module-3

October 20, 2023

1 Module 3: Data Exploration (Homework 2)

The following tutorial contains examples of Python code for data exploration. You should refer to the "Data Exploration" chapter of the "Introduction to Data Mining" book (available at https://www-users.cs.umn.edu/~kumar001/dmbook/index.php) to understand some of the concepts introduced in this tutorial notebook. The notebook can be downloaded from http://www.cse.msu.edu/~ptan/dmbook/tutorials/tutorial3/tutorial3.ipynb.

Data exploration refers to the preliminary investigation of data in order to better understand its specific characteristics. There are two key motivations for data exploration: 1. To help users select the appropriate preprocessing and data analysis technique used. 2. To make use of humans' abilities to recognize patterns in the data.

Read the step-by-step instructions below carefully. To execute the code, click on the cell and press the SHIFT-ENTER keys simultaneously.

1.1 3.1. Summary Statistics

Summary statistics are quantities, such as the mean and standard deviation, that capture various characteristics of a potentially large set of values with a single number or a small set of numbers. In this tutorial, we will use the Iris sample data, which contains information on 150 Iris flowers, 50 each from one of three Iris species: Setosa, Versicolour, and Virginica. Each flower is characterized by five attributes:

- sepal length in centimeters
- sepal width in centimeters
- petal length in centimeters
- petal width in centimeters
- class (Setosa, Versicolour, Virginica)

In this tutorial, you will learn how to:

- Load a CSV data file into a Pandas DataFrame object.
- Compute various summary statistics from the DataFrame.

To execute the sample program shown here, make sure you have installed the Pandas library (see Module 2).

1. First, you need to download the Iris dataset from the UCI machine learning repository.

Code: The following code uses Pandas to read the CSV file and store them in a DataFrame object named data. Next, it will display the first five rows of the data frame.

[ ]:
```
import pandas as pd

data = pd.read_csv('http://archive.ics.uci.edu/ml/machine-learning-databases/
↪iris/iris.data',header=None)
data.columns = ['sepal length', 'sepal width', 'petal length', 'petal width',␣
↪'class']

data.head()
```

[ ]:

	sepal length	sepal width	petal length	petal width	class
0	5.1	3.5	1.4	0.2	Iris-setosa
1	4.9	3.0	1.4	0.2	Iris-setosa
2	4.7	3.2	1.3	0.2	Iris-setosa
3	4.6	3.1	1.5	0.2	Iris-setosa
4	5.0	3.6	1.4	0.2	Iris-setosa

2. For each quantitative attribute, calculate its average, standard deviation, minimum, and maximum values.

Code:

[ ]:
```
from pandas.api.types import is_numeric_dtype

for col in data.columns:
    if is_numeric_dtype(data[col]):
        print('%s:' % (col))
        print('\t Mean = %.2f' % data[col].mean())
        print('\t Standard deviation = %.2f' % data[col].std())
        print('\t Minimum = %.2f' % data[col].min())
        print('\t Maximum = %.2f' % data[col].max())
```

```
sepal length:
         Mean = 5.84
         Standard deviation = 0.83
         Minimum = 4.30
         Maximum = 7.90
sepal width:
         Mean = 3.05
         Standard deviation = 0.43
         Minimum = 2.00
         Maximum = 4.40
petal length:
         Mean = 3.76
         Standard deviation = 1.76
         Minimum = 1.00
         Maximum = 6.90
```

```
petal width:
        Mean = 1.20
        Standard deviation = 0.76
        Minimum = 0.10
        Maximum = 2.50
```

3. For the qualitative attribute (class), count the frequency for each of its distinct values.

Code:

```
[ ]: data['class'].value_counts()
```

```
[ ]: Iris-setosa        50
     Iris-versicolor    50
     Iris-virginica     50
     Name: class, dtype: int64
```

4. It is also possible to display the summary for all the attributes simultaneously in a table using the describe() function. If an attribute is quantitative, it will display its mean, standard deviation and various quantiles (including minimum, median, and maximum) values. If an attribute is qualitative, it will display its number of unique values and the top (most frequent) values.

