

stat-analysis

February 26, 2023

#Central Tendency

```
[2]: import pandas as pd
import numpy as np
```

```
[15]: df = pd.read_excel('dat.xlsx')
df.head()
```

```
[15]:
```

	Product	Type	Sales_Region	Price	Unit_Sold
0		A	North	2345	23
1		B	East	1234	45
2		C	South	6467	15
3		B	West	2142	84
4		C	North	8764	12

1. Mean

```
[6]: mu = df['Unit_Sold'].mean()
print("Mean is: ", mu)

# Gives the average unit of product that was sold. It basically gives
# arithmetic mean
```

Mean is: 40.9

2. Median

```
[17]: med = df['Price'].median()
print("Median is: ", med)

# Gives the mid value of price for a particular product. Since the median value
# is a decimal type and no price rate is in decimal it means there is even
↪ number
# of product
```

Median is: 4779.5

3. Mode

```
[9]: from statistics import mode
md = mode(df['Product Type'])
print("Mode is: ", md)

# Gives the highest frequency of product type that was sold
```

Mode is: A

#Measure of Variation

1. Range

```
[10]: mx = max(df['Unit_Sold'])
mn = min(df['Unit_Sold'])
range = mx - mn
print("Range is: ", range)

# Gives the difference between upper and lower limit of sold product
```

Range is: 92

2. Variance

```
[11]: var = df['Unit_Sold'].var()
print("Variance is: ", var)

# Since the value of variance is very high therefore the distribution is highly
# spread and covers wide area
```

Variance is: 740.0241379310344

3. Standard Deviation

```
[5]: sd = df['Unit_Sold'].std()
print("Standard deviation is: ", sd)

# Baically gives square root of variance and concises distribution of data
```

Standard deviation is: 27.203384677849087

4. Quartile

```
[12]: q1 = np.percentile(df['Unit_Sold'], 25)
q2 = np.percentile(df['Unit_Sold'], 50)
q3 = np.percentile(df['Unit_Sold'], 75)

print("First Quartile: ", q1)
print("Second Quartile: ", q2)
print("Third Quartile: ", q3)

# First quartile gives the sold unit from lowest to highest (25%)
```

```
# Second quartile gives the median value of sold unit (50%)  
# Third quartile gives the sold unit above the median (75%)
```

First Quartile: 19.25
Second Quartile: 30.0
Third Quartile: 63.75

5. Coefficient of Variance

```
[7]: cov = sd/mu  
print("Coefficient of variance is: ", cov)  
  
# Gives ratio of standard deviation and mean which basically shows how the data  
# is relatively spread with respect to mean
```

Coefficient of variance is: 0.6651194297762613

6. Skewness

```
[13]: from scipy.stats import skew  
sn = skew(df['Price'])  
print("Skewness is: ", sn)  
  
# skew is positive hence the distribution is positive and has long right tail  
# i.e., right directed
```

Skewness is: 0.0833271872211764

7. Kurtosis

```
[14]: from scipy.stats import kurtosis  
kt = kurtosis(df['Price'])  
print("Kurtosis is: ", kt)  
  
# kurtosis is negative hence the distribution has less peak and short tail  
# and extremities
```

Kurtosis is: -1.5150512421522444

OTHER STATISTICAL FUNCTIONS

1. Correlation

```
[23]: correlate = df.corr()  
print("Correlation of data is: \n\n", correlate)  
  
# The value of correlation is negative which shows that the unit sold and price  
# are not related to each other and hence are not varying with respect to each  
↪ other
```

Correlation of data is:

	Price	Unit_Sold
Price	1.000000	-0.055865
Unit_Sold	-0.055865	1.000000

2. Sum of squares

```
[24]: sos = np.sum((df['Unit_Sold']-mu)**2)
print("Sum of square is: ", sos)

# There are two types of sum of squares i.e., total sum of squares and residual
# sum of squares. In the above, total sum of squares is given for the unit sold
# and this helps in finding the coefficient of determination, which measures
# the
# proportion of the total variation in the dependent variable that is explained
# by independent variable
```

Sum of square is: 21460.699999999997

3. Z- Score

```
[27]: zs = (df['Unit_Sold']-mu) / sd
print("Z- Score is:\n", zs)

# Gives how many standard deviations are below and above mean value of unit sold
```

Z- Score is:

0	-0.658006
1	0.150717
2	-0.952087
3	1.584362
4	-1.062368
5	0.922679
6	0.113956
7	-1.025608
8	2.099004
9	0.224237
10	-0.841807
11	-0.437446
12	-0.253645
13	0.959439
14	1.216760
15	1.547602
16	-0.621246
17	-0.805047
18	-0.584486
19	-0.437446
20	0.591838
21	1.106480
22	1.878443

```
23    -0.437446
24    -0.768287
25    -1.282929
26    -1.135888
27    -0.363925
28     0.371277
29    -1.099128
Name: Unit_Sold, dtype: float64
```