Sinhgad Institutes

Sinhgad Technical Educational Society‟s

SKN SINHGAD INSTITUTE of TECHNOLOGY & SCIENCE, LONAVALA – 410 401.

DEPARTMENT OF COMPUTER ENGINEERING

CERTIFICATE

*This is to certify that Mr./Miss.*_____

Of class- TE, has completed all the practical work in the Data Science And Big Data Analytics Laboratory(310251) *satisfactorily as prescribed by Savitribai Phule Pune University, in the academic year 2023-2024.*

Prof.C.P.Lachake	Dr.S.M.Patil	Dr.M.S.Rohokale
Subject In-charge	Head of Department	Principal

DEPARTMENT OF COMPUTER ENGINEERING, SKNSITS, LONAVALA.

INDEX

Sr. No	Title of Experiment	Page No.	Date	Marks	Sign
1	**Data Wrangling, I** Perform the following operations using Python on any open source dataset (e.g., data.csv) 1.Import all the required Python Libraries. 2.Locate an open source data from the web 3.Load the Dataset into pandas dataframe. 4.Data Preprocessing 4.Data Formatting and Data Normalization 5.Turn categorical variables into quantitative variables in Python.				
2	**Data Wrangling II** Create an "Academic performance" dataset of students and perform the following operations using Python. 1.Scan all variables for missing values and inconsistencies.. 2.Scan all numeric variables for outliers. 3.Apply data transformations on at least one of the variables.				
3	**Descriptive Statistics - Measures of Central Tendency and variability** Perform the following operations on any open source dataset. 1. Provide summary statistics for a dataset with numeric variables grouped by one of the qualitative variable. Create a list that contains a numeric value for each response to the categorical variable. 2. Write a Python program to display some basic statistical details like percentile, mean, standard deviation etc. of the species of 'Iris-setosa', 'Iris-versicolor' and 'Iris-versicolor' of iris.csv dataset.				
4	**Data Analytics I** Create a Linear Regression Model using Python/R to predict home prices using Boston Housing Dataset .The Boston Housing dataset contains information about various houses in Boston through different parameters. There are 506 samples and 14 feature variables in this dataset.				

5	**Data Analytics II** 1.Implement logistic regression using Python/R to perform classification on Social_Network_Ads.csv dataset. Compute Confusion matrix to find TP, FP, TN, FN, Accuracy, Error rate, Precision, Recall on the given dataset.				
6	**Data Analytics III** 1.Implement Simple Naïve Bayes classification algorithm using Python/R on iris.csv dataset. Compute Confusion matrix to find TP, FP, TN, FN, Accuracy, Error rate, Precision, Recall on the given dataset.				
7	**Text Analytics** 1.Extract Sample document and apply following document preprocessing methods: Tokenization, POS Tagging, stop words removal, Stemming and Lemmatization. Create representation of document by calculating Term Frequency and Inverse Document Frequency.				
8	**Data Visualization I** 1.Use the inbuilt dataset 'titanic'. The dataset contains 891 rows and contains information about the passengers who boarded the unfortunate Titanic ship. Use the Seaborn library to see if we can find any patterns in the data. Write a code to check how the price of the ticket (column name: 'fare') for each passenger is distributed by plotting a histogram.				
9	**Data Visualization II** 1.Use the inbuilt dataset 'titanic' as used in the above problem. Plot a box plot for distribution of age with respect to each gender along with the information about whether they survived or not. (Column names : 'sex' and 'age') Write observations on the inference from the above statistics.				
10	**Data Visualization III** Download the Iris flower dataset or any other dataset into a DataFrame. (e.g., https://archive.ics.uci.edu/ml/datasets/Iris). Scan the dataset and give the inference as: 1. List down the features and their types (e.g., numeric, nominal) available in the dataset. 2. Create a histogram for each feature in the dataset to illustrate the feature distributions.				

11	Write a code in JAVA for a simple WordCount application that counts the number of occurrences of each word in a given input set using the Hadoop MapReduce framework on local-standalone set-up.				
12	Design a distributed application using MapReduce which processes a log file of a system.				
13	Write a simple program in SCALA using Apache Spark framework				
14	Use the following covid_vaccine_statewise.csv dataset and perform following analytics on the given datails. a. Describe the dataset b. Number of persons state wise vaccinated for first dose in India c. Number of persons state wise vaccinated for second dose in India d. Number of Males vaccinated d. Number of females vaccinated				
15	Write a case study to process data driven for Digital Marketing **OR** Health care systems with Hadoop Ecosystem components as shown. 1.HDFS: Hadoop Distributed File System 2.YARN: Yet Another Resource Negotiator 3.MapReduce: Programming based Data Processing 4.Spark: In-Memory data processing 5.PIG, HIVE: Query based processing of data services 6.HBase: NoSQL Database 7.Mahout, Spark MLLib				

Prof.C.P.Lachake
Subject In-charge

Dr.S.M.Patil
Head of Department

Dr.M.S.Rohokale
Principal

Practical No. 01

Data Wrangling Perform the following operations using Python on any open source dataset (e.g., data.csv)

1. Import all the required Python Libraries.
2. Locate an open source data from the web (e.g. https://www.kaggle.com). Provide a clear

description of the data and its source (i.e., URL of the web site). 3. Load the Dataset into pandas data frame. 4. Data Preprocessing: check for missing values in the data using pandas insult(), describe() function to get some initial statistics. Provide variable descriptions. Types of variables etc. Check the dimensions of the data frame. 5. Data Formatting and Data Normalization: Summarize the types of variables by checking the data types (i.e., character, numeric, integer, factor, and logical) of the variables in the data set. If variables are not in the correct data type, apply proper type conversions. 6. Turn categorical variables into quantitative variables in Python. In addition to the codes and outputs, explain every operation that you do in the above steps and explain everything that you do to import/read/scrape the data set.

In [ ]:
```
import pandas as pd
```

In [ ]:
```
df = pd.read_csv('/home/kartik/Documents/Python Notebooks/StudentsPerformance.csv')
```

In [ ]:
```
df
```

Out[ ]:

	gender	race/ethnicity	parental level of education	lunch	test preparation course	math score	reading score	writing score
0	female	group B	bachelor's degree	standard	none	72	72	74
1	female	group C	some college	standard	completed	69	90	88
2	female	group B	master's degree	standard	none	90	95	93
3	male	group A	associate's degree	free/reduced	none	47	57	44
4	male	group C	some college	standard	none	76	78	75
...	...	...	...	...	...	...	...	...
995	female	group E	master's degree	standard	completed	88	99	95
996	male	group C	high school	free/reduced	none	62	55	55
997	female	group C	high school	free/reduced	completed	59	71	65
998	female	group D	some college	standard	completed	68	78	77
999	female	group D	some college	free/reduced	none	77	86	86

1000 rows × 8 columns

In [ ]:
```
df.isnull()
```

Out[ ]:

	gender	race/ethnicity	parental level of education	lunch	test preparation course	math score	reading score	writing score
0	False	False	False	False	False	False	False	False
1	False	False	False	False	False	False	False	False
2	False	False	False	False	False	False	False	False
3	False	False	False	False	False	False	False	False
4	False	False	False	False	False	False	False	False
...	...	...	...	...	...	...	...	...
995	False	False	False	False	False	False	False	False
996	False	False	False	False	False	False	False	False
997	False	False	False	False	False	False	False	False
998	False	False	False	False	False	False	False	False
999	False	False	False	False	False	False	False	False

1000 rows × 8 columns

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

Out[ ]:

	math score	reading score	writing score
count	1000.00000	1000.000000	1000.000000
mean	66.08900	69.169000	68.054000
std	15.16308	14.600192	15.195657
min	0.00000	17.000000	10.000000
25%	57.00000	59.000000	57.750000
50%	66.00000	70.000000	69.000000
75%	77.00000	79.000000	79.000000
max	100.00000	100.000000	100.000000

In [ ]:
```
df.isnull().sum()
```

Out[ ]:
```
gender                       0
race/ethnicity               0
parental level of education  0
lunch                        0
test preparation course      0
math score                   0
reading score                0
writing score                0
dtype: int64
```

In [ ]:
```
df.notnull().sum()
```

Out[ ]:
```
gender                         1000
race/ethnicity                 1000
parental level of education    1000
lunch                          1000
test preparation course        1000
math score                     1000
reading score                  1000
writing score                  1000
dtype: int64
```

In [ ]:
```
df.notnull()
```

Out[ ]:

	gender	race/ethnicity	parental level of education	lunch	test preparation course	math score	reading score	writing score
0	True	True	True	True	True	True	True	True
1	True	True	True	True	True	True	True	True
2	True	True	True	True	True	True	True	True
3	True	True	True	True	True	True	True	True
4	True	True	True	True	True	True	True	True
...	...	...	...	...	...	...	...	...
995	True	True	True	True	True	True	True	True
996	True	True	True	True	True	True	True	True
997	True	True	True	True	True	True	True	True
998	True	True	True	True	True	True	True	True
999	True	True	True	True	True	True	True	True

1000 rows × 8 columns

In [ ]:
```
df.size
```

Out[ ]:
```
8000
```

In [ ]:
```
df.ndim
```

Out[ ]:
```
2
```

In [ ]:
```
df.shape
```

Out[ ]:
```
(1000, 8)
```

In [ ]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 8 columns):
 #   Column                       Non-Null Count  Dtype
---  ------                       --------------  -----
 0   gender                       1000 non-null   object
 1   race/ethnicity               1000 non-null   object
 2   parental level of education  1000 non-null   object
 3   lunch                        1000 non-null   object
 4   test preparation course      1000 non-null   object
 5   math score                   1000 non-null   int64
 6   reading score                1000 non-null   int64
 7   writing score                1000 non-null   int64
dtypes: int64(3), object(5)
memory usage: 62.6+ KB
```

In [ ]:
```
df['writing score'].astype(int)
```

Out[ ]:
```
0      74
1      88
2      93
3      44
4      75
      ..
995    95
996    55
997    65
998    77
999    86
Name: writing score, Length: 1000, dtype: int64
```

In [ ]:
```
df.dropna()
```

Out[ ]:

	gender	race/ethnicity	parental level of education	lunch	test preparation course	math score	reading score	writing score
0	female	group B	bachelor's degree	standard	none	72	72	74
1	female	group C	some college	standard	completed	69	90	88
2	female	group B	master's degree	standard	none	90	95	93
3	male	group A	associate's degree	free/reduced	none	47	57	44
4	male	group C	some college	standard	none	76	78	75
...	...	...	...	...	...	...	...	...
995	female	group E	master's degree	standard	completed	88	99	95
996	male	group C	high school	free/reduced	none	62	55	55
997	female	group C	high school	free/reduced	completed	59	71	65
998	female	group D	some college	standard	completed	68	78	77
999	female	group D	some college	free/reduced	none	77	86	86

1000 rows × 8 columns

In [ ]:
```
df['writing score'] = df['writing score'].astype(int)
```

In [ ]:
```
df
```

Out[ ]:

	gender	race/ethnicity	parental level of education	lunch	test preparation course	math score	reading score	writing score
0	female	group B	bachelor's degree	standard	none	72	72	74
1	female	group C	some college	standard	completed	69	90	88
2	female	group B	master's degree	standard	none	90	95	93
3	male	group A	associate's degree	free/reduced	none	47	57	44
4	male	group C	some college	standard	none	76	78	75
...	...	...	...	...	...	...	...	...
995	female	group E	master's degree	standard	completed	88	99	95
996	male	group C	high school	free/reduced	none	62	55	55
997	female	group C	high school	free/reduced	completed	59	71	65
998	female	group D	some college	standard	completed	68	78	77
999	female	group D	some college	free/reduced	none	77	86	86

1000 rows × 8 columns

In [ ]:
```
df["gender"] = df["gender"].replace({"female":0,"male":1})
```

In [ ]:
```
df
```

Out[ ]:

	gender	race/ethnicity	parental level of education	lunch	test preparation course	math score	reading score	writing score
0	0	group B	bachelor's degree	standard	none	72	72	74
1	0	group C	some college	standard	completed	69	90	88
2	0	group B	master's degree	standard	none	90	95	93
3	1	group A	associate's degree	free/reduced	none	47	57	44
4	1	group C	some college	standard	none	76	78	75
...	...	...	...	...	...	...	...	...
995	0	group E	master's degree	standard	completed	88	99	95
996	1	group C	high school	free/reduced	none	62	55	55
997	0	group C	high school	free/reduced	completed	59	71	65
998	0	group D	some college	standard	completed	68	78	77
999	0	group D	some college	free/reduced	none	77	86	86

1000 rows × 8 columns

Practical No. 02

Data Wrangling II Create an “Academic performance” dataset of students and perform the following operations using Python.

1. Scan all variables for missing values and inconsistencies. If there are missing values

and/or inconsistencies, use any of the suitable techniques to deal with them. 2. Scan all numeric variables for outliers. If there are outliers, use any of the suitable techniques to deal with them. 3. Apply data transformations on at least one of the variables. The purpose of this transformation should be one of the following reasons: to change the scale for better understanding of the variable, to convert a non-linear relation into a linear one, or to decrease the skewness and convert the distribution into a normal distribution. Reason and document your approach properly.

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

In [5]:
```
df = pd.read_csv("/home/kartik/Documents/Python Notebooks/StudentsPerformance.csv")
```

In [6]:
```
df
```

Out[6]:

	gender	race/ethnicity	parental level of education	lunch	test preparation course	math score	reading score	writing score
0	female	group B	bachelor's degree	standard	none	72	72	74
1	female	group C	some college	standard	completed	69	90	88
2	female	group B	master's degree	standard	none	90	95	93
3	male	group A	associate's degree	free/reduced	none	47	57	44
4	male	group C	some college	standard	none	76	78	75
...	...	...	...	...	...	...	...	...
995	female	group E	master's degree	standard	completed	88	99	95
996	male	group C	high school	free/reduced	none	62	55	55
997	female	group C	high school	free/reduced	completed	59	71	65
998	female	group D	some college	standard	completed	68	78	77
999	female	group D	some college	free/reduced	none	77	86	86

1000 rows × 8 columns

In [7]:
```
df.isnull().sum()
```

Out[7]:
```
gender                         0
race/ethnicity                 0
parental level of education    0
lunch                          0
test preparation course        0
math score                     0
reading score                  0
writing score                  0
dtype: int64
```

In [8]:
```
df.dropna()
```

Out[8]:

	gender	race/ethnicity	parental level of education	lunch	test preparation course	math score	reading score	writing score
0	female	group B	bachelor's degree	standard	none	72	72	74
1	female	group C	some college	standard	completed	69	90	88
2	female	group B	master's degree	standard	none	90	95	93
3	male	group A	associate's degree	free/reduced	none	47	57	44
4	male	group C	some college	standard	none	76	78	75
...	...	...	...	...	...	...	...	...
995	female	group E	master's degree	standard	completed	88	99	95
996	male	group C	high school	free/reduced	none	62	55	55
997	female	group C	high school	free/reduced	completed	59	71	65
998	female	group D	some college	standard	completed	68	78	77
999	female	group D	some college	free/reduced	none	77	86	86

1000 rows × 8 columns

In [9]:
```
math_score_mean = df["math score"].mean()
df["math score"] = df["math score"].fillna(math_score_mean)
df
```

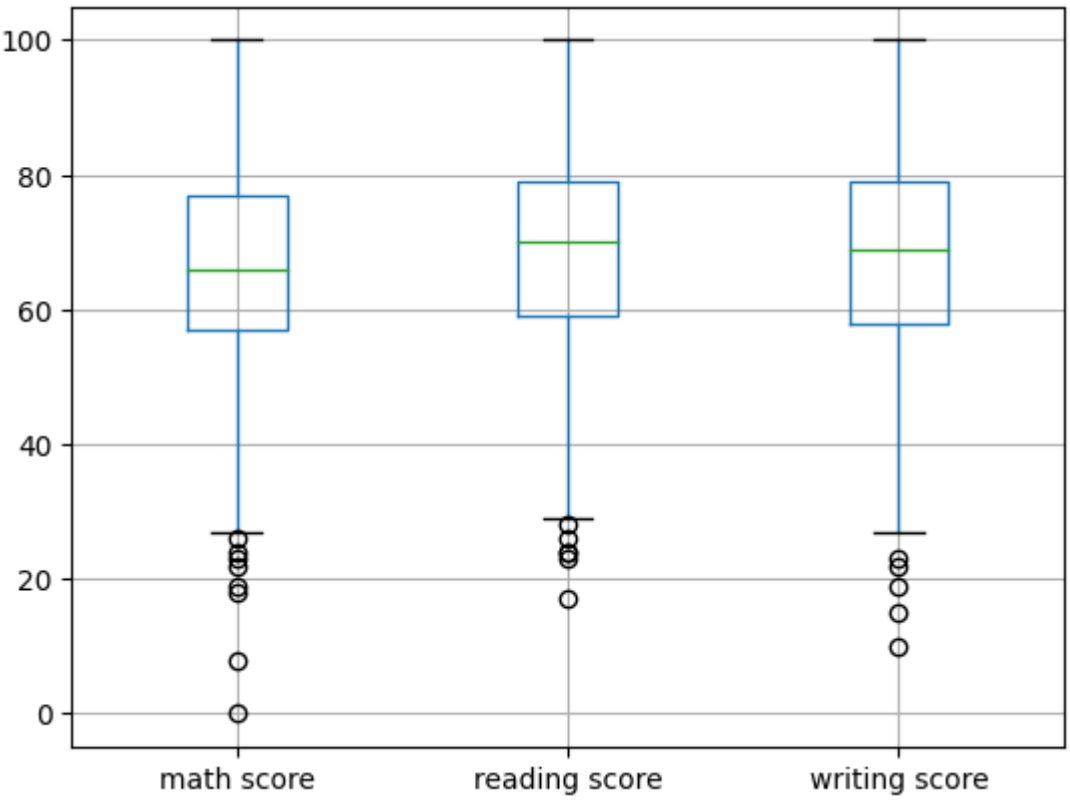
Out[9]:

	gender	race/ethnicity	parental level of education	lunch	test preparation course	math score	reading score	writing score
0	female	group B	bachelor's degree	standard	none	72	72	74
1	female	group C	some college	standard	completed	69	90	88
2	female	group B	master's degree	standard	none	90	95	93
3	male	group A	associate's degree	free/reduced	none	47	57	44
4	male	group C	some college	standard	none	76	78	75
...	...	...	...	...	...	...	...	...
995	female	group E	master's degree	standard	completed	88	99	95
996	male	group C	high school	free/reduced	none	62	55	55
997	female	group C	high school	free/reduced	completed	59	71	65
998	female	group D	some college	standard	completed	68	78	77
999	female	group D	some college	free/reduced	none	77	86	86

1000 rows × 8 columns

In [10]:
```
df.boxplot()
```

Out[10]: <Axes: >



In [11]:
```
newdf = df[df["math score"] > 30]
newdf
```

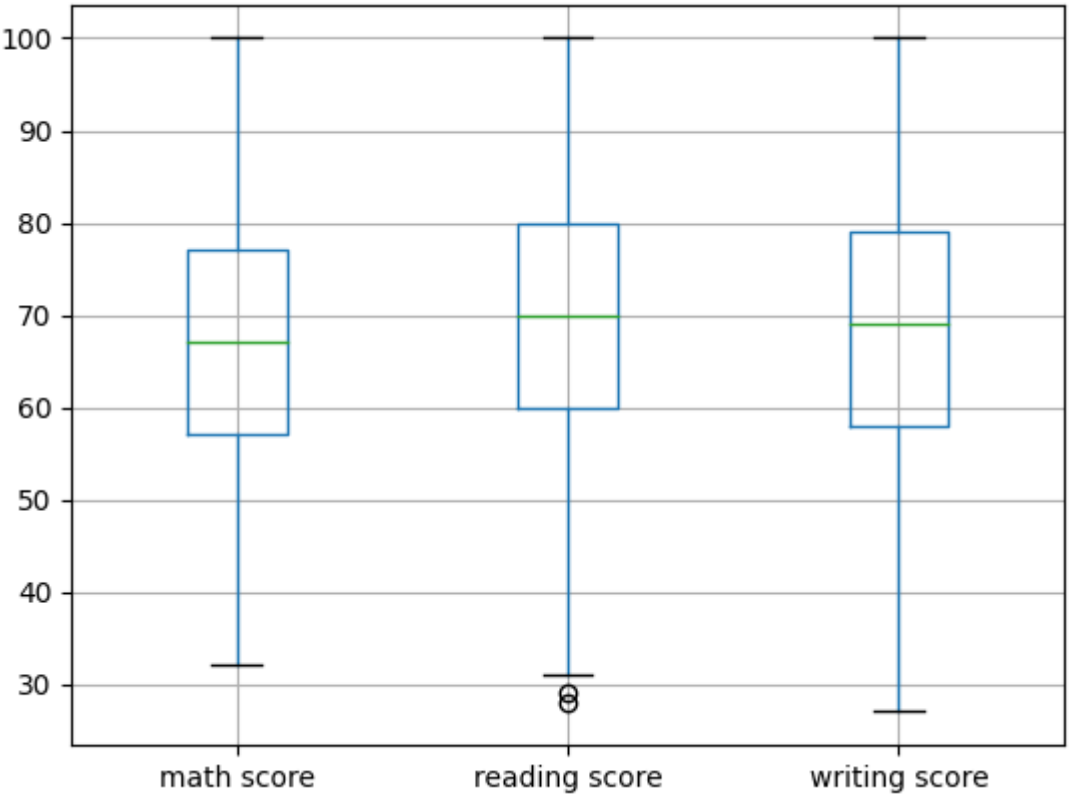
Out[11]:

	gender	race/ethnicity	parental level of education	lunch	test preparation course	math score	reading score	writing score
0	female	group B	bachelor's degree	standard	none	72	72	74
1	female	group C	some college	standard	completed	69	90	88
2	female	group B	master's degree	standard	none	90	95	93
3	male	group A	associate's degree	free/reduced	none	47	57	44
4	male	group C	some college	standard	none	76	78	75
...	...	...	...	...	...	...	...	...
995	female	group E	master's degree	standard	completed	88	99	95
996	male	group C	high school	free/reduced	none	62	55	55
997	female	group C	high school	free/reduced	completed	59	71	65
998	female	group D	some college	standard	completed	68	78	77
999	female	group D	some college	free/reduced	none	77	86	86

984 rows × 8 columns

In [12]:
```
newdf.boxplot()
```

Out[12]: <Axes: >



In [ ]:

Practical No. 03

Descriptive Statistics - Measures of Central Tendency and variability Perform the following operations on any open source dataset (e.g., data.csv)

1. Provide summary statistics (mean, median, minimum, maximum, standard deviation) for

a dataset (age, income etc.) with numeric variables grouped by one of the qualitative (categorical) variable. For example, if your categorical variable is age groups and quantitative variable is income, then provide summary statistics of income grouped by the age groups. Create a list that contains a numeric value for each response to the categorical variable. 2. Write a Python program to display some basic statistical details like percentile, mean, standard deviation etc. of the species of 'Iris-setosa', 'Iris-versicolor' and 'Iris- versicolor' of iris.csv dataset. Provide the codes with outputs and explain everything that you do in this step.

In [4]:
```
import pandas as pd
```

In [6]:
```
df = pd.read_csv("/home/kartik/Documents/Python Notebooks/Iris.csv")
```

In [7]:
```
df
```

Out[7]:

	sepallength	sepalwidth	petallength	petalwidth	class
0	5.1	3.5	1.4	0.2	Iris-setosa
1	4.9	3.0	1.4	0.2	Iris-setosa
2	4.7	3.2	1.3	0.2	Iris-setosa
3	4.6	3.1	1.5	0.2	Iris-setosa
4	5.0	3.6	1.4	0.2	Iris-setosa
...	...	...	...	...	...
145	6.7	3.0	5.2	2.3	Iris-virginica
146	6.3	2.5	5.0	1.9	Iris-virginica
147	6.5	3.0	5.2	2.0	Iris-virginica
148	6.2	3.4	5.4	2.3	Iris-virginica
149	5.9	3.0	5.1	1.8	Iris-virginica

150 rows × 5 columns

In [8]:
```
df.describe()
```

Out[8]:

	sepallength	sepalwidth	petallength	petalwidth
count	150.000000	150.000000	150.000000	150.000000
mean	5.843333	3.054000	3.758667	1.198667
std	0.828066	0.433594	1.764420	0.763161
min	4.300000	2.000000	1.000000	0.100000
25%	5.100000	2.800000	1.600000	0.300000
50%	5.800000	3.000000	4.350000	1.300000
75%	6.400000	3.300000	5.100000	1.800000
max	7.900000	4.400000	6.900000	2.500000

In [10]:
```
df["sepallength"].describe()
```

Out[10]:
```
count    150.000000
mean       5.843333
std        0.828066
min        4.300000
25%        5.100000
50%        5.800000
75%        6.400000
max        7.900000
Name: sepallength, dtype: float64
```

In [11]:
```
df.groupby("class").describe()
```

Out[11]:

	sepallength								sepalwidth		...	petalle
	count	**mean**	**std**	**min**	**25%**	**50%**	**75%**	**max**	**count**	**mean**	**...**	**75%**
class												
Iris-setosa	50.0	5.006	0.352490	4.3	4.800	5.0	5.2	5.8	50.0	3.418	...	1.575
Iris-versicolor	50.0	5.936	0.516171	4.9	5.600	5.9	6.3	7.0	50.0	2.770	...	4.600
Iris-virginica	50.0	6.588	0.635880	4.9	6.225	6.5	6.9	7.9	50.0	2.974	...	5.875

3 rows × 32 columns

In [12]:
```
df.groupby("class").describe().sum()
```

Out[12]:

sepallength	count	150.000000
	mean	17.530000
	std	1.504540
	min	14.100000
	25%	16.625000
	50%	17.400000
	75%	18.400000
	max	20.700000
sepalwidth	count	150.000000
	mean	9.162000
	std	1.017319
	min	6.500000
	25%	8.450000
	50%	9.200000
	75%	9.850000
	max	11.600000
petallength	count	150.000000
	mean	11.276000
	std	1.195317
	min	8.500000
	25%	10.500000
	50%	11.400000
	75%	12.050000
	max	13.900000
petalwidth	count	150.000000
	mean	3.596000
	std	0.579612
	min	2.500000
	25%	3.200000
	50%	3.500000
	75%	4.100000
	max	4.900000

dtype: float64

In [ ]:

Practical No. 04

Data Analytics I Create a Linear Regression Model using Python/R to predict home prices using Boston Housing Dataset (https://www.kaggle.com/c/boston-housing). The Boston Housing dataset contains information about various houses in Boston through different parameters. There are 506 samples and 14 feature variables in this dataset. The objective is to predict the value of prices of the house using the given features.

In [24]:
```
import pandas as pd
import numpy as np

from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error
```

In [51]:
```
df = pd.read_csv("/home/kartik/Documents/Python Notebooks/BostonHousing.csv")
df = df.dropna()
df.head()
```

Out[51]:

	crim	zn	indus	chas	nox	rm	age	dis	rad	tax	ptratio	b	lstat	medv
0	0.00632	18.0	2.31	0	0.538	6.575	65.2	4.0900	1	296	15.3	396.90	4.98	24.0
1	0.02731	0.0	7.07	0	0.469	6.421	78.9	4.9671	2	242	17.8	396.90	9.14	21.6
2	0.02729	0.0	7.07	0	0.469	7.185	61.1	4.9671	2	242	17.8	392.83	4.03	34.7
3	0.03237	0.0	2.18	0	0.458	6.998	45.8	6.0622	3	222	18.7	394.63	2.94	33.4
4	0.06905	0.0	2.18	0	0.458	7.147	54.2	6.0622	3	222	18.7	396.90	5.33	36.2

In [52]:
```
df.columns
```

Out[52]:
```
Index(['crim', 'zn', 'indus', 'chas', 'nox', 'rm', 'age', 'dis', 'rad', 'tax',
       'ptratio', 'b', 'lstat', 'medv'],
      dtype='object')
```

In [53]:
```
x = df[['crim', 'zn', 'indus', 'chas', 'nox', 'rm', 'age', 'dis', 'rad', 'tax',
       'ptratio', 'b', 'lstat']]
x.head()
```

Out[53]:

	crim	zn	indus	chas	nox	rm	age	dis	rad	tax	ptratio	b	lstat
0	0.00632	18.0	2.31	0	0.538	6.575	65.2	4.0900	1	296	15.3	396.90	4.98
1	0.02731	0.0	7.07	0	0.469	6.421	78.9	4.9671	2	242	17.8	396.90	9.14
2	0.02729	0.0	7.07	0	0.469	7.185	61.1	4.9671	2	242	17.8	392.83	4.03
3	0.03237	0.0	2.18	0	0.458	6.998	45.8	6.0622	3	222	18.7	394.63	2.94
4	0.06905	0.0	2.18	0	0.458	7.147	54.2	6.0622	3	222	18.7	396.90	5.33

In [54]:
```
y = df['medv']
y.head()
```

Out[54]:
```
0    24.0
1    21.6
2    34.7
3    33.4
4    36.2
Name: medv, dtype: float64
```

In [55]:
```
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.25,random_state=42)
```

In [56]:
```
model = LinearRegression()
model.fit(x_train,y_train)
```

Out[56]:
```
▾ LinearRegression
LinearRegression()
```

In [59]:
```
y_pred = model.predict(x_test)
y_pred
```

Out[59]:
```
array([10.82520289, 22.97716771, 15.45617932, 33.55363131, 22.96357871,
       11.52151263, 12.76018157, 19.74412591, 21.33180568, 11.7372368 ,
       18.75187948, 30.04070255, -0.73011025, 25.78030298,  3.02335542,
        8.49359394, 24.07065874, 18.57018302, 25.24003893, -6.24945751,
       13.33486252, 19.08911255, 27.0053246 , 19.59024598, 22.40273032,
       16.47206196, 28.79995249, 26.24334357, 18.42194929, 21.27338464,
       20.62838908, 30.49181729, 17.87807473, 31.53661897, 31.16125663,
       22.20316674,  7.79878712, 23.70737642,  8.54510946, 25.0261323 ,
       12.99764774, 36.12050346, 14.45054578, 30.51121076, 13.02756177,
       28.48505695, 30.34475695, 20.15771804, 18.46362559, 13.69183882,
       24.00613417, 32.99780499, 16.4544118 , 11.66937979, 34.39689874,
       33.37924364, 17.77929903, 18.70970757, 16.25656178, 27.35347057,
       20.48252629, 40.60322048, 20.53694472,  8.20383246, 25.97767891,
       27.81783878, 12.08008232,  7.62795819, 27.14868012, 16.44871208,
       23.46295285, 14.63324084, 40.28319824, 28.66936219, 23.1422757 ,
       23.95467347, 35.49409707, 24.49032705, 20.75456047, 15.97157605,
       27.18392572, 27.90827964, 21.23340735, 29.37584949, 23.9104647 ,
       29.29067164, 24.22591482, 20.08729338, 18.20901184, 44.2614741 ,
        4.63790216, 19.31301769, 17.2763475 , 23.72223401,  7.38111706,
       17.02032604, 31.01206401, 21.14872276, 10.9653362 , 20.85193641,
       24.18859543, 17.31441353, 12.19815419, 19.11197493, 19.50296116,
       22.01189876, 35.62919117, 31.55632051, 20.24630891, 20.2365227 ,
       14.26461161, 11.71171865, 21.66497318, 15.74320729, 20.29084409,
       19.52326714, 25.23052357, 23.83851879, 23.14474265, 22.78985166,
       40.21608485, 27.45423907, 24.8064738 , 30.06864408, 30.07124307,
       38.69282771])
```

In [61]:
```
model.score(x_train,y_train)
```

Out[61]:
```
0.7335900413194543
```

In [62]:
```
model.score(x_test,y_test)
```

Out[62]:
```
0.7459403901980342
```

In [63]:
```
np.sqrt(mean_squared_error(y_test,y_pred))
```

Out[63]:
```
4.387285229095364
```

In [ ]:
```

```

Practical No. 05

Data Analytics II

1. Implement logistic regression using Python/R to perform classification on

Social_Network_Ads.csv dataset. 2. Compute Confusion matrix to find TP, FP, TN, FN, Accuracy, Error rate, Precision, Recall on the given dataset.

In [14]:
```
import pandas as pd
import numpy as np

from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import confusion_matrix, accuracy_score, precision_score, recall_score
```

In [15]:
```
df = pd.read_csv("/home/kartik/Documents/Python Notebooks/Social_Network_Ads.csv")
```

In [16]:
```
df['Gender'].replace({"Male":0,"Female":1},inplace = True)
df
```

Out[16]:

	User ID	Gender	Age	EstimatedSalary	Purchased
0	15624510	0	19	19000	0
1	15810944	0	35	20000	0
2	15668575	1	26	43000	0
3	15603246	1	27	57000	0
4	15804002	0	19	76000	0
...	...	...	...	...	...
395	15691863	1	46	41000	1
396	15706071	0	51	23000	1
397	15654296	1	50	20000	1
398	15755018	0	36	33000	0
399	15594041	1	49	36000	1

400 rows × 5 columns

In [25]:
```
df.columns
```

Out[25]:
```
Index(['User ID', 'Gender', 'Age', 'EstimatedSalary', 'Purchased'], dtype='object')
```

In [24]:
```
x = df[['User ID', 'Gender', 'Age', 'EstimatedSalary']]
y = df[['Purchased']]
```

In [23]:
```
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.25,random_state=29)
```

In [27]:
```
model = LogisticRegression()
model.fit (x_train,y_train)
```

```
/home/kartik/anaconda3/lib/python3.11/site-packages/sklearn/utils/validation.py:1143: DataConversionWarning: A c
olumn-vector y was passed when a 1d array was expected. Please change the shape of y to (n_samples, ), for examp
le using ravel().
  y = column_or_1d(y, warn=True)
```

Out[27]:
```
▾ LogisticRegression

LogisticRegression()
```

In [28]:
```
y_pred = model.predict(x_test)
```

In [29]:
```
y_pred
```

Out[29]:
```
array([0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0,
       1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0,
       0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 1, 1, 0, 0,
       0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0])
```

In [30]:
```
model.score(x_train,y_train)
```

Out[30]:
```
0.7833333333333333
```

In [31]:
```
model.score(x,y)
```

Out[31]:
```
0.785
```

In [32]:
```
cm = confusion_matrix(y_test,y_pred)
cm
```

Out[32]:
```
array([[64,  5],
       [16, 15]])
```

In [33]:
```
tn, fp, fn, tp = confusion_matrix(y_test,y_pred).ravel()
```

In [34]:
```
print (tn, fp, fn, tp)
```

```
64 5 16 15
```

In [35]:
```
a = accuracy_score(y_test,y_pred)
a
```

Out[35]:
```
0.79
```

In [36]:
```
e = 1 - a
e
```

Out[36]:
```
0.20999999999999996
```

In [39]:
```
precision_score(y_test,y_pred)
```

Out[39]:
```
0.75
```

In [40]:
```
recall_score(y_test,y_pred)
```

Out[40]:
```
0.4838709677419355
```

In [ ]:
```

```

Practical No. 06

Data Analytics III

1. Implement Simple Naïve Bayes classification algorithm using Python/R on iris.csv

dataset. 2. Compute Confusion matrix to find TP, FP, TN, FN, Accuracy, Error rate, Precision, Recall on the given dataset.

In [4]:
```
import pandas as pd
import numpy as np

from sklearn.preprocessing import MinMaxScaler
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix, accuracy_score, precision_score, recall_score
from sklearn.naive_bayes import GaussianNB
```

In [5]:
```
df = pd.read_csv("/home/kartik/Documents/Python Notebooks/Iris.csv")
df.head()
```

Out[5]:

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
0	1	5.1	3.5	1.4	0.2	Iris-setosa
1	2	4.9	3.0	1.4	0.2	Iris-setosa
2	3	4.7	3.2	1.3	0.2	Iris-setosa
3	4	4.6	3.1	1.5	0.2	Iris-setosa
4	5	5.0	3.6	1.4	0.2	Iris-setosa

In [6]:
```
x = df.drop(['Species'],axis = 1)
y = df['Species']
```

In [21]:
```
scaler = MinMaxScaler()
x_scaled = scaler.fit_transform(x)
x_scaled
```

Out[21]:
```
array([[0.        , 0.22222222, 0.625     , 0.06779661, 0.04166667],
       [0.00671141, 0.16666667, 0.41666667, 0.06779661, 0.04166667],
       [0.01342282, 0.11111111, 0.5       , 0.05084746, 0.04166667],
       [0.02013423, 0.08333333, 0.45833333, 0.08474576, 0.04166667],
       [0.02684564, 0.19444444, 0.66666667, 0.06779661, 0.04166667],
       [0.03355705, 0.30555556, 0.79166667, 0.11864407, 0.125     ],
       [0.04026846, 0.08333333, 0.58333333, 0.06779661, 0.08333333],
       [0.04697987, 0.19444444, 0.58333333, 0.08474576, 0.04166667],
       [0.05369128, 0.02777778, 0.375     , 0.06779661, 0.04166667],
       [0.06040268, 0.16666667, 0.45833333, 0.08474576, 0.        ],
       [0.06711409, 0.30555556, 0.70833333, 0.08474576, 0.04166667],
       [0.0738255 , 0.13888889, 0.58333333, 0.10169492, 0.04166667],
       [0.08053691, 0.13888889, 0.41666667, 0.06779661, 0.        ],
       [0.08724832, 0.        , 0.41666667, 0.01694915, 0.        ],
       [0.09395973, 0.41666667, 0.83333333, 0.03389831, 0.04166667],
       [0.10067114, 0.38888889, 1.        , 0.08474576, 0.125     ],
       [0.10738255, 0.30555556, 0.79166667, 0.05084746, 0.125     ],
       [0.11409396, 0.22222222, 0.625     , 0.06779661, 0.08333333],
       [0.12080537, 0.38888889, 0.75      , 0.11864407, 0.08333333],
       [0.12751678, 0.22222222, 0.75      , 0.08474576, 0.08333333],
       [0.13422819, 0.30555556, 0.58333333, 0.11864407, 0.04166667],
       [0.1409396 , 0.22222222, 0.70833333, 0.08474576, 0.125     ],
       [0.14765101, 0.08333333, 0.66666667, 0.        , 0.04166667],
       [0.15436242, 0.22222222, 0.54166667, 0.11864407, 0.16666667],
       [0.16107383, 0.13888889, 0.58333333, 0.15254237, 0.04166667],
       [0.16778523, 0.19444444, 0.41666667, 0.10169492, 0.04166667],
       [0.17449664, 0.19444444, 0.58333333, 0.10169492, 0.125     ],
       [0.18120805, 0.25      , 0.625     , 0.08474576, 0.04166667],
       [0.18791946, 0.25      , 0.58333333, 0.06779661, 0.04166667],
       [0.19463087, 0.11111111, 0.5       , 0.10169492, 0.04166667],
       [0.20134228, 0.13888889, 0.45833333, 0.10169492, 0.04166667],
       [0.20805369, 0.30555556, 0.58333333, 0.08474576, 0.125     ],
       [0.2147651 , 0.25      , 0.875     , 0.08474576, 0.        ],
       [0.22147651, 0.33333333, 0.91666667, 0.06779661, 0.04166667],
       [0.22818792, 0.16666667, 0.45833333, 0.08474576, 0.        ],
       [0.23489933, 0.19444444, 0.5       , 0.03389831, 0.04166667],
       [0.24161074, 0.33333333, 0.625     , 0.05084746, 0.04166667],
       [0.24832215, 0.16666667, 0.45833333, 0.08474576, 0.        ],
       [0.25503356, 0.02777778, 0.41666667, 0.05084746, 0.04166667],
```

```
[0.26174497, 0.22222222, 0.58333333, 0.08474576, 0.04166667],
[0.26845638, 0.19444444, 0.625     , 0.05084746, 0.08333333],
[0.27516779, 0.05555556, 0.125     , 0.05084746, 0.08333333],
[0.28187919, 0.02777778, 0.5       , 0.05084746, 0.04166667],
[0.2885906 , 0.19444444, 0.625     , 0.10169492, 0.20833333],
[0.29530201, 0.22222222, 0.75      , 0.15254237, 0.125     ],
[0.30201342, 0.13888889, 0.41666667, 0.06779661, 0.08333333],
[0.30872483, 0.22222222, 0.75      , 0.10169492, 0.04166667],
[0.31543624, 0.08333333, 0.5       , 0.06779661, 0.04166667],
[0.32214765, 0.27777778, 0.70833333, 0.08474576, 0.04166667],
[0.32885906, 0.19444444, 0.54166667, 0.06779661, 0.04166667],
[0.33557047, 0.75      , 0.5       , 0.62711864, 0.54166667],
[0.34228188, 0.58333333, 0.5       , 0.59322034, 0.58333333],
[0