GDP and life satisfaction

May 14, 2021

[ ]:
```
# Here are imported the libraries/modules that are used below for the analysis.

%matplotlib inline
import numpy as np
import matplotlib.pyplot as plt
import scipy as sp
import pandas as pd
import sklearn as sk
```

[ ]:
```
# Here I use my local file path on reading the .csv files.
# The reader must use the appropriate path where files are located in their␣
↪computers.

# I select as header the third row of the file gdp per capita (n=2). The␣
↪delimiter is ","
GDP=pd.read_csv("/Users/damianejlli/Downloads/gdp per capita.csv",␣
↪delimiter=",", header=2)

# There is no need to specify the header for the "better life index.csv" file.
LS=pd.read_csv("/Users/damianejlli/Downloads/better life index.csv")
```

[ ]:
```
#I display some of the content of the GDP dataframe.

GDP.head()
```

[ ]:
```
# I select the columns "Country Name" and year "2015" for the analysis in the␣
↪GDP dataframe.

GPD1=GDP.loc[:,["Country Name", '2015']]
```

[ ]:
```
# I display the GDP1 dataframe content.

GPD1
```

[ ]:
```
# I set as index the "Country Name" column and rename the column "2015" to "GPD␣
↪per capita 2015 (USD)"
```

```
GDP2=GPD1.set_index("Country Name").rename(columns={"2015": "GPD per capita␣
↪2015 (USD)"})
```

```
[ ]: # I show the first ten rows of the GDP2 dataframe as a matter of example

GDP2.head(10)
```

```
[ ]: # I show the first ten rows of the LS dataframe as a matter of example.

LS.head()
```

```
[ ]: # I show the shape of the LS dataframe. It has 2369 rows and 17 columns.

LS.shape
```

```
[ ]: # I use a conditional to choose all those rows with values "Life satisfaction"␣
↪in the column "Indicator"
# and all those equal to "TOT" in the "INEQUALITY" column in the LS dataframe.
# "TOT" is the total value of life satisfaction for men and women in a given␣
↪country.

LS1=LS[(LS["Indicator"]=="Life satisfaction") & (LS["INEQUALITY"]=="TOT")]
```

```
[ ]: # I show the first 10 entries of the LS1 dataframe as a matter of exmple.

LS1.head(10)
```

```
[ ]: # First, in the LS1 dataframe, I rename the columns "Country" and "Value"␣
↪respectively to "Country Name" and "Life Satisfaction Value".
# Second, I set as index of the new dataframe the "Country Name" and after I␣
↪select all rows in the "Indicator" column
# with entries equal to "Life Satisfaction Value".

LS2=LS1.rename(columns={"Country" : "Country Name", "Value": "Life Satisfaction␣
↪Value"}).set_index("Country Name").loc[:, ["Life Satisfaction Value"]]
```

```
[ ]: # I show the first 10 entries of the LS2 dataframe as a matter of exmple.

LS2.head(10)
```

```
[ ]: # I remove the entry "OECD-Total" country index from the LS2 dataframe because␣
↪it is unneccessary for the analysis.

LS3=LS2[LS2.index != "OECD - Total"]
```

[ ]:
```
# I show the first 10 entries of the LS3 dataframe as a matter of exmple.

LS3.head(10)
```

[ ]:
```
# I join the LS3 dataframe with the GPD2 dataframe in order to form the final␣
↪dataframe, df.

df=LS3.join(GDP2)
```

[ ]:
```
# I display the entries in the joint dataframe, df.

df
```

[ ]:
```
# I remove the NaN values from the "df" dataframe to form the final dataframe␣
↪for the analysis, "df1".

df1=df.dropna()
```

[ ]:
```
# I display the df1 dataframe.

df1
```

[ ]:
```
# I calculate the shape of the df1 dataframe. The dataframe has 38 rows and 2␣
↪columns.

df1.shape
```

[ ]:
```
# I create a scatter plot for the data in the df1 dataframe.

df1.plot(kind="scatter", x="GPD per capita 2015 (USD)", y="Life Satisfaction␣
↪Value", color="b", figsize=(10,6))
```

[ ]:
```
# I calculate the Pearson correlation coeffeicient r for the data in the df1␣
↪dataframe
# and display the correlation dataframe.

df1.corr()
```

[ ]:
```
# I extract all values of the "GPD per capita 2015 (USD)" and "Life␣
↪Satisfaction Value" columns and
# form new (38x1) column arrays "a" and "b".

a=df1.loc[:, ["GPD per capita 2015 (USD)"]].values
b=df1.loc[:, ["Life Satisfaction Value"]].values
```

[ ]:
```
# I reshape the original (38x1) column arrary "a" to a (1x38) row array "X".

X=a.reshape(38)
```

[ ]:
```
# I display the "X" array.

X
```

[ ]:
```
# I reshape the original (38x1) column "b" array to a (1x38) row array "y".

y=b.reshape(38)
```

[ ]:
```
# I display the "y" array.
y
```

[ ]:
```
# First, I assume a simple linear regression model for the data in "X" and "y"␣
↪arrays
# and calculate the slope, intercept etc., of the linear regression method.
# Here I use the "stats" module of "Scipy" library and its linear regression␣
↪built in method.

result = sp.stats.linregress(X, y)
```

[ ]:
```
# I print the results of the simple linear regression method.

print(result)
```

[ ]:
```
# I create a figure with a single subplot where the original data of the df1␣
↪dataframe
# and the linear regression line Y(X) are shown.

fig, ax=plt.subplots(figsize=(10, 6))
ax.scatter(X, y, color='b', label="Original data")
ax.plot(X, result.intercept + (result.slope)*X, color="m", label="Linear␣
↪regression line: $Y(X)=5.74+2.39\cdot 10^{-5} X$")
ax.set_xlabel("GPD per capita 2015 (USD)")
ax.set_ylabel("Life Satifaction Value")
plt.legend()
```

[ ]:
```
# I calculate the t-score in order to estimate the Confidence Intervals (CIs)
# of the linear regression coefficients "beta_0" and "beta_1" at significance
# level of alpha=0.05 and Confidence Level (CL) of 95%. The number of degrees␣
↪of freedom for the data is n=38.

n=38
alpha=0.05
```

```
t_score = sp.stats.t.ppf(1-alpha/2, n-2)
print(t_score)
```

[ ]:
```
# Second, I use the KNN regression method to find a relationship between the␣
↪data for K=5 (default value).

model=sk.neighbors.KNeighborsRegressor()
```

[ ]:
```
# I use the fit() function to fit the data of the KNN method
# and use the original column vectors "a" and "b" and redefine
# them as "X1" and "y1" to use in the KNN regression method.

X1=a
y1=b
model.fit(X1,y1)
```

[ ]:
```
# I calculate the predicted values of the KNN method for
# the GDP data "X" not present in the df1 dataframe for the countries of␣
↪Albania, United Arab Emirates and Armenia.

X_new=[[3607.296697],[38663.383807],[3607.296697]]
```

[ ]:
```
# I Print the predicted values of "Life Satisfaction Value" respectively
# for Albania, United Arab Emirates and Armenia.

print(model.predict(X_new))
```

[ ]:
```
# I print the value of the generalized correlation coefficient R^2.

print(model.score(X1, y1, sample_weight=None))
```