DE_IMF

October 21, 2024

DE MINI PROJECT

Model used : Linear Regression

Dataset : International Monetary Fund

By : Hrishikesh Iyer

1 Necessary Libraries

[3]:
```
# Importing necessary libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
```

2 Data Pre-processing

2.1 Importing Data

[5]:
```
df = pd.read_csv('imf.csv')
```

[6]:
```
df.head()
```

[6]:
```
   Year  Population (Millions of people)  Inflatation Rate  \
0  1980                          696.828              11.3
1  1981                          712.869              12.7
2  1982                          729.169               7.7
3  1983                          745.827              12.6
4  1984                          762.895               6.5

   GDP based on PPP, share of world (Percent of World)  GDP Per Capita  HDI  \
0                                              3.022           267.167  NaN
1                                              3.137           270.951  NaN
2                                              3.234           274.332  NaN
3                                              3.388           292.585  NaN
```

```
4                                                3.368           277.683  NaN

   Life expectancy - Women  Life expectancy - Men  Life expectancy
0                    53.70                  53.55            53.61
1                    54.40                  53.99            54.18
2                    55.07                  54.45            54.73
3                    55.71                  54.91            55.28
4                    56.32                  55.39            55.82
```

[7]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 50 entries, 0 to 49
Data columns (total 9 columns):
 #   Column                                           Non-Null Count  Dtype
---  ------                                           --------------  -----
 0   Year                                             50 non-null     int64
 1   Population (Millions of people)                  50 non-null
float64
 2   Inflatation Rate                                 50 non-null
float64
 3   GDP based on PPP, share of world (Percent of World)  50 non-null
float64
 4   GDP Per Capita                                   50 non-null
float64
 5   HDI                                              32 non-null
float64
 6   Life expectancy - Women                          43 non-null
float64
 7   Life expectancy - Men                            43 non-null
float64
 8   Life expectancy                                  43 non-null
float64
dtypes: float64(8), int64(1)
memory usage: 3.6 KB
```

2.2 Handling Missing Values and Duplicates

[9]:
```
# Handling missing values
df.isnull().sum()  # Check for missing values in each column
df.dropna(inplace=True)  # Remove rows with missing values

# Removing duplicates
df.duplicated().sum()  # Check for duplicate rows
df.drop_duplicates(inplace=True)  # Remove duplicate rows
```

3 Linear Regression on Various Attributes

3.1 Year *vs* Population

[11]:
```
# Selecting Year as the independent variable and Population as the dependent␣
↪variable
X = df[['Year']]  # Independent variable (Year)
y = df['Population (Millions of people)']  # Dependent variable (Population)
```

[12]:
```
# Initialize the Linear Regression model
model = LinearRegression()

# Fit the model to the data
model.fit(X, y)
```

[12]:
```
LinearRegression()
```

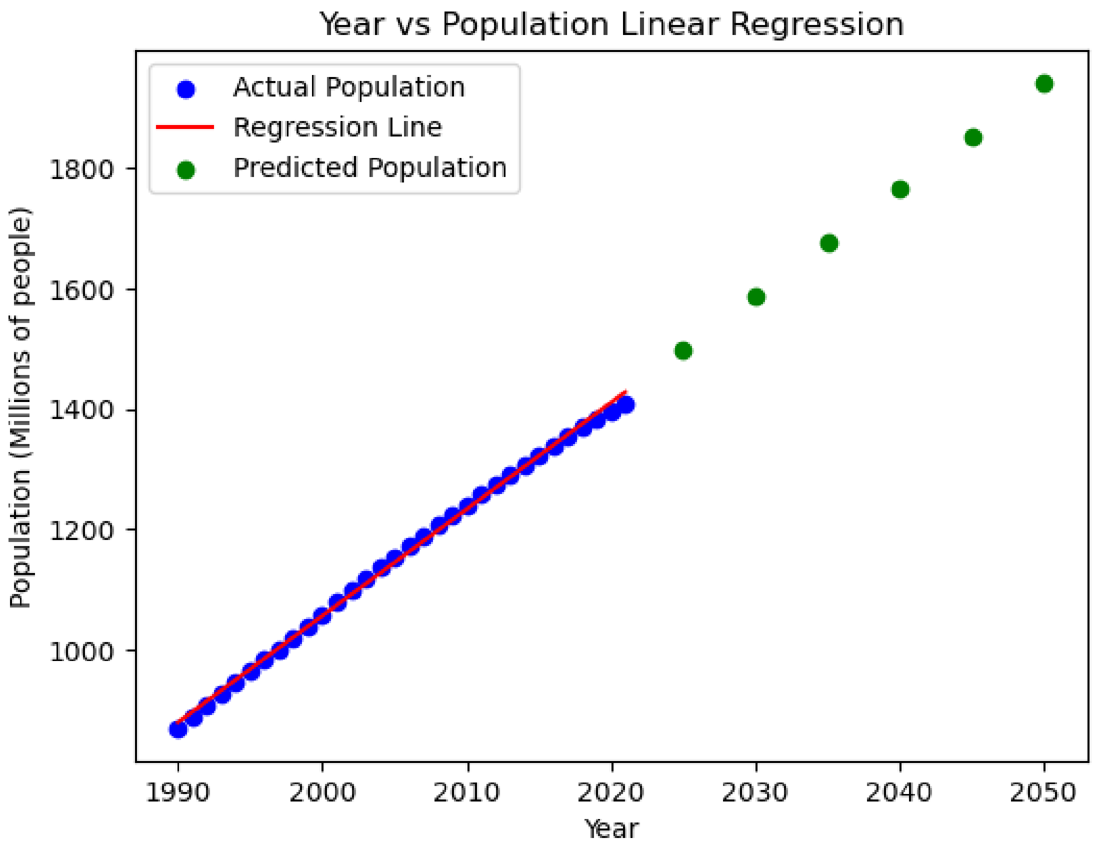
[13]:
```
# Create a numpy array of future years for prediction
future_years = np.array([[2025], [2030], [2035], [2040], [2045], [2050]])

# Predict population for the future years
future_population = model.predict(future_years)

# Output the predictions
for year, pop in zip(future_years, future_population):
    print(f"Year: {year[0]}, Predicted Population: {pop:.2f} million")
```

```
Year: 2025, Predicted Population: 1498.62 million
Year: 2030, Predicted Population: 1586.96 million
Year: 2035, Predicted Population: 1675.30 million
Year: 2040, Predicted Population: 1763.64 million
Year: 2045, Predicted Population: 1851.98 million
Year: 2050, Predicted Population: 1940.32 million
```

```
/opt/anaconda3/lib/python3.12/site-packages/sklearn/base.py:493: UserWarning: X
does not have valid feature names, but LinearRegression was fitted with feature
names
  warnings.warn(
```

[14]:
```
# Plot the actual data points
plt.scatter(X, y, color='blue', label='Actual Population')

# Plot the regression line based on the model
plt.plot(X, model.predict(X), color='red', label='Regression Line')

# Plot the predicted future values
plt.scatter(future_years, future_population, color='green', label='Predicted␣
↪Population')
```

```
# Add labels, title, and legend
plt.xlabel('Year')
plt.ylabel('Population (Millions of people)')
plt.title('Year vs Population Linear Regression')
plt.legend()

# Display the plot
plt.show()
```



3.2 Year *vs* GDP Per Capita

[16]:
```
# Select Year as the independent variable and GDP Per Capita as the dependent␣
↪variable
X = df[['Year']]  # Independent variable (Year)
y = df['GDP Per Capita']  # Dependent variable (GDP Per Capita)
```

[17]:
```
# Initialize the Linear Regression model
model = LinearRegression()
```

```
# Fit the model to the data
model.fit(X, y)
```

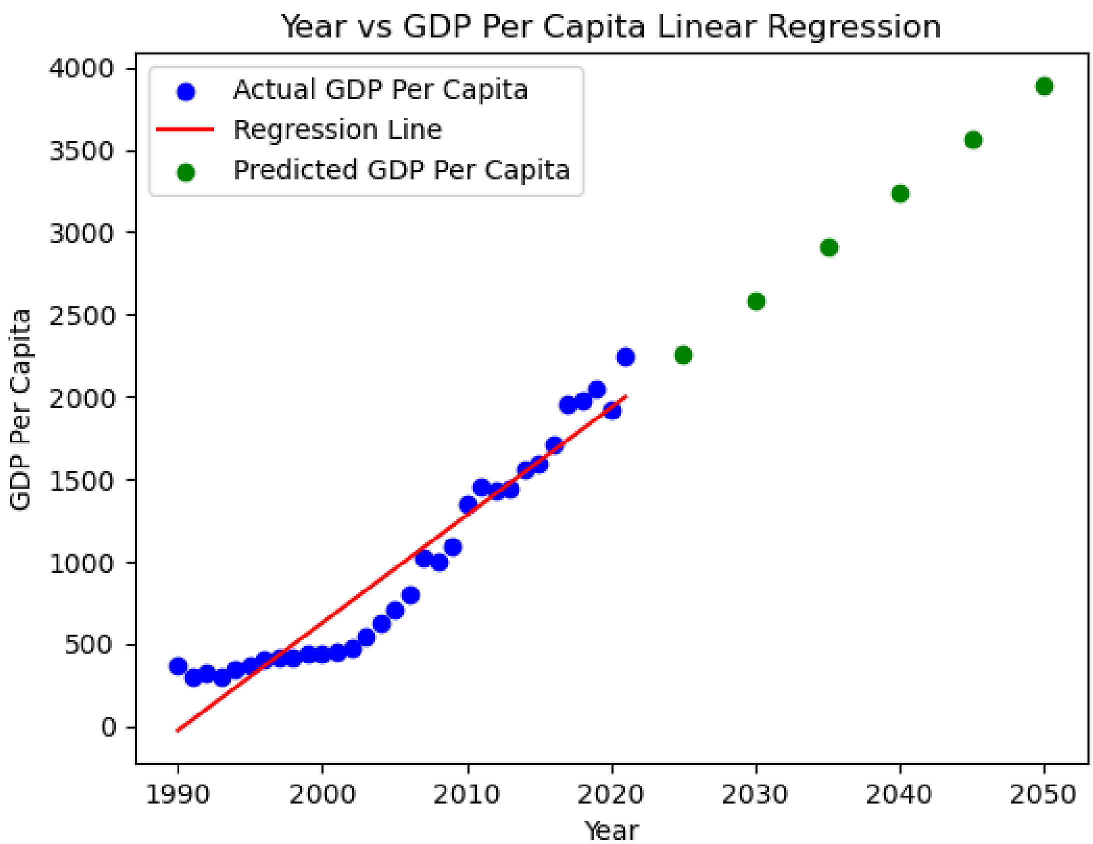
[17]: LinearRegression()

[18]:
```
# Create a numpy array of future years for prediction
future_years = np.array([[2025], [2030], [2035], [2040], [2045], [2050]])

# Predict GDP Per Capita for the future years
future_gdp_per_capita = model.predict(future_years)

# Output the predictions
for year, gdp in zip(future_years, future_gdp_per_capita):
    print(f"Year: {year[0]}, Predicted GDP Per Capita: {gdp:.2f}")
```

```
Year: 2025, Predicted GDP Per Capita: 2258.70
Year: 2030, Predicted GDP Per Capita: 2585.30
Year: 2035, Predicted GDP Per Capita: 2911.90
Year: 2040, Predicted GDP Per Capita: 3238.50
Year: 2045, Predicted GDP Per Capita: 3565.10
Year: 2050, Predicted GDP Per Capita: 3891.71
```

```
/opt/anaconda3/lib/python3.12/site-packages/sklearn/base.py:493: UserWarning: X
does not have valid feature names, but LinearRegression was fitted with feature
names
  warnings.warn(
```

[19]:
```
# Plot the actual data points
plt.scatter(X, y, color='blue', label='Actual GDP Per Capita')

# Plot the regression line based on the model
plt.plot(X, model.predict(X), color='red', label='Regression Line')

# Plot the predicted future values
plt.scatter(future_years, future_gdp_per_capita, color='green',␣
 ↪label='Predicted GDP Per Capita')

# Add labels, title, and legend
plt.xlabel('Year')
plt.ylabel('GDP Per Capita')
plt.title('Year vs GDP Per Capita Linear Regression')
plt.legend()

# Display the plot
plt.show()
```



3.3 Year *vs* HDI

[21]:
```
# Select Year as the independent variable and HDI as the dependent variable
X = df[['Year']]  # Independent variable (Year)
y = df['HDI']  # Dependent variable (HDI)
```

[22]:
```
# Drop rows where HDI is missing
df = df.dropna(subset=['HDI'])

# Redefine X and y after removing missing values
X = df[['Year']]  # Independent variable (Year)
y = df['HDI']  # Dependent variable (HDI)
```

[23]:
```
# Initialize the Linear Regression model
model = LinearRegression()

# Fit the model to the data
model.fit(X, y)
```

[23]: LinearRegression()

[24]:

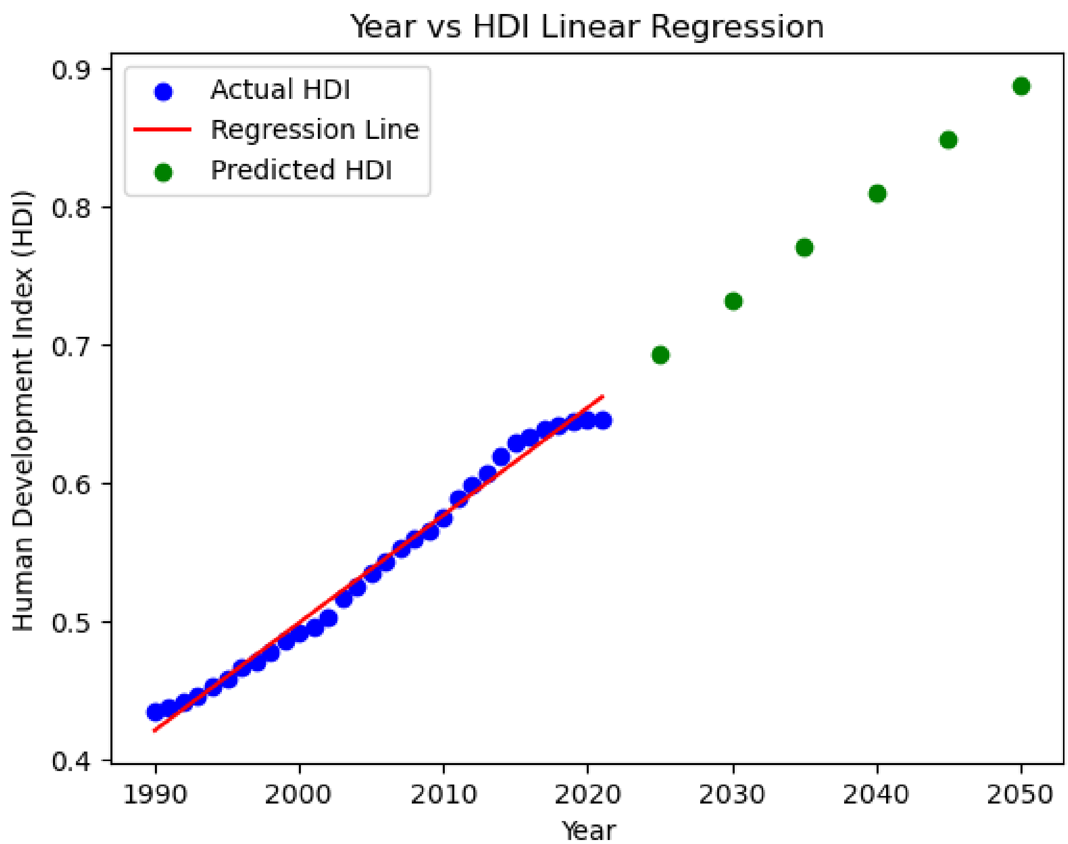
```
# Create a numpy array of future years for prediction
future_years = np.array([[2025], [2030], [2035], [2040], [2045], [2050]])

# Predict HDI for the future years
future_hdi = model.predict(future_years)

# Output the predictions
for year, hdi in zip(future_years, future_hdi):
    print(f"Year: {year[0]}, Predicted HDI: {hdi:.3f}")
```

```
Year: 2025, Predicted HDI: 0.693
Year: 2030, Predicted HDI: 0.732
Year: 2035, Predicted HDI: 0.771
Year: 2040, Predicted HDI: 0.810
Year: 2045, Predicted HDI: 0.848
Year: 2050, Predicted HDI: 0.887
```

