Numerical Computing

Dr. Amin Eskandari

TA: Hosein Mirhoseini

Final project

Mohammadreza Naziri

40030505

I used numpy, pandas, matplotlib, seaborn, sklearn python libraries in this project.

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn import preprocessing, linear_model, model_selection, metrics
```

Read data from .csv file with pandas:

```
df = pd.read_csv("Life_Expectancy_Data.csv")
print(df)
```

```
----------------------------------------
         Country  Year  ... Income composition of resources  Schooling
0     Afghanistan  2015  ...                          0.479       10.1
1     Afghanistan  2014  ...                          0.476       10.0
2     Afghanistan  2013  ...                          0.470        9.9
3     Afghanistan  2012  ...                          0.463        9.8
4     Afghanistan  2011  ...                          0.454        9.5
...           ...   ...  ...                            ...        ...
2933     Zimbabwe  2004  ...                          0.407        9.2
2934     Zimbabwe  2003  ...                          0.418        9.5
2935     Zimbabwe  2002  ...                          0.427       10.0
2936     Zimbabwe  2001  ...                          0.427        9.8
2937     Zimbabwe  2000  ...                          0.434        9.8

[2938 rows x 22 columns]
----------------------------------------
```

Show information of each column: `print(df.info())`

```
----------------------------------------
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2938 entries, 0 to 2937
Data columns (total 22 columns):
 #   Column                           Non-Null Count  Dtype
---  ------                           --------------  -----
 0   Country                          2938 non-null   object
 1   Year                             2938 non-null   int64
 2   Status                           2938 non-null   object
 3   Life expectancy                  2928 non-null   float64
 4   Adult Mortality                  2928 non-null   float64
 5   infant deaths                    2938 non-null   int64
 6   Alcohol                          2744 non-null   float64
 7   percentage expenditure           2938 non-null   float64
 8   Hepatitis B                      2385 non-null   float64
 9   Measles                          2938 non-null   int64
 10   BMI                             2904 non-null   float64
 11  under-five deaths                2938 non-null   int64
 12  Polio                            2919 non-null   float64
 13  Total expenditure                2712 non-null   float64
 14  Diphtheria                       2919 non-null   float64
 15   HIV/AIDS                        2938 non-null   float64
 16  GDP                              2490 non-null   float64
 17  Population                       2286 non-null   float64
 18   thinness  1-19 years            2904 non-null   float64
 19   thinness 5-9 years             2904 non-null   float64
 20  Income composition of resources  2771 non-null   float64
 21  Schooling                        2775 non-null   float64
dtypes: float64(16), int64(4), object(2)
memory usage: 505.1+ KB
None
----------------------------------------
```

Describe some more information for example mid, max, min:

```
print(df.describe(include='all'))
```

```
----------------------------------------
            Country         Year  ... Income composition of resources    Schooling
count          2938  2938.000000  ...                      2771.000000  2775.000000
unique          193          NaN  ...                              NaN          NaN
top     Afghanistan          NaN  ...                              NaN          NaN
freq             16          NaN  ...                              NaN          NaN
mean            NaN  2007.518720  ...                         0.627551    11.992793
std             NaN     4.613841  ...                         0.210904     3.358920
min             NaN  2000.000000  ...                         0.000000     0.000000
25%             NaN  2004.000000  ...                         0.493000    10.100000
50%             NaN  2008.000000  ...                         0.677000    12.300000
75%             NaN  2012.000000  ...                         0.779000    14.300000
max             NaN  2015.000000  ...                         0.948000    20.700000

[11 rows x 22 columns]
----------------------------------------
```

Scaling population to smaller number:

```
df['Population'] = df['Population']/1000000
print(df['Population'])
```

```
----------------------------------------
0        33.736494
1         0.327582
2        31.731688
3         3.696958
4         2.978599
           ...
2933     12.777511
2934     12.633897
2935      0.125525
2936     12.366165
2937     12.222251
Name: Population, Length: 2938, dtype: float64
----------------------------------------
```

Delete rows with null value:

```
df = df.dropna(axis=0)
df.info()
```

```
----------------------------------------
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1649 entries, 0 to 2937
Data columns (total 22 columns):
 #   Column                           Non-Null Count  Dtype
---  ------                           --------------  -----
 0   Country                          1649 non-null   object
 1   Year                             1649 non-null   int64
 2   Status                           1649 non-null   object
 3   Life expectancy                  1649 non-null   float64
 4   Adult Mortality                  1649 non-null   float64
 5   infant deaths                    1649 non-null   int64
 6   Alcohol                          1649 non-null   float64
 7   percentage expenditure           1649 non-null   float64
 8   Hepatitis B                      1649 non-null   float64
 9   Measles                          1649 non-null   int64
 10   BMI                             1649 non-null   float64
 11  under-five deaths                1649 non-null   int64
 12  Polio                            1649 non-null   float64
 13  Total expenditure                1649 non-null   float64
 14  Diphtheria                       1649 non-null   float64
 15   HIV/AIDS                        1649 non-null   float64
 16  GDP                              1649 non-null   float64
 17  Population                       1649 non-null   float64
 18   thinness  1-19 years            1649 non-null   float64
 19   thinness 5-9 years             1649 non-null   float64
 20  Income composition of resources  1649 non-null   float64
 21  Schooling                        1649 non-null   float64
dtypes: float64(16), int64(4), object(2)
memory usage: 296.3+ KB
----------------------------------------
```

Select and save data of Turkey:

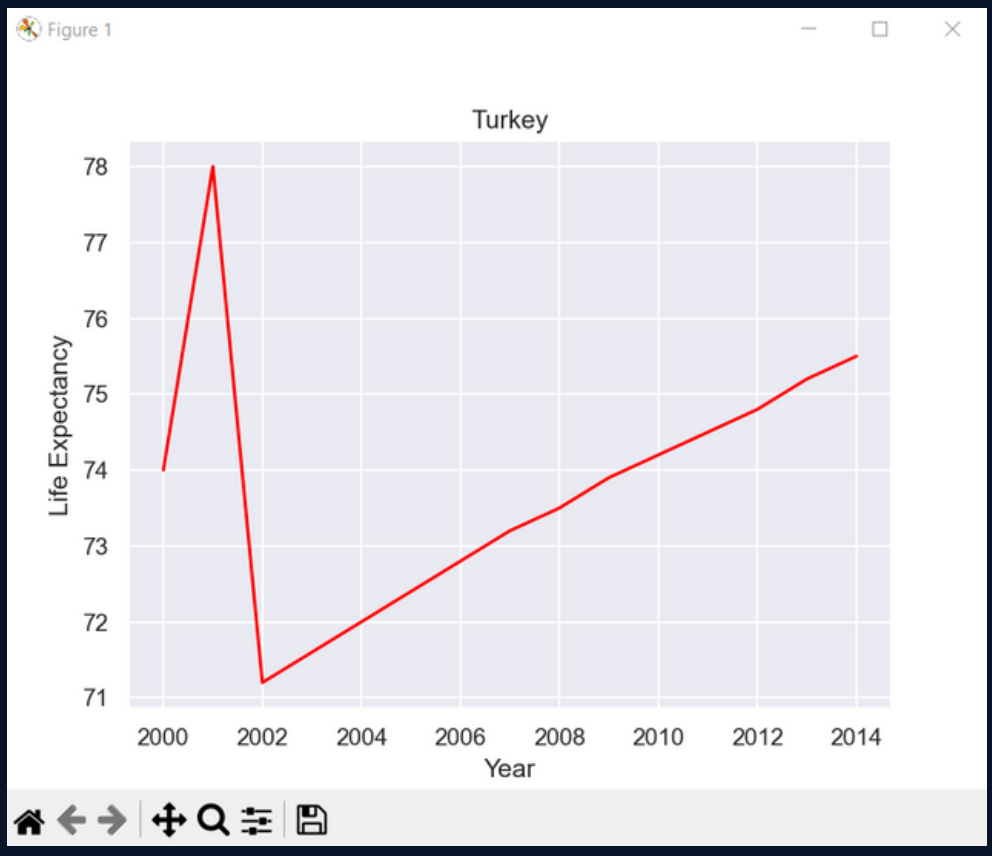
```
Turkey =df[df['Country'] == 'Turkey']
print(Turkey)
```

```
----------------------------------------
     Country  Year  ... Income composition of resources  Schooling
2682  Turkey  2014  ...                           0.759       14.5
2683  Turkey  2013  ...                           0.754       14.4
2684  Turkey  2012  ...                           0.750       14.3
2685  Turkey  2011  ...                           0.737       13.8
2686  Turkey  2010  ...                           0.715       13.0
2687  Turkey  2009  ...                           0.709       12.5
2688  Turkey  2008  ...                           0.705       12.5
2689  Turkey  2007  ...                           0.697       12.3
2690  Turkey  2006  ...                           0.687       11.9
2691  Turkey  2005  ...                           0.681       11.9
2692  Turkey  2004  ...                           0.675       12.0
2693  Turkey  2003  ...                           0.668       11.9
2694  Turkey  2002  ...                           0.658       11.5
2695  Turkey  2001  ...                           0.653       11.1
2696  Turkey  2000  ...                           0.641       10.7

[15 rows x 22 columns]
----------------------------------------
```

Plot Life Expectancy of Turkey:

```
plt.plot(Turkey['Year'], Turkey['Life expectancy '], color='red')
plt.xlabel('Year')
plt.ylabel('Life Expectancy')
plt.title("Turkey")
plt.show()
```



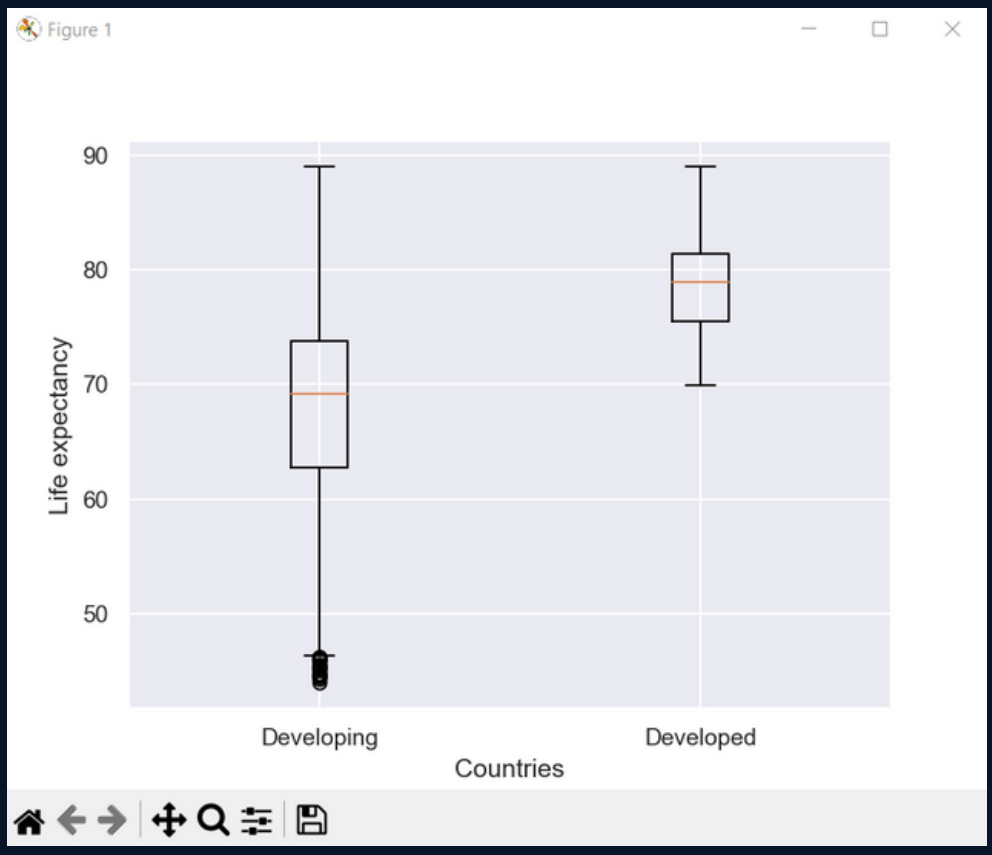
Status has two values: Developing, Developed

```
print(df['Status'].value_counts())
```

```
----------------------------------------
Developing    1407
Developed      242
Name: Status, dtype: int64
----------------------------------------
```

Plot Life Expectancy in Developed countries and Developing countries with box plot:

```
plt.boxplot([df[df['Status']=='Developing']['Life expectancy '],
             df[df['Status']=='Developed']['Life expectancy ']],
             labels=['Developing','Developed'])
plt.ylabel('Life expectancy')
plt.xlabel('Countries')
plt.show()
```



Sort countries by Life Expectancy:

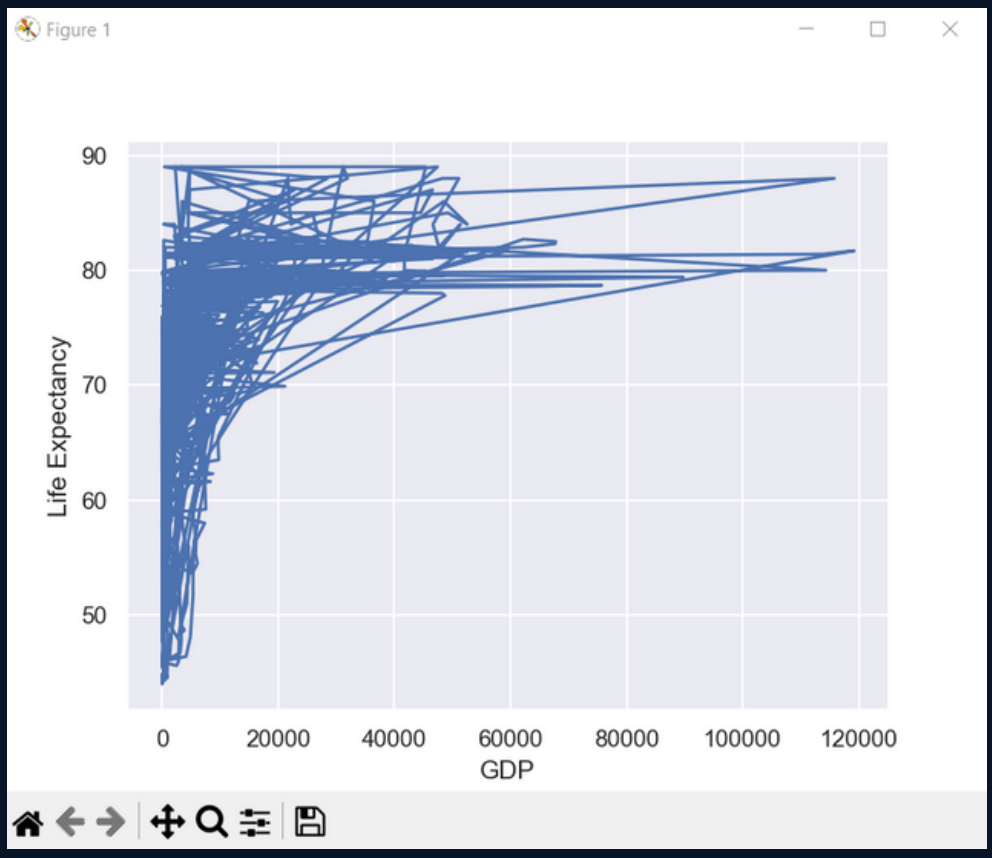
```
print(df.sort_values("Life expectancy "))
```

```
----------------------------------------
      Country  Year  ... Income composition of resources  Schooling
1583   Malawi  2002  ...                           0.388       10.4
2933 Zimbabwe  2004  ...                           0.407        9.2
1484  Lesotho  2005  ...                           0.437       10.7
2934 Zimbabwe  2003  ...                           0.418        9.5
2932 Zimbabwe  2005  ...                           0.406        9.3
...        ...   ...  ...                             ...        ...
937    France  2008  ...                           0.877       16.1
2056 Portugal  2014  ...                           0.837       16.8
995   Germany  2014  ...                           0.920       17.0
241   Belgium  2014  ...                           0.890       16.3
2433    Spain  2007  ...                           0.849       16.0

[1649 rows x 22 columns]
----------------------------------------
```

Plot Life Expectancy by Gross Domestic Product:

```
plt.plot(df['GDP'], df['Life expectancy '])
plt.xlabel('GDP')
plt.ylabel('Life Expectancy')
plt.show()
```



Convert qualitative data(Status, Country) to quantitative:

```
l_encoder= preprocessing.LabelEncoder()
df.loc[:,'Status'] = l_encoder.fit_transform(df.loc[:,'Status'])
print(df['Status'].value_counts())
```

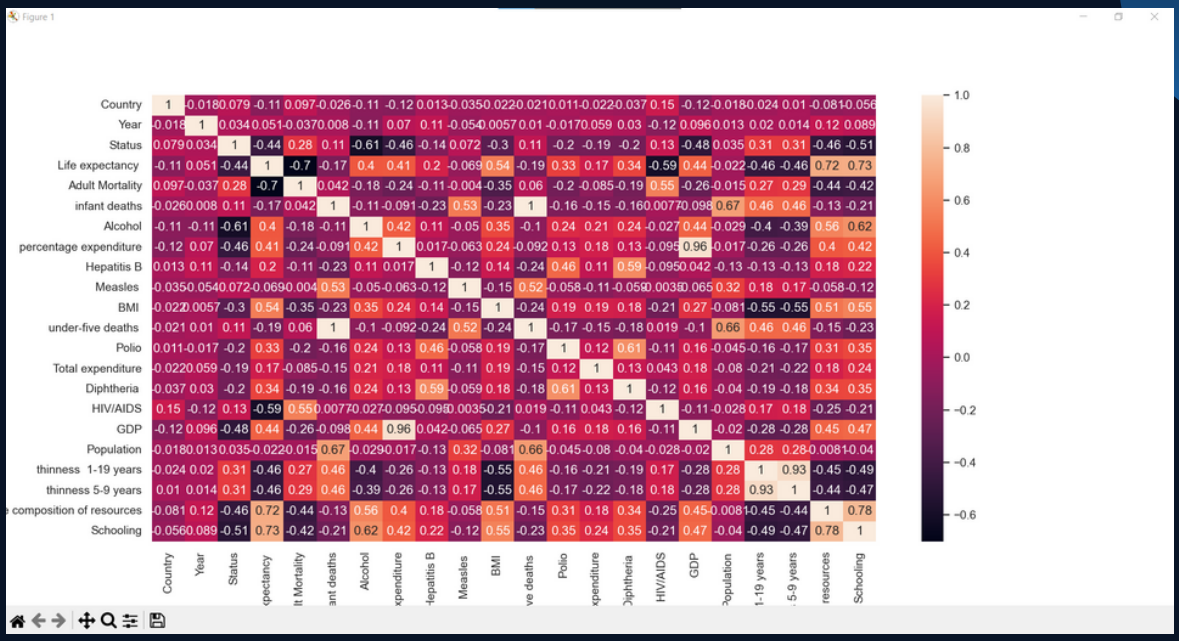
```
df.loc[:, 'Country'] = l_encoder.fit_transform(df.loc[:, 'Country'])
print(df['Country'].value_counts())
```

```
1    1407
0     242
Name: Status, dtype: int64
--------------------------------------
```

```
0      16
1      16
65     15
80     15
79     15
       ..
58      5
115     4
88      4
53      2
38      1
Name: Country, Length: 133, dtype: int64
----------------------------------------
```

This plot shows the ratio of data to each other:

```
fig, ax = plt.subplots(figsize=(20,20))
sns.heatmap(df.corr(), annot=True, ax=ax)
plt.show()
```



Delete 'Income composition of resources', ' thinness 5-9 years', 'infant deaths', , 'percentage expenditure' columns.

```
df =df.drop(labels=['Income composition of resources', ' thinness 5-9 years',
                    'infant deaths', 'percentage expenditure'], axis=1)
```

```
---------------------------------------
      Country  Year  Status  ...  Population   thinness  1-19 years  Schooling
0           0  2015       1  ...   33.736494                   17.2       10.1
1           0  2014       1  ...    0.327582                   17.5       10.0
2           0  2013       1  ...   31.731688                   17.7        9.9
3           0  2012       1  ...    3.696958                   17.9        9.8
4           0  2011       1  ...    2.978599                   18.2        9.5
...       ...   ...     ...  ...         ...                    ...        ...
2933      132  2004       1  ...   12.777511                    9.4        9.2
2934      132  2003       1  ...   12.633897                    9.8        9.5
2935      132  2002       1  ...    0.125525                    1.2       10.0
2936      132  2001       1  ...   12.366165                    1.6        9.8
2937      132  2000       1  ...   12.222251                   11.0        9.8

[1649 rows x 18 columns]
---------------------------------------
```

Keep Life Expectancy in y and delete it from data:

```
y = df['Life expectancy ']
df = df.drop(labels='Life expectancy ', axis=1)
print(df)
```

```
--------------------------------------
      Country  Year  Status  ...  Population   thinness  1-19 years  Schooling
0           0  2015       1  ...   33.736494                  17.2       10.1
1           0  2014       1  ...    0.327582                  17.5       10.0
2           0  2013       1  ...   31.731688                  17.7        9.9
3           0  2012       1  ...    3.696958                  17.9        9.8
4           0  2011       1  ...    2.978599                  18.2        9.5
...       ...   ...     ...  ...         ...                   ...        ...
2933      132  2004       1  ...   12.777511                   9.4        9.2
2934      132  2003       1  ...   12.633897                   9.8        9.5
2935      132  2002       1  ...    0.125525                   1.2       10.0
2936      132  2001       1  ...   12.366165                   1.6        9.8
2937      132  2000       1  ...   12.222251                  11.0        9.8

[1649 rows x 17 columns]
--------------------------------------
```

y(Life Expectancy) to array with numpy:

```
y = y.to_numpy(dtype='float64')
print(y)
```

```
----------------------------------------
[65.  59.9 59.9 ... 44.8 45.3 46. ]
----------------------------------------
```

Splitting a dataset into training, validation, and test sets.
Line 1: 20% for test and 80% for training.
Line 2: 50% for test and 50% for validation.

```
x_train, x_test, y_train, y_test = model_selection.train_test_split(df, y, test_size=0.2, random_state=42)
x_valid, x_test, y_valid, y_test = model_selection.train_test_split(x_test, y_test, test_size=0.5, random_state=42)
print(f'X_train shape -->{x_train.shape}')
print(f'X_valid shape -->{x_valid.shape}')
print(f'X_test shape -->{x_test.shape}')
print(f'y_train shape -->{y_train.shape}')
print(f'y_valid shape -->{y_valid.shape}')
print(f'y_test shape -->{y_test.shape}')
```

```
----------------------------------------
X_train shape -->(1319, 17)
X_valid shape -->(165, 17)
X_test shape -->(165, 17)
y_train shape -->(1319,)
y_valid shape -->(165,)
y_test shape -->(165,)
----------------------------------------
```

Standardize data using sklearn:

```
scaler = preprocessing.StandardScaler()
x_train = scaler.fit_transform(x_train)
x_valid = scaler.transform(x_valid)
x_test = scaler.transform(x_test)
print(x_train)
```

```
----------------------------------------
[[-0.51128185  0.78367891  0.41023228 ...  0.17827985  0.44055795
  -0.42801883]
 [-1.61551902 -1.43186106  0.41023228 ... -0.16605599 -0.79715621
   1.49260545]
 [ 0.74703494 -0.20100552  0.41023228 ...  0.22946212 -0.79715621
   0.39002484]
 ...
 [ 0.07935664  0.29133669 -2.43764338 ... -0.17616066 -0.44972767
   1.59930679]
 [ 1.31199349  0.04516559  0.41023228 ... -0.11980142  0.78798649
   0.28332349]
 [ 0.6443152   0.5375078   0.41023228 ... -0.19886689 -0.58001337
  -1.63730078]]
----------------------------------------
```

Linear regression modeling and makes predictions using the trained model:

```
model = linear_model.LinearRegression()
model.fit(X=x_train, y=y_train)
val_hat = model.predict(x_valid)
print(metrics.mean_squared_error(y_valid, val_hat))
```

```
----------------------------------------
14.462229070381316
----------------------------------------
```

last line: the mean squared error (MSE) between the predicted values and the actual target values.

Use the trained linear regression model (**model**) to make predictions on the test data.

```
poi_test = model.predict(x_test)
print(metrics.r2_score(y_test, poi_test))
```

```
----------------------------------------
0.8004492503031824
----------------------------------------
```

calculates the R-squared score between the predicted values and the actual target values.
The R-squared score ranges from 0 to 1, with 1 indicating a perfect fit where the predicted values perfectly match the actual values. A higher R-squared score indicates that the model's predictions explain more of the variance in the target variable.
So this linear regression is good enough to predict WHO.