De_Guzman_Hands_on_Activity_6_1_Introduction_to_Data_Analysis_a

March 7, 2024

1 Hands-on Activity 6.1 Introduction to Data Analysis and Tools

CPE311 Computational Thinking with Python **Name:** De Guzman, Jemuel Endrew C.

Section: CPE22S3

Performed by: 06/03/2024

Submitted on:

Submitted to: Engr. Roman M. Richard

1.1 6.1 Intended Learning Outcomes

1. Use pandas and numpy data analysis tools.
2. Demonstrate how to analyze data using numpy and pandas.

1.2 6.2 Resources:

- Personal Computer
- Jupyter Notebook
- Internet Connection

1.3 6.3 Supplementary Activities:

1.3.1 Exercise 1

Run the given code below for exercises 1 and 2, perform the given tasks without using any Python modules.

[166]:
```
import random
random.seed(0)
salaries = [round(random.random()*1000000, -3) for _ in range(100)]
```

Using the data generated above, calculate the following statistics without importing anything from the statistics module in the standard library (https://docs.python.org/3/library/statistics.html) and then confirm your results match up to those that are obtained when using the statistics module (where possible):

- Mean

[167]:
```
# Returns the mean of the dataset
def mean(dataset):
  length = len(dataset)
  sum = 0
  for i in dataset:
    sum += i
  return sum/length

print(mean(salaries))
```

```
585690.0
```

- Median

[168]:
```
# Returns the middle value of the dataset after sorting
def median(dataset):
  dataset.sort()
  length = len(dataset)
  if length % 2 == 1:
    return dataset[int(length/2)]
  else:
    a = dataset[int((length/2)-1)]
    b = dataset[int((length/2))]
    ave = (a+b)/2
    return ave

print(median(salaries))
```

```
589000.0
```

- Mode (hint: check out the Counter in the collections module of the standard library at https://docs.python.org/3/library/collections.html#collections.Counter)

[169]:
```
# Returns the value with the highest count in the dataset
def mode(dataset):
  counts = {}
  for i in dataset:
    if i not in counts:
      counts[i] = 0
    counts[i] += 1

  maximum = 0
  for i in counts:
    maximum = max(counts.values())
  return [x for x, y in counts.items() if y == maximum]

print(mode(salaries))
```

```
[477000.0]
```

- Sample Variance

[170]:
```
# Returns the dispersion of the data with respect to the mean value.
# It gives an actual value to how much the numbers in a data set vary from the␣
 ↪mean.
def variance(dataset):
  sum = 0
  ave = mean(dataset)
  n = len(dataset)
  for i in dataset:
    x = (i - ave) ** 2
    sum += x
  return sum / (n - 1)

print(variance(salaries))
```

```
70664054444.44444
```

- Sample Standard Deviation

[171]:
```
from math import sqrt
# Returns the dispersion of the data with respect to the mean value.
# It measures how far apart numbers are in a data set.
def standard_deviation(dataset):
  print(sqrt(variance(dataset)))
  return (variance(dataset)) ** 0.5

print(standard_deviation(salaries))
```

```
265827.11382484
265827.11382484
```

1.3.2 Exercise 2

Using the same data, calculate the following statistics using the functions in the statistics module where appropriate:

- Range

[172]:
```
# Returns the range of the dataset
def data_range(dataset):
  return max(dataset) - min(dataset)

print(data_range(salaries))
```

```
995000.0
```

- Coefficient of Variation Interquartile Range

[173]:
```
import statistics as stat
# Returns the Interquartile Range of the dataset
def iqr(dataset):
  ave = mean(dataset)
  sd = standard_deviation(dataset)
  nd = stat.NormalDist(ave, sd)
  q = list(map(round, nd.quantiles()))
  return q[2] - q[0]

print(iqr(salaries))
```

```
265827.11382484
358596
```

- Quartile Coefficients of Dispersion

[174]:
```
# Returns the Quartile Coefficients of Dispersion
def quartiles(dataset):
  ave = mean(dataset)
  sd = standard_deviation(dataset)
  nd = stat.NormalDist(ave, sd)
  q = list(map(round, nd.quantiles()))
  return q

q = quartiles(salaries)
for i in range(len(q)):
  print(f"Q{i+1}: {q[i]}")
```

```
265827.11382484
Q1: 406392
Q2: 585690
Q3: 764988
```

1.3.3 Exercise 3: Pandas for Data Analysis

Load the diabetes.csv file. Convert the diabetes.csv into dataframe

[175]:
```
import pandas as pd
filepath = '/content/diabetes.csv'
df = pd.read_csv(filepath)
df
```

