Q1: Create a 1D array of 9 elements using numpy module and reshape it into 2D array of size 3*3

In [305…
```python
#importing certain libraries
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.preprocessing import StandardScaler, MinMaxScaler
import scipy.stats as stats
```

In [306…
```python
a=np.array([1,2,3,4,5,6,7,8,9])
print('Actual array is:',a)
#reshaping 1D array to 2D
b=a.reshape(3,3)
print('After reshaping:\n',b)
```

```
Actual array is: [1 2 3 4 5 6 7 8 9]
After reshaping:
 [[1 2 3]
 [4 5 6]
 [7 8 9]]
```

Q2: What will be the output: list3=[x for x in range(10) if x%2==0] print(list3)

In [307…
```python
list3=[x for x in range(10) if x%2==0]
print(list3)
#the range is from 0 to 9
#and x is even so x must be 0,2,4,6,8
```

```
[0, 2, 4, 6, 8]
```

Q3: Let we have a string text = "PythonProgramming"

Perform Slicing as:

a)Slice the string to obtain first 6 elements.

b)Extract the elements from index 6 to index 13

c)Extract last 5 characters of the string

d)Create a new string by sclicing and concatinating the first 4 and last 3 characters of the string

In [308…
```python
text="PythonProgramming"
#a) to obtain first 6 elements
#Note: indexing begins from 0 to n-1
t1=text[0:6]
print(t1)
#b)elements from index 6 to index 13
t2=text[6:14]
print(t2)
#c) last 5 characters
t3=text[-5:]
print(t3)
#d)sclicing and concatinating the first 4 and last 3 characters of the string
t4=text[0:4]+text[-3:]
print(t4)
```

```
Python
Programm
mming
Pything
```

Q4: Write a python programme to create two data frames namely d1,d2. Construct d1 utilizing a two dimensional list and create d2 using dictionary

In [309…
```python
lst=[['Raman',18,50000],['Aryan',20,45000],['Vinay',24,55000]]
dict={'Name':['Krish','Ayush','Sandy'],
      'Age':[22,25,28],
      'Salary':[60000,25000,35000]}
print(lst)
print(dict)
tp=['Name','Age','Salary']
d1=pd.DataFrame(lst, columns=tp)
print(d1)
print('\n')
d2=pd.DataFrame(dict)
print(d2)
```

```
[['Raman', 18, 50000], ['Aryan', 20, 45000], ['Vinay', 24, 55000]]
{'Name': ['Krish', 'Ayush', 'Sandy'], 'Age': [22, 25, 28], 'Salary': [60000, 25000, 35000]}
    Name  Age  Salary
0  Raman   18   50000
1  Aryan   20   45000
2  Vinay   24   55000


    Name  Age  Salary
0  Krish   22   60000
1  Ayush   25   25000
2  Sandy   28   35000
```

Q5: Write a python code to discuss in detail about the variance, standard deviation, covariance, correlation

In [310…
```python
#loding a csv file with data in it
data=pd.read_csv('rawdata.csv')
print(data)
print('\n')
v=np.var(data)
print('Variance in x,y:\n',v)
s=np.std(data)
print('Standard Deviation in x,y:\n',s)
c=np.cov(data)[0,1]
print('Covariance b/w x and y:',c)
d=np.corrcoef(data)[0,1]
print('Correlation b/w x and y:',d)
```

```
     x    y
0  2.5  2.4
1  0.5  0.7
2  2.2  2.9
3  1.9  2.2
4  3.1  3.0
5  2.3  2.7
6  2.0  1.6
7  1.0  1.1
8  1.5  1.6
9  1.1  0.9


Variance in x,y:
 x    0.5549
y    0.6449
dtype: float64
Standard Deviation in x,y:
 x    0.744916
y    0.803057
dtype: float64
Covariance b/w x and y: -0.010000000000000005
Correlation b/w x and y: -0.9999999999999998
```

```
/opt/anaconda3/lib/python3.11/site-packages/numpy/core/fromnumeric.py:3785: FutureWarning: The behavior of DataFrame.var with axis=None is deprecated, in a future version this will reduce over both axes and return a scalar. To retain the old behavior, pass axis=0 (or do not pass axis)
  return var(axis=axis, dtype=dtype, out=out, ddof=ddof, **kwargs)
/opt/anaconda3/lib/python3.11/site-packages/numpy/core/fromnumeric.py:3643: FutureWarning: The behavior of DataFrame.std with axis=None is deprecated, in a future version this will reduce over both axes and return a scalar. To retain the old behavior, pass axis=0 (or do not pass axis)
  return std(axis=axis, dtype=dtype, out=out, ddof=ddof, **kwargs)
```