Code:

```
[ ]: data.describe(include='all')
```

[ ]:

	sepal length	sepal width	petal length	petal width	class
count	150.000000	150.000000	150.000000	150.000000	150
unique	NaN	NaN	NaN	NaN	3
top	NaN	NaN	NaN	NaN	Iris-setosa
freq	NaN	NaN	NaN	NaN	50
mean	5.843333	3.054000	3.758667	1.198667	NaN
std	0.828066	0.433594	1.764420	0.763161	NaN
min	4.300000	2.000000	1.000000	0.100000	NaN
25%	5.100000	2.800000	1.600000	0.300000	NaN
50%	5.800000	3.000000	4.350000	1.300000	NaN
75%	6.400000	3.300000	5.100000	1.800000	NaN
max	7.900000	4.400000	6.900000	2.500000	NaN

Note that count refers to the number of non-missing values for each attribute.

5. For multivariate statistics, you can compute the covariance and correlation between pairs of attributes.

Code:

```
[ ]: print('Covariance:')
     data.cov()
```

```
Covariance:
```

```
<ipython-input-27-4f52c089a412>:2: FutureWarning: The default value of
numeric_only in DataFrame.cov is deprecated. In a future version, it will
default to False. Select only valid columns or specify the value of numeric_only
to silence this warning.
  data.cov()
```

[ ]:

	sepal length	sepal width	petal length	petal width
sepal length	0.685694	-0.039268	1.273682	0.516904
sepal width	-0.039268	0.188004	-0.321713	-0.117981
petal length	1.273682	-0.321713	3.113179	1.296387
petal width	0.516904	-0.117981	1.296387	0.582414

[ ]:

```python
print('Correlation:')
data.corr()
```

```
Correlation:
```

```
<ipython-input-28-1826941e9562>:2: FutureWarning: The default value of
numeric_only in DataFrame.corr is deprecated. In a future version, it will
default to False. Select only valid columns or specify the value of numeric_only
to silence this warning.
  data.corr()
```

[ ]:

	sepal length	sepal width	petal length	petal width
sepal length	1.000000	-0.109369	0.871754	0.817954
sepal width	-0.109369	1.000000	-0.420516	-0.356544
petal length	0.871754	-0.420516	1.000000	0.962757
petal width	0.817954	-0.356544	0.962757	1.000000

1.2 3.2. Data Visualization

Data visualization is the display of information in a graphic or tabular format. Successful visualization requires that the data (information) be converted into a visual format so that the characteristics of the data and the relationships among data items or attributes can be analyzed or reported.

In this tutorial, you will learn how to display the Iris data created in Section 3.1. To execute the sample program shown here, make sure you have installed the matplotlib library package (see Module 0 on how to install Python packages).

1. First, we will display the histogram for the sepal length attribute by discretizing it into 8 separate bins and counting the frequency for each bin.

Code:

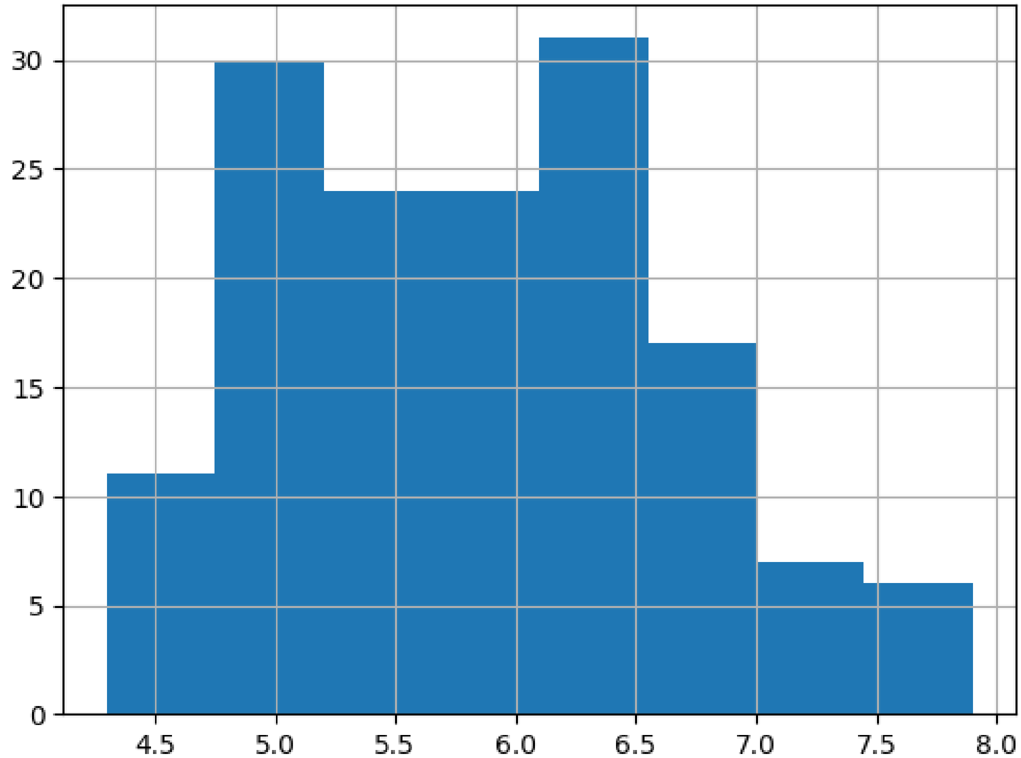
[ ]:

```python
%matplotlib inline

data['sepal length'].hist(bins=8)
```