```
/opt/anaconda3/lib/python3.12/site-packages/sklearn/base.py:493: UserWarning: X
does not have valid feature names, but LinearRegression was fitted with feature
names
  warnings.warn(
```

[25]:

```
# Plot the actual data points
plt.scatter(X, y, color='blue', label='Actual HDI')

# Plot the regression line based on the model
plt.plot(X, model.predict(X), color='red', label='Regression Line')

# Plot the predicted future values
plt.scatter(future_years, future_hdi, color='green', label='Predicted HDI')

# Add labels, title, and legend
plt.xlabel('Year')
plt.ylabel('Human Development Index (HDI)')
plt.title('Year vs HDI Linear Regression')
plt.legend()

# Display the plot
plt.show()
```



3.4 Year *vs* Life Expectancy (Male, Female and combined)

[27]:
```
# Drop rows where Life Expectancy values are missing
df_cleaned = df.dropna(subset=['Life expectancy - Women', 'Life expectancy -␣
↪Men', 'Life expectancy'])

# Prepare the independent variable (Year)
X = df_cleaned[['Year']]

# Prepare dependent variables for Women, Men, and Overall Life Expectancy
y_women = df_cleaned['Life expectancy - Women']
y_men = df_cleaned['Life expectancy - Men']
y_overall = df_cleaned['Life expectancy']

# Initialize Linear Regression model
model_women = LinearRegression()
model_men = LinearRegression()
model_overall = LinearRegression()
```

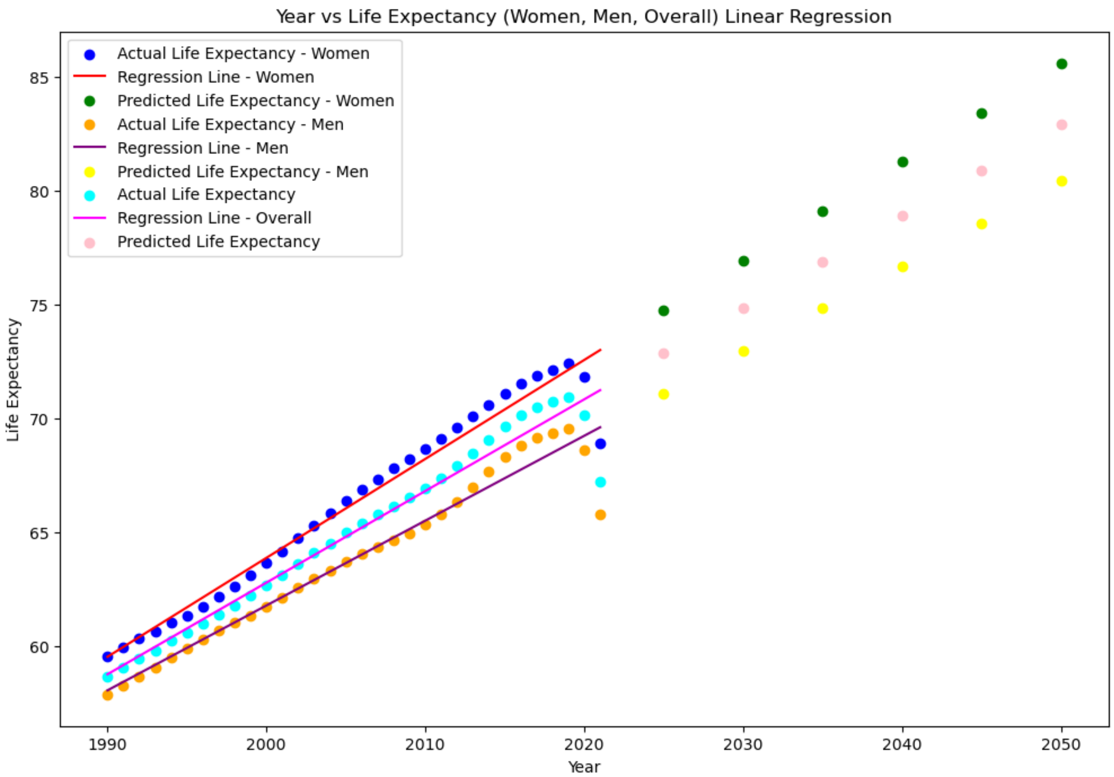
```
# Fit the model for each dependent variable
model_women.fit(X, y_women)
model_men.fit(X, y_men)
model_overall.fit(X, y_overall)

# Create a numpy array of future years for prediction
future_years = np.array([[2025], [2030], [2035], [2040], [2045], [2050]])

# Predict Life Expectancy for future years
future_life_expectancy_women = model_women.predict(future_years)
future_life_expectancy_men = model_men.predict(future_years)
future_life_expectancy_overall = model_overall.predict(future_years)
```

```
/opt/anaconda3/lib/python3.12/site-packages/sklearn/base.py:493: UserWarning: X
does not have valid feature names, but LinearRegression was fitted with feature
names
  warnings.warn(
/opt/anaconda3/lib/python3.12/site-packages/sklearn/base.py:493: UserWarning: X
does not have valid feature names, but LinearRegression was fitted with feature
names
  warnings.warn(
/opt/anaconda3/lib/python3.12/site-packages/sklearn/base.py:493: UserWarning: X
does not have valid feature names, but LinearRegression was fitted with feature
names
  warnings.warn(
```

[28]:

```
# Plot the actual data points and predictions
plt.figure(figsize=(12, 8))

# Plot actual data for Women
plt.scatter(X, y_women, color='blue', label='Actual Life Expectancy - Women')
plt.plot(X, model_women.predict(X), color='red', label='Regression Line -␣
 ↪Women')
plt.scatter(future_years, future_life_expectancy_women, color='green',␣
 ↪label='Predicted Life Expectancy - Women')

# Plot actual data for Men
plt.scatter(X, y_men, color='orange', label='Actual Life Expectancy - Men')
plt.plot(X, model_men.predict(X), color='purple', label='Regression Line - Men')
plt.scatter(future_years, future_life_expectancy_men, color='yellow',␣
 ↪label='Predicted Life Expectancy - Men')

# Plot actual data for Overall Life Expectancy
plt.scatter(X, y_overall, color='cyan', label='Actual Life Expectancy')
plt.plot(X, model_overall.predict(X), color='magenta', label='Regression Line -␣
 ↪Overall')
```

```
plt.scatter(future_years, future_life_expectancy_overall, color='pink',␣
↪label='Predicted Life Expectancy')

# Add labels, title, and legend
plt.xlabel('Year')
plt.ylabel('Life Expectancy')
plt.title('Year vs Life Expectancy (Women, Men, Overall) Linear Regression')
plt.legend()

# Display the plot
plt.show()
```

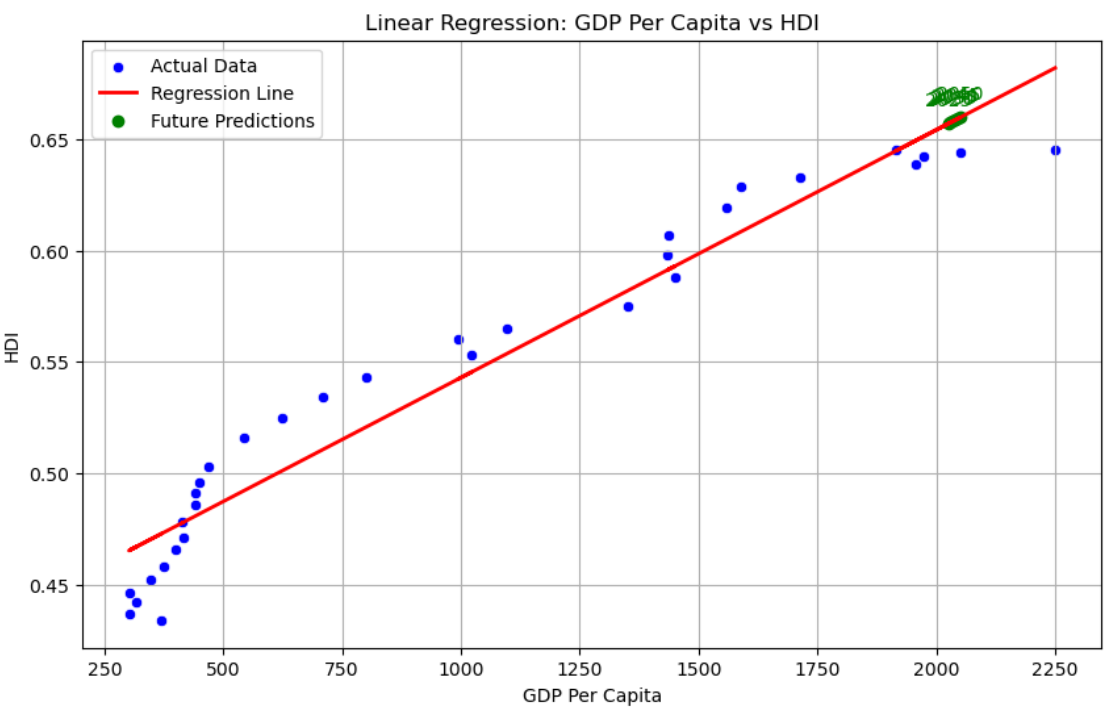


3.5 GDP Per Capita *vs* HDI

[30]:
```
# Define the future years for prediction
future_years = np.array([[2025], [2030], [2035], [2040], [2045], [2050]])

# Reshape the data
X = df['GDP Per Capita'].values.reshape(-1, 1)
y = df['HDI'].values.reshape(-1, 1)

# Create a linear regression model
```

```
model = LinearRegression()
model.fit(X, y)

# Predict y values for the original and future years
y_pred = model.predict(X)
future_y_pred = model.predict(future_years)
```

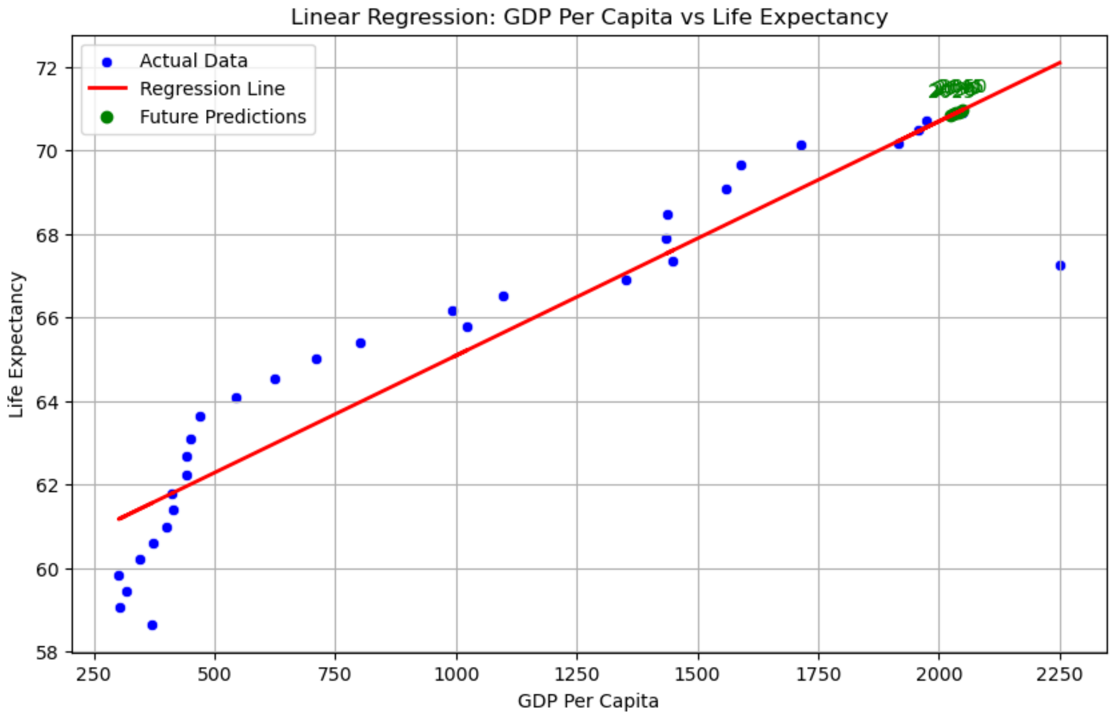
[31]:
```
# Plotting
plt.figure(figsize=(10, 6))
sns.scatterplot(x=X.flatten(), y=y.flatten(), color='blue', label='Actual Data')
plt.plot(X, y_pred, color='red', linewidth=2, label='Regression Line')

# Plot future predictions
plt.scatter(future_years, future_y_pred, color='green', marker='o',␣
↪label='Future Predictions')
for i, txt in enumerate(future_years.flatten()):
    plt.annotate(txt, (future_years[i], future_y_pred[i]), textcoords="offset␣
↪points", xytext=(0,10), ha='center', color='green')

plt.title('Linear Regression: GDP Per Capita vs HDI')
plt.xlabel('GDP Per Capita')
plt.ylabel('HDI')
plt.legend()
plt.grid()
plt.show()
```



3.6 GDP Per Capita *vs* Life Expectancy

[33]:
```
# Define the future years for prediction
future_years = np.array([[2025], [2030], [2035], [2040], [2045], [2050]])

# Step 3: Perform Linear Regression and Plot for GDP Per Capita vs Life␣
↪Expectancy
# Reshape the data
X = df['GDP Per Capita'].values.reshape(-1, 1)
y = df['Life expectancy'].values.reshape(-1, 1)

# Create a linear regression model
model = LinearRegression()
model.fit(X, y)

# Predict y values for the original and future years
y_pred = model.predict(X)
future_y_pred = model.predict(future_years)
```

[34]:
```
# Plotting
plt.figure(figsize=(10, 6))
sns.scatterplot(x=X.flatten(), y=y.flatten(), color='blue', label='Actual Data')
plt.plot(X, y_pred, color='red', linewidth=2, label='Regression Line')

# Plot future predictions
plt.scatter(future_years, future_y_pred, color='green', marker='o',␣
↪label='Future Predictions')
for i, txt in enumerate(future_years.flatten()):
    plt.annotate(txt, (future_years[i], future_y_pred[i]), textcoords="offset␣
↪points", xytext=(0,10), ha='center', color='green')

plt.title('Linear Regression: GDP Per Capita vs Life Expectancy')
plt.xlabel('GDP Per Capita')
plt.ylabel('Life Expectancy')
plt.legend()
plt.grid()
plt.show()
```

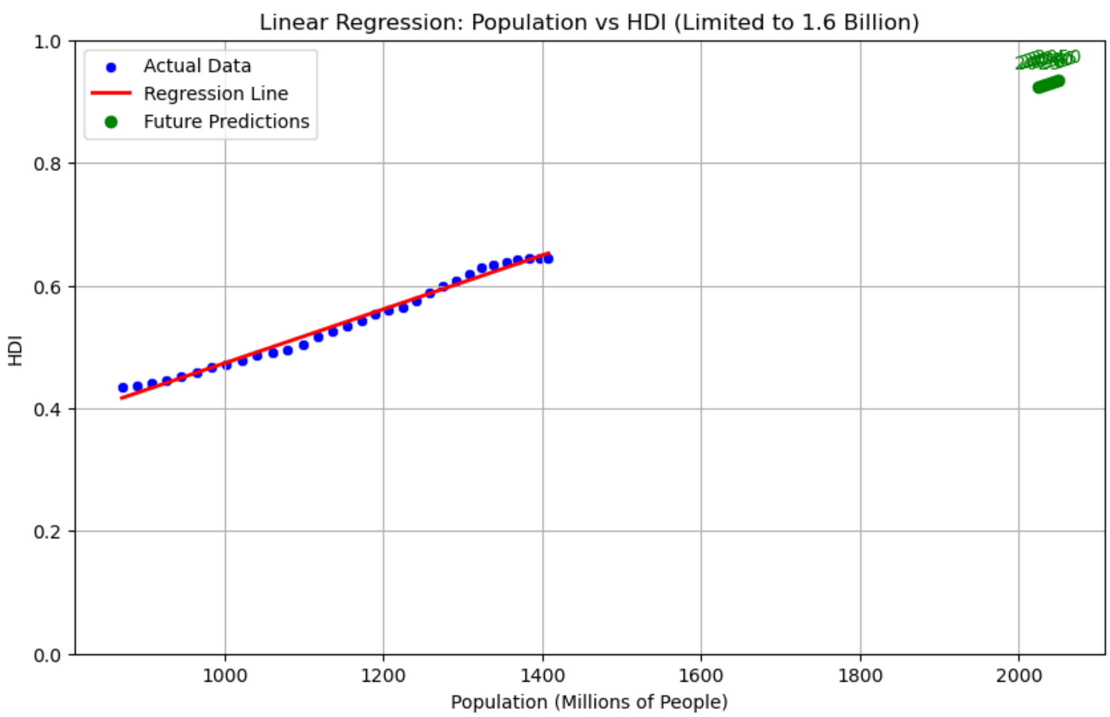


3.7 Population *vs* HDI

[36]:
```
# Define the future years for prediction
future_years = np.array([[2025], [2030], [2035], [2040], [2045], [2050]])

# Filter the DataFrame to limit population to 1.6 billion
filtered_df = df[df['Population (Millions of people)'] <= 1600]

# Step 3: Perform Linear Regression and Plot for Population vs HDI
# Reshape the data
X = filtered_df['Population (Millions of people)'].values.reshape(-1, 1)
y = filtered_df['HDI'].values.reshape(-1, 1)

# Create a linear regression model
model = LinearRegression()
model.fit(X, y)

# Predict y values for the original and future years
y_pred = model.predict(X)
future_y_pred = model.predict(future_years)
```

[37]:
```
# Plotting
plt.figure(figsize=(10, 6))
```

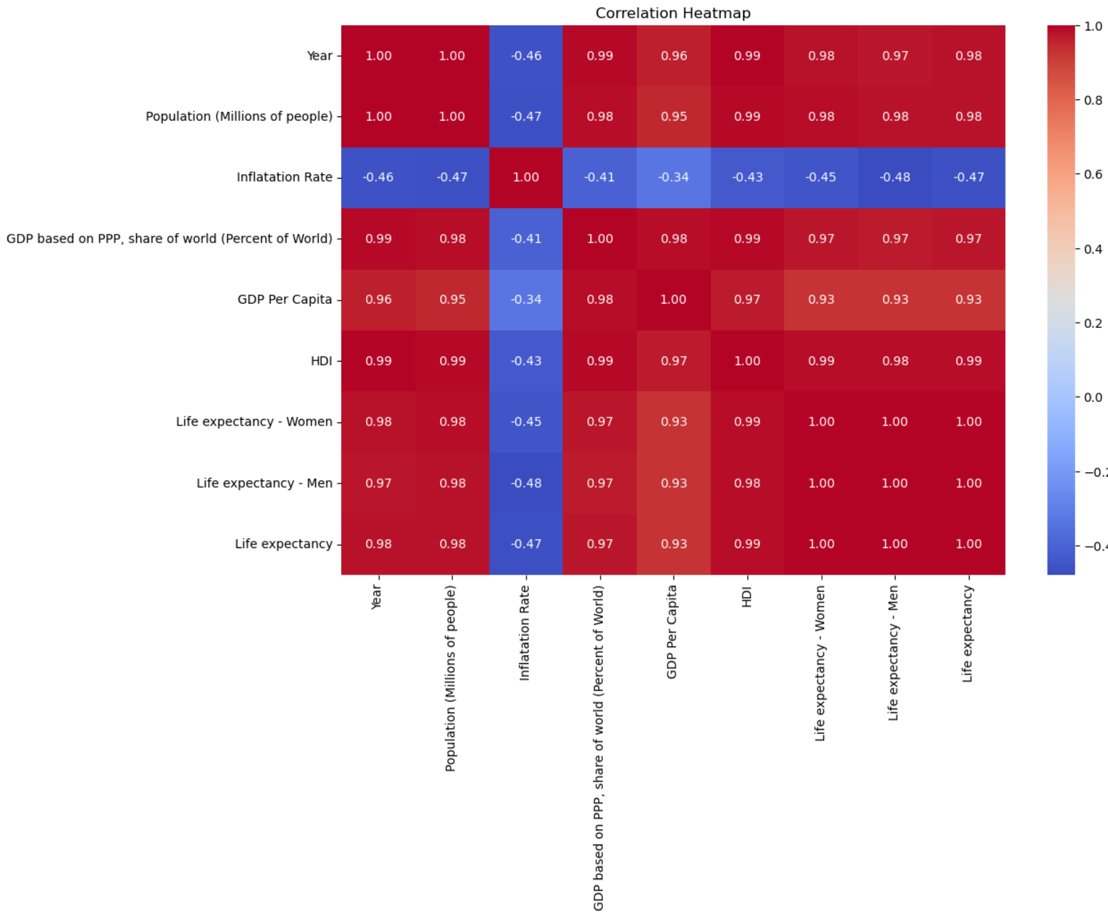
```
sns.scatterplot(x=X.flatten(), y=y.flatten(), color='blue', label='Actual Data')
plt.plot(X, y_pred, color='red', linewidth=2, label='Regression Line')

# Plot future predictions
plt.scatter(future_years, future_y_pred, color='green', marker='o',␣
 ↪label='Future Predictions')
for i, txt in enumerate(future_years.flatten()):
    plt.annotate(txt, (future_years[i], future_y_pred[i]), textcoords="offset␣
 ↪points", xytext=(0,10), ha='center', color='green')

# Set y-axis limit to a reasonable range based on your data
plt.ylim(0, 1)  # Adjust the upper limit as needed based on your HDI data range

plt.title('Linear Regression: Population vs HDI (Limited to 1.6 Billion)')
plt.xlabel('Population (Millions of People)')
plt.ylabel('HDI')
plt.legend()
plt.grid()
plt.show()
```



4 Some Heatmaps

4.1 Correlation heatmap

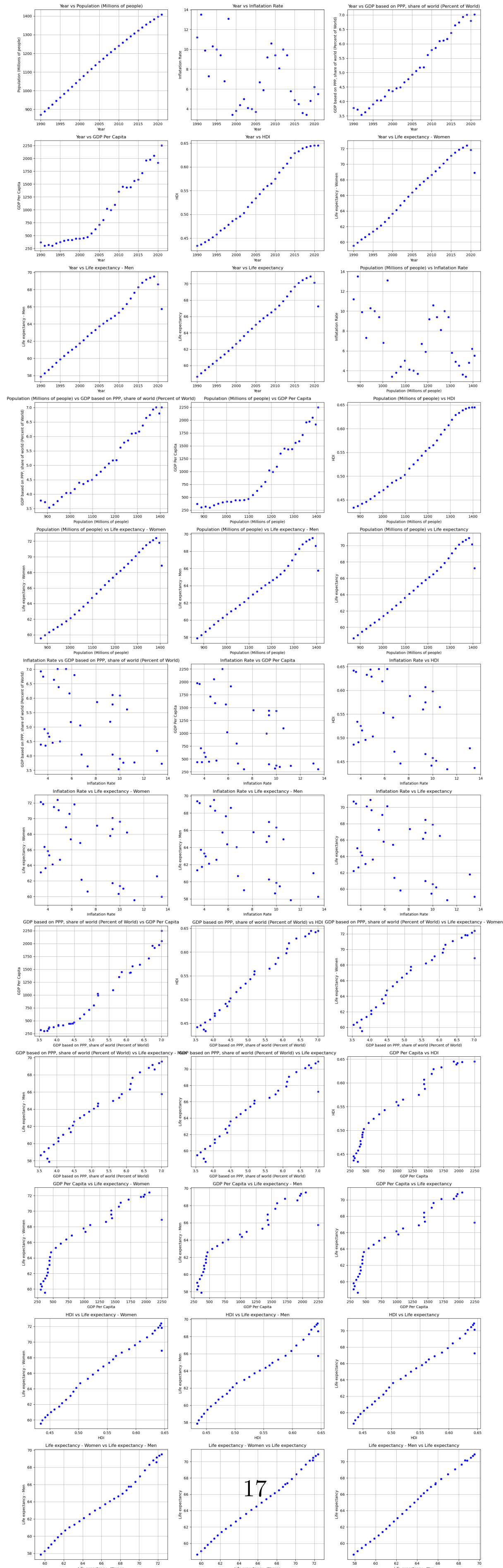
[39]:
```
plt.figure(figsize=(12, 8))
correlation_matrix = df.corr()
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt='.2f')
plt.title('Correlation Heatmap')
plt.show()
```



5 Scatter Plot Array

[41]:
```
import matplotlib.pyplot as plt
import seaborn as sns

# Define the numerical columns from your dataset
numerical_columns = df.select_dtypes(include=['float64', 'int64']).columns
```

```
# Set the number of rows and columns for the subplot grid
num_vars = len(numerical_columns)
ncols = 3  # Set the number of columns for the array
nrows = (num_vars * (num_vars - 1) // 2 + ncols - 1) // ncols  # Calculate the␣
↪number of rows needed

# Create the figure and axes for the subplots
fig, axs = plt.subplots(nrows=nrows, ncols=ncols, figsize=(18, nrows * 5))
axs = axs.flatten()  # Flatten the array of axes for easy iteration

# Initialize the index for subplot axes
index = 0

# Loop through each combination of numerical variables and create scatterplots
for i in range(num_vars):
    for j in range(i + 1, num_vars):
        x_var = numerical_columns[i]
        y_var = numerical_columns[j]
        sns.scatterplot(data=df, x=x_var, y=y_var, ax=axs[index], color='blue')
        axs[index].set_title(f'{x_var} vs {y_var}')
        axs[index].set_xlabel(x_var)
        axs[index].set_ylabel(y_var)
        axs[index].grid()
        index += 1

# Remove any unused subplots
for j in range(index, nrows * ncols):
    fig.delaxes(axs[j])

# Adjust layout
plt.tight_layout()
plt.show()
```


Year vs Population (Millions of people)
Year vs Inflatation Rate
Year vs GDP based on PPP, share of world (Percent of World)
Year vs GDP Per Capita
Year vs HDI
Year vs Life expectancy - Women
Year vs Life expectancy - Men
Year vs Life expectancy
Population (Millions of people) vs Inflatation Rate
Population (Millions of people) vs GDP based on PPP, share of world (Percent of World)
Population (Millions of people) vs GDP Per Capita
Population (Millions of people) vs HDI
Population (Millions of people) vs Life expectancy - Women
Population (Millions of people) vs Life expectancy - Men
Population (Millions of people) vs Life expectancy
Inflatation Rate vs GDP based on PPP, share of world (Percent of World)
Inflatation Rate vs GDP Per Capita
Inflatation Rate vs HDI
Inflatation Rate vs Life expectancy - Women
Inflatation Rate vs Life expectancy - Men
Inflatation Rate vs Life expectancy
GDP based on PPP, share of world (Percent of World) vs GDP Per Capita
GDP based on PPP, share of world (Percent of World) vs HDI
GDP based on PPP, share of world (Percent of World) vs Life expectancy - Women
GDP based on PPP, share of world (Percent of World) vs Life expectancy - Men
GDP based on PPP, share of world (Percent of World) vs Life expectancy
GDP Per Capita vs HDI
GDP Per Capita vs Life expectancy - Women
GDP Per Capita vs Life expectancy - Men
GDP Per Capita vs Life expectancy
HDI vs Life expectancy - Women
HDI vs Life expectancy - Men
HDI vs Life expectancy
Life expectancy - Women vs Life expectancy - Men
Life expectancy - Women vs Life expectancy
Life expectancy - Men vs Life expectancy