifiers under this category but all of them are incompatible with our dataset.

Misc Classifier- This category consists of different types of classifiers. Only two of them, out of three are compatible with our dataset.

Rules Classifier-Some kind of association rule is used for correct prediction of class among all the attributes. The amount of correct prediction is defined by the term coverage and is expressed in percentage or accuracy form. The association rules are mutually exclusive. More than one conclusion can be predicted. A total of 11 classifiers are available under this category, but 8 are compatible with our dataset.

Trees-It is a technique in which a flow-like tree diagram is generated where each node depicts a test on the attribute value and the outcome of each test is represented by each branch. The model generated is both predictive and descriptive in nature. The predicted classes are being signified by the tree leaves. There are 16 classifiers available out of which 10 are deemed to be compatible with this dataset.

The classification algorithms which we have executed in this paper are discussed in detail below.

BayesNet- It is a widely used technique which works on the basic Bayes theorem and forms a Bayesian network [9] after calculating conditional probability on each node. It is a graphical model which is probabilistic in nature and portrays a group of arbitrary variables along with their conditional dependencies through a directed acyclic graph.

Logistic- This technique employs regression to predict the probability of an outcome which can have only two values. One or several predictors are used to make the prediction. Logistic regression produces a logistic curve that is confined to values between 0 and 1. The curve is constructed using the natural logarithm of the odds of the target variable and not the probability.

IBK- It stands for instance based knowledge representation of the training instances [10] and does not conclude or predict a rule set or a decision tree. After a set of training instances has been stored, the memory is searched for the new training instance. So it is time consuming and requires space also.

JRip-This technique executes a proposed rule learner and cumulative error pruning method to reduce error. It is based on association rules with reduced error pruning techniques, thus making it an effective technique.

PART-It uses a divide and conquer approach to construct a C4.5 decision tree partially for each iteration specifying the optimal rule association. Using an entropic distance measure technique, it performs instance based learning.

J48- It is an enhanced version of C 4.5 which revolves on the ID3 algorithm with some extra functionalities to resolve issues that ID3 was incompetent in [11]. However, this technique is time and space consuming. Initially, it builds a tree using the divide and conquer algorithm and then applies heuristic criteria. The rules according to which the tree is generated are precise and intuitive.

Random Forest- This classification algorithm uses ensemble methods to obtain better predictive performance. It produces output in the form of individual trees and is based upon the decision tree algorithm. It is considered to be a highly accurate classifier and can handle multiple variables.

Random Tree-It generates a tree by randomly selecting branches from a possible set of trees. The trees are distributed in a uniform way so chances of getting sampled are equiprobable.

REPTree- It is a rapid decision tree learner. A decision tree is constructed with the help of information obtained on gain/variance [12] and uses reduced error pruning techniques to reduce the error. The sorting of the values for numeric attributes is done exactly once by the algorithm and then it deals with the missing ones by splitting the subsequent instances into pieces.

In this paper, we have used certain machine learning algorithms. They construct a model from example inputs and use it for decision and prediction making. The algorithms which we have used are AdaBoost, Stacking and Bagging. They have been discussed in details below.

AdaBoost-It actually stands for adaptive boosting algorithm [13]. It is an ensemble based method initiating with a base classifier which is built on the training data. Then a second classifier is established behind it to concentrate on the instances in the training data which were obtained wrongly from the base classifier. Addition of further classifiers continues till a specified limit is reached in number of models or accuracy. Boosting uses the J48 algorithm for the base classifier. Boosting helps in enhancement of accuracy of any machine learning algorithm.

Bagging-Bagging or bootstrapping aggregating [14] is an ensemble method which creates different samples of the training dataset and for each sample a classifier is created. Finally, the results of these various classifiers are combined using average or majority voting. Since each sample of the training set is different from the other, so each trained classifier is given a different focus and outlook to the problem. It also uses the J48 as the base classifier. Bagging reduces variance and helps in avoidance of over fitting. It improves the accuracy and stability of machine learning algorithms.

Stacking-Stacking or blending is another ensemble method where preparation of different multiple algorithms takes place on the training data. A Meta classifier is prepared which learns to take predictions of each classifier and make precise predictions on data which cannot be seen. J48 and IBk are the two classifiers which are used and the Meta classifier used is Logistic Regression. Blending is basically the combination of different types of algorithms. So we are using J48, under tree section, and IBk (k-nearest neighbor), under lazy section, which are entirely different sets of algorithms. They can have a different perspective on the problem and can make varying useful predictions. Logistic regression is a simple and reliable method to learn how the predictions from the above two methods can be combined. It produces binary outputs, so it is suitable for binary classification problems.