[ ]:

```
<Axes: >
```

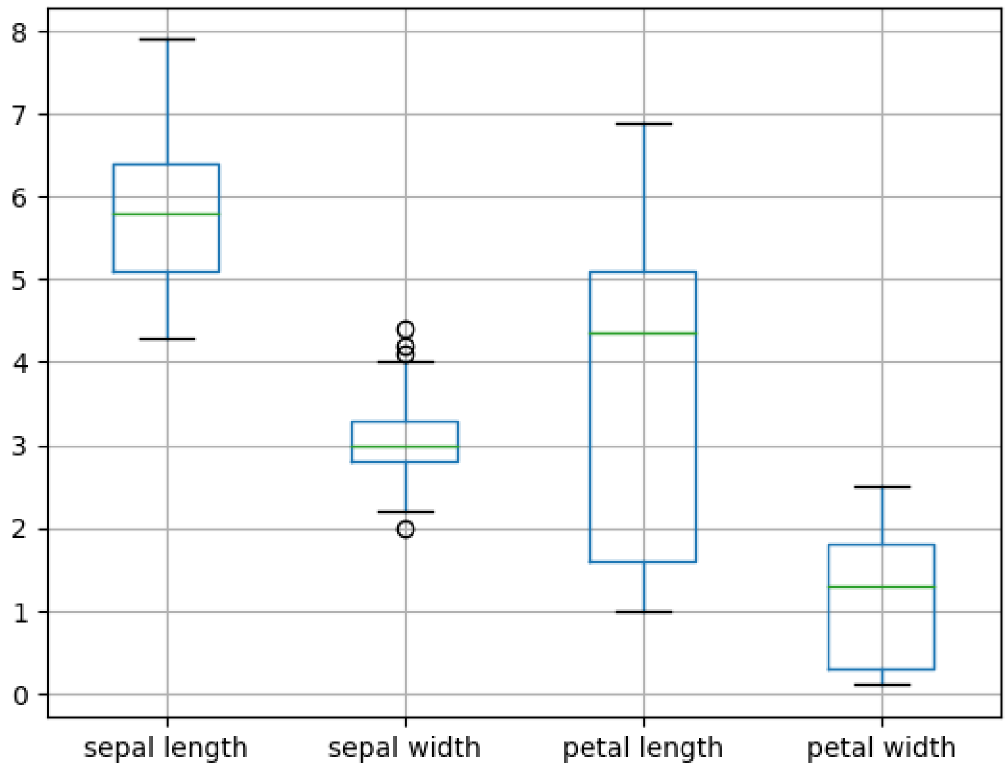


2. A boxplot can also be used to show the distribution of values for each attribute.

Code:

```
[ ]: data.boxplot()
```

```
[ ]: <Axes: >
```

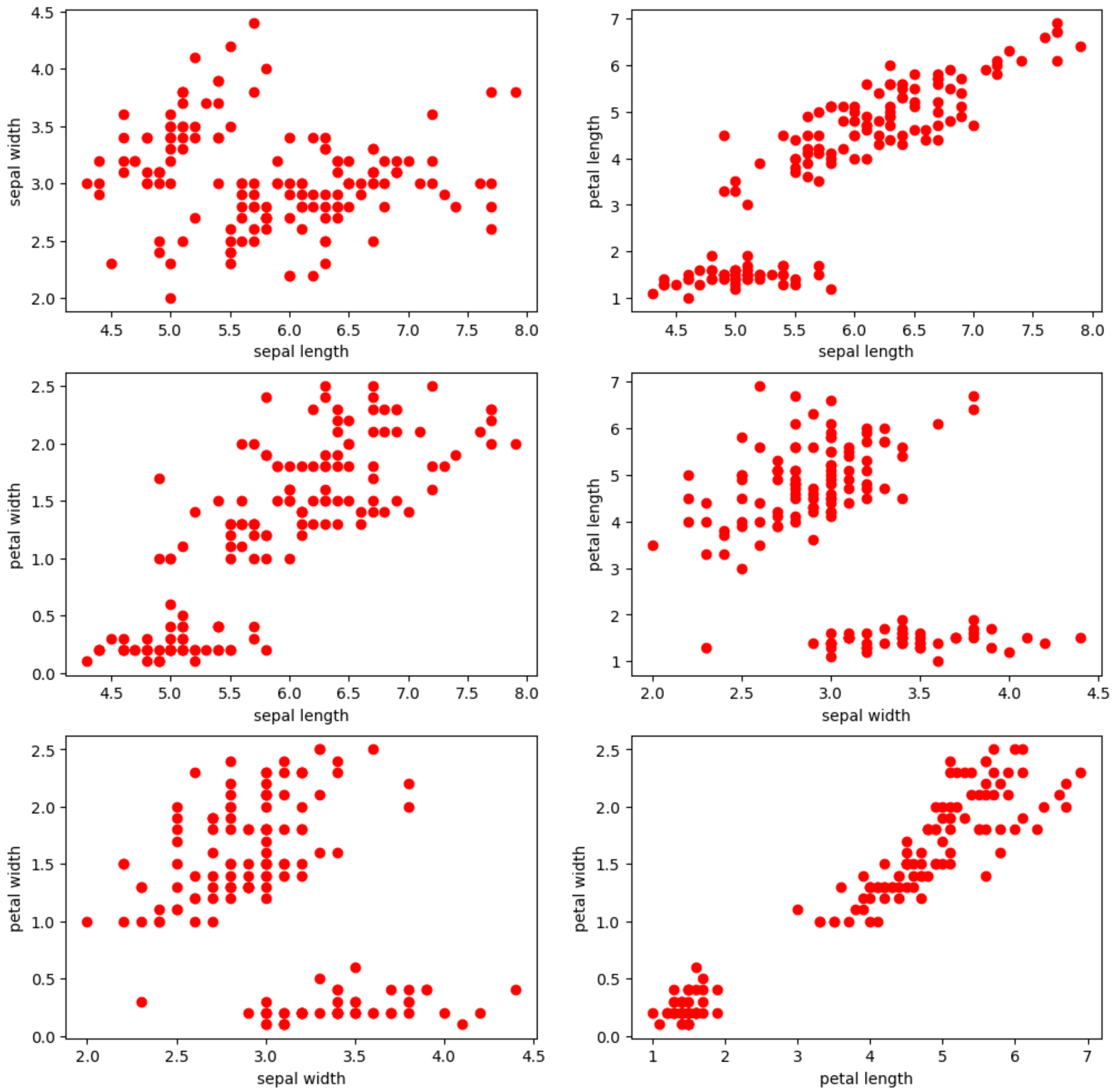


3. For each pair of attributes, we can use a scatter plot to visualize their joint distribution.

Code:

```
[ ]: import matplotlib.pyplot as plt

fig, axes = plt.subplots(3, 2, figsize=(12,12))
index = 0
for i in range(3):
    for j in range(i+1,4):
        ax1 = int(index/2)
        ax2 = index % 2
        axes[ax1][ax2].scatter(data[data.columns[i]], data[data.columns[j]],␣
