Project Title	**Life Expectancy Analysis**
language	Machine learning, python, SQL, Excel
Tools	VS code, Jupyter notebook
Domain	Data Analyst
Project Difficulties level	Advance

Dataset : Dataset is available in the given link. You can download it at your convenience.

Click here to download data set

About Dataset

Context

Although there have been a lot of studies undertaken in the past on factors affecting life expectancy considering demographic variables, income composition and mortality rates. It was found that affect of immunization and human development index was not taken into account in the past. Also, some of the past research was done considering multiple linear regression based on a data set of one year for all the countries. Hence, this gives motivation to resolve both the factors stated previously by formulating a

Immunizations and human development index regression model based on mixed effects model and multiple linear regression while considering data from a period of 2000 to 2015 for all the countries. Important immunization like Hepatitis B, Polio and Diphtheria will also be considered. In a nutshell, this study will focus on immunization factors, mortality factors, economic factors, social factors and other health related factors as well. Since the observations in this dataset are based on different countries, it will be easier for a country to determine the predicting factor which is contributing to lower value of life expectancy. This will help in suggesting a country which area should be given importance in order to efficiently improve the life expectancy of its population.

Out of the given factors

Content

The project relies on accuracy of data. The Global Health Observatory (GHO) data repository under World Health Organization (WHO) keeps track of the health status as well as many other related factors for all countries The data-sets are made available to public for the purpose of health data analysis. The data-set related to life expectancy, health factors for 193 countries has been collected from the same WHO data repository website and its corresponding economic data was collected from United Nation website. Among all categories of health-related factors only those critical factors were chosen which are more representative. It has been observed that in the past 15 years , there has been a huge development in health sector resulting in improvement of human mortality rates especially in the developing nations in comparison to the past 30 years. Therefore, in this project we have considered data from year 2000-2015 for 193 countries for further analysis. The individual data files have been merged together into a single data-set. On initial visual inspection of the data showed some missing values. As the data-sets were from WHO, we found no evident errors. Missing data was handled in R software by using Missmap command. The result indicated that most of the missing data was for population, Hepatitis B and GDP. The missing data were from less known countries like Vanuatu, Tonga, Togo, Cabo Verde etc. Finding all data for these countries was difficult and hence, it was decided that we exclude these countries from the final model data-set. The final merged file(final dataset) consists of

Need to fix this

Missing data was found out and the countries with the missing datasets were excluded from the final data-set

22 Columns and 2938 rows which meant 20 predicting variables. All predicting variables was then divided into several broad categories:Immunization related factors, Mortality factors, Economical factors and Social factors.

Acknowledgements

The data was collected from WHO and the United Nations website with the help of Deeksha Russell and Duan Wang.

Inspiration

The data-set aims to answer the following key questions:

1. Do various predicting factors which have been chosen initially really affect the Life expectancy? What are the predicting variables actually affecting life expectancy?
2. Should a country having a lower life expectancy value(<65) increase its healthcare expenditure in order to improve its average lifespan?
3. How does Infant and Adult mortality rates affect life expectancy?
4. Does Life Expectancy has positive or negative correlation with eating habits, lifestyle, exercise, smoking, drinking alcohol etc.
5. What is the impact of schooling on the lifespan of humans?
6. Does Life Expectancy have positive or negative relationship with drinking alcohol?
7. Do densely populated countries tend to have lower life expectancy?
8. What is the impact of Immunization coverage on life Expectancy?

Analyzing life expectancy data involves examining various factors that may affect life expectancy rates in different countries. For a finance analyst project, we can approach this analysis by looking at economic indicators, healthcare spending, and other socio-economic factors that might influence life expectancy. Here, I'll outline a basic life expectancy analysis project using Python, incorporating data collection, cleaning, EDA (Exploratory Data Analysis), and some basic statistical analysis.

Example: You can get the basic idea how you can create a project from here

Step-by-Step Life Expectancy Analysis

1. Data Collection

For this analysis, we'll use a publicly available dataset. Let's use the "Life Expectancy Data" from the World Health Organization (WHO), which you can find on Kaggle or directly from WHO's repository.

2. Data Cleaning

We'll clean the data by handling missing values, duplicates, and any anomalies.

3. Exploratory Data Analysis (EDA)

We'll perform EDA to understand the distribution and relationships in the data.

4. Statistical Analysis and Visualization

We'll conduct statistical tests and visualize the relationships between life expectancy and various economic indicators.

Here is a sample code for this analysis:

```
import pandas as pd
import numpy as np
import seaborn as sns
```

```
import matplotlib.pyplot as plt
from scipy import stats

# Load the dataset
url = "https://path/to/your/life_expectancy_data.csv"  # Replace with the actual URL
or file path
df = pd.read_csv(url)

# Inspect the data
print(df.head())
print(df.info())
print(df.describe())

# Data Cleaning
# Handle missing values
df = df.fillna(df.mean())

# Check for duplicates
df = df.drop_duplicates()

# Basic EDA
# Distribution of life expectancy
plt.figure(figsize=(10, 6))
sns.histplot(df['Life expectancy '], bins=30, kde=True)
plt.title('Distribution of Life Expectancy')
plt.xlabel('Life Expectancy')
plt.ylabel('Frequency')
plt.show()
```

```
# Correlation matrix
plt.figure(figsize=(12, 8))
correlation_matrix = df.corr()
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm')
plt.title('Correlation Matrix')
plt.show()

# Relationship between Life Expectancy and GDP
plt.figure(figsize=(10, 6))
sns.scatterplot(x='GDP', y='Life expectancy ', data=df)
plt.title('Life Expectancy vs GDP')
plt.xlabel('GDP')
plt.ylabel('Life Expectancy')
plt.show()

# Statistical Analysis
# Correlation between life expectancy and key indicators
gdp_corr, _ = stats.pearsonr(df['GDP'], df['Life expectancy '])
print(f'Correlation between GDP and Life Expectancy: {gdp_corr:.2f}')

# Hypothesis Testing
# Is there a significant difference in life expectancy between high-income and
low-income countries?
high_income = df[df['Income classification according to World Bank'] == 'High']['Life
expectancy ']
low_income = df[df['Income classification according to World Bank'] == 'Low']['Life
expectancy ']

t_stat, p_value = stats.ttest_ind(high_income, low_income)
```

```
print(f'T-test between High and Low Income Countries: t-statistic={t_stat:.2f}, 
p-value={p_value:.2f}')

if p_value < 0.05:
    print("The difference in life expectancy between high-income and low-income 
countries is statistically significant.")
else:
    print("There is no statistically significant difference in life expectancy between 
high-income and low-income countries.")

# Regression Analysis
import statsmodels.api as sm

# Regression model: Life Expectancy ~ GDP + Healthcare Expenditure + Education 
Index + other factors
X = df[['GDP', 'Total expenditure', 'Schooling', 'Adult Mortality']]  # Add relevant 
features
y = df['Life expectancy ']

# Add constant to the model (intercept)
X = sm.add_constant(X)

# Fit the model
model = sm.OLS(y, X).fit()

# Print model summary
print(model.summary())

# Visualize the regression results
```

```
plt.figure(figsize=(10, 6))
sns.regplot(x='GDP', y='Life expectancy ', data=df)
plt.title('Regression Line: Life Expectancy vs GDP')
plt.xlabel('GDP')
plt.ylabel('Life Expectancy')
plt.show()
```

