Statistics:

The branch of mathematics that deals with collecting, organizing, analyzing, and interpreting data for decision-making.

Types

- Descriptive Statistics
- Inferences
- Probability Distribution

Descriptive Statistics

A branch of statistics that summarizes and describes the main features of a dataset using measures like mean, median, mode, variance, and visualizations (charts, graphs, tables) without making predictions or inferences.

Import the Required Libraries

In [2]:
```
from sklearn.datasets import load_breast_cancer
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

Load the Dataset

In [3]:
```
df=pd.read_csv(r"C:\Users\User-PC\Downloads\Statistics-20250831T034916Z-1-001\Stati
df.head()
```

Out[3]:

	school	sex	age	address	famsize	Pstatus	Medu	Fedu	Mjob	Fjob	...	famrel
0	GP	F	18	U	GT3	A	4	4	at_home	teacher	...	4
1	GP	F	17	U	GT3	T	1	1	at_home	other	...	5
2	GP	F	15	U	LE3	T	1	1	at_home	other	...	4
3	GP	F	15	U	GT3	T	4	2	health	services	...	3
4	GP	F	16	U	GT3	T	3	3	other	other	...	4

5 rows × 33 columns

1) Measure of Central Tendacy

It summarize large datasets into a single representative value, helping AI models understand data distribution.

Techniques (Mean, Median, Mode)

Mean

Average Value Sensitive to Outliers

In [4]:
```
mn=np.mean(df['age'])
print(mn)
```

```
16.696202531645568
```

Median

Middle Value in an ordered Numemric Sequence

In [5]:
```
md=np.median(df['age'])
print(md)
```

```
17.0
```

Mode

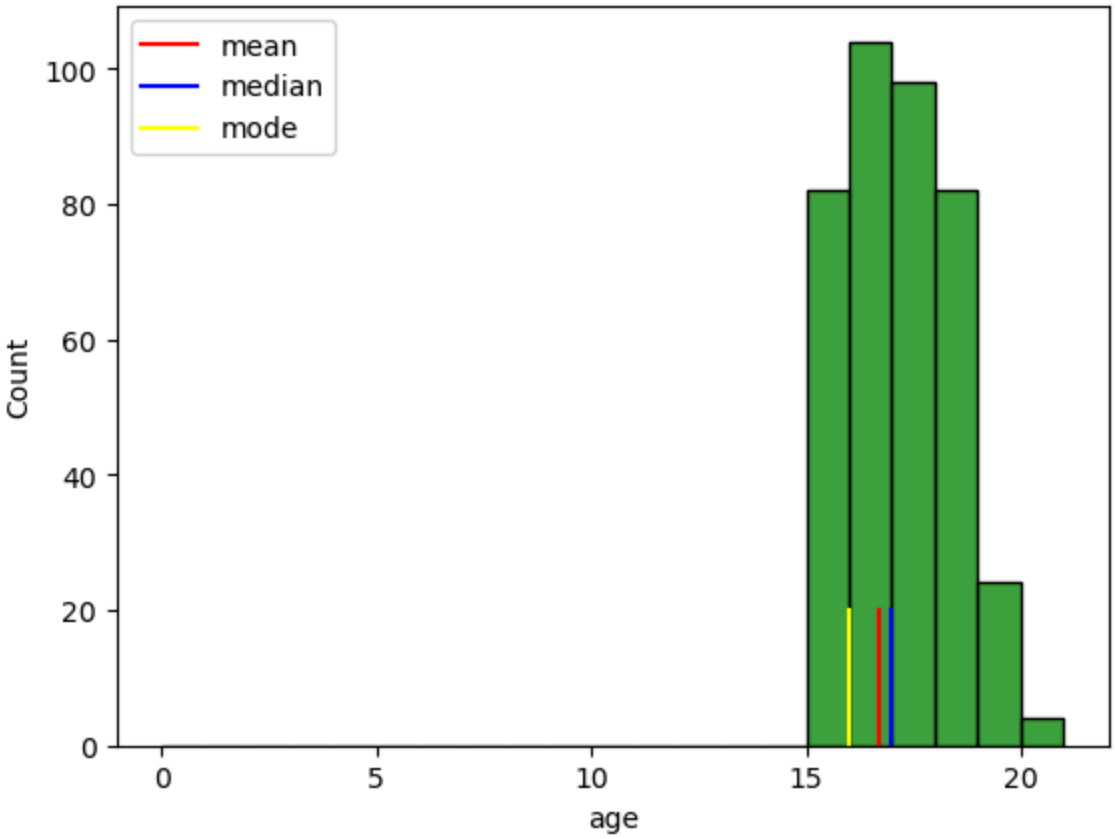
The most frequent value in the dataset.

In [6]:
```
mo=df['age'].mode()[0]
print(mo)
```

```
16
```

Visualization

In [7]:
```
sns.histplot(x='age',data=df,bins=[i for i in range(0,22,1)],color='green')
plt.plot([mn for i in range(0,21)],[i for i in range(0,21)], c='red',label='mean')
plt.plot([md for i in range(0,21)],[i for i in range(0,21)], c='blue',label='median
plt.plot([mo for i in range(0,21)],[i for i in range(0,21)], c='yellow',label="mode
plt.legend()
plt.show()
```



2) Measure of Variability

A measure of variability (or dispersion) describes how spread out or scattered the data values are around the center (mean/median). It shows the degree to which data points differ from each other.

- ## Common Measures of Variability

Range

Difference between maximum and minimum values.

In [8]:
```
min_r=df['G1'].min()
max_r=df['G1'].max()

range=max_r - min_r
range
```

Out[8]: 16

In [9]:
```
class_1=np.array([75,65,73,68,72,76])
class_2=np.array([90,47,43,96,93,51])
no = [1,2,3,4,5,6]
```

Standard Deviation

It is a measure of amount of variation or dispersion of set of values. A low standard deviation indicates the value tends to be close to mean and higher S.D tells that values are spread out over a wide range.

Example:

In [10]:
```
np.std(class_1),np.std(class_2)
```

Out[10]: (np.float64(3.8622100754188224), np.float64(23.18045153428495))

Variance

Average of squared differences from the mean.

In [11]:
```
np.var(class_1), np.var(class_2)
```

Out[11]: (np.float64(14.916666666666666), np.float64(537.3333333333334))

Mean Absolute Deviation

$$\text{MAD} = \frac{\sum |x_i - \bar{x}|}{n}$$

The mean Absolute deviation of a dataset is the average distance between each data point and the mean. It gives us idea about the dispersion in a dataset.

Example:

In [12]:
```
mean=np.mean(class_1)
mean
```

Out[12]: np.float64(71.5)

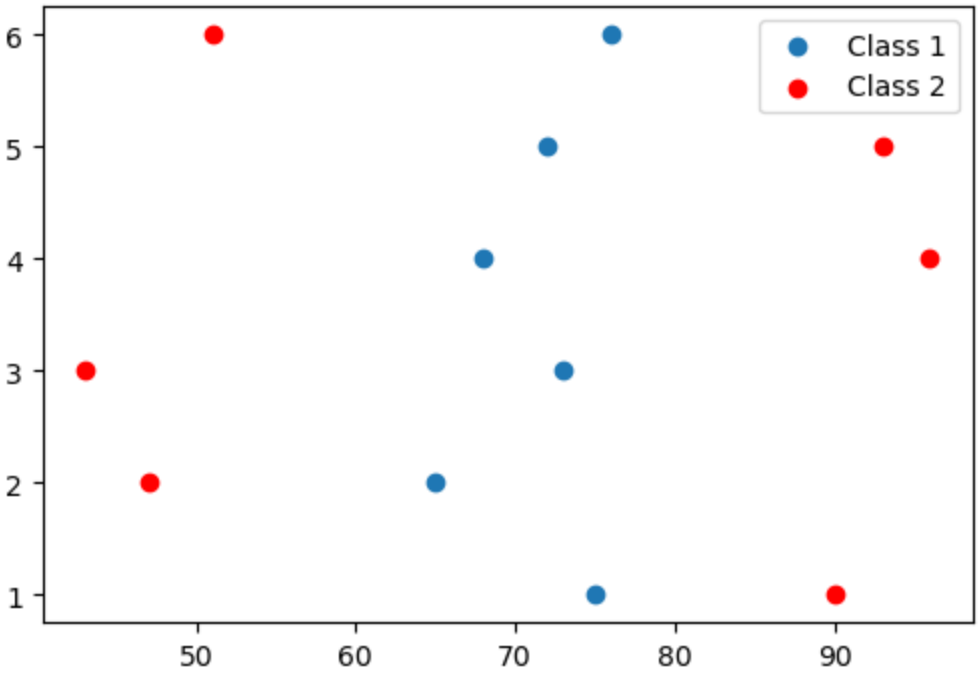
In [13]:
```
class_1_mad=np.sum(np.abs(class_1-mean)/len(class_1))
class_2_mad=np.sum(np.abs(class_2-mean)/len(class_2))
```

In [14]:
```
class_1_mad,class_2_mad
```

Out[14]:
```
(np.float64(3.3333333333333335), np.float64(23.0))
```

Visualization

In [15]:
```
plt.figure(figsize=(6,4))
plt.scatter(class_1,no,label='Class 1')
plt.scatter(class_2,no, color='red',label='Class 2')
plt.legend()
plt.show()
```



Percentiles & Quartiles

Percentiles:

Percentiles divide ordered data into 100 equal parts. The k-th percentile is the value below which k% of the data falls.

Example: If a student's test score is at the 90th percentile, they scored better than 90% of students.

In [16]:
```
q1 = np.percentile(df['age'], 25)
q2 = np.percentile(df['age'], 50)
q3 = np.percentile(df['age'], 75)
```

