Date :

Aim: Basic characteristics and operation of array using numpy.

Theory: NumPy is a powerful library in Python for numerical computations, especially when working with arrays. It provides an efficient way to perform operations on large datasets.

* Characteristics of NumyPy Arrays

a) Homogeneous Data Type
- Unlike Python lists, NumPy arrays store elements of the same data type, which makes operations faster.

b) Fixed Size
- Once a NumPy array is created, its size is fixed. However, you can create a new array if resizing is required.

c) Efficient Memory Usage
- NumPy arrays consume less memory compared to Python lists due to their compact memory storage.

d) Support for vectorized Operations
- NumPy allows element-wise operation without using loops, making computations significantly faster.

e) Multi-dimensional Support

Date :

- Numpy arrays support 1D, 2D, and multi-dimensional arrays.

Program :

```
import numpy as np

# Creating arrays
array 1 = np.array([1,2,3,4,5])
array 2 = np.array([[1,2,3],[4,5,6]])

# Basic operations

print("Array1+2:", array1 + 2)

# Universal functions

print("square root of Array1:", np.sqrt(array1))
print("Exponential of Array2:", np.exp(array1))

# slicing

print("First two elements of Array1:", array1[:2]
print("Last three elements of Array1:", array1[-3:])
                                          [:]

# Reshaping

array3 = np.arange(9).reshape(3,3)
print("Reshaped array3:\n", array3)
```

Date :

indexing
print ("Element at (0,1) in array3:",
array3[0,1])

Aggregation
print("sum of all elements in Array1:",
np.sum(array1))
print("Mean value in Array1:", np.max(array1))

# Transposing
print("Transposed array2:\n", np.transpose(array2))

Output:
Array1 + 2 : [3 4 5 6 7]

Square root of Array1 : [1 1.414 1.732 2 2.236]
Exponential of Array1 : [2.718 7.389 20.085 54.598
148.413]
Reshaped array3 : [[0 1 2]
[3 4 5]
[6 7 8]]
Element at (0,1) in array3 : 1
sumof all elements in Array1 : 15
Mean value in Array1 : 1
Transposed array2 : [[1 4]
[2 5]
[3 6]]

Result : Hence a program executed Successfully.

Date :

Aim: To create a dataframe using list of elements (using pandas dataframe).

Theory: A DataFrame is one of the commonly used data structures in Pandas. It allows for efficient data storage, manipulation and analysis. DataFrames can be created using various data sources, including lists, dictionaries, NumPy arrays, csv files, and SQL databases.

* Characteristics of a Pandas DataFrame
  - Tabular Structure: Organizes data in a tabular format with labeled rows and columns.
  - Heterogeneous Data: can store different types of data (integers, floats, strings, etc).
  - Indexing: Supports both row and column indexing.
  - Size Mutable: Allows insertion and deletion of rows and columns.
  - Data Manipulation: Supports operations like filtering sorting, grouping, and aggregation.

Program:

```
import pandas as pd

data = {
    'Name': ['Alice', 'Bob', 'Charlie', 'David',
             'Eva'],
```

Date :

```
'Age': [24, 30, 18, 35, 28],
'City': ['New York', 'Los Angeles', 'Chicago',
         'Houston', 'Phoenix']
}
df = pd.DataFrame(data)
print(df)
```

Output :

	Name	Age	City
0	Alice	24	New York
1	Bob	30	Los Angeles
2	Charlie	18	Chicago
3	David	35	Houston
4	Eva	28	Phoenix

Result : Hence a Program executed Successfully.

(A) (H)

Date :

Aim : Write a program to calculate the estimate of location
a. Mean b. median c. mode
d. weighted mean

Theory :
* Estimate of location
The estimate of location refers to a single value that represents the central tendency of a dataset. Various measures such as Mean, Median, Trimmed mean, and weighted mean help summarize data effectively. These measures provide insights into the dataset's general behaviour and distribution.

1. Mean (np.mean(data)) → Sum of all values divided by the number of values.
2. Median (np.median(data)) → Middle value when data is sorted
3. Mode (stats.mode(data)) → Most frequent value in the dataset.
4. Trimmed Mean (stats.trim_mean(data, proportiontocut = 0.1)) → Mean after removing 10% of the smallest and largest values.
5. Weighted Mean (np.average(data, weights = weights)) → Mean where values have different weights.

