assignment-4

October 1, 2024

1 Problem 1: Split the data

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

[299]:
```
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
)
```

2 Problem 2: Single variable linear regression model

2.0.1 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:

[300]:
```
target_variable = \
        'Life Expectancy at Birth, both sexes (years)'

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

```
[300]: Life Expectancy at Birth, both sexes (years)
1.000000
Human Development Index (value)
0.918341
Crude Birth Rate (births per 1,000 population)
0.864138
Coefficient of human inequality
0.849600
Total Fertility Rate (live births per woman)
0.838654
Expected Years of Schooling, female (years)
0.814086
Adolescent Birth Rate (births per 1,000 women ages 15-19)
0.799662
Expected Years of Schooling (years)
0.799646
Median Age, as of 1 July (years)
0.797353
Expected Years of Schooling, male (years)
0.778834
Net Reproduction Rate (surviving daughters per woman)
0.777402
Mean Years of Schooling, female (years)
0.749029
Mean Years of Schooling (years)
0.743001
Mean Years of Schooling, male (years)
0.728092
Rate of Natural Change (per 1,000 population)
0.714862
Population with at least some secondary education, female (% ages 25 and older)
0.691909
Inequality in eduation
0.678548
Population with at least some secondary education, male (% ages 25 and older)
0.656120
Gross National Income Per Capita (2017 PPP$)
0.651471
Gender Development Index (value)
0.609221
Material footprint per capita (tonnes)
0.594345
Crude Death Rate (deaths per 1,000 population)
0.565175
Carbon dioxide emissions per capita (production) (tonnes)
0.457530
Inequality in income
```

0.430711
Population Annual Doubling Time (years)
0.417656
Sex Ratio at Birth (males per 100 female births)
0.402793
Share of seats in parliament, male (% held by men)
0.284705
Share of seats in parliament, female (% held by women)
0.284705
Population Growth Rate (percentage)
0.266573
Labour force participation rate, female (% ages 15 and older)
0.252507
Year
0.238989
Population Density, as of 1 July (persons per square km)
0.196511
Labour force participation rate, male (% ages 15 and older)
0.189015
Net Number of Migrants (thousands)
0.160673
Net Migration Rate (per 1,000 population)
0.150976
Births by women aged 15 to 19 (thousands)
0.140622
Population Sex Ratio, as of 1 July (males per 100 females)
0.120724
Natural Change, Births minus Deaths (thousands)
0.104400
Population Change (thousands)
0.086739
Births (thousands)
0.069438
Live births Surviving to Age 1 (thousands)
0.064848
Mean Age Childbearing (years)
0.036044
Female Population, as of 1 July (thousands)
0.026446
Total Population, as of 1 January (thousands)
0.025933
Total Population, as of 1 July (thousands)
0.025304
Male Population, as of 1 July (thousands)
0.024213
Female Deaths (thousands)
0.011025

```
Total Deaths (thousands)
0.008766
Male Deaths (thousands)
0.006861
Country
0.000433
Name: Life Expectancy at Birth, both sexes (years), dtype: float64
```

[301]:
```
candidate_variable = correlations.index[1]
print(
        f'Candidate variable: "{candidate_variable}"'
)
```

```
Candidate variable: "Human Development Index (value)"
```

2.0.2 Constructing the model

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

[302]:
```
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)
```

[302]:
```
LinearRegression()
```

Computing the metrics of the model:

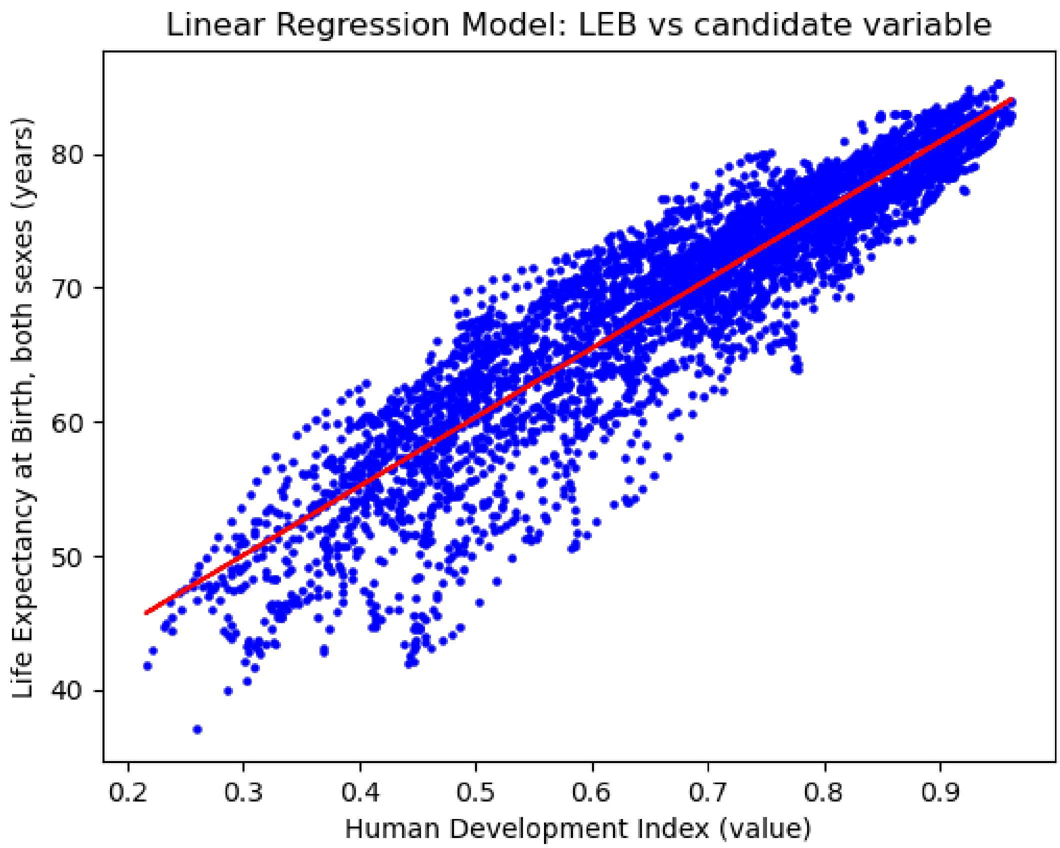
[303]:
```
r_squared = model.score(X_train, y_train)
coefficients = model.coef_
intercept = model.intercept_

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

```
R-squared: 0.8433493090941087
Coefficients: [51.42339338]
Intercept: 34.60462419807184
```

Plot the linear regression model:

[304]:
```
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"
)
```



2.0.3 Predict the test set

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

[305]:
```
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}')
```

```
Mean squared error: 12.519251362188522
Correlation: 0.920387001630666
```

3 Problem 3: Non-linear relationship

3.0.1 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:

[306]:
```
data_train = data_train.drop(
        columns=[candidate_variable]
)

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

[306]:
```
Life Expectancy at Birth, both sexes (years)
1.000000
Gross National Income Per Capita (2017 PPP$)
0.864828
Median Age, as of 1 July (years)
0.863765
Crude Birth Rate (births per 1,000 population)
0.848640
Expected Years of Schooling, female (years)
0.834567
Coefficient of human inequality
0.828904
Expected Years of Schooling (years)
0.819759
Total Fertility Rate (live births per woman)
0.816936
```

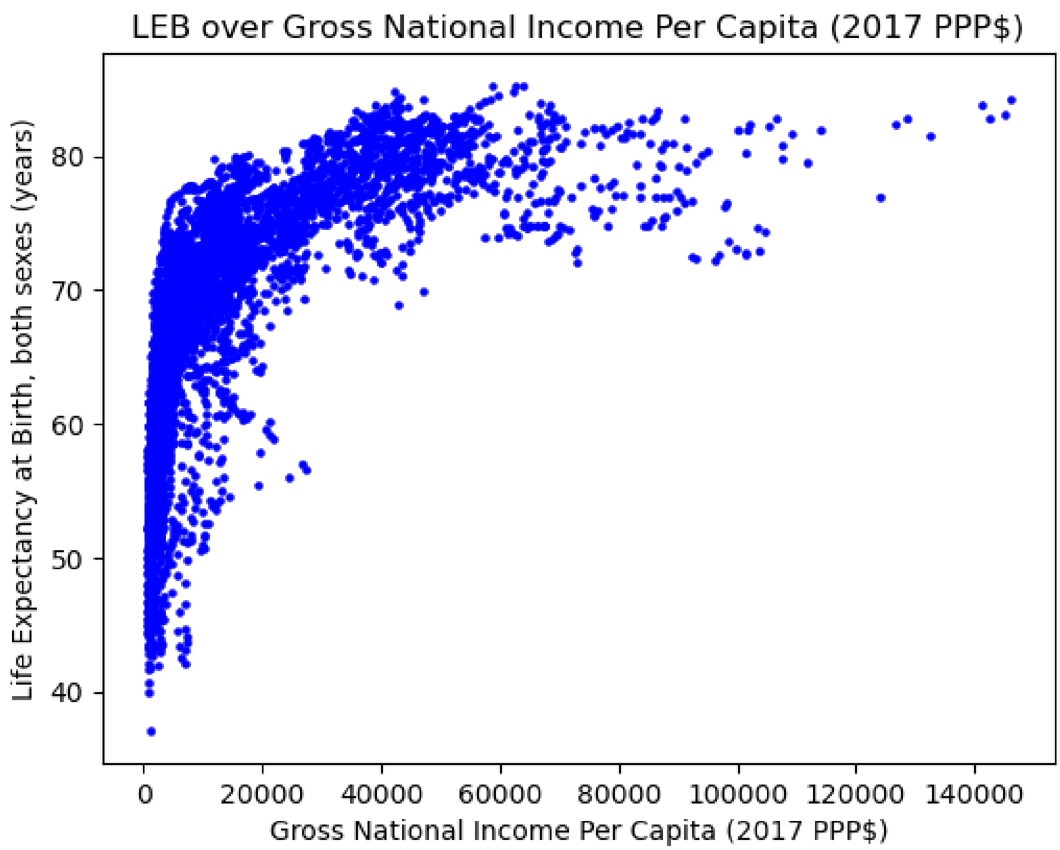
Adolescent Birth Rate (births per 1,000 women ages 15-19)
0.811920
Expected Years of Schooling, male (years)
0.806563
Material footprint per capita (tonnes)
0.789033
Net Reproduction Rate (surviving daughters per woman)
0.784511
Carbon dioxide emissions per capita (production) (tonnes)
0.762387
Rate of Natural Change (per 1,000 population)
0.746721
Mean Years of Schooling, female (years)
0.745441
Mean Years of Schooling (years)
0.742560
Mean Years of Schooling, male (years)
0.731290
Population with at least some secondary education, female (% ages 25 and older)
0.694487
Inequality in eduation
0.654956
Population with at least some secondary education, male (% ages 25 and older)
0.653522
Gender Development Index (value)
0.605025
Population Annual Doubling Time (years)
0.490936
Births by women aged 15 to 19 (thousands)
0.488655
Population Growth Rate (percentage)
0.485993
Sex Ratio at Birth (males per 100 female births)
0.437146
Natural Change, Births minus Deaths (thousands)
0.428493
Inequality in income
0.408609
Crude Death Rate (deaths per 1,000 population)
0.395167
Net Migration Rate (per 1,000 population)
0.378329
Net Number of Migrants (thousands)
0.371957
Share of seats in parliament, male (% held by men)
0.327508
Share of seats in parliament, female (% held by women)

```
0.327508
Population Density, as of 1 July (persons per square km)
0.297218
Population Change (thousands)
0.285238
Births (thousands)
0.274848
Live births Surviving to Age 1 (thousands)
0.264739
Labour force participation rate, male (% ages 15 and older)
0.231725
Year
0.220283
Labour force participation rate, female (% ages 15 and older)
0.193560
Female Deaths (thousands)
0.139715
Total Deaths (thousands)
0.138426
Male Deaths (thousands)
0.137720
Mean Age Childbearing (years)
0.108880
Female Population, as of 1 July (thousands)
0.051371
Total Population, as of 1 July (thousands)
0.047173
Total Population, as of 1 January (thousands)
0.045615
Male Population, as of 1 July (thousands)
0.043831
Population Sex Ratio, as of 1 July (males per 100 females)
0.043340
Country
0.014885
Name: Life Expectancy at Birth, both sexes (years), dtype: float64
```

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

[307]:
```
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)

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



The relationship appears to be logarithmic.

3.0.2 Construct the model on the transformed scale

Applying the logarithmic transformation to the candidate variable:

[308]:
```
import numpy as np

X_train = data_train[[second_candidate_variables]]
y_train = data_train[target_variable]
```

```
log_X_train = np.log(X_train)

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

[308]: LinearRegression()

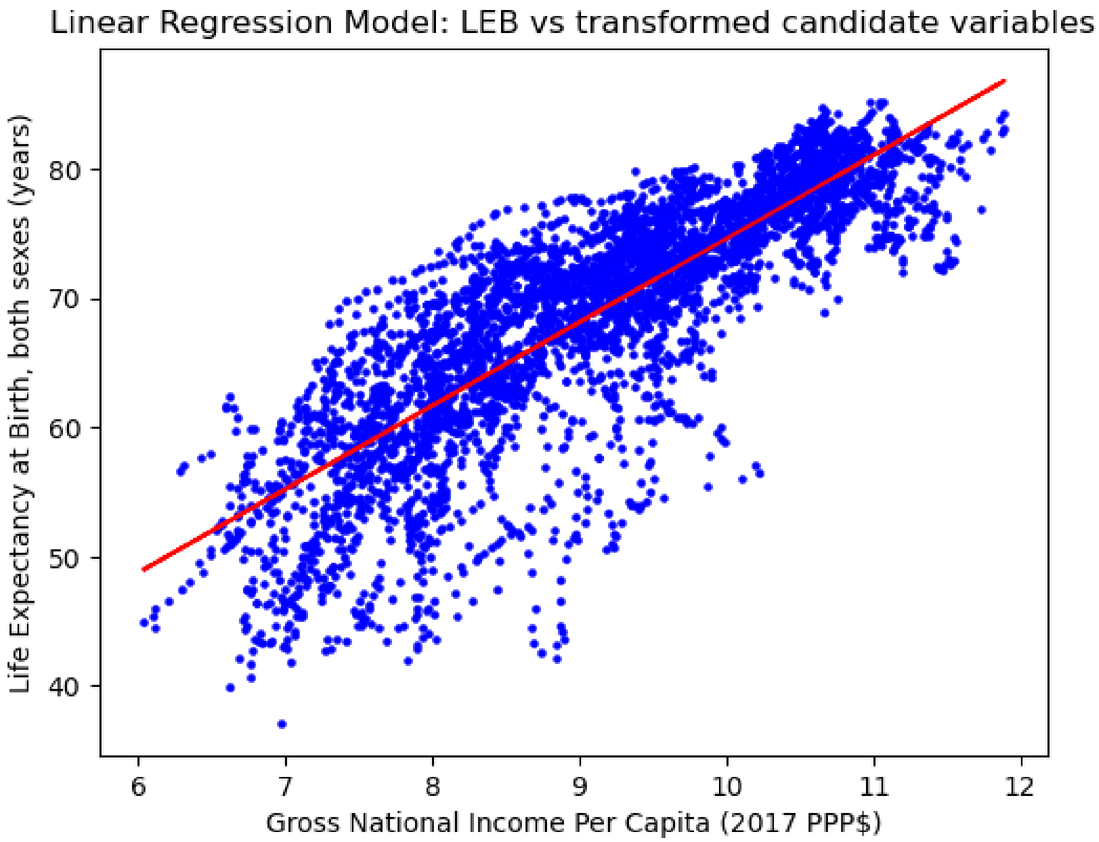
Computing the metrics of the model:

[309]:
```
r_squared = model.score(log_X_train, y_train)
coefficients = model.coef_
intercept = model.intercept_

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

```
R-squared: 0.6939267719160521
Coefficients: [6.46720934]
Intercept: 9.942320920335249
```

Plotting the linear regression model:

[310]:
```
plt.scatter(
        log_X_train, y_train,
        color='blue', s=5
)
plt.plot(
        log_X_train, model.predict(log_X_train),
        color='red'
)
plt.title(
        'Linear Regression Model: LEB vs transformed candidate variables'
)
plt.xlabel(second_candidate_variables)
plt.ylabel(target_variable)

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



3.0.3 Comparing the transformation

Computing the correlation coefficient before and after the transformation

[311]:

```
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}
''')
```

```
Original correlation:    0.6514708331957302
Transformed correlation: 0.833022671909986
```

4 Problem 4: Multiple linear regression model

4.0.1 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 8 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)

However, `Expected Years of Schooling (years)` is able to capture the information from `Expected Years of Schooling. female (years)` and `Expected Years of Schooling. male (years)`

This leaves us with 6 remaining variables to consider further and test.

[313]:
```
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)'
]
```

4.0.2 Construct the model

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

[317]:
```
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}')
```

```
R-squared: 0.8673778882429013
Coefficients: [-7.88494739e-01 -1.12419551e-01 -8.60456539e+00 -1.51153253e-02
  3.21929313e-01  8.46505738e-02  2.90244232e+01]
Intercept: 71.76617234571023
```

4.0.3 Predict the test set

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

[315]:
```
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}')
```

```
Mean squared error: 8.879386934439953
Correlation: 0.9363399256423098
```