[175]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	\
0	6	148	72	35	0	33.6	
1	1	85	66	29	0	26.6	
2	8	183	64	0	0	23.3	
3	1	89	66	23	94	28.1	
4	0	137	40	35	168	43.1	
..	…	…	…	…	…	…	…

```
763           10      101             76             48      180  32.9
764            2      122             70             27        0  36.8
765            5      121             72             23      112  26.2
766            1      126             60              0        0  30.1
767            1       93             70             31        0  30.4

     DiabetesPedigreeFunction  Age  Outcome
0                       0.627   50        1
1                       0.351   31        0
2                       0.672   32        1
3                       0.167   21        0
4                       2.288   33        1
..                        …  …      …
763                     0.171   63        0
764                     0.340   27        0
765                     0.245   30        0
766                     0.349   47        1
767                     0.315   23        0

[768 rows x 9 columns]
```

Perform the following tasks in the diabetes dataframe:

1. Identify the column names

[176]:
```
df.columns
```

[176]:
```
Index(['Pregnancies', 'Glucose', 'BloodPressure', 'SkinThickness', 'Insulin',
       'BMI', 'DiabetesPedigreeFunction', 'Age', 'Outcome'],
      dtype='object')
```

2. Identify the data types of the data

[177]:
```
df.dtypes
```

[177]:
```
Pregnancies                 int64
Glucose                     int64
BloodPressure               int64
SkinThickness               int64
Insulin                     int64
BMI                       float64
DiabetesPedigreeFunction  float64
Age                         int64
Outcome                     int64
dtype: object
```

3. Display the total number of records

[178]:
```
df.count()
```

[178]:
```
Pregnancies                 768
Glucose                     768
BloodPressure               768
SkinThickness               768
Insulin                     768
BMI                         768
DiabetesPedigreeFunction    768
Age                         768
Outcome                     768
dtype: int64
```

4. Display the first 20 records

[179]:
```
df.head(20)
```

[179]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
0	6	148	72	35	0	33.6	0.627	50	1
1	1	85	66	29	0	26.6	0.351	31	0
2	8	183	64	0	0	23.3	0.672	32	1
3	1	89	66	23	94	28.1	0.167	21	0
4	0	137	40	35	168	43.1	2.288	33	1
5	5	116	74	0	0	25.6	0.201	30	0
6	3	78	50	32	88	31.0	0.248	26	1
7	10	115	0	0	0	35.3			
8	2	197	70	45	543	30.5			
9	8	125	96	0	0	0.0			
10	4	110	92	0	0	37.6			
11	10	168	74	0	0	38.0			
12	10	139	80	0	0	27.1			
13	1	189	60	23	846	30.1			
14	5	166	72	19	175	25.8			
15	7	100	0	0	0	30.0			
16	0	118	84	47	230	45.8			
17	7	107	74	0	0	29.6			
18	1	103	30	38	83	43.3			
19	1	115	70	30	96	34.6			

7	0.134	29	0
8	0.158	53	1
9	0.232	54	1
10	0.191	30	0
11	0.537	34	1
12	1.441	57	0
13	0.398	59	1
14	0.587	51	1
15	0.484	32	1
16	0.551	31	1
17	0.254	31	1
18	0.183	33	0
19	0.529	32	1

5. Display the last 20 records

[180]:
```
df.tail(20)
```

[180]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	\
748	3	187	70	22	200	36.4	
749	6	162	62	0	0	24.3	
750	4	136	70	0	0	31.2	
751	1	121	78	39	74	39.0	
752	3	108	62	24	0	26.0	
753	0	181	88	44	510	43.3	
754	8	154	78	32	0	32.4	
755	1	128	88	39	110	36.5	
756	7	137	90	41	0	32.0	
757	0	123	72	0	0	36.3	
758	1	106	76	0	0	37.5	
759	6	190	92	0	0	35.5	
760	2	88	58	26	16	28.4	
761	9	170	74	31	0	44.0	
762	9	89	62	0	0	22.5	
763	10	101	76	48	180	32.9	
764	2	122	70	27	0	36.8	
765	5	121	72	23	112	26.2	
766	1	126	60	0	0	30.1	
767	1	93	70	31	0	30.4	

	DiabetesPedigreeFunction	Age	Outcome
748	0.408	36	1
749	0.178	50	1
750	1.182	22	1
751	0.261	28	0
752	0.223	25	0
753	0.222	26	1

```
754                     0.443   45        1
755                     1.057   37        1
756                     0.391   39        0
757                     0.258   52        1
758                     0.197   26        0
759                     0.278   66        1
760                     0.766   22        0
761                     0.403   43        1
762                     0.142   33        0
763                     0.171   63        0
764                     0.340   27        0
765                     0.245   30        0
766                     0.349   47        1
767                     0.315   23        0
```

6. Change the Outcome column to Diagnosis

[181]:
```
df.rename(columns = {'Outcome':'Diagnosis'}, inplace=True)
df
```

[181]:
```
     Pregnancies  Glucose  BloodPressure  SkinThickness  Insulin   BMI  \
0              6      148             72             35        0  33.6
1              1       85             66             29        0  26.6
2              8      183             64              0        0  23.3
3              1       89             66             23       94  28.1
4              0      137             40             35      168  43.1
..           ...      ...            ...            ...      ...  ...
763           10      101             76             48      180  32.9
764            2      122             70             27        0  36.8
765            5      121             72             23      112  26.2
766            1      126             60              0        0  30.1
767            1       93             70             31        0  30.4

     DiabetesPedigreeFunction  Age  Diagnosis
0                       0.627   50          1
1                       0.351   31          0
2                       0.672   32          1
3                       0.167   21          0
4                       2.288   33          1
..                        ...  ...        ...
763                     0.171   63          0
764                     0.340   27          0
765                     0.245   30          0
766                     0.349   47          1
767                     0.315   23          0

[768 rows x 9 columns]
```

7. Create a new column Classification that display “Diabetes” if the value of outcome is 1 , otherwise “No Diabetes”

[182]:
```
df["Classification"] = ["Diabetes" if i == 1 else "No Diabetes" for i in df.
↪Diagnosis]
df
```

[182]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	\
0	6	148	72	35	0	33.6	
1	1	85	66	29	0	26.6	
2	8	183	64	0	0	23.3	
3	1	89	66	23	94	28.1	
4	0	137	40	35	168	43.1	
..	…	…	…	…	…	…	
763	10	101	76	48	180	32.9	
764	2	122	70	27	0	36.8	
765	5	121	72	23	112	26.2	
766	1	126	60	0	0	30.1	
767	1	93	70	31	0	30.4	

	DiabetesPedigreeFunction	Age	Diagnosis	Classification
0	0.627	50	1	Diabetes
1	0.351	31	0	No Diabetes
2	0.672	32	1	Diabetes
3	0.167	21	0	No Diabetes
4	2.288	33	1	Diabetes
..	…	…	…	…
763	0.171	63	0	No Diabetes
764	0.340	27	0	No Diabetes
765	0.245	30	0	No Diabetes
766	0.349	47	1	Diabetes
767	0.315	23	0	No Diabetes

[768 rows x 10 columns]

8. Create a new dataframe “withDiabetes” that gathers data with diabetes

[183]:
```
withDiabetes = df[(df['Classification'] == "Diabetes")]
withDiabetes
```

[183]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	\
0	6	148	72	35	0	33.6	
2	8	183	64	0	0	23.3	
4	0	137	40	35	168	43.1	
6	3	78	50	32	88	31.0	
8	2	197	70	45	543	30.5	
..	…	…	…	…	…	…	
755	1	128	88	39	110	36.5	

```
757            0      123             72              0        0  36.3
759            6      190             92              0        0  35.5
761            9      170             74             31        0  44.0
766            1      126             60              0        0  30.1

     DiabetesPedigreeFunction  Age  Diagnosis Classification
0                       0.627   50          1       Diabetes
2                       0.672   32          1       Diabetes
4                       2.288   33          1       Diabetes
6                       0.248   26          1       Diabetes
8                       0.158   53          1       Diabetes
..                        …  …        …             …
755                     1.057   37          1       Diabetes
757                     0.258   52          1       Diabetes
759                     0.278   66          1       Diabetes
761                     0.403   43          1       Diabetes
766                     0.349   47          1       Diabetes

[268 rows x 10 columns]
```

9. Create a new dataframe "noDiabetes" thats gathers data with no diabetes

[184]:
```
noDiabetes = df[(df['Classification'] == "No Diabetes")]
noDiabetes
```

[184]:
```
     Pregnancies  Glucose  BloodPressure  SkinThickness  Insulin   BMI  \
1              1       85             66             29        0  26.6
3              1       89             66             23       94  28.1
5              5      116             74              0        0  25.6
7             10      115              0              0        0  35.3
10             4      110             92              0        0  37.6
..           …      …             …              …      …   …
762            9       89             62              0        0  22.5
763           10      101             76             48      180  32.9
764            2      122             70             27        0  36.8
765            5      121             72             23      112  26.2
767            1       93             70             31        0  30.4

     DiabetesPedigreeFunction  Age  Diagnosis Classification
1                       0.351   31          0    No Diabetes
3                       0.167   21          0    No Diabetes
5                       0.201   30          0    No Diabetes
7                       0.134   29          0    No Diabetes
10                      0.191   30          0    No Diabetes
..                        …  …        …             …
762                     0.142   33          0    No Diabetes
763                     0.171   63          0    No Diabetes
```

```
764                     0.340   27          0    No Diabetes
765                     0.245   30          0    No Diabetes
767                     0.315   23          0    No Diabetes

[500 rows x 10 columns]
```

10. Create a new dataframe “Pedia” that gathers data with age 0 to 19

[185]:
```
Pedia = df[(df['Age'] >= 0) & (df['Age'] <= 19)]
Pedia
```

[185]:
```
Empty DataFrame
Columns: [Pregnancies, Glucose, BloodPressure, SkinThickness, Insulin, BMI,
DiabetesPedigreeFunction, Age, Diagnosis, Classification]
Index: []
```

11. Create a new dataframe “Adult” that gathers data with age greater than 19

[186]:
```
Adult = df[(df['Age'] > 19)]
Adult
```

[186]:
```
     Pregnancies  Glucose  BloodPressure  SkinThickness  Insulin   BMI  \
0              6      148             72             35        0  33.6
1              1       85             66             29        0  26.6
2              8      183             64              0        0  23.3
3              1       89             66             23       94  28.1
4              0      137             40             35      168  43.1
..           …       …              …              …       …   …
763           10      101             76             48      180  32.9
764            2      122             70             27        0  36.8
765            5      121             72             23      112  26.2
766            1      126             60              0        0  30.1
767            1       93             70             31        0  30.4

     DiabetesPedigreeFunction  Age  Diagnosis Classification
0                       0.627   50          1       Diabetes
1                       0.351   31          0    No Diabetes
2                       0.672   32          1       Diabetes
3                       0.167   21          0    No Diabetes
4                       2.288   33          1       Diabetes
..                        …  …        …             …
763                     0.171   63          0    No Diabetes
764                     0.340   27          0    No Diabetes
765                     0.245   30          0    No Diabetes
766                     0.349   47          1       Diabetes
767                     0.315   23          0    No Diabetes

[768 rows x 10 columns]
```

12. Use numpy to get the average age and glucose value.

[187]:
```
import numpy as np

age_average = np.mean(df['Age'])
print(f'Age Average: {age_average}')

glucose_average = np.mean(df['Glucose'])
print(f'Glucose Average: {glucose_average}')
```

```
Age Average: 33.240885416666664
Glucose Average: 120.89453125
```

13. Use numpy to get the median age and glucose value.

[188]:
```
age_median = np.median(df['Age'])
print(f'Age Median: {age_median}')

glucose_median = df.loc[age_median].Glucose
print(f'Glucose Median: {glucose_median}')
```

```
Age Median: 29.0
Glucose Median: 117
```

14. Use numpy to get the middle values of glucose and age.

[189]:
```
age_middle = np.quantile(df['Age'], 0.5)
print(f'Age Middle Value: {age_middle}')

glucose_middle = np.quantile(df['Glucose'], 0.5)
print(f'Glucose Middle Value: {glucose_middle}')
```

```
Age Middle Value: 29.0
Glucose Middle Value: 117.0
```

15. Use numpy to get the standard deviation of the skinthickness.

[190]:
```
std_skinthickness = np.std(df['SkinThickness'])
print(f'Standard Deviation of Skin Thickness: {std_skinthickness}')
```

```
Standard Deviation of Skin Thickness: 15.941828626496939
```

1.4 6.4 Conclusion:

From this activity, I was able to explore the different quantities that we need when dealing with data and statistics. By writing code that computes for these values, I was able to understand and recall how they are obtained and how they can be used to analyze and interpret a given set of data. It also allowed me to imagine how the imported functions work, specifically the ones from the **statistics** module as well as **pandas** and **numpy**. The use of imported library functions allow us to manipulate and interact with a large data set with ease. All these functions will come in

handy when I will begin to analyze and interpret specific sets of data. This is why it is important that I am familiarized with these values and functions.