Program :

Date :

```
import pandas as pd
import numpy as np
from scipy import stats from scipy import stats

np.random.seed(42)
data = np.random.randn(1000,4)
df = pd.DataFrame(data, columns = ['A', 'B', 'C', 'D'])

mean = df.mean()
print("\n Mean of each column:")
print(mean)

median = df.median()
print("\nMedian of each column:")
print(median)

mode = df.apply(lambda x: np.percentile(x, [25,
    50, 75]))
print("\n Summary statistics of the Dataframe:")
print(trimmed

mode = df.apply(lambda x: stats.mode(x)[0])
print("\nMode of each column:")
print(mode)

weights = np.random.rand(1000)
weighted_mean = df.apply(lambda x: np.average
    (x, weights = weights))
print("\nWeighted mean of each column:")
print(weighted_mean)
```

Date :

Output :

Mean of each column :

A	0.030624
B	0.024828
C	-0.008255
D	0.030086

dtype : float64

Median of each column :

A	0.056187
B	0.020210
C	-0.007509
D	0.021158

dtype : float64

Mode of each column

	A	B	C	D
0	-3.0195	-2.8962	-3.24126	-2.9911

Trimmed mean (10%) of each column :

A	0.03234
B	0.01628
C	-0.010729
D	0.0277

Weighted mean of each column :

A	-0.00976
B	0.05410
C	-0.004505
D	0.0154

dtype : float64

Result : Hence a program executed Successfully.

Date

Aim : Write a program to calculate the estimate of variability
a. Variance b. Standard deviation

Theory:

* Estimating Variability.

- Variability (or dispersion) measures how spread out data points are in a dataset. It helps us understand the degree of uncertainity and fluctuation in the data.

1. Variance

- Variance quantifies the degree to which data points deviate from the mean. It measures the average squared difference between each data point and the mean of do the dataset

Formula

$$s^2 = \frac{\sum (x_i - \bar{x})^2}{N-1}$$

x_i = each data point
$\bar{x}$ = Sample mean
N = Total number of observation

2. Standard Deviation

- Standard deviation is the square root of variance. It provides a measure of dispersion in the same units as the original data, making it easier to interpret.

Date :

$$\text{formula} = \sqrt{s^2} = \sqrt{\frac{\Sigma(x_i - \bar{x})^2}{N-1}}$$

Program :

```
import pandas as pd

df = pd.read_csv('car_sales.csv')

numerical_columns = df.select_dtype(include=['number'
                    ]).columns

if numerical_columns.empty:
    print("No numerical columns found in the
          csv file.")
else:
    column_name = numerical_columns[0]

    variance = df[column_name].var()
    std_dev = df[column_name].std()

    print(f"Column: {column_name}")
    print(f"variance: {variabtence:.4f}")
    print(7"Standard Deviation: {std_dev:.4f}")
```

Output : Column : Sales in thousands
Variance : 4628.00225
Standard Deviationtion : 68.0294

Result : Hence a Program executed Successfully.

Page No.

Date :

Aim : Write a Program to find coefficient correlation.

Theory : The coefficient of correlation is a statistical measure that quantifies the strength and direction of a linear relationship between two variables. It helps determine how changes in one variable are associated with changes in another.

Program :

```
import math numpy as np

x = [20, 25, 30, 40, 50, 60]
y = [60, 60, 70, 73, 64, 75]

correlation_coefficient = np.corrcoef (x, y) [0, 1]

print ("correlation Coefficient :", correlation_coefficient)
```

output :

```
Correlation Coefficient : 0.7742
```

Result : Hence a Program executed successfully.

Page No.

Aim: Write a program for data distribution using histogram and boxplot.

Theory:

- Data distribution refers to how data points are spread across different values in a dataset. Two of the most effective visualizations for understanding distribution are histograms and boxplots.

* Histogram

- A histogram is a type of bar chart that represents the frequency distribution of a dataset. It divides the data into bins (intervals) and counts how many values fall into each bin.

* Boxplot (Box-and-Whisker-Plot)

- A boxplot is a graphical representation of the five-number summary of a dataset.

- The presence of outliers (shown as 'individual points outside the whiskers).
- Skewness (if the median is not centered in the box).
- The overall spread and variability of the dataset.

Date :

Program :