Q6: Write a python programme to explain the concept of standardization and normalization. Discuss the circumstances under which it is appropiate to utilize these techniques in data processing

In [311…
```python
d = np.array([[1, 2],
              [3, 4],
              [5, 6]])
print(d)
# Standardization
data_standardized = StandardScaler().fit_transform(d)
print("Standardized data:\n",data_standardized)

# Normalization
data_normalized = MinMaxScaler().fit_transform(d)
print("Normalized data:\n",data_normalized)

#Standardization is appropriate when:
# *The features in the dataset have different units or scales.
# *The dataset contains outliers that may affect the performance of the model.
# *The algorithm being used assumes that the features are normally distributed.

#Normalization is appropriate when:
# *The algorithm being used relies on the magnitude of features rather than their distribution.
# *The features have different ranges and units.
# *The dataset does not contain significant outliers.
```

```
[[1 2]
 [3 4]
 [5 6]]
Standardized data:
 [[-1.22474487 -1.22474487]
 [ 0.          0.        ]
 [ 1.22474487  1.22474487]]
Normalized data:
 [[0.  0. ]
 [0.5 0.5]
 [1.  1. ]]
```

Q7: Write a Python Programme for reading the weather data from a file named "weather.csv" for 50 cities of a state in India. The file has following fields: City, Wind,Prescipitaion, Highest_Temperature, Lowest_Temperature Humidity, Atmospheric_pressure. Your programme should do following

a)Displays the 5 entries of "City"

b)Display the first 5 entries of dataframe

c)Displays the total null entries in "Precipitation" and "Humidity"

d)Displays all the rows by removing an instance of missing values

e)Find the dimentions of dataframe

f)Fill the missing values in "humidity" and "prescipition" with the mean of the columns.

In [312…
```python
#loading the weather csv file
d1=pd.read_csv('Weather.csv')
print(d1)
```

```
    Unnamed: 0 City_name Wind Speed Precipitation  Highest_temp  Lowest_temp  \
0          NaN     Jammu    10 km/h            6%            47            3
1          NaN   Kashmir    13 km/h           10%            39           -9
2          NaN  Udhampur    10 km/h            6%            45            2
3          NaN   Rajouri    11 km/h            8%            40           -3
4          NaN   Akhnoor    15 km/h            6%            44            4
5          NaN     Reasi    13 km/h           NaN            43            5
6          NaN   Pulwama    14 km/h            8%            36           -5
7          NaN   Gulmarg    15 km/h           12%            30           -7
8          NaN    Kargil    16 km/h            3%            35          -10
9          NaN       Leh    17 km/h            4%            38          -19
10         NaN  Baramula    18 km/h           NaN            29          -10
11         NaN   Shopian    19 km/h           10%            27           -4
12         NaN    Poonch    20 km/h           11%            25           -7
13         NaN     Samba    21 km/h            7%            45            4
14         NaN    Kathua    22 km/h           NaN            46            6
15         NaN      Siot    23 km/h            8%            40            3
16         NaN   Lamberi    24 km/h            8%            42            2
17         NaN  Nowshera    25 km/h            7%            39            4
18         NaN    Mendar    26 km/h            6%            40           -1

   Humidity Atmospheric_pressure
0       22%              1007hPa
1       38%              1007hPa
2       NaN              1007hPa
3       25%              1007hPa
4       19%              1007hPa
5       25%              1007hPa
6       NaN              1007hPa
7       25%              1007hPa
8       25%              1007hPa
9       25%              1007hPa
10      25%              1007hPa
11      NaN              1007hPa
12      25%              1007hPa
13      25%              1007hPa
14      25%              1007hPa
15      NaN              1007hPa
16      25%              1007hPa
17      25%              1007hPa
18      25%              1007hPa
```

