Assignment II: Python Basics (Datafile Import, Plotting, Functions, Classes)

Exercises in Machine Learning (190.013), SS2022
Stefan Nehl[1]

[1]*stefan-christopher.nehl@stud.unileoben.ac.at, MNr: 00935188,* Montanuniversität Leoben, Austria

March 6, 2022

In the second assignment, I had to create basic statistic functions for calculating the mean, median, variance and standard deviation. Furthermore, I had to read in a CSV file, use the created functions and plot the values.

1 Introduction

The statistic functions I implemented were calculating the mean, median, variance, and standard deviation. Furthermore, I added a function for normalizing and standardizing a data set.

2 Implementation

For the implementation I created the class *BasicStatistics* with all the statistic functions. First, I implemented the mean.

2.1 Mean

The mean is the average of a collection of numbers. The formula for the calculation is the following (Institute, n.d.[a]):

$$mean = \frac{x_1 + x_2 + \ldots + x_n}{n}$$

Where n is the number of elements in the collection and x_n the element on the position n. The calculation was implemented in the *getMean* function.

```
1    def getMean(self):
2        length = len(self.dataSet)
3        sumValue = sum(self.dataSet)
4        mean = sumValue / length
```

2.2 Median

The Median is the middle value of a collection of numbers. The steps of the implemented algorithm are (Institute, n.d.[b]):

- Sort the collection of numbers
- Calculate the mid index
- if the length of the collection is uneven: take the element with the mid index
- if the length is even calculate the median:

$$median = \frac{x_{\text{mid}} + x_{\text{mid-1}}}{2}$$

- return the result

```
1
2    def getMedian(self):
3        length = len(self.dataSet)
4        mid = length // 2
5
6        if length % 2 == 0:
7            median1 = self.
                 sortedDataSet[mid]
8            median2 = self.
                 sortedDataSet[mid - 1]
9            median = (median1 +
                 median2) / 2
10       else:
11           median = self.
                 sortedDataSet[mid]
12
13       return median
```

2.3 Variance

The variance is the expected variation between values in a collection of numbers. The formula for the

calculation of the variance it the following (Ramos, 2021):

$$Variance\ \sigma^2 = \frac{\sum_{i=0}^{n-1}(x_i - \overline{x})}{n-1}$$

Where n is the number of elements, x_i the element on the index i and $\overline{x}$ the mean. This formula uses the Bessel's correction for smaller numbers. Therefore, instead of dividing the aggregated values with n, I divided them with n-1 (Ramos, 2021).

```
def getVariance(self):
    length = len(self.dataSet)
    mean = self.getMean()

    squareDeviations = [(x - mean)
        ** 2 for x in self.dataSet
        ]

    #Bessel's correction (n-1)
        instead of n for better
        results
    variance = sum(
        squareDeviations) / (length
        -1)
    return variance
```

2.4 Standard Deviation

The standard deviation is the amount of the variation of a collection of numbers. The formula for the calculation of the standard deviation is (Ramos, 2021):

$$Standard\ Deviation\ \sigma = \sqrt{\sigma^2}$$

```
def getStandardDeviation(self):
    variance = self.getVariance()
    standardDeviation = math.sqrt(
        variance)
    return standardDeviation
```

2.5 Normalize Data

Standardization is a preprocessing step, to standardize the range of values of a collection of numbers. I used the following formula to standardize the data (Jaadi, 2019).

$$z = \frac{x_i - \overline{x}}{\sigma}$$

Where z is the standardize value, x_i the value in the collection on index i, $\overline{x}$ the mean and σ the standard deviation.

```
def getNormalizeDataSet(self):
    mean = self.getMean()
    standardDeviation = self.
        getStandardDeviation()
    standardizeDataSet = [((x -
        mean)/standardDeviation)
        for x in self.dataSet]
    return standardizeDataSet
```

2.6 Testing the functions

I implemented a script file with the name *BasicStatistics_Test* which tests all the function with two data sets.

- 1, 2, 3, 4, 5
- 1, 2, 3, 4, 5, 6

2.7 Loading the data

The data which I should analyse is stored in the file *gauss.csv*. The data is one dimensional and I read the lines in the file with the function *reader()* of the imported csv package. The data was then stored in the array *dataSet*.

```
with open('gauss.csv', mode='r') as
    file:
    csvFile = csv.reader(file)

    for line in csvFile:
        dataSet.append(float(line[0]))
```

2.8 Plotting the results

For plotting the results, I created a plot with the *.figure()* function of *matplotlib* with the width of 8 inch and the height of 6 inch. I set the subtitle of the plot to *Data Distribution* and created a subplot. We need three different subplots in the plot. One histogram and two scatter plots. First, I created the histogram with the function *.subplot(2,1,1)*. This subplot consumes two columns and one row and starts at the first position. I added the needed labels and plotted the values with the *.hist()* function. I used 20 bins and the density property for the histogram. Also four vertical lines with the mean, median and the standard deviation with plus and minus was added to the subplot of the histogram. The other two subplots contain the raw and the standardized data, where each of the plots had three horizontal lines with the mean and the standard deviation.

3 Results

The charts in Figure 1 display the result of the basic statistic functions. The histogram at the top displays the data distribution of the normalized data with the mean, median and standard deviation, the scatter chart at the bottom left the raw data with the mean and standard deviation and the scatter chart at the bottom right the normalized data with mean and standard deviation.

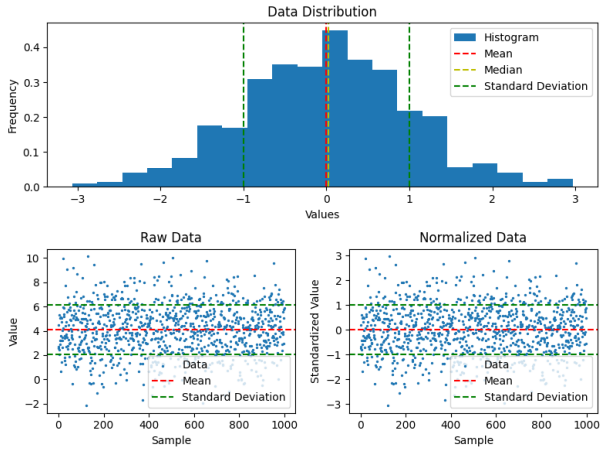

Figure 1: *Illustrates the raw data in a Histogram and the raw and standardized data in a scatter chart*

4 Conclusion

After some clarification of the tasks, implementing the statistic function was straightforward. The code needed for the implementation could keep clean and minimalistic. Only creating the plot functions needed some additional lines of code and the function *.tight_layout()* to improve the readability of the charts and labels.

APPENDIX

BasicStatistics class:

```
"""
@author: Nehl Stefan
"""

import math

class BasicStatistics:

    def __init__(self, dataSet):
        if self.checkDataSet(dataSet)
            == False:
            return None

        #clone dataSet
        self.dataSet = dataSet[:]
        self.sortedDataSet = dataSet
            [:]
        self.sortedDataSet.sort()

    def checkDataSet(self, dataSet):
        if dataSet == None:
            return False

        if len(dataSet) == 0:
            return False

        return True

    def getMean(self):
        length = len(self.dataSet)
        sumValue = sum(self.dataSet)
        mean = sumValue / length
        return mean

    def getMedian(self):
        length = len(self.dataSet)
        mid = length // 2

        if length % 2 == 0:
            median1 = self.
                sortedDataSet[mid]
            median2 = self.
                sortedDataSet[mid - 1]
            median = (median1 +
                median2) / 2
        else:
            median = self.
                sortedDataSet[mid]

        return median

    def getVariance(self):
        length = len(self.dataSet)
        mean = self.getMean()

        squareDeviations = [(x - mean)
            ** 2 for x in self.dataSet
            ]

        #Bessel's correction (n-1)
            instead of n for better
            results
        variance = sum(
            squareDeviations) / (length
            -1)
        return variance

    def getStandardDeviation(self):
        variance = self.getVariance()
        standardDeviation = math.sqrt(
            variance)
        return standardDeviation

    def getMinValue(self):
        return self.sortedDataSet[0]

    def getMaxValue(self):
        return self.sortedDataSet[len(
            self.sortedDataSet) - 1]

    def getNormalizeDataSetOld(self):
        min = self.getMinValue()
        max = self.getMaxValue()
```

```
71          dataNorm = [(i - min) / (max -
                min) for i in self.dataSet
                ]
72          return dataNorm
73
74      def getNormalizeDataSet(self):
75          mean = self.getMean()
76          standardDeviation = self.
                getStandardDeviation()
77          standardizeDataSet = [((x -
                mean)/standardDeviation)
                for x in self.dataSet]
78
79          return standardizeDataSet
```

BasicStatistics_Test class:

```
1   """
2   @author: Nehl Stefan
3   """
4
5   from BasicStatistics import
        BasicStatistics
6
7   dataSet1 = [1, 2, 3, 4, 5]
8   dataSet2 = [1, 2, 3, 4, 5, 6]
9
10  basicStatistics1 = BasicStatistics(
        dataSet1)
11  basicStatistics2 = BasicStatistics(
        dataSet2)
12  result = -1
13
14  #Mean
15  print('Mean')
16  expectedResult = 3
17  result = basicStatistics1.getMedian()
18  print('Expected: ' + str(
        expectedResult) + ' Result: ' + str
        (result))
19
20  expectedResult = 3.5
21  result = basicStatistics2.getMedian()
22  print('Expected: ' + str(
        expectedResult) + ' Result: ' + str
        (result))
23  print()
24
25  #Median
26  print('Median')
27  expectedResult = 3
28  result = basicStatistics1.getMedian()
29  print('Expected: ' + str(
        expectedResult) + ' Result: ' + str
        (result))
30
31  expectedResult = 3.5
32  result = basicStatistics2.getMedian()
33  print('Expected: ' + str(
        expectedResult) + ' Result: ' + str
        (result))
34  print()
35
36  #Variance
37  print('Variance')
38  expectedResult = 2.5
39  result = basicStatistics1.getVariance
        ()
40  print('Expected: ' + str(
        expectedResult) + ' Result: ' + str
        (result))
41
42  expectedResult = 3.5
43  result = basicStatistics2.getVariance
```

```
    ()
print('Expected: ' + str(
    expectedResult) + ' Result: ' + str
    (result))
print()

#StandardDefiation
print('Standard Defiation')
expectedResult = 1.5811388
result = basicStatistics1.
    getStandardDeviation()
print('Expected: ' + str(
    expectedResult) + ' Result: ' + str
    (result))

expectedResult = 1.8708287
result = basicStatistics2.
    getStandardDeviation()
print('Expected: ' + str(
    expectedResult) + ' Result: ' + str
    (result))
print()

#FindMinValue
print('Find MinValue')
expectedResult = 1
result = basicStatistics1.getMinValue
    ()
print('Expected: ' + str(
    expectedResult) + ' Result: ' + str
    (result))

print('Find MinValue')
expectedResult = 1
result = basicStatistics2.getMinValue
    ()
print('Expected: ' + str(
    expectedResult) + ' Result: ' + str
    (result))

#FindMaxValue
print('Find MaxValue')
expectedResult = 5
result = basicStatistics1.getMaxValue
    ()
print('Expected: ' + str(
    expectedResult) + ' Result: ' + str
    (result))

print('Find MaxValue')
expectedResult = 6
result = basicStatistics2.getMaxValue
    ()
print('Expected: ' + str(
    expectedResult) + ' Result: ' + str
    (result))

#Normalize Data
print('Standardized Data')
expectedResult = "??"
result = basicStatistics1.
    getNormalizeDataSetOld()
print('Expected: ' + str(
    expectedResult) + ' Result: ' + str
    (result))

expectedResult = "??"
result = basicStatistics2.
    getNormalizeDataSetOld()
print('Expected: ' + str(
    expectedResult) + ' Result: ' + str
    (result))
print()

#Standardized Data
print('Standardized Data')
expectedResult = "??"
result = basicStatistics1.
    getNormalizeDataSet()
print('Expected: ' + str(
    expectedResult) + ' Result: ' + str
    (result))

expectedResult = "??"
result = basicStatistics2.
    getNormalizeDataSet()
print('Expected: ' + str(
    expectedResult) + ' Result: ' + str
    (result))
print()
```

Plotting of the values:

```
"""
@author: Nehl Stefan
"""

import csv

import matplotlib.pyplot as plt
import numpy
from BasicStatistics import
    BasicStatistics

dataSet = []


with open('gauss.csv', mode='r') as
    file:
     csvFile = csv.reader(file)

     for line in csvFile:
         dataSet.append(float(line[0]))

basicStatistics = BasicStatistics(
    dataSet)

#dataSet = basicStatistics.
    normalizeDataSet(dataSet)
mean = basicStatistics.getMean()
median = basicStatistics.getMedian()
variance = basicStatistics.getVariance
    ()
standardDeviation = basicStatistics.
    getStandardDeviation()
standardizedDataSet = basicStatistics.
    getNormalizeDataSet()

standardizedStatistics =
    BasicStatistics(standardizedDataSet
    )
standardizedMean =
    standardizedStatistics.getMean()
standardizedMedian =
    standardizedStatistics.getMedian()
standardizedStandardDeviation =
    standardizedStatistics.
    getStandardDeviation()


length = len(standardizedDataSet)
plotRange = range(length)

plt.suptitle('Data Distribution')

plt.figure(figsize=(8,6))
# plot histogram (left, right, top)
plt.subplot(2, 1, 1)

plt.hist(standardizedDataSet, bins=20,
     density=True, label='Histogram')
plt.axvline(standardizedMean, label='
    Mean', color='r', ls='--')
plt.axvline(standardizedMedian, label=
    'Median', color='y', ls='--')
plt.axvline(standardizedMean -
    standardizedStandardDeviation,
    label='Standard Deviation', color='
    g', ls='--')
plt.axvline(standardizedMean +
    standardizedStandardDeviation,
    color='g', ls='--')

plt.title("Data Distribution")
plt.xlabel('Values')
plt.ylabel('Frequency')
plt.legend(loc="upper right")

# plot raw data (left, bottom)
plt.subplot(2, 2, 3)

plt.scatter(plotRange, dataSet, label=
    'Data', s=2)
plt.axhline(mean, label='Mean', color=
    'r', ls='--')
#plt.axhline(median, label='Median',
    color='y', ls='--')
plt.axhline(mean + standardDeviation,
    label='Standard Deviation', color='
    g', ls='--')
plt.axhline(mean - standardDeviation,
    color='g', ls='--')

plt.title('Raw Data')
plt.xlabel('Sample')
plt.ylabel('Value')
plt.legend(loc="best")

# plot standardizedData(left, bottom)
plt.subplot(2, 2, 4)
plt.title('Normalized Data')
plt.xlabel('Sample')
plt.ylabel('Standardized Value')

plt.scatter(plotRange,
    standardizedDataSet, label='Data',
    s=2)
plt.axhline(standardizedMean, label='
    Mean', color='r', ls='--')
#plt.axhline(standardizedMedian, label
    ='Median', color='y', ls='--')
plt.axhline(standardizedMean +
    standardizedStandardDeviation,
    label='Standard Deviation', color='
    g', ls='--')
plt.axhline(standardizedMean -
    standardizedStandardDeviation,
    color='g', ls='--')

```

```
81 plt.legend(loc="best")
82
83 plt.tight_layout()
84 plt.show()
```

Bibliography

Institute, Corporate Finance (n.d.[a]). *Mean*. URL: https://corporatefinanceinstitute.com/resources/knowledge/other/mean/ (visited on 04/03/2022).

– (n.d.[b]). *Median*. URL: https://corporatefinanceinstitute.com/resources/knowledge/other/median/ (visited on 04/03/2022).

Jaadi, Zakaria (2019). *Standardization*. URL: https://builtin.com/data-science/when-and-why-standardize-your-data (visited on 04/03/2022).

Ramos, Leodanis Pozo (2021). *Calculating Variance and Standard Deviation in Python*. URL: https://stackabuse.com/calculating-variance-and-standard-deviation-in-python/ (visited on 04/03/2022).

Zach (2021). *How to Normalize Data in Python*. URL: https://www.statology.org/normalize-data-in-python/ (visited on 04/03/2022).