Lab 2: Basics of Statistical Analysis in Python.

DV1604 Interactive Laboratories.

Blekinge Institute of Technology.

2024.

Designed and developed by Diego Navarro Tek.Lic., modified by Milena Angelova PhD 2024.

Description.

The following document presents some of the common techniques used to explore and analyze different variables in a data set, and how to conduct a hypothesis testing using Python and the Jupyter Notebook. It simulates a research exercise in which a research question is proposed, and set of hypotheses are evaluated based on a dataset published by the United Nations (UN).

The topics that will be covered int this lab are:

- Importing and creating datasets from external files.
- Plotting datasets.
- Checking for dataset completeness.
- Data distributions.
- Regressions.
- Statistical significance test.

---

1. Research Questions and Hypotheses.

Every year the UN Sustainable Development Solutions Network analyses the happiness levels in several countries around the world, based on several indicators. In particular, this analysis reviews the Global Happiness Index dataset for the year 2019.

Based on that information, this analysis proposes a couple research questions:

1. What relationship may exist between the Happiness Index Score, and the GDP and the healthy life expectancy indicators of the surveyed countries in 2019?

To address this research question, the following hypotheses are proposed:

- H1: Healthy life expectancy and GDP ***will have a directly proportional relationship*** with the happiness score.
- H0: Healthy life expectancy and GDP ***will not have a directly proportional relationship*** with the happiness score.

2. How does the Global Happyness Index variate between the countries with low, average, and high happiness scores?

For this research question we will propose another set of hypothesis:

- H1: The values of high, average, and low countries ***are significantly different*** between one another.

- H0: The values of high, average, and low countries ***are not significantly different*** between one another.

---

2. Libraries.

The libraries that are used in this exercise are:

1. Matplotlib: A library for plotting graphs and data visualization (covered in lab 1).
2. Numpy: A library to compute large multidimensional arrays and matrixes. Also compiles a broad set of high-level mathematical functions.
3. Pandas: A library for data manipulation and dataset analysis.
4. Scipy: Comprehensive library for numerical integration, interpolation, optimization, linear algebra, and statistics.

We now import all the different libraries that are needed for this exercise, making sure to run the `%matplotlib inline` magic command before the libraries themselves:

In [ ]:
```
%matplotlib inline

# Importing pandas
import pandas as pd

#Importing Matplotlib
import matplotlib.pyplot as mpl

#Importing Numpy
import numpy as np

#Importing Scipy
import scipy.stats as scp
```

---

3. Importing and Reviewing Data Sets.

To import a dataset into our Python application, we can use several different commands depending on the type of file we try to import. For this example, we will be working with a *CSV* (Comma Separated Values) file. However, pandas support a wide variety of formats to use as an input of data.

> 📚 **Additional Material**: For more information regarding the input options supported by Pandas check the Input/Output documentation.

To manipulate the Global Happiness Index dataset, we will use a *Data Frame*. Data Frames are two-dimensional data structures, that specialize in the handling of tabular data. For this example, we will load the CSV file into a variable using the `pd.read_csv` command, specifying the name of the file between quotation marks `""`, and the character used as a delimiter to separate the file into columns in the `sep` parameter.

> 📚 **Additional Material**: For more information about Data Frames, check the DataFrame documentation.

To visualize the dataset we can simply type the name of the variable to get an overview, or we can use the command `head()` from Pandas, specifying the number of rows we want to visualize.

> 💡 **Tip**: The `head()` command can visualize up to 60 data points in a dataset.

In [ ]:
```
# visualize the dataset
# add your code here
```

Out[ ]:

	Overall_rank	Country_or_region	Score	GDP_per_capita	Social_support	Healthy_life_expectancy	Freedom_to_
0	1	Finland	7.769	1.340	1.587	0.986	
1	2	Denmark	7.600	1.383	1.573	0.996	
2	3	Norway	7.554	1.488	1.582	1.028	
3	4	Iceland	7.494	1.380	1.624	1.026	
4	5	Netherlands	7.488	1.396	1.522	0.999	
...	...	...	...	...	...	...	
151	152	Rwanda	3.334	0.359	0.711	0.614	
152	153	Tanzania	3.231	0.476	0.885	0.499	
153	154	Afghanistan	3.203	0.350	0.517	0.361	
154	155	Central African Republic	3.083	0.026	0.000	0.105	
155	156	South Sudan	2.853	0.306	0.575	0.295	

156 rows × 9 columns

In [ ]:
```
# visualize the first 20 rows
# add your code here
```

Out[ ]:

	Overall_rank	Country_or_region	Score	GDP_per_capita	Social_support	Healthy_life_expectancy	Freedom_to_m
0	1	Finland	7.769	1.340	1.587	0.986	
1	2	Denmark	7.600	1.383	1.573	0.996	
2	3	Norway	7.554	1.488	1.582	1.028	
3	4	Iceland	7.494	1.380	1.624	1.026	
4	5	Netherlands	7.488	1.396	1.522	0.999	
5	6	Switzerland	7.480	1.452	1.526	1.052	
6	7	Sweden	7.343	1.387	1.487	1.009	
7	8	New Zealand	7.307	1.303	1.557	1.026	
8	9	Canada	7.278	1.365	1.505	1.039	
9	10	Austria	7.246	1.376	1.475	1.016	
10	11	Australia	7.228	1.372	1.548	1.036	
11	12	Costa Rica	7.167	1.034	1.441	0.963	
12	13	Israel	7.139	1.276	1.455	1.029	
13	14	Luxembourg	7.090	1.609	1.479	1.012	
14	15	United Kingdom	7.054	1.333	1.538	0.996	
15	16	Ireland	7.021	1.499	1.553	0.999	
16	17	Germany	6.985	1.373	1.454	0.987	
17	18	Belgium	6.923	1.356	1.504	0.986	
18	19	United States	6.892	1.433	1.457	0.874	
19	20	Czech Republic	6.852	1.269	1.487	0.920	

An additional method to retrieve the number of rows and columns (in that respective order) in our dataset, is the command `happydf.shape`:

In [ ]:
```
# retrieve the number of rows and columns
# add your code here
```

Out[ ]: (156, 9)

Finally, to complete the review of our data set, we must verify **data completeness**. We can do this in Pandas by using the command `happydf.isnull().values.any()`. The command will review all the columns in the data set and will return `True` if there is any missing value in any of them. A `False` result, on the other hand, means that there are no null values in the dataset and that the data we have is complete.

> 📚 **Additional Material**: For more information, check the isnull() and the values.any() documentation.