In [314…
```python
#5 city names :
r=d1['City_name'][1:10:2]
print(r)
```

```
1    Kashmir
3    Rajouri
5      Reasi
7    Gulmarg
9        Leh
Name: City_name, dtype: object
```

In [315…
```python
#first 5 city names
s=d1.head(5)
print(s)
```

```
   Unnamed: 0 City_name Wind Speed Precipitation  Highest_temp  Lowest_temp  \
0         NaN     Jammu    10 km/h            6%            47            3
1         NaN   Kashmir    13 km/h           10%            39           -9
2         NaN  Udhampur    10 km/h            6%            45            2
3         NaN   Rajouri    11 km/h            8%            40           -3
4         NaN   Akhnoor    15 km/h            6%            44            4

  Humidity Atmospheric_pressure
0      22%              1007hPa
1      38%              1007hPa
2      NaN              1007hPa
3      25%              1007hPa
4      19%              1007hPa
```

In [316…
```python
#total null entries in precipitation and humidity
f=pd.isnull(d1['Precipitation']).sum()
g=pd.isnull(d1['Humidity']).sum()
print('Null values in Precipitation:',f)
print('Null values in Humidity:',g)
```

```
Null values in Precipitation: 3
Null values in Humidity: 4
```

In [317…
```python
d=d1.dropna(axis=1)
print(d)
```

```
   City_name Wind Speed  Highest_temp  Lowest_temp Atmospheric_pressure
0      Jammu    10 km/h            47            3              1007hPa
1    Kashmir    13 km/h            39           -9              1007hPa
2   Udhampur    10 km/h            45            2              1007hPa
3    Rajouri    11 km/h            40           -3              1007hPa
4    Akhnoor    15 km/h            44            4              1007hPa
5      Reasi    13 km/h            43            5              1007hPa
6    Pulwama    14 km/h            36           -5              1007hPa
7    Gulmarg    15 km/h            30           -7              1007hPa
8     Kargil    16 km/h            35          -10              1007hPa
9        Leh    17 km/h            38          -19              1007hPa
10  Baramula    18 km/h            29          -10              1007hPa
11   Shopian    19 km/h            27           -4              1007hPa
12    Poonch    20 km/h            25           -7              1007hPa
13     Samba    21 km/h            45            4              1007hPa
14    Kathua    22 km/h            46            6              1007hPa
15      Siot    23 km/h            40            3              1007hPa
16   Lamberi    24 km/h            42            2              1007hPa
17  Nowshera    25 km/h            39            4              1007hPa
18    Mendar    26 km/h            40           -1              1007hPa
```

In [318…
```python
# to find dimension of data frame
print(d1.shape)
```

```
(19, 8)
```

Q8: Explain Following with suitable example

a) Supervised learning and Unsupervised learning

b)Nominal and Ordinal Variable

c)Normalise the following data using min-max normalization by setting min=0, max=1:

1000,2000,3000,9000

Ans: a) In supervised learning, the algorithm learns from labeled data, meaning that each training example in the dataset is paired with an associated label or output. The goal of supervised learning is to learn a mapping from input variables (features) to output variables (labels) based on the labeled training data

Example: linear regression,logistic regression, decision trees

b)Nominal variables are categorical variables that represent categories or groups with no inherent order or ranking between them. In nominal variables the order does not matters.

Example: Colour, Gender, Type of fruits

Ordinal variables are categorical variables with distinct, ordered categories or groups. Unlike nominal variables, ordinal variables have a natural order or ranking between their categories, but the differences between the categories may not be uniformly meaningful. Here a hierchy is followed

Examples: Very Hot - Hot - Cold - Very Cold, Grades(10,9,8,7....)

c) To normalize the given data using min-max normalization with the range [0, 1], you can use the following formula: x(normalized) =(x - x(min))/ (x(max) - x(min))