	Decision Trees	Neural Network	Naïve Bayes	K-Nearest Neighbor	Support Vector Machine
Proposed By	Quinlan	Rosenblatt	Duda and Hurt	Cover and Hart	Vapnik
Accuracy in general	Good	V. Good	Average	Good	Excellent
Speed of learning	V. Good	Average	Excellent	Excellent	Average
Speed of classification	Excellent	Excellent	Excellent	Average	Excellent
Tolerance to missing values	V. Good	Average	Excellent	Average	Good
Tolerance to irrelevant attributes	V. Good	Average	Good	Good	Excellent
Tolerance to redundant attributes	Good	Good	Average	Good	V. Good
Tolerance to highly interdependent attributes	Good	V. Good	Average	Average	V. Good
Dealing with discrete/ binary/continuous attributes	All	Not discrete	Not continuous	All	Not discrete
Tolerance to noise	Good	Good	V. Good	Average	Good
Dealing with danger of overfitting	Good	Average	V. Good	V. Good	Good
Attempts for incremental learning	Good	V. Good	Excellent	Excellent	Good
Explanation ability/ transparency of knowledge/ classification	Excellent	Average	Excellent	Good	Average
Support Multiclassification	Excellent	Naturally extended	Naturally extended	Excellent	Binary Classifier

6.2 RESULTS AND DISCUSSION

This study found that during 2014-15 widespread declines in life expectancy occurred across high income countries. Of 18 countries, 12 experienced declines in life expectancy for women and 11 for men. This is the first time in recent decades that these many high income countries simultaneously experienced such large declines in life expectancy for both men and women. The magnitude of these declines are fairly large compared with previous declines. These most recent declines were around 0.20 years on average for both men and women—roughly twice as large as the average past declines for women and 70% larger for men. In other words, these recent declines were notable both for the number of countries and for the magnitude of the declines.

These increases in life expectancy gaps between the USA and other high income countries are substantial. If life expectancy in the other countries was frozen at their 2016 levels while life expectancy in the USA was allowed to increase at the rate of improvement it experienced in the 2000s—a period of fairly rapid increase in life expectancy for the USA (1.7 years per decade for women and 2.1 years per decade for men), it would take American women 18 years to match the average of the other countries and 34 years to match the world leader, while American men would need 16 years and 2.5 decades, respectively. If, instead, the USA's current slow rate of increase in life expectancy was to hold (0.32 and 0.06 years per decade for women and men, respectively), it would take American men and women more than a century to reach the average life expectancy levels of the other countries.

Although other high income countries experienced recent declines in life expectancy, different factors appear to have been responsible for the declines in the USA. The USA experienced larger declines in life expectancy at younger ages and relatively small declines in life expectancy at older ages. In the other countries, declines at older ages were largely responsible for the declines in life expectancy at birth. The sizable declines in life expectancy at younger ages for American men and women are strongly related to the USA's ongoing, large scale drug overdose epidemic stemming from misuse of prescription opioids, heroin, and fentany and from external causes. This is particularly true for American men, for whom drug overdose increased sharply in the past two years. Previous studies have documented that in international comparisons of life expectancy, the USA performs relatively well at the older ages but poorly at the youngest ages. The age pattern of the recent declines in life expectancy serves to heighten the already sizable US disadvantage at younger ages while reinforcing its more favorable performance at older ages.

Italy experienced the largest declines in life expectancy during 2014-15, roughly half a year for both men and women. The three causes of death contributing most to these declines were circulatory diseases, nervous system diseases, and external causes for men, and circulatory diseases, mental disorders, and respiratory diseases for women (supplementary figures A2 and A3). During 2015-16, life expectancy increased by about 0.45 years, returning Italian men to their 2014 life expectancy

level and Italian women to a 10th of a year below their life expectancy in 2014.

7. CONCLUSION AND FUTURE ENHANCEMENT

Life expectancy declined across many high income countries during 2014-15. In some of these countries, life expectancy rebounded in the following year. Though this suggests that these declines may be a fluctuation rather than a new trend, it remains to be seen whether such simultaneous declines across high income countries will become more common in the coming years or whether these countries will continue to achieve robust gains in longevity.

