DIT407 Introduction to data science and AI
Assignment 4

Reynir Siik
reynir.siik@gmail.com

Franco Zambon
guszamfr@student.gu.se

2024-02-19

Abstract

In this assignment we try out different regression models and investigate the underlying data and the quality of the prediction model. When the underlying data for an index is calculated from the index you want to predict a high correlation can be observed. People live longer partly because they live longer. To be careful to use unbiased indexes to base predictions on. Making a non-linear model and taking more indexes into the calculation makes a better model, as shown in problem 3 and 4.

1 Problem 1: Splitting the data

The data is split in two parts to get enough data points in each part to be statistically significant. The splitting is quite straight forward and the code can be easily adapted to splitting in a number of equally large chunks.

2 Problem 2: Single-variable model

2.1 Introduction

After analyzing the Pearson correlation[1] between life expectancy and every other column, the data with the highest correlation, **0.9216**, is Human Development Index. The highest possible correlation **(1.0)** is when the column for life expectancy is correlated to itself, but since that is trivial in this case that result is ignored.

The Human Development Index (HDI) is a summary measure of average achievement in key dimensions of human development: a long and healthy life, being knowledgeable and having a decent standard of living. The HDI is the geometric mean of normalized indices for each of the three dimensions.[2]

The reason for the high correlation is presumed to be due to the underlying data that HDI is calculated from. One factor is Life Expectancy. Another factor is being knowledgeable, which indicates some level of education. *Expected Years of Schooling (years)* has a correlation to *Life Expectancy at Birth* of **0.8202**.

2.2 Training

The prediction graph has Coefficient: **51.06**, and Intercept: **34.83**.

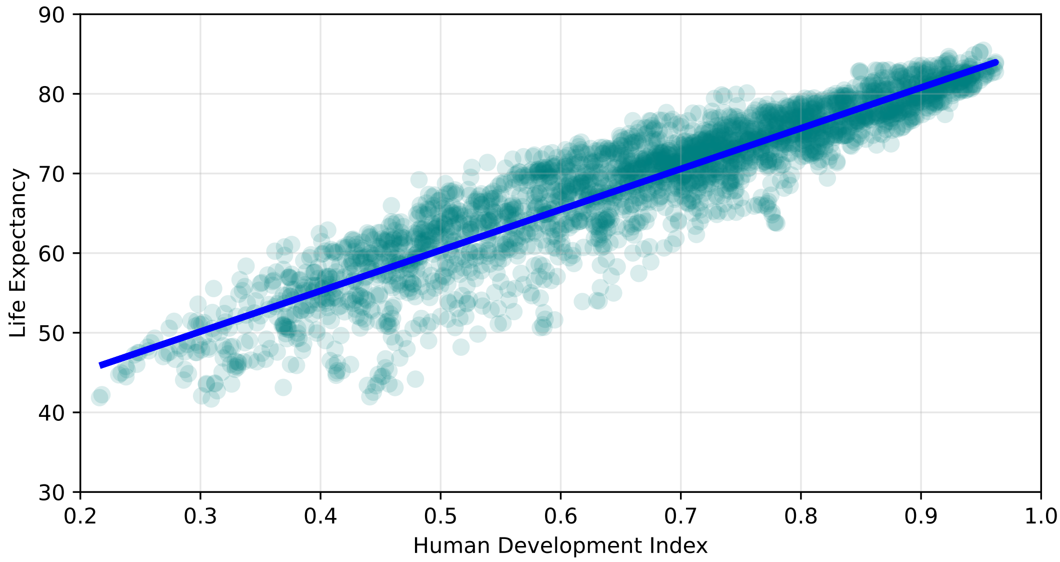

Figure 1: Scatter plot of Human Development Index vs Life Expectancy, training data
Linear prediction model in blue.