 ↪color='red')
        axes[ax1][ax2].set_xlabel(data.columns[i])
        axes[ax1][ax2].set_ylabel(data.columns[j])
        index = index + 1
```



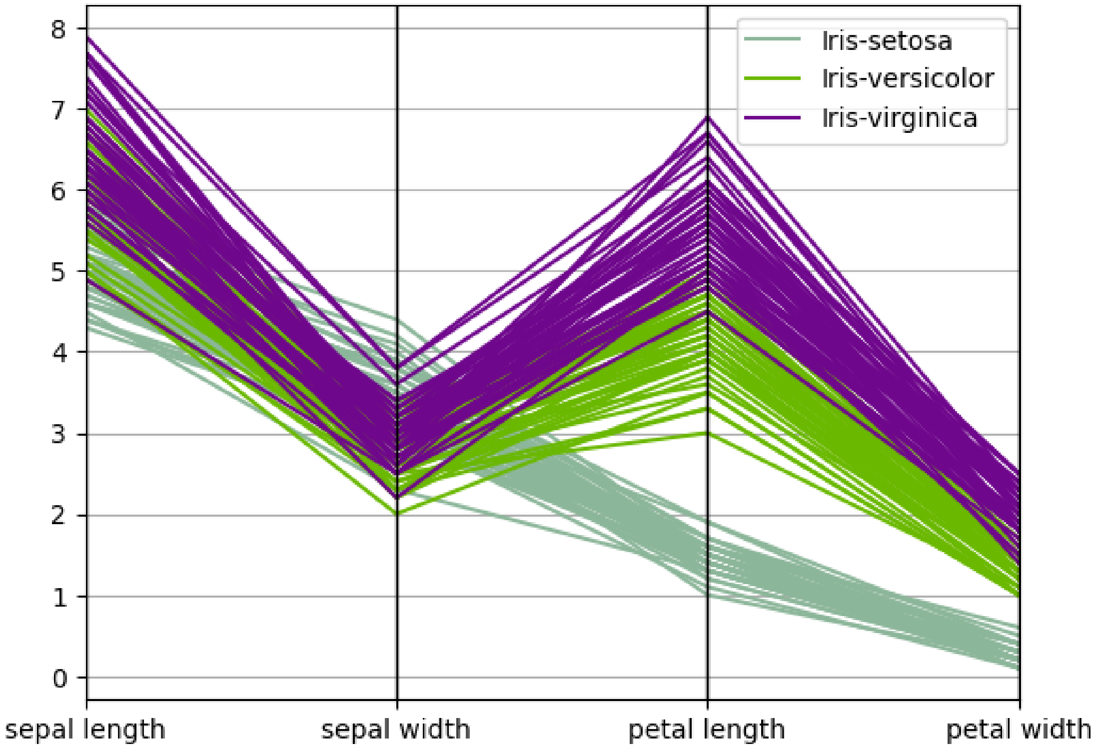
4. Parallel coordinates can be used to display all the data points simultaneously. Parallel coordinates have one coordinate axis for each attribute, but the different axes are parallel to one other instead of perpendicular, as is traditional. Furthermore, an object is represented as a line instead of as a point. In the example below, the distribution of values for each class can be identified in a separate color.

