Quantifying Wellbeing: A Data Science Approach to Understanding Human Longevity

2025-06-20

Project Motivation

This project was conducted independently to analyze socio-economic factors affecting Life Expectancy at Birth (LEB). Using a real-world dataset, the aim was to explore linear and non-linear relationships and build predictive models. The project uses Python and statistical techniques such as Pearson and Spearman correlations, linear regression, and data transformation to draw conclusions and validate model accuracy.

Step 1: Splitting the Data

To begin the analysis, I split the dataset into two subsets using the `train_test_split` function from scikit-learn. The training set comprises **75%** of the data, while the remaining **25%** is reserved for testing. A random seed (SEED = 123456) ensures reproducibility. All columns, including `Country`, were retained to allow flexible post-analysis.

Step 2: Single-Variable Model

a) Identifying the Strongest Correlate to LEB

Using the Pearson correlation coefficient, I identified the Human Development Index (HDI) as the variable with the strongest linear relationship to LEB.

Table 1: Variable with the Strongest Linear Relationship with LEB

Variable	**Pearson Correlation Coefficient**
Human Development Index (value)	0.93

b) Building the Linear Regression Model

A simple linear regression model was created using HDI as the predictor and LEB as the target. The model was trained on the training set and yielded the following results:

Table 2: Linear Model Results

Metric	Value
Slope	48.85
Intercept	36.86
R^2 (Coefficient of Determination)	0.87

Figure 1: Scatter plot showing relationship between HDI and LEB

c) Test Set Prediction and Evaluation

The model was applied to the test set to evaluate its performance. I calculated both the Pearson correlation between predicted and actual LEB values and the Mean Squared Error (MSE).

Table 3: Model Performance Metrics on the Test Set

Metric	Value
Correlation between predicted and actual LEB	0.9133
Mean Squared Error (MSE)	11.4083

d) Interpretation

Since HDI incorporates life expectancy, education, and income levels, its strong relationship with LEB is intuitive. These elements are known to influence health outcomes. Higher HDI is generally associated with improved living standards and healthcare access, which directly affects life expectancy [2, 1].

Step 3: Exploring Non-Linear Relationships

To identify non-linear associations, I visualized scatter plots of all variables against LEB. One such relationship was between Gross National Income (GNI) and LEB, which suggested a logarithmic curve.

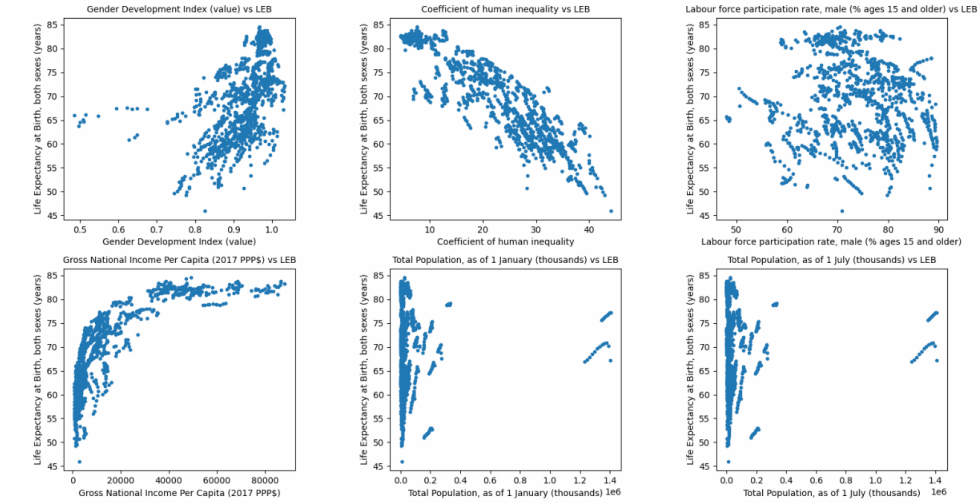

Figure 2: Scatter plots between LEB and selected variables

The correlations for GNI before and after logarithmic transformation are as follows:

Table 4: Correlation between GNI and LEB (Before Transformation)

Correlation Type	**Correlation Value**
Pearson Correlation	0.77
Spearman Correlation	0.89

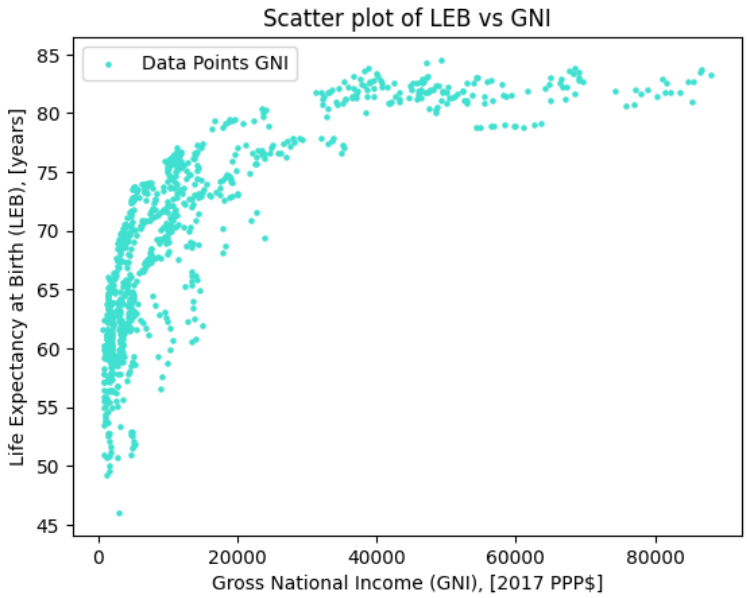

Figure 3: LEB vs GNI (before log transformation)

Table 5: Correlation after Logarithmic Transformation

Correlation Type	**Correlation Value**
Pearson Correlation (after transformation)	0.88

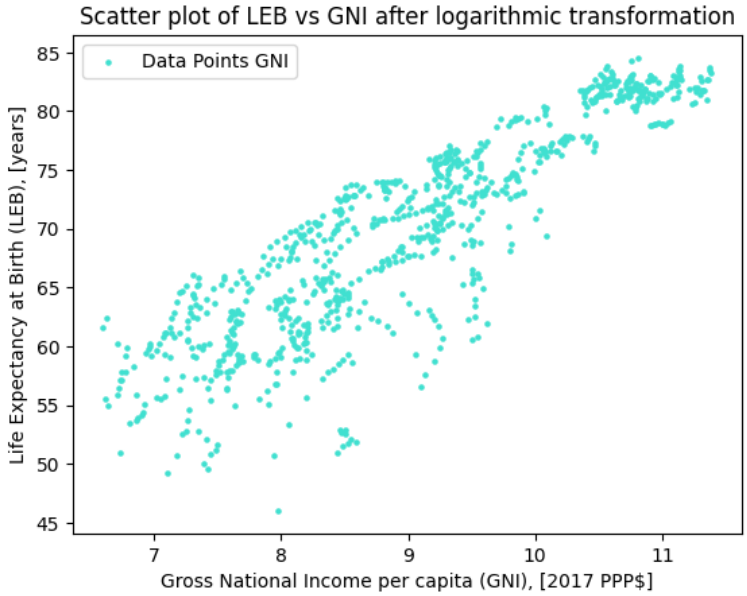

Figure 4: LEB vs log(GNI)

Step 4: Multiple Linear Regression Model

To enhance prediction performance, I constructed a multiple linear regression model using the top three most correlated features with LEB:

- Median Age
- Expected Years of Schooling, female
- Expected Years of Schooling (overall)

Table 6: Top Variables Correlated with LEB

Variable	**Pearson Correlation**
Median Age	0.8955
Expected Years of Schooling, female	0.8453
Expected Years of Schooling (overall)	0.8261

Table 7: Multiple Linear Model Coefficients

Metric	**Value**
Coefficients	0.72, 2.64, -2.25
Intercept	46.84
R^2 (Training)	0.836

Table 8: Model Performance Metrics on Test Set

Metric	**Value**
Correlation between predicted and actual LEB	0.9209
Mean Squared Error (MSE)	10.2688

Conclusion

This project demonstrated that both single and multiple regression models can predict life expectancy reasonably well using socio-economic indicators. The HDI was the strongest single predictor, but using a combination of education and demographic variables led to slightly improved performance. Transforming variables like GNI further enhanced correlation, validating the non-linear nature of some relationships.

References

- UNDP: Human Development Reports. `https://hdr.undp.org`
- Our World in Data: Life Expectancy. `https://ourworldindata.org/life-expectancy`

References

[1] Our World in Data. *Life Expectancy: How is it calculated and how should it be interpreted?* Retrieved 2024-11-29. URL: https://ourworldindata.org/life-expectancy-how-is-it-calculated-and-how-should-it-be-interpreted.

[2] UNDP (United Nations Development Programme). *Human Development Index (HDI) Data Center*. Retrieved 2024-11-27. 2023. URL: https://hdr.undp.org/data-center/human-development-index#/indicies/HDI.

Appendix

Problem 1 - Importing necessary libraries and splitting the data

```
#Import necessary libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error
from scipy.stats import pearsonr, spearmanr
from sklearn.metrics import r2_score

#Connect to drive to be able to import files
from google.colab import drive
drive.mount('/content/drive')

#Load the csv file and dropping empty rows
df = pd.read_csv('/content/drive/MyDrive/
    life_expectancy.csv').dropna()
print(df.columns)

#Split the dataset into training and test
SEED = 123456
df_train, df_test = train_test_split(df, random_state=
    SEED, test_size=0.25)
print("Training data size:", df_train.shape)
print("Testing data size:", df_test.shape)

Dataset = len(df)
TrainingSet = len(df_train)
TestSet= len(df_test)

print(f"Dataset (Length: {Dataset}) divided to  \
    nTraining Set (Length: {TrainingSet}) and Test Set
    (Length: {TestSet}) ")
```

Problem 2 - Single-variable model

a) Variable with the strongest linear relationship to LEB

```
#Target variable
LEB = "Life Expectancy at Birth, both sexes (years)"

#only numeric columns are selected
df_train = df_train.select_dtypes(include='number')
df_train = df_train.dropna()

#Calculate the correlation for all variables with LEB
pearson = df_train.corr(method='pearson')

```

```
#Drop the correlation of LEB with itself
pearson_drop = pearson[LEB].drop(LEB)

#Find the maximum correlation and the corresponding
    variable
strongest_corr = pearson_drop.max()
strongest_var = pearson_drop.loc[pearson_drop ==
    strongest_corr].index[0]

#Print the strongest variable and its correlation
print(f"The variable with the strongest linear
    relationship with LEB is '{strongest_var}' "
       f"with a Pearson correlation coefficient of {
    strongest_corr:.2f}.")
```

b) Single-variable linear model

```
#Initialize the Linear Regression model
lr = LinearRegression()

#Define the independent and dependent variables
independent_variables = ['Human Development Index (
    value)']
dependent_variable = LEB

#Prepare the training set
X_train = df_train[independent_variables]
y_train = df_train[dependent_variable]

#Fit the model to the training data
lr.fit(X_train,y_train)

#Coefficient of determination slope and intercept
slope = lr.coef_[0]
intercept = lr.intercept_

# R2 coefficient of determination
r2=lr.score(X_train,y_train)

# calculate the regression line by slope and intercept
regression_line = intercept + slope * X_train

print(f"Slope: {slope:.2f}")
print(f"Intercept: {intercept:.2f}")
print(f"R2 (Coefficient of Determination): {r2:.2f}")

#Scatter plot showing the LEB as the function of the
    variable
plt.scatter(X_train, y_train, color='Turquoise', label
    ='Data Points',s=10)
```

```
#plot the regression line
plt.plot(X_train, regression_line, color='red', label='Regression Line')

#Plot the regression line over the scatter plot
plt.xlabel('Human Development Index (value)')
plt.ylabel('Life Expectancy at Birth, both sexes (years)')
plt.title(f"Linear Regression: {dependent_variable} vs Human Development Index (value)")
plt.legend()
plt.show()
```

c) Predict the LEB for the test set

```
#Prepare the test set
X_test = df_test[independent_variables]
y_test = df_test[dependent_variable]

#predict the LEB for test set
y_pred_test = lr.predict(X_test)

#calculate the correlation between the predicted values and the target variable LEB
correlation, _ = pearsonr(y_test, y_pred_test)

# Mean Squared Error (MSE) between the predictions and the true target variable values
mse = mean_squared_error(y_test, y_pred_test)

print(f"Correlation between predicted and the target variable LEB: {correlation:.4f}")
print(f"Mean squared error (MSE): {mse:.4f}")
```

Problem 3 - Non-linear relationship

```
#Plotting all variables compared to LEB for visual study

#Plotting done on all data, not just training data

dependent_variable = LEB
excluded_columns = [LEB, 'Human Development Index (value)'] #Removing LEB and HDI
independent_variables = [col for col in df.columns if col not in excluded_columns]

#Create subplots
n_cols = 3
n_rows = 16
```

```
fig, axes = plt.subplots(n_rows, n_cols, figsize=(15, 64))

#Flatten axes for easy iteration
axes = axes.ravel()

#Plot scatter plots
for i, column in enumerate(independent_variables):
    axes[i].scatter(df[column], df[dependent_variable], s=10)
    axes[i].set_title(f"{column} vs LEB", fontsize = 10)
    axes[i].set_xlabel(column, fontsize = 10)
    axes[i].set_ylabel(dependent_variable, fontsize = 10)

#Adjust layout and show
plt.tight_layout()
plt.show()
#Pearson and spearman coefficient: GNI and LEB

#Define the independent and dependent variables
GNI = "Gross National Income Per Capita (2017 PPP$)"
#Using training data
independent_variable = df_train[GNI]
dependent_variable = df_train[LEB]

#Calculate Pearson and Spearman correlations
pearson_corr, _ = pearsonr(independent_variable, dependent_variable)
spearman_corr, _ = spearmanr(independent_variable, dependent_variable)

#Print results
print(f"Correlation between GNI and LEB:")
print(f"  Pearson Correlation: {pearson_corr:.2f}")
print(f"  Spearman Correlation: {spearman_corr:.2f}")

#Scatter plot showing the LEB as the function of the GNI
plt.scatter(independent_variable, dependent_variable, color='Turquoise', label='Data Points GNI',s=5)

#Plot
plt.xlabel('Gross National Income (GNI), [2017 PPP$]')
plt.ylabel('Life Expectancy at Birth (LEB), [years]')
plt.title(f"Scatter plot of LEB vs GNI")
plt.legend()
plt.show()

```

```
#Logarithmic transformation on GNI data
independent_variable_transformed = np.log(
    independent_variable)

#Calculate Pearson and Spearman correlations after
    transformation
pearson_corr_transformed, _ = pearsonr(
    independent_variable_transformed,
    dependent_variable)

#Print results
print(f"Correlation between GNI and LEB after
    transformation:")
print(f"  Pearson Correlation: {
    pearson_corr_transformed:.2f}")

#Scatter plot showing the LEB as the function of the
    GNI after transformation
plt.scatter(independent_variable_transformed,
    dependent_variable, color='Turquoise', label='Data
    Points GNI',s=5)

#Plot
plt.xlabel('Gross National Income per capita (GNI),
    [2017 PPP$]')
plt.ylabel('Life Expectancy at Birth (LEB), [years]')
plt.title(f"Scatter plot of LEB vs GNI after
    logarithmic transformation")
plt.legend()
plt.show()
```

Problem 4 - Multiple linear regression

```
#Finding variables with high pearson correlation to
    LEB

#Only numeric columns are selected
df_train = df_train.select_dtypes(include='number')
df_train = df_train.dropna()

#Calculate the correlation for all variables with LEB
pearson = df_train.corr(method='pearson')

#Drop the correlation of LEB with itself and HDI
pearson_drop = pearson[LEB].drop([LEB, 'Human
    Development Index (value)'])

#Sort in descending order
sorted_pearson = pearson_drop.sort_values(ascending=
    False)
```

```
print(sorted_pearson)


#Initialize the Linear Regression model
lr = LinearRegression()

#Define the independent and dependent variables
independent_variables = ['Median Age, as of 1 July (years)', 'Expected Years of Schooling, female (years)', 'Expected Years of Schooling (years)']
dependent_variable = LEB

#Prepare the training set
X_train = df_train[independent_variables]
y_train = df_train[dependent_variable]

#Prepare the test set
X_test = df_test[independent_variables]
y_test = df_test[dependent_variable]

#Fit the model to the training data
lr.fit(X_train,y_train)

#Coefficients intercept
coefficients = lr.coef_
intercept = lr.intercept_

#R^2 coefficient of determination
r2=lr.score(X_train,y_train)

print("Coefficients:", coefficients)
print(f"Intercept: {intercept:.2f}")
print(f"R^2 (Coefficient of Determination): {r2:.3f}")


#Predict the LEB for test set
y_pred_test = lr.predict(X_test)

#calculate the correlation between the predicted values and the target variable LEB
correlation, _ = pearsonr(y_test, y_pred_test)

# Mean Squared Error (MSE) between the predictions and the true target variable values
mse = mean_squared_error(y_test, y_pred_test)

print(f"Pearson Correlation between predicted and the target variable LEB: {correlation:.4f}")
print(f"Mean squared error (MSE): {mse:.4f}")
```