2.3 Testing

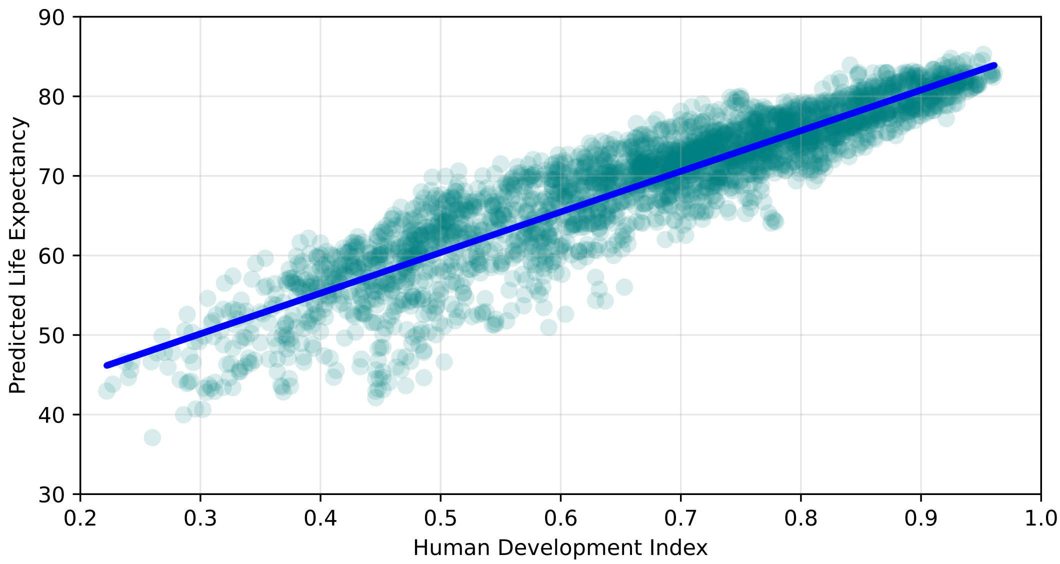

Figure 2: Scatter plot of Human Development Index vs Life Expectancy
Prediction vs real data

The prediction graph versus the test data, figure 2, results in a mean squared error of **13.32**

The coefficient of determination $R^2 = 0.8384$

The correlation between the predicted values and the target variable **0.9157**

3 Problem 3: Non-linear relationship

3.1 Introduction

Looking at the scatter-plots between the different predictors and the dependent variable (Life Expectancy at Birth, both sexes (years)) we found that some of them had a logarithmic monotone relationship with the target variable. We decided to use as a predictor the 'Gross National Income Per Capita (2017 PPP$)' (Figure 3).

