Hands-on Activity 6.1 Introduction to Data Analysis and Tools

CPE311 Computational Thinking with Python

Name: Dolores, Marc Joseph S.

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

Internet Connection

6.3 Supplementary Activities:

EXERCISE 1:

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

```
[844000.0, 758000.0, 421000.0, 259000.0, 511000.0, 405000.0, 784000.0, 303000.0, 477000.0, 583000.0, 908000.0, 505000.0, 282
```

```
#MEAN
def getmean(salaries):
  #GET THE SUM OF ALL NUMBERS IN ARRAY AND DIVIDE IT BY ITS SIZE
  mean = sum(salaries)/len(salaries)
  return mean

print(getmean(salaries))
```

```
585690.0
```

```
#MEDIAN
def getmedian(salaries):
  sortarr = sorted(salaries)
  midex = len(sortarr)//2
  if (len(sortarr) % 2 == 0):
    #IF THE SIZE OF ARRAY IS EVEN DIVED THE TWO MIDDLE NUMBERS
    median = (sortarr[midex-1]+sortarr[midex])/2
  else:
    #IF THE SIZE OF ARRAY IS ODD THEN THE MIDDLE IS THE MEDIAN
    median = sortarr[midex]
  return median

print(getmedian(salaries))

```

```
589000.0
```

```
#MODE
def getmode(salaries):
  #INITIALIZE A DICTIONARY TO STORE THE FREQUENCY
  frequendict = {}
  for i in salaries:
    #ITERATES TO GET THE FREQUENCY OF EACH NUMBER
    frequendict[i] = frequendict.get(i,0)+1
  maxc = max(frequendict.values())
  #FIND THE NUMBER WITH HIGHEST FREQUENCY
  mode = [i for i, count in frequendict.items() if count == maxc]
  freq = frequendict[mode[0]]
  return mode, freq

print(getmode(salaries))
```

```
([477000.0], 3)
```

```
#SAMPLE VARIANCE
def getsamvar(salaries):
  index = len(salaries)
  #CALLING THE GETMEAN FUNCTION
  mean = getmean(salaries)
  #CALCULATE BY GETTING THE SUM OF SQUARED DIFFERENCE OF MEAN AND EACH SALARY
  variance = sum((i-mean)**2 for i in salaries)/(index-1)
  return variance

print(getsamvar(salaries))
```

```
70664054444.44444
```

```
#STANDARD DEVIATION
def getstandev(salaries):
  #RAISE TO 1/2 TO GET THE STANDARD DEVIATION
  standev = getsamvar(salaries)**(1/2)
  return standev

print(getstandev(salaries))
```

```
265827.11382484
```

EXERCISE 2:

```
#RANGE
def getrange(salaries):
  range = max(salaries) - min(salaries)
  return range

print(getrange(salaries))
```

```
995000.0
```

```
#COEFFICIENT OF VARIATION & INTERQUARTAL RANGE
def getcviqr(salaries):
  mean = getmean(salaries)
  stdv = getstandev(salaries)
  cv = (stdv/mean)*100
  import statistics as st
  q1 = st.quantiles(salaries, n=4)[0]
  q3 = st.quantiles(salaries, n=4)[2]
  iqr = q3 - q1
  return cv, iqr

cv, iqr = getcviqr(salaries)
print("Coefficient of Variation: " + str(cv))
print("InterQuartal Range: " + str(iqr))
```

```
Coefficient of Variation: 45.38699889443903
InterQuartal Range: 421750.0
```

```
#QUARTILE COEFFICIENT OF DISPERSION
def getqcd(salaries):
  import statistics as st
  q1 = st.quantiles(salaries, n=4)[0]
  q3 = st.quantiles(salaries, n=4)[2]
  iqr = q3 - q1
  median = getmedian(salaries)
  qcd = iqr/median
  return qcd

```

```
qcd = getqcd(salaries)
print("QCD: "+str(qcd))
```

```
QCD: 0.716044142614601
```

EXERCISE 3:

```
import pandas as pd
import numpy as np
filepath = '/content/diabetes.csv'
df = pd.read_csv(filepath)
```

```
df
```

	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
...	...	...	...	...	...	...	...	...	...
763	10	101	76	48	180	32.9	0.171	63	0
764	2	122	70	27	0	36.8	0.340	27	0
765	5	121	72	23	112	26.2	0.245	30	0
766	1	126	60	0	0	30.1	0.349	47	1
767	1	93	70	31	0	30.4	0.315	23	0

768 rows × 9 columns

Next steps: View recommended plots

```
#COLUMNS
columns = df.columns.tolist()
print(columns)
```

```
['Pregnancies', 'Glucose', 'BloodPressure', 'SkinThickness', 'Insulin', 'BMI', 'DiabetesPedigreeFunction', 'Age', 'Outcome']
```

```
#DATA TYPES
dtype = df.dtypes
print(dtype)
```

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

```
#TOTAL RECORDS
total = df.shape[0]
print(total)
```

```
768
```

```
#FIRST 20 RECORDS
df[:20]
```

	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	0.134	29	0
8	2	197	70	45	543	30.5	0.158	53	1
9	8	125	96	0	0	0.0	0.232	54	1
10	4	110	92	0	0	37.6	0.191	30	0
11	10	168	74	0	0	38.0	0.537	34	1
12	10	139	80	0	0	27.1	1.441	57	0
13	1	189	60	23	846	30.1	0.398	59	1
14	5	166	72	19	175	25.8	0.587	51	1
15	7	100	0	0	0	30.0	0.484	32	1
16	0	118	84	47	230	45.8	0.551	31	1
17	7	107	74	0	0	29.6	0.254	31	1
18	1	103	30	38	83	43.3	0.183	33	0
19	1	115	70	30	96	34.6	0.529	32	1
20	3	126	88	41	235	39.3	0.704	27	0

```
#LAST 20 RECORDS
df[748:]
```

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
748	3	187	70	22	200	36.4	0.408	36	1
749	6	162	62	0	0	24.3	0.178	50	1
750	4	136	70	0	0	31.2	1.182	22	1
751	1	121	78	39	74	39.0	0.261	28	0
752	3	108	62	24	0	26.0	0.223	25	0
753	0	181	88	44	510	43.3	0.222	26	1
754	8	154	78	32	0	32.4	0.443	45	1
755	1	128	88	39	110	36.5	1.057	37	1
756	7	137	90	41	0	32.0	0.391	39	0
757	0	123	72	0	0	36.3	0.258	52	1
758	1	106	76	0	0	37.5	0.197	26	0
759	6	190	92	0	0	35.5	0.278	66	1
760	2	88	58	26	16	28.4	0.766	22	0
761	9	170	74	31	0	44.0	0.403	43	1
762	9	89	62	0	0	22.5	0.142	33	0
763	10	101	76	48	180	32.9	0.171	63	0
764	2	122	70	27	0	36.8	0.340	27	0
765	5	121	72	23	112	26.2	0.245	30	0
766	1	126	60	0	0	30.1	0.349	47	1
767	1	93	70	31	0	30.4	0.315	23	0

```
#RENAME THE COLUMN OUTCOME INTO DIAGNOSIS
df.rename(columns={'Outcome':'Diagnosis'}, inplace=True)
df
```

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Diagnosis
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
...	...	...	...	...	...	...	...	...	...
763	10	101	76	48	180	32.9	0.171	63	0
764	2	122	70	27	0	36.8	0.340	27	0
765	5	121	72	23	112	26.2	0.245	30	0
766	1	126	60	0	0	30.1	0.349	47	1
767	1	93	70	31	0	30.4	0.315	23	0

768 rows × 9 columns

Next steps: View recommended plots

```
#CREATE CLASSIFICATION WHERE THE VALUE DEPENDS ON DIAGNOSIS
df['Classification'] = np.where(df['Diagnosis']==1, 'Diabetes', 'No Diabetes')
df
```

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Diagnosis	Classification
0	6	148	72	35	0	33.6	0.627	50	1	Diabetes
1	1	85	66	29	0	26.6	0.351	31	0	No Diabetes
2	8	183	64	0	0	23.3	0.672	32	1	Diabetes
3	1	89	66	23	94	28.1	0.167	21	0	No Diabetes
4	0	137	40	35	168	43.1	2.288	33	1	Diabetes
...	...	...	...	...	...	...	...	...	...	...
763	10	101	76	48	180	32.9	0.171	63	0	No Diabetes
764	2	122	70	27	0	36.8	0.340	27	0	No Diabetes
765	5	121	72	23	112	26.2	0.245	30	0	No Diabetes
766	1	126	60	0	0	30.1	0.349	47	1	Diabetes
767	1	93	70	31	0	30.4	0.315	23	0	No Diabetes

768 rows × 10 columns

Next steps: View recommended plots

```
#CREATING NEW DATAFRAME WITHDIABETES THAT ONLY HAS DATA WITH DIABETES IN CLASSIFICATION
withDiabetes = df[df['Classification']=='Diabetes']
withDiabetes
```

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Diagnosis	Classification
0	6	148	72	35	0	33.6	0.627	50	1	Diabetes
2	8	183	64	0	0	23.3	0.672	32	1	Diabetes
4	0	137	40	35	168	43.1	2.288	33	1	Diabetes
6	3	78	50	32	88	31.0	0.248	26	1	Diabetes
8	2	197	70	45	543	30.5	0.158	53	1	Diabetes
...	...	...	...	...	...	...	...	...	...	...
755	1	128	88	39	110	36.5	1.057	37	1	Diabetes
757	0	123	72	0	0	36.3	0.258	52	1	Diabetes
759	6	190	92	0	0	35.5	0.278	66	1	Diabetes
761	9	170	74	31	0	44.0	0.403	43	1	Diabetes
766	1	126	60	0	0	30.1	0.349	47	1	Diabetes

268 rows × 10 columns

Next steps: View recommended plots

```
#CREATING NEW DATAFRAME WITHDIABETES THAT ONLY HAS DATA WITH NO DIABETES IN CLASSIFICATION
noDiabetes = df[df['Classification']=='No Diabetes']
noDiabetes
```

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Diagnosis	Classification
1	1	85	66	29	0	26.6	0.351	31	0	No Diabetes
3	1	89	66	23	94	28.1	0.167	21	0	No Diabetes
5	5	116	74	0	0	25.6	0.201	30	0	No Diabetes
7	10	115	0	0	0	35.3	0.134	29	0	No Diabetes
10	4	110	92	0	0	37.6	0.191	30	0	No Diabetes
...	...	...	...	...	...	...	...	...	...	...
762	9	89	62	0	0	22.5	0.142	33	0	No Diabetes
763	10	101	76	48	180	32.9	0.171	63	0	No Diabetes
764	2	122	70	27	0	36.8	0.340	27	0	No Diabetes
765	5	121	72	23	112	26.2	0.245	30	0	No Diabetes
767	1	93	70	31	0	30.4	0.315	23	0	No Diabetes

500 rows × 10 columns

Next steps: View recommended plots

```
#CREATING NEW DATAFRAME PEDIA THAT ONLY HAS DATA WITH LESS THAN 19 IN AGE
Pedia = df[df['Age'] <= 19 ]
Pedia
```

Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Diagnosis	Classification

```
#CREATING NEW DATAFRAME PEDIA THAT ONLY HAS DATA WITH MORE THAN 19 IN AGE
Adult = df[df['Age'] > 19 ]
Adult
```

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Diagnosis	Classificatio
0	6	148	72	35	0	33.6	0.627	50	1	Diabete
1	1	85	66	29	0	26.6	0.351	31	0	No Diabete
2	8	183	64	0	0	23.3	0.672	32	1	Diabete
3	1	89	66	23	94	28.1	0.167	21	0	No Diabete
4	0	137	40	35	168	43.1	2.288	33	1	Diabete
...	...	...	...	...	...	...	...	...	...	..
763	10	101	76	48	180	32.9	0.171	63	0	No Diabete

Next View recommended plots

```
#GETTING THE AVERAGE OF AGE AND GLUCOSE
ages = np.array(df['Age'])
glucose = np.array(df['Glucose'])
aveage = int(np.average(ages))
aveglucose = int(np.average(glucose))
print("Average Age: " + str(aveage))
print("Average Glucose: " + str(aveglucose))
```

```
Average Age: 33
Average Glucose: 120
```

```
#GETTING THE MEDIAN OF AGE AND GLUCOSE
ages = np.array(df['Age'])
glucose = np.array(df['Glucose'])
medianage = int(np.median(ages))
medianglucose = int(np.median(glucose))
print("Median Age: " + str(medianage) + " years old")
print("Median Glucose: " + str(medianglucose))
```

```
Median Age: 29 years old
Median Glucose: 117
```

```
#GETTING THE MIDDLE VALUES OF AGE AND GLUCOSE
ages = np.array(df['Age'])
glucose = np.array(df['Glucose'])
index = int(len(ages)/2)
midage = ages[index]
midglucose = glucose[index]
print("Middle Age: " + str(midage) + " years old")
print("Middle Glucose: " + str(midglucose))
```

```
Middle Age: 25 years old
Middle Glucose: 125
```

```
#GETTING THE STANDARD DEVIATION OF SKIN THICKNESS
skinthicc = np.array(df['SkinThickness'])
stdevskin = np.std(skinthicc)
print("Standard Deviation of Skin Thickness: " + str(stdevskin))
```

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

CONLUSION:

i was enlightened in handling datas and how to make handling and organizing datas using import pandas and numpy which makes it more easier as there are built in functions on the import so that you dont need to make your own anymore