Code:

[ ]:
```
from pandas.plotting import parallel_coordinates
%matplotlib inline

parallel_coordinates(data, 'class')
```

[ ]: <Axes: >



1.3 3.3. Summary

This tutorial presents several examples for data exploration and visualization using the Pandas and matplotlib library packages available in Python.

References:

1. Documentation on Pandas. https://pandas.pydata.org/
2. Documentation on matplotlib. https://matplotlib.org/
3. Lichman, M. (2013). UCI Machine Learning Repository [http://archive.ics.uci.edu/ml]. Irvine, CA: University of California, School of Information and Computer Science.

1.4 In-clas (Homework 2)

Please do not manually look for answers even if you can.

[ ]:
```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

[ ]:
```
import pandas as pd
import numpy as np
```

[ ]:
```
happiness_df = pd.read_csv('/content/drive/MyDrive/shared/happiness_2017.csv')
happiness_df.head()
```

[ ]:
```
       Country          Region  Rank  HappinessScore  Life Ladder  \
0       Norway  Western Europe     1           7.537     7.578745
1      Denmark  Western Europe     2           7.522     7.593702
2      Iceland  Western Europe     3           7.504     7.476214
3  Switzerland  Western Europe     4           7.494     7.473593
4      Finland  Western Europe     5           7.469     7.788252

   Log GDP per capita  Social support  Healthy life expectancy at birth  \
0           11.081789        0.950128                         71.086586
1           10.748989        0.952100                         71.662498
2           10.760409        0.966753                         72.755981
3           10.955548        0.949661                         73.173759
4           10.612338        0.963826                         71.696960

   Freedom to make life choices  Generosity  Perceptions of corruption  \
0                      0.953017    0.210104                   0.249711
1                      0.955416    0.145387                   0.181148
2                      0.938783    0.235479                   0.726845
3                      0.924997    0.167875                   0.316183
4                      0.962199   -0.012174                   0.192413

   Positive affect  Negative affect  Confidence in national government
0         0.849100         0.202914                           0.717160
1         0.823667         0.205775                           0.572353
2         0.895255         0.148160                           0.365042
3         0.773997         0.195871                           0.819707
4         0.787137         0.176066                           0.597539
```

[ ]:
```
print(happiness_df.shape)
happiness_df.columns
```

```
(140, 14)
```

[ ]:
```
Index(['Country', 'Region', 'Rank', 'HappinessScore', 'Life Ladder',
       'Log GDP per capita', 'Social support',
       'Healthy life expectancy at birth', 'Freedom to make life choices',
       'Generosity', 'Perceptions of corruption', 'Positive affect',
       'Negative affect', 'Confidence in national government'],
      dtype='object')
```

[ ]:
```
life_ladder_df = happiness_df[['Life Ladder','Generosity']]
print(life_ladder_df['Life Ladder'].min())
print(life_ladder_df.shape)
life_ladder_df.head(2)
```

```
2.66171813
(140, 2)
```

[ ]:
```
   Life Ladder  Generosity
0     7.578745    0.210104
1     7.593702    0.145387
```

[ ]:
```python
# selecting multiple columns by names.
df_1 = happiness_df.loc[:, 'Life Ladder':'Generosity']
df_1.head()
```

[ ]:
```
   Life Ladder  Log GDP per capita  Social support  \
0     7.578745           11.081789        0.950128
1     7.593702           10.748989        0.952100
2     7.476214           10.760409        0.966753
3     7.473593           10.955548        0.949661
4     7.788252           10.612338        0.963826

   Healthy life expectancy at birth  Freedom to make life choices  Generosity
0                         71.086586                      0.953017    0.210104
1                         71.662498                      0.955416    0.145387
2                         72.755981                      0.938783    0.235479
3                         73.173759                      0.924997    0.167875
4                         71.696960                      0.962199   -0.012174
```

[ ]:
```python
# slicing
df_2 = happiness_df.iloc[10:100, 5:10]
df_2.head()
```

[ ]:
```
    Log GDP per capita  Social support  Healthy life expectancy at birth  \
10            9.670634        0.921697                         69.867302
11           10.716226        0.906218                         72.359711
12           10.899869        0.921003                         69.770920
13           11.066487        0.943482                         71.709785
14           10.711184        0.892166                         71.079102