For 1000

x=1000

x(min)=0

x(max)=1

So, x(normalized) i.e. 1000(normalized) = (1000 - 0)/(1 - 0) = 1000

For 2000

x=2000

x(min)=0

x(max)=1

So, x(normalized) i.e. 2000(normalized) = (1000 - 0)/(1 - 0) = 2000

For 3000

x=3000

x(min)=0

x(max)=1

So, x(normalized) i.e. 3000(normalized) = (3000 - 0)/(1 - 0) = 3000

For 9000

x=9000

x(min)=0

x(max)=1

So, x(normalized) i.e. 9000(normalized) = (9000 - 0)/(1 - 0) = 9000

Q9: Provide a detailed explanation of the PCA technique for dimensionality reduction including its methology and application

In [319…
```python
import numpy as np
import pandas as pd
from sklearn.decomposition import PCA
import matplotlib.pyplot as plt
from numpy.linalg import eig
```

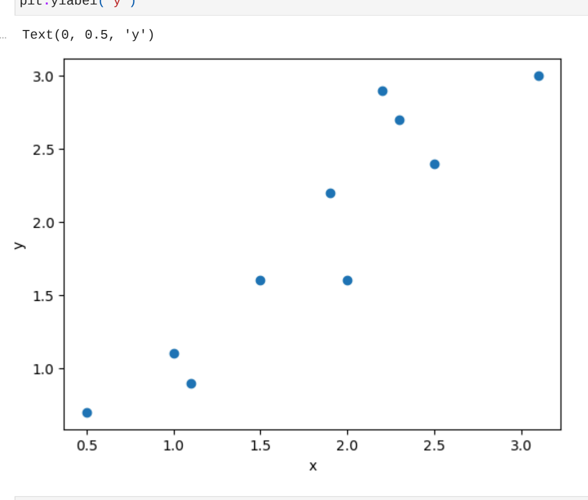
In [320…
```python
#Step 1: Imported a data sheet
data = pd.read_csv('rawdata.csv')
data
```

Out[320…

| | x | y |
|---|---|---|
| 0 | 2.5 | 2.4 |
| 1 | 0.5 | 0.7 |
| 2 | 2.2 | 2.9 |
| 3 | 1.9 | 2.2 |
| 4 | 3.1 | 3.0 |
| 5 | 2.3 | 2.7 |
| 6 | 2.0 | 1.6 |
| 7 | 1.0 | 1.1 |
| 8 | 1.5 | 1.6 |
| 9 | 1.1 | 0.9 |

In [321…
```python
#ploted the actal given data
plt.scatter(data["x"],data["y"])
plt.xlabel('x')
plt.ylabel('y')
```

Out[321… Text(0, 0.5, 'y')




In [322…
```python
#finding(mean centric)
mean = np.mean(data.T,axis=1)
print(mean)
```

```
x    1.81
y    1.91
dtype: float64
```

In [323…
```python
#finding standard deviation
std1 = np.std(data, axis=0)
print(std1)
```

```
x    0.744916
y    0.803057
dtype: float64
```

In [324…
```python
#step 2 i.e. Scaling data
scale_data = (data - mean)
scale_data
```

Out[324…

| | x | y |
|---|---|---|
| 0 | 0.69 | 0.49 |
| 1 | -1.31 | -1.21 |
| 2 | 0.39 | 0.99 |
| 3 | 0.09 | 0.29 |
| 4 | 1.29 | 1.09 |
| 5 | 0.49 | 0.79 |
| 6 | 0.19 | -0.31 |
| 7 | -0.81 | -0.81 |
| 8 | -0.31 | -0.31 |
| 9 | -0.71 | -1.01 |

In [325…
```python
#Step 3 i.e. finding covarience matrix
cov_matrix=np.cov(scale_data.T)
print(cov_matrix)
```

```
[[0.61655556 0.61544444]
 [0.61544444 0.71655556]]
```

In [326…
```python
#step 4 :finding eigen values and eigen vector
Eval,Evec=eig(cov_matrix)
print('eigen values are',Eval)
print('eigen vector are\n',Evec)
```

```
eigen values are [0.0490834  1.28402771]
eigen vector are
 [[-0.73517866 -0.6778734 ]
 [ 0.6778734  -0.73517866]]
```