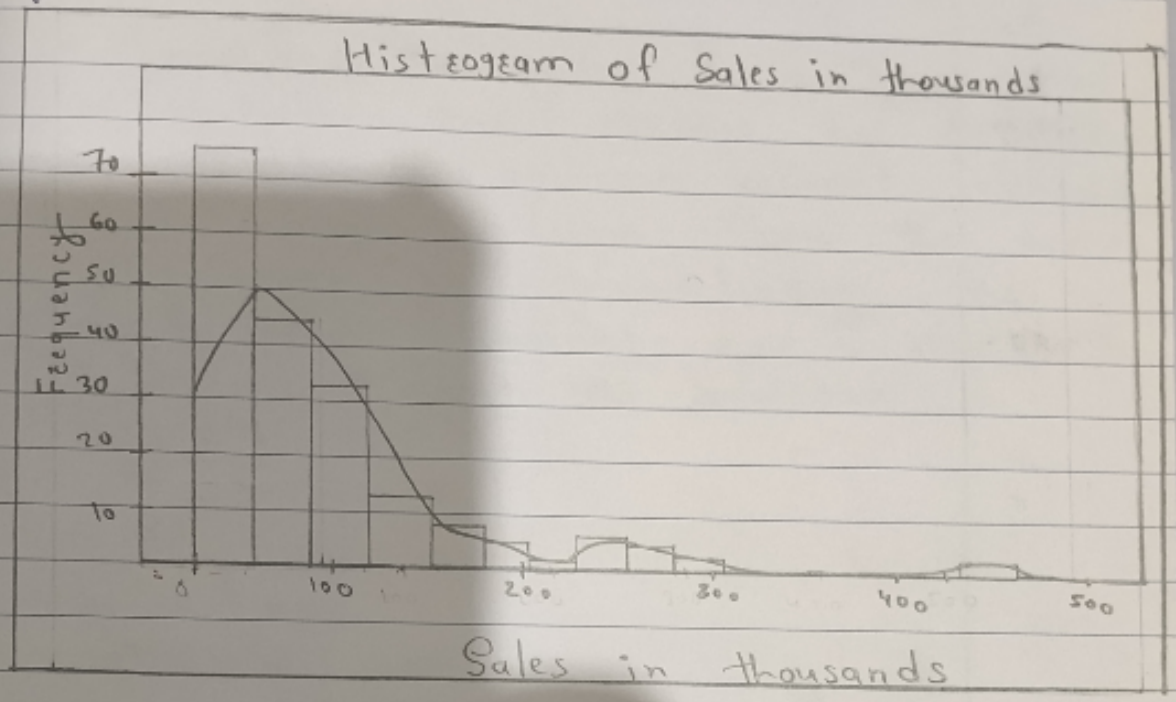
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

df = pd.read_csv('car_sales.csv')

numerical_columns = df.select_dtypes(include =
                    ['number']).columns

if numerical_columns.empty:
    print("No numerical columns found in the
          csv file.")
else:
    column_name = numerical_columns[0]

    fig, axes = plt.subplots(1,2, figsize =(12,5))

    sns.histplot(df[column_name], bins = 20, kde=True
        , ax = axes[0], color = 'blue')
    axes[0].set_title(f'Histogram of {column_
                      name}')
    axes[0].set_xlabel(column_name)
    axes[0].set_ylabel('Frequency')

    sns.boxplot(x=df[column_name], bins = 20,
                ax = axes[1], color = 'red')
    axes[1].set_title(f'Boxplot of {column_name}')
    axes[1].set_xlabel(column_name)
```

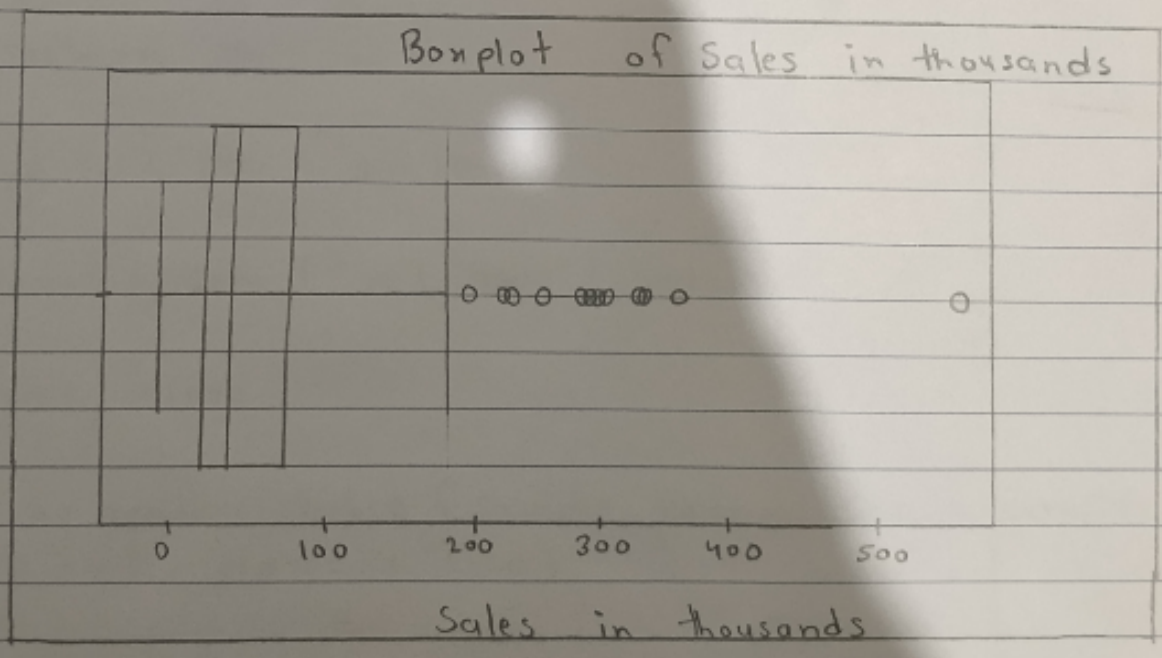
Page No.

Date :