In [ ]:
```
# check if there are missing values
# add your code here
```

Out[ ]: False

---

4. Plotting data in a Data Frame

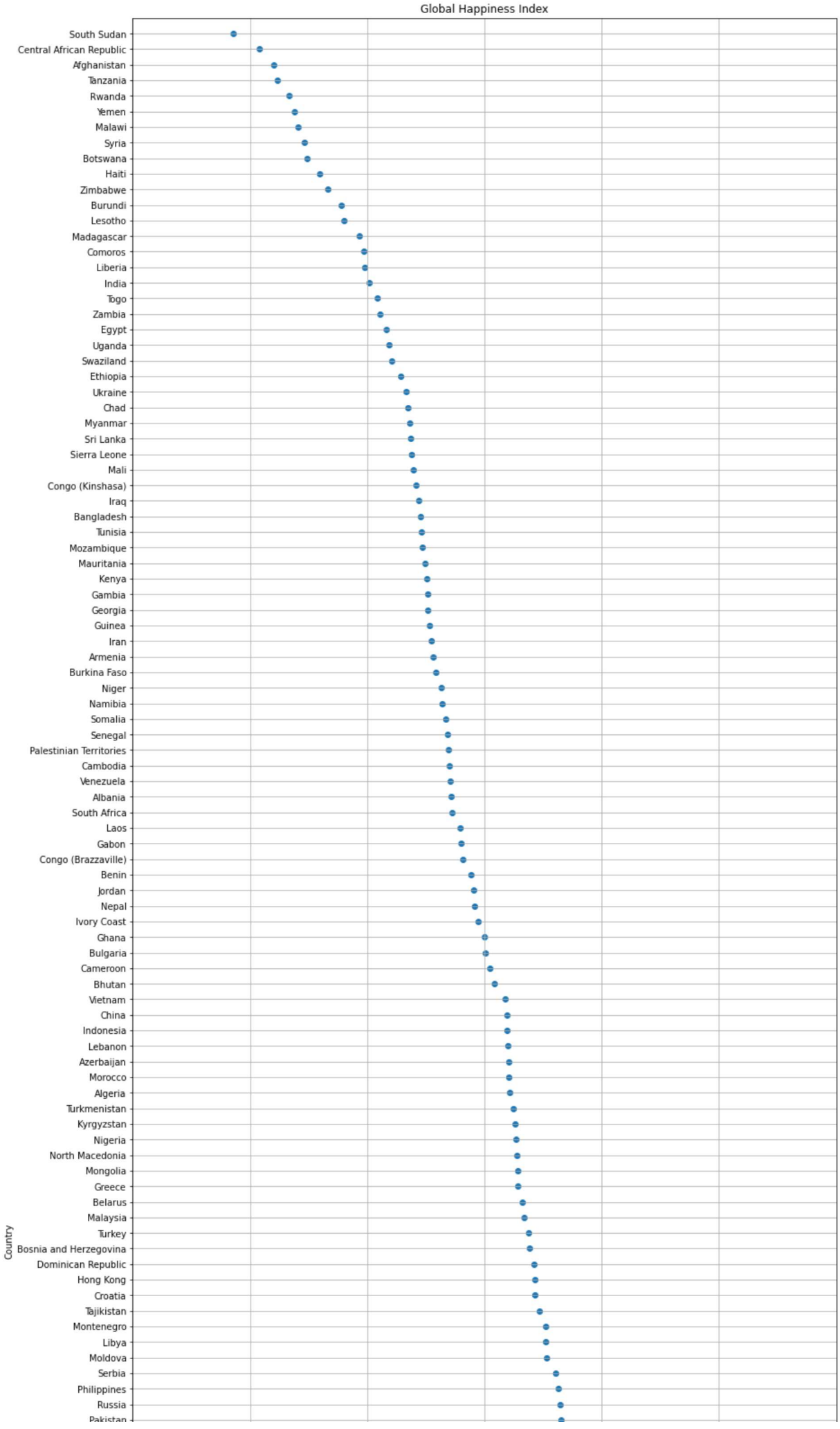
To start analyzing our data, we must first find a way of comprehensively visualize the most relevant data that may contribute to answer the proposed research question.

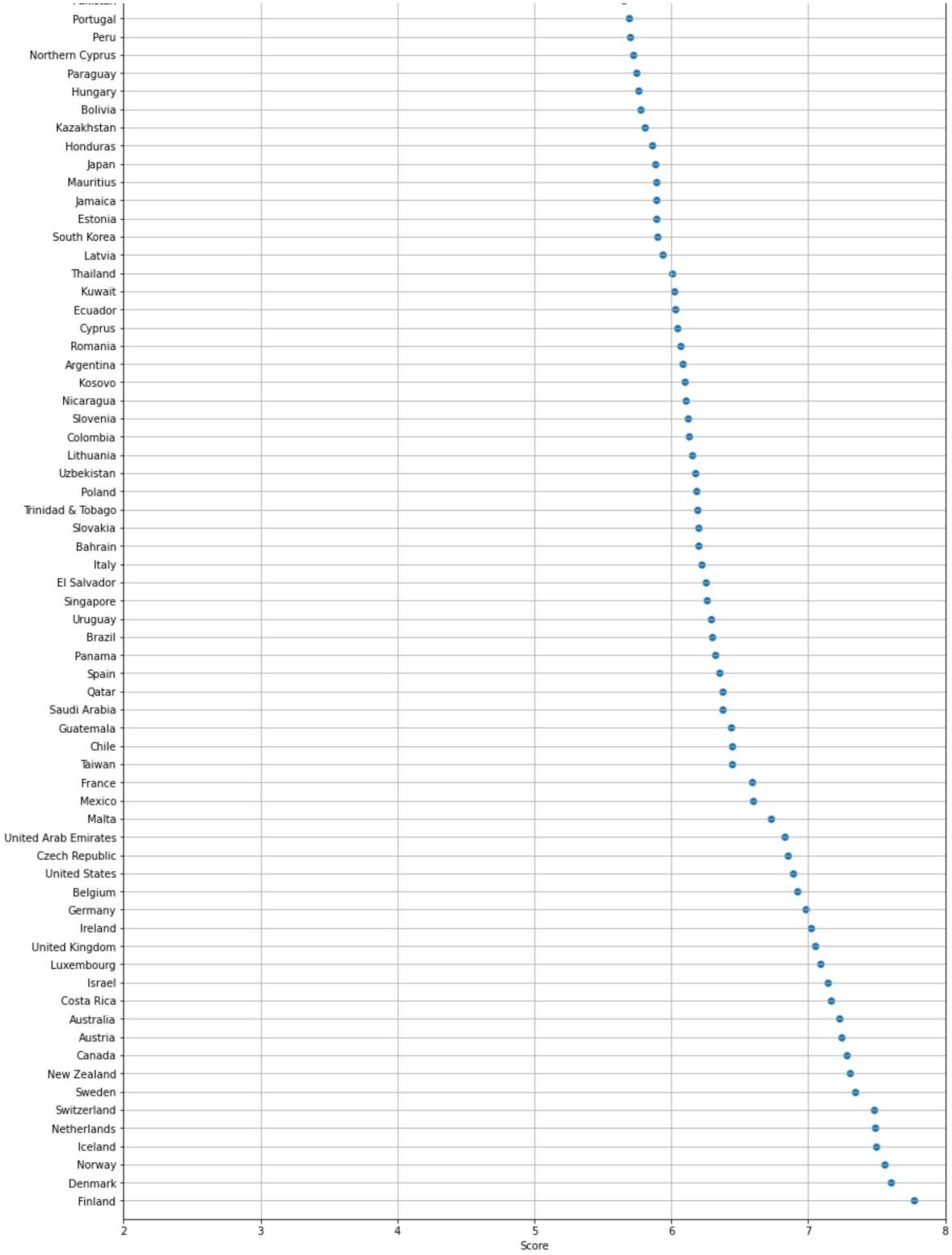
First, we will visualize the ***Global Happiness Index Score*** data.

> 💡 **Tip**: when visualizing large datasets, make sure to take advange of the medium in which the visualization is going to be displayed, and adjust the figure size accordingly. Remember *readibility* is a top priority when visualizing data.

In [ ]:
```
# visualize Global Happiness Index Score data
# add the code here
```

Out[ ]: <matplotlib.collections.PathCollection at 0x261576eefd0>

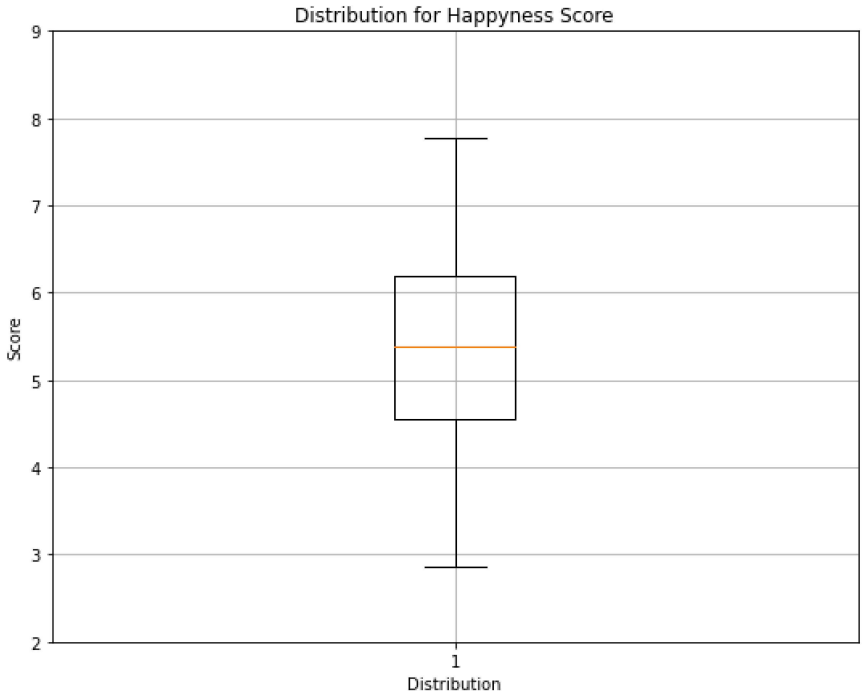

Global Happiness Index
Country
South Sudan
Central African Republic
Afghanistan
Tanzania
Rwanda
Yemen
Malawi
Syria
Botswana
Haiti
Zimbabwe
Burundi
Lesotho
Madagascar
Comoros
Liberia
India
Togo
Zambia
Egypt
Uganda
Swaziland
Ethiopia
Ukraine
Chad
Myanmar
Sri Lanka
Sierra Leone
Mali
Congo (Kinshasa)
Iraq
Bangladesh
Tunisia
Mozambique
Mauritania
Kenya
Gambia
Georgia
Guinea
Iran
Armenia
Burkina Faso
Niger
Namibia
Somalia
Senegal
Palestinian Territories
Cambodia
Venezuela
Albania
South Africa
Laos
Gabon
Congo (Brazzaville)
Benin
Jordan
Nepal
Ivory Coast
Ghana
Bulgaria
Cameroon
Bhutan
Vietnam
China
Indonesia
Lebanon
Azerbaijan
Morocco
Algeria
Turkmenistan
Kyrgyzstan
Nigeria
North Macedonia
Mongolia
Greece
Belarus
Malaysia
Turkey
Bosnia and Herzegovina
Dominican Republic
Hong Kong
Croatia
Tajikistan
Montenegro
Libya
Moldova
Serbia
Philippines
Russia
Pakistan



We can further explore the data stored in the dataset by visualizing different properties of the data that may be relevant to address the research question. For example the *data distribution* with a box plot, or *frequencies* with a histogram:

In [ ]:
```
# visualize the data distribution of Score
# add the code here
```

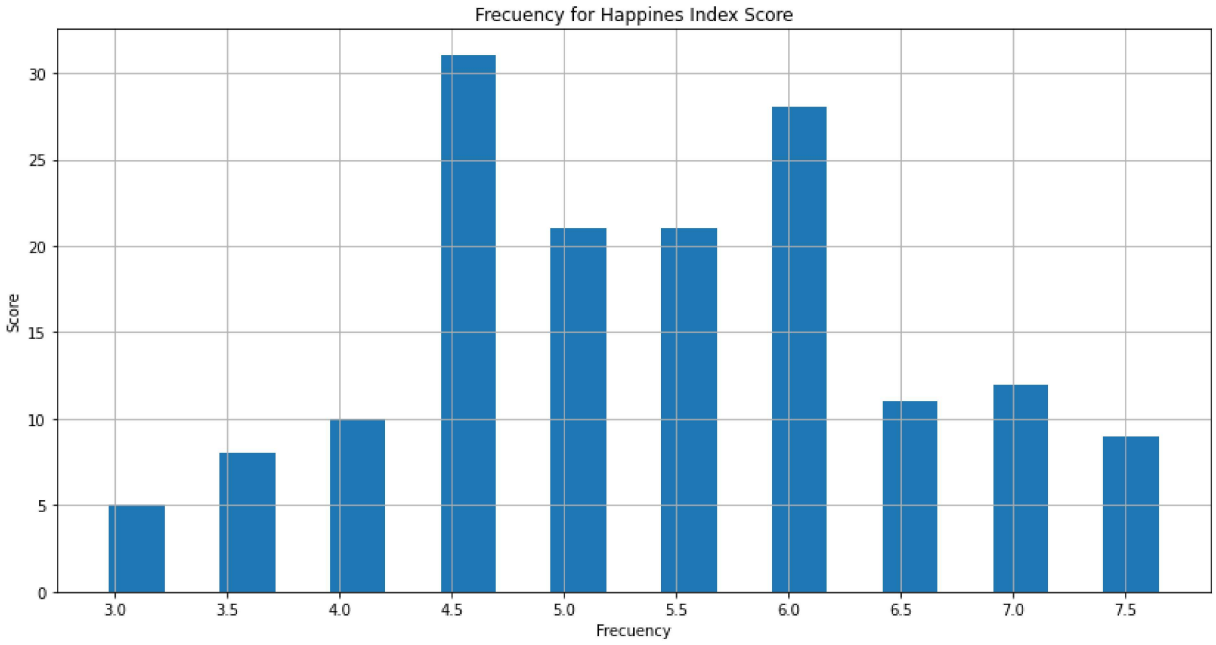
Out[ ]:
```
{'whiskers': [<matplotlib.lines.Line2D at 0x26157dfab20>,
  <matplotlib.lines.Line2D at 0x26157dfadf0>],
 'caps': [<matplotlib.lines.Line2D at 0x26157e0f1c0>,
  <matplotlib.lines.Line2D at 0x26157e0f3d0>],
 'boxes': [<matplotlib.lines.Line2D at 0x26157dfa850>],
 'medians': [<matplotlib.lines.Line2D at 0x26157e0f6a0>],
 'fliers': [<matplotlib.lines.Line2D at 0x26157e0f970>],
 'means': []}
```



For the histogram, we can tell Matplotlib how many intervals we wish the data to be divided in. The `ax.hist` command will then automatically calculate the number of values that are repeated within each interval. In this example, we divide the data into `10` intervals, and we modify the width of the bar using the `rwidth` parameter.

