Hands-on Activity 6.1 Introduction to Data Analysis and Tools

CPE311 Computational Thinking with Python

Name: Marquez, Keith Leigh Zhen R.

Section: CPE22S3

Performed on: 03/07/2024

Submitted on: 03/07/2024

Submitted to: Engr. Roman M. Richard

6.1 Intended Learning Outcome

Use pandas and numpy data analysis tools.

Demonstrate how to analyze data using numpy and pandas

6.2 Resources:

Personal Computer

Jupyter Notebook

Intern Python modules.

6.3 Supplementary Activities: Exercise 1 Run the given code below for exercises 1 and 2, perform the given tasks without using any Python modules.

```
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)- : Me- an Med- ian Mode (hint: check out the Counter in the collections module of the standard libra y at https://docs.python.org/3/library/collections.html#collections.Cou- nter) Sample va- riance Sample standard deviation

```
# Mean

import statistics

mean = sum(salaries) / len(salaries)

print("Mean:", mean)
print("Mean (Statistics Mode):", statistics.mean(salaries))
```

```
Mean: 585690.0
Mean (Statistics Mode): 585690.0
```

```
# Median

import statistics

n_salaries = sorted(salaries)
n = len(n_salaries)

if n % 2 == 0:
    median1 = n_salaries[n//2]
    median2 = n_salaries[n//2 - 1]
    median = (median1 + median2)/2
else:
    median = n_salaries[n//2]

print("Median:", median)
print("Median (Statistics Mode):", statistics.median(salaries))
```

```
Median: 589000.0
Median (Statistics Mode): 589000.0
```

```
# Mode
import statistics

print("Mode (Statistics Mode):", statistics.mode(salaries))
```

```
Mode (Statistics Mode): 477000.0
```

```
# Sample variance
import statistics

deviations = [(x - mean) ** 2 for x in salaries]
variance = sum(deviations) / (len(salaries) - 1)

print("Sample Variance:", variance)
print("Variance (Statistics Mode):", statistics.variance(salaries))
```

```
Sample Variance: 70664054444.44444
Variance (Statistics Mode): 70664054444.44444
```

```
# Sample standard deviation

import statistics
std_dev = variance ** 0.5
print("Sample Standard Deviation:", std_dev)
print("Standard deviation (Statistics mode):", statistics.stdev(salaries))
```

```
Sample Standard Deviation: 265827.11382484
Standard deviation (Statistics mode): 265827.11382484
```

Exercise 2 Using the same data, calculate the following statistics using the functions in the statistics module where appropriate:

- Range
- Coefficient of variation Interquartile range
- Quartile coefficient of dispersion

```
# Range
range = max(salaries) - min(salaries)
print("Range:", range)
```

```
Range: 995000.0
```

```
# Coefficient of variation Interquartile range
import statistics

# Calculate coefficient of variation
cv = statistics.stdev(salaries) / statistics.mean(salaries)

# Calculate interquartile range
q3, q1 = statistics.median_high(salaries), statistics.median_low(salaries)
iqr = q3 - q1

print("Coefficient of Variation (CV):", cv)
print("Interquartile Range (IQR):", iqr)
```

```
Coefficient of Variation (CV): 0.45386998894439035
Interquartile Range (IQR): 2000.0
```

```
# Quartile coefficient of dispersion
qcd = (q3 - q1) / (q3 + q1)
print("Quartile Coefficient of Dispersion (QCD):", qcd)
```

```
Quartile Coefficient of Dispersion (QCD): 0.001697792869269949
```

Exercise 3: Pandas for Data Analysis Load the diabetes.csv file. Convert the diabetes.csv into dataframe Perform the following tasks in the diabetes datafram

. Identify the column names . Identify the data types of the data . Display the total number of records . Display the first 20 records . Display the last 20 records . Change the Outcome column to Diag . Create a new column Classification that display "Diabetes" if the value of outcome is 1 , otherwise "No Diabetes" . Create a new dataframe "withDiabetes" that gathers data with diabetes . Create a new dataframe "noDiabetes" thats gathers data with no diabet . Create a new dataframe "Pedia" that gathers data with age 0 to 19 . Create a new dataframe "Adult" that gathers data with age greater than 19 . Use numpy to get the average age and glucose value. . Use numpy to get the median age and glucose value. . Use numpy to get the middle values of glucose and age. . Use numpy to get the standard deviation of the skinthickness.esnosis the skinthickness.e:

```
filepath = '/content/diabetes.csv'

import pandas as pd
import numpy as np

diabetes = pd.read_csv(filepath)
diabetes
```

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigree
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	

768 rows × 9 columns

Next steps: View recommended plots

```
# Identify the column names
diabetes.columns
```

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

```
# Identify the data types of the data
diabetes.dtypes
```

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

```
#Display the total number of records

print('The total number of records:', len(diabetes))
```

```
The total number of records: 768
```

```
#Display the first 20 records
diabetes.head(20)
```

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeF
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	
5	5	116	74	0	0	25.6	
6	3	78	50	32	88	31.0	
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	

Next steps: View recommended plots

```
# Display the last 20 records

diabetes.tail(20)
```

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigree
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	

```
# Change the Outcome column to Diagnosis
diabetes.rename(columns = {'Outcome':'Diagnosis'}, inplace = True)
diabetes
```

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigree
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	

768 rows × 9 columns

Next steps: View recommended plots

```
# Create a new column Classification that display "Diabetes" if the value of outcome is 1 , otherwise "No Diabetes"

diabetes['Classification'] = np.where(diabetes['Diagnosis'] == 1, 'Diabetes', 'No Diabetes')
diabetes
```

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigree
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	

768 rows × 10 columns

Next steps: View recommended plots

```
# Create a new dataframe "withDiabetes" that gathers data with diabetes

new_diabetes = pd.DataFrame(diabetes)
new_diabetes = diabetes[diabetes['Diagnosis'] == 1].copy()
new_diabetes
```

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigree
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	

268 rows × 10 columns

Next steps: View recommended plots

```
# Create a new dataframe "noDiabetes" thats gathers data with no diabetes

new_diabetes = pd.DataFrame(diabetes)
new_diabetes = diabetes[diabetes['Diagnosis'] == 0].copy()
new_diabetes
```

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigree
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	

500 rows × 10 columns

Next steps: View recommended plots

```
#Create a new dataframe "Pedia" that gathers data with age 0 to 19
new_diabetes = pd.DataFrame(diabetes)
new_diabetes = diabetes[diabetes['Age'] <= 19].copy()
new_diabetes
```

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFun

```
#Create a new dataframe "Adult" that gathers data with age greater than 19
new_diabetes = pd.DataFrame(diabetes)
new_diabetes = diabetes[diabetes['Age'] >= 19].copy()
new_diabetes
```

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigree
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	

768 rows × 10 columns

Next steps: View recommended plots

```
#Use numpy to get the average age and glucose value.
mean_glucose = np.mean(diabetes['Glucose'])
mean_age = np.mean(diabetes['Age'])
print('Average Glucose:', mean_glucose)
print('Average Age:', mean_age)
```

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

```
#Use numpy to get the median age and glucose value.
median_glucose = np.median(diabetes['Glucose'])
median_age = np.median(diabetes['Age'])
print('Median Glucose:', median_glucose)
print('Median Age:', median_age)
```

```
Median Glucose: 117.0
Median Age: 29.0
```