Explanation:

1. **Data Collection:** We load the dataset into a Pandas DataFrame.
2. **Data Cleaning:** Handle missing values and remove duplicates.
3. **EDA:**
    - Visualize the distribution of life expectancy.
    - Display a correlation matrix to understand relationships between variables.
    - Plot scatter plots to visualize the relationships between life expectancy and economic indicators.
4. **Statistical Analysis:**
    - Calculate and print the Pearson correlation coefficient between GDP and life expectancy.
    - Perform a t-test to compare life expectancy between high-income and low-income countries.
5. **Regression Analysis:** Fit a linear regression model to understand the impact of various factors on life expectancy and visualize the regression line.

This project provides a structured approach to analyzing life expectancy data, incorporating various statistical and visualization techniques to derive insights.

Sample code and output

Dataset Metadata

Field	Description
Country	Country
Year	Year
Status	Developed or Developing status
Life expectancy	Life Expectancy in age
Adult Mortality	Adult Mortality Rates of both sexes (probability of dying between 15 and 60 years per 1000 population)
infant deaths	Number of Infant Deaths per 1000 population
Alcohol	Alcohol, recorded per capita (15+) consumption (in litres of

	pure alcohol)
percentage expenditure	Expenditure on health as a percene of Gross Domestic Product per capita(%)
Hepatitis B	Hepatitis B (HepB) immunization coverage among 1-year-olds (%)
Measles	Measles - number of reported cases per 1000 population
BMI	Average Body Mass Index of entire population
under-five deaths	Number of under-five deaths per 1000 population
Polio	Polio (Pol3) immunization coverage among 1-year-olds (%)
Total expenditure	General government expenditure on health as a percene of total government expenditure (%)
Diphtheria	Diphtheria tetanus toxoid and pertussis (DTP3) immunization

	coverage among 1-year-olds (%)
HIV/AIDS	Deaths per 1 000 live births HIV/AIDS (0-4 years)
GDP	Gross Domestic Product per capita (in USD)
Population	Population of the country
thinness 1-19 years	Prevalence of thinness among children and adolescents for Age 10 to 19 (%)
thinness 5-9 years	Prevalence of thinness among children for Age 5 to 9(%)
Income composition of resources	Income composition of resources
Schooling	Number of years of Schooling(years)

Load Important Packages

In [2]:

```
import pandas as pd
import numpy as np
from sklearn.impute import SimpleImputer
import seaborn as sns
import matplotlib.pyplot as plt
plt.style.use('ggplot')
import plotly.express as px
import plotly.graph_objects as go
from sklearn.preprocessing import StandardScaler, LabelEncoder
import warnings
warnings.filterwarnings('ignore')
```

Load Dataset

In [3]:

```
df = pd.read_csv('/kaggle/input/life-expectancy-who/Life
Expectancy Data.csv')
```

In [4]:

```
df.head(3)
```

Out[4]:

	Country	Year	Status	Life expectancy	Adult Mortality	infant deaths	Alcohol	percentage expenditure	Hepatitis B	Measles	...	Polio	Total expenditure	Diphtheria	HIV/AIDS	GDP	Population	thinness 1-19 years	thinness 5-9 years	Income composition of resources	Schooling
0	Afghanistan	2015	Developing	65.0	263.0	62	0.01	71.279624	65.0	1154	...	6.0	8.16	65.0	0.1	584.259210	33736494.0	17.2	17.3	0.479	10.1

1	Afghanistan	2014	Developing	59.9	271.0	64	0.01	73.523582	62.0	492	...	58.0	8.18	62.0	0.1	612.696514	327582.0	17.5	17.5	0.476	10.0
2	Afghanistan	2013	Developing	59.9	268.0	66	0.01	73.219243	64.0	430	...	62.0	8.13	64.0	0.1	631.744976	31731688.0	17.7	17.7	0.470	9.9

3 rows × 22 columns

In [5]:

```
df.shape
```

Out[5]:

```
(2938, 22)
```

Data Cleaning

In [6]:

```python
df.isnull().sum()
```

Out[6]:

```
Country                            0
Year                               0
Status                             0
Life expectancy                   10
Adult Mortality                   10
infant deaths                      0
Alcohol                          194
percentage expenditure             0
Hepatitis B                      553
Measles                            0
 BMI                              34
under-five deaths                  0
Polio                             19
Total expenditure                226
Diphtheria                        19
 HIV/AIDS                          0
GDP                              448
Population                       652
 thinness  1-19 years             34
 thinness 5-9 years               34
```

```
Income composition of resources    167
Schooling                          163

dtype: int64
```

In [7]:

```
for cols in df.columns:
    if df[cols].isnull().sum()>0:
        print(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
```

In [8]:

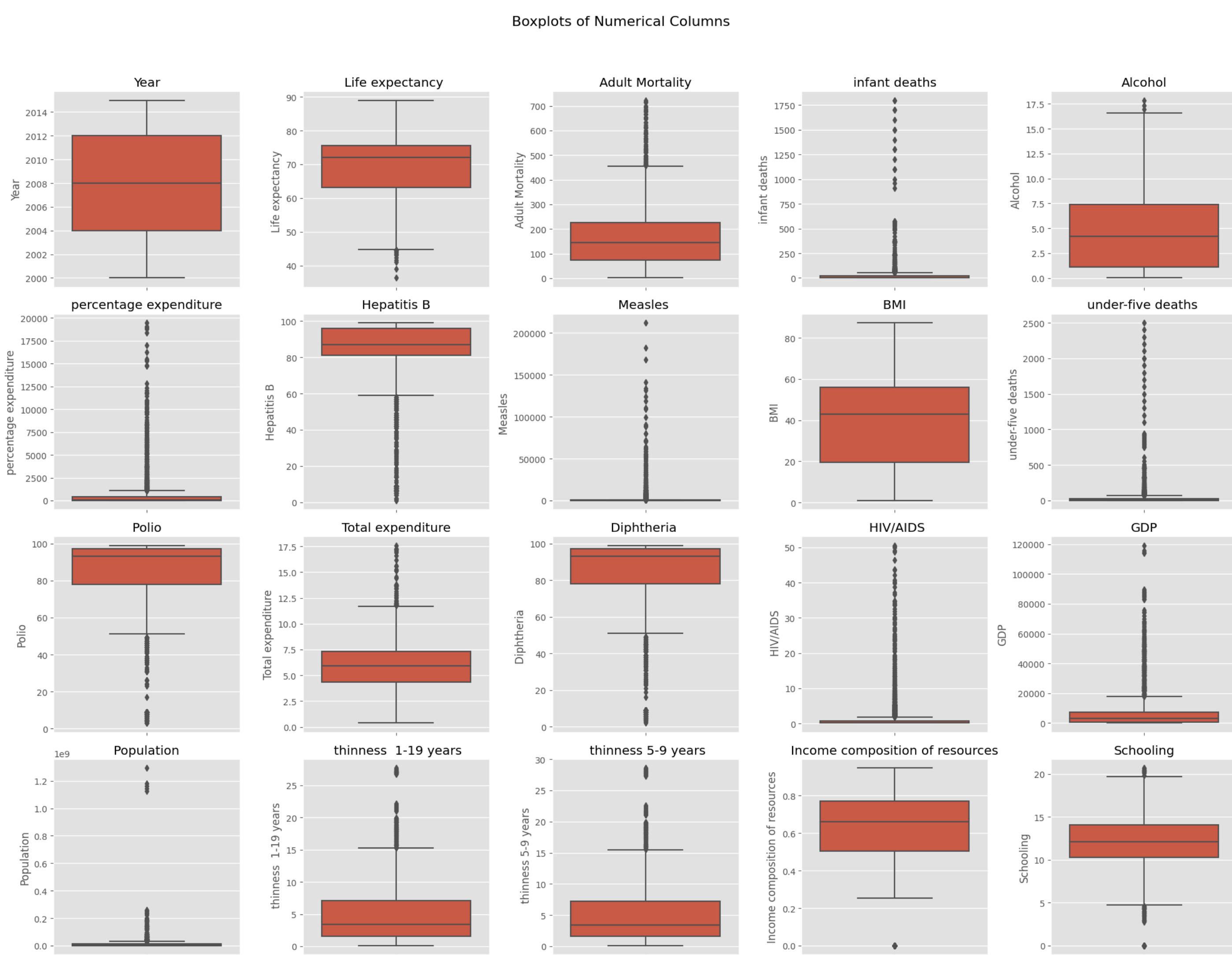
```
imputer = SimpleImputer(missing_values=np.nan, strategy='mean',
fill_value=None)
for cols in df.columns:
    if df[cols].isnull().sum()>0:
        df[cols] =  imputer.fit_transform(df[[cols]])
```

Check Outliers

In [9]:

```
# Select only numerical columns
numerical_cols = df.select_dtypes(include=['float64',
'int64']).columns
fig, axes = plt.subplots(4, 5, figsize=(20, 16))
fig.suptitle('Boxplots of Numerical Columns', fontsize=16)

# Flatten the axes array for easy iteration
axes = axes.flatten()

# Plot boxplots for each numerical column
for i, col in enumerate(numerical_cols):
    sns.boxplot(y=df[col], ax=axes[i])
    axes[i].set_title(col)
```

```python
# Remove any empty subplots
for j in range(len(numerical_cols), len(axes)):
    fig.delaxes(axes[j])


plt.tight_layout(rect=[0, 0.03, 1, 0.95])
plt.show()
```




```
In [10]:
```

```
# Specify the list of columns you want to handle outliers for
outlier_cols = [
    'Adult Mortality', 'infant deaths', 'Alcohol', 'percentage expenditure',
    'Hepatitis B', 'Measles ', ' BMI ', 'under-five deaths ', 'Polio',
    'Total expenditure', 'Diphtheria ', ' HIV/AIDS', 'GDP', 'Population',
    ' thinness  1-19 years', ' thinness 5-9 years',
    'Income composition of resources', 'Schooling'
]

# Perform outlier handling for each specified column
for col_name in outlier_cols:
    # Calculate quartiles and IQR
    q1 = df[col_name].quantile(0.25)
    q3 = df[col_name].quantile(0.75)
    iqr = q3 - q1

    # Define the lower and upper bounds for outliers
    lower_bound = q1 - 1.5 * iqr
    upper_bound = q3 + 1.5 * iqr

    # Replace outliers with the mean value of the column
    df[col_name] = np.where((df[col_name] > upper_bound) |
```

```
(df[col_name] < lower_bound),
                                np.mean(df[col_name]),
df[col_name])
```

In [11]:

```
df.shape
```

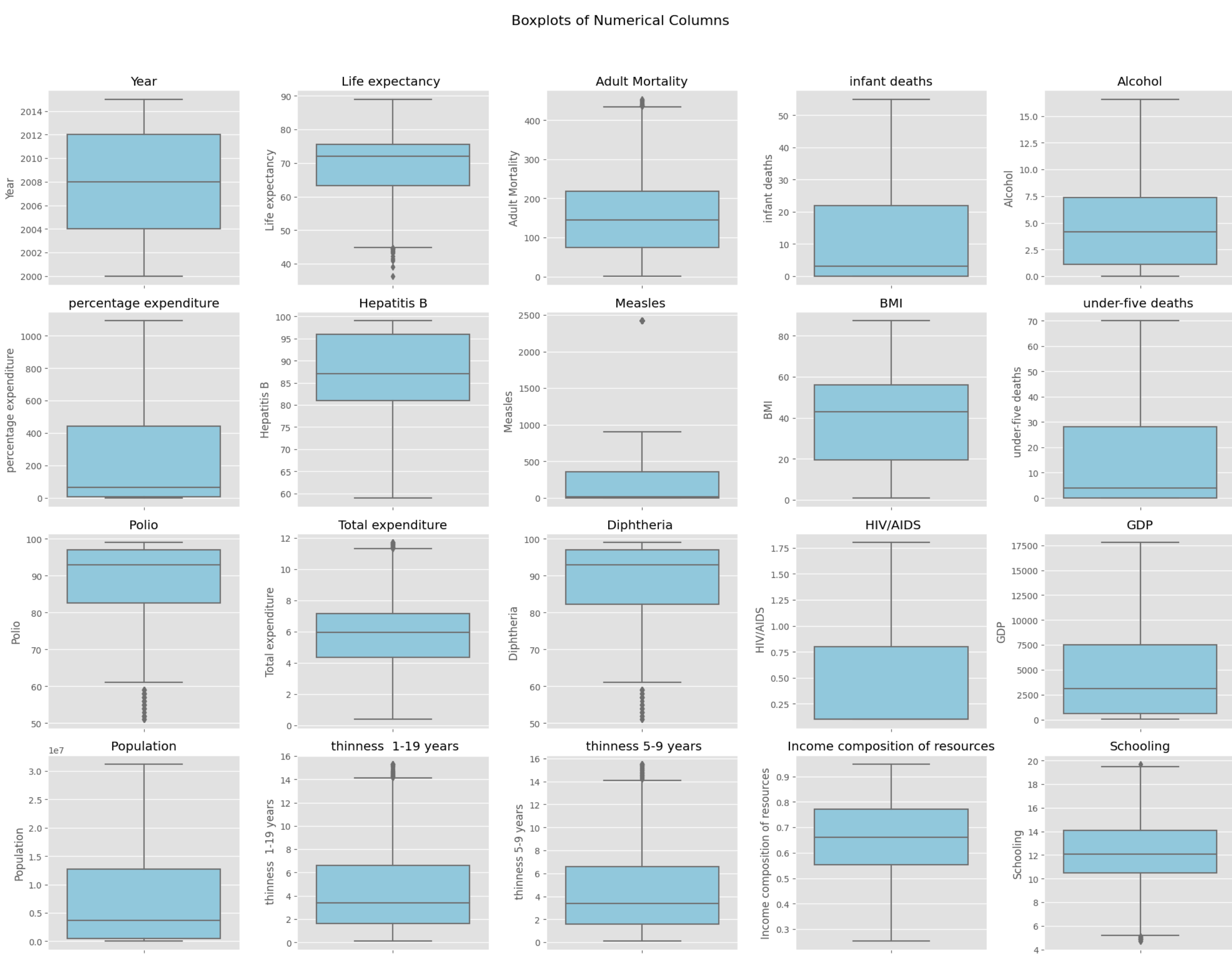
Out[11]:

```
(2938, 22)
```

In [12]:

```
# Select only numerical columns
numerical_cols = df.select_dtypes(include=['float64',
'int64']).columns
fig, axes = plt.subplots(4, 5, figsize=(20, 16))
fig.suptitle('Boxplots of Numerical Columns', fontsize=16)

# Flatten the axes array for easy iteration
axes = axes.flatten()

# Plot boxplots for each numerical column
for i, col in enumerate(numerical_cols):
    sns.boxplot(y=df[col], ax=axes[i], color='skyblue')
```

```
    axes[i].set_title(col)

# Remove any empty subplots
for j in range(len(numerical_cols), len(axes)):
    fig.delaxes(axes[j])

plt.tight_layout(rect=[0, 0.03, 1, 0.95])
plt.show()
```



Exploratory Data Analysis

In [13]:

```
df.head()
```

Out[13]:

	Country	Year	Status	Life expectancy	Adult Mortality	infant deaths	Alcohol	percentage expenditure	Hepatitis B	Measles	...	Polio	Total expenditure	Diphtheria	HIV/AIDS	GDP	Population	thinness 1-19 years	thinness 5-9 years	Income composition of resources	Schooling

0	Afghanistan	2015	Developing	65.0	263.0	30.303948	0.01	71.279624	65.0	2419.592424	...	82.550188	8.16	65.0	0.1	584.259210	1.275338e+07	4.839704	4.870317	0.479	10.1
1	Afghanistan	2014	Developing	59.9	271.0	30.303948	0.01	73.523582	62.0	492.000000	...	58.000000	8.18	62.0	0.1	612.696514	3.275820e+05	4.839704	4.870317	0.476	10.0
2	Afghanistan	2013	Developin	59.9	268.0	30.30394	0.01	73.219243	64.0	430.000000	...	62.000000	8.13	64.0	0.1	631.744976	1.275338e+07	4.83970	4.87031	0.470	9.9

			g			8						0						4	7		
3	Afghanistan	2012	Developing	59.5	272.0	30.303948	0.01	78.184215	67.0	2419.59224	...	67.000000	8.52	67.0	0.1	669.959000	3.696958e+06	4.839704	4.870317	0.463	9.8
4	Afghanistan	2011	Developing	59.2	275.0	30.303948	0.01	7.097109	68.0	2419.59224	...	68.000000	7.87	68.0	0.1	63.537231	2.978599e+06	4.839704	4.870317	0.454	9.5

5 rows × 22 columns

In [14]:

```
df.Country.value_counts()
```

```
Out[14]:
Country
Afghanistan            16
Peru                   16
Nicaragua              16
Niger                  16
Nigeria                16
                       ..
Niue                    1
San Marino              1
Nauru                   1
Saint Kitts and Nevis   1
Dominica                1

Name: count, Length: 193, dtype: int64
```

```
In [15]:
# Calculate the average life expectancy for each year
average_life_expectancy = df.groupby('Year')['Life expectancy
'].mean().reset_index()


# Create the interactive line plot
fig = px.line(average_life_expectancy, x='Year', y='Life
expectancy ',
              title='Average Life Expectancy over the Years',
              labels={'Year':'Year', 'Life expectancy':'Life
```

```
Expectancy (years)'},
              template='plotly_dark')


# Show the plot
fig.show()
```

2000200220042006200820102012201467686970 71

Average Life Expectancy over the YearsYearLife expectancy

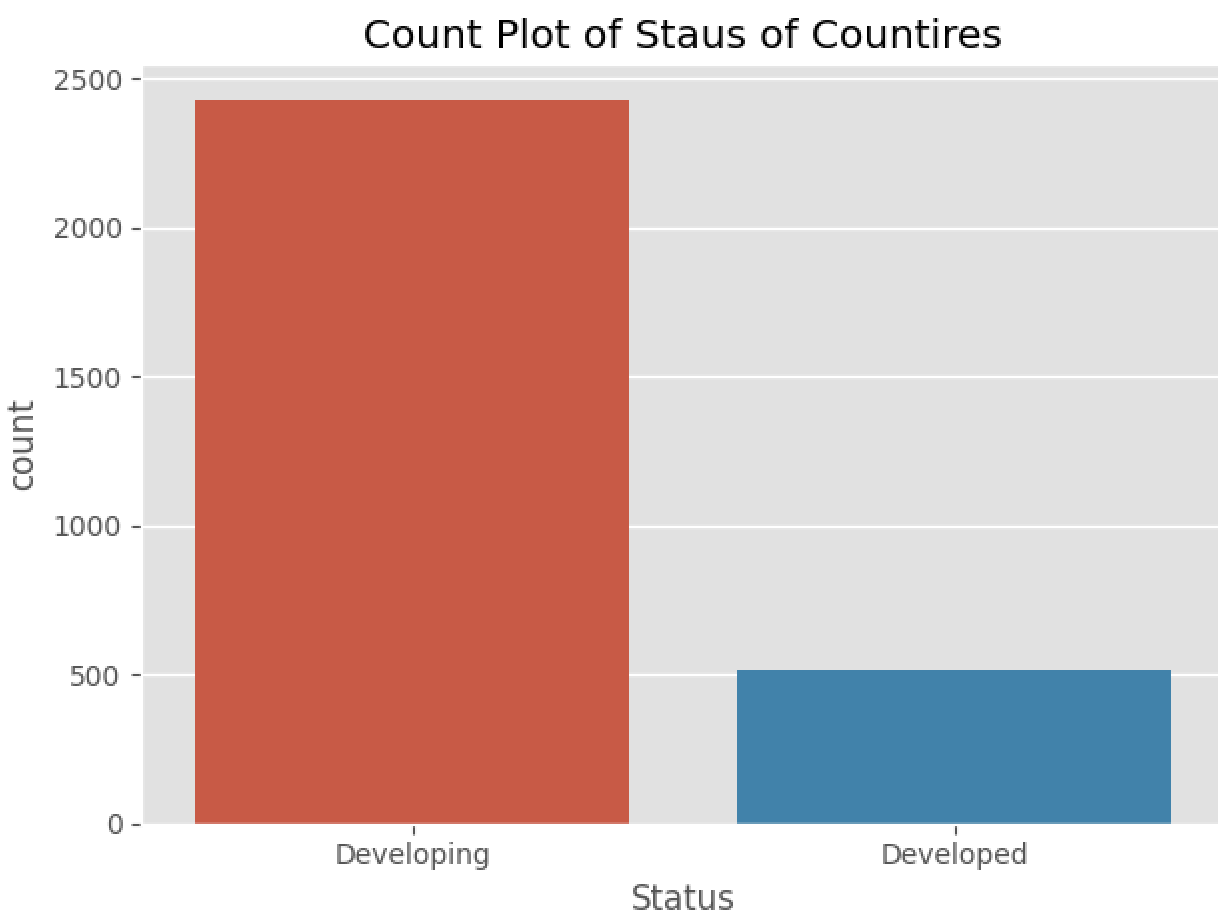
```
In [16]:
# Create the interactive scatter plot
fig = px.scatter(df, x='Population', y='Life expectancy ',
                 hover_name='Country',
#                  color='Status',
                 animation_frame='Year',
                 title='Population vs Life Expectancy',
                 labels={'Population':'Population', 'Life
expectancy':'Life Expectancy (years)'},
                 template='plotly_dark')


# Show the plot
fig.show()
```

05M10M15M20M25M30M5060708090

Year=201520152013201120092007200520032001Population vs Life ExpectancyPopulationLife expectancy▶■

In [17]:

```
sns.countplot(x=df['Status'])
plt.title('Count Plot of Staus of Countires')
plt.tight_layout()
plt.show()
```



In [18]:

```
life_expact_status = df.groupby('Status')['Life expectancy
'].mean().reset_index()
fig = px.histogram(life_expact_status, x = 'Status', y='Life
expectancy ',
                  color='Status')
fig.update_layout(
    title=dict(text='<b>Average Life Expactancy for Status of
Country</b>', x=0.5)
)

fig.show()
```

DevelopedDeveloping01020304050607080

StatusDevelopedDevelopingAverage Life Expactancy for Status of CountryStatussum of Life expectancy

In [19]:

```
# Calculate the average life expectancy and average alcohol
consumption for each year
average_data = df.groupby('Year').agg({
    'Life expectancy ': 'mean',
    'Alcohol': 'mean'
}).reset_index()
```

```
# Create the interactive plot with dual y-axes
fig = go.Figure()

# Add life expectancy trace
fig.add_trace(go.Scatter(x=average_data['Year'],
y=average_data['Life expectancy '],
                         mode='lines+markers', name='Life
Expectancy',
                         yaxis='y1'))

# Add alcohol consumption trace
fig.add_trace(go.Scatter(x=average_data['Year'],
y=average_data['Alcohol'],
                         mode='lines+markers', name='Alcohol
Consumption',
                         yaxis='y2'))

# Update layout for dual y-axes
fig.update_layout(
    title='Life Expectancy and Alcohol Consumption over the
Years',
    xaxis=dict(title='Year'),
    yaxis=dict(title='Life Expectancy (years)', side='left'),
    yaxis2=dict(title='Alcohol Consumption (liters)',
```

```
side='right', overlaying='y'),
    template='plotly_dark'
)

# Show the plot
fig.show()
```

2000200520102015676869707 13.23.43.63.844.24.44.64.85

Life ExpectancyAlcohol ConsumptionLife Expectancy and Alcohol Consumption over the YearsYearLife Expectancy (years)Alcohol Consumption (liters)

In [20]:

```
fig = px.bar(df.groupby('Status', as_index=False).agg({'Alcohol':'mean'}), y='Alcohol', x='Status',
                 title='Average Alcohol consumption of Developing and Developed Countries',
                 labels={'Alcohol':'Alcohol Consumption (liters per capita)', 'Life expectancy':'Life Expectancy (years)'},
                 template='plotly_dark')

# Show the plot
fig.show()
```

DevelopedDeveloping0246810

Average Alcohol consumption of Developing and Developed CountriesStatusAlcohol Consumption (liters per capita)

In [21]:

```
df.head(2)
```

Out[21]:

	Country	Year	Status	Life expectancy	Adult Mortality	infant deaths	Alcohol	percentage expenditure	Hepatitis B	Measles	...	Polio	Total expenditure	Diphtheria	HIV/AIDS	GDP	Population	thinness 1-19 years	thinness 5-9 years	Income composition of resources	Schooling

																				s	
0	Afghanistan	2015	Developing	65.0	263.0	30.303948	0.01	71.279624	65.0	2419.59224	...	82.550188	8.16	65.0	0.1	584.259210	1.2753338e+07	4.839704	4.870317	0.479	10.1
1	Afghanistan	2014	Developing	59.9	271.0	30.303948	0.01	73.523582	62.0	492.000000	...	58.000000	8.18	62.0	0.1	612.696514	3.275820e+05	4.839704	4.870317	0.476	10.0

2 rows × 22 columns

In [22]:

```
aggregated_data = df.groupby('Schooling')['Life expectancy
'].mean().reset_index()
```

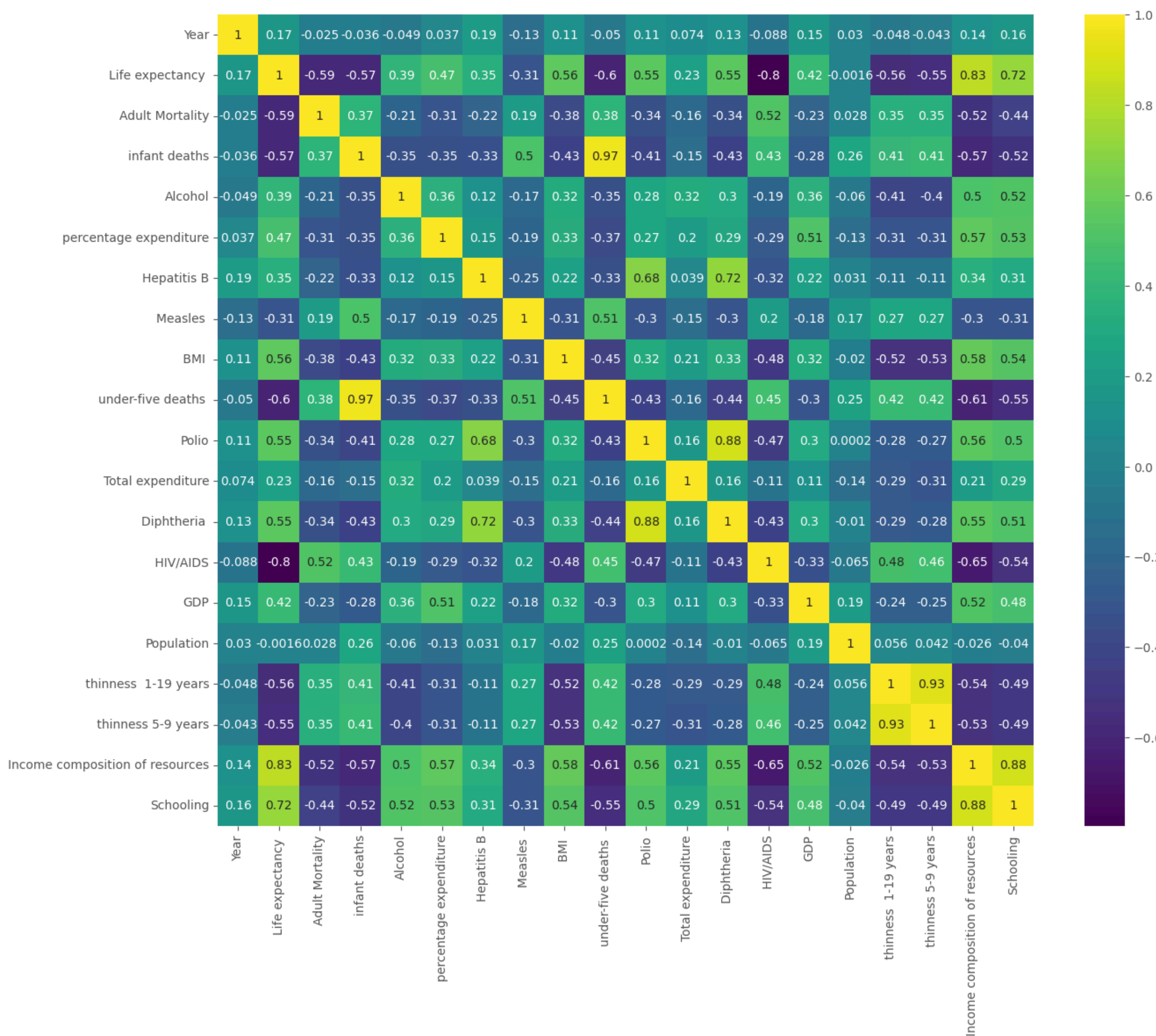
```python
# Create the interactive line plot
fig = px.line(aggregated_data, x='Schooling', y='Life expectancy ',
              title='Average Life Expectancy vs Years of Schooling',
              labels={'Schooling':'Years of Schooling', 'Life expectancy':'Life Expectancy (years)'},
              template='plotly_dark')

# Show the plot
fig.show()
```

6810121416185055606570758085

Average Life Expectancy vs Years of SchoolingYears of SchoolingLife expectancy

In [23]:

```python
plt.figure(figsize=(15, 12))
sns.heatmap(df[numerical_cols].corr(), cmap='viridis', annot=True)
plt.show()
```



Data Preprocessing

In [24]:

```
le = LabelEncoder()
cat_cols = df.select_dtypes(include = 'object').columns
for cols in cat_cols:
    df[cols] = le.fit_transform(df[cols])
```

In [25]:

```python
x =  df.drop(columns='Life expectancy ')
y = df['Life expectancy ']
```

In [26]:

```python
scaler = StandardScaler()
cols_to_scale = x.drop(columns='Status').columns

# for cols in cols_to_scale:
x[cols_to_scale] = scaler.fit_transform(x[cols_to_scale])
```

In [27]:

```python
x.head()
```

Out[27]:

	Count	Year	Stat	Adult	infantd	Alcoh	percenta	Hepat	Measl	BMI	...	Polio	Totalexpe	Diphth	HIV/AI	GDP	Popul	thinne	thinne	Income	Scho

	ry		us	Mortality	eaths	ol	geexpenditure	itisB	es				nditure	eria	DS		ation	ss1-19years	ss5-9years	compositionofresources	oling
0	-1.691042	1.621762	1	1.057680	1.351263	-1.176057	-0.553370	-2.391880	2.071994	-0.964715	...	-0.539321	1.094972	-2.254819	-0.634808	-0.926044	0.963765	0.145396	0.148652	-1.144346	-0.726426
1	-1.691044	1.404498	1	1.134494	1.351262	-1.176055	-0.545858	-2.717044	-0.023888	-0.989891	...	-2.792112	1.104265	-2.541900	-0.634800	-0.918552	-0.939099	0.145399	0.148665	-1.163816	-0.761551

	2	6		3	3	7		3	1	0		8		9	8	5	9	6	2	4
2	-1.691042	1.118210	1	1.105970	1.351263	-1.176057	-0.546877	-2.502268	-0.091294	-1.014905	...	-2.425075	1.081034	-2.350516	-0.634808	-0.913488	0.963765	0.145396	0.148652	-1.202757	-0.796602
3	-1.691042	0.971434	1	1.144601	1.351263	-1.176057	-0.530255	-2.175105	2.071994	-1.040000	...	-1.966258	1.262237	-2.063426	-0.634808	-0.903384	-0.423119	0.145396	0.148652	-1.248189	-0.831691
4	-1.69104	0.75465	1	1.17357	1.35126	-1.17605	-0.768242	-2.06671	2.07199	-1.06007	...	-1.87449	0.960231	-1.96773	-0.63480	-1.06372	-0.53312	0.14539	0.14865	-1.306601	-0.93695

	2	8		5	3	7		7	4	6		5		0	8	7	7	6	2	6

5 rows × 21 columns

Model Building

In [28]:

```
from sklearn.model_selection import train_test_split
from sklearn.metrics import r2_score, mean_squared_error
from sklearn.ensemble import RandomForestRegressor,
ExtraTreesRegressor, GradientBoostingRegressor
from xgboost import XGBRegressor
from sklearn.metrics import r2_score, mean_squared_error
```

In [29]:

```
x_train, x_test, y_train, y_test = train_test_split(x, y,
test_size = 0.2, random_state = 30)
```

In [30]:

```
print(f"Shape of X_train is: {x_train.shape}")
print(f"Shape of Y_train is: {y_train.shape}\n")
print(f"Shape of X_test is: {x_test.shape}")
print(f"Shape of Y_test is: {y_test.shape}")
```

```
Shape of X_train is: (2350, 21)
Shape of Y_train is: (2350,)

Shape of X_test is: (588, 21)
Shape of Y_test is: (588,)
```

In [31]:

```
models = {
    'Random Forest': RandomForestRegressor(random_state=42),
    'Extra Trees Regressor':
ExtraTreesRegressor(random_state=42),
    'GradientBoost Regressor':
GradientBoostingRegressor(random_state=42),
    'XGB Regressor': XGBRegressor()
}

# list to store results
results = []

# Train and evaluate each model
for model_name, model in models.items():
    # Train the model
```

```
    model.fit(x_train, y_train)

    # Make predictions
    y_pred = model.predict(x_test)

    # Calculate metrics
    rmse = mean_squared_error(y_test, y_pred, squared=False)
    r2 = r2_score(y_test, y_pred)

    # Store results in list
    results.append({'Model': model_name, 'RMSE': rmse, 'R2 Score': r2})

results_df = pd.DataFrame(results)
```

In [32]:

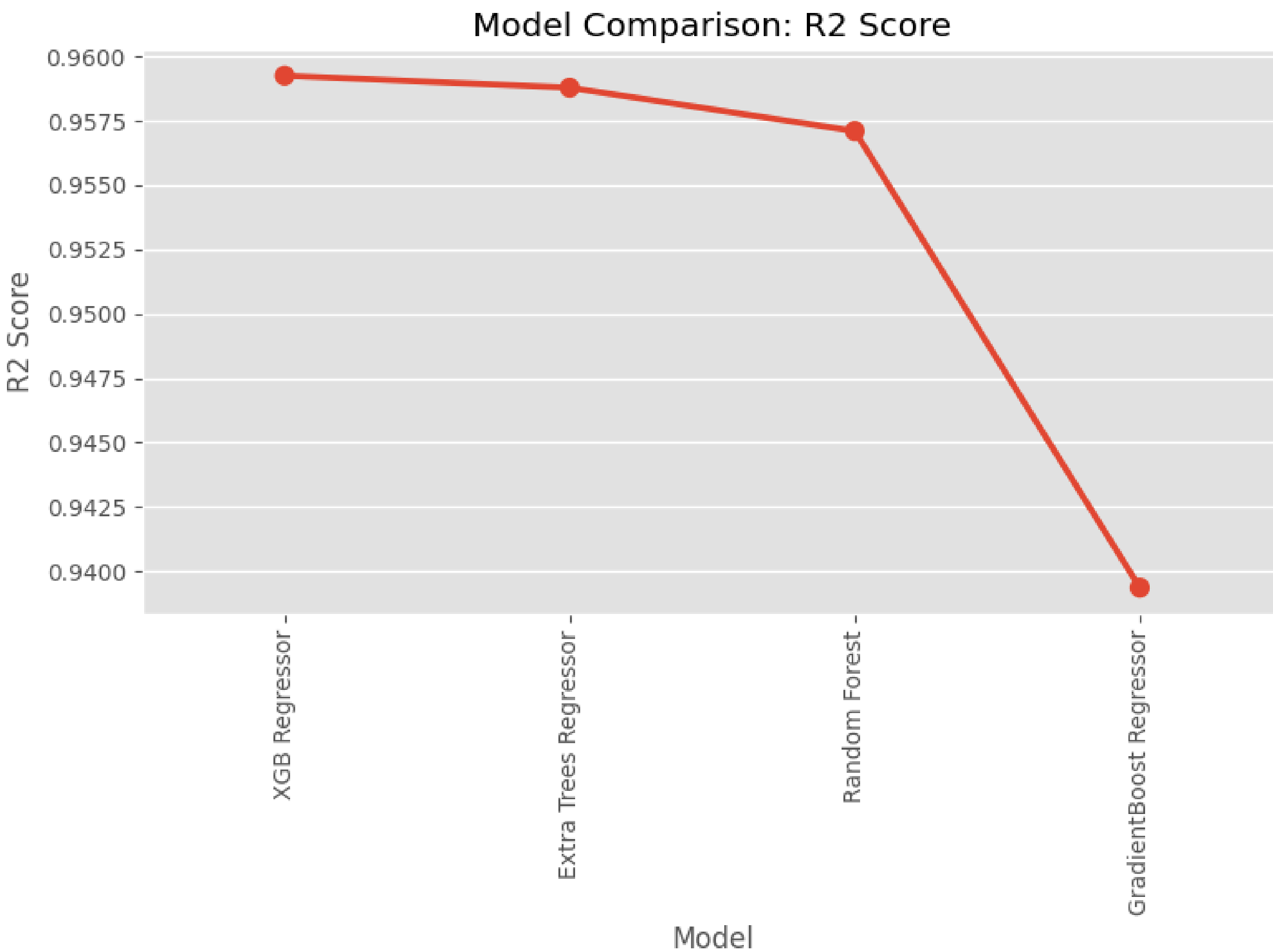
```
results_df=results_df.sort_values("R2 Score", ascending = False)
results_df
```

Out[32]:

	Model	RMSE	R2 Score
3	XGB Regressor	1.981816	0.959251
1	Extra Trees Regressor	1.993149	0.958783
0	Random Forest	2.033441	0.957100
2	GradientBoost Regressor	2.417593	0.939360

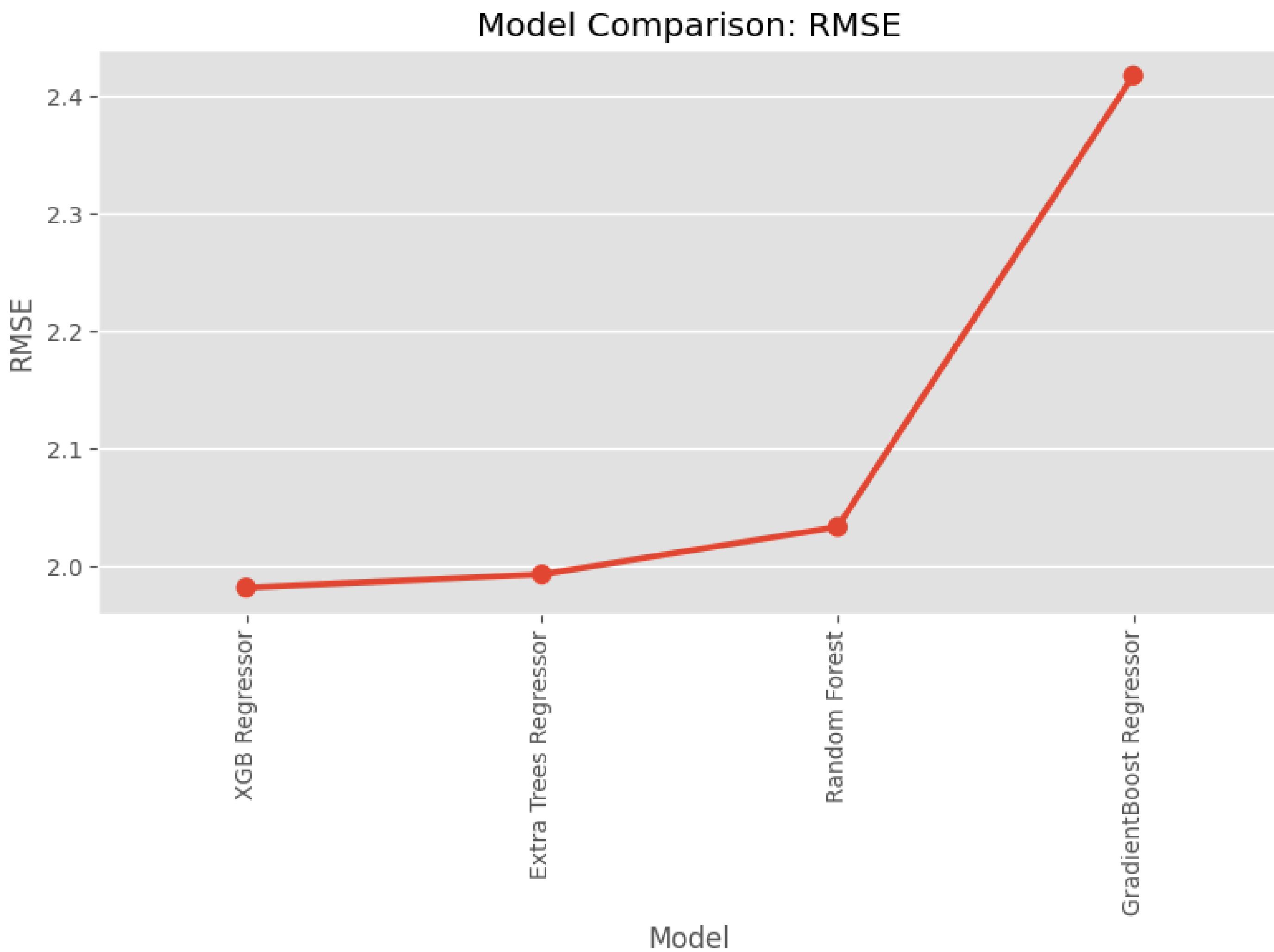
In [33]:

```
plt.figure(figsize=(8,6))
sns.pointplot(x='Model',y='R2 Score',data=results_df)
plt.xticks(rotation=90)
plt.title('Model Comparison: R2 Score')
plt.tight_layout()
plt.show()
```



In [34]:

```
plt.figure(figsize=(8,6))
sns.pointplot(x='Model',y='RMSE',data=results_df)
plt.xticks(rotation=90)
plt.title('Model Comparison: RMSE')
plt.tight_layout()
plt.show()
```



Cross Validate Final Model

In [35]:

```
from sklearn.model_selection import cross_val_score, KFold


best_model = XGBRegressor()


kf = KFold(n_splits=20, shuffle=True, random_state=42)
```

```
cross_val = cross_val_score(best_model, x, y, cv= kf,
scoring='r2')
cross_val
```

Out[35]:

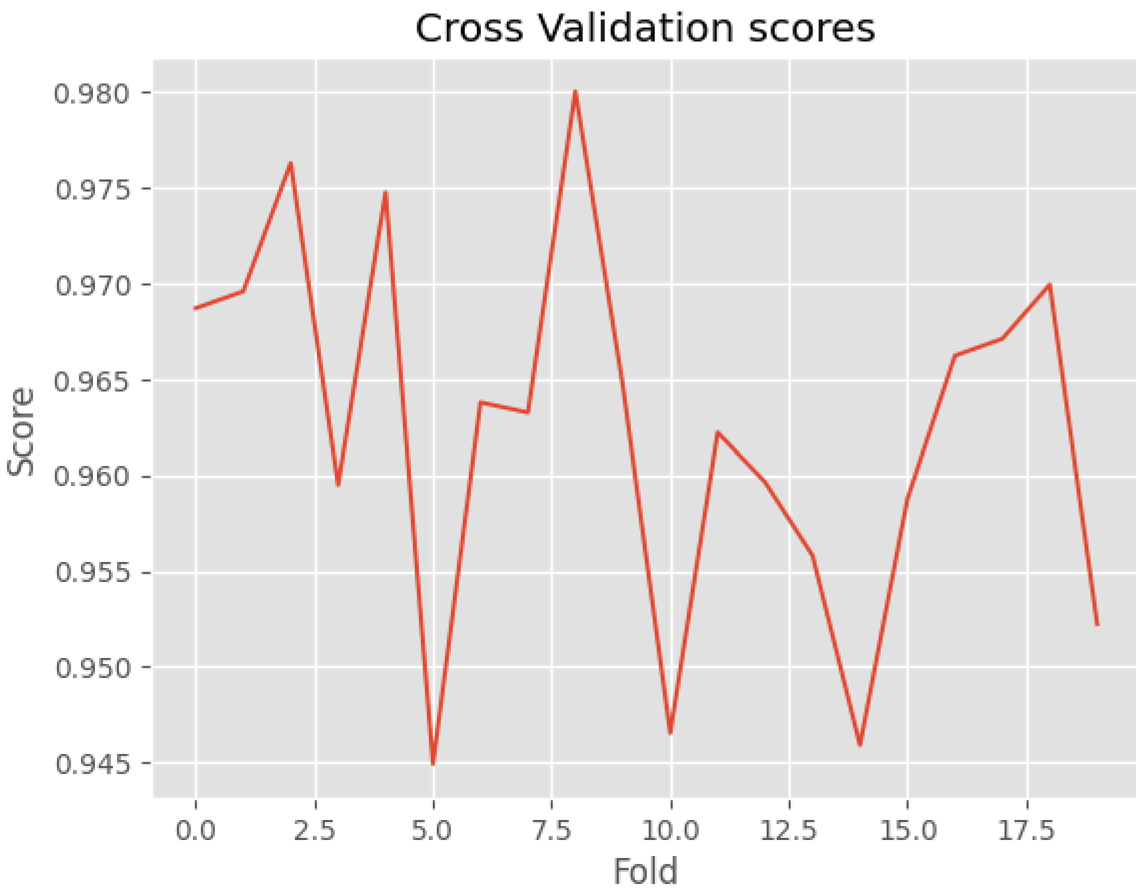
```
array([0.96870258, 0.96957899, 0.97628745, 0.95949212,
0.9747437 ,
       0.94493638, 0.96378644, 0.96325678, 0.98002501,
0.96468556,
       0.94654413, 0.96222682, 0.95963476, 0.95581229,
0.94591112,

       0.95876267, 0.96622263, 0.96710515, 0.96993108,
0.95223263])
```

In [36]:

```
plt.plot(cross_val)
plt.xlabel('Fold')
plt.ylabel('Score')
plt.title("Cross Validation scores")
```

Out[36]:

```
Text(0.5, 1.0, 'Cross Validation scores')
```



In [37]:

```
cross_val.mean()
```

Out[37]:

```
0.9624939137696394
```

In [38]:

```
cross_val.std()
```

```
Out[38]:

0.009624725292876094

In [ ]:
```

[Reference link]