Important exceptions to this rebound were the UK and the USA, which experienced either continued declines or stagnation during 2015-16. Life expectancy trends in the USA appear to be strongly related to its ongoing opioid epidemic; while drug overdose mortality is high in several high income countries (eg, Sweden, Norway), it seems that the American epidemic has not yet spilled over to most other high income countries.However, there are indications of recent increases in drug overdose in the other Anglophone countries (Australia, Canada, and the UK), although levels of mortality due to drug overdose in these countries remain much lower than in the USA. For the UK, the declines in 2014-15 were concentrated at older ages (≥65). Respiratory diseases, circulatory diseases, Alzheimer's disease, nervous system diseases, and mental disorders, as well as drug overdose for men, were key drivers of these declines. Previous studies**1** of England and Wales suggested that decreases in funding to healthcare and social welfare programs may be driving increases in mortality among older adults, but further testing of this hypothesis is needed. While this study does not examine how socioeconomic inequality may be contributing to these declines in life expectancy across countries, it is possible that greater inequality within a country renders that country more vulnerable to declines in life expectancy. Previous studies have found a negative relation between income inequality and poverty and life expectancy across countries, and countries such as the USA, which is known to have high levels of socioeconomic inequality, have experienced recent declines in life expectancy among those of lower socioeconomic status. Furthermore, countries with greater inequality between social classes may be more susceptible to phenomena such as drug overdose epidemics.

Policies that have been suggested to address the US's drug overdose epidemic include greater implementation and use of prescription drug monitoring programs, expanding access to substance misuse treatment programs, establishing supervised injection centers and needle exchange programs, increasing the availability of naloxone, and addressing the underlying social and economic conditions that may underpin drug use. These policies might have relevance not only for the USA but also for other high income countries that have also reported recent increases in opioid prescribing, including Australia, Canada, Denmark, Finland, Germany, Sweden, and the UK.

In many of these high income countries, mortality from influenza, pneumonia, other respiratory diseases, and cardiovascular disease played an important role in the recent declines in life expectancy. Countries should continue to encourage high rates of vaccination against influenza, increase awareness of the importance of vaccination, and maintain sufficient stocks of antiviral drugs such as Tamiflu. The influenza vaccine was known to be a poor match to the predominant influenza strain in 2014-15, which resulted in lower efficacy of the influenza vaccine. Once information about expected vaccine efficacy is known, public health and healthcare systems can take proactive approaches to reducing influenza related mortality, including increasing awareness of influenza symptoms and complications and intensifying outreach efforts, especially for vulnerable populations (eg, children, elderly people, and those who are immunocompromised).

This study also highlights the importance of maintaining and updating vital registration systems. A large number of high income countries simultaneously experienced declines in life expectancy, but it was not possible to identify this phenomenon until several years after the fact. To date, cause specific mortality data for 2016 are not yet available for the complete set of countries, and almost no countries have all cause mortality data for 2017 available. In the interests of timely identification of shared threats to life expectancy and population health more broadly, countries should make the release of accurate vital statistics data a priority. This would contribute to improved monitoring of trends in life expectancy and population health worldwide.

Causes of death that predominate at older ages, including influenza and pneumonia, cardiovascular disease, Alzheimer's disease, and other nervous system diseases were primarily responsible for declines in the other high income countries. H3N2 viruses, which are associated with increased hospital admissions and deaths, predominated during a particularly severe 2014-15 influenza season and contributed to reduced efficacy of influenza vaccine and increased mortality.It is possible that those with Alzheimer's disease and other nervous system disorders had an increased risk of mortality in this year owing to influenza but their underlying cause of death was ultimately coded as being due to a non-influenza cause. Studies have also found evidence that influenza may precipitate cardiovascular events such as acute myocardial infarction and consequently mortality from cardiovascular disease.

8.REFERENCES

1. Life expectancy at birth, total (years) [https://data.worldbank.org/indicator/ SP.DYN.LE00.IN]. 2. Schultz TP. Health human capital and economic development. J Afr Econ. 2010;19:iii12–80.
3. Preston SH. The changing relation between mortality and level of economic development. Popul Stud. 1975;29:231–48.
4. Jetter M, Laudage S, Stadelmann D. The intimate link between income levels and life expectancy: global evidence from 213 years*. Soc Sci Q. 2019; 100:1387–403.
5. Bloom DE, Canning D. Commentary: the Preston c

6. Baum F, Popay J, Delany-Crowe T, Freeman T, Musolino C, Alvarez-Dardet C, Ariyaratne V, Baral K, Basinga P, Bassett M, et al. Punching above their weight: a network to understand broader determinants of increasing life expectancy. Int J Equity Health. 2018;17:117.

7. Halstead S, Walsh J, Warren K. Good health at low cost. New York: The Rockefeller Foundation; 1985. New York: The Rockefeller Foundation; 1985.

8. Balabanova D, McKee M, Mills A. Good health at low cost' 25 years on. What makes a successful health system? London: School of Hygeine and Tropical Medicine; 2011.

9. Baum F. The new public health. 4th ed. Melbourne: Oxford University Press; 2016.

10. CSDH. Closing the gap in a generation. In: Health equity through action on the social detrminants of health. Geneva: World Health Organization; 2008.

11. Fullman N, GBD 2016 SDG collaborators, et al. Measuring progress and projecting attainment on the basis of past trends of the health-related sustainable development goals in 188 countries: an analysis from the Global Burden of Disease Study 2016. Lancet. 2017;390:1423–59.

12. Acemoglu D, Robinson JA: Why nations fail: the origins of power, prosperity, and poverty. Currency; 2012.

13. Greenhalgh T, Thorne S, Malterud K. Time to challenge the spurious hierarchy of systematic over narrative

reviews? Eur J Clin Investig. 2018;48: e12931.

14. GDP per capita and GDP (current US$) [https://data.worldbank.org/ indicator/NY.GDP.PCAP.CD].

15. Rowe G, Wright G. The Delphi technique as a forecasting tool: issues and analysis. Int J Forecast. 1999;15:353–75.

16. Ethiopia: life expectancy, GDP [https://www.theglobaleconomy.com/ Ethiopia/Life_expectancy/].

17. Tranvåg EJ, Ali M, Norheim OF. Health inequalities in Ethiopia: modeling inequalities in length of life within and between population groups. Int J Equity Health. 2013;12:52.

18. Ruducha J, Mann C, Singh NS, Gemebo TD, Tessema NS, Baschieri A, Friberg I, Zerfu TA, Yassin M, Franca GA, Berman P. How Ethiopia achieved millennium development goal 4 through multisectoral interventions: a countdown to 2015 case study. Lancet Glob Health. 2017;5:e1142–51.

19. Doherty T, Rohde S, Besada D, Kerber K, Manda S, Loveday M, Nsibande D, Daviaud E, Kinney M, Zembe W, et al. Reduction in child mortality in Ethiopia: analysis of data from demographic and health surveys. J Glob Health. 2016;6:–020401.

20. CSA, ICF. Ethiopian demographic health survey 2016. Addis Ababa and Calverton: Central Statistical Agency (Ethiopia) and ICF International; 2018. p. 36–41.

21. Girum T, Wasie A, Worku A. Trend of HIV/AIDS for the last 26 years and predicting achievement of the 90–90-90 HIV prevention targets by 2020 in Ethiopia: a time series analysis. BMC Infect Dis. 2018;18:320.

22. van Erwin V. Perpetuating power: Ethiopia's political settlement and the organization of security. In: Dynamics of political power in Ethiopia: past and present. The Clingendael Institute: The Hague; 2016.

23. Iso H, Noda H, Ikeda A, Yamagishi K, Inoue M, Iwasaki M, Tsugane S, Grp JS. The impact of C-reactive protein on risk of stroke, stroke subtypes, and ischemic heart disease in middle-aged Japanese: the Japan public health center-based study. J Atheroscler Thromb. 2012;19:756–66.

24. de Waal A. The theory and practice of Meles Zenawi. Afr Aff. 2012;112:148– 55.

25. Emanuele F: Developmental state, economic transformation and social diversification in Ethiopia. ISPI Analysis: Instituto per gli Studi di Politica Internazionale 2013:3.

26. Hayek FA. The road to serfdom. George Routledge and Sons: London; 1944.

27. Labonté R. The austerity agenda: how did we get here and where do we go next? Crit Public Health. 2012;22:257–65.

28. UNESCO. Adult and youth literacy: national, regional and global trends, 1985–2015. Montreal: Institute for Statistics; 2013.

29. School enrollment, primary, female (% gross) - Ethiopia, Brazil, United States [https://data.worldbank.org/indicator/SE.PRM.ENRR.FE?locations=ET-BR-US].

30. Karin H, Dehab B, Asegedech B, Anbesu B, Nuri K. Taking stock of Girls' education in Ethiopia: preparing for ESD P III; 2005.

31. Ethiopia: Statistics [http://www.unicef.org/infobycountry/ethiopia_statistics. html].

32. Percentage of women in national parliaments [https://data.ipu.org/womenranking?month=9&year=2019].

33. Helmut K. Primary health care in Ethiopia: from Haile Sellassie to Meles Zenawi. Northeast Afr Stud. 1998;5:83–113.