Page No. _____
Date 28 08 23

PRACTICAL - 4

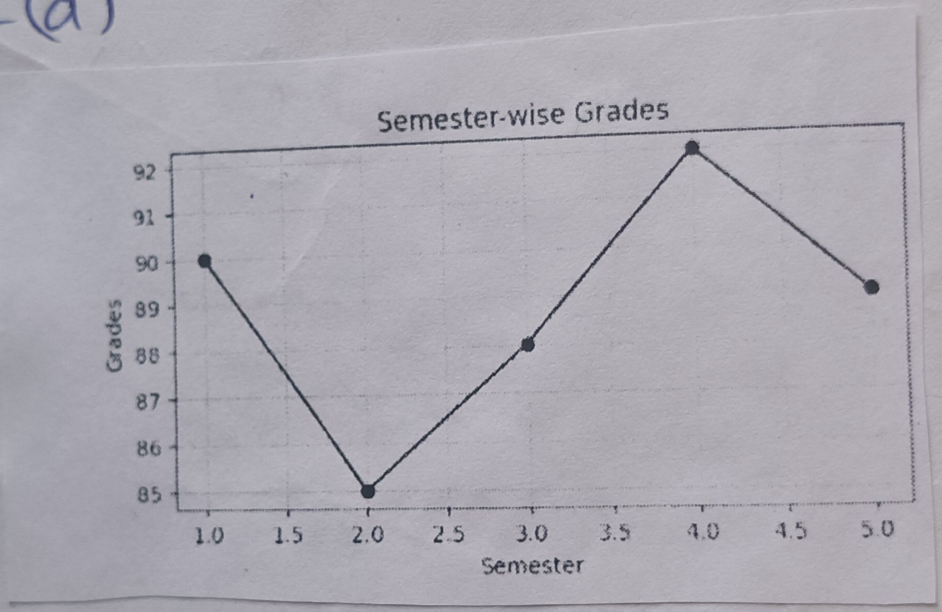
AIM:- Create the following plots using matplotlib pyplot.

(a) line graph for semester-wise grades.

```
import matplotlib.pyplot as plt
import numpy as np

semesters = [1,2,3,4,5]
grades = [90,85,88,92,89]
programming_languages = ['Python', 'Java', 'C++', 'JavaScrip
competencies = [90,75,80,95]
percentages = np.array([75,60,70,80,65])
colors = ['blue', 'green', 'red', 'purple', 'orange']

plt.plot(semesters, grades, marker='o', color='b')
plt.title('Semester-Wise Grades')
plt.xlabel('Semester')
plt.ylabel('Grades')
plt.grid(True)
plt.show()
```

(b) Pie-chart depicting your core competency in each of the following language learnt. Explode the wedges from the centre for each of the language. Add labels to the wedges & a legend.

Output (a)


Semester-wise Grades
Grades
92
91
90
89
88
87
86
85
1.0
1.5
2.0
2.5
3.0
3.5
4.0
4.5
5.0
Semester

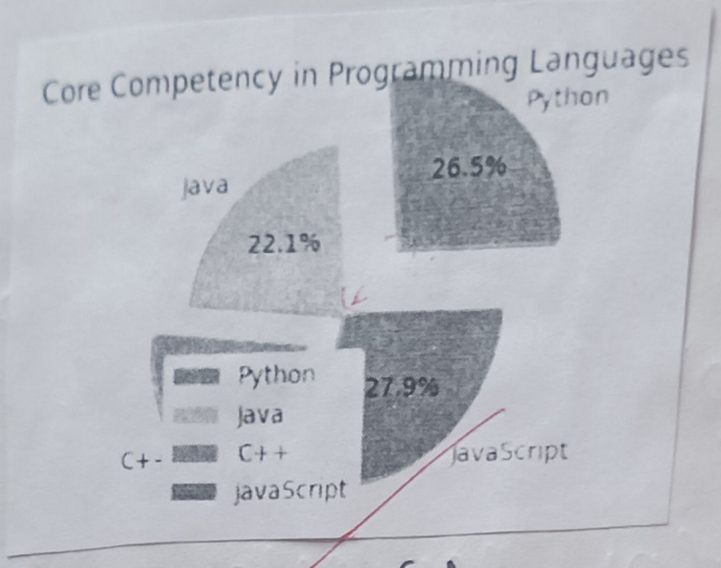
```
explode = (0.5, 0, 0.3, 0)
plt.pie(competencies, labels = programming_languages,
        explode = explode, autopct = '%.1.1f%%%')
plt.title('Core Competency in Programming Language')
plt.legend(programming_languages, loc = 3)
plt.show().
```

(c) Scatter plot depicting your %'s till Sem 5, use diff. colors for different semesters.

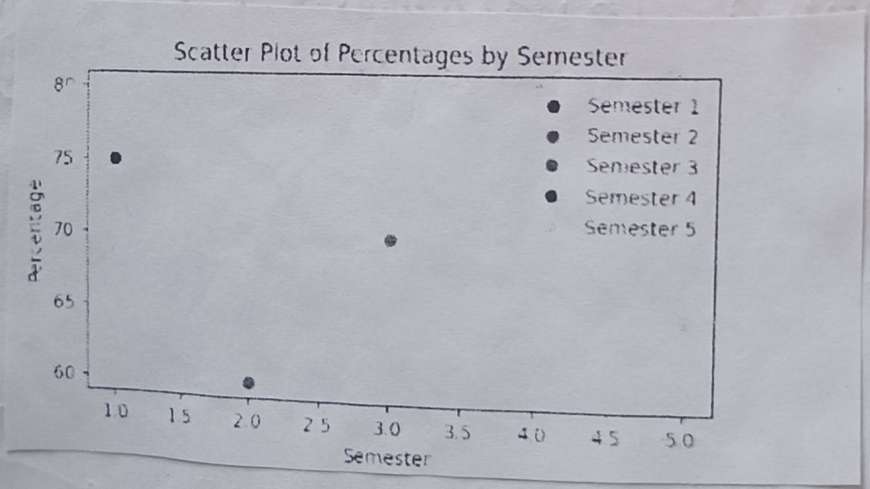
```
plt.figure(figsize=(6,3))
for semester, color in zip(semesters, colors):
    plt.scatter(semesters, percentages[semester - 1],
                label = f'Semester {semester}', color = color)
plt.title('Scatter plot of Percentages by Semester')
plt.xlabel('Semester')
plt.ylabel('Percentage')
plt.legend()
plt.show().
```

(d) Bar chart depicting the % upto 75. Use height & width attribute, different colors for vertical & horizontal Barcharts.

```
plt.figsize
plt.figure(figsize = (6,3))
# Horizontal Bar chart.
plt.subplot(121)
plt.barh(semesters, percentages, color='skyblue',
         height = 0.6)
```



Output (b)



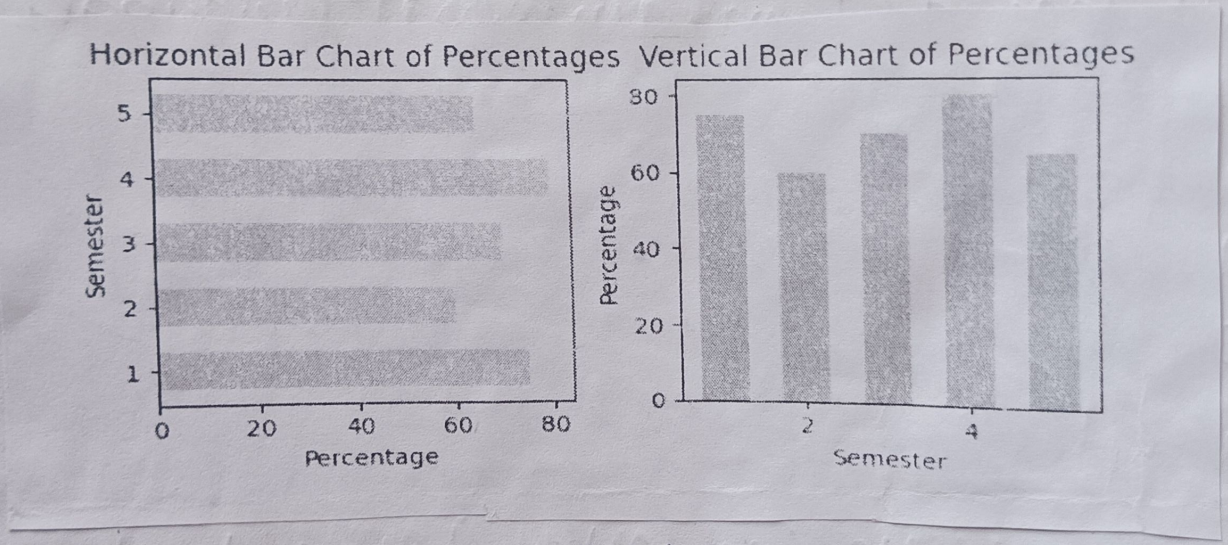
Output (c)

```
plt.title('Horizontal Bar Chart of Percentages')
plt.xlabel('Percentage')
plt.ylabel('Semester')
# Vertical bar chart.
plt.subplot(122)
plt.bar(semesters, percentages, color='lightcoral',
        width=0.6)
plt.title('Vertical Bar Chart of Percentages')
plt.xlabel('Semester')
plt.ylabel('Percentage')
plt.tight_layout()
plt.show().
```

(e) Create Subplots of 3×2 dimensions.

```
fig, axes = plt.subplots(3,2, figsize=(12,10))
fig.suptitle('Subplots (3×2)')
```

```
import matplotlib.pyplot as plt.
import numpy as np

x = np.linspace(0, 2*np.pi, 100)
y1 = np.sin(x)
y2 = np.cos(x)
y3 = np.tan(x)
y4 = np.exp(x)
y5 = np.log(x+1)
y6 = np.sqrt(x)
```

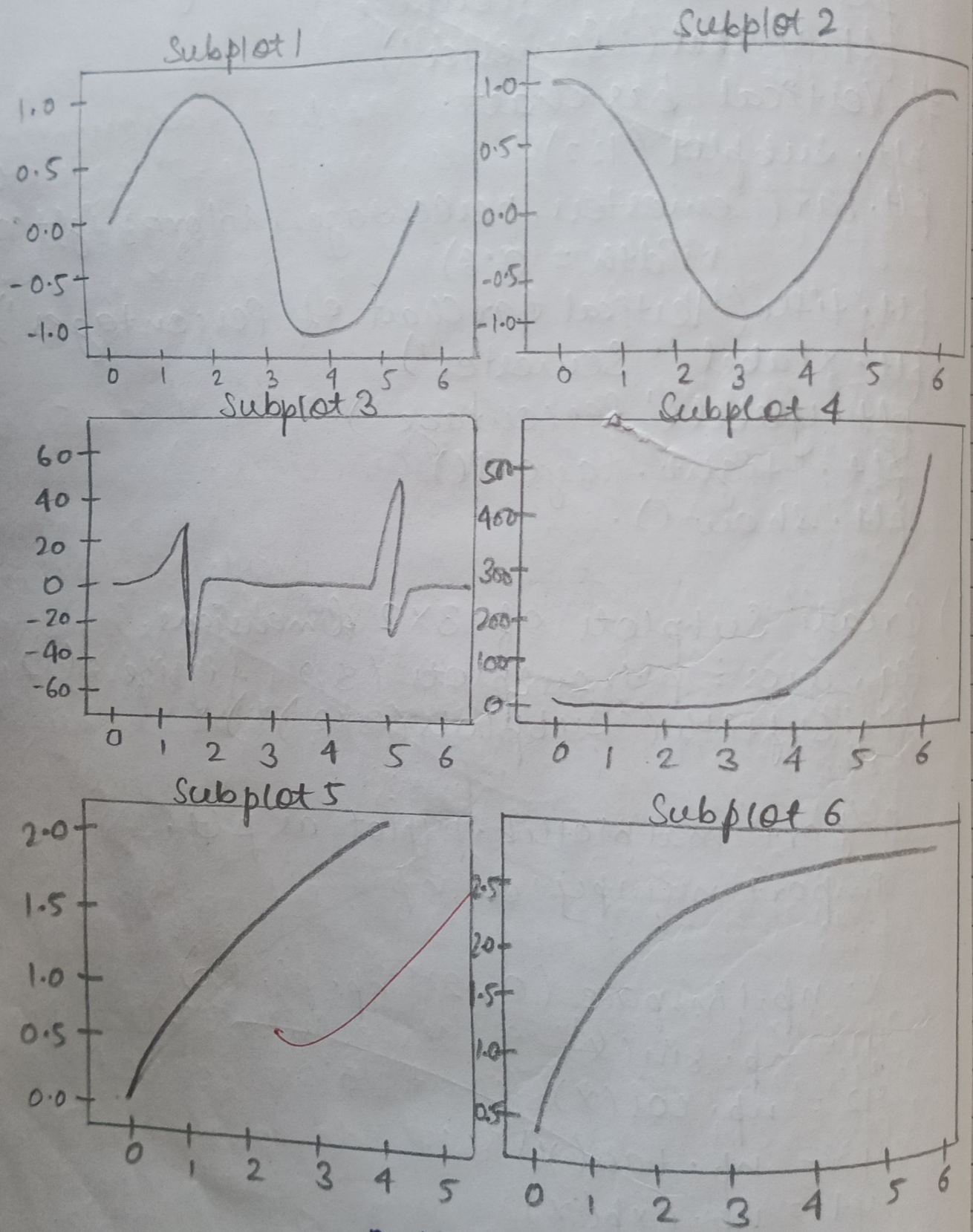

Horizontal Bar Chart of Percentages
Semester
5
4
3
2
1
0
20
40
60
80
Percentage
Vertical Bar Chart of Percentages
Percentage
80
60
40
20
0
2
4
Semester

Output (d)

Page No. _____
Date

OBSERVATION:-

Here, we have learnt different visualizations through Matplotlib. like lineplot, pie chart, subplot, etc. Matplotlib is one of the most popular & oldest plotting libraries in python which is used in Machine learning. In Machine learning, it helps to understand the huge amount of data through different visualizations. Eg. Boxplot, Histogram, Scatterplot, etc.


Subplot 1
1.0
0.5
0.0
-0.5
-1.0
0 1 2 3 4 5 6
Subplot 2
1.0
0.5
0.0
-0.5
-1.0
0 1 2 3 4 5 6
Subplot 3
60
40
20
0
-20
-40
-60
0 1 2 3 4 5 6
Subplot 4
500
400
300
200
100
0
0 1 2 3 4 5 6
Subplot 5
2.0
1.5
1.0
0.5
0.0
0 1 2 3 4 5
Subplot 6
2.5
2.0
1.5
1.0
0.5
0 1 2 3 4 5 6

Output (c)

```
plt.figure(figsize=(10, 8))

plt.subplot(3, 2, 1)
plt.plot(x, y1)
plt.title('Subplot 1')

plt.subplot(3, 2, 2)
plt.plot(x, y2)
plt.title('Subplot 2')

plt.subplot(3, 2, 3)
plt.plot(x, y3)
plt.title('Subplot 3')

plt.subplot(3, 2, 4)
plt.plot(x, y4)
plt.title('Subplot 4')

plt.subplot(3, 2, 5)
plt.plot(x, y5)
plt.title('Subplot 5')

plt.subplot(3, 2, 6)
plt.plot(x, y6)
plt.title('Subplot 6')

plt.tight_layout()  # Add some space b/w
plt.show()          # subplots.
```

Page No.
Date 28 08 23

PRACTICAL-5

AIM:- Working with a Seaborn Library to -

(a) Create a distribution plot with histogram.

```
import matplotlib.pyplot as plt.
import seaborn as sns.
from numpy import random.
import pandas as pd.

sns.distplot([0,1,2,3,4,5])
plt.show()
```

(b) Create a distribution plot without histogram.

```
sns.distplot([0,1,2,3,4,5], hist=False)
plt.show().
```

(c) Create a Normal distribution plot.

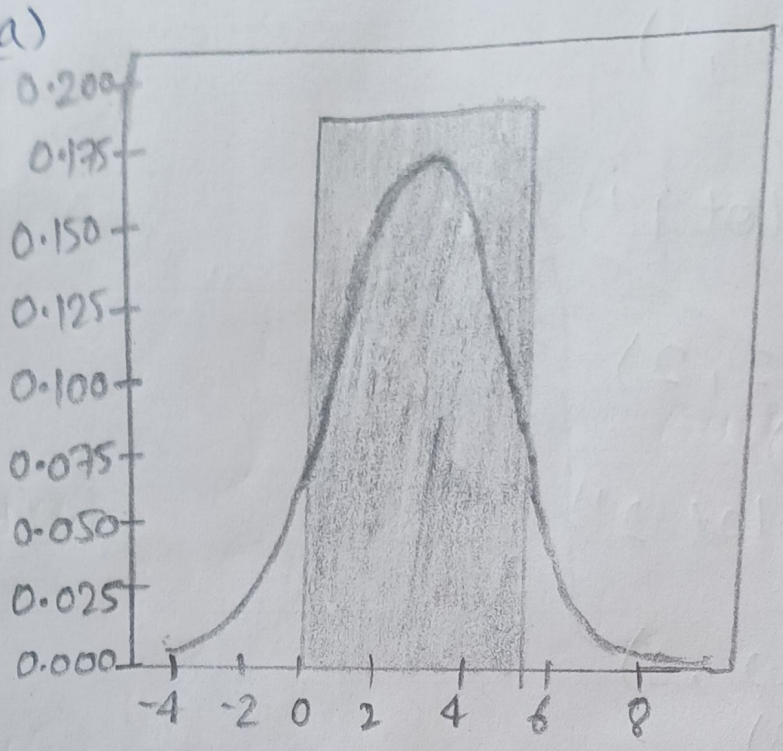
```
sns.distplot(random.normal(loc=50, scale=5,
size=1000), label='normal', hist=False)
plt.show().
```

(d) Create a Binomial distribution plot & add the corresponding label.

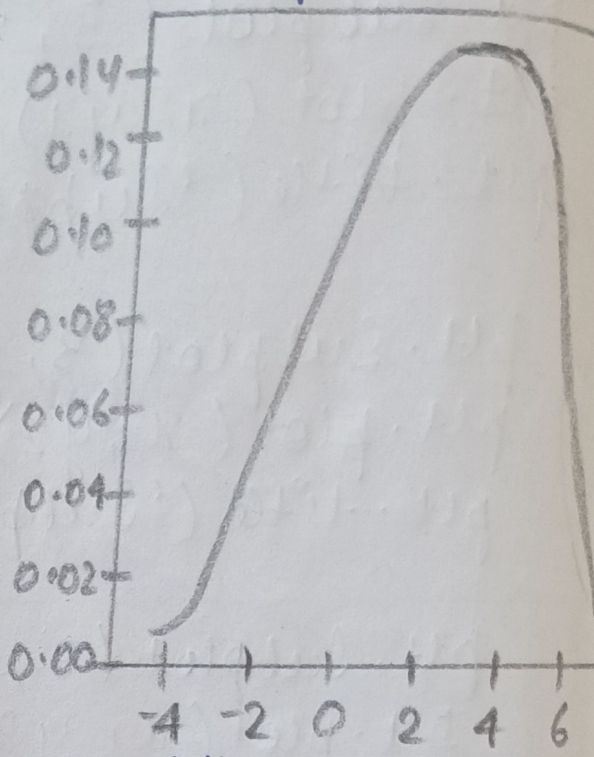
```
sns.distplot(random.binomial(n=10, p=0.5, size=1000),
label='Binomial', hist=False)
plt.show().
```

(e) Use a csv file to obtain lineplot b/w 2 parameters

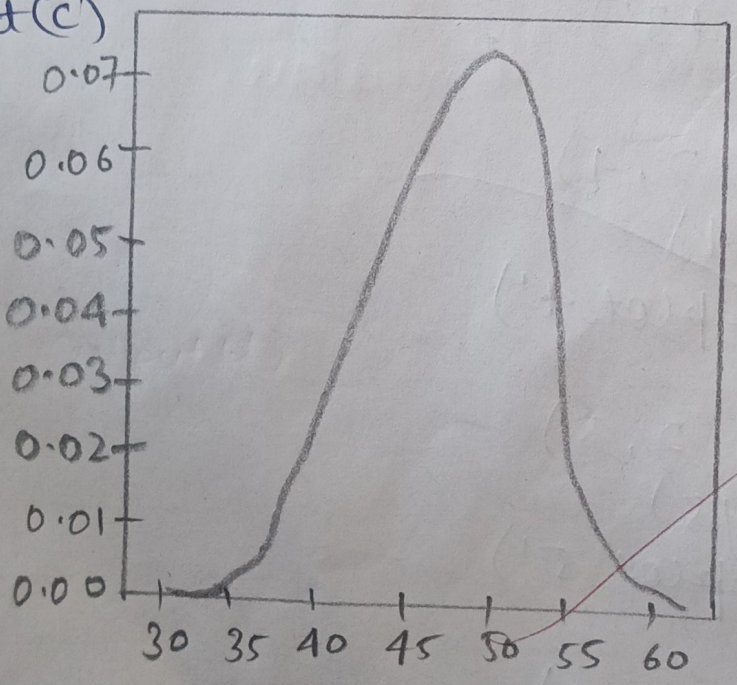
Output (a)



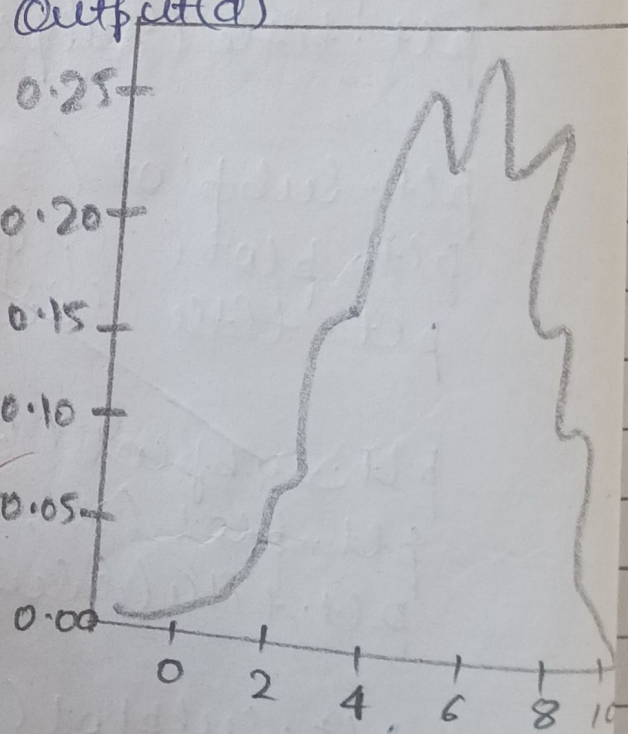
Output (b)



Output (c)



Output (d)



```
df = pd.read_csv("data.csv")
sns.lineplot(x=df['Col_A'], y=df['Col_B'])
plt.show()
```

(f) Draw a Scatterplot between 2 parameters.

```
sns.scatterplot(x=df['Col_A'], y=df['Col_B'])
plt.show()
```

(g) Draw a boxplot between 2 parameters.

```
sns.boxplot(x=df['Duration'], y=df['Maxpulse'])
plt.show()
```

(i) Draw a violinplot between the parameter

```
sns.violinplot(x=df['Duration'], y=df['Maxpulse'])
plt.show()
```

(j) Draw a Swarmplot between 2 parameters.

```
sns.swarmplot(x=df['Duration'], y=df['Maxpulse'])
plt.show()
```

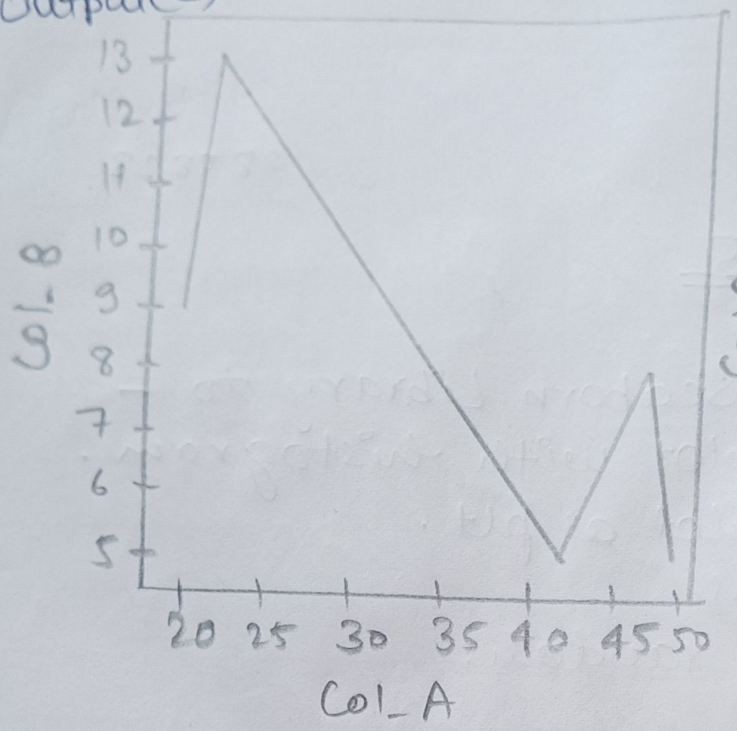
(k) Draw a barplot between 2 parameters.

```
sns.barplot(x=df['Col_A'], y=df['Col_B'])
plt.show()
```

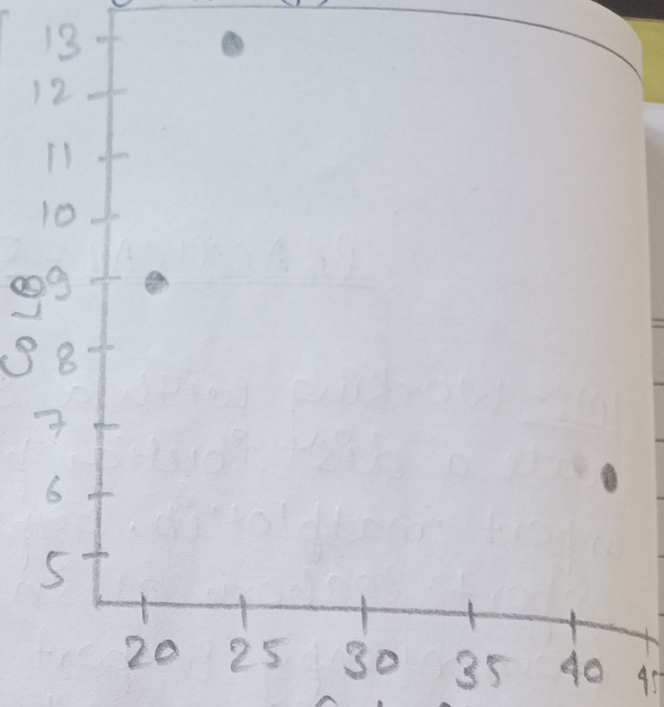
(l) Draw a catplot between 2 parameter.

```
sns.catplot(x=df['Duration'], y=df['Maxpulse'])
plt.show()
```

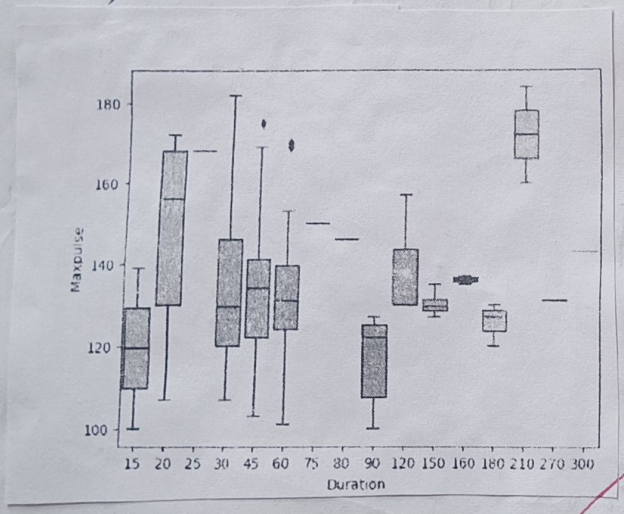
Output(e)



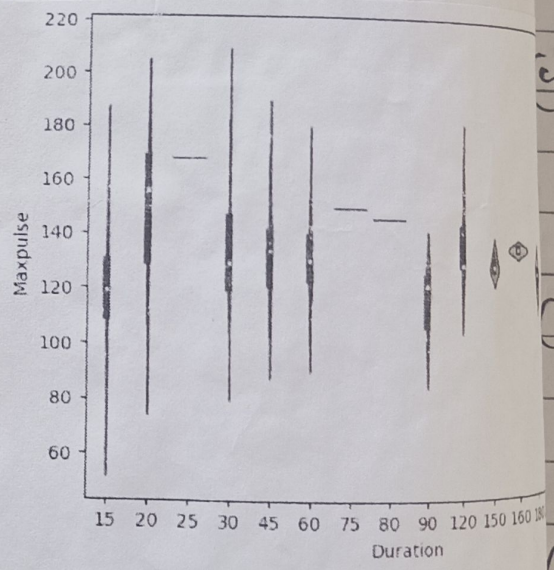
Output(f)



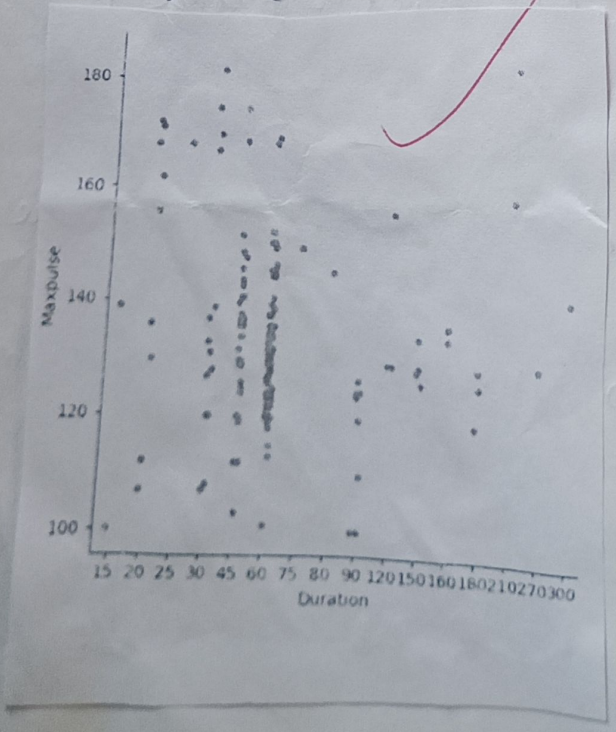
Output(g)



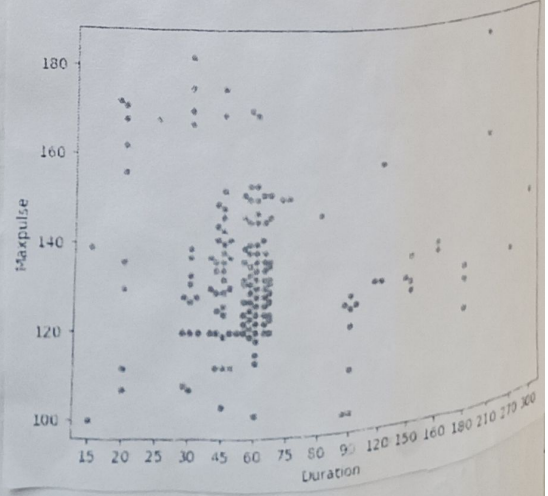
Output(h)



Output(i)



Output(j)



(m) Draw a pie chart between two parameters.

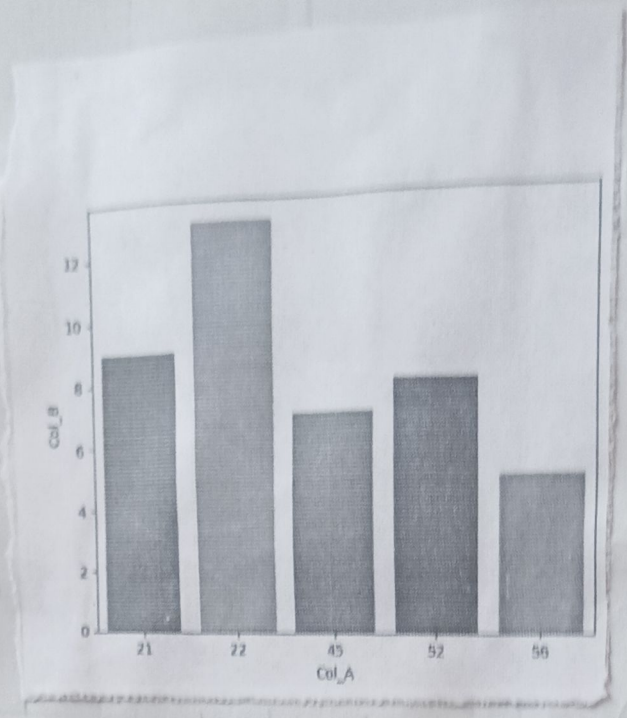
```
import matplotlib.pyplot as plt.
plt.pie(d df = pd.read_csv('data.csv')
duration_sum = df.groupby('Duration')['Calories'].
                sum()

plt.figure(figsize=(4,4))
plt.pie(duration_sum, labels = duration_sum.index,
        autopct='%0.1f%%', startangle = 90).
plt.title('Pie chart between Duration & Calories')
plt.axis('equal')
plt.show().
```

boxplot ? - with mean, 25th, 50th & 75th.

OBSERVATION

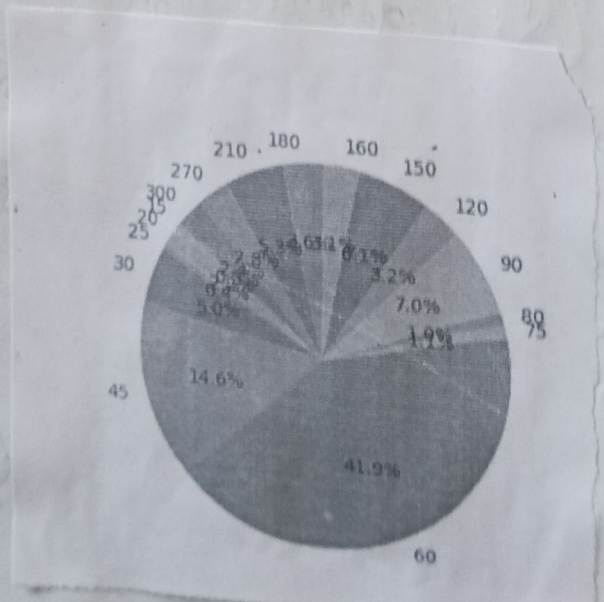
Seaborn with Matplotlib provide us with numerous alluring graphs through which one can easily analyze weak points, explore data with a deeper understanding & end up with a great insight into data & gaining the highest accuracy after training it through different Algorithms.

Output (k)



Output (m)

Pie-Chart of Duration & Calories



Page No.
Date 04 08 23

PRACTICAL - 6

AIM: (a) WAP to import load_iris from sklearn dataset and describe all the parameters in load_iris.

```
from sklearn.datasets import load_iris
import pandas as pd

iris = load_iris()
df = pd.DataFrame(iris.data, columns = iris.
                                    feature_names)
df['target'] = iris.target
df.

df.describe()   # describe the dataframe.

# Attributes of load_iris.
from sklearn.datasets import load_iris.
 iris = load_iris()
 x = iris.data
 y = iris.target
 feature_names = iris.feature_names
 target_names = iris.target_names
 print("Feature names:", feature_names)
 print("Target names:", target_names)
```

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	target
count	150.000000	150.000000	150.000000	150.000000	150.000000
mean	5.843333	3.057333	3.758000	1.199333	1.000000
std	0.828066	0.435866	1.765298	0.762238	0.819232
min	4.300000	2.000000	1.000000	0.100000	0.000000
25%	5.100000	2.800000	1.600000	0.300000	0.000000
50%	5.800000	3.000000	4.350000	1.300000	1.000000
75%	6.400000	3.300000	5.100000	1.800000	2.000000
max	7.900000	4.400000	6.900000	2.500000	2.000000

(b) Draw a Scatter plot between the sepal-length and sepal-width.

```
data = sns.load_dataset('iris')
sns.scatterplot(data, x='sepal_length',
                y='sepal_width', hue='sepal_length')
plt.show()
```

PRACTICAL-7

(a) To perform Dataset loading using numpy, pandas & csv module.

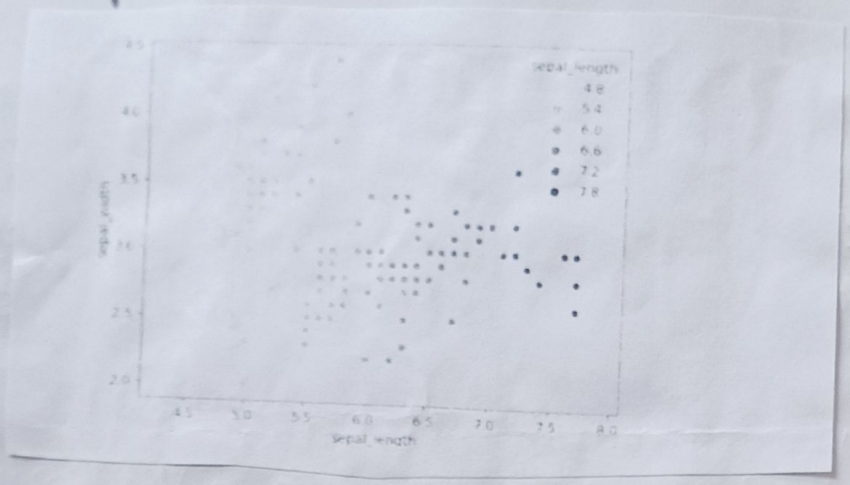
- From Numpy

```
from Numpy import loadtxt
data = r"C:\Users\hp\Downloads\pima-indians-diabetes.csv"
datapath = open(data, 'r')
data = loadtxt(datapath, delimiter = ',')
print(data.shape)
print(data[:3])
```

- Read a csv file using header from pandas (without header)

```
from pandas import read_csv
path = r"C:\Users\hp\Downloads\pima-indians-diabetes.csv"
df = read_csv(path, delimiter = ',')
headernames = ['sepal-length', 'sepal-width', 'petal-length',
               'petal-width', 'target']
df = read_csv(path, names = headernames)
df.
```

Output(b)



Output (a) (768, 9)

[[6. 148. 72. 35. 0. 33.6 0.627 50.
[1. 85. 66. 29. 0. 26.6 0.351 31.
[8. 183. 64. 0. 0. 23.3 0.672 32.

				Sepal-length	Sepal-width	petal-length	petal-width	Target
6	148	72	35	0	33.6	0.627	50	1
1	85	66	29	0	26.6	0.351	31	0
8	183	64	0	0	23.3	0.672	32	1
1	89	66	23	94	28.1	0.167	21	0
0	137	40	35	168	43.1	2.288	33	1
...	...	...	...	...	...	...	...	...
10	101	76	48	180	32.9	0.171	63	0
2	122	70	27	0	36.8	0.340	27	0
5	121	72	23	112	26.2	0.245	30	0
1	126	60	0	0	30.1	0.349	47	1
	93	70	31	0	30.4	0.315	23	0

768 rows × 5 columns

Page No.
Date

- Read a csv file without header using pandas.

```
from pandas import read_csv
path = r"C:|Users|hp|Downloads|diabetes.csv"
df = read_csv(path, delimiter = ',')
df.
```

- Using csv module.

```
import numpy as np
import csv
with open(path, 'r') as f:
    readdata = csv.reader(f)
    header = next(readdata)
    listdata = list(readdata)
    data = np.array(listdata)
print(header)
print(data[:3])
```

OBSERVATION

Datasets structure of the groundwork of effective Machine-learning projects. Understanding the various kinds of dataset, the significance of data pre-processing, & the job of training & testing datasets are key stages towards building powerful Models. By utilizing well-known sources such as Kaggle, Aws, Data Researchers, Microsoft Datasets, government Datasets etc.

	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

Output :- csv module

['Pregnancies', 'Glucose', 'BloodPressure', 'SkinThickn
'Insulin', 'BMI', 'DiabetesPedigreeFunction', 'Age',
[['6', '148' '72' '35' '0' '33.6' '0.627' '50'
['1' '85' '66' '29' '0' '26.6' '0.351' '31' '0
['8' '183' '64' '0' '0' '23.3' '0.672' '32' '

Page No.
Date 11 08 23

PRACTICAL-8

AIM:- To perform the Outlier Detection using statistical and Visualization Technique i.e, (box plot and Scatter plot).

- Statistical technique i.e Z-score and IQR Range.

```
from sklearn.datasets import load_iris
iris = load_iris()
x = iris.data
y = iris.target
feature_names = iris.feature_names
target_names = iris.target_names
# Z-Score Method
import pandas as pd
threshold = 2
column_name = ['sepal_length', 'sepal_width',
               'petal_length', 'petal_width']
df = pd.DataFrame(x, columns = column_name)
sepal_length = df["sepal_length"]
z_score = (sepal_length - sepal_length.mean())/
                          sepal_length.std()
print("Z-score:", z_score)
outliers = df[abs(z_score) > threshold]
print("Outliers:\n", outliers)
null_values = df.isnull() # check Null Values
deleted_df = df.drop(outliers.index, axis=0)
```

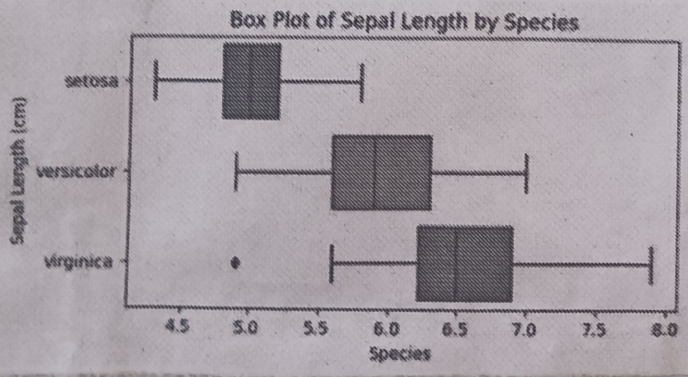
```
z-score:  0     -0.897674
1      -1.139200
2      -1.380727
3      -1.501490
4      -1.018437
          ...
145     1.034539
146     0.551486
147     0.793012
148     0.430722
149     0.068433
Name: sepal_length, Length: 150, dtype: float64
outliers:
      sepal_length  sepal _width  peta_length   petal_width
105            7.6           3.0           6.6           2.1
117            7.7           3.8           6.7           2.2
118            7.7           2.6           6.9           2.3
122            7.7           2.8           6.7           2.0
131            7.9           3.8           6.4           2.0
135            7.7           3.0           6.1           2.3
```

Page No.
Date

```
# IQR (Inter-Quartile Range)
Q1 = df[sepal_length].quantile(0.25)
Q3 = df[sepal_length].quantile(0.75)
IQR = Q3 - Q1
print("IQR: ", IQR)
threshold = 1.5 * IQR
outliers = df[(df[sepal_length] < (q1 - threshold))
              | (df[sepal_length] > (q3 + threshold))]
print("outliers are: \n", outliers)
```

- Visualization Technique i.e, boxplot and scatterplot.

```
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.datasets import load_iris
iris = load_iris()
data = iris.data
target = iris.target
import pandas as pd
df = pd.DataFrame(data, column = iris.feature_names)
df['Species'] = iris.target_names[target]
# Boxplot
plt.figure(figsize = (6,3))
sns.boxplot(x = 'sepal length (cm)', y = 'Species',
            data = df)
plt.title("Boxplot of Sepal length by Species")
plt.xlabel("Species")
plt.ylabel("Sepal length (cm)")
plt.show()
```


Box Plot of Sepal Length by Species
setosa
versicolor
virginica
Sepal Length (cm)
4.5
5.0
5.5
6.0
6.5
7.0
7.5
8.0
Species

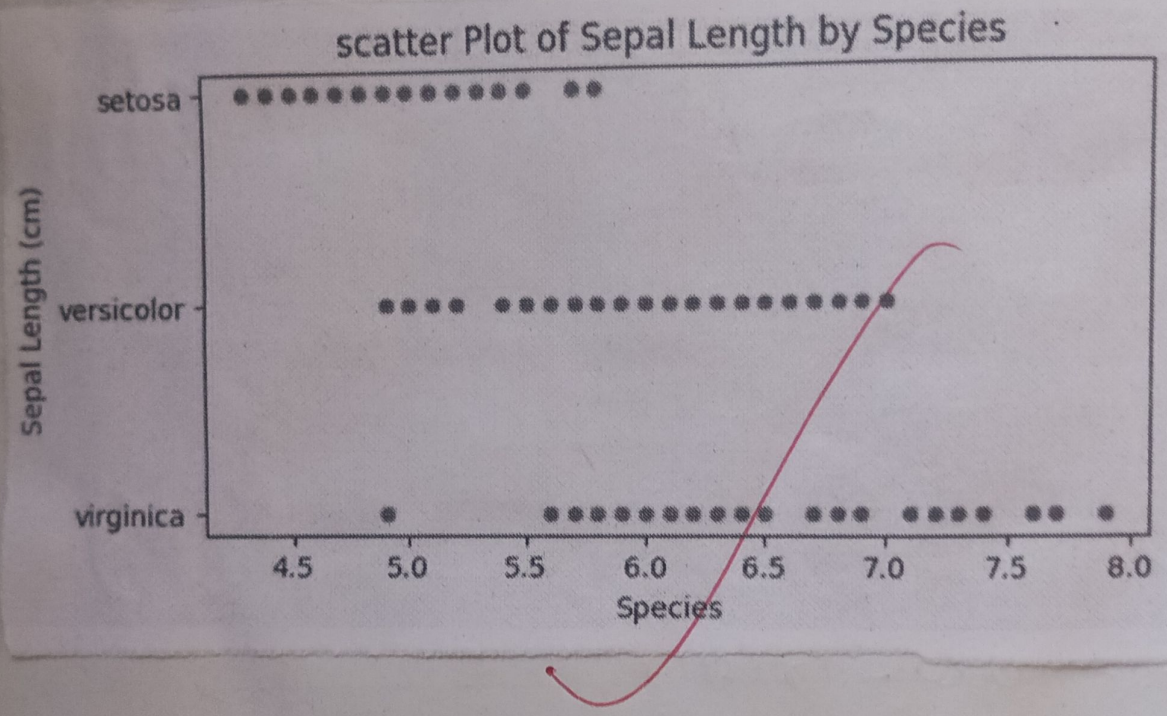
Page No.
Date

Scatter plot

```
import seaborn as sns
import pandas as pd
import matplotlib.pyplot as plt.
from sklearn.datasets import load_iris

iris = load_iris()
data = iris.data
target = iris.target
df = pd.DataFrame(data, columns = iris.feature_names)
df['Species'] = iris.target_names[target]
plt.figure(figsize = (6,3))
sns.scatterplot(x = 'sepal length (cm)', y = 'Species',
                data = df)
plt.title("scatter plot of sepal length by Species")
plt.xlabel("Species")
plt.ylabel("sepal length (cm)")
plt.show().
```

OBSERVATION

In this, using the IQR method can find more "outliers" than using the z-score (standard deviation) method. It depends on the distribution of the data. Data that's peaked with long tails will have a comparatively low IQR, so the IQR method will find lot of outliers.


scatter Plot of Sepal Length by Species
setosa
versicolor
virginica
Sepal Length (cm)
4.5
5.0
5.5
6.0
6.5
7.0
7.5
8.0
Species

Page No. _____
Date 18 09 23

PRACTICAL-9

AIM:- To handle the missing numerical data and perform One hot Encoding on Categorical Data.

- Handling the missing numerical data :-

```
import pandas as pd
import numpy as np.

data = {'Age': [20, 30, np.nan, 26, 35, np.nan],
        'Salary': [40000, np.nan, 60000, 70000,
                   np.nan, 80000],
        'Rank_score': [92, np.nan, 78, np.nan, 81, 92]
       }

df = pd.DataFrame(data)
print("dataframe :\n", df)
mean_data = df.mean()
df.fillna(mean_data, inplace=True)
print("after missing values handled :\n", df)
```

- One-Hot Encoding by get_dummies()

```
import numpy as np.
import pandas as pd.
data = pd.read_csv('Employee.csv')
one_hot_encoded_data = pd.get_dummies(data,
                       columns=['Gender', 'Remarks'])
one_hot_encoded_data.
```

dataframe:

	Age	salary	Rank_score
0	20.0	40000.0	82.0
1	30.0	NaN	NaN
2	NaN	60000.0	78.0
3	26.0	70000.0	NaN
4	35.0	NaN	81.0
5	NaN	80000.0	92.0

after missing values handled:

	Age	salary	Rank_score
0	20.00	40000.0	82.00
1	30.00	62500.0	83.25
2	27.75	60000.0	78.00
3	26.00	70000.0	83.25
4	35.00	62500.0	81.00
5	27.75	80000.0	92.00

	Employee_ID	Gender_Female	Gender_Male	Remarks_Good	Remarks_Great	Remarks_Nice
0	45	0	1	0	0	1
1	12	1	0	1	0	0
2	78	1	0	0	1	0
3	35	1	0	0	0	1
4	7	0	1	0	1	0

Page No. _____
Date

- By sklearn.prepocessing module.

```
import pandas as pd.
import numpy as np.
from sklearn.preprocessing import OneHotEncoder

data = pd.read_csv('Employee.csv')
data['Gender'] = data['Gender'].astype('category')
data['Remarks'] = data['Remarks'].astype('category')

data['Gen_new'] = data['Gender'].cat.codes
data['Rem_new'] = data['Remarks'].cat.codes
# Create an instance of one-Hot Encoder.
enc = OneHotEncoder()
# passing encoded columns
enc_data = pd.DataFrame(enc.fit_transform(data[[
                        'Gen_new','Rem_new']]).
                                        toarray())
# Merge with main.
New_df = data.join(enc_data)
print("one-hot encoding: \n", New_df)
```

OBSERVATION:-

Here, we have converted the enc.fit_transform() method to an array because the fit_tranform method of OneHotEncoder returns SCIPy Sparse matrix so converting to an array first enables us to save space when we have a huge number of categorical variables.

23/07/23

one-hot encoding:

	Employee_ID	Gender	Remarks	Gen_new	Rem_new	0	1	2	3	4
0	45	Male	Nice	1	2	0.0	1.0	0.0	0.0	1.0
1	12	Female	Good	0	0	1.0	0.0	1.0	0.0	0.0
2	78	Female	Great	0	1	1.0	0.0	0.0	1.0	0.0
3	35	Female	Nice	0	2	1.0	0.0	0.0	0.0	1.0
4	7	Male	Great	1	1	0.0	1.0	0.0	1.0	0.0