```
print("Q1:", q1, "Q2:", q2, "Q3:", q3, "IQR:", q3-q1)
```

```
Q1: 16.0 Q2: 17.0 Q3: 18.0 IQR: 2.0
```

Quartiles

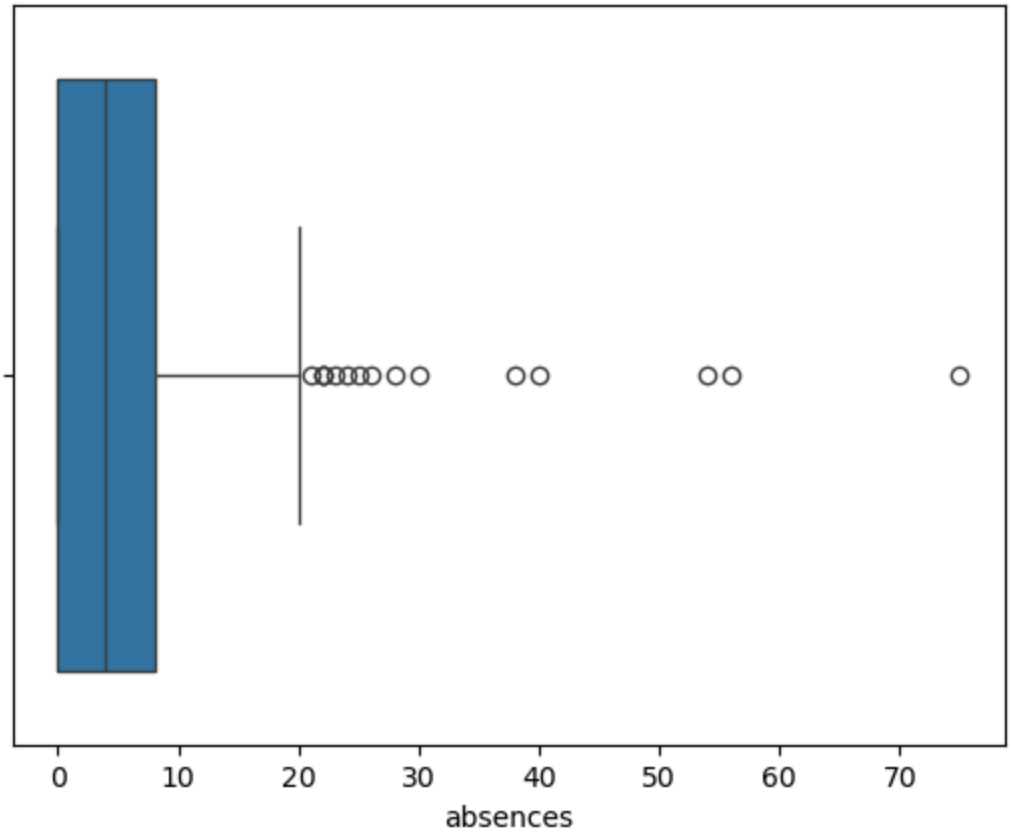
- Quartiles divide ordered data into 4 equal parts (25% each).
- Q1 (25th percentile): 25% of data is below this value.
- Q2 (50th percentile / Median): Middle of the data.
- Q3 (75th percentile): 75% of data is below this value.
- IQR (Interquartile Range): Q3 − Q1 → measures spread of the middle 50% (helps detect outliers).

In [17]:
```
df.describe()
```

Out[17]:

	age	Medu	Fedu	traveltime	studytime	failures	famrel
count	395.000000	395.000000	395.000000	395.000000	395.000000	395.000000	395.000000
mean	16.696203	2.749367	2.521519	1.448101	2.035443	0.334177	3.944304
std	1.276043	1.094735	1.088201	0.697505	0.839240	0.743651	0.896659
min	15.000000	0.000000	0.000000	1.000000	1.000000	0.000000	1.000000
25%	16.000000	2.000000	2.000000	1.000000	1.000000	0.000000	4.000000
50%	17.000000	3.000000	2.000000	1.000000	2.000000	0.000000	4.000000
75%	18.000000	4.000000	3.000000	2.000000	2.000000	0.000000	5.000000
max	22.000000	4.000000	4.000000	4.000000	4.000000	3.000000	5.000000

In [18]:
```
sns.boxplot(x='absences',data=df)
plt.show()
```



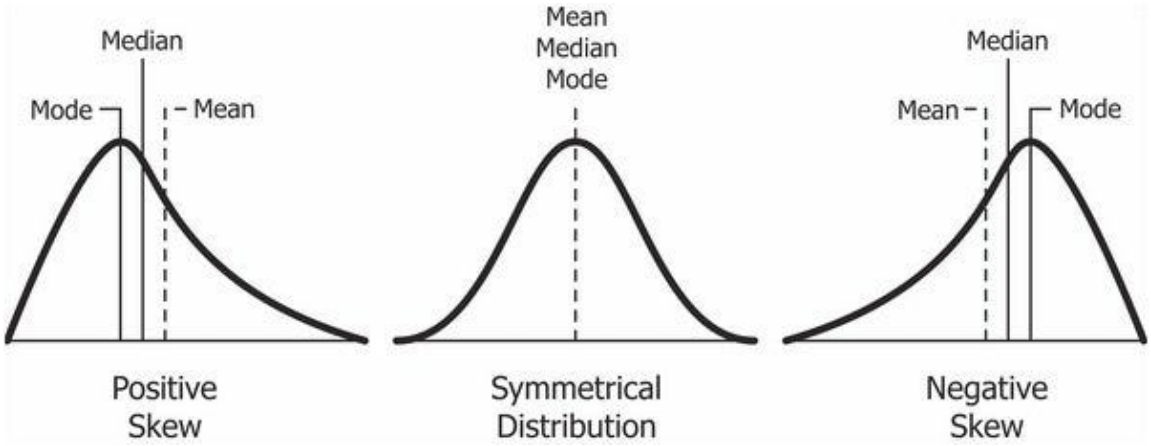
Measure of Shapes

1.) Skewness

$$\text{Skewness} = \frac{\sum(x_i - \bar{x})^3}{(N-1)\cdot\sigma^3}$$

It tells where the distribution is symmetrical or tilted. There are two types of Skewness.

- Positive Skewness- Tail on the right (Mode < Median < Mode)
- Negative Skewness- Tail on the left (Mean < Median < Mode)

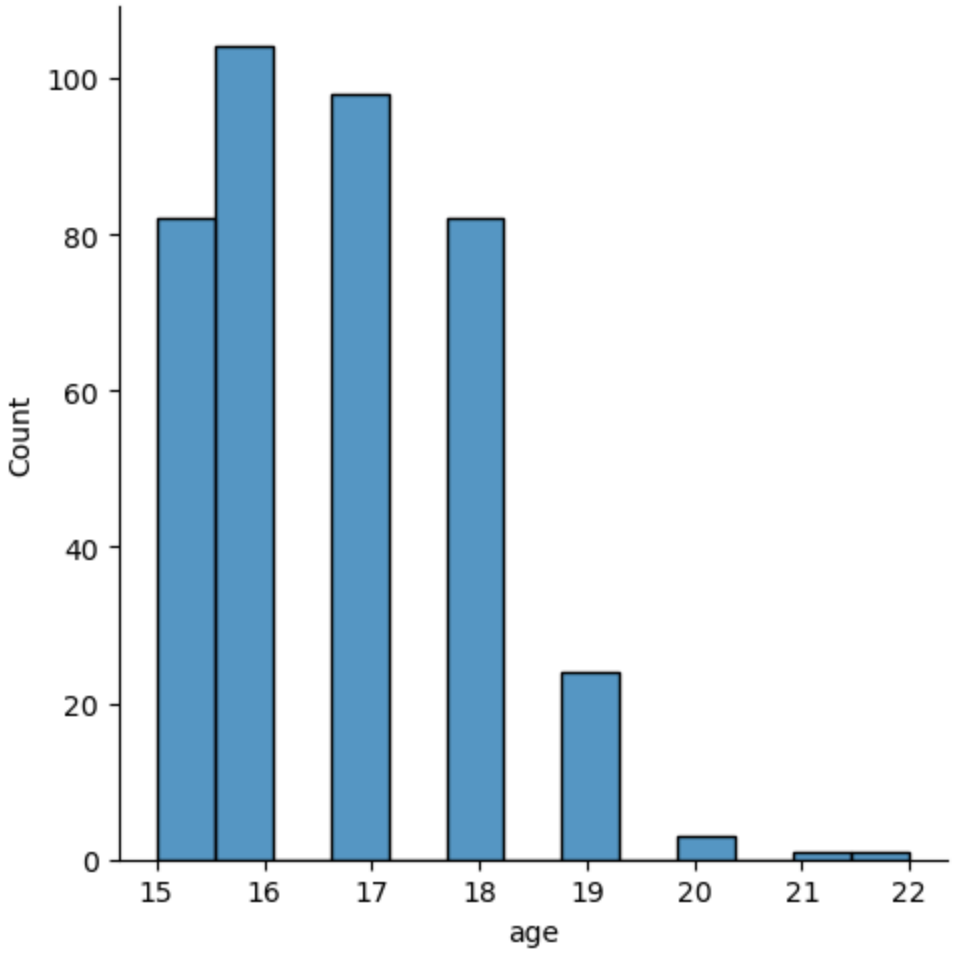


Positive Skew

In [19]:
```
df['age'].skew()
```

Out[19]:
```
np.float64(0.46627016141836425)
```

In [20]:
```
sns.displot(df['age'])
plt.show()
```

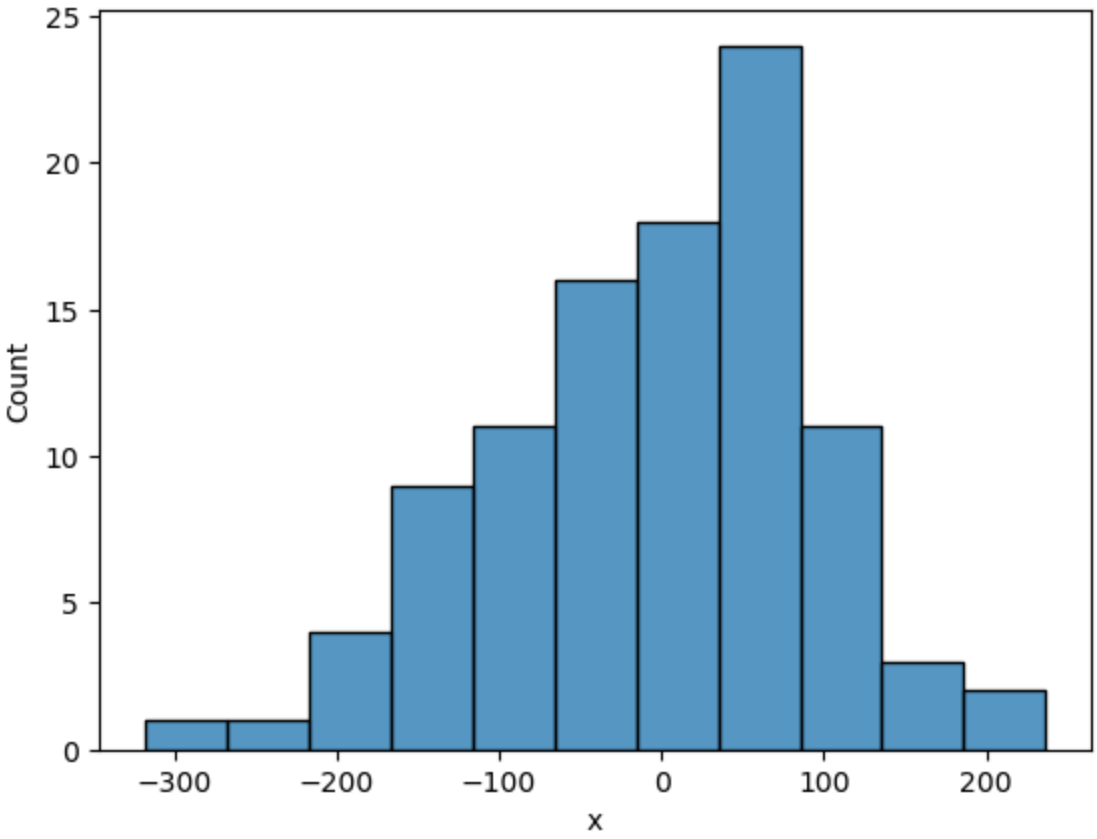


Negative Skew

In [21]:
```
data= np.random.normal(0,100,100)
df_2=pd.DataFrame({"x":data})
df_2['x'].skew()
```

Out[21]: np.float64(-0.41915340636494075)

In [22]:
```
sns.histplot(df_2['x'])
plt.show()
```



In [23]:
```
df_2['x'].mean(),df_2['x'].median()
```

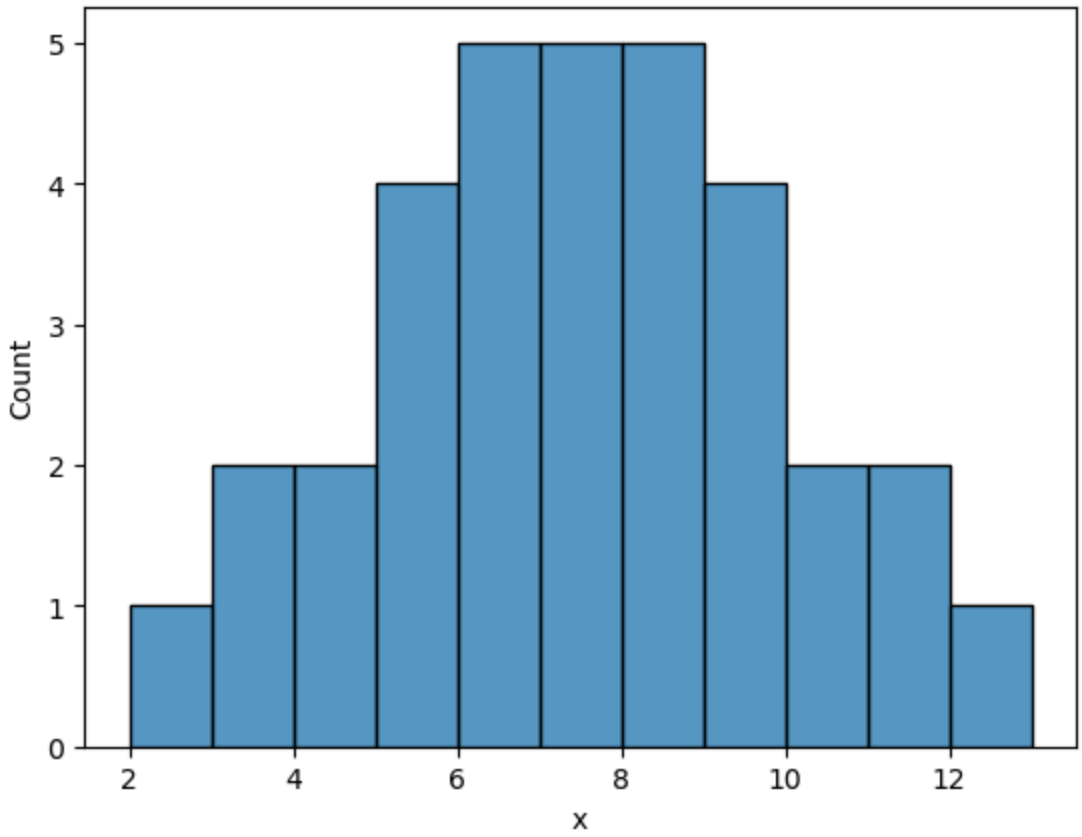
Out[23]: (np.float64(-5.4244234194927525), -3.100787838801281)

Symmetric Distribution

In [24]:
```
data_2=[2,3,3,4,4,5,5,5,5,6,6,6,6,6,7,7,7,7,7,8,8,8,8,8,9,9,9,9,10,10,11,11,12]
df_3=pd.DataFrame({"x":data_2})
df_3['x'].skew()
```

Out[24]: np.float64(0.0)

In [25]:
```
sns.histplot(x='x',data=df_3,bins=[2,3,4,5,6,7,8,9,10,11,12,13])
plt.show()
```



In [26]:
```
print(df_3['x'].mean(), df_3['x'].median(), df_3['x'].mode())
```