In [ ]:
```
# create a histogram of Score data
# add the code here
```

Out[ ]:
```
(array([ 5.,  8., 10., 31., 21., 21., 28., 11., 12.,  9.]),
 array([2.853 , 3.3446, 3.8362, 4.3278, 4.8194, 5.311 , 5.8026, 6.2942,
        6.7858, 7.2774, 7.769 ]),
 <BarContainer object of 10 artists>)
```



Now that we are able to visualize our dataset, we can start comapring different categorical variables against one another, and gather evidence to accept or reject our hypothesis.

---

5. Regressions

For the next example, we will compare the ***Global Happiness Index Score*** agains the ***GDP*** of each country.

In [ ]:
```
# compare Global Happiness Index Score vs. GDP of each country
# add the code here
```

Out[ ]:
```
<matplotlib.collections.PathCollection at 0x26158b3bac0>
```

The scatter plot suggests a *directly proportional* relationship between GDP and the Happiness Score: the higher the GDP of the country, the higher the Happiness Score.

To highlight this relationship, we can calculate a **linear regression** and plot it together with the scatter plot. To do this we will use the `np.polyfit` method from Numpy, and store the results in `data_linreg`. As parameters for `np.polyfit` we indicate the data from where the regression is calculated (GDP and Happiness Score), and the level of the regression (since we are focusing on a linear regression, the level is `1`).

> **Additional Material**: For more information, review the polyfit() documentation.

If we then print `data_linreg`, we see the values of the coefficients (the *slope* and the *intercept* respectively) of the linear model $y = a + bx$.

In [ ]:
```
# calculate the linear regression
# add your code here
```

```
[2.218148   3.39934518]
```

If we, for any reason, would like to save the regression data in the dataset, we will need to store these data in a unidimensional structure, and then insert it as a column in our data set.

To store the regression data in a unidimensional structure, we can use the `np.poly1d` function from Numpy. In this example, we use `Linreg` to store this data.

Finally, we can insert `Linreg` as a column in our dataset using the `happy.insert` command from Pandas. As paremeters, we indicate pandas after which column the data should be inserted (`4` in this example), the name of the column (`'Lin_Reg_Col'`), and finally the regression data (`Linreg(happyde.GDP_per_Capita)`).

> **Additional Material**: For more information, review the poly1d() and the insert documentation.

In [ ]:
```
# insert the linear regression results as a column
# add your code here
```

In [ ]:
```
# print the dataset
# add your code here
```

Out[ ]:

	Overall_rank	Country_or_region	Score	GDP_per_capita	Lin_Reg_Col	Social_support	Healthy_life_expectancy
0	1	Finland	7.769	1.340	6.371663	1.587	0.986
1	2	Denmark	7.600	1.383	6.467044	1.573	0.996
2	3	Norway	7.554	1.488	6.699949	1.582	1.028
3	4	Iceland	7.494	1.380	6.460389	1.624	1.026
4	5	Netherlands	7.488	1.396	6.495880	1.522	0.999
...	...	...	...	...	...	...	...
151	152	Rwanda	3.334	0.359	4.195660	0.711	0.614
152	153	Tanzania	3.231	0.476	4.455184	0.885	0.499
153	154	Afghanistan	3.203	0.350	4.175697	0.517	0.361
154	155	Central African Republic	3.083	0.026	3.457017	0.000	0.105
155	156	South Sudan	2.853	0.306	4.078098	0.575	0.295

156 rows × 10 columns

With the regression data calculated and stored, we simply need to plot the variable `data_linreg` together with the data from the previous scatter plot. To do this, we will add 2 variables to our previous plot. The variable `lr` will store the data from the function `np.polyval` from Numpy, which evaluates the values of a polynomial expression, using the `data_linreg` variable. Additionally, to establish a smooth plotting, we will create linear space for the X axis using `np.linspace`, and store it in `xp`.

> **Additional Material**: For more information, check the polyval() and the linspace() documentation.

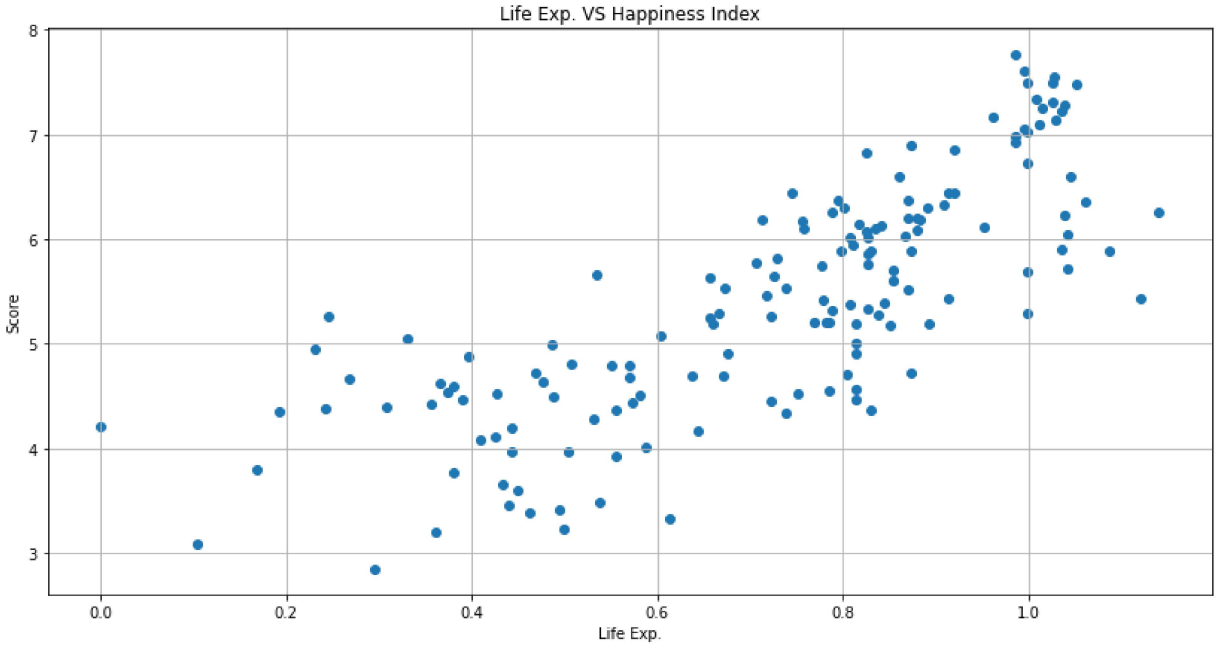
In [ ]:
```
# plot the linear regression data
# add your code here
```

Out[ ]:
```
[<matplotlib.lines.Line2D at 0x261598657f0>]
```

Now we have a look at other categorical variables in the dataset. We will plot now ***Healthy Life Expectancy*** against the ***Happiness Index Score***, and analyze the type of relationship between them.

In [ ]:
```
# plot a different graph between columns Healthy Life Expectancy vs. Happiness Index Score
# add the code here
```

Out[ ]:
```
<matplotlib.collections.PathCollection at 0x261589794f0>
```



For Healthy Life Expectancy and the Index Score, even if the data suggest a directly proportional relationship between these variables, it seems that the rate at which the score grows increases when the life expectancy surpasses 0.6. Therefore, in this scenario, a polynomial regression will offer a more accurate representation of this data behavior than a linear regression.

Given this, we calculate a second-level polynomial regression using `np.polyfit`.

In [ ]:
```
# calculate a second-level polynomial regression
# add your code here
```

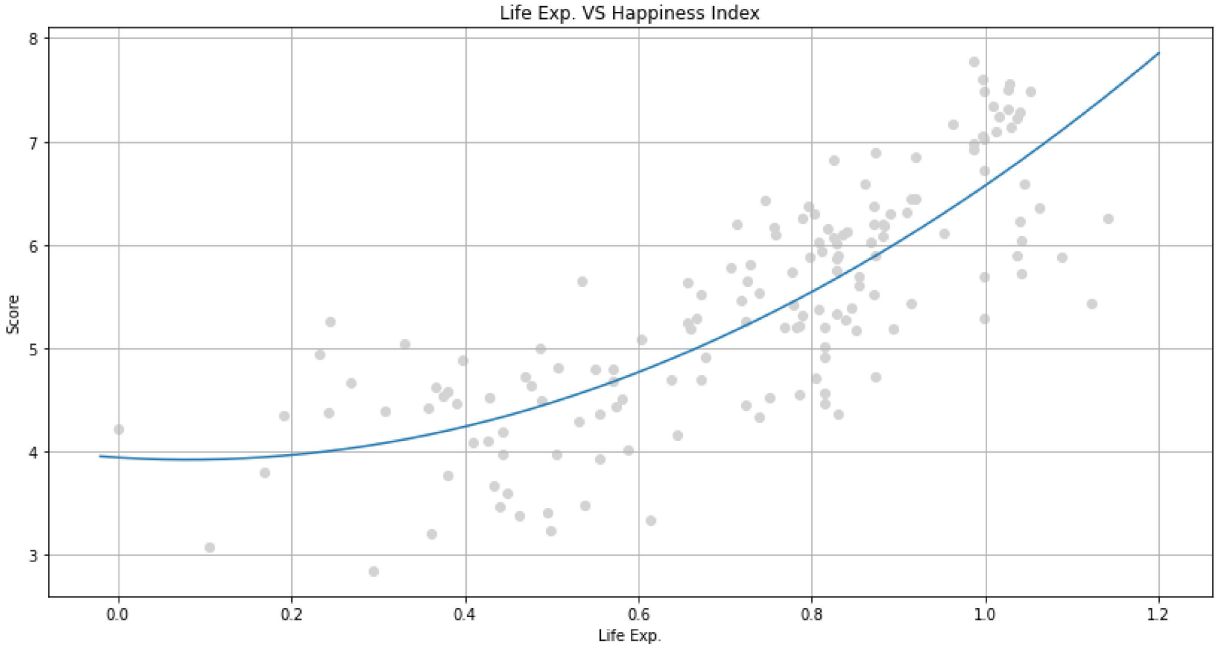
Out[ ]:
```
array([ 3.14015622, -0.50866522,  3.94143987])
```

If we print the result from the `np.polyfit` function we obtain the four coeficients for the cubic model $y = a + bx + cx^2$.

We now repeat the same process we did in our lineal model example to plot the polynomial regression.

In [ ]:
```
# plot the polinomial regression
# add your code here
```

Out[ ]:
```
[<matplotlib.lines.Line2D at 0x26157f92e80>]
```



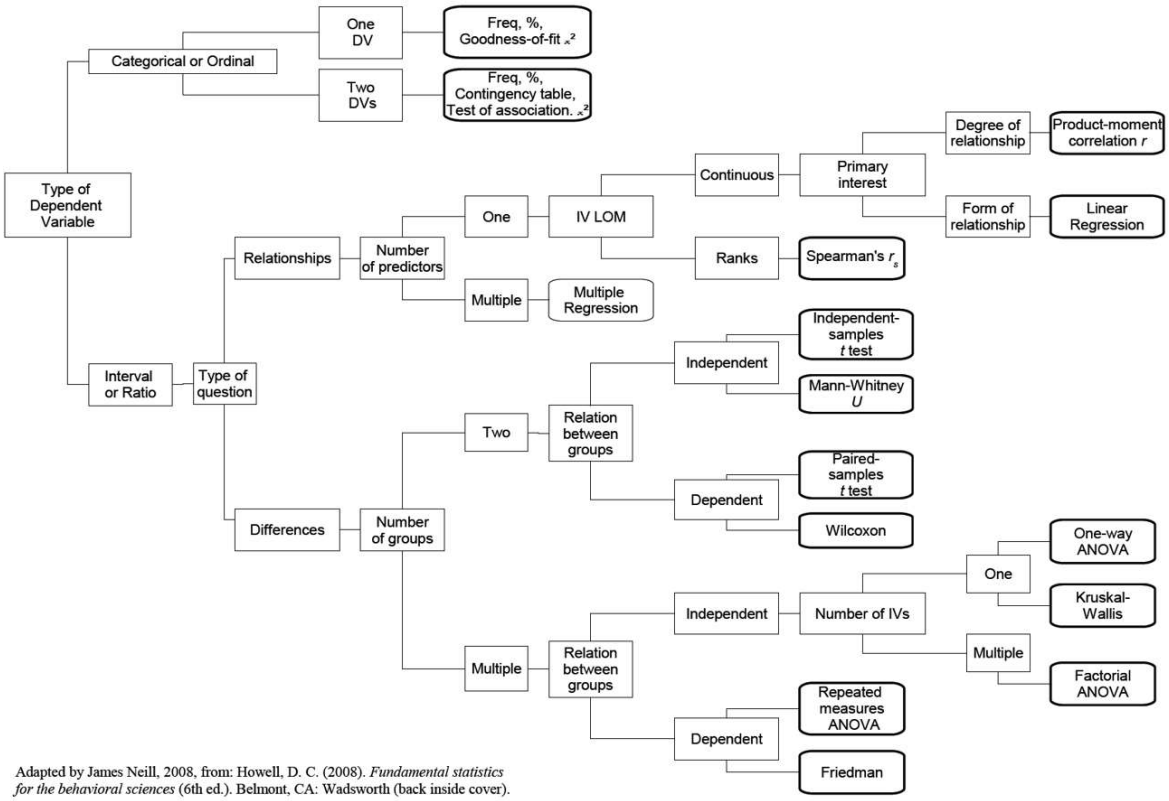
> 💡 **Tip**: Regressions are not only a statistical method. Regressions are also commonly used in machine learning, in the implementation of classifiers. In that area, the challenge is to design a regression model (called hypothesis) that can differentiate positives and negatives inputs. Put special attention to this technique! 🤠

---

6. Statistical Significance Tests.

We have shown graphically the kind of relationship that the ***Global Happiness Index Score*** has with both, the ***Healthy Life Expectancy*** and the ***GDP***. However, we must conduct a statistical test to be certain that any assumptions or conclusions we present are significant.

To find an appropriate statistical test, we can refer to the following chart that was shared in the slides from Lecture 3:



Adapted by James Neill, 2008, from: Howell, D. C. (2008). *Fundamental statistics for the behavioral sciences* (6th ed.). Belmont, CA: Wadsworth (back inside cover).

From this chart, we can see that the *Product-Moment Correlation*, better known as the ***Pearson Correlation Coefficient***, is one suitable test to evaluate the degree of the relationship between the ***Global Happiness Index Score***, and the ***GDP*** and the ***Healthy Life Expectancy***.

To implement a Pearson Correlation Coefficient test, we use the SciPy method `scp.stats.pearsonr`, sending as parameters the data from `happydf.Score`, `happydf.GDP_per_capita`, and `happydf.Healthy_life_expectancy`. The method `scp.stats.pearsonr` returns a tuple with two values: the *correlation coefficient* r, and the *p-value* respectively. In this case, we are interested in the correlation coefficient r.

> 📚 **Additional Material**: For more information, review the scp.stats.pearsonr documentation.

In [ ]:
```
# evaluate the degree of the relationship between the Global Happiness Index Score, GDP, and Healthy Life
# add your code here
```

```
The Pearson Correaltion r for the Happiness Index score and the Healthy Life Expectancy is 0.779883149242
5831.
The Pearson Correaltion r for the Happiness Index score and the GDP is 0.7938828678781276.
```

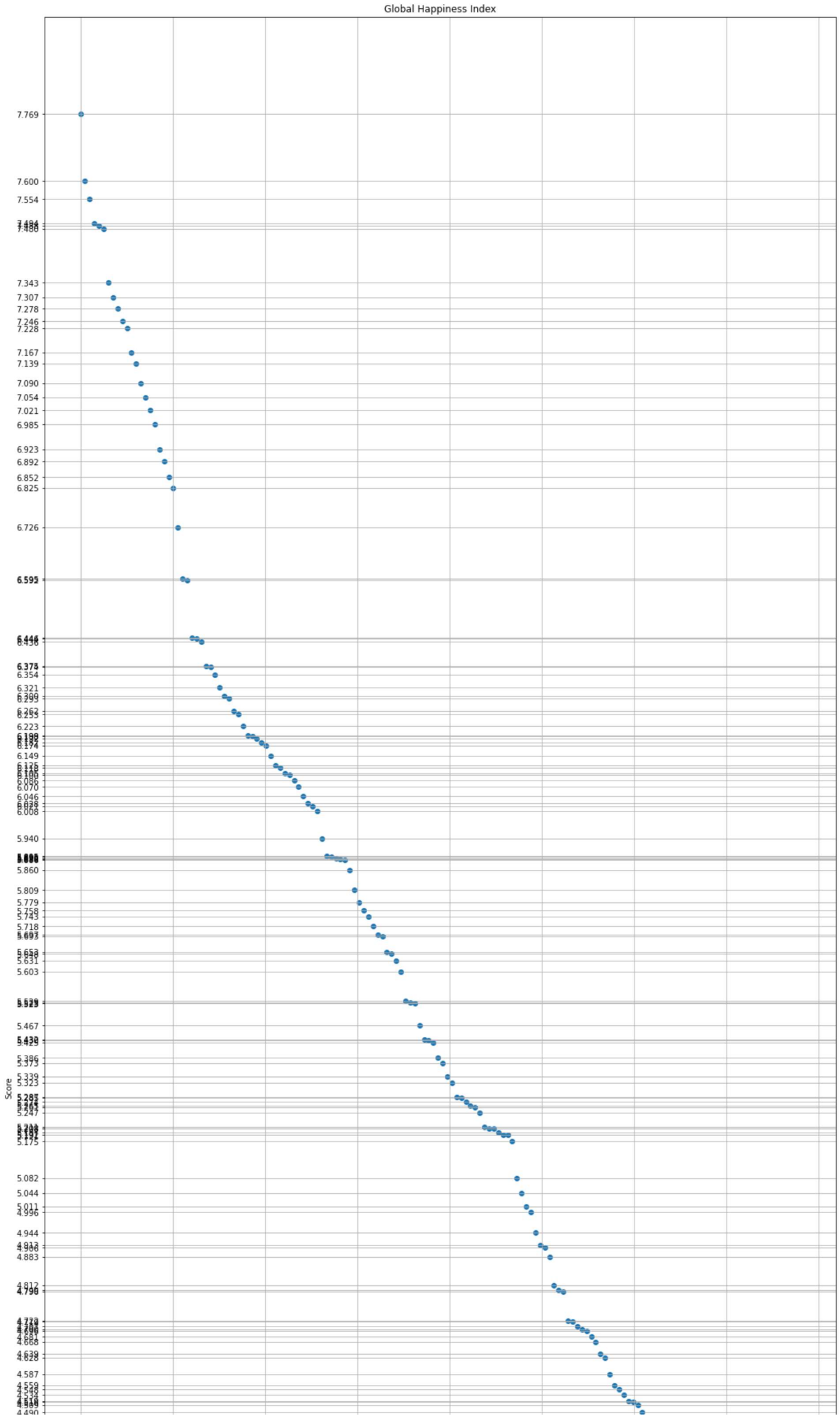
> 💡 **Tip**: The Pearson Correlation Coefficient r is a value between `0` and `1`, where `0` represents no linear correlation at all, and `1` represents a perfect linear correlation (all points fit within a single

