HW1-Q3_F23-Sol

August 31, 2023

1 Homework 1

Name: <insert name here> ***

This assignment is due on Canvas by **11:59 pm on June 15th**. Submit only this Jupyter notebook to Canvas. Do not compress it using tar, rar, zip, etc. Your solutions to analysis questions should be done in Markdown directly below the associated question. Remember that you are encouraged to discuss the problems with your instructors and classmates, but **you must write all code and solutions on your own**.

The rules to be followed for the assignment are:

- Do **NOT** load or use any Python packages that are not available in Anaconda for Python 3.9.
- Some problems with code may be autograded. If we provide a function or class API **do not** change it.
- Do not change the location of the data or data directory. Use only relative paths to access the data.

[1]:
```
import argparse
import pandas as pd
import numpy as np
import pickle
from pathlib import Path
```

1.0.1 Problem 1

The Function below should return the following attributes for the values within range [range_min, range_max] (both sides inclusive) in the ith column which are: - Number of objects - The minimum value - The maximum value - The mean value - The standard deviation value - The Q1 value - The median value - The Q3 value - The IQR value

Note: - The data file is already at "data/dataset.csv". Please maintain this as it would be necessary while grading. You don't need to download it yourself - Do not change the variable names of the returned values. - After calculating each of those values, assign them to the corresponding value that is being returned. - The col_ind can range from 0 - 8 - ONLY consider the values within [range_min, range_max] (both sides are INCLUSIVE)

For example, if a column has value 1,2,3,4,5,6 and [range_min, range_max] = [2,4], only the values 2,3,4 will be used to calculate the attributes.

[2]:
```
def calculate(dataFile, col_num, range_min, range_max):
    """
    Input Parameters:
        dataFile: The dataset file (in csv).
        col_num: The index of the column for which the various properties will␣
↪be calculated.
        [range_min, range_max]: the range of the values to be considered for␣
↪calculation
    Default value of 0,infinity,-infinity are assigned to all the variables as␣
↪required.
    """
    numObj, minValue, maxValue, mean, stdev, Q1, median, Q3, IQR =␣
↪[0,"inf","-inf",0,0,0,0,0,0]

    #YOUR TASK: Write code to assign the values to the respective variables.
    return numObj, minValue, maxValue, mean, stdev, Q1, median, Q3, IQR
```

Here is the ChatGPT generated solution:

[3]:
```
def calculate(dataFile, col_num, range_min, range_max):
    """
    Input Parameters:
        dataFile: The dataset file (in csv).
        col_num: The index of the column for which the various properties will␣
↪be calculated.
        [range_min, range_max]: the range of the values to be considered for␣
↪calculation
    Default value of 0,infinity,-infinity are assigned to all the variables as␣
↪required.
    """
    numObj, minValue, maxValue, mean, stdev, Q1, median, Q3, IQR = [0,␣
↪float('inf'), float('-inf'), 0, 0, 0, 0, 0, 0]

    # Load the dataset
    df = pd.read_csv(dataFile)

    # Filter values within the specified range
    values_within_range = df.iloc[:, col_num][(df.iloc[:, col_num] >=␣
↪range_min) & (df.iloc[:, col_num] <= range_max)]

    if not values_within_range.empty:
        numObj = len(values_within_range)
        minValue = np.min(values_within_range)
        maxValue = np.max(values_within_range)
```

```
        mean = np.mean(values_within_range)
        stdev = np.std(values_within_range)
        Q1 = np.percentile(values_within_range, 25)
        median = np.median(values_within_range)
        Q3 = np.percentile(values_within_range, 75)
        IQR = Q3 - Q1

    return numObj, minValue, maxValue, mean, stdev, Q1, median, Q3, IQR
```

1.0.2 Here are the unit tests. You don't need to modify them. Simply execute the cell and observe the output.

[4]:
```
import unittest

class TestAttr(unittest.TestCase):
    def setUp(self):
        self.loc = "data/dataset.csv"
        file = open('data/testing', 'rb')
        self.data = pickle.load(file)
        file.close()

    def test0(self):
        """
        Test calculation result
        """
        column, range_min, range_max = self.data[0]
        result = calculate(self.loc, column, range_min, range_max)
        self.assertEqual(result[0],self.data[1][0])
        self.assertAlmostEqual(result[1],self.data[1][1], places = 1)
        self.assertAlmostEqual(result[2],self.data[1][2], places = 1)
        self.assertAlmostEqual(result[3],self.data[1][3], places = 1)
        self.assertAlmostEqual(result[4],self.data[1][4], places = 1)
        self.assertAlmostEqual(result[5],self.data[1][5], places = 1)
        self.assertAlmostEqual(result[6],self.data[1][6], places = 1)
        self.assertAlmostEqual(result[7],self.data[1][7], places = 1)
        self.assertAlmostEqual(result[8],self.data[1][8], places = 1)

tests = TestAttr()
tests_to_run = unittest.TestLoader().loadTestsFromModule(tests)
unittest.TextTestRunner().run(tests_to_run)
```

```
.
----------------------------------------------------------------------
Ran 1 test in 0.087s

OK
```

[4]:
```
<unittest.runner.TextTestResult run=1 errors=0 failures=0>
```

1.0.3 [Part B] Scatter Plot. Use the cell below to add your code

[5]:
```
df = pd.read_csv('data/dataset.csv')
df
```

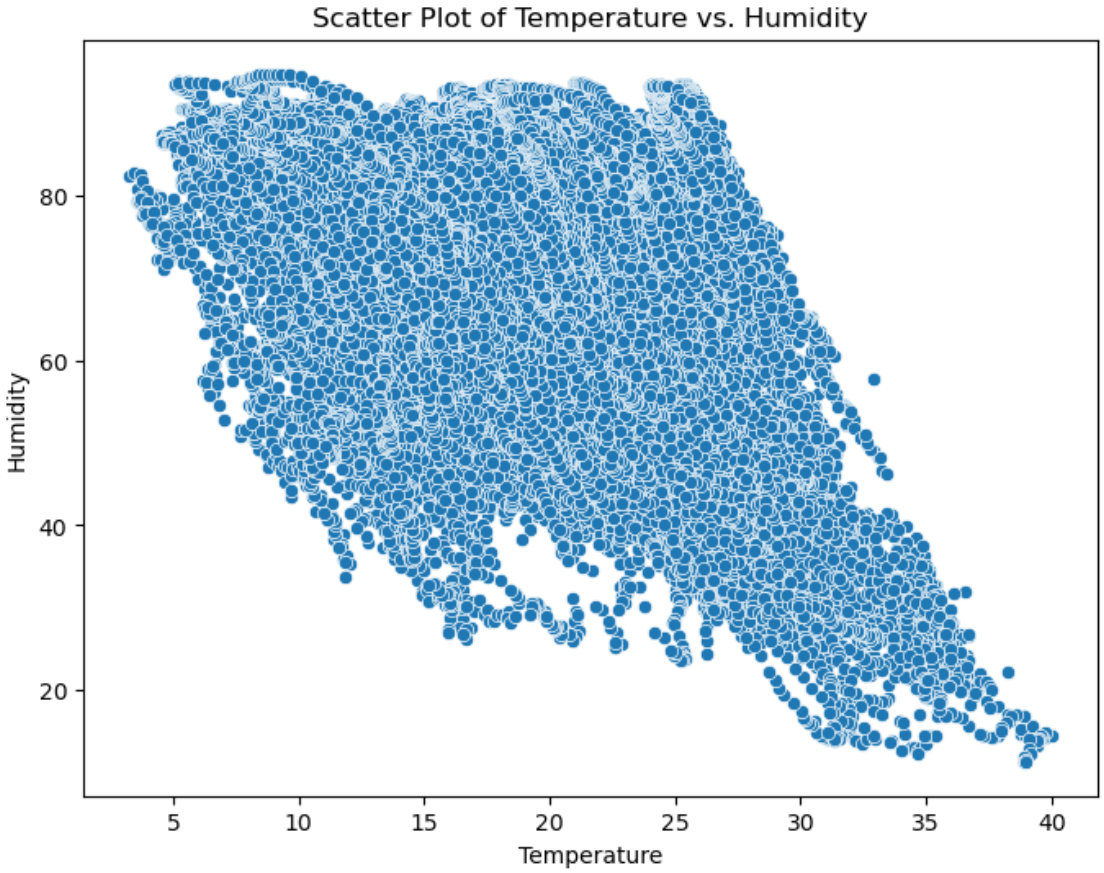
[5]:

	DateTime	Temperature	Humidity	Wind Speed	general diffuse flows	diffuse flows	Zone 1 Power Consumption	Zone 2 Power Consumption	Zone 3 Power Consumption
0	1/1/2017 0:00	6.559	73.8	0.083	0.051	0.119	34055.69620	16128.87538	20240.96386
1	1/1/2017 0:10	6.414	74.5	0.083	0.070	0.085	29814.68354	19375.07599	20131.08434
2	1/1/2017 0:20	6.313	74.5	0.080	0.062	0.100	29128.10127	19006.68693	19668.43373
3	1/1/2017 0:30	6.121	75.0	0.083	0.091	0.096	28228.86076	18361.09422	18899.27711
4	1/1/2017 0:40	5.921	75.7	0.081	0.048	0.085	27335.69620	17872.34043	18442.40964
…	…	…	…	…	…	…	…	…	…
52411	12/30/2017 23:10	7.010	72.4	0.080	0.040	0.096	31160.45627	26857.31820	14780.31212
52412	12/30/2017 23:20	6.947	72.6	0.082	0.051	0.093	30430.41825	26124.57809	14428.81152
52413	12/30/2017 23:30	6.900	72.8	0.086	0.084	0.074	29590.87452	25277.69254	13806.48259
52414	12/30/2017 23:40	6.758	73.0	0.080	0.066	0.089	28958.17490	24692.23688	13512.60504
52415	12/30/2017 23:50	6.580	74.1	0.081	0.062	0.111	28349.80989	24055.23167	13345.49820

[52416 rows x 9 columns]

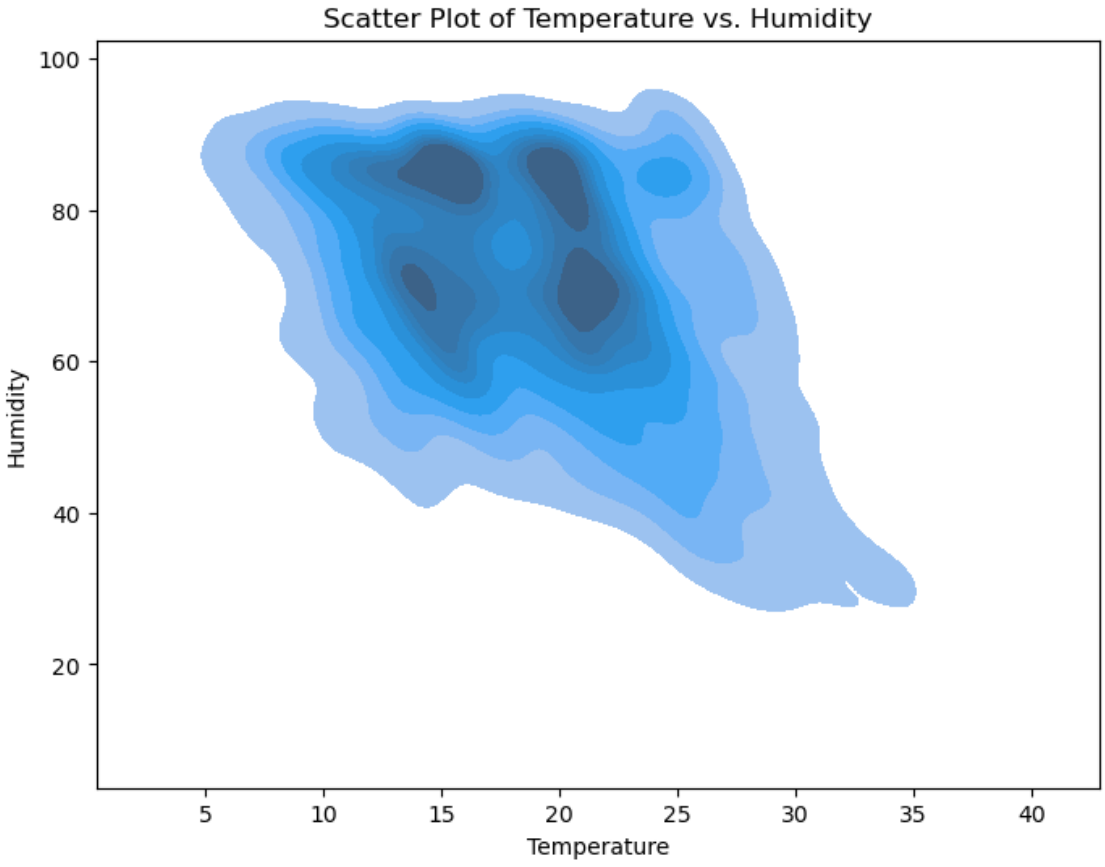
Here's a scatter plot

[6]:
```
import matplotlib.pyplot as plt
import seaborn as sns

# Set style
#sns.set(style="whitegrid")

# Create a scatter plot
plt.figure(figsize=(8, 6))
sns.scatterplot(x="Temperature", y="Humidity", data=df)
plt.title("Scatter Plot of Temperature vs. Humidity")
plt.xlabel("Temperature")
plt.ylabel("Humidity")
plt.show()

#plt.title('scatter plot')
```



Here's a KDE plot

[8]:
```
# Create a KDE plot of same 2 variables
plt.figure(figsize=(8, 6))
sns.kdeplot(x="Temperature", y="Humidity", data=df, fill=True)
plt.title("Scatter Plot of Temperature vs. Humidity")
plt.xlabel("Temperature")
plt.ylabel("Humidity")
plt.show()
```



Now look at the correlation between each column. Plot the heatmap of correlations (Triangular)

[10]:
```
# Compute correlations between variables
correlation_matrix = df.iloc[:, 1:].corr()

# Create a mask for the upper triangle
mask = np.triu(np.ones(correlation_matrix.shape), k=1)

# Set up the matplotlib figure
plt.figure(figsize=(10, 8))
```

```
# Generate a heatmap with the mask
sns.heatmap(correlation_matrix, annot=True, cmap="coolwarm", linewidths=.5,␣
 ↪fmt=".2f", square=True, center=0, mask=mask)

# Set title
plt.title("Correlation Heatmap of Variables")

# Show plot
plt.show()
```