```
7.0 7.0 0    6
1    7
2    8
Name: x, dtype: int64
```

Probability

Probability measures the likelihood of a particular outcome or event occuring. It is typically expressed as a number between 0 and 1, where 0 indicates impossiblity (event will not occur) and 1 indicates event certainity (event will occur).

- P(A) = Number of times A occur /Total number of possible outcomes

Random Variable

A random variable is a variable whose possible values are outcomes of a random experiment. It assigns a numeric value to each outcome.

Types of Random Variable

1.) Discrete Random Variable:

Takes countable values (finite or infinite).

Example: Number of students in a class, number of heads in 10 coin tosses.

2.) Continuous Random Variable:

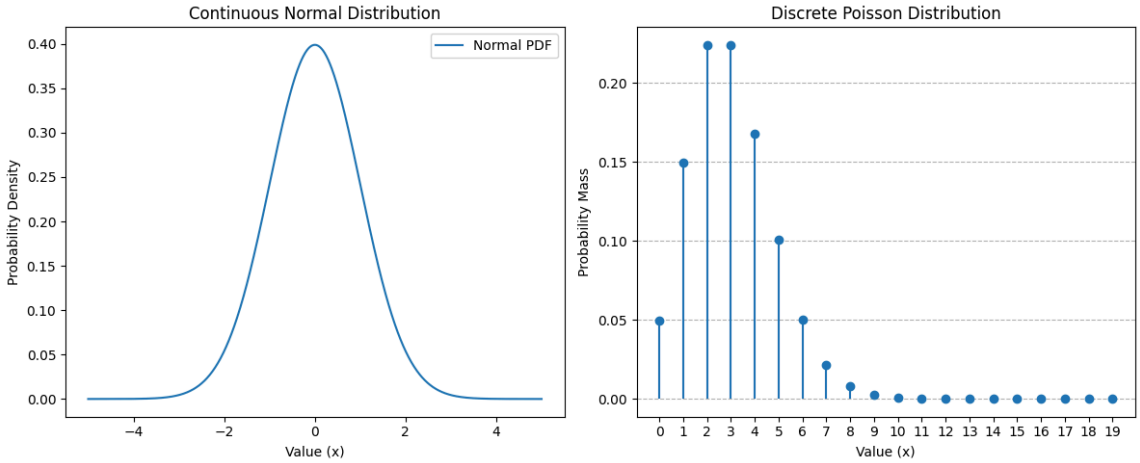
Takes any value within a range (infinite, uncountable).

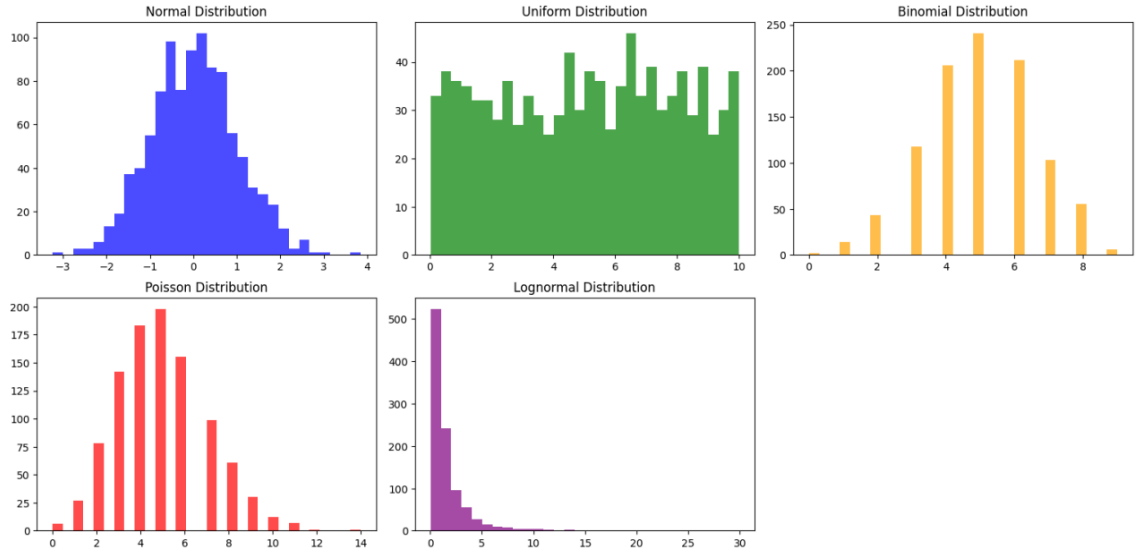
Example: Height of students, time taken to complete a task, temperature.

Probability Distribution

Probability distribution describes how the probabilities of different outcomes are distributed over the sample space of random variable.

- Discrete Probability Distribution
- Continuous Probability Distribution



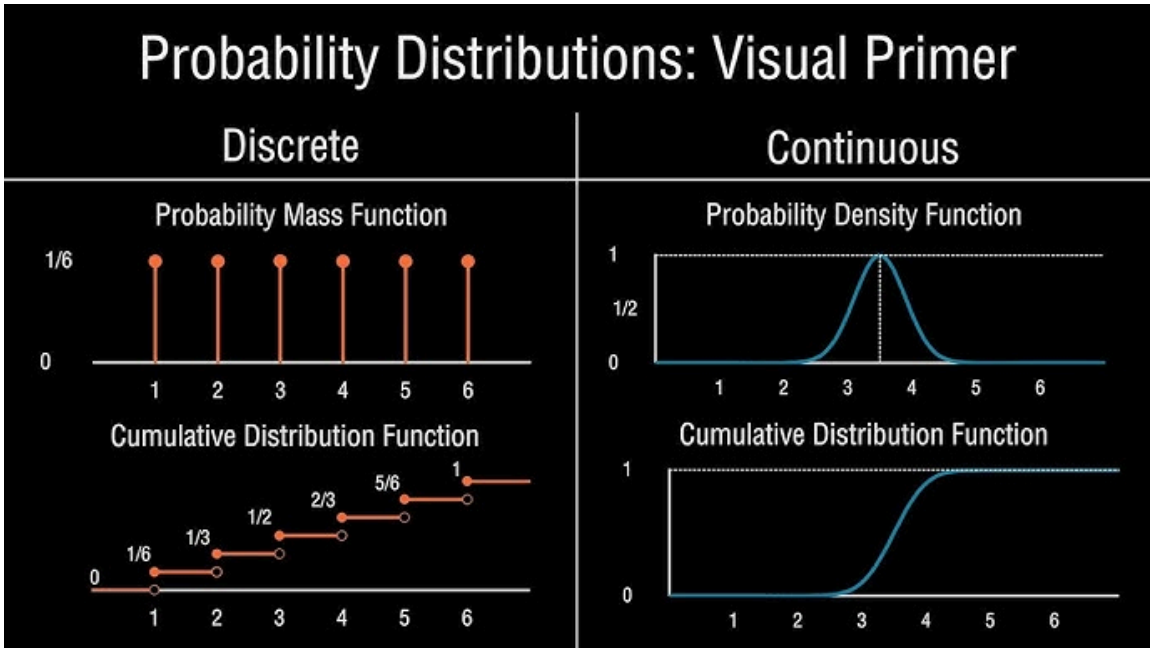


How Probability Distributions Shape Our World: A Dive into Normal, Uniform, Binomial, Poisson, and Lognormal Distributions

Probability Distribution Function

It is a mathematical function that gives the probabilities of occurence of different possible outcomes for an expirement.

- Probability Distributive Function (PDF)
- Probability Mass Function (PMF)
- Cumulative Density Function (CDF)

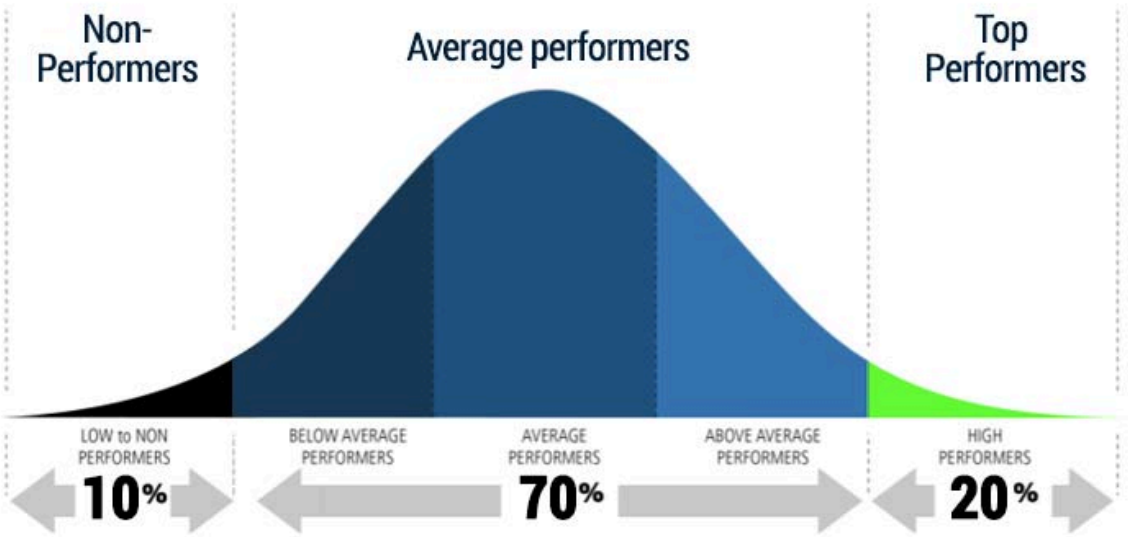


Normal Distribution

It is known as Gaussian Distribution, that is symmetic about the mean, showing that the data near the mean are more frequent in occurence than the data from the mean.

- Formula:

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2}$$



- In graph form normal distribution will appear as a bell curve

Standard Normal Distribution

- The Standard normal distribution, as known as Z-distribution or Z-score, is a special case of the normal distribution.
- mean(u) of 0 and a standard deviation of 1.

