DAT565/DIT407 Assignment 4

Ola Bratt
ola.bratt@gmail.com

Patrick Attimont
patrickattimont@gmail.com

2024-02-xx

This paper is addressing the assignment 3 study queries within the *Introduction to Data Science & AI* course, DIT407 at the University of Gothenburg and DAT565 at Chalmers. The main source of information for this project is derived from the lectures and Skiena [2]. Assignment 4 is about correlation and linear regression.

Problem 1: Splitting the data

The dataset is large enough to be separated into a train and a test set. We use the function `train_test_split` with a test size of 0.2.

Problem 2: Single-variable model

To identify the variable with the strongest linear relationship with the target variable, we use the `corr` function specifying the Pearson method to get a correlation matrix between all the variables (Fig 1), and take the column corresponding to the target variable.

The variable with the highest absolute pearson coefficient is the Human development index, with a value of 0.92. Fig 2 shows the linear regression applied on the test set, using the model built on the train set. The model has a slope of 48.2, an intercept of 37.3, and a determination coefficient of 0.86. These values obviously slightly change depending on the seed used to split the data. The mean squared error between the test and predicted sets is 9.45.

The human development index measures a country's social and economic development. As specified on UNDP's website ([1]), it is calculated based on the following factors: life expectancy, education, and standard of living. Thus the strong relationship between the two variables comes from the fact that life expectancy at birth is used to calculate the human development index.

Problem 3: Non-linear relationship

To explore variables with non-linear relationships, we initially assess Spearman correlation coefficients, Figure 3. Initially, we eliminate the variable exhibiting the strongest Pearson correlation, namely 'Human Development Index (value)'. Subsequently, we examine the Spearman correlation.

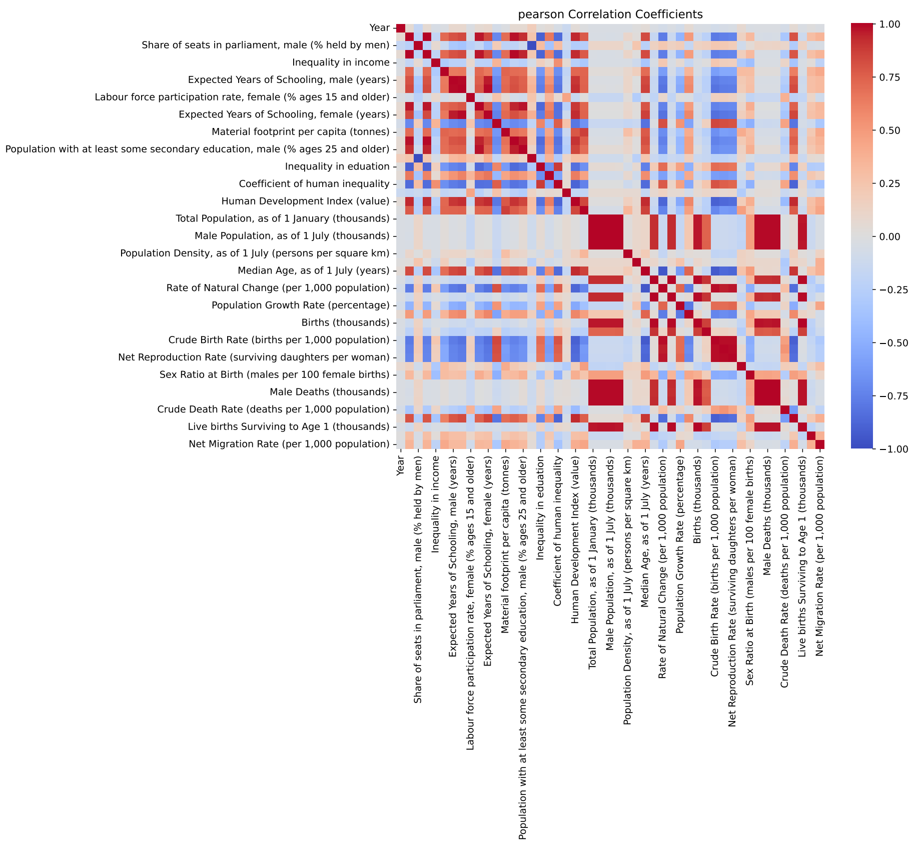


Figure 1: Correlation Pearson

Our analysis reveals that 'Median Age, as of 1 July (years)' exhibits the most robust Spearman correlation. Subsequently, we develop a linear model using this variable to determine the mean squared error, which amounts to 13.58, Figure 5a.

Next, we apply transformations to the variable using logarithmic, square root, and reciprocal functions, respectively Figure 4. For each transformation, we train a linear model to ascertain the resulting mean squared error. The logarithmic transformation yields the lowest mean squared error, measuring 11.52. Figure 5b.

Before transformation, the Pearson correlation registers at 0.898, while after transformation, it increases marginally to 0.913.

Problem 4: Mulitple linear regression

We employ Spearman correlation analysis to identify variables with strong correlations. Utilizing a predefined threshold, we pinpoint variables with significant impact. Through experimentation with various thresholds, we determine that a threshold of 0.85 yields favorable results without excessive variable inclusion.

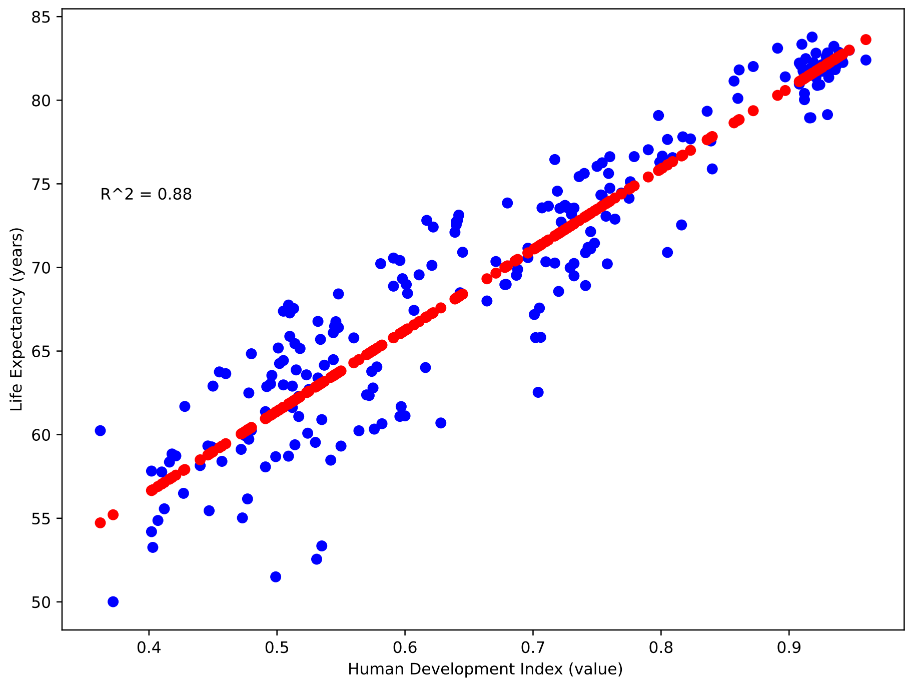


Figure 2: Linear Regression Human Development Index (value)

The identified variables include: Expected Years of Schooling, female (years), Coefficient of human inequality, Gross National Income Per Capita (2017), Median Age as of 1 July (years), Rate of Natural Change (per 1,000 population), Crude Birth Rate (births per 1,000 population), Total Fertility Rate (live births per woman), and Net Reproduction Rate (surviving daughters per woman).

Conducting a multiple linear regression analysis results in a mean squared error of 2.03. The coefficients for the model are as follows: [19, -571, 243338, 61, 1.7, -2.1, 1.9, 3].

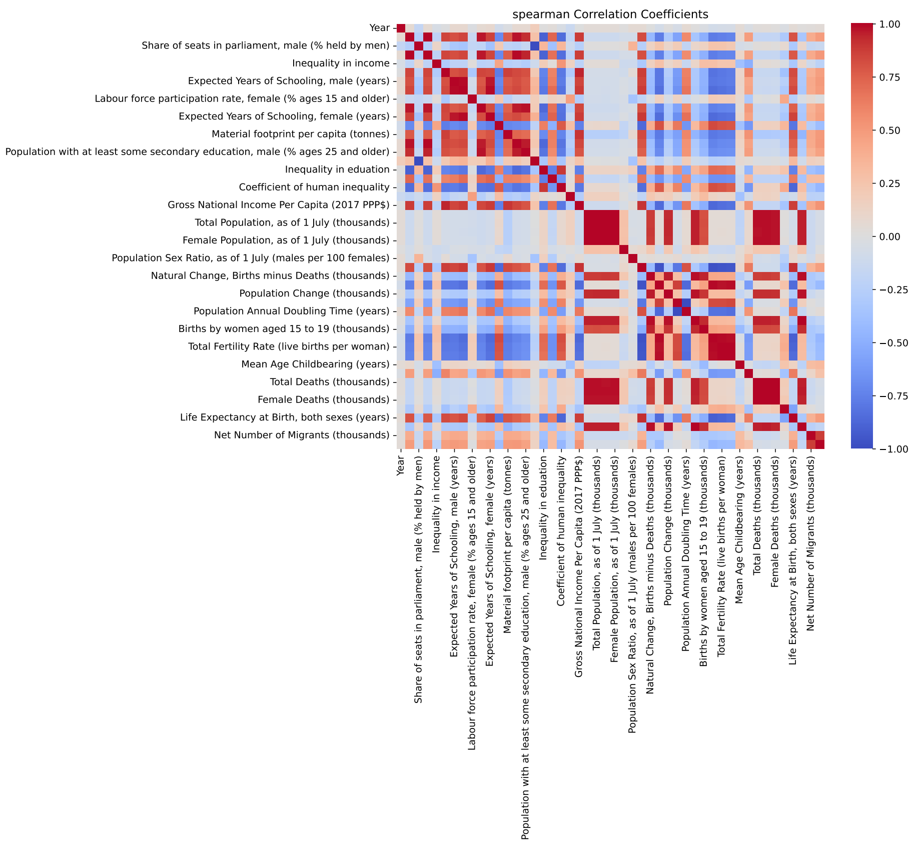


Figure 3: Correlation Spearman

References

[1] Human development reports. *Human Development Index (HDI)*. Retrieved 2024-02-14. 2023. URL: `https://hdr.undp.org/data-center/human-development-index#/indicies/HDI`.

[2] Steven S Skiena. *The Data Science Design Manual*. Retrieved 2024-01-20. 2024. URL: `https://ebookcentral.proquest.com/lib/gu/detail.action?docID=6312797`.

Appendix: Source Code

```python
from matplotlib import pyplot
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
import seaborn as sns
from sklearn.metrics import mean_squared_error
from sklearn.metrics import r2_score

```

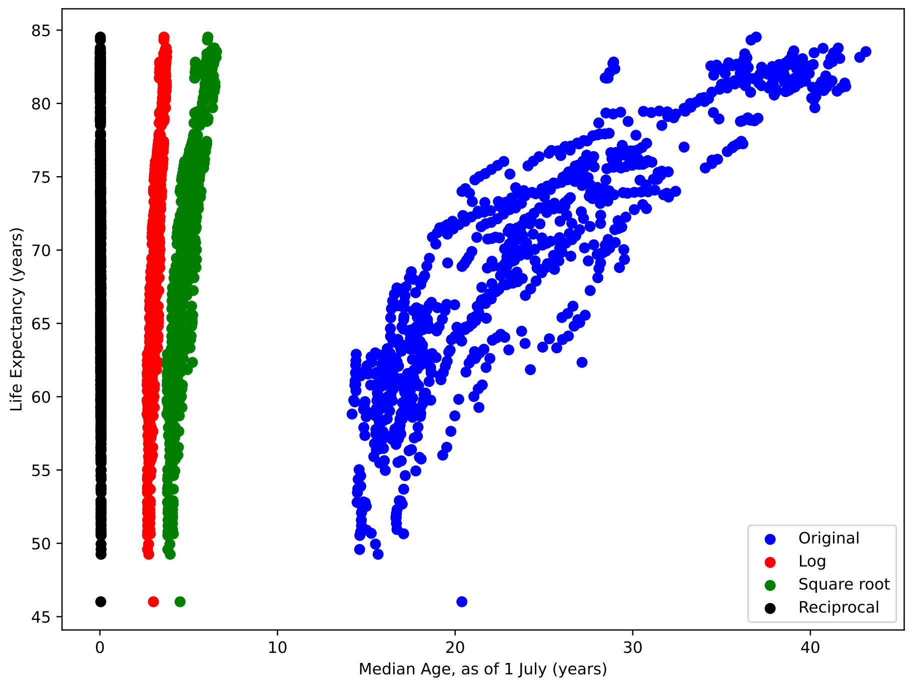
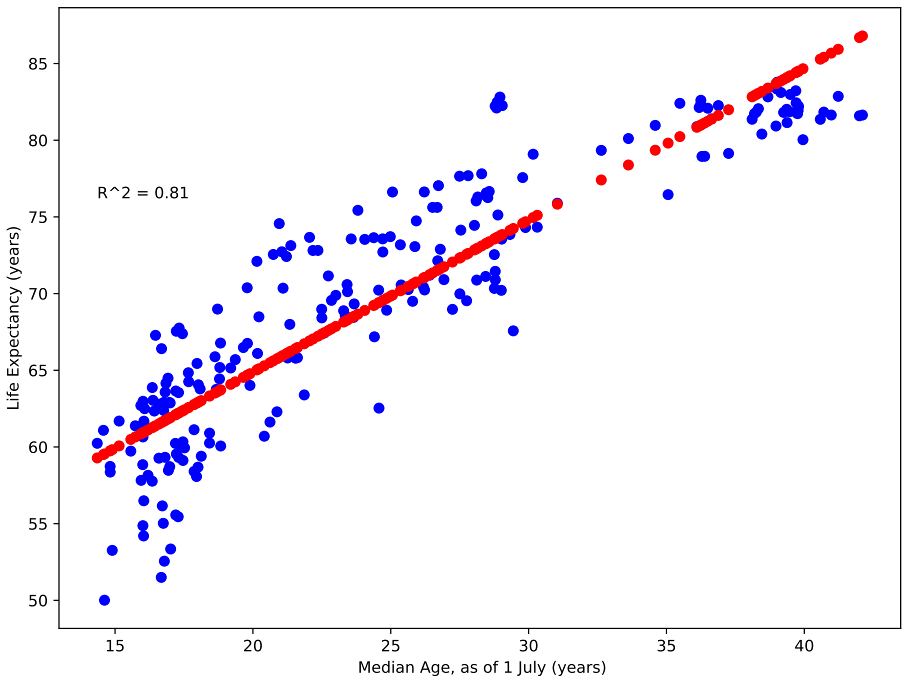
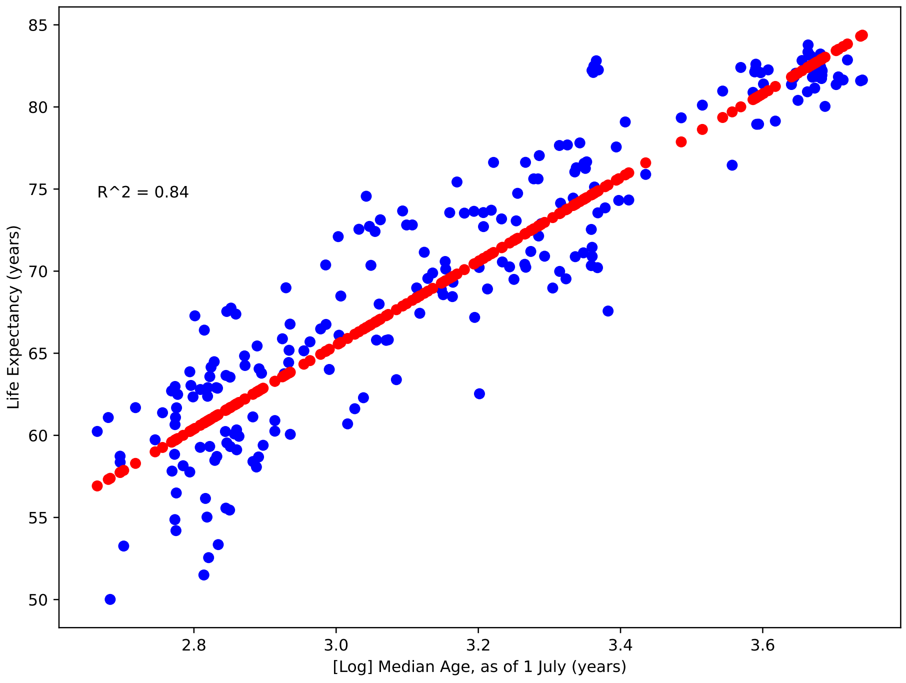


Figure 4: Linear transformation

```python
def calculate_correlation(data, variable, method):
    # Compute Pearson correlation coefficients
    correlation_matrix = data.corr(method = method)

    # Extract correlation coefficients of the target variable (life
        ↪ expectancy)
    correlation_with_life_expectancy = correlation_matrix[variable]
    # Remove the target variable from the correlation coefficients
    correlation_without_life_expectancy =
        ↪ correlation_with_life_expectancy.drop(variable)

    # Find the variable with the highest absolute correlation
        ↪ coefficient
    strongest_correlation_variable =
        ↪ correlation_without_life_expectancy.abs().idxmax()
    strongest_correlation_coefficient =
        ↪ correlation_without_life_expectancy.abs().max()

    print(f"The variable '{strongest_correlation_variable}' has the
        ↪  strongest " + method + f" correlation with a
        ↪ coefficient of {strongest_correlation_coefficient:.2f}."
        ↪ )

    fig, ax = pyplot.subplots(figsize=(10, 8))
    sns.heatmap(correlation_matrix, annot=False, cmap='coolwarm')
    ax.set_title(method + ' Correlation Coefficients')
    fig.savefig(method + "_correlation.pdf", bbox_inches='tight')

    return strongest_correlation_variable, correlation_matrix

```



(a) Linear Regression Median Age (original)



(b) Linear Regression Median Age (log)

Figure 5: Linear Regression Median Age

```python

def train_linear_regression_model(X_train, X_test, y_train, y_test, variables, prefix = ''):
    # Train a linear regression model using the variable with the strongest correlation
    model = LinearRegression().fit(X_train, y_train)
    # Make predictions
    y_pred = model.predict(X_test)
    r2 = r2_score(y_test, y_pred)
    _, rows = X_test.shape
    if rows == 1:
        fig, ax = pyplot.subplots(figsize=(8, 6), layout='constrained')
        ax.scatter(X_test, y_test, color='blue')
        ax.scatter(X_test, y_pred, color='red')
        ax.set_xlabel(prefix + " " + variables)
        ax.set_ylabel('Life Expectancy (years)')
        ax.text(0.1, 0.7, f'R^2 = {r2:.2f}', ha='center', va='center', transform=ax.transAxes)
        filename = prefix + "_linear_regression_" + variables + ".pdf"
        filename = filename.replace(' ', '_').lower()
        fig.savefig(filename, bbox_inches='tight')



    mse = mean_squared_error(y_test, y_pred)
    print("Trained model with the following variables " + prefix + ": ", variables)
    print(f"The mean squared error for is {mse:.2f}.")

def transform_variable(X_train, y_train, correlation_variable):
    pd.options.mode.chained_assignment = None

    X_train_selected = X_train[[correlation_variable]]

    X_train_selected['log'] = np.log(X_train[[correlation_variable]])
    X_train_selected['sqrt'] = np.sqrt(X_train[[correlation_variable]])
    X_train_selected['reciprocal'] = 1/(X_train[[correlation_variable]])

    fig, ax = pyplot.subplots(figsize=(8, 6), layout='constrained')
    ax.scatter(X_train_selected.iloc[:, 0], y_train, color='blue', label='Original')
    ax.scatter(X_train_selected['log'], y_train, color='red', label='Log')
    ax.scatter(X_train_selected['sqrt'], y_train, color='green', label='Square root')
    ax.scatter(X_train_selected['reciprocal'], y_train, color='black', label='Reciprocal')
    ax.set_ylabel('Life Expectancy (years)')
    ax.set_xlabel(correlation_variable)

    ax.legend()
    fig.savefig("linear_transformation.pdf", bbox_inches='tight')

file_path = "../life_expectancy.csv"
life_expectancy = pd.read_csv(file_path, sep=',',).dropna()
LEB = 'Life Expectancy at Birth, both sexes (years)'
life_expectancy.set_index('Country', inplace=True)

```

```python
life_expectancy_train, life_expectancy_test = train_test_split(life_expectancy, test_size=0.2)

X_train = life_expectancy_train.drop(LEB, axis=1)
X_test =  life_expectancy_test.drop(LEB, axis=1)
y_train = life_expectancy_train[LEB]
y_test = life_expectancy_test[LEB]



strongest_pearson_correlation_variable, correlation_pearson = calculate_correlation(life_expectancy_train, LEB, 'pearson')
train_linear_regression_model(X_train[[strongest_pearson_correlation_variable]], X_test[[strongest_pearson_correlation_variable]], y_train, y_test, strongest_pearson_correlation_variable)

strongest_spearman_correlation_variable, correlation_spearman = calculate_correlation(life_expectancy_train.drop(strongest_pearson_correlation_variable, axis=1), LEB, 'spearman')
train_linear_regression_model(X_train[[strongest_spearman_correlation_variable]], X_test[[strongest_spearman_correlation_variable]], y_train, y_test, strongest_spearman_correlation_variable)
transform_variable(X_train, y_train, strongest_spearman_correlation_variable)
train_linear_regression_model(np.log(X_train[[strongest_spearman_correlation_variable]]), np.log(X_test[[strongest_spearman_correlation_variable]]), y_train, y_test, strongest_spearman_correlation_variable, "[Log]")
train_linear_regression_model(np.sqrt(X_train[[strongest_spearman_correlation_variable]]), np.sqrt(X_test[[strongest_spearman_correlation_variable]]), y_train, y_test, strongest_spearman_correlation_variable, "[Sqrt]")
train_linear_regression_model(1/(X_train[[strongest_spearman_correlation_variable]]), 1/(X_test[[strongest_spearman_correlation_variable]]), y_train, y_test, strongest_spearman_correlation_variable, "[Reciprocal]")

threshold = 0.85
correlation_spearman_no_LEB = correlation_spearman.drop([LEB])

relevant_variables = correlation_spearman_no_LEB[abs(correlation_spearman_no_LEB['Life Expectancy at Birth, both sexes (years)']) > threshold].index.tolist()
train_linear_regression_model(X_train[relevant_variables], X_test[relevant_variables], y_train, y_test, relevant_variables)
```