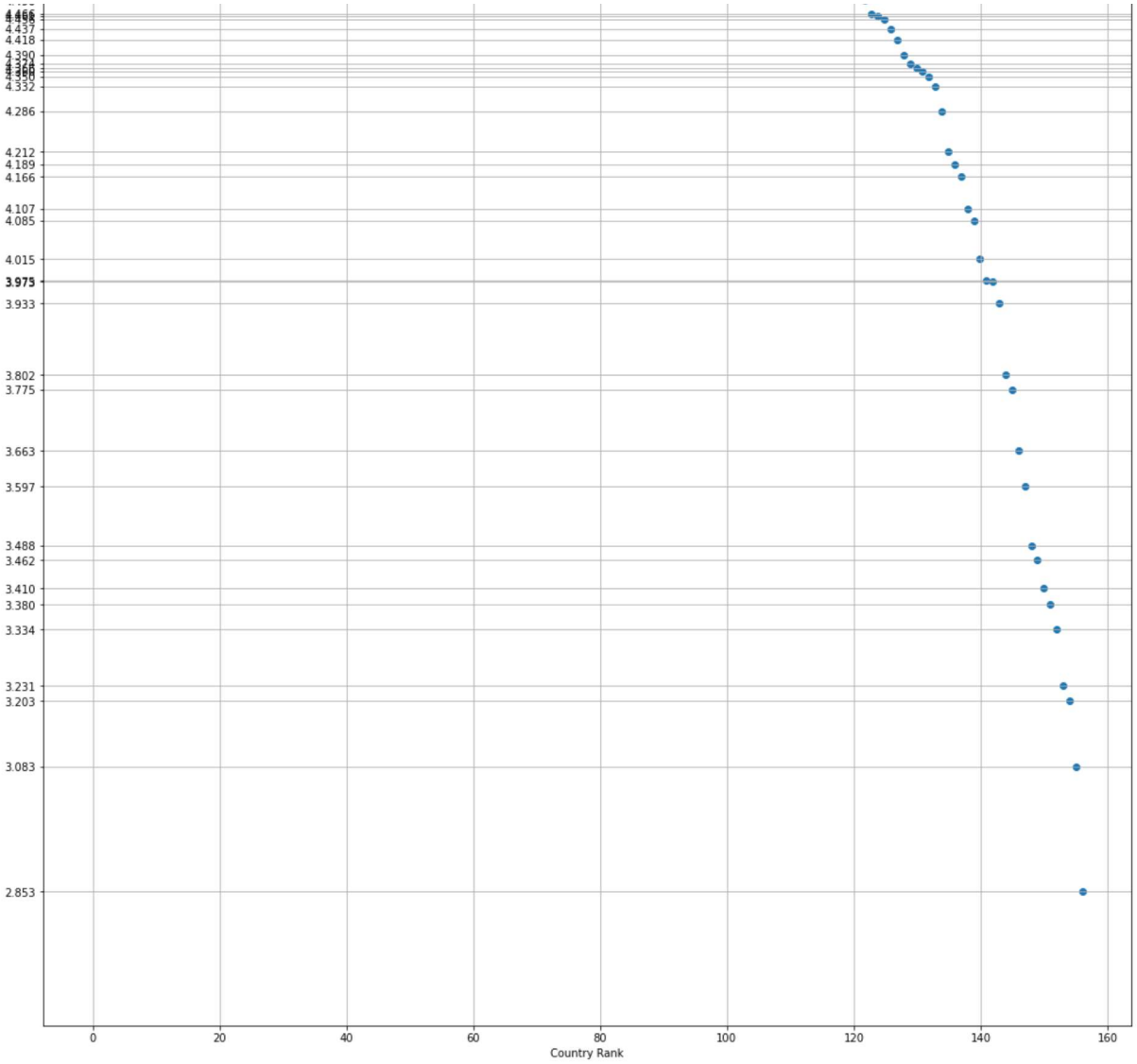
> straight line). It is considered that r results equal or greater than `0.8` show a strong linear relationship between the evaluated samples.

Now, to address the second research question, we will need to create intervals to define low, average, and high Scores in the Global Happiness Index. Let's plot this variable:

In [ ]:
```
# plot Country Rank vs. Score
# add your code here
```

Out[ ]:
```
<matplotlib.collections.PathCollection at 0x26159da13d0>
```


Global Happiness Index
Score
7.769
7.600
7.554
7.343
7.307
7.278
7.246
7.228
7.167
7.139
7.090
7.054
7.021
6.985
6.923
6.892
6.852
6.825
6.726
6.595
6.444
6.430
6.378
6.354
6.321
6.299
6.262
6.253
6.223
6.174
6.149
6.125
6.100
6.086
6.070
6.046
6.008
5.940
5.860
5.809
5.779
5.758
5.743
5.718
5.693
5.653
5.631
5.603
5.529
5.523
5.467
5.430
5.425
5.386
5.373
5.339
5.323
5.287
5.247
5.175
5.082
5.044
5.011
4.996
4.944
4.912
4.900
4.883
4.812
4.790
4.730
4.668
4.639
4.628
4.587
4.559
4.490



We can now see that there are 2 points in which there is a change of slope on the plot for the Scores in the dataset: one around **4.3**, and another around **6.5**. Once we have identified these intervals, we can use them to determine high, average, and low values in our data.

In [ ]:
```
# find all scores that are more than or equal to 4.3
# add your code here
```

Out[ ]:

	Overall_rank	Country_or_region	Score	GDP_per_capita	Lin_Reg_Col	Social_support	Healthy_life_expectancy
133	134	Ethiopia	4.286	0.336	4.144643	1.033	0.532
134	135	Swaziland	4.212	0.811	5.198263	1.149	0.000
135	136	Uganda	4.189	0.332	4.135770	1.069	0.443
136	137	Egypt	4.166	0.913	5.424514	1.039	0.644
137	138	Zambia	4.107	0.578	4.681435	1.058	0.426
138	139	Togo	4.085	0.275	4.009336	0.572	0.410
139	140	India	4.015	0.755	5.074047	0.765	0.588
140	141	Liberia	3.975	0.073	3.561270	0.922	0.443
141	142	Comoros	3.973	0.274	4.007118	0.757	0.505
142	143	Madagascar	3.933	0.274	4.007118	0.916	0.555
143	144	Lesotho	3.802	0.489	4.484020	1.169	0.168
144	145	Burundi	3.775	0.046	3.501380	0.447	0.380
145	146	Zimbabwe	3.663	0.366	4.211187	1.114	0.433
146	147	Haiti	3.597	0.323	4.115807	0.688	0.449
147	148	Botswana	3.488	1.041	5.708437	1.145	0.538
148	149	Syria	3.462	0.619	4.772379	0.378	0.440
149	150	Malawi	3.410	0.191	3.823011	0.560	0.495
150	151	Yemen	3.380	0.287	4.035954	1.163	0.463
151	152	Rwanda	3.334	0.359	4.195660	0.711	0.614
152	153	Tanzania	3.231	0.476	4.455184	0.885	0.499
153	154	Afghanistan	3.203	0.350	4.175697	0.517	0.361
154	155	Central African Republic	3.083	0.026	3.457017	0.000	0.105
155	156	South Sudan	2.853	0.306	4.078098	0.575	0.295

In [ ]:
```
# find all scores that are less than 6.5
# add your code here
```

Out[ ]:

	Overall_rank	Country_or_region	Score	GDP_per_capita	Lin_Reg_Col	Social_support	Healthy_life_expectancy
0	1	Finland	7.769	1.340	6.371663	1.587	0.986
1	2	Denmark	7.600	1.383	6.467044	1.573	0.996
2	3	Norway	7.554	1.488	6.699949	1.582	1.028
3	4	Iceland	7.494	1.380	6.460389	1.624	1.026
4	5	Netherlands	7.488	1.396	6.495880	1.522	0.999
5	6	Switzerland	7.480	1.452	6.620096	1.526	1.052
6	7	Sweden	7.343	1.387	6.475916	1.487	1.009
7	8	New Zealand	7.307	1.303	6.289592	1.557	1.026
8	9	Canada	7.278	1.365	6.427117	1.505	1.039
9	10	Austria	7.246	1.376	6.451517	1.475	1.016
10	11	Australia	7.228	1.372	6.442644	1.548	1.036
11	12	Costa Rica	7.167	1.034	5.692910	1.441	0.963
12	13	Israel	7.139	1.276	6.229702	1.455	1.029
13	14	Luxembourg	7.090	1.609	6.968345	1.479	1.012
14	15	United Kingdom	7.054	1.333	6.356136	1.538	0.996
15	16	Ireland	7.021	1.499	6.724349	1.553	0.999
16	17	Germany	6.985	1.373	6.444862	1.454	0.987
17	18	Belgium	6.923	1.356	6.407154	1.504	0.986
18	19	United States	6.892	1.433	6.577951	1.457	0.874
19	20	Czech Republic	6.852	1.269	6.214175	1.487	0.920
20	21	United Arab Emirates	6.825	1.503	6.733222	1.310	0.825
21	22	Malta	6.726	1.300	6.282938	1.520	0.999
22	23	Mexico	6.595	1.070	5.772764	1.323	0.861
23	24	France	6.592	1.324	6.336173	1.472	1.045

In [ ]:
```
# find all scores that are more than or equal to 4.3 and less than 6.5
# add your code here
```

Out[ ]:

	Overall_rank	Country_or_region	Score	GDP_per_capita	Lin_Reg_Col	Social_support	Healthy_life_expectancy
24	25	Taiwan	6.446	1.368	6.433772	1.430	0.914
25	26	Chile	6.444	1.159	5.970179	1.369	0.920
26	27	Guatemala	6.436	0.800	5.173864	1.269	0.746
27	28	Saudi Arabia	6.375	1.403	6.511407	1.357	0.795
28	29	Qatar	6.374	1.684	7.134706	1.313	0.871
...	...	...	...	...	...	...	...
128	129	Sierra Leone	4.374	0.268	3.993809	0.841	0.242
129	130	Sri Lanka	4.366	0.949	5.504368	1.265	0.831
130	131	Myanmar	4.360	0.710	4.974230	1.181	0.555
131	132	Chad	4.350	0.350	4.175697	0.766	0.192
132	133	Ukraine	4.332	0.820	5.218227	1.390	0.739

109 rows × 10 columns

In order to apply a statistical test that measures if the differences between two or more variables are significant, we must evaluate two things:

1. The factors and levels of factors that we are evaluating.
2. The ANOVA assumptions.

In this example, we have **1 factor (Score)** and **3 levels for that factor (low scores, average scores, and high scores).**

The ANOVA assumptions area set of premises that, if met, will allow us to apply parametric tests in our hypothesis testing. The ANOVA assumptions are:

1. Independence: every data point is sampled independently (the data set fulfills that).
2. Normality: data is normally distributed. When a dataset has 30 or more data points, you can assume that the data is normally distributed thanks to the central limit theorem. Otherwise, you can test the normality of your data by performing a Shapiro-Wilk test (we have 156 data points we assume normality in this case).
3. Homoscedasticity: variance among data samples is similar. We can test for homoscedasticity by applying the Levene's test or by using the Bartlett test.

> **Additional Material**: For a more detailed description of the previously mentioned methods, review the scipy.stats.shapiro, the scipy.stats.levene, and the scipy.stats.bartlett documentation.

Since assumptions 1 and 2 are met, let's test for homoscedasticity using `scp.levene`. The test will return 2 values: an *l-statistic value*, and a *p-value*. In this case, we are interested in the p-value. Specifically, if the **p-value is greater than 0.05**, since it will represent that the variance between the data samples is not significantly different.

In [ ]:
```
#Saving results in independent unidimensional structures
# add your code here
```

```
l-statistic = 13.43875083106697
p value = 4.279566491406237e-06
Different variance between samples
```

Since the p-value is **lesser** than 0.05, it means **that there is a statistically significant difference in the variance of the 3 samples** from the Global Happinnes Index score. Therefore, this ANOVA assumption is not met and we **can not apply a parametric test**.

To have a better perspective of the non-parametric test suitable for this scenario, we can refer to the table shared in the slides from Lecture 3:

Tests of Proportions

Samples	Response Categories	Tests
1	2	One-sample χ^2 test, binomial test
1	>2	One-sample χ^2 test, multinomial test
>1	≥2	*N*-sample χ^2 test, *G*-test, Fisher's exact test

OTHER TESTS: Shapiro-Wilk; Kolmogorov-Smirnov; Levene's test; Brown-Forsythe test

Analyses of Variance

Factors	Levels	(B)etween or (W)ithin	Parametric Tests	Nonparametric tests
1	2	B	Independent-samples *t*-test	Mann-Whitney *U* test
1	>2	B	One-way ANOVA	Kruskal-Wallis test
1	2	W	Paired-samples *t*-test	Wilcoxon signed-rank test
1	>2	W	One-way repeated measures ANOVA	Friedman test
>1	≥2	B	Factorial ANOVA Linear Models (LM)	Aligned Rank Transform (ART) Generalized Linear Models (GLM)
>1	≥2	W	Factorial repeated measures ANOVA Linear Mixed Models (LMM)	Aligned Rank Transform (ART) Generalized Linear Mixed Models (GLMM)

Table of Analyses © 2016 by Jacob O. Wobbrock
The Information School, University of Washington
Seattle, WA USA 98195 | wobbrock@uw.edu

Knowing that we have 1 factor, 3 levels in the factor, and we must apply a non-parametric test, we have now to select the best suiting test for our example. In this case, the *Kruskal-Wallis* test or the *Friedman* test are good options to test our hypothesis. In this case, we use `scp.kurskal` to carry out the statistical test.

> **Additional Material**: For more details, check the scp.stats.kurskal documentation.

In [ ]:
```
# add your code here
```

```
f-statistic = 101.06973347937198
p value = 1.1297600964782507e-22
There is a statistical significance difference between samples (Accept H1, Reject H0)
```

Conclusion

Based on the previous analysis we can now present the final answers to our research questions.

Question 1

What relationship may exist between the Happiness Index Score and the GDP and life expectancy of the survey countries in 2019?

Based on the graphical evidence gathered through the regression analysis, we may accept H1 and reject H0, since life expectancy and GDP clearly **expose a directly proportional relationship with the happiness score**. However, if we care for linearity and we take into account the results from the Pearson correlation coefficient *r*, we may not be able to do this since the threshold of 0.8 was not surprassed by either variable.

Question 2

How does the Global Happyness Index variate between the countries with low, average, and high happiness scores?

Based on the plot from the Scores, we identify 2 points in which values increased noticeably. We used those points to determine evaluation intervals for the Score variable. We conducted an ANOVA assumption analysis and selected a Kruskal-Wallis test to test our hypothesis, accepting H1 and rejecting H0: *The values of high, average, and low countries **are significantly different** between one another.*

This concludes Lab II. 🥳🎉

When combined together, Matplotlib, Pandas, Numpy, and Scypy become a very comprehensive and powerful set of tools for data science and machine learning applications. Make sure to familiarize yourself with those libraries, and to take advantage of all their potential! 🤯

> 📚 **Additional Material**: For a complete overview of these libraries, check the Pandas official documentation, the Numpy official documentation, and the Scipy official documentation.