    Freedom to make life choices  Generosity
10                      0.935618   -0.078269
11                      0.890031    0.124997
12                      0.868497    0.181657
13                      0.905341    0.206802
14                      0.840728    0.135308
```

[ ]:
```python
happiness_df['Region'].unique()
```

[ ]:
```
array(['Western Europe', 'North America and ANZ',
       'Middle East and North Africa', 'Latin America and Caribbean',
```

```
       'Central and Eastern Europe', 'Southeast Asia', 'East Asia',
       'Commonwealth of Independent States', 'Sub-Saharan Africa',
       'South Asia'], dtype=object)
```

[ ]:
```python
western_enrope_df = happiness_df[happiness_df['Region'] == "Western Europe"]
print(western_enrope_df.shape)
western_enrope_df.head(2)
```

```
(20, 14)
```

[ ]:
```
   Country          Region  Rank  HappinessScore  Life Ladder  \
0   Norway  Western Europe     1           7.537     7.578745
1  Denmark  Western Europe     2           7.522     7.593702

   Log GDP per capita  Social support  Healthy life expectancy at birth  \
0           11.081789        0.950128                         71.086586
1           10.748989        0.952100                         71.662498

   Freedom to make life choices  Generosity  Perceptions of corruption  \
0                      0.953017    0.210104                   0.249711
1                      0.955416    0.145387                   0.181148

   Positive affect  Negative affect  Confidence in national government
0         0.849100         0.202914                           0.717160
1         0.823667         0.205775                           0.572353
```

1.4.1 Q-1: Calculating the average, standard deviation, maximum, mininum, median of happiness scores.

Your solution should only show these statistics for happiness scores.

[ ]:

1.4.2 Q-2: What is the name and happiness score of the country with the lowest confidence in their national government?

[ ]:

1.4.3 Q-3 How many countries are in Western Europe?

This will be very easy wiht grouping function, but you can still do it without it

[ ]:

1.4.4 Q-4: Which two factors have the largest positive correlation and Which two factors have the largest negative correlation?

[ ]:
```
# this is how I would normally do this!
correlation_matrix = happiness_df.corr()
largest_positive_corr = (correlation_matrix[correlation_matrix < 1].stack().
 ↪idxmax())
factor1_pos, factor2_pos = largest_positive_corr
largest_negative_corr = (
    correlation_matrix[correlation_matrix > -1]
    .stack()
    .idxmin())
factor1_neg, factor2_neg = largest_negative_corr
largest_positive_corr_value = correlation_matrix.loc[factor1_pos, factor2_pos]
largest_negative_corr_value = correlation_matrix.loc[factor1_neg, factor2_neg]
print(f"The two factors with the largest positive correlation are␣
 ↪'{factor1_pos}' and '{factor2_pos}' with a correlation of␣
 ↪{largest_positive_corr_value:.2f}.")
print(f"The two factors with the largest negative correlation are␣
 ↪'{factor1_neg}' and '{factor2_neg}' with a correlation of␣
 ↪{largest_negative_corr_value:.2f}.")
```

```
The two factors with the largest positive correlation are 'HappinessScore' and
'Life Ladder' with a correlation of 0.93.
The two factors with the largest negative correlation are 'Rank' and
'HappinessScore' with a correlation of -0.99.
```

```
<ipython-input-58-35fdebe80184>:2: FutureWarning: The default value of
numeric_only in DataFrame.corr is deprecated. In a future version, it will
default to False. Select only valid columns or specify the value of numeric_only
to silence this warning.
  correlation_matrix = happiness_df.corr()
```

[ ]:
```
correlation_matrix = happiness_df.corr()
correlation_matrix.style.background_gradient(cmap='coolwarm')
```