```
plt.tight_layout()
plt.show()
```

Output :





Result : Hence a Program executed successfully.

Page No.

Date :

Aim : Implement Sampling Distribution using Python.

Theory :

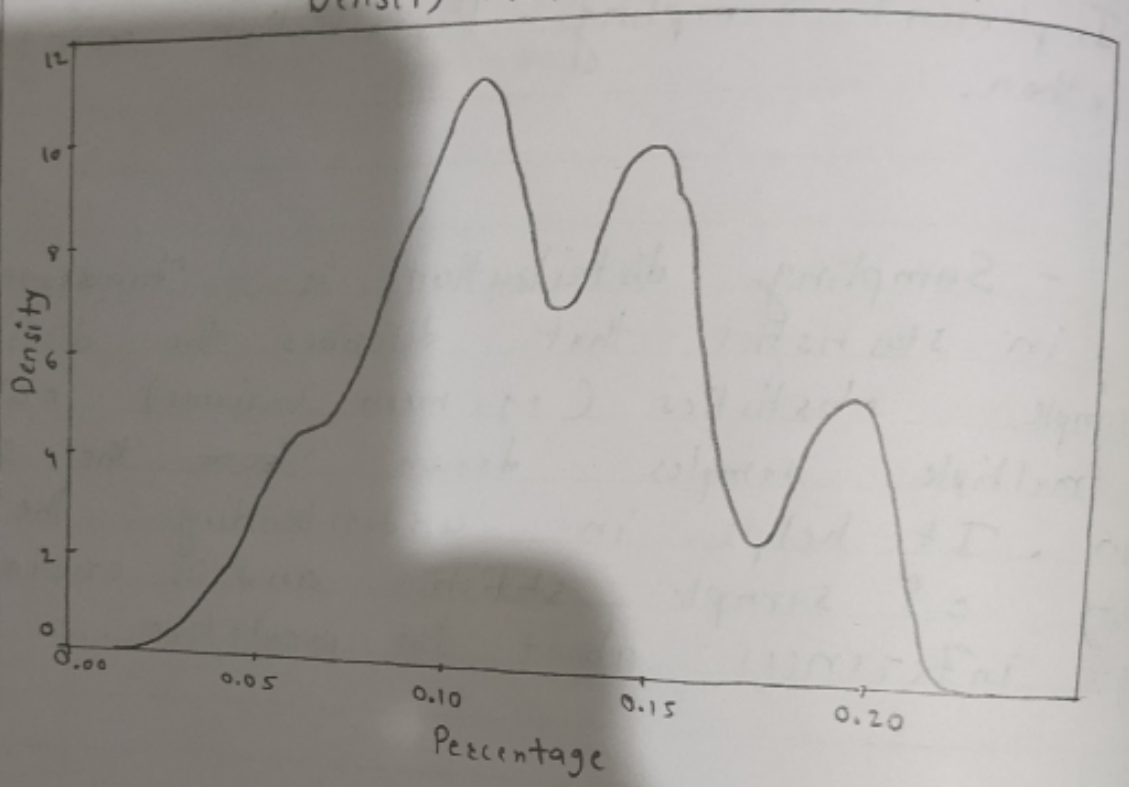
- Sampling distribution is a fundamental concept in statistics that describes the distribution of sample statistics (e.g. mean, variance) obtained from multiple samples drawn from the same population. It helps in understanding the variability of sample statistics and is crucial for making inferences about the population.

Program :

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as ~~snt~~ pd
import seaborn as sns

data = pd.read_csv ('health.csv')

sample_data = data ['pct_2014'].values

sample_size = 1000

samples = np.random.choice (sample_data, size = sample_
          size, replace = True)

plt.figure (figsize = (10,6))
sns.kdeplot (samples, color = 'red', fill = True)
plt.title ('Density Plot of Sample Data')
```

Page No.

Density Plot of Sample Data



Result: Hence a Program executed Successfully.

Date :

```
plt.xlabel ('Percentage')
plt.ylabel ('Density')
plt.grid (axis = 'y', alpha = 0.75)
plt.show()

mean = np.mean (samples)
std_dev = np.std (samples)

print (f'Mean of Sample Distribution : {mean}')
print (f'Standard Deviation of Sample Distribution :
      {std_dev}')
```

Output :

Mean of Sample Distribution : 0.11217
Standard Deviation of Sample Distribution : 0.0397

Result : Hence a Program executed successfully.

Page No.