Covariance & Correlation

- Covariance signifies the direction of the linear relationship between the two variables. By direction we mean if the variable is directly propotional or inversely propotional to each other.

$$Cov(x,y) = \frac{\sum (x_i - \bar{x}) * (y_i - \bar{y})}{N}$$

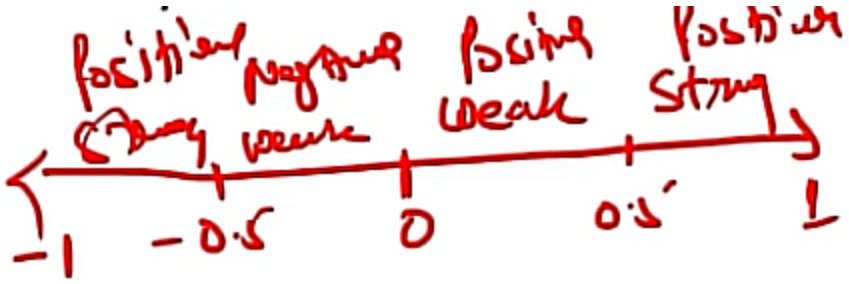
- Increasing the value of one variable might have a positive or negative impact on the value of other variable.)
- x- Positive, y- Positive -> Positive Covariance/Correlation
- x- Negative, y- Positive -> Negative Covariance/Correlation
- x- Positive, y- Negative -> 0 Covariance/Correlation

Correlation

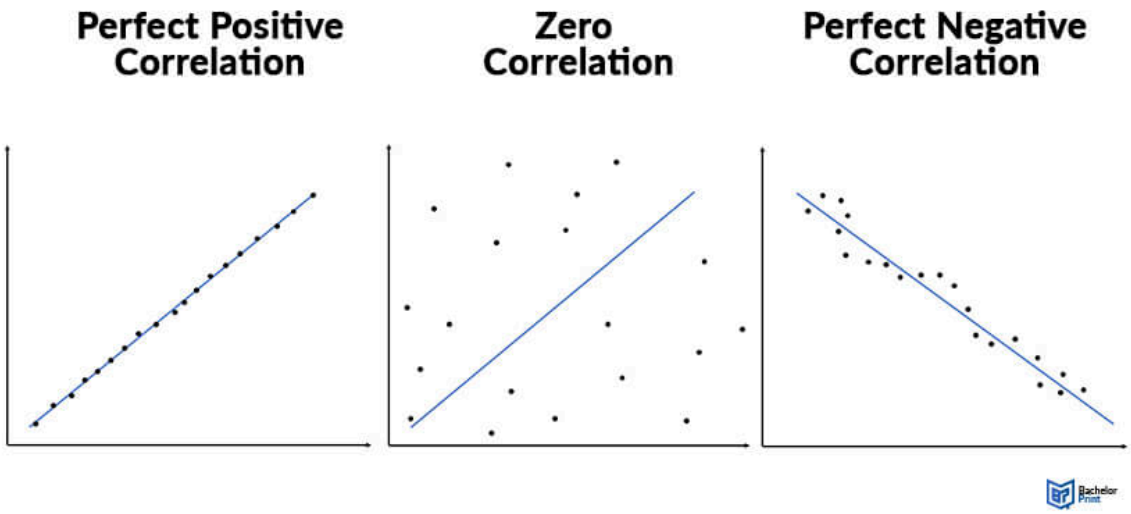
- Correlation analysis is a method of statistical evaluation used to study the strength of a relationship between two, numerically measured, continuous variables.

$$Correlation = \frac{Cov\ (x, y)}{\sigma x * \sigma y}$$

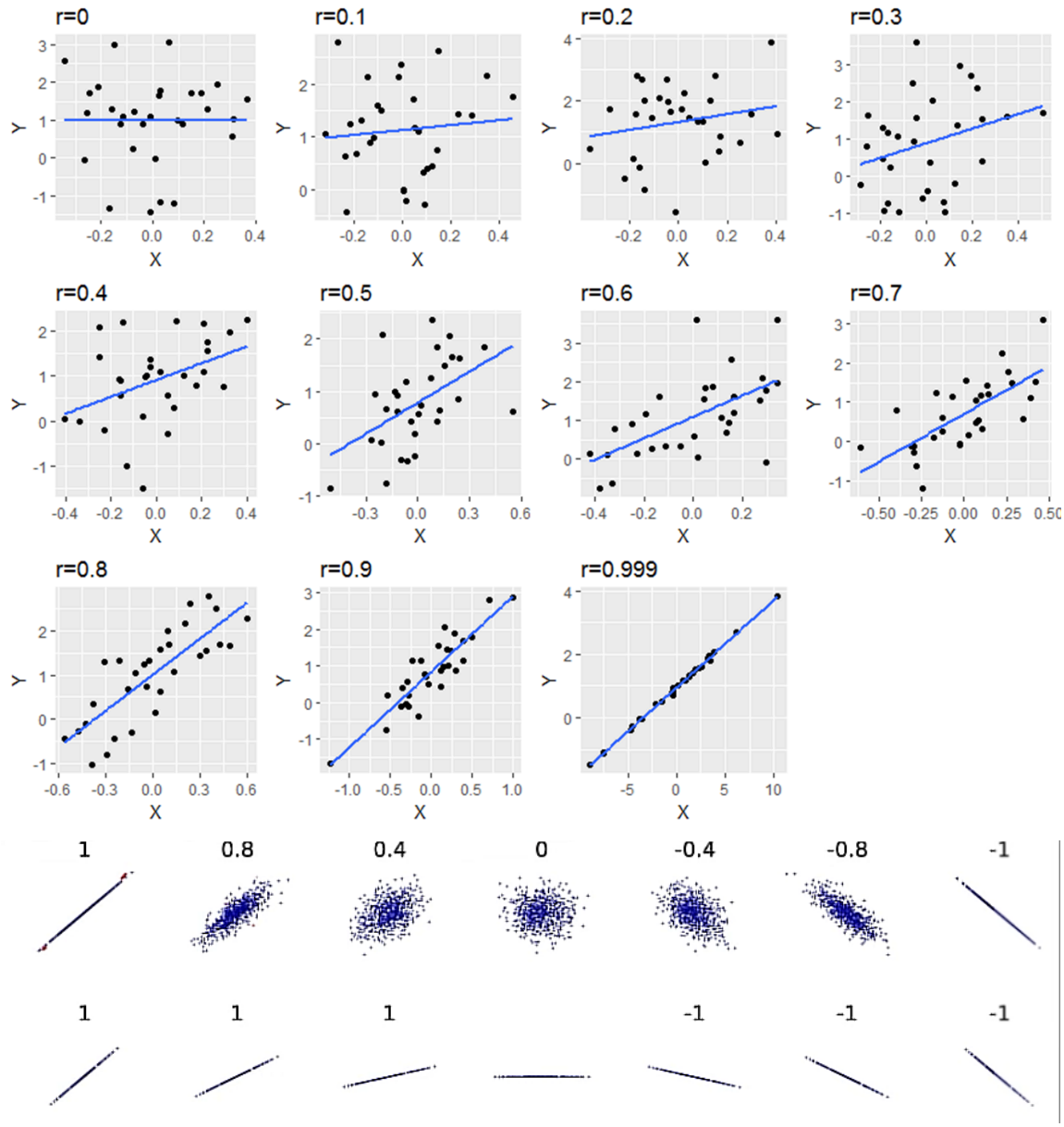
- where Cov is the covariance
- varianc x is the standard Deviation of x
- variance y is the standard deviation of y



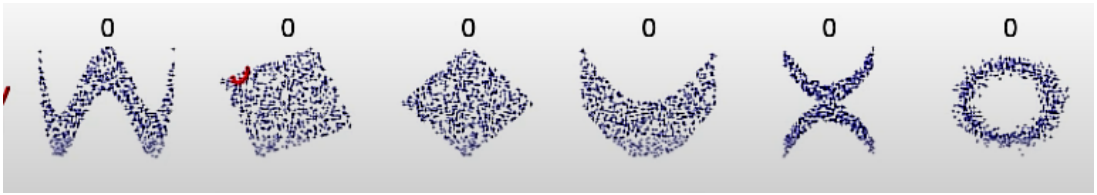
Correlation Graph:



Pearson Correlation Graph



- We notice that graph is getting scattered towards 0 and -0 values



Example

In [27]:
```
df.head(3)
```

Out[27]:

	school	sex	age	address	famsize	Pstatus	Medu	Fedu	Mjob	Fjob	...	famrel
0	GP	F	18	U	GT3	A	4	4	at_home	teacher	...	4
1	GP	F	17	U	GT3	T	1	1	at_home	other	...	5
2	GP	F	15	U	LE3	T	1	1	at_home	other	...	4

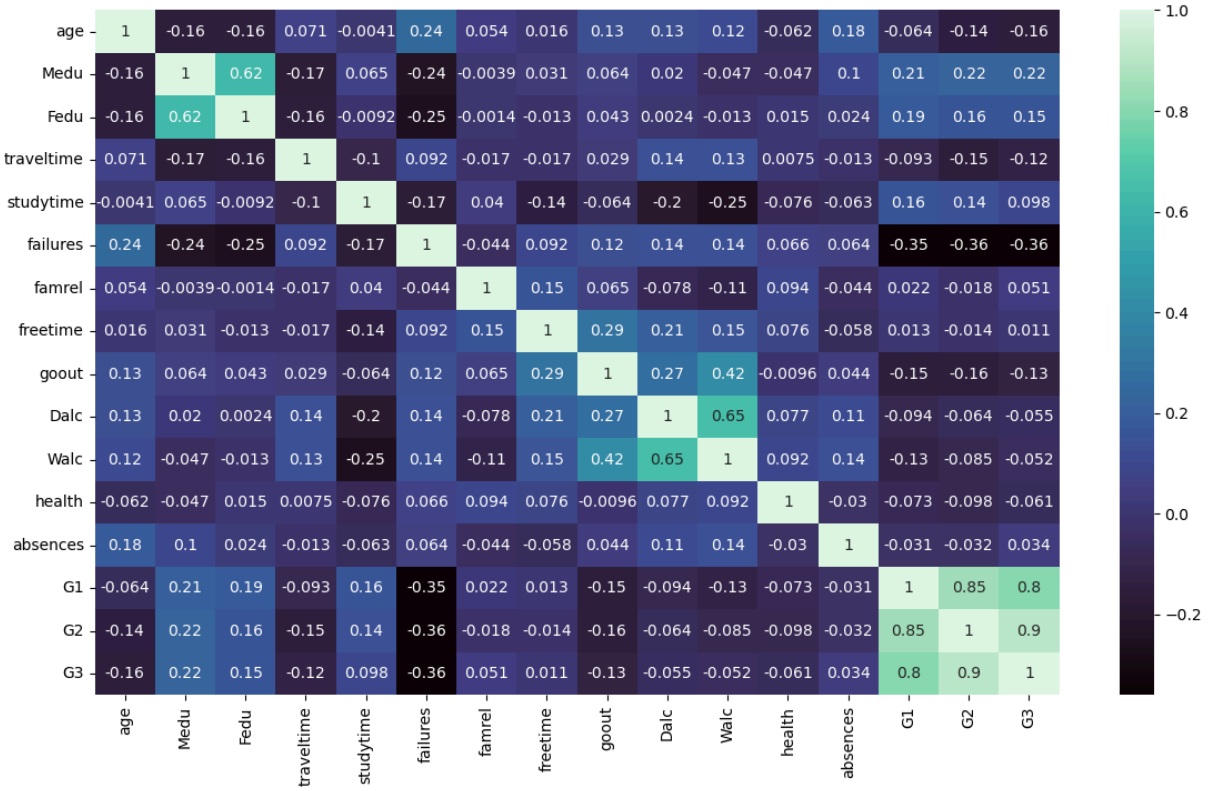
3 rows × 33 columns

In [28]:
```
data_corr=df.select_dtypes(include= ['int']).corr()
data_corr
```

Out[28]:

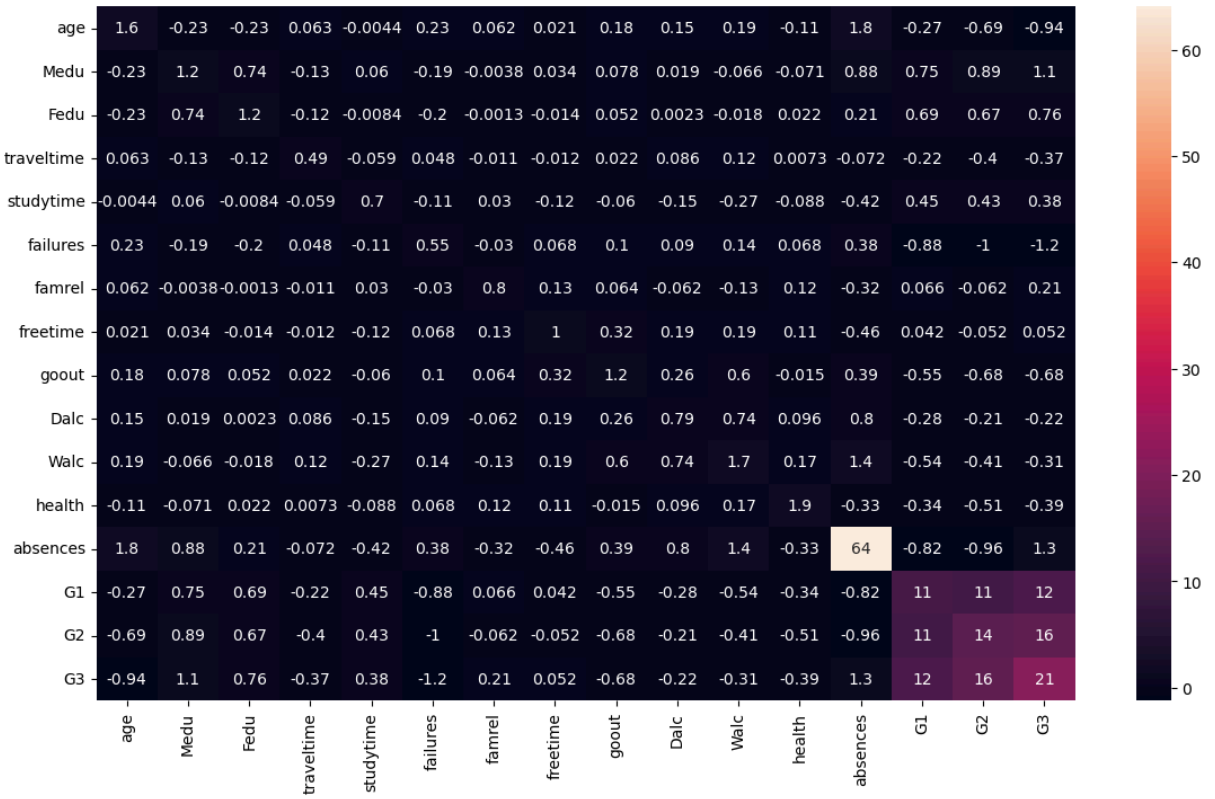
	age	Medu	Fedu	traveltime	studytime	failures	famrel	fr
age	1.000000	-0.163658	-0.163438	0.070641	-0.004140	0.243665	0.053940	0.
Medu	-0.163658	1.000000	0.623455	-0.171639	0.064944	-0.236680	-0.003914	0.
Fedu	-0.163438	0.623455	1.000000	-0.158194	-0.009175	-0.250408	-0.001370	-0.
traveltime	0.070641	-0.171639	-0.158194	1.000000	-0.100909	0.092239	-0.016808	-0.
studytime	-0.004140	0.064944	-0.009175	-0.100909	1.000000	-0.173563	0.039731	-0.
failures	0.243665	-0.236680	-0.250408	0.092239	-0.173563	1.000000	-0.044337	0.
famrel	0.053940	-0.003914	-0.001370	-0.016808	0.039731	-0.044337	1.000000	0.
freetime	0.016434	0.030891	-0.012846	-0.017025	-0.143198	0.091987	0.150701	1.
goout	0.126964	0.064094	0.043105	0.028540	-0.063904	0.124561	0.064568	0.
Dalc	0.131125	0.019834	0.002386	0.138325	-0.196019	0.136047	-0.077594	0.
Walc	0.117276	-0.047123	-0.012631	0.134116	-0.253785	0.141962	-0.113397	0.
health	-0.062187	-0.046878	0.014742	0.007501	-0.075616	0.065827	0.094056	0.
absences	0.175230	0.100285	0.024473	-0.012944	-0.062700	0.063726	-0.044354	-0.
G1	-0.064081	0.205341	0.190270	-0.093040	0.160612	-0.354718	0.022168	0.
G2	-0.143474	0.215527	0.164893	-0.153198	0.135880	-0.355896	-0.018281	-0.
G3	-0.161579	0.217147	0.152457	-0.117142	0.097820	-0.360415	0.051363	0.

In [29]:
```
plt.figure(figsize=(14,8))
sns.heatmap(data_corr, annot=True, cmap='mako')
plt.show()
```



In [30]:
```
data_cov=df.select_dtypes(include= ['int']).cov()
```

In [31]:
```
plt.figure(figsize=(14,8))
sns.heatmap(data_cov, annot=True)
plt.show()
```

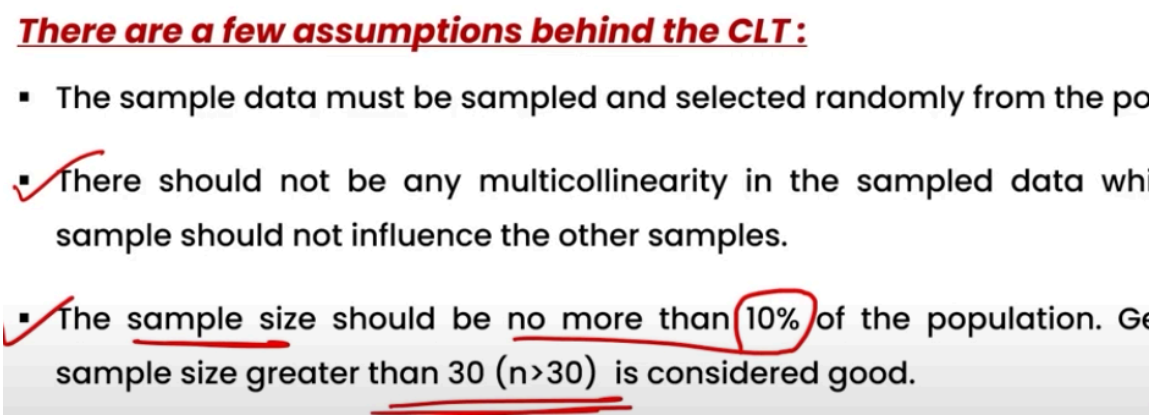


Inferential Statistics

It is a branch of statistics that uses data from a sample to draw conclusions or make predictions about a larger population. It involves using mathematical analysis, such as hypothesis testing and confidence intervals, to make generalizations and assess the validity of hypotheses about the whole group from which the sample was drawn

Central Limit Theorem

The Central Limit Theorem states that when plotting a sample distribution of means the means of the sample will be equal to the population mean and the sample distribution will approach sample distribution with variance equal to standard error.



In [25]:
```
import pandas as pd
import numpy as np
import random
import seaborn as sns
import matplotlib.pyplot as plt
```

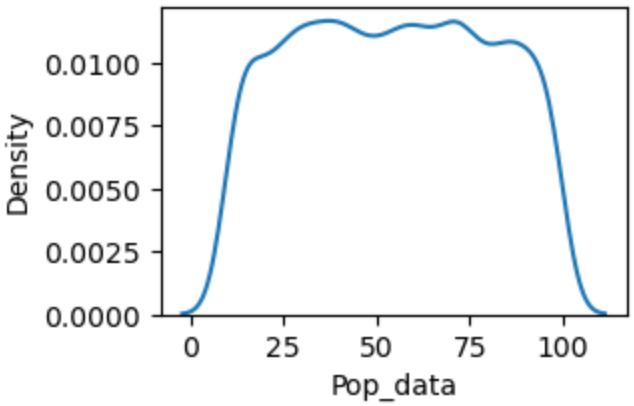
In [26]:
```
pop_data = [np.random.randint(10, 100) for i in range(10000)]
pop_table=pd.DataFrame({'Pop_data': pop_data})
pop_table
```

Out[26]:

	Pop_data
0	73
1	49
2	75
3	75
4	21
...	...
9995	24
9996	60
9997	31
9998	75
9999	42

10000 rows × 1 columns

In [27]:
```
plt.figure(figsize=(3,2))
sns.kdeplot(x="Pop_data",data=pop_table)
plt.show()
```

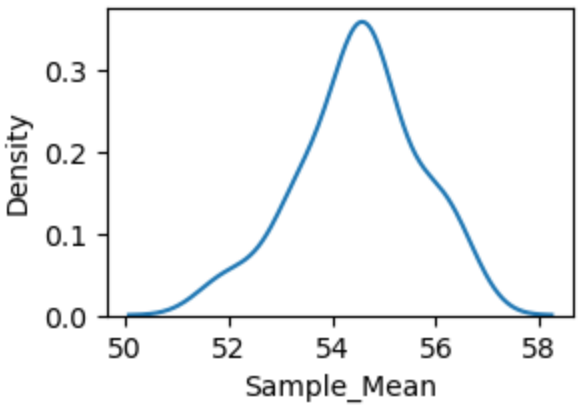


In [28]:
```
sam_mean=[]

for no_sample in range (50):
    sample_data=[]
    for data in range(500):
        sample_data.append(np.random.choice(pop_data))
    sam_mean.append(np.mean(sample_data))
```

In [29]:
```
sample_M=pd.DataFrame({'Sample_Mean':sam_mean})
```

In [30]:
```
plt.figure(figsize=(3,2))
sns.kdeplot(x="Sample_Mean",data=sample_M)
```

```
plt.show()
```



Hypothesis Testing

- It is a part of statistical analysis, where we tet the assumptions made regarding a population parameter.
- It is generally used when we were to compare a single group with an external standard and two or more groups with each other

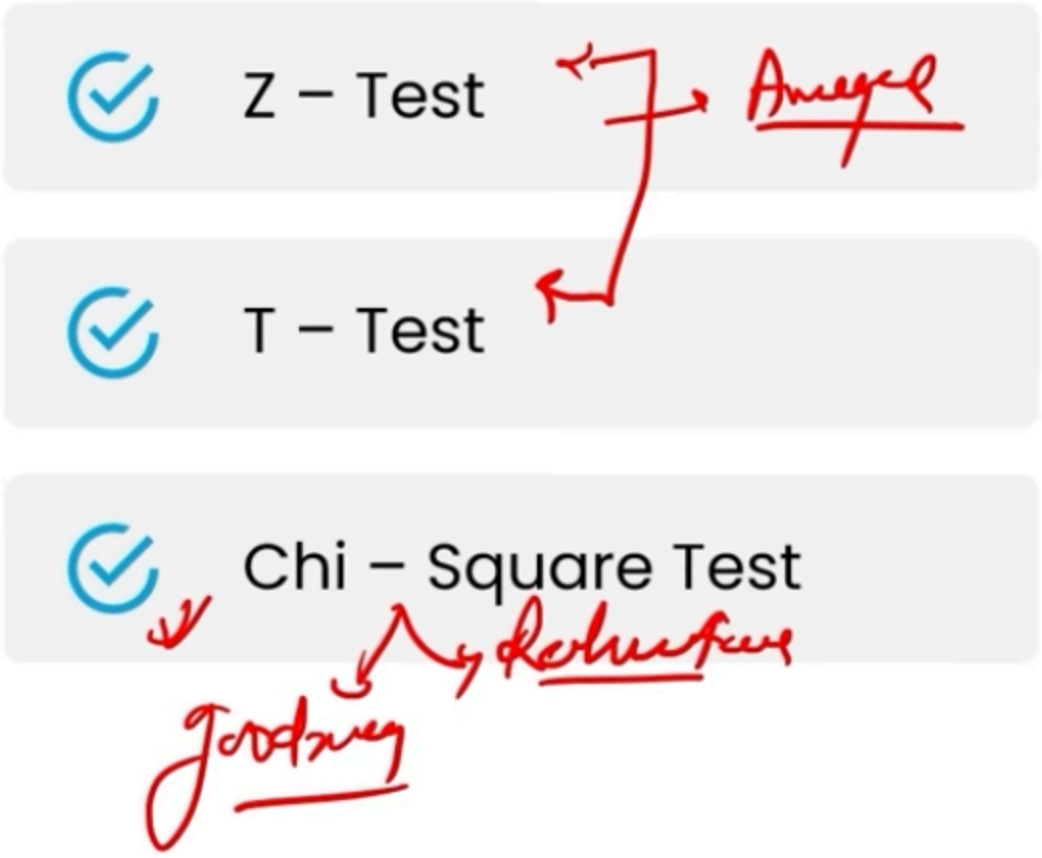
Null Hypothesis Testing

Null Hypothesis is a stastical theory that suggests there is no statistical significance exists between the population. It is denoted by HO and reaad as H-naught.

Alternative Hypothesis:

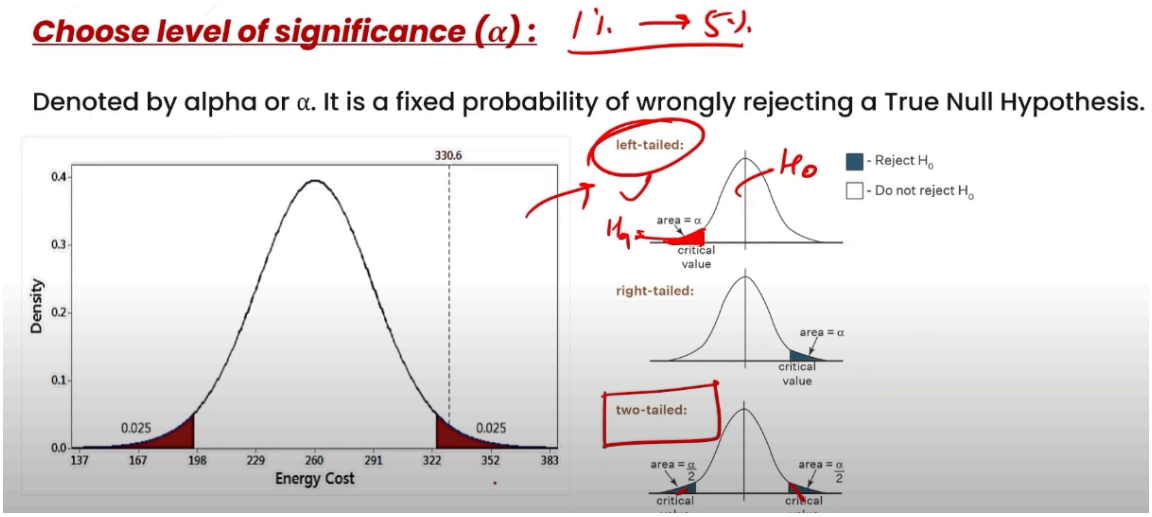
An Alternative hypothesis suggests there is a statistical difference between the population parameters. It could be greater. Basically, it is the contrast of the Null Hypothesis, it is denoted by H1 or Ha

Types of Hypothesis Testing



Steps of Hypothesis Testing:

- State null(H0) and alternative(H1) hypothesis
- Choose level of significance(a)
- Find critical values
- Find test statistic
- Draw your conclusion



Z-Test

A z-test checks if a sample mean differs from a known/target population mean when the population standard deviation (σ) is known (or n is large so σ≈s works via CLT).

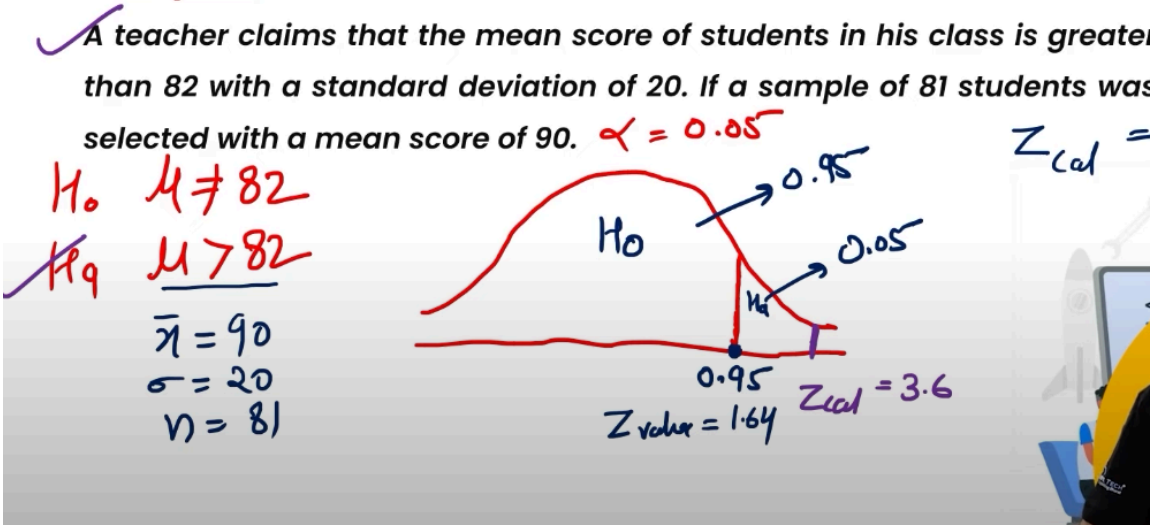
$$z = \frac{\bar{x} - \mu_0}{\sigma/\sqrt{n}}$$

¯x = sample mean μ0= hypothesized mean, σ = population SD, n = sample size.

Compare |z| to the standard normal (N(0,1)) to get the p-value.

Assumptions (practical):

Random/independent sample, 2) σ known (or n large), 3) Data roughly normal or n large (CLT)



Example:

Suppose the average exam score of students in a university is known to be 70 with a standard deviation of 10. A professor believes his class performs better, so he takes a sample of 30 students with an average score of 74. We want to test (using z-score) if his class mean is significantly higher.

In [31]:
```
import numpy
import math
import scipy.stats as st
from scipy.stats import norm

alpha=0.05
mu0=70 #sample mean
sigma=10 #population SD(assumed known)
n=30 #sample size
x_bar=74  #observed_mean

# Step 1: Calculate z-score
z_score = (x_bar - mu0) / (sigma / math.sqrt(n))

# Step 2: Get critical z for two-tailed test
z_critical = norm.ppf(1 - alpha/2)  # positive side

# Step 3: Apply if-else logic
if abs(z_score) > z_critical:
    print(f"Reject H0 | z_score={z_score:.3f}, critical={z_critical:.3f}")
else:
    print(f"Fail to Reject H0 | z_score={z_score:.3f}, critical={z_critical:.3f}")
```

```
Reject H0 | z_score=2.191, critical=1.960
```

Since Z-score is greater than Critical Z-score, we reject H0.

Example 2 (Z Test)

- Scenario: Imagine you work for an ecommerce comapny, and your team is responsible for analyzing customer purchase data. You want to determine whether new website design has led to significant increase in the average purchase amoumt as comapred to old design.
- Data: You have collected data from a random sample of 30 customers who made the purhcase on the old website design and 30 customers who made purchase on the new website design. You have the sample mean , sample S.D, and sample size for both groups.
- H0 (new website == old website)
- H1 (new website > old website)

In [32]:
```
old_data= np.array([51.96, 46.03, 60.04, 48.90, 42.19, 54.52, 40.66, 63.90, 47.40,
        62.03, 65.60, 37.14, 51.13, 45.24, 55.74, 44.91, 49.61, 61.75, 52.94,
        44.74, 42.48, 42.42, 52.93, 47.75, 55.20, 33.34, 42.56, 52.48, 46.54])

new_data=np.array([56.35, 52.13, 62.76, 52.92, 66.60, 58.44, 49.97, 67.54, 61.04, 5
    50.72, 48.77, 63.33, 58.41, 54.96, 64.83, 53.12, 57.89, 61.12, 55.66,
    60.51, 56.22, 45.88, 59.74, 57.98, 52.87, 63.14, 54.39, 58.30, 60.51])
```

In [33]:
```
pop_std=2.5
n_sp = len(new_data)
mean_new=np.mean(new_data)
mean_old=np.mean(old_data)
ap=0.05
```

In [34]:
```
z_table=st.norm.ppf(1-ap)
print(f"{z_table:.4f}")
```

```
1.6449
```

In [35]:
```
z_cal=(mean_new - mean_old) /(pop_std / np.sqrt(n_sp))
print(f"The calculated Z-score is: {z_cal:.3f}")
```

```
The calculated Z-score is: 16.323
```

In [36]:
```
if z_cal > z_table:
    print ("HA is correct. The new website design has led to significant increase i
else:
    print("H0 is correct. Old design is still leading in sales.")
```

```
HA is correct. The new website design has led to significant increase in purchase am
ount.
```

T-test

A t-test is a statistical hypothesis test used to determine whether there is a significant difference between the means of one or two groups, especially when the population standard deviation (σ) is unknown and the sample size is small (n < 30).

Formula (One-Sample t-test)

$$t = \frac{\bar{X} - \mu_0}{\frac{s}{\sqrt{n}}}$$

Where:

- $\bar{X}$ = sample mean
- μ_0 = population mean (hypothesized)
- s = sample standard deviation
- n = sample size

This compares the sample mean to a hypothesized population mean.

Example

- A manufacturer claims that the average weight of a bag of potato chips is 150 grams. A sample of 25 bags is taken, and the average weight is found to be 148 grams, with a standard deviation of 5 grams. Test the manufacturer's claim using a one-tailed t-test with a significance level of 0.05

In [37]:
```
import scipy.stats as st
```

In [38]:
```
t =st.t.ppf(0.05, 24)
```

In [39]:
```
x_bar=148
pop_mean=150
s=5
sample_size=25
```

In [40]:
```
t_cal=(x_bar - pop_mean) / (s/np.sqrt(sample_size))
t_cal
```

Out[40]:
```
np.float64(-2.0)
```

In [41]:
```
if t_cal > t:
    print("HA is right")
else:
    print("H0 is right")
```

```
H0 is right
```

Example 2

- A company wants to test whether there is a difference in productivity between two teams. They randomly select 20 employees from each team and record their productivity scores. The mean productivity score for Team A is 80 with a standard devition of 5, while the mean productivity score for Team B is 75 with a S.D of 6. Test at a 5% level of significance whether there is a differnce in productivity betweent two team A.
- H0 => PA-Pb = 0
- HA => PA - PB != 0

In [42]:
```
t_table_2= st.t.ppf(1-0.02, 38)
print(f"{t_table_2:.4f}")
```

```
2.1267
```

In [43]:
```
t_cal2= (80 -75) / (np.sqrt((25/20) + (36/20)))

print(f"{t_cal2:.4f}")
```

```
2.8630
```

In [44]:
```
if t_cal2 > t_table_2:
    print("HA is right. There is difference in productivity between Team A and Team
else:
    print("H0 is right.There is no difference.")
```

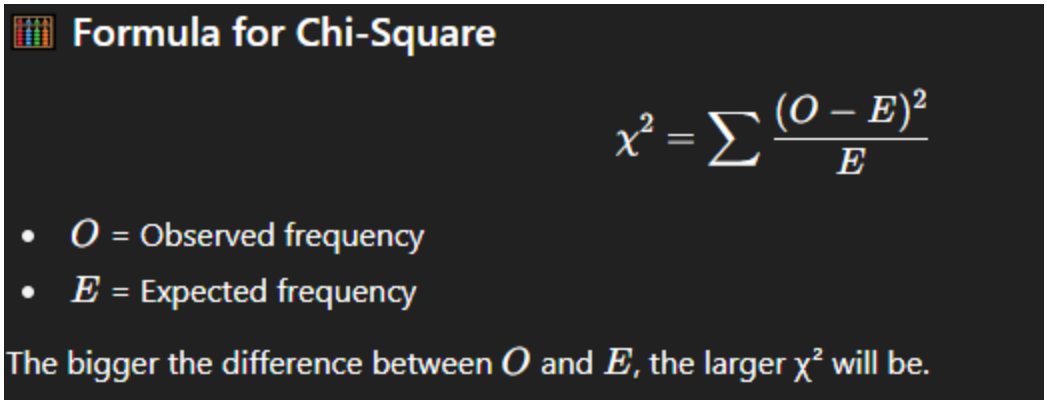
```
HA is right. There is difference in productivity between Team A and Team B.
```

Chi-Square Test

- The Chi-Square test (χ^2 test) is used to check if there's a significant relationship between categorical variables or if the observed frequencies differ from expected frequencies.

Example

A study was conducted to investigate whether there is a relationship between gender and preferred genre of music. A sample of 235 people were selectd, and the data collected is shown below. Test at a 5% level of significance whether there is a significant association between gender and music preference.



In [45]:
```
st.chi2.ppf(1-0.025,3)
```

Out[45]:
```
np.float64(9.348403604496148)
```

In [46]:
```
row_1=np.array([40,45,25,10])
row_2 = np.array([35,30,20,30])
```

In [48]:
```
sum_r_1=np.sum(row_1)
sum_r2=np.sum(row_2)
sum_row=np.array([sum_r_1,sum_r2])
sum_row
```

Out[48]:
```
array([120, 115])
```

In [49]:
```
sum_cal= row_1+row_2
sum_cal
```

Out[49]:
```
array([75, 75, 45, 40])
```

In [51]:
```
exp=[]
for i in sum_row:
    for j in sum_cal:
        value = (i * j)/235
        exp.append(value)
```

In [52]:
```
obj=np.array([40,45,25,10,35,30,20,30])
```

In [53]:
```
result= (np.sum(np.square(obj-exp) / exp))
print(f"{result:.4f}")
```

```
13.7887
```

H0 is correct.

In [ ]:
```

```

Linear Algebra

- Linear algebra isthe branch of mathematics concerned with the study of vectors, vector spaces, linear transformations, and systems of linear equations.
- It explores the properties of matrices, vectors, and linear functions, applying principles of algebra to understand geometric concepts like lines and plane.

1.) Key Concepts

- Scalar: A single number (e.g., temperature = 37°C).
- Vector: A 1D array of numbers (e.g., [1, 2, 3]) → represents direction & magnitude.
- Matrix: A 2D array of numbers (rows × columns).
- Tensor: Higher-dimensional generalization of matrices (e.g., images in deep learning).

Scalar = 5, Vector = [1,2,3], Matrix=[1,2][3,4]

In [7]:
```
import numpy as np

s = 5  # scalar
v = np.array([1, 2, 3])  # vector
M = np.array([[1, 2], [3, 4]])  # matrix
T = np.random.rand(3, 3, 3)  # tensor

print("Scalar:", s)
print("Vector:", v)
print("Matrix:\n", M)
print("Tensor shape:", T.shape)
```

```
Scalar: 5
Vector: [1 2 3]
Matrix:
 [[1 2]
 [3 4]]
Tensor shape: (3, 3, 3)
```

2.) Vector Operations

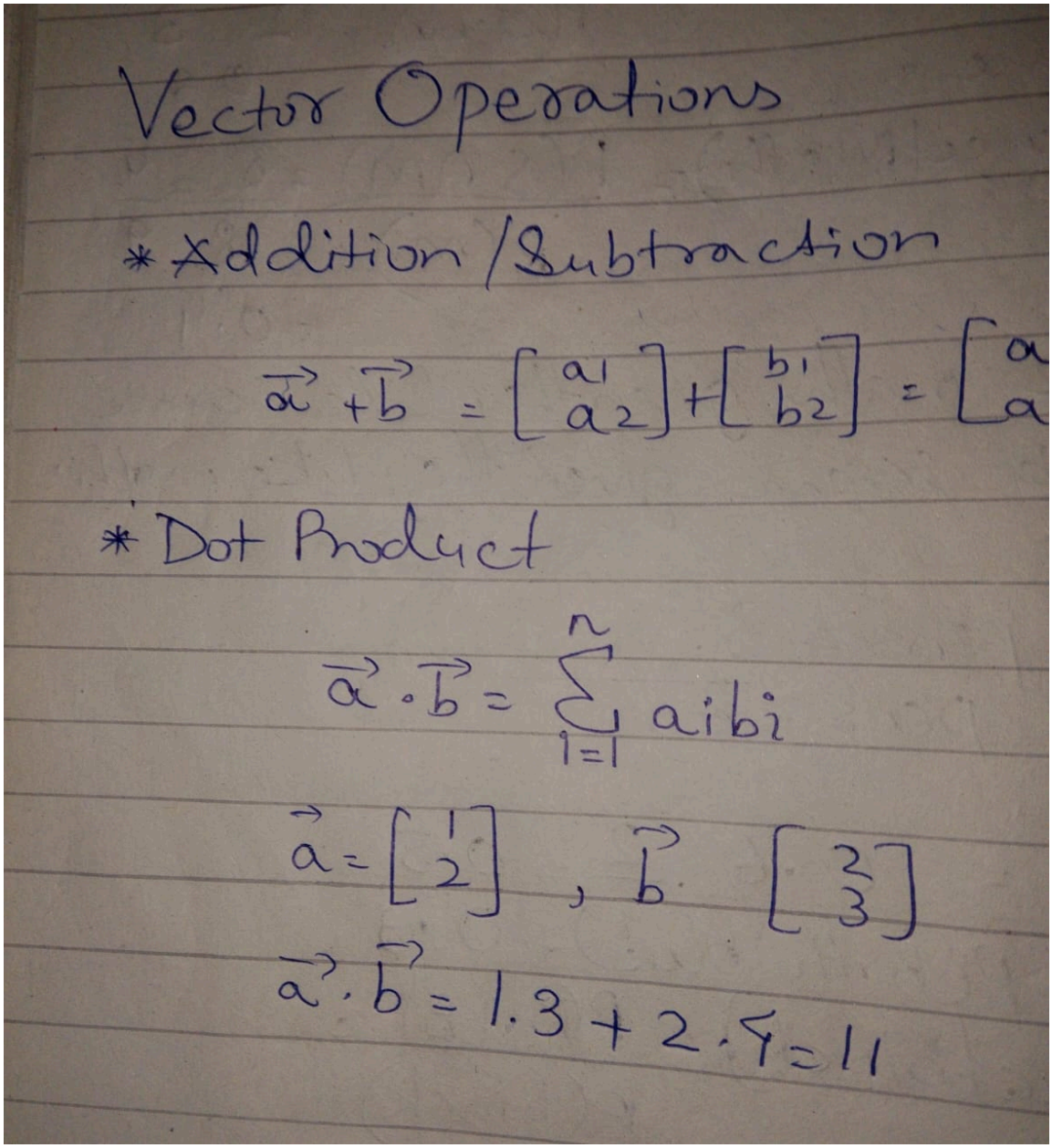
Vectors represent quantities with both direction and size (like force, velocity). Operations on vectors let us compare or combine them.

Addition:

Combine two vectors (head-to-tail rule).

Dot Product:

Measures similarity (cosine of angle between them). If dot product = 0 → vectors are orthogonal (independent).



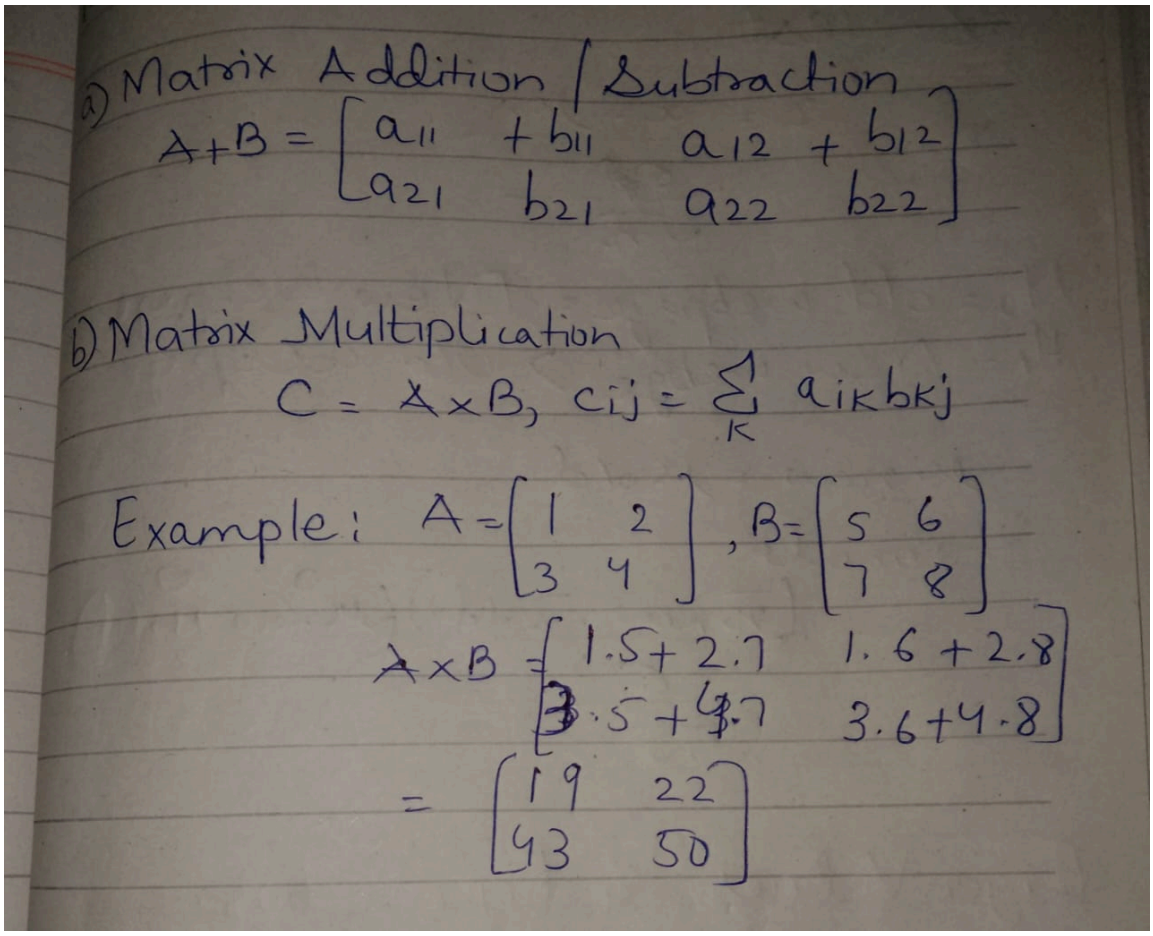
In [8]:
```
a = np.array([1, 2])
b = np.array([3, 4])

print("Addition:", a + b)
print("Dot Product:", np.dot(a, b))
```

```
Addition: [4 6]
Dot Product: 11
```

3.) Matrices Operations

Matrices are like containers of vectors.

- Addition/Subtraction = element-wise.
- Multiplication = composition of transformations (like rotating + scaling vectors). In data science, multiplication is used in linear regression (Xβ), neural networks (weights × inputs), etc.



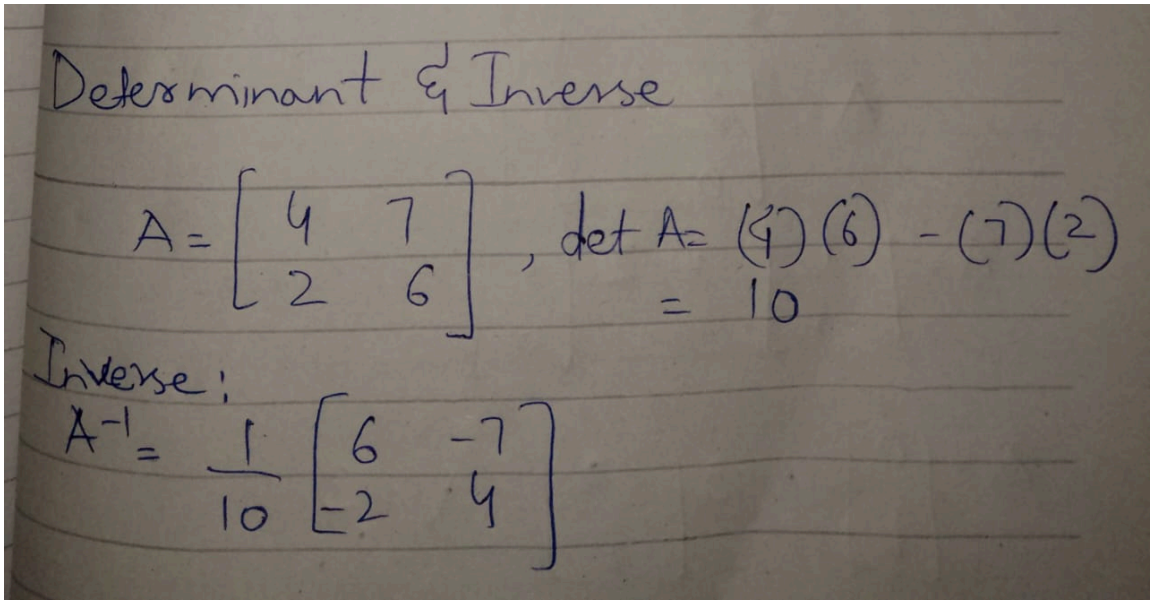
In [9]:
```
A = np.array([[1, 2], [3, 4]])
B = np.array([[5, 6], [7, 8]])

print("Matrix Multiplication:\n", np.dot(A, B))
```

```
Matrix Multiplication:
 [[19 22]
 [43 50]]
```

4) Determinant & Inverse

- Determinant measures how much a matrix scales space.
- det = 0 → matrix squashes space → not invertible.
- Inverse is like division for matrices.

- If A.A(inverse) = I, then A-1 undoes the transformation of A.



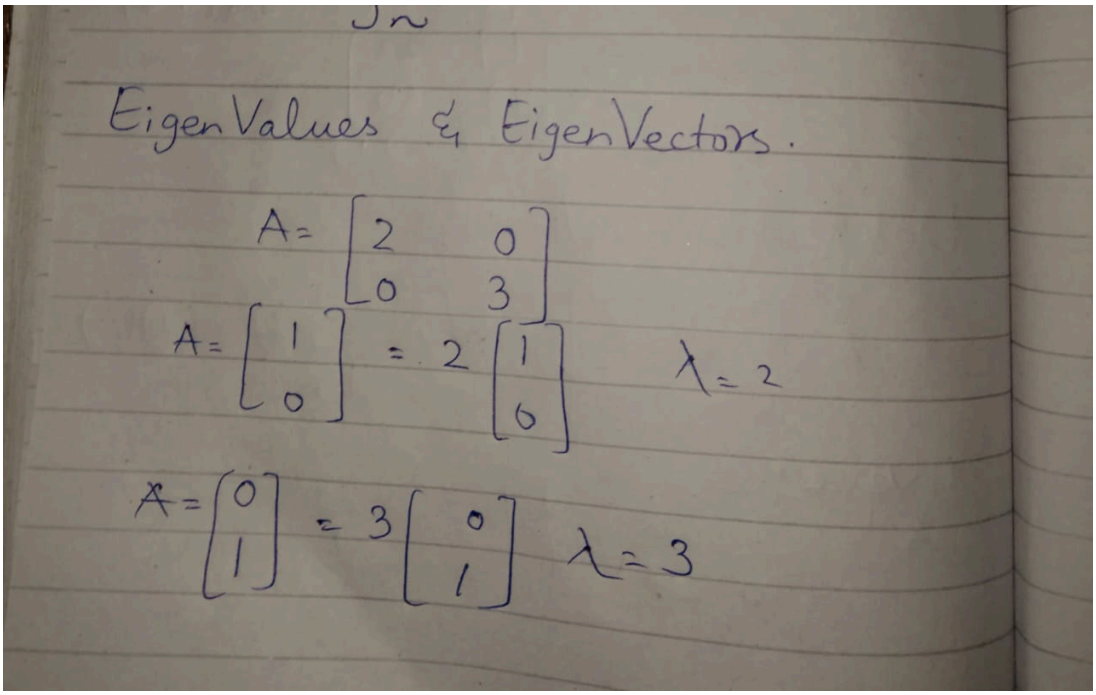
In [10]:
```
A = np.array([[4, 7], [2, 6]])
print("Determinant:", np.linalg.det(A))
print("Inverse:\n", np.linalg.inv(A))
```

```
Determinant: 10.000000000000002
Inverse:
 [[ 0.6 -0.7]
 [-0.2  0.4]]
```

5.) Eigenvalues & Eigenvectors

- Eigenvectors are directions that do not change direction when multiplied by a matrix, only their length changes.
- Eigenvalues tell us how much they are stretched.

- Used in PCA (Principal Component Analysis) to find directions of maximum variance.



In [11]:
```
A = np.array([[2, 0], [0, 3]])
eigvals, eigvecs = np.linalg.eig(A)

print("Eigenvalues:", eigvals)
print("Eigenvectors:\n", eigvecs)
```

```
Eigenvalues: [2. 3.]
Eigenvectors:
 [[1. 0.]
 [0. 1.]]
```

6.) Singular Value Decomposition (SVD)

SVD decomposes any matrix into 3 parts:

$$A = U\Sigma V^T$$

- U: directions of data in original space
- Σ: strength (variance) in each direction
- V^T: directions in feature space

👉 Used in **dimensionality reduction, noise filtering, recommendation systems (Netflix, YouTube).**

In [12]:
```
A = np.array([[3, 1], [1, 3]])
U, S, Vt = np.linalg.svd(A)

print("U:\n", U)
print("Singular values:", S)
print("V^T:\n", Vt)
```

```
U:
 [[-0.70710678 -0.70710678]
 [-0.70710678  0.70710678]]
Singular values: [4. 2.]
V^T:
 [[-0.70710678 -0.70710678]
 [-0.70710678  0.70710678]]
```

In [ ]: