DAT565/DIT407 Assignment 4

Giacomo Guidotto
gusguigi@student.gu.se

Leong Jia Yi, Janna
gusleoji@student.gu.se

2024-10-03

1 Problem 1: Splitting the data

We used the scikit learn *train_test_split()* function to split the data, with an 80-20 ratio between train and test data.

We chose this amount as it provides a substantial amount of training data for the model to learn from while giving enough data to test its accuracy. Furthermore, this 80-20 split has been shown to prevent overfitting [1]. We also randomized the train_test split to prevent biases within the train and test data.

2 Problem 2: Single-variable model

To find a variable in the dataset with the strongest linear relationship with the *life expectancy at birth (LEB)* we computed the Pearson product-moment correlation coefficient with all the dataset variables.

As reported in table 1, we can identify the variable **Human Development Index (value)** with a Pearson product-moment correlation coefficient of **0.918341** as the top candidate.

By definition, the Human Development Index is a "summary measure of average achievement in key dimensions of human development: a long and healthy life..." where one of its measurements is "life expectancy at birth" [2]. Thus, as a higher life expectancy in a country will directly cause the HDI value to increase, it explains why they have a strong relationship.

After training a linear regression model with this variable we were able to acquire the following metrics:

- Coefficient of determination R^2: **0.843**
- Coefficients: **51.423**
- Y-intercept: **34.605**

Figure 1 shows the relationship between the target variable *LEB* and the candidate variable *Human Development Index (value)* with our model that predicts its trend.

Finally, we predicted the test set and evaluated the Pearson product-moment correlation coefficient between the prediction and the real data at **0.920** while the Mean Squared Error at **12.519**

Life Expectancy at Birth, both sexes (years)	1.000000
Human Development Index (value)	**0.918341**
Expected Years of Schooling, female (years)	0.814086
Expected Years of Schooling (years)	0.799646
Median Age, as of 1 July (years)	0.797353
Expected Years of Schooling, male (years)	0.778834
Mean Years of Schooling, female (years)	0.749029
Mean Years of Schooling (years)	0.743001
⋮	⋮
Crude Death Rate (deaths per 1,000 population)	-0.565175
Inequality in eduation	-0.678548
Rate of Natural Change (per 1,000 population)	-0.714862
Net Reproduction Rate (surviving daughters per woman)	-0.777402
Adolescent Birth Rate (births per 1,000 women ages 15-19)	-0.799662
Total Fertility Rate (live births per woman)	-0.838654
Coefficient of human inequality	-0.849600
Crude Birth Rate (births per 1,000 population)	-0.864138

Table 1: Pearson product-moment correlation coefficient between the dataset variables and LEB

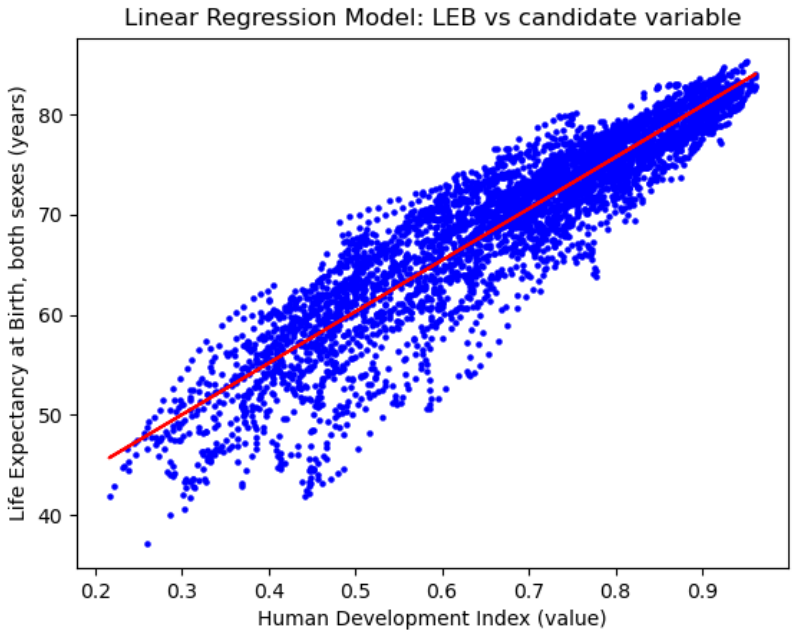

Figure 1: Scatter plot showing LEB as a function of the the 'Human Development Index (value)' variable

3 Problem 3: Non-linear relationship

To find a variable in the dataset with a strong non-linear relationship with the target variable, we computed the Spearman's rank correlation coefficient across the dataset, as shown in table 2. We identify the **Gross National Income Per Capita (2017 PPP$)** as the candidate variable with the strongest relationship. From figure 2, we see that the two variables have a logarithmic relationship.

Life Expectancy at Birth, both sexes (years)	1.00000
Gross National Income Per Capita (2017 PPP$)	**0.86483**
Median Age, as of 1 July (years)	0.86376
Crude Birth Rate (births per 1,000 population)	0.84864
Expected Years of Schooling, female (years)	0.83457
Coefficient of human inequality	0.82890
Expected Years of Schooling (years)	0.81976
Total Fertility Rate (live births per woman)	0.81694
⋮	⋮
Male Deaths (thousands)	0.13772
Mean Age Childbearing (years)	0.10888
Female Population, as of 1 July (thousands)	0.05137
Total Population, as of 1 July (thousands)	0.04717
Total Population, as of 1 January (thousands)	0.04562
Male Population, as of 1 July (thousands)	0.04383
Population Sex Ratio, as of 1 July (males per 100 females)	0.04334
Country	0.01488

Table 2: Spearman rank correlation coefficient between the dataset variables and LEB

Thus, by performing a log transformation on this second candidate, we get an approximately linear relationship with the target variable as seen in the scatter plot below 3.

We can see that a linear regression model is more efficient in the transformed scale simply by looking at the Pearson correlation coefficient before and after the transformation:

- r before the transformation: **0.651**
- r after the transformation: **0.833**

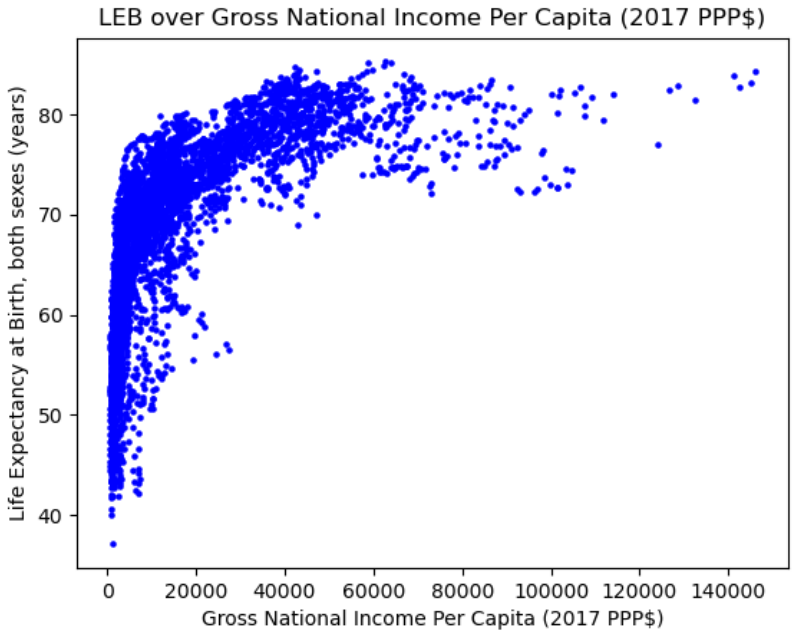

Figure 2: Scatter plot of 'Gross National Income Per Capita (2017 PPP$)' against LEB

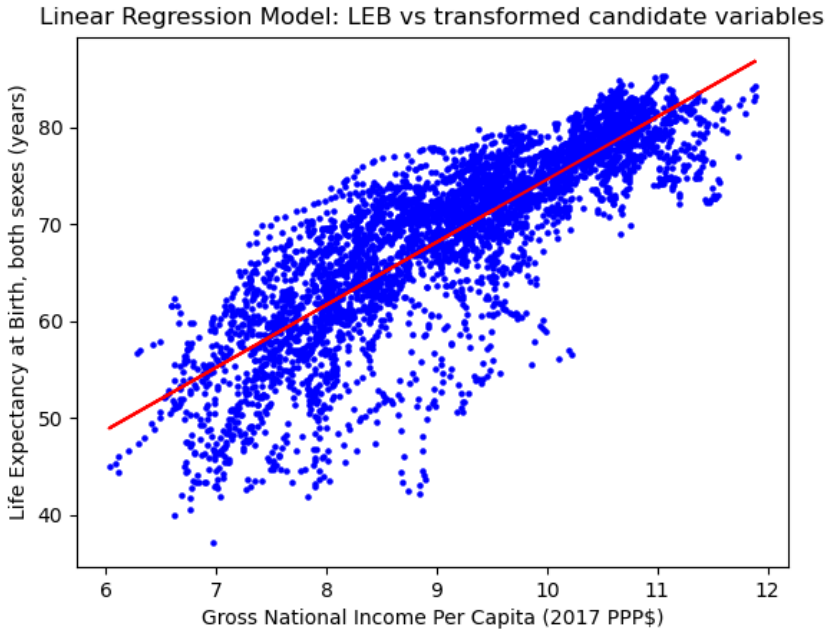

Figure 3: Transformed scatter plot of 'Gross National Income Per Capita (2017 PPP$)' against LEB

4 Problem 4: Multiple linear regression model

We believe the Pearson correlation coefficient effectively selects variables as it indicates how strong the relationship between a variable and LEB is. Thus we consider the top 9 variables with Pearson correlation coefficients of the highest magnitude.

These 9 variables included *Expected Years of Schooling (years)*, *Expected Years of Schooling. female (years)* and *Expected Years of Schooling. male (years)*.

However, *Expected Years of Schooling (years)* can capture information from *Expected Years of Schooling. female (years)* and *Expected Years of Schooling. male (years)*.

This leaves us with 7 remaining variables to consider further and test.

- Crude Birth Rate (births per 1,000 population)
- Coefficient of human inequality
- Total Fertility Rate (live births per woman)
- Adolescent Birth Rate (births per 1,000 women ages 15-19)
- Expected Years of Schooling (years)
- Median Age, as of 1 July (years)
- Net Reproduction Rate (surviving daughters per woman)

After training a linear regression model with those, we were able to acquire the following metrics:

- Coefficient of determination R^2: **0.867**
- Coefficients:
  - **-0.788**
  - **-0.113**
  - **-8.605**
  - **-0.015**
  - **0.322**
  - **0.085**
  - **29.0244**
- Y-intercept: **71.766**

Finally, after predicting the test set we also evaluated the Pearson product-moment correlation coefficient between the prediction and the real data to be **0.936** and the Mean Squared Error to be **8.88**, a slight improvement from the single-variable model.

References

[1] Kosheleva Gholamy Kreinovich. *Why 70/30 or 80/20 Relation Between Training and Testing Sets: A Pedagogical Explanation.* Retrieved 2023-10-28. 2018. URL: https://scholarworks.utep.edu/cs_techrep/1209/.

[2] United Nations Human Development Reports. *Human Development Index (HDI).* Retrieved 2023-09-28. nd. URL: https://hdr.undp.org/data-center/human-development-index#/indicies/HDI.

A Code

Following is the Python code that we used to extract and visualize the information presented in this report:

A.1 Problem 1: Split the data

Prepare the data by encoding the non-numeric variables and applying a train-test split:

```
import pandas as pd
from sklearn.model_selection import \
        train_test_split
from sklearn.preprocessing import LabelEncoder

data = pd.read_csv('data/life_expectancy.csv')

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

data_train, data_test = train_test_split(
        data,
        test_size=0.2,
        random_state=42
)
```

A.2 Problem 2: Single variable linear regression model

A.2.1 Find a candidate variable

Identify the variables with the strongest relationship with the target variable 'Life Expectancy at Birth, both sexes (years)' using the Pearson correlation coefficient:

```
target_variable = \
        'Life␣Expectancy␣at␣Birth,␣both␣sexes␣(years)'

correlations = (
        data_train
```

```
        .corr(method='pearson')[target_variable]
        .abs().sort_values(ascending=False)
)
correlations
```

```
candidate_variable = correlations.index[1]
print(f'Candidate variable: "{candidate_variable}"')
```

A.2.2 Constructing the model

Construct a linear regression model using the variable with the strongest relationship with the target variable:

```
from sklearn.linear_model import LinearRegression

X_train = data_train[[candidate_variable]]
y_train = data_train[target_variable]

model = LinearRegression()
model.fit(X_train, y_train)
```

Computing the metrics of the model:

```
r_squared = model.score(X_train, y_train)
coefficients = model.coef_
intercept = model.intercept_

print(f'R-squared: {r_squared}')
print(f'Coefficients: {coefficients}')
print(f'Intercept: {intercept}')
```

Plot the linear regression model:

```
import matplotlib.pyplot as plt

plt.scatter(X_train, y_train, color='blue', s=5)
plt.plot(
        X_train, model.predict(X_train),
        color='red'
)
plt.title(
        'Linear Regression Model: LEB vs candidate variable'
)
plt.xlabel(candidate_variable)
plt.ylabel(target_variable)

plt.savefig(
        "scatter_plot_single_linear_model.png"
)
```

A.2.3 Predict the test set

Predict the target variable using the test data and computing the **mean squared error** and **correlation coefficient**:

```
from scipy.stats import pearsonr
from sklearn.metrics import mean_squared_error

X_test = data_test[[candidate_variable]]
y_test = data_test[target_variable]
y_pred = model.predict(X_test)

mse = mean_squared_error(y_test, y_pred)
correlation, _ = pearsonr(y_pred, y_test)

print(f'Mean squared error: {mse}')
print(f'Correlation: {correlation}')
```

A.3 Problem 3: Non-linear relationship

A.3.1 Find a second candidate

Identify the variables with the strongest relationship with the target variable ‘Life Expectancy at Birth, both sexes (years)‘ using the Spearman correlation coefficient:

```
data_train = data_train.drop(
        columns=[candidate_variable]
)

correlations = (
        data_train
        .corr(method='spearman')[target_variable]
        .abs().sort_values(ascending=False)
)
correlations
```

After choosing ‘Gross National Income Per Capita (2017 PPP$)‘ as candidate, we plot the relationship:

```
second_candidate_variables = 'Gross National Income Per Capita (2017 PPP$)'

plt.scatter(
        data_train[second_candidate_variables],
        data_train[target_variable],
        color='blue', s=5
)
plt.title(
        f'LEB over {second_candidate_variables}'
)
plt.xlabel(second_candidate_variables)
plt.ylabel(target_variable)
```

```
13
14 plt.savefig(
15         "scatter_plot_nonlinear_mono_model.png"
16 )
```

The relationship appears to be logarithmic.

A.3.2 Construct the model on the transformed scale

Applying the logarithmic transformation to the candidate variable:

```
1 import numpy as np
2
3 X_train = data_train[[second_candidate_variables]]
4 y_train = data_train[target_variable]
5
6 log_X_train = np.log(X_train)
7
8 model = LinearRegression()
9 model.fit(log_X_train, y_train)
```

Computing the metrics of the model:

```
1 r_squared = model.score(log_X_train, y_train)
2 coefficients = model.coef_
3 intercept = model.intercept_
4
5 print(f'R-squared: {r_squared}')
6 print(f'Coefficients: {coefficients}')
7 print(f'Intercept: {intercept}')
```

Plotting the linear regression model:

```
1 plt.scatter(
2         log_X_train, y_train,
3         color='blue', s=5
4 )
5 plt.plot(
6         log_X_train, model.predict(log_X_train),
7         color='red'
8 )
9 plt.title(
10         'Linear Regression Model: LEB vs transformed candidate variables'
11 )
12 plt.xlabel(second_candidate_variables)
13 plt.ylabel(target_variable)
14
15 plt.savefig(
16         "scatter_plot_nonlinear_mono_transformed_model.png"
17 )
```

A.3.3 Comparing the transformation

Computing the correlation coefficient before and after the transformation

```
original_correlation, _ = (
        pearsonr(X_train.values.flatten(), y_train)
)
transformed_correlation, _ = (
        pearsonr(log_X_train.values.flatten(), y_train)
)

print(f'''
Original correlation:    {original_correlation}
Transformed correlation: {transformed_correlation}
''')
```

A.4 Problem 4: Multiple linear regression model

A.4.1 Research the candidates

We believe the Pearson coefficient is an effective way to select variables as it indicates how strong the relationship between a variable and LEB is.

We consider the top 9 variables with Pearson coefficients of the highest magnitude. These are:

- Crude Birth Rate (births per 1,000 population)
- Coefficient of human inequality
- Total Fertility Rate (live births per woman)
- Expected Years of Schooling, female (years)
- Adolescent Birth Rate (births per 1,000 women ages 15-19)
- Expected Years of Schooling (years)
- Median Age, as of 1 July (years)
- Expected Years of Schooling, male (years)
- Net Reproduction Rate (surviving daughters per woman)

However, 'Expected Years of Schooling (years)' can capture information from 'Expected Years of Schooling. female (years)' and 'Expected Years of Schooling. male (years)'

This leaves us with 7 remaining variables to consider further and test.

```
candidates = [
        'Crude Birth Rate (births per 1,000 population)',
        'Coefficient of human inequality',
        'Total Fertility Rate (live births per woman)',
        'Adolescent Birth Rate (births per 1,000 women ages 15-19)',
        'Expected Years of Schooling (years)',
        'Median Age, as of 1 July (years)',
```

```
        "Net Reproduction Rate (surviving daughters per woman)"
]
```

A.4.2 Construct the model

Construct a linear regression model using the found candidates with the target variable:

```
data_train = data_train[
        candidates + [target_variable]
].dropna()
data_test = data_test[
        candidates + [target_variable]
].dropna()

X_train = data_train[candidates]
y_train = data_train[target_variable]

model = LinearRegression()
model.fit(X_train, y_train)

r_squared = model.score(X_train, y_train)
coefficients = model.coef_
intercept = model.intercept_

print(f'R-squared: {r_squared}')
print(f'Coefficients: {coefficients}')
print(f'Intercept: {intercept}')
```

A.4.3 Predict the test set

Predict the target variable using the test data and computing the **mean squared error** and **correlation coefficient**:

```
X_test = data_test[candidates]
y_test = data_test[target_variable]
y_pred = model.predict(X_test)

mse = mean_squared_error(y_test, y_pred)
correlation, _ = pearsonr(y_pred, y_test)

print(f'Mean squared error: {mse}')
print(f'Correlation: {correlation}')
```