In [327…
```python
#step 5: projecting data to new axis
project_data= Evec.T.dot(scale_data.T)
print(project_data.T)
```

```
[[-0.17511531 -0.82797019]
 [ 0.14285723  1.77758033]
 [ 0.38437499 -0.99219749]
 [ 0.13041721 -0.27421042]
 [-0.20949846 -1.67580142]
 [ 0.17528244 -0.9129491 ]
 [-0.3498247   0.09910944]
 [ 0.04641726  1.14457216]
 [ 0.01776463  0.43804614]
 [-0.16267529  1.22382056]]
```

In [328…
```python
pca=PCA(n_components=2)
pca.fit_transform(data)
```

Out[328…
```
array([[-0.82797019, -0.17511531],
       [ 1.77758033,  0.14285723],
       [-0.99219749,  0.38437499],
       [-0.27421042,  0.13041721],
       [-1.67580142, -0.20949846],
       [-0.9129491 ,  0.17528244],
       [ 0.09910944, -0.3498247 ],
       [ 1.14457216,  0.04641726],
       [ 0.43804614,  0.01776463],
       [ 1.22382056, -0.16267529]])
```

In [329…
```python
#variance ratio of each PCA i.e. which PCA gives more information
pca.explained_variance_ratio_
```

Out[329… array([0.96318131, 0.03681869])

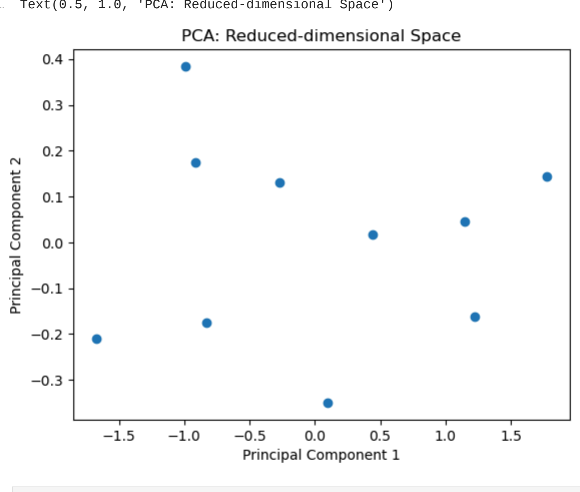
In [330…
```python
PCDF=pd.DataFrame(data=pca.fit_transform(data),columns=['PCA1','PCA2'])
PCDF
```

Out[330…

| | PCA1 | PCA2 |
|---|---|---|
| 0 | -0.827970 | -0.175115 |
| 1 | 1.777580 | 0.142857 |
| 2 | -0.992197 | 0.384375 |
| 3 | -0.274210 | 0.130417 |
| 4 | -1.675801 | -0.209498 |
| 5 | -0.912949 | 0.175282 |
| 6 | 0.099109 | -0.349825 |
| 7 | 1.144572 | 0.046417 |
| 8 | 0.438046 | 0.017765 |
| 9 | 1.223821 | -0.162675 |

In [331…
```python
#ploting pca data
plt.scatter(PCDF['PCA1'],PCDF['PCA2'])
plt.xlabel('Principal Component 1')
plt.ylabel('Principal Component 2')
plt.title('PCA: Reduced-dimensional Space')
```

Out[331… Text(0.5, 1.0, 'PCA: Reduced-dimensional Space')




In [332…
```python
#now lets see how much weight each variable has in principal components
loadings = pd.DataFrame(pca.components_.T,columns=['PCA1','PCA2'],index=["x","y"])
loadings
```

Out[332…

| | PCA1 | PCA2 |
|---|---|---|
| x | -0.677873 | -0.735179 |
| y | -0.735179 | 0.677873 |

Applications : Principal Component Analysis (PCA) is a dimensionality reduction technique commonly used in machine learning and data analysis. It's often applied to reduce the number of features in a dataset while retaining its essential information. Here are some practical examples of PCA:

1: Image Compression: PCA can be applied to reduce the dimensionality of image data while retaining most of its important features. This is useful in image compression to save storage space or bandwidth.