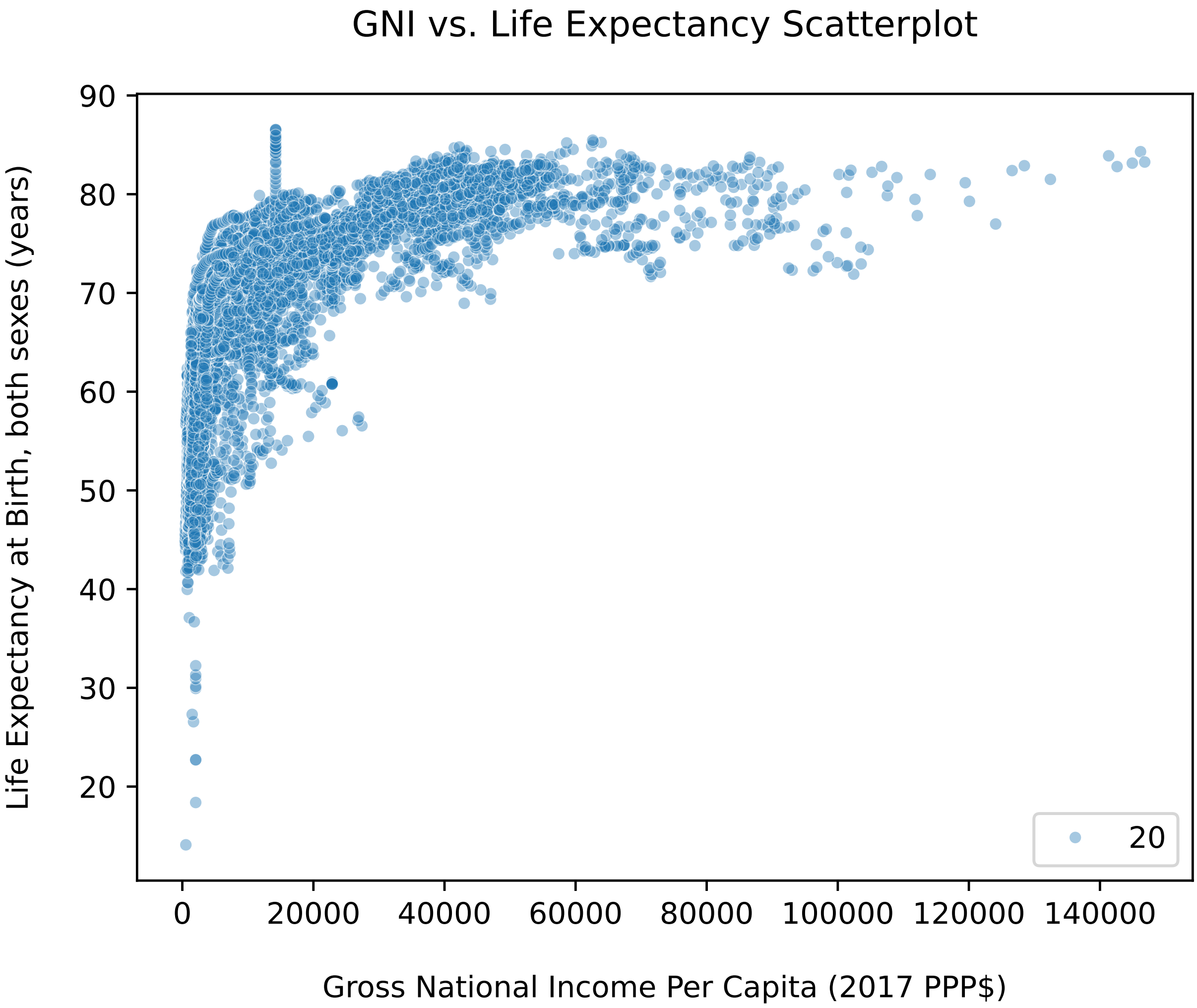

Figure 3: Scatter plot of Gross National Income Per Capita vs Life Expectancy at birth

3.2 Before the transformation

Using this predictor the R-squared that we obtained was 0.4088969210530252, a really low value. Its Pearson coefficient was 0.634651.

3.3 After the transformation

After applying a logarithmic transformation on the x-variable we obtained the scatter plot in figure 4. As we can see the plot shows a much more linear behaviour than before.

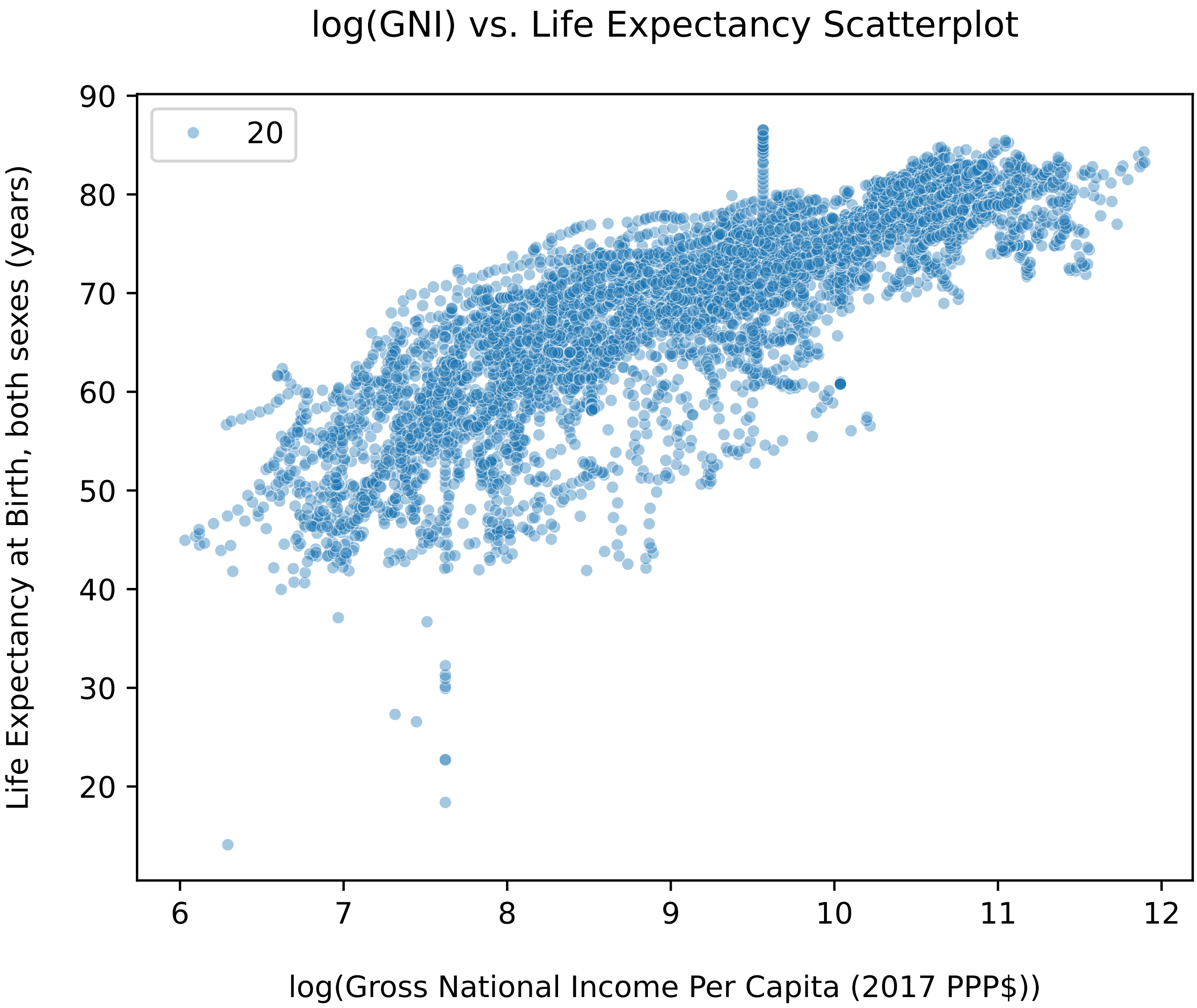

Figure 4: Scatter plot of log(Gross National Income Per Capita) vs Life Expectancy at birth

The R-squared after the transformation is 0.6796613657568554. The Pearson coefficient is 0.8244157723848186.

4 Problem 4: Multiple linear regression

4.1 Introduction

The goal here was to find a minimal subset of variables that allow us to perform a better prediction.

4.2 The process

After printing the list of independent variables and their correlation with the target variable, we selected those which had a coefficient superior to 0.7 or inferior to -0.7 and we tried to apply different combinations of them to the multiple regression model and we evaluated how the the prediction changed just looking at the Adjusted R-squared value.
We also avoided using variables that had a strong correlation between one another (for example we chose to used 'Expected Years of Schooling (years)' but not 'Expected Years of

Schooling, female (years)' nor 'Expected Years of Schooling, male (years)').
The subset of variables selected to generate the multiple regression model are these (P.c. indicates the Person coefficient of each variable):

- Expected Years of Schooling (years) [P.c.: 0.896938]
- Mean Years of Schooling (years) [P.c.: 0.889827]
- Total Fertility Rate (live births per woman) [P.c.: -0.851443]
- Crude Birth Rate (births per 1,000 population) [P.c.: -0.874986]
- Adolescent Birth Rate (births per 1,000 women ages 15-19) [P.c.: -0.790792]
- Net Reproduction Rate (surviving daughters per woman) [P.c.: -0.761124]
- Coefficient of human inequality [P.c.: -0.703339]
- Median Age, as of 1 July (years) [P.c.: 0.786724]

The linear model obtained has the following coefficients:
Intercept: 60.230469183112646
Coefficients: [3.84139630e-01 -2.66957876e-01 -8.54299584e+00 -3.78035444e-01 -1.98550446e-02 2.50527878e+01 -6.89802117e-02 3.83722522e-01]
The resulting R-Squared is 0.8781166276189235.
The MSE is 29.899182603718497.
The new model shows a significant improvement from the previous one.

References

[1] F. Pedregosa, G. Varoquaux, A. Gramfort, *et al.*, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[2] UNDP, *Human development index (hdi)*, UN web, Read: 2024-FEB-15, 2024. [Online]. Available: `https://hdr.undp.org/data-center/human-development-index#/indicies/HDI`.

A Source code

This is the complete listing of the source code.

A.1 Problem 1 and 2

Code for splitting the datasets into training and test sets and training a linear model.

```python
# Reynir Siik, Franco Zambon
# DIT407 lp3 2024-02-11
# Assignment 4
# Problem 1, 2
# Life expectancy
# Splitting the dataset by random
#########################################################################
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score
#########################################################################
filelocation_in = r"Uppgift04\\00_Uppg_Data\\" # folder to get data from
filelocation_out = r"Uppgift04\\test_data\\" # folder to write data to
## ######################################################################
## How many parts to split data into ####################################
chunks = 2
## ######################################################################
df = pd.read_csv(filelocation_in + "life_expectancy.csv")
df.insert(1, 'Dataset', 0)
df_shape = df.shape[0]
df_chunk = df_shape//chunks
# print(df_chunk, df_shape/chunks)
DFtrainingSet, DFtestingSet  = train_test_split(
    df, train_size=df_chunk, random_state=42)
# Open a new (empty) csv file and write the data to it, skip index
DFtrainingSet.to_csv(
    filelocation_out + "life_expectancy.csv", index=False)

## Split the dataset and mark each chunk with a unique number under the
## column 'Dataset'. Append data to existing file
for i in range(chunks-2):
    DFtrainingSet, DFtestingSet = train_test_split(
        DFtestingSet, train_size=df_chunk, random_state=42)
    DFtrainingSet['Dataset'] = i + 1 # Mark the data as chunk i+1
    # Append data chunk to csv file, no header, no index
    DFtrainingSet.to_csv(
        filelocation_out + "life_expectancy.csv", mode='a',
        header=False, index=False)

DFtestingSet['Dataset'] = chunks-1
# Append data chunk to csv file, no header, no index
DFtestingSet.to_csv(filelocation_out + "life_expectancy.csv", mode='a',
                    header=False, index=False)

## Problem 2 ############################################################
# print(DFtrainingSet.shape[0])
numdf = DFtrainingSet.drop(["Country","Dataset"] , axis='columns')
```

```
## Calculate Pearson correlation for all columns in relation to all other
## columns, two by two
df_corr = numdf.corr('pearson')
df_corr['Life Expectancy at Birth, both sexes (years)'].to_csv(filelocation_out + "c

plot_colour = '#008080'
plot_alpha = 0.1
plot_width = 8
plot_height = 4
## Scatter plot of the life expectancy and HDI
plt.figure(figsize=(plot_width, plot_height))
# Set the range of x- and y-axis
plt.xlim(0.2, 1.0)
plt.ylim(30, 90)

plt.scatter(
    DFtrainingSet['Human Development Index (value)'],
    DFtrainingSet['Life Expectancy at Birth, both sexes (years)'],
    color=plot_colour, alpha=plot_alpha, s=50)
plt.grid(True, linestyle='-', alpha=0.3)
plt.xlabel('Human Development Index\n')
plt.ylabel('\nLife Expectancy')

DFtrainingSet.dropna(subset=[
    'Human Development Index (value)',
    'Life Expectancy at Birth, both sexes (years)'],
    inplace=True)
DFtestingSet.dropna(subset=[
    'Human Development Index (value)',
    'Life Expectancy at Birth, both sexes (years)'],
    inplace=True)
### DF_numpy = DFtestingSet.to_numpy()
y_train = DFtrainingSet[
    'Life Expectancy at Birth, both sexes (years)'
    ].to_numpy().reshape(-1, 1)
X_train = DFtrainingSet[
    'Human Development Index (value)'
    ].to_numpy().reshape(-1, 1)

y_test = DFtestingSet[
    'Life Expectancy at Birth, both sexes (years)'
    ].to_numpy().reshape(-1, 1)
X_test = DFtestingSet[
    'Human Development Index (value)'
    ].to_numpy().reshape(-1, 1)
# print(X_test)
# # Create linear regression object
regr = LinearRegression()

# # Train the model using the training sets
regr.fit(X_train, y_train)

print('Coefficients: \n', regr.coef_)
print('Intercept: ', regr.intercept_)
# # The mean square error
print("Residual sum of squares: %.2f"
      % np.mean((regr.predict(X_train) - y_train) ** 2))
# Explained variance score: 1 is perfect prediction
print('Variance score: %.2f' % regr.score(X_train, y_train))
```

```

plt.plot(X_train, regr.predict(X_train), color='blue',
         linewidth=3)
plt.savefig(filelocation_out + 'scatter_plot_problem2_train.pdf')
#plt.show()

plt.figure(figsize=(plot_width, plot_height))
# Set the range of x- and y-axis
plt.xlim(0.2, 1.0)
plt.ylim(30, 90)
plt.grid(True, linestyle='-', alpha=0.3)
plt.xlabel('Human␣Development␣Index\n')
plt.ylabel('\nPredicted␣Life␣Expectancy')

plt.scatter(
    DFtestingSet['Human␣Development␣Index␣(value)'],
    DFtestingSet['Life␣Expectancy␣at␣Birth,␣both␣sexes␣(years)'],
    color=plot_colour, alpha=plot_alpha, s=50)

plt.plot(X_test, regr.predict(X_test), color='blue',
         linewidth=3)

plt.savefig(filelocation_out + 'scatter_plot_problem2_predict.pdf')

# Explained variance score: 1 is perfect prediction
print('Variance␣score:␣%.2f' % regr.score(X_test, y_test))

# Predict the dependent variable using the fitted model
y_pred = regr.predict(X_test)

# Calculate the R-squared
r_squared = r2_score(y_test, y_pred)
print("R-squared:",r_squared)

# # The mean square error
print("Residual␣sum␣of␣squares:␣%.2f"
      % np.mean((regr.predict(X_test) - y_test) ** 2))

# The correlation between the predicted values and the target variable
print(pd.DataFrame(
    {'Predict':regr.predict(X_test).T[0],
     'Target': y_test.T[0]}).corr('pearson'))
```

A.2 Problem 3 and 4

Code for Non-linear relationship and multiple linear regression.

```
# Reynir Siik, Franco Zambon
# DIT407 lp3 2024-02-11
# Assignment 4

## PROBLEM 3

import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score
from sklearn.metrics import mean_squared_error
```

```


df = pd.read_csv('C:\\Users\\ilmio\\Downloads\\life_expectancy.csv')
df.head(60)

df_filled = df.groupby('Country').ffill().bfill()
df_filled.shape

num_rows_to_select = int(len(df_filled) * 70 / 100)
training_df = df_filled.sample(n=num_rows_to_select, random_state=42)
#random_sample.shape
training_df.head()

# Select only numeric columns
numeric_df = training_df.select_dtypes(include='number')

# Compute the correlation matrix
correlation_matrix = numeric_df.corr()
correlation_matrix

human_dev_corr = correlation_matrix['Human␣Development␣Index␣(value)']
most_correlated_variables = human_dev_corr.drop('Human␣Development␣Index␣(value)').dr
filtered_variables = most_correlated_variables[(most_correlated_variables > 0.85) | (
filtered_variables

filtered_variable_names = list(filtered_variables.index)
training_df = training_df.drop(filtered_variable_names, axis=1)
training_df

numeric_df = training_df.select_dtypes(include='number')
correlation_matrix = numeric_df.corr()
life_expectancy_corr = correlation_matrix['Life␣Expectancy␣at␣Birth,␣both␣sexes␣(year
most_correlated_variables = life_expectancy_corr.drop('Life␣Expectancy␣at␣Birth,␣both
most_correlated_variables

X = df_filled[['Expected␣Years␣of␣Schooling␣(years)','Mean␣Years␣of␣Schooling␣(years)

y = df_filled['Life␣Expectancy␣at␣Birth,␣both␣sexes␣(years)']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state

model = LinearRegression()

model.fit(X, y)

print("Intercept:", model.intercept_)
print("Coefficients:", model.coef_)

y_pred = model.predict(X)

r_squared = r2_score(y, y_pred)

# Print the R-squared
print("R-squared:", r_squared)

mse = mean_squared_error(y, y_pred)

print("Mean␣Squared␣Error␣(MSE):", mse)

```

```python
## PROBLEM 4

import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np

X = df_filled.drop('Life Expectancy at Birth, both sexes (years)', axis=1)

for column in X.columns:
    # Create a scatterplot
    sns.scatterplot(x=X[column], y=y, size=20, alpha=0.2)
    plt.xlabel(column)
    plt.ylabel('Life Expectancy at Birth, both sexes (years)')
    plt.title(f'Scatterplot of {column} vs. Life Expectancy at Birth')
    plt.show()

X = df_filled['Median Age, as of 1 July (years)']
X = X.to_frame()

# Check the shape of X
print(X.shape)
model.fit(X, y)

print("Intercept:", model.intercept_)
print("Coefficients:", model.coef_)

y_pred = model.predict(X)

r_squared = r2_score(y, y_pred)

print("R-squared:", r_squared)

log_X = np.log(X['Median Age, as of 1 July (years)'])
log_X = log_X.to_frame()

model.fit(log_X, y)
y_pred = model.predict(log_X)

r_squared = r2_score(y, y_pred)

print("R-squared:", r_squared)

x_values = X.squeeze().values  # Convert DataFrame or Series to 1D array
y_values = y.values

# Plot scatterplot
sns.scatterplot(x=x_values, y=y_values, size=20, alpha=0.2)
plt.xlabel('Median Age, as of 1 July (years)')
plt.ylabel('Life Expectancy at Birth, both sexes (years)')
plt.show()

x_values = log_X.squeeze().values  # Convert DataFrame or Series to 1D array
y_values = y.values

# Plot scatterplot
sns.scatterplot(x=x_values, y=y_values, size=20, alpha=0.4)
```

```python
plt.xlabel('log(Median Age, as of 1 July (years))')
plt.ylabel('Life Expectancy at Birth, both sexes (years)')
plt.show()
plt.savefig('log(MedianAge)vsLifeExpectancy.pdf', bbox_inches='tight')

X = df_filled['Gross National Income Per Capita (2017 PPP$)']
X = X.to_frame()

model.fit(X, y)

print("Intercept:", model.intercept_)
print("Coefficients:", model.coef_)

y_pred = model.predict(X)

r_squared = r2_score(y, y_pred)

print("R-squared:", r_squared)

log_X = np.log(X['Gross National Income Per Capita (2017 PPP$)'])
log_X = log_X.to_frame()
model.fit(log_X, y)
y_pred = model.predict(log_X)

r_squared = r2_score(y, y_pred)

print("R-squared:", r_squared)

x_values = X.squeeze().values  # Convert DataFrame or Series to 1D array
y_values = y.values

# Plot scatterplot
sns.scatterplot(x=x_values, y=y_values, size=20, alpha=0.4)
plt.title('GNI vs. Life Expectancy Scatterplot\n')
plt.xlabel('\nGross National Income Per Capita (2017 PPP$)')
plt.ylabel('Life Expectancy at Birth, both sexes (years)\n')

#plt.show()
plt.savefig('GNIvsLifeExpect.pdf', bbox_inches='tight')

x_values = log_X.squeeze().values  # Convert DataFrame or Series to 1D array
y_values = y.values

# Plot scatterplot
sns.scatterplot(x=x_values, y=y_values, size=20, alpha=0.4)
plt.title('log(GNI) vs. Life Expectancy Scatterplot\n')
plt.xlabel('\nlog(Gross National Income Per Capita (2017 PPP$))')
plt.ylabel('Life Expectancy at Birth, both sexes (years)\n')
#plt.show()
plt.savefig('log(GNI)vsLifeExpect.pdf', bbox_inches='tight')

########################################################################
```