DAT565/DIT407 Assignment 4

Saif Sayed
gussayedfa@student.gu.se

Gona Ibrahim Abdulrahman
gusibrgo@student.gu.se

2024-05-08

This is a report for assignment 4 for the course *Introduction to Data Science & AI* from Chalmers and Gothenburg University.

Problem 1: Splitting the data

In problem 1, we utilized the train_test_split function provided by the *scikit−learn* library. We start by separating the independent variables, also known as features, from the target variable. In this particular example, the target variable is "Life Expectancy at Birth, both sexes (years)" and we store it in the variable y. The remaining columns in the dataset are deemed as features, and we store them in the variable X.

To split the data into training and testing sets, we employ the train_test_split function. By utilizing this function, we can determine the proportion of the dataset allocated for testing through the test_size parameter. For instance, we set the test_size to 0.3 implying that 30% of the data will be reserved for testing. To ensure reproducibility and consistency across different runs, we also specify a specific value for the random_state parameter, such as 42. This ensures that the same random split is generated each time the code is executed, allowing for reliable comparisons and evaluations.

Our source code can be found in Appendix A of this document.

Problem 2: Single Variable Model

(A) Here, we calculate the pearson coefficient to identify the variable which has the strongest correlation with the target variable (LEB). The identified variable was 'Human Development Index' with a pearsonr correlation coefficient of 0.92 (2sf).

(B) We developed a linear model for the correlation and found out the coefficient of the model, coefficient of determination and the intercept. And finally plotted a scatter plot with the regression line over it for visualisation (see **Figure 1**). **Table 1** shows our obtained results.

Table 1: Obtained results from our linear model.

Coefficient of determination (2sf)	Coefficient of model (2sf)	Intercept (2sf)
0.84	51.5	34.5

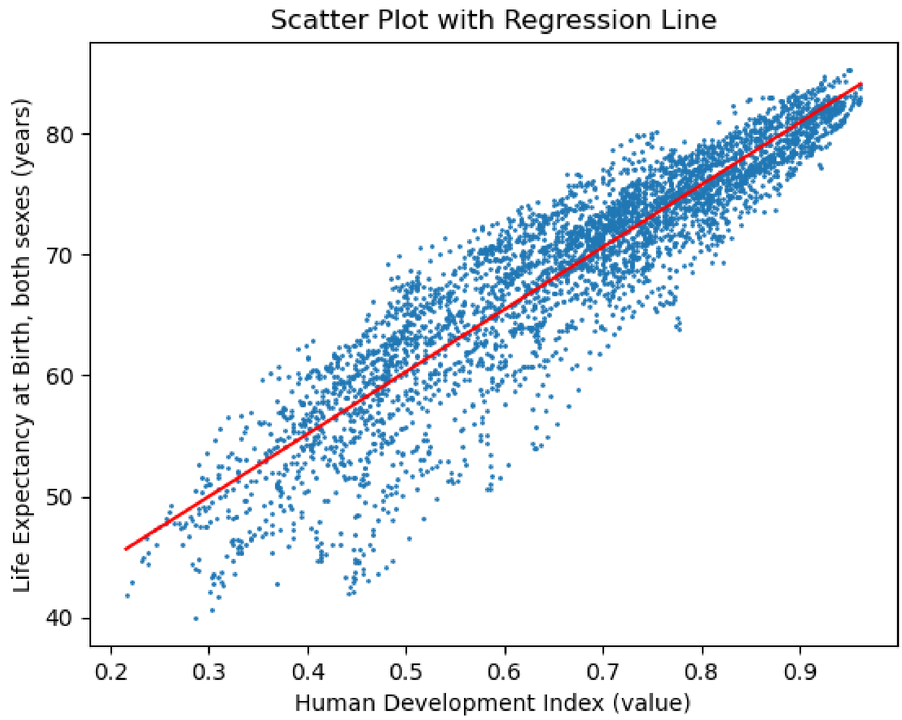

Figure 1: Scatter Plot with regression.

(C) Finally, we predicted the life expectancy for the testing split of our data. Then we calculated the correlation between the predicted and actual values of life expectancy, along with the mean square error. **Table 2** shows our obtained results.

Table 2: Results between predicted and actual LEB.

Pearsonr correlation (2sf)	Mean square error
0.92	12.4

(D) The Human Development Index (HDI) determines the development of countries to ensure the well-being of humans. It takes into consideration a country's economy, education, and healthcare. A higher Human Development Index would mean a better healthcare and education system for its citizens. Thus, humans would have access to quality healthcare that can address health issues, and education that can raise self-awareness within themselves. A higher economy would also mean a higher income and a better life.

Problem 3: Non Linear Relationship

In problem 3, we used the spearmanr correlation coefficient to deduce a candidate variable that represents a non-linear relation. We removed the identified variable 'Human Development Index' from the previous task since we already know it exhibits a linear relationship. Unlike the pearsonr correlation coefficient, Spearman's coefficient is designed to capture monotonic relationships, regardless of their linearity. Based on our data analysis, we identified the variable "Gross National Income Per Capita (2017 PPP$)" as having the highest Spearman's correlation coefficient of 0.87 (2sf). To visualize the correlation between the identified variable and the variable LEB, we created a scatter plot (see **Figure 2**). Upon inspecting the scatter plot, we observed that the relationship between the variables could be represented by a linear pattern after applying a logarithmic transformation to the identified variable (see **Figure 3**). Consequently, we applied the logarithmic function to transform the data. Before the transformation, the Pearson correlation coefficient between the identified variable and LEB was calculated to be 0.65 (2sf). After applying the logarithmic transformation to the identified variable and re-evaluating the correlation, we obtained a Pearson correlation coefficient of 0.83 (2sf).

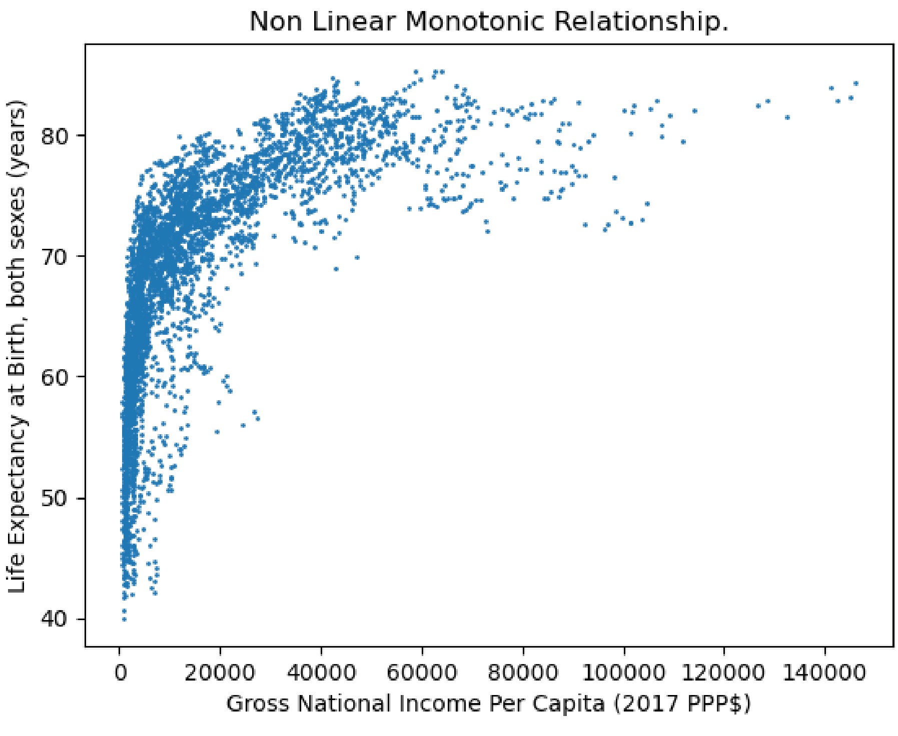

Figure 2: Non Linear Monotonic Relationship.

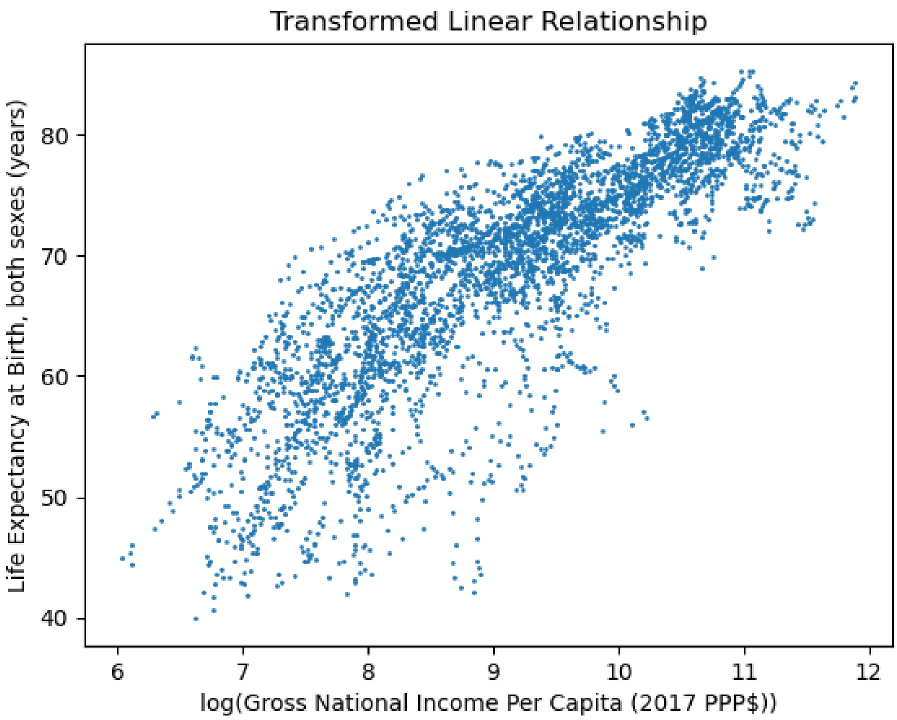

Figure 3: Transformed Linear Relationship.

Problem 4: Multiple Linear Regression

To identify a subset of variables that can potentially improve the linear model compared to the one in problem 2, we followed a systematic approach. First, we checked the Pearson correlation coefficient between each variable and the target variable. Based on the correlation coefficients, we restricted our selection to the variables exhibiting high correlation (> 0.75) with the target variable. These variables have a stronger linear relationship and are more likely to contribute to a better linear model. Finally, we selected a smaller subset of identified variables that are likely to have a meaningful impact on the target variable based on our understanding of the dataset. We calculated the necessary metrics and expanded our set of selected variables if our model didn't outperform the single variable model from problem 2.

The selected variables are:

- Adolescent Birth Rate (births per 1,000 women ages 15-19)
- Expected Years of Schooling (years)
- Median Age, as of 1 July (years)
- Crude Birth Rate (births per 1,000 population)

- Total Fertility Rate (live births per woman)
- Net Reproduction Rate (surviving daughters per woman)

In our multiple linear regression model, we obtained the following results: a coefficient of determination of 0.87 (2sf), an intercept of 58.15 (2sf), and coefficients for the model variables as follows: -0.02, 0.21, 0.40, -0.47, -10.16, 30.52 (2sf). The mean square error of our model was found to be 11.68 (2sf), and the Pearson correlation coefficient between the predicted and actual life expectancy at birth (LEB) was calculated to be 0.93 (in 2sf).
We observed that the mean square error in our multiple linear regression model is slightly lower compared to the mean square error in the single variable model from problem 2. Additionally, we obtained a higher coefficient of determination, indicating a greater proportion of variance explained by our model. These findings suggest that our multiple linear regression model performs slightly better in capturing the relationships between the variables and the target variable.

Appendix

A Python code

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

'''
Problem 1: Splitting the data
'''
data = pd.read_csv("life_expectancy.csv", encoding= "utf-8")

# Split the data into features (X) and target variable (y)
X = data.drop("Life Expectancy at Birth, both sexes (years)", axis=1)
y = data["Life Expectancy at Birth, both sexes (years)"]

# Perform train-test split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)

# Print the shapes of the resulting datasets to verify the split
```

```python
print("Training set shape:", X_train.shape, y_train.shape)
print("Testing set shape:", X_test.shape, y_test.shape)
'''
Problem 2: Single Variable Model
'''
# A.

# Exclude non-numeric columns from correlation calculation
numeric_columns = X_train.select_dtypes(include=[float, int]).columns
numeric_data = X_train[numeric_columns]

# Handling missing values and replacing them with the mean of the corresponding column
numeric_data = numeric_data.replace([np.inf, -np.inf], np.nan)
numeric_data = numeric_data.fillna(numeric_data.mean())

y_train = y_train.replace([np.inf, -np.inf], np.nan)
y_train = y_train.fillna(y_train.mean())

strongest_variable = ""
pearsonr_correlation = 0

for column in numeric_data.columns:
    corr= pearsonr(y_train, numeric_data[column])[0]
    if (corr > pearsonr_correlation):
        pearsonr_correlation = corr
        strongest_variable = column

# Print the results
print("Variable with the strongest correlation:", strongest_variable)
print("Correlation coefficient:", pearsonr_correlation)

# B.

target_variable = "Life Expectancy at Birth, both sexes (years)"
X = numeric_data[["Human Development Index (value)"]]
y = y_train

# Initialize and fit the linear regression model
model = LinearRegression().fit(X, y)

```

```
actual_LEB = y_train
predicted_LEB = model.predict(X)

determination_coefficient =  r2_score(actual_LEB, predicted_LEB)
model_coefficient = model.coef_
intercept = model.intercept_

print("Coefficient of determination:", determination_coefficient)
print("Coefficient of model:", model_coefficient)
print("Intercept:", intercept)

# Create a scatter plot
plt.scatter(X, y, s=1)

# Plot the regression line
xfit = np.linspace(X["Human Development Index (value)"].min(), X["Human Development Index (value)"].max(), 100)
yfit = model.predict(xfit.reshape(-1, 1))
plt.plot(xfit, yfit, color='red')

# Set plot labels
plt.xlabel("Human Development Index (value)")
plt.ylabel(target_variable)
plt.title("Scatter Plot with Regression Line")

# C.

test_numeric_columns = X_test.select_dtypes(include=[float, int]).columns
test_numeric_data = X_test[test_numeric_columns]
test_numeric_data = test_numeric_data.replace([np.inf, -np.inf], np.nan)
test_numeric_data = test_numeric_data.fillna(test_numeric_data.mean())

y_test = y_test.replace([np.inf, -np.inf], np.nan)
y_test = y_test.fillna(y_test.mean())

X_test= test_numeric_data[["Human Development Index (value)"]]
y_test= y_test
X_test = X_test.to_numpy()

test_y_pred= model.predict(X_test.reshape(-1, 1))
pred_target_coefficient = pearsonr(y_test, test_y_pred)[0]
mean_squared_error= mean_squared_error(y_test,
```

```
    test_y_pred)

print("The Pearsonr correlation coefficient between the predicted LEB and true LEB is:", pred_target_coefficient)
print("The mean squared error between the predicted LEB and true LEB is:", mean_squared_error)
'''
Problem 3: Non-linear relationship
'''
numeric_data = numeric_data.drop("Human Development Index (value)", axis=1)

non_linear_variable = ""
spearmanr_correlation = 0
for column in numeric_data.columns:
    corr= spearmanr(y_train, numeric_data[column])[0]
    if (corr > spearmanr_correlation):
        spearmanr_correlation = corr
        non_linear_variable = column

print("The non linear variable is:", non_linear_variable)
print("Spearmanr Correlation:", spearmanr_correlation)

x = numeric_data[[non_linear_variable]]

plt.scatter(x, y, s=1)
plt.xlabel(non_linear_variable)
plt.ylabel(target_variable)
plt.title("Non Linear Monotonic Relationship")

pearson_coefficient_before = pearsonr(y, x[non_linear_variable])[0]
print("Pearsonr Correlation Coefficient before:", pearson_coefficient_before)

logarithmic_x = np.log(x)

plt.scatter(logarithmic_x, y, s=1)
plt.xlabel("log(" + non_linear_variable + ")")
plt.ylabel(target_variable)
plt.title("Transformed Linear Relationship")

pearson_coefficient_after = pearsonr(y, logarithmic_x[non_linear_variable])[0]
print("Pearsonr Correlation Coefficient after:", pearson_coefficient_after)

'''
```

```
Problem 4: Multiple Linear Regression
'''
for column in numeric_data.columns:
    corr= pearsonr(y_train, numeric_data[column])[0]
    print(column, ":",corr)

selected_columns= ["Adolescent Birth Rate (births per 1,000 women ages 15-19)", "Expected Years of Schooling (years)", "Median Age, as of 1 July (years)", "Crude Birth Rate (births per 1,000 population)", "Total Fertility Rate (live births per woman)", "Net Reproduction Rate (surviving daughters per woman)"]
X = numeric_data[selected_columns]
y = y_train

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

actual_LEB = y_train
predicted_LEB = model.predict(X)

determination_coefficient =  r2_score(actual_LEB, predicted_LEB)
model_coefficient = model.coef_
intercept = model.intercept_

print("Coefficient of determination:", determination_coefficient)
print("Coefficient of model:", model_coefficient)
print("Intercept:", intercept)

X_test= test_numeric_data[selected_columns].values
y_test= y_test.values

test_y_pred= model.predict(X_test.reshape(-1, 6))
pred_target_coefficient = pearsonr(y_test, test_y_pred)[0]
print("The Pearsonr correlation coefficient between the predicted LEB and true LEB is:", pred_target_coefficient)

'''to run the lines below, the previous lines where the "mean_squared_error" function was called in problem 2 needs to be commented out'''
#mse = mean_squared_error(y_test, test_y_pred)
#print("The mean squared error between the predicted LEB and true LEB is:", mse)
```