2: Face Recognition: In facial recognition systems, PCA can be used to reduce the dimensionality of facial features. This helps in focusing on the most important components for distinguishing faces, leading to more efficient and accurate recognition algorithms.

3: Speech Recognition: PCA can be applied to reduce the dimensionality of speech signals. This can improve the efficiency of speech recognition algorithms by focusing on the most relevant features for distinguishing different phonemes and words.

4: Quality Control in Manufacturing: PCA can be applied to analyze data from manufacturing processes to identify the most important factors contributing to product quality. This can help streamline production processes and improve overall product quality.

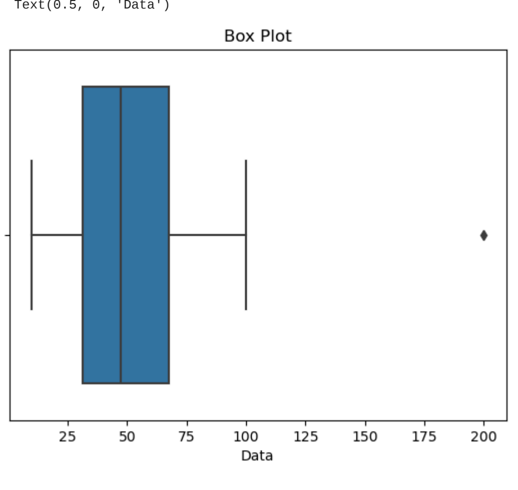
5: Climate Science: PCA can be used to analyze climate data, such as temperature and precipitation patterns. It helps identify the principal components of variability in climate data, aiding in understanding and modeling climate phenomena.

Q10: a)Write a python programme illustrating box plot. Explain how box plot aid in understanding outliers in data

b) Why is the normal distribution important in statistic, and how can it be visualized with a diagram? Define skewness and describe the characteristics of left and right skewed distributions

In [333…
```python
#a) creating a box plot
d = [10, 20, 25, 30, 35, 40, 45, 50, 55, 60, 70, 80, 100, 200]
# Creating a box plot
sns.boxplot(x=d) #for x axis
plt.title("Box Plot")
plt.xlabel("Data")
```

Out[333… Text(0.5, 0, 'Data')




Box plots provide a visual summary of the distribution of the data and help in identifying outliers. In a box plot, the central rectangle represents the interquartile range (IQR) of the data, with the median value marked as a line inside the rectangle. The "whiskers" extending from the box indicate variability outside the upper and lower quartiles, and any points beyond the whiskers are considered outliers. Outliers are identified as individual data points that fall outside the "whiskers" of the box plot. By visualizing the distribution of data using a box plot, it becomes easier to identify extreme values that may be erroneous or unusual compared to the rest of the data. Box plots provide a clear and concise summary of the data's central tendency, spread, and presence of outliers, making them valuable tools for exploratory data analysis and outlier detection.

In [ ]:

b) The normal distribution, also known as the Gaussian distribution or bell curve, is one of the most important concepts in statistics for several reasons:

Common Occurrence in Nature: Many natural phenomena follow a normal distribution. Examples include the heights of individuals in a population, IQ scores, measurement errors, and many more. Understanding the normal distribution allows statisticians to model and analyze various real-world processes accurately.

Central Limit Theorem: The normal distribution plays a crucial role in the Central Limit Theorem, which states that the distribution of the sample means approaches a normal distribution as the sample size increases, regardless of the shape of the population distribution. This theorem is fundamental in inferential statistics and hypothesis testing.

Statistical Inference: Many statistical methods, such as hypothesis testing, confidence intervals, and regression analysis, rely on assumptions of normality. When data approximate a normal distribution, these methods tend to be more reliable and powerful.

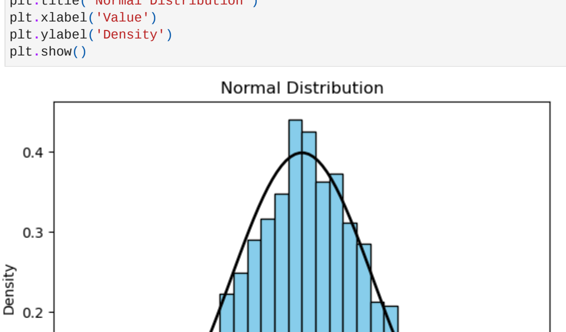
Simplicity and Symmetry: The normal distribution is characterized by its simplicity and symmetry, making it mathematically tractable and easier to work with than other distributions.

In [334…
```python
# Generate random data following a normal distribution
data = np.random.normal(loc=0, scale=1, size=1000)

# Plot a histogram
plt.hist(data, bins=30, density=True, color='skyblue', edgecolor='black')

# Plot the probability density function (PDF)
xmin, xmax = plt.xlim()
x = np.linspace(xmin, xmax, 100)
p = stats.norm.pdf(x, 0, 1)
plt.plot(x, p, 'k', linewidth=2)

plt.title('Normal Distribution')
plt.xlabel('Value')
plt.ylabel('Density')
plt.show()
```




Skewness: Skewness measures the asymmetry of the probability distribution of a real-valued random variable. It indicates whether the data is concentrated more on one side of the mean than the other.

Negative Skewness (Left Skew): The distribution has a longer left tail and is skewed towards the lower end of the range. The mean is less than the median, and the mode is greater than the median.

Positive Skewness (Right Skew): The distribution has a longer right tail and is skewed towards the higher end of the range. The mean is greater than the median, and the mode is less than the median.

Characteristics of Left and Right Skewed Distributions:

Left Skewed Distribution: Also known as negatively skewed distribution. The mean is less than the median. The tail of the distribution extends towards the left. The bulk of the data points are concentrated on the right side of the distribution. In a left skewed distribution, outliers tend to be on the left side of the histogram.

Right Skewed Distribution: Also known as positively skewed distribution. The mean is greater than the median. The tail of the distribution extends towards the right. The bulk of the data points are concentrated on the left side of the distribution. In a right skewed distribution, outliers tend to be on the right side of the histogram.

Q12: Explain the concept of hypothesis testing in statistical analysis ? Define the p-value in the context of hypothesis testing and explain its significance

Hypothesis testing is a fundamental concept in statistical analysis used to make decisions or inferences about a population based on sample data. It involves testing a hypothesis or claim about a population parameter, such as a population mean or proportion, using sample data.

The general process of hypothesis testing involves the following steps:

1)Formulate Hypotheses:

Null Hypothesis ( H 0 H 0): Represents the status quo or the hypothesis to be tested. It often states that there is no effect, no difference, or no association in the population.

Alternative Hypothesis ( H 1 H 1or H a H a): Represents the opposite of the null hypothesis. It typically states that there is an effect, difference, or association in the population.

2)Choose a Significance Level ( α α):

The significance level, often denoted by α α, represents the probability of rejecting the null hypothesis when it is actually true. Commonly used values for α α are 0.05 (5%) and 0.01 (1%).

3)Select a Test Statistic and Distribution:

Choose an appropriate test statistic based on the hypothesis being tested and the type of data. Determine the appropriate sampling distribution (e.g., normal distribution, t-distribution) for the test statistic.

4)Collect Data and Calculate the Test Statistic:

Collect sample data and calculate the value of the test statistic using the sample data.

5)Make a Decision:

Compare the calculated test statistic to the critical value(s) from the appropriate distribution. If the calculated test statistic falls in the rejection region (i.e., beyond the critical value(s)), reject the null hypothesis in favor of the alternative hypothesis. Otherwise, fail to reject the null hypothesis.

6)Interpret the Results:

Based on the decision made in the previous step, draw conclusions about the population parameter being tested.

The p-value is a key concept in hypothesis testing and represents the probability of obtaining the observed results (or more extreme results) if the null hypothesis is true. In other words, it quantifies the strength of the evidence against the null hypothesis. Here's how it works:

**If the p-value is less than or equal to the significance level (α), typically denoted as p ≤ α p≤α, then we reject the null hypothesis. This suggests that the observed results are statistically significant, and we have sufficient evidence to support the alternative hypothesis.

**If the p-value is greater than the significance level (p>α), then we fail to reject the null hypothesis. This indicates that the observed results are not statistically significant, and we do not have sufficient evidence to reject the null hypothesis.

In [ ]: