DAT565/DIT407 Assignment 4

Connell Hagen
connellh@student.chalmers.se

Måns Redin
mansre@student.chalmers.se

2024-09-30

1 Problem 1: Splitting the data

The dataset was initially splitted into a 75-25 split, using the train_test_split function in the scikit-learn library in Python. The dataset was split into these ratios, because it was a fairly large one, and there was no reason for not using the standard split. The target variable of the dataset is the life expectancy at birth.

2 Problem 2: Single-variable model

2.1 Identifying the strongest correlating variable

To decide the variable that had the strongest correlation with the target variable, life expectancy at birth, the Pearson coefficient was determined for each variable. Equation 1 defines the Pearson coefficient.

$$\sum_{i=1}^{n} \frac{(x_i - \bar{x}) \cdot (y_i - \bar{y})}{s_x \cdot s_y} \tag{1}$$

In the Pearson coefficient the x and y are independent respective to the dependent variable. The s variables are the standard deviations for the variable, calculated on the dataset. The variables with the magnitude of the Pearson coefficients closest to one, are the ones with the strongest positive or negative correlation to the target variable. Table 2.1 shows the four variables with the magnitude of the Pearson coefficient closest to one.

Variable	Pearson coefficient
Human Development Index	0.919
Crude Birth Rate	-0.865
Coefficient of Human Inequality	-0.852
Total Fertility Rate	-0.840

Table 1: The Pearson coefficient of some of the variables with the highest correlation to the target variable

2.2 The linear model of the Human Development Index

A linear regression model was constructed using the Human Development Index as the independent variable and the Life Expectancy at Birth as the target variable. The coefficients of this model is shown in table 2.2. Figure 2.2 shows a scatter plot of the data points with the fitted regression line.

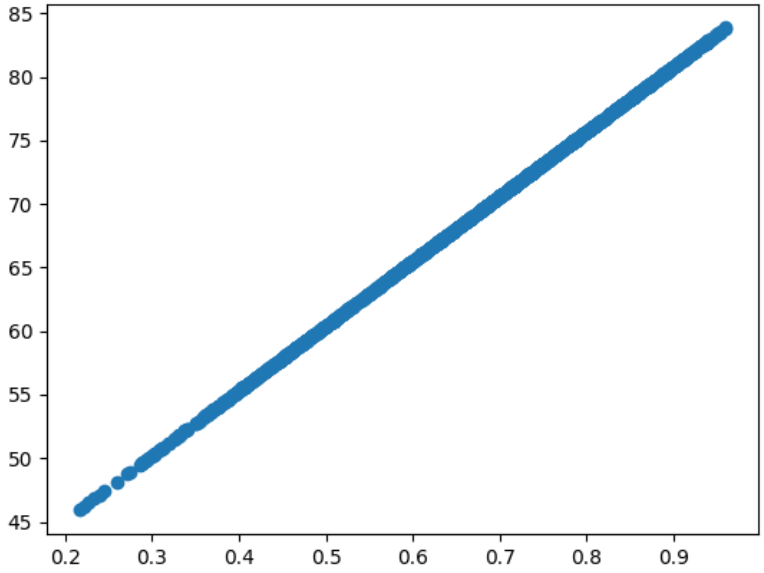

Figure 1: Predictions for each data point in the test set

Variable	Value
Slope	51.011
Intercept	34.900
Coefficient of Determination	0.844

Table 2: The constants for the linear model

2.3 Prediction

The test set was used to predict new values of the Life Expectancy at Birth. The correlation between the predicted values and the test values was determined by a coefficient of determination. This coefficient of determination and the mean squared error for the model is shown in table 2.3.

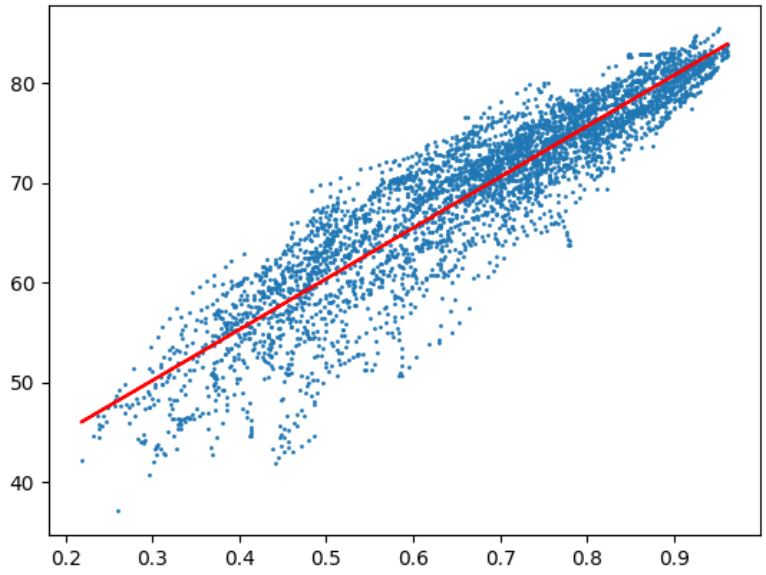

Figure 2: The test dataset and the linear regression model

Variable	Value
Mean squared error	13.730
Coefficient of Determination	0.844

Table 3: The Mean squared error and coefficient of determination between the predicted variable and the test dataset. Note that the Coefficient of Determination is 0.844 as before but it is another one rounded to the same value.

2.4 Human Development Index

The human development index is a indication on how well developed some key dimensions of human life is. Some factors that contribute to this index is, among many, a long and healthy life and the standard of living. The following article from UNDP describes it further. A link to their article is provided in the appendix. The strong correlation between Life Expectancy at Birth and the Human Development Index is best described by the amount of variables taken in consideration to create the independent variable. Generally, good health is a significant factor in living a longer life.

3 Problem 3: Non-linear relationship

The data was explored and a variable that was found to have a non-linear correlation between it and the target variable was Median Age. The relationship seemed to be power-law distributed and the final transformation executed was taking every value to the power of five. Figure 3 shows this relationship and model graphed. This was determined by trying many different values, and taking the one with the least mean squared error. The Pearson coefficient for the untransformed and the transformed data can be shown in table 3.

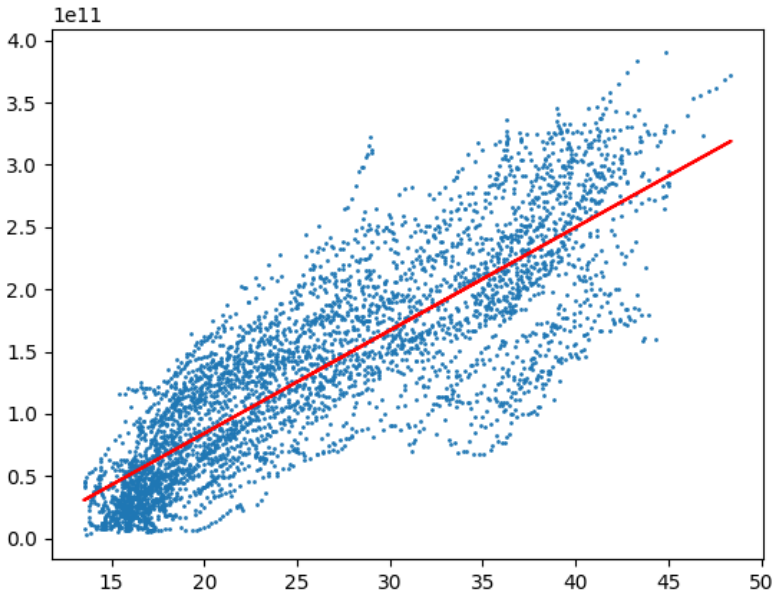

Figure 3: Median Age vs Life Expectancy at Birth to the 5th power

	Pearson coefficient
Transformed data	0.832
Untransformed data	0.799

Table 4: The Pearson coefficient for the un-transformed and transformed data of the Median Age.

4 Problem 4: Multiple linear regression

Excluding Human Development Index, we sought to find a small set of variables that could collectively predict the Life Expectancy better than HDI alone. We

started by taking the variable with the highest Pearson coefficient magnitude, Crude Birth Rate, and we put it in our set of variables. Then, we looped through every variable and determined whether adding it into the set, or excluding it from the set gave a better mean squared error. If it was better in the set, then we left it in the set permanently.

At this point we had a list of 16 variables, so we removed variables that were similar to others in the set already. This left us with 7 variables. We decided to remove all remaining variables that had a Pearson coefficient from earlier that had a magnitude less than 0.5. This left us with 4 remaining variables for our final model: Crude Birth Rate, Total Fertility Rate, Net Reproduction Rate, & Crude Death Rate.

Variable	Coefficient
Crude Birth Rate	-1.361
Total Fertility Rate	2.765
Net Reproduction Rate	6.981
Crude Death Rate	-0.953

Table 5: Coefficients of the variables Crude Birth Rate, Total Fertility Rate, Net Reproduction Rate, & Crude Death Rate.

Important Value Name	Value
Intercept	90.861
Mean Squared Error	9.267
Coefficient of Determination	0.895
Pearson between Prediction & Test Set	0.946

Table 6: Values of the variables Crude Birth Rate, Total Fertility Rate, Net Reproduction Rate, & Crude Death Rate.

A Code

```
1
2  import numpy as np
3  import pandas as pd
4  import matplotlib.pyplot as plt
5  from sklearn.model_selection import train_test_split
6  from sklearn.metrics import r2_score,
       mean_squared_error
7  from sklearn.linear_model import LinearRegression
8
9  SEED = 12345678
10
11 df = pd.read_csv("assignment_4/life_expectancy.csv")
12 df = df.drop("Country", axis="columns")
13 df_train, df_test = train_test_split(df, random_state=
       SEED, test_size=0.25)
14
15 variables = list(df_train.columns.values)
16
17 LEB = "Life␣Expectancy␣at␣Birth,␣both␣sexes␣(years)"
18 HDE = "Human␣Development␣Index␣(value)"
19
20 pearson = df_train.corr(method='pearson', min_periods
       =1, numeric_only=False)
21 pearson_drop = pearson[LEB].drop(LEB)
22 # max 0.9186657942068257 Human Development Index (
       value),
23 # min -0.8646498454415058 Crude Birth Rate
24 # print(f"{max(pearson_drop)}, {min(pearson_drop)}")
25
26 df_train_hde = df_train[[HDE, LEB]]
27
28 linreg = LinearRegression()
29 linreg.fit(df_train_hde[[HDE]], df_train_hde[LEB])
30 pred = df_train_hde[HDE] * linreg.coef_ + linreg.
       intercept_
31
32 r2_train = r2_score(df_train_hde[LEB], pred)
33
34 predictions = linreg.predict(df_test[[HDE]])
35 # 13.730158926302014
36 mse = mean_squared_error(df_test[LEB], predictions)
37 correlation = df_test[[HDE, LEB]].corr(method='pearson
       ', min_periods=1, numeric_only=False)
38 r2_test = r2_score(df_test[LEB], predictions)
39 # 0.918989
40 # print(correlation)
41
```

```
plt.scatter(df_test[HDE], predictions)
plt.show()

plt.scatter(df_train_hde[HDE], df_train_hde[LEB], s=1)
plt.plot(df_train_hde[HDE], pred, color="red")
plt.show()


MA = "Median Age, as of 1 July (years)"

spearman = df_train.corr(method='spearman',
    min_periods=1, numeric_only=False)
spearman_drop = spearman[LEB].drop(LEB)
print(f"{max(spearman_drop)}, {min(spearman_drop)}")

# no transform for comparison
df_train_nt_ma = df_train[[MA, LEB]]
linreg_nt_ma = LinearRegression()
linreg_nt_ma.fit(df_train_nt_ma[[MA]], df_train_nt_ma[
    LEB])
pred_nt_leb = df_train_nt_ma[MA] * linreg_nt_ma.coef_
    + linreg_nt_ma.intercept_
# 0.799285
correlation_nt_ma = df_train_nt_ma[[MA, LEB]].corr(
    method='pearson', min_periods=1, numeric_only=False
    )

# with transform
df_train_ma = df_train[[MA, LEB]]
df_train_ma["LEB Transform"] = (np.pow(df_train_ma[LEB
    ], 6))

df_test_ma = df_test[[MA, LEB]]
df_test_ma["LEB Transform"] = (np.pow(df_test_ma[LEB],
     6))

linreg_ma = LinearRegression()
linreg_ma.fit(df_train_ma[[MA]], df_train_ma["LEB
    Transform"])
pred_leb = df_train_ma[MA] * linreg_ma.coef_ +
    linreg_ma.intercept_

# for 4: 0.7052068299250327
# for 5: 0.7103432261911896
# for 6: 0.7089855707748298
# for 7: 0.7023901539198063
r2_ma = r2_score(df_train_ma["LEB Transform"],
    pred_leb)

# 0.831497
```

```
correlation_ma = df_test_ma[[MA, "LEB Transform"]].corr(method='pearson', min_periods=1, numeric_only=False)
# print(correlation_ma)

plt.plot(df_train_ma[MA], pred_leb, color="red")
plt.scatter(df_train_ma[MA], df_train_ma["LEB Transform"], s=1)
plt.show()


print(pearson_drop)

for feature in variables:
    df_train[feature].fillna(value=df_train[feature].dropna().mean(), inplace=True)
    df_test[feature].fillna(value=df_test[feature].dropna().mean(), inplace=True)


# for s in subsets:
s = ["Crude Birth Rate (births per 1,000 population)"]
best_mse = 20

for v in variables:
    if v == "Crude Birth Rate (births per 1,000 population)":
        continue
    if v == HDE or v == LEB:
        continue
    s.append(v)

    ln = LinearRegression()
    ln.fit(df_train[[*s]], df_train[LEB])

    predictions = ln.predict(df_test[[*s]])
    mse = mean_squared_error(df_test[LEB], predictions)
    if mse < best_mse:
        best_mse = mse
    else:
        s.pop()


# the initial list was everything listed here,
# then we removed entries that were similar to each other (the ones with no number next to them)
# then we found the pearson correlation coefficients for each of the remaining variables, and removed
```

```
    all
# with coefficient c such that |c| < 0.5
newvars = [
    'Crude Birth Rate (births per 1,000 population)', # -0.86
    # 'Rate of Natural Change (per 1,000 population)',
    # 'Population Change (thousands)',
    # 'Population Growth Rate (percentage)', # -0.2853
    # 'Population Annual Doubling Time (years)',
    # 'Births (thousands)',
    # 'Births by women aged 15 to 19 (thousands)',
    'Total Fertility Rate (live births per woman)', # -0.83
    'Net Reproduction Rate (surviving daughters per woman)', # -0.78
    # 'Mean Age Childbearing (years)', # 0.0423
    # 'Sex Ratio at Birth (males per 100 female births)', # 0.407
    # 'Total Deaths (thousands)',
    'Crude Death Rate (deaths per 1,000 population)', # -0.54
    # 'Live births Surviving to Age 1 (thousands)',
    # 'Net Number of Migrants (thousands)',
    # 'Net Migration Rate (per 1,000 population)' # 0.13
]

ln = LinearRegression()
ln.fit(df_train[[*newvars]], df_train[LEB])

predictions = ln.predict(df_test[[*newvars]])
mse = mean_squared_error(df_test[LEB], predictions)
print(mse)
print(f"{s}: mse:{mse}, intercept: {ln.intercept_}, coeff: {ln.coef_}")
df_test2 = pd.DataFrame({"leb": df_test[LEB], "pred": predictions})
pearson = df_test2.corr(method='pearson', min_periods=1, numeric_only=False)
```

A Sources

https://hdr.undp.org/data-center/human-development-index#/indicies/HDI