Life Expectancy Prediction Project

1.Loading Libraries And Data Sets

In [1]:
```
import pandas as pd
import numpy as np
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.impute import SimpleImputer
from sklearn.preprocessing import StandardScaler, LabelEncoder
import warnings
warnings.filterwarnings('ignore')
```

In [2]:
```
df = pd.read_csv("E:\DATA ANALST Project Work\Life Expectancy Data.csv")
```

In [3]:
```
df.head()
```

Out[3]:

	Country	Year	Status	Life expectancy	Adult Mortality	infant deaths	Alcohol	percentage expenditure	Hepatitis B
0	Afghanistan	2015	Developing	65.0	263.0	62	0.01	71.279624	65.0
1	Afghanistan	2014	Developing	59.9	271.0	64	0.01	73.523582	62.0
2	Afghanistan	2013	Developing	59.9	268.0	66	0.01	73.219243	64.0
3	Afghanistan	2012	Developing	59.5	272.0	69	0.01	78.184215	67.0
4	Afghanistan	2011	Developing	59.2	275.0	71	0.01	7.097109	68.0

5 rows × 22 columns

2. Data Cleaning

In [4]:
```
print("missing values after imputation:")
print(df.isnull().sum())
```

```
missing values after imputation:
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 [5]:
```
Imputer = SimpleImputer(missing_values = np.nan, strategy = 'mean')
for col in df.columns:
    if df[col].isnull().sum() > 0:
        df[col] = Imputer.fit_transform(df[[col]])
```

In [6]:
```
total_missing = df.isnull().sum().sum()
print(f"Total missing values in the dataset: {total_missing}")
```

```
Total missing values in the dataset: 0
```

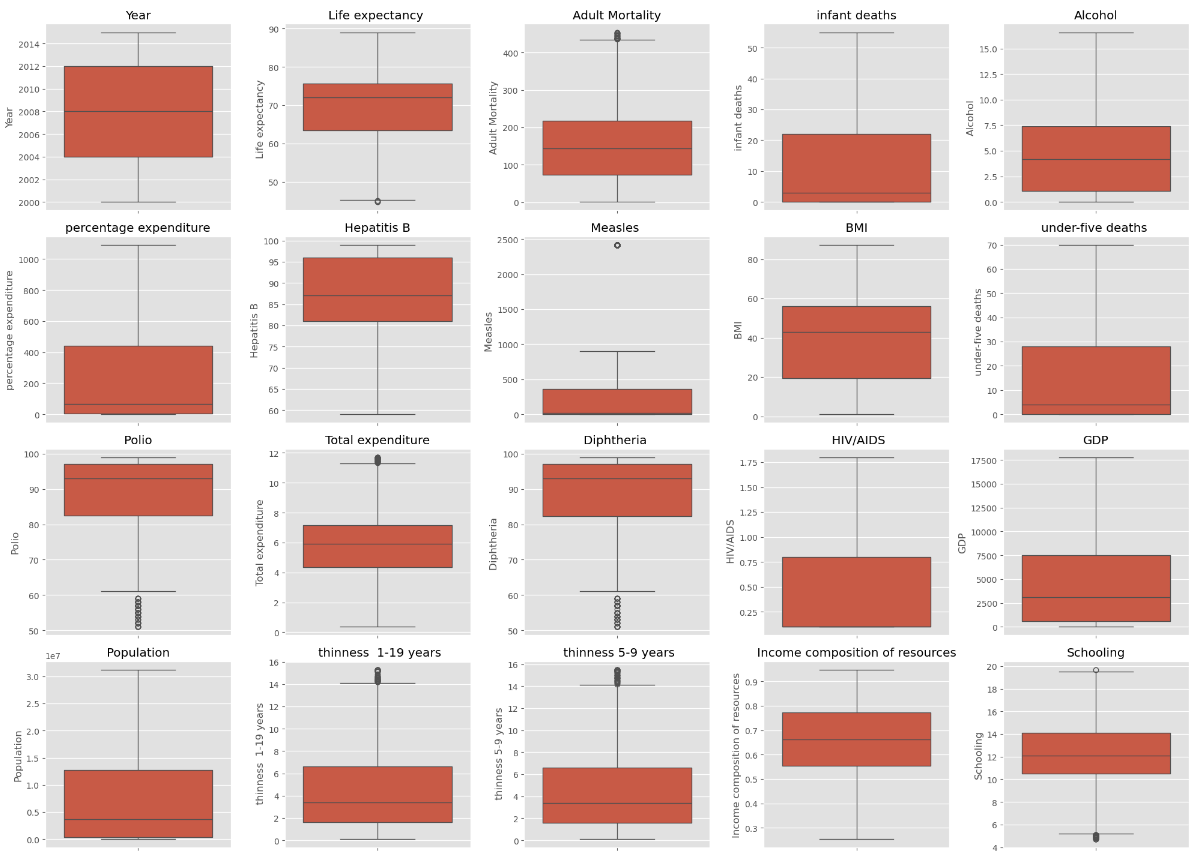
In [7]:
```
print(df.describe())
```

```
              Year  Life expectancy   Adult Mortality  infant deaths  \
count  2938.000000       2938.000000       2938.000000    2938.000000   
mean   2007.518720         69.224932        164.796448      30.303948   
std       4.613841          9.507640        124.080302     117.926501   
min    2000.000000         36.300000          1.000000       0.000000   
25%    2004.000000         63.200000         74.000000       0.000000   
50%    2008.000000         72.000000        144.000000       3.000000   
75%    2012.000000         75.600000        227.000000      22.000000   
max    2015.000000         89.000000        723.000000    1800.000000   

           Alcohol  percentage expenditure  Hepatitis B       Measles   \
count  2938.000000             2938.000000  2938.000000    2938.000000   
mean      4.602861              738.251295    80.940461    2419.592240   
std       3.916288             1987.914858    22.586855   11467.272489   
min       0.010000                0.000000     1.000000       0.000000   
25%       1.092500                4.685343    80.940461       0.000000   
50%       4.160000               64.912906    87.000000      17.000000   
75%       7.390000              441.534144    96.000000     360.250000   
max      17.870000            19479.911610    99.000000  212183.000000   

              BMI   under-five deaths         Polio  Total expenditure  \
count  2938.000000         2938.000000  2938.000000         2938.000000   
mean     38.321247           42.035739    82.550188            5.938190   
std      19.927677          160.445548    23.352143            2.400274   
min       1.000000            0.000000     3.000000            0.370000   
25%      19.400000            0.000000    78.000000            4.370000   
50%      43.000000            4.000000    93.000000            5.938190   
75%      56.100000           28.000000    97.000000            7.330000   
max      87.300000         2500.000000    99.000000           17.600000   

       Diphtheria      HIV/AIDS            GDP    Population  \
count  2938.000000  2938.000000    2938.000000  2.938000e+03   
mean     82.324084     1.742103    7483.158469  1.275338e+07   
std      23.640073     5.077785   13136.800417  5.381546e+07   
min       2.000000     0.100000       1.681350  3.400000e+01   
25%      78.000000     0.100000     580.486996  4.189172e+05   
50%      93.000000     0.100000    3116.561755  3.675929e+06   
75%      97.000000     0.800000    7483.158469  1.275338e+07   
max      99.000000    50.600000  119172.741800  1.293859e+09   

        thinness  1-19 years   thinness 5-9 years  \
count            2938.000000          2938.000000   
mean                4.839704             4.870317   
std                 4.394535             4.482708   
min                 0.100000             0.100000   
25%                 1.600000             1.600000   
50%                 3.400000             3.400000   
75%                 7.100000             7.200000   
max                27.700000            28.600000   

       Income composition of resources    Schooling  
count                      2938.000000  2938.000000  
mean                          0.627551    11.992793  
std                           0.204820     3.264381  
min                           0.000000     0.000000  
25%                           0.504250    10.300000  
50%                           0.662000    12.100000  
75%                           0.772000    14.100000  
max                           0.948000    20.700000  
```

3. Outlier Handling(IQR Method)

In [9]:
```
# Select numerical columns
numerical_cols = df.select_dtypes(include=['float64', 'int64']).columns

# Create subplots (4 rows × 5 columns = 20 plots)
fig, axes = plt.subplots(4, 5, figsize=(20, 16))
fig.suptitle('Boxplots of Numerical Columns', fontsize=16)

# Flatten the 2D axes array for easy indexing
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)

# Remove any unused subplots (if you have less than 20 columns)
for j in range(len(numerical_cols), len(axes)):
    fig.delaxes(axes[j])

# Apply layout and show the final figure (outside the loop!)
plt.tight_layout(rect=[0, 0.03, 1, 0.95])
plt.show()
```

Boxplots of Numerical Columns



In [10]:
```
#specify the list of columns you want to handle ouliers for
outlier_cols = ['Adult Mortality','infant deaths','Alcohol','percentage expenditure
                'BMI','under-five deaths','Polio','Total expenditure','Diphtheria',
                'Population','thinness  1-19 years','thinness 5-9 years','Income co
```

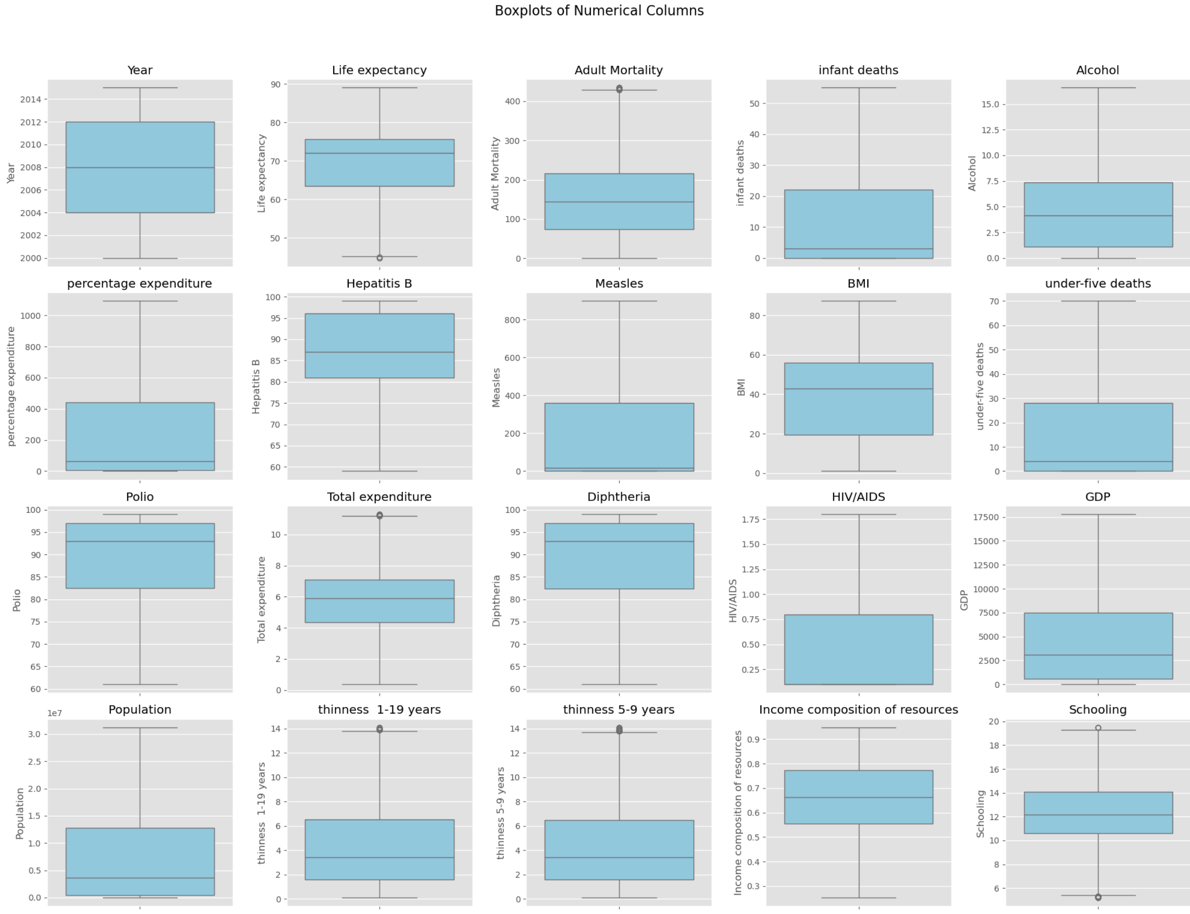
In [12]:
```
df.columns = df.columns.str.strip()  # remove spaces

# Keep only columns that exist
outlier_cols = [col for col in outlier_cols if col in df.columns]

for col_name in outlier_cols:
```

```
    q1 = df[col_name].quantile(0.25)
    q3 = df[col_name].quantile(0.75)
    iqr = q3 - q1
    lower_bound = q1 - 1.5 * iqr
    upper_bound = q3 + 1.5 * iqr

    df[col_name] = np.where(
        (df[col_name] > upper_bound) | (df[col_name] < lower_bound),
        df[col_name].mean(),
        df[col_name]
    )
```

In [13]:
```
df.shape
```

Out[13]:
```
(2938, 22)
```

In [ ]:
```
# Select only numerical columns (no .columns here!)
numerical_cols = df.select_dtypes(include=['float64', 'int64']).columns

# Create subplots
fig, axes = plt.subplots(4, 5, figsize=(20, 16))
fig.suptitle('Boxplots of Numerical Columns', fontsize=16)

# Flatten axes for easy access
axes = axes.flatten()

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

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

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



4. Exploratory Data Analysis

In [15]:
```
df.head()
```

Out[15]:

	Country	Year	Status	Life expectancy	Adult Mortality	infant deaths	Alcohol	percentage expenditure	Hepati
0	Afghanistan	2015.0	Developing	65.0	263.0	30.303948	0.01	71.279624	6
1	Afghanistan	2014.0	Developing	59.9	271.0	30.303948	0.01	73.523582	6
2	Afghanistan	2013.0	Developing	59.9	268.0	30.303948	0.01	73.219243	6
3	Afghanistan	2012.0	Developing	59.5	272.0	30.303948	0.01	78.184215	6
4	Afghanistan	2011.0	Developing	59.2	275.0	30.303948	0.01	7.097109	6

5 rows × 22 columns

In [16]:
```
df.Country.value_counts()
```

Out[16]:
```
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
```

Life Expectancy Vs Year

In [ ]:
```python
import plotly.express as px

# Strip column names to remove any trailing spaces
df.columns = df.columns.str.strip()

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

# Create the interactive line plot using Plotly
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()
```

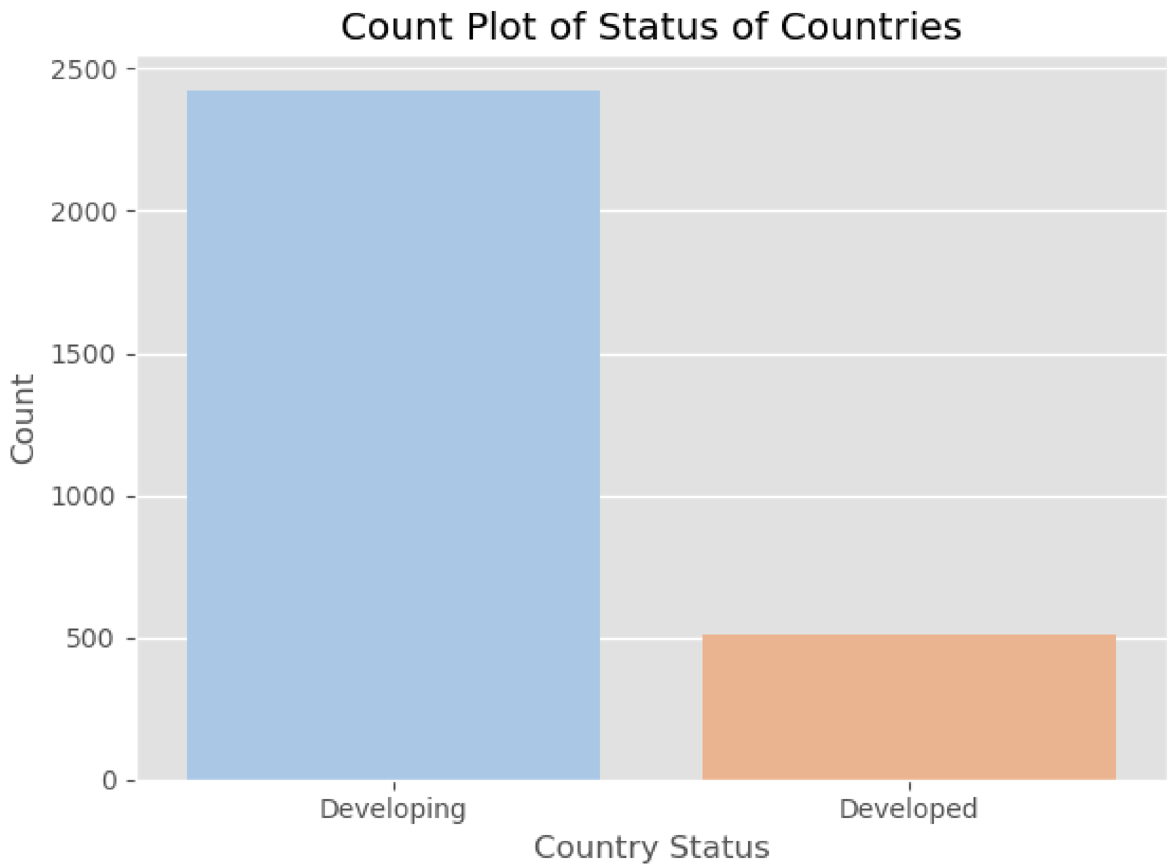
Population vs Life Expectancy

In [ ]:
```python
import plotly.express as px

# Strip column names to avoid issues with trailing spaces
df.columns = df.columns.str.strip()

# Create the interactive animated scatter plot
fig = px.scatter(
    df,
    x='Population',
    y='Life expectancy',
    hover_name='Country',
    color='Status',
    animation_frame='Year',
    title='Population vs Life Expectancy Over the Years',
    labels={
        'Population': 'Population',
        'Life expectancy': 'Life Expectancy (years)'
    },
    template='plotly_dark',
    size_max=60
)
```

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

Country Status Count

In [ ]:
```
import seaborn as sns
import matplotlib.pyplot as plt

# Strip column names (important if 'Status' has extra spaces)
df.columns = df.columns.str.strip()

# Create a count plot for 'Status'
sns.countplot(x=df['Status'], palette='pastel')

# Add title and layout formatting
plt.title('Count Plot of Status of Countries')
plt.xlabel('Country Status')
plt.ylabel('Count')

# Show plot
plt.tight_layout()
plt.show()
```



Average Life Expectancy Vs Country Status

In [ ]:
```
# Histogram of Average Life Expectancy by Country Status
import plotly.express as px

# Clean column names to remove trailing spaces
df.columns = df.columns.str.strip()
```

```
# Group by 'Status' and calculate mean life expectancy
life_expect_status = df.groupby('Status')['Life expectancy'].mean().reset_index()

# Create histogram using Plotly
fig = px.histogram(
    life_expect_status,
    x='Status',
    y='Life expectancy',
    color='Status',
    text_auto=True
)

# Update layout and title
fig.update_layout(
    title=dict(
        text='<b>Average Life Expectancy for Country Status</b>',
        x=0.5,
        font=dict(size=18)
    ),
    yaxis_title='Life Expectancy (years)',
    xaxis_title='Country Status',
    template='plotly_dark'
)

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

Life Expectancy VS Alcohol Consumption

In [ ]:

```
# Life Expectancy Vs Alcohol Consumption(Dual Y-axis)
import plotly.graph_objects as go

# Clean column names
df.columns = df.columns.str.strip()

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

# Create a figure with dual Y-axes
fig = go.Figure()

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

# Alcohol Consumption trace (right Y-axis)
fig.add_trace(go.Scatter(
    x=average_data['Year'],
    y=average_data['Alcohol'],
    mode='lines+markers',
    name='Alcohol Consumption',
    yaxis='y2'
))
```

```
# Update layout to use 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()
```

Alcohol Consumption VS Country Status

In [ ]:

```
# Average Alcohol Consumption by Country Status
import plotly.express as px

# Clean column names
df.columns = df.columns.str.strip()

# Group by 'Status' and calculate mean Alcohol consumption
alcohol_by_status = df.groupby('Status', as_index=False).agg({'Alcohol': 'mean'})

# Create bar chart using Plotly
fig = px.bar(
    alcohol_by_status,
    x='Status',
    y='Alcohol',
    title='Average Alcohol Consumption of Developing and Developed Countries',
    labels={
        'Alcohol': 'Alcohol Consumption (liters per capita)',
        'Status': 'Country Status'
    },
    template='plotly_dark',
    text_auto=True  # Adds value labels on bars
)

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

Life Expectancy VS Year of Schooling

In [ ]:

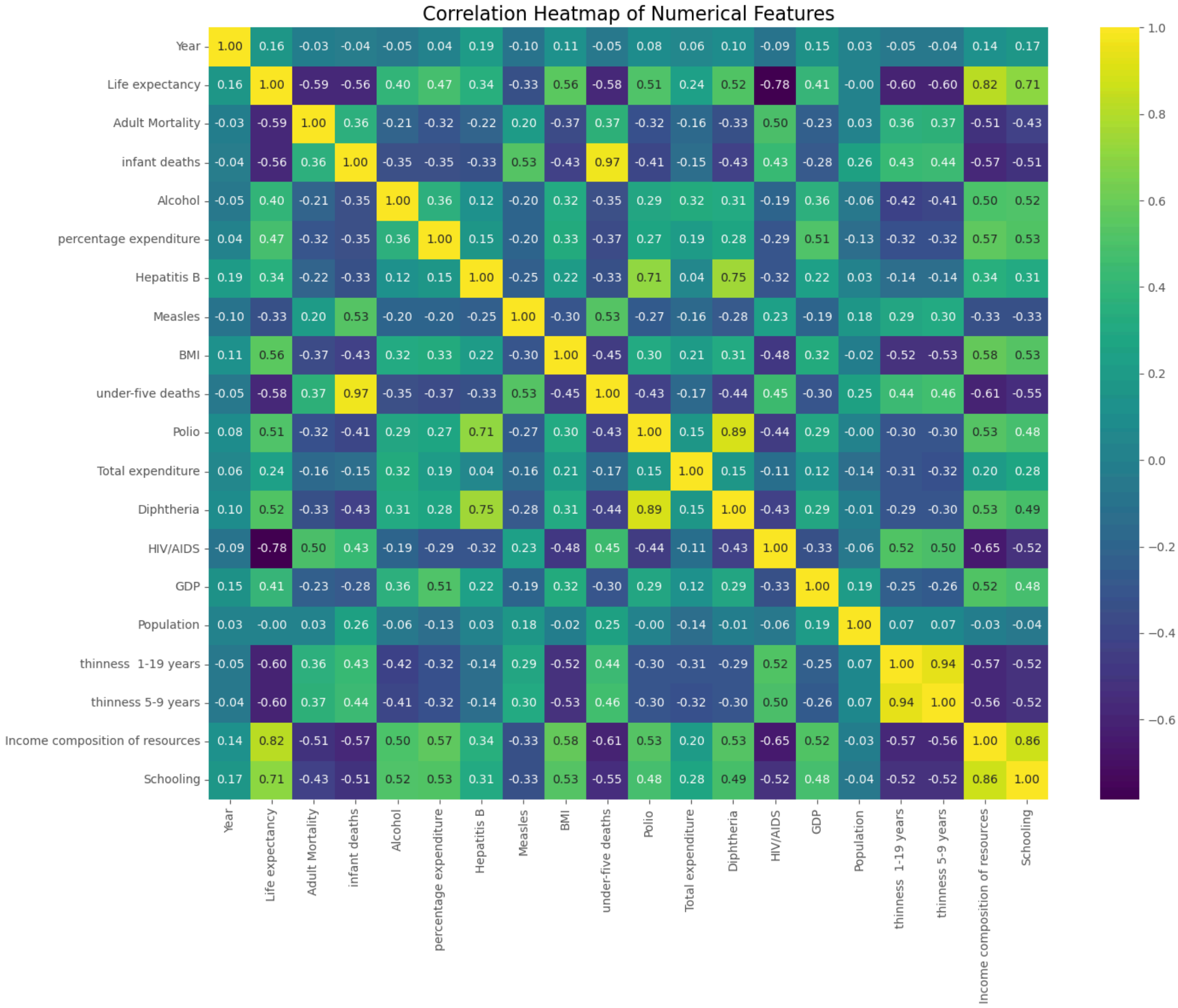
```
# Life Expectancy vs Years of schooling(interactive line plot)
import plotly.express as px

# Clean column names
df.columns = df.columns.str.strip()

# Group by 'Schooling' and calculate average life expectancy
aggregated_data = df.groupby('Schooling')['Life expectancy'].mean().reset_index()
```

```
# Create 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',
    markers=True
)

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

Correltion HeatMap

In [ ]:

```
# Correlation Heatmap of Numerical Columns
import matplotlib.pyplot as plt
import seaborn as sns

# Clean column names
df.columns = df.columns.str.strip()

# Select numerical columns
numerical_cols = df.select_dtypes(include=['float64', 'int64']).columns

# Create correlation heatmap
plt.figure(figsize=(15, 12))
sns.heatmap(
    df[numerical_cols].corr(),
    cmap='viridis',
    annot=True,
    fmt=".2f"
)
plt.title('Correlation Heatmap of Numerical Features', fontsize=16)
plt.tight_layout()
plt.show()
```



5. Data Preprocessing

In [ ]:
```
from sklearn.preprocessing import LabelEncoder

# Initialize the label encoder
le = LabelEncoder()

# Identify categorical columns
cat_cols = df.select_dtypes(include='object').columns

# Apply label encoding to each categorical column
for col in cat_cols:
    df[col] = le.fit_transform(df[col])
```

In [26]:
```
x= df.drop(columns='Life expectancy')
y=df['Life expectancy']
```

In [27]:
```
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 [28]:
```
x.head()
```

Out[28]:

	Country	Year	Status	Adult Mortality	infant deaths	Alcohol	percentage expenditure	Hepatitis B	Measles
0	-1.691042	1.621762	1	1.112066	1.351263	-1.176057	-0.553370	-2.391880	1.536427
1	-1.691042	1.404986	1	1.191723	1.351263	-1.176057	-0.545858	-2.717043	1.440436
2	-1.691042	1.188210	1	1.161852	1.351263	-1.176057	-0.546877	-2.500268	1.169468
3	-1.691042	0.971434	1	1.201680	1.351263	-1.176057	-0.530255	-2.175105	1.536427
4	-1.691042	0.754658	1	1.231551	1.351263	-1.176057	-0.768242	-2.066717	1.536427

5 rows × 21 columns

6. Model Building And Evaluation

In [29]:
```
from sklearn.model_selection import train_test_split
from sklearn.metrics import r2_score,mean_squared_error
from sklearn.ensemble import RandomForestRegressor,ExtraTreesRegressor,GradientBoos
from xgboost import XGBRegressor
from sklearn.metrics import r2_score,mean_squared_error
```

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

In [31]:
```
print(f"ShapeofX_trainis:{x_train.shape}")
print(f"ShapeofY_trainis:{y_train.shape}\n")
print(f"ShapeofX_testis:{x_test.shape}")
print(f"ShapeofY_testis:{y_test.shape}")
```

```
ShapeofX_trainis:(2350, 21)
ShapeofY_trainis:(2350,)

ShapeofX_testis:(588, 21)
ShapeofY_testis:(588,)
```

In [32]:
```
models = {
 'Random Forest': RandomForestRegressor(random_state=42),
 'Extra Trees Regressor':
 ExtraTreesRegressor(random_state=42),
 'GradientBoost Regressor':
 GradientBoostingRegressor(random_state=42),
 'XGB Regressor': XGBRegressor()
 }
```

In [33]:
```
result = []
```

In [34]:
```
# Loop through all models
for model_name, model in models.items():
    # Train
    model.fit(x_train, y_train)

    # Predict
    y_pred = model.predict(x_test)

    # Evaluate
    rmse = np.sqrt(mean_squared_error(y_test, y_pred))
    r2 = r2_score(y_test, y_pred)
```

```
    # Store
    result.append({'Model': model_name, 'RMSE': rmse, 'R2 Score': r2})

# Convert to DataFrame
results_df = pd.DataFrame(result)

# Display
print(results_df)
```

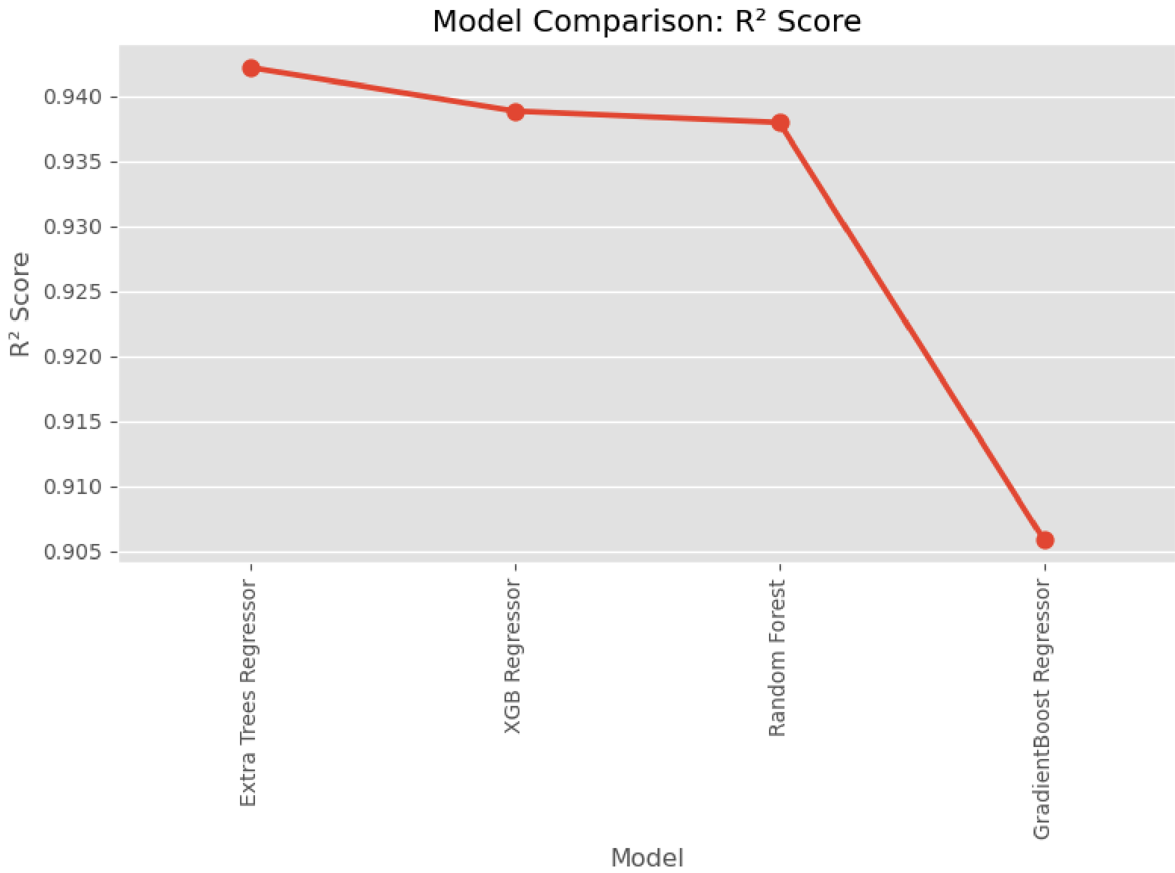
```
                    Model      RMSE  R2 Score
0           Random Forest  2.376430  0.937948
1   Extra Trees Regressor  2.294144  0.942171
2  GradientBoost Regressor  2.925963  0.905932
3           XGB Regressor  2.359460  0.938831
```

In [35]:
```
results_df=results_df.sort_values("R2 Score", ascending = False)
results_df
```

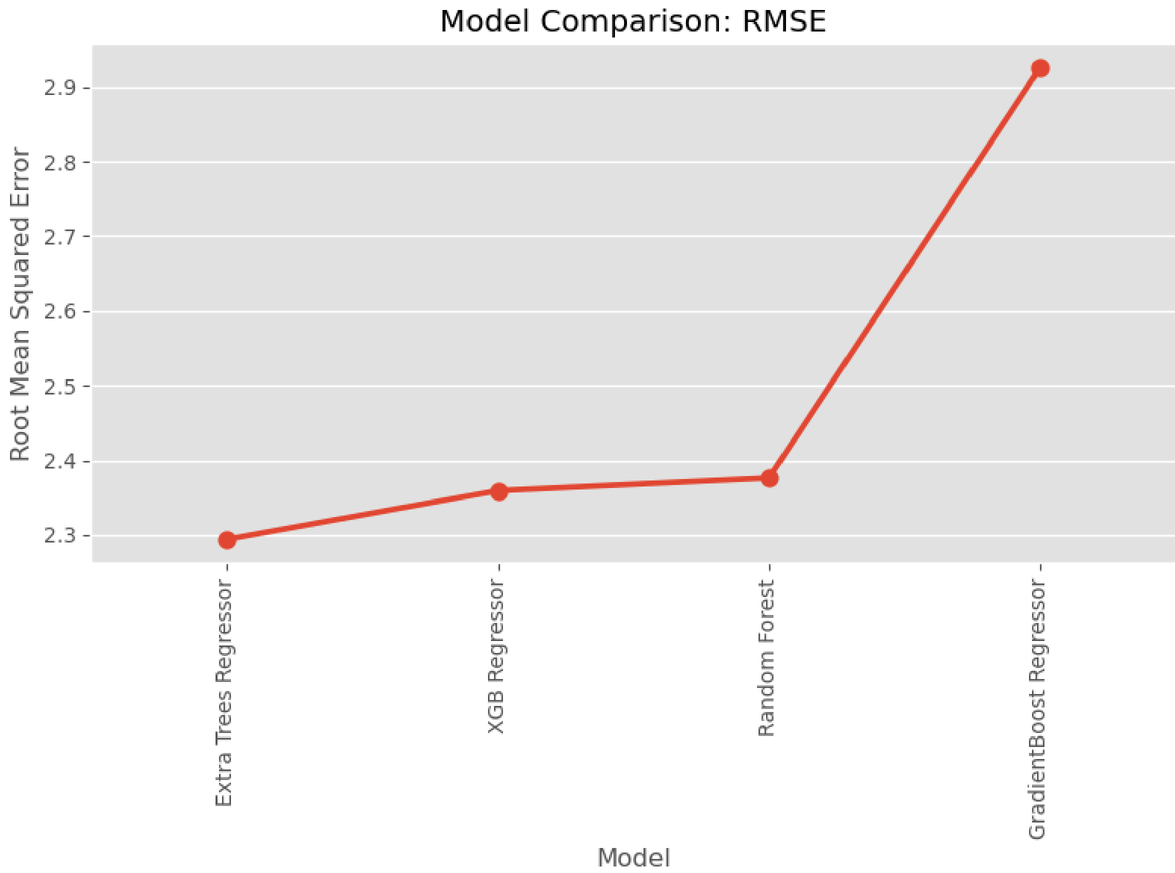
Out[35]:

	Model	RMSE	R2 Score
1	Extra Trees Regressor	2.294144	0.942171
3	XGB Regressor	2.359460	0.938831
0	Random Forest	2.376430	0.937948
2	GradientBoost Regressor	2.925963	0.905932

In [ ]:
```
import matplotlib.pyplot as plt
import seaborn as sns

# Fix spacing in column name (if needed)
results_df.columns = results_df.columns.str.strip()

# Rename column if it has a space (optional)
# If your column is actually named 'R2 Score', rename it for consistency
results_df = results_df.rename(columns={'R2 Score': 'R2Score'})

# Create the point plot
plt.figure(figsize=(8, 6))
sns.pointplot(x='Model', y='R2Score', data=results_df)
plt.xticks(rotation=90)
plt.title('Model Comparison: R² Score')
plt.ylabel('R² Score')
plt.xlabel('Model')
plt.tight_layout()
plt.show()
```



In [ ]:
```
import matplotlib.pyplot as plt
import seaborn as sns

# Ensure clean column names
results_df.columns = results_df.columns.str.strip()

# Create the point plot for RMSE
plt.figure(figsize=(8, 6))
sns.pointplot(x='Model', y='RMSE', data=results_df)
plt.xticks(rotation=90)
plt.title('Model Comparison: RMSE')
plt.ylabel('Root Mean Squared Error')
plt.xlabel('Model')
plt.tight_layout()
plt.show()
```



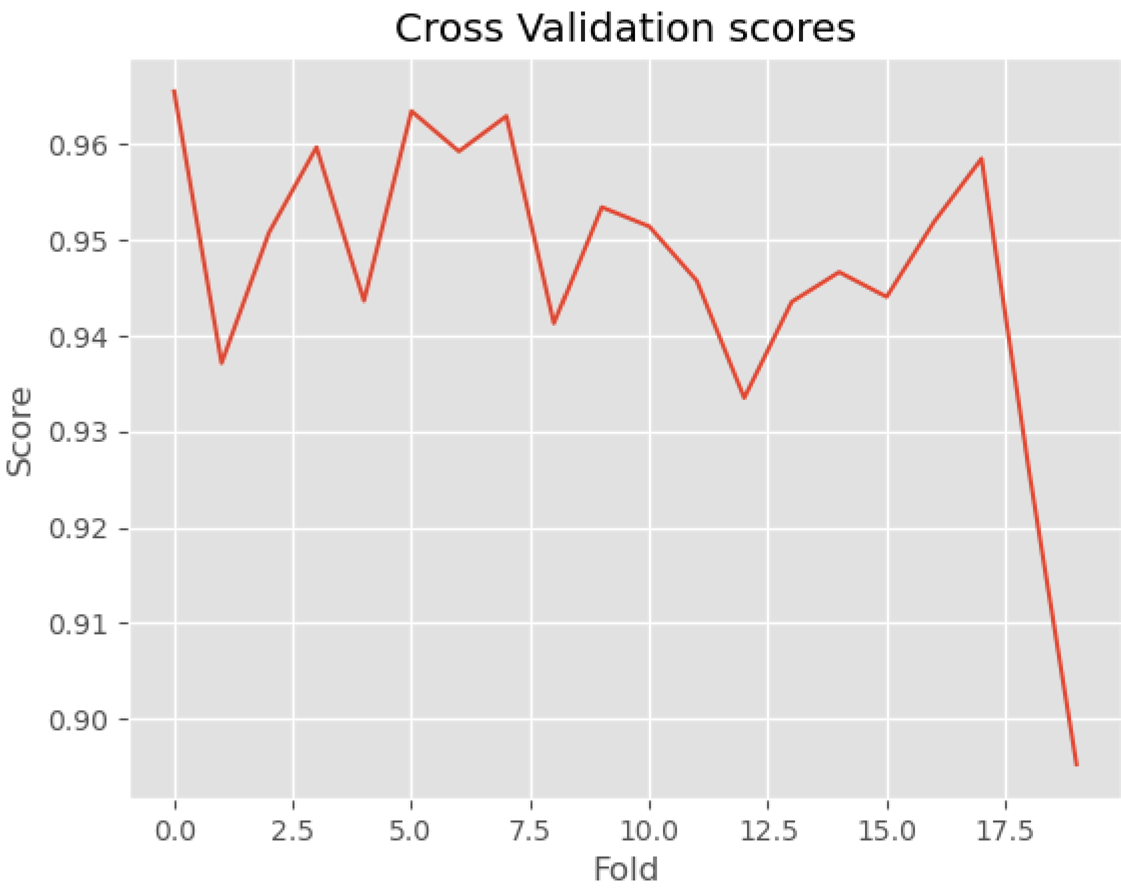
7. Cross Validation Final Model

In [ ]:
```
from sklearn.model_selection import cross_val_score, KFold
from xgboost import XGBRegressor

# Define model
best_model = XGBRegressor()

# Create 20-fold cross-validation setup
kf = KFold(n_splits=20, shuffle=True, random_state=42)

# Perform cross-validation
cross_val_scores = cross_val_score(
    best_model,
    x,
    y,
    cv=kf,
    scoring='r2'
)

# Display results
print("Cross-validation R² scores:")
print(cross_val_scores)
print("\nAverage R² score:", round(cross_val_scores.mean(), 4))
print("Standard Deviation:", round(cross_val_scores.std(), 4))
```

```
Cross-validation R² scores:
[0.96551357 0.93711351 0.95077552 0.95967247 0.94363115 0.9634485
 0.95924749 0.96292845 0.94127479 0.95340885 0.95139463 0.94572323
 0.93347476 0.94351532 0.94662614 0.9440392  0.95187541 0.95849145
 0.92589777 0.8952312 ]

Average R² score: 0.9467
Standard Deviation: 0.0156
```

In [39]:
```
plt.plot(cross_val_scores)
plt.xlabel('Fold')
plt.ylabel('Score')
plt.title("Cross Validation scores")
```

Out[39]:
```
Text(0.5, 1.0, 'Cross Validation scores')
```



In [40]:
```
cross_val_scores.mean()
```

Out[40]:
```
0.9466641711313615
```

In [41]:
```
cross_val_scores.std()
```

Out[41]:
```
0.015618282444082386
```

Objectives

Q. Do various predicting factors which have been chosen initially really affect the Life expectancy? What are the predicting variables actually affecting life expectancy?

Ans - Life expectancy is not random — it is strongly affected by education, income, healthcare access, disease prevalence, and lifestyle factors. Countries with low adult mortality, high schooling, higher GDP, good healthcare, and low infectious disease rates tend to have higher life expectancy.

Q. Should a country having a lower life expectancy value(<65) increase its healthcare expenditure in order to improve its average lifespan?

Ans - Yes, increasing healthcare expenditure can improve life expectancy in countries with low average lifespan, but it should be targeted and well-managed to address the main health issues in the country.

Q. How does Infant and Adult mortality rates affect life expectancy?

Ans - Both infant and adult mortality rates have a negative relationship with life expectancy. Higher mortality rates (more deaths) → lower life expectancy. Lower mortality rates (fewer deaths) → higher life expectancy.

Q. Does Life Expectancy has positive or negative correlation with eating habits,lifestyle, exercise, smoking, drinking alcohol etc.

Ans- Healthy lifestyle choices (good diet, regular exercise, avoiding smoking and excess drinking) are positively linked to life expectancy, while unhealthy habits have a negative effect and shorten lifespan.

Q. What is the impact of schooling on the lifespan of humans?

Ans - Schooling has a positive correlation with life expectancy. More education = more knowledge, better income, healthier lifestyle → people live longer.

Q. Does Life Expectancy have positive or negative relationship with drinking alcohol?

Ans - In many countries, very high alcohol consumption is linked to lower life expectancy. Countries with low to moderate alcohol use often have higher life expectancy, but that’s also because they have better healthcare, education, and living conditions. Overall, life expectancy has a negative relationship with heavy alcohol consumption. The more people drink excessively, the shorter their average lifespan tends to be.

Q. Do densely populated countries tend to have lower life expectancy?

Ans - Densely populated countries can have lower life expectancy if resources and healthcare are poor. But in developed nations with strong infrastructure, high density does not always reduce lifespan.

Q. What is the impact of Immunization coverage on life Expectancy?

Ans - Immunization coverage has a positive relationship with life expectancy. The more people are vaccinated, the longer the population tends to live.