```
<ipython-input-61-9287acdac567>:1: FutureWarning: The default value of
numeric_only in DataFrame.corr is deprecated. In a future version, it will
default to False. Select only valid columns or specify the value of numeric_only
to silence this warning.
  correlation_matrix = happiness_df.corr()
```

[ ]: <pandas.io.formats.style.Styler at 0x7fe04ec822c0>

[ ]:
```
correlation_matrix = happiness_df.corr()
#print(type(correlation_matrix))
correlation_matrix=correlation_matrix[correlation_matrix < 1].stack()
#print(type(correlation_matrix))
```

```
correlation_matrix_pos = correlation_matrix.idxmax()
#print(type(correlation_matrix_pos))
print(correlation_matrix_pos)
max_corr_value = correlation_matrix[correlation_matrix_pos]
print(max_corr_value)
```

```
('HappinessScore', 'Life Ladder')
0.9305290155706081
```

```
<ipython-input-65-56bb16c2e0e6>:1: FutureWarning: The default value of
numeric_only in DataFrame.corr is deprecated. In a future version, it will
default to False. Select only valid columns or specify the value of numeric_only
to silence this warning.
  correlation_matrix = happiness_df.corr()
```

1.5 Merging data

Let's load the world polulation data.

[ ]:
```
world_pop_df = pd.read_csv('/content/drive/MyDrive/shared/world_countries.csv').
↪dropna(axis=1, how='all')
world_pop_df.head()
```

[ ]:
```
          Country Code                 Region  Population     Area  \
0     Afghanistan  AFG  ASIA (EX. NEAR EAST)    31056997   647500
1         Albania  ALB         EASTERN EUROPE     3581655    28748
2         Algeria  DZA        NORTHERN AFRICA    32930091  2381740
3  American Samoa  ASM                OCEANIA       57794      199
4         Andorra  AND         WESTERN EUROPE       71201      468

   Pop. Density  Coastline  Net migration  Infant mortality      GDP  …  \
0          48.0       0.00          23.06            163.07    700.0  …
1         124.6       1.26          -4.93             21.52   4500.0  …
2          13.8       0.04          -0.39             31.00   6000.0  …
3         290.4      58.29         -20.71              9.27   8000.0  …
4         152.1       0.00           6.60              4.05  19000.0  …

   Phones  Arable  Crops  Other  Climate  Birthrate  Deathrate  Agriculture  \
0     3.2   12.13   0.22  87.65      1.0      46.60      20.34        0.380
1    71.2   21.09   4.42  74.49      3.0      15.11       5.22        0.232
2    78.1    3.22   0.25  96.53      1.0      17.14       4.61        0.101
3   259.5   10.00  15.00  75.00      2.0      22.46       3.27          NaN
4   497.2    2.22   0.00  97.78      3.0       8.71       6.25          NaN

   Industry  Service
0     0.240    0.380
1     0.188    0.579
2     0.600    0.298
```

```
3       NaN      NaN
4       NaN      NaN

[5 rows x 21 columns]
```

To extract populations from world_pop_df, we have to merge happiness_df with world_pop_df. As you probably can remember that some of the country names in world_counties.csv and happiness_2007.csvdo not match (Optional).

There are 4 kinds of merge: 'inner', 'outer', 'left', and 'right'. We practiced inner merge previously.

You may find examples from https://jakevdp.github.io/PythonDataScienceHandbook/03.07-merge-and-join.html: Example: US States Data

1.5.1 Q-5. Which country has the largest population in Latin America and Caribbean.

[ ]:

```

```

1.5.2 Q-6. Find the average population of East Asia.

[ ]:

```

```

[ ]:

```
!apt-get install texlive texlive-xetex texlive-latex-extra pandoc
!pip install pypandoc
!pip install nbconvert
```

[ ]:

```
!jupyter nbconvert  '/content/drive/MyDrive/datamining/module-3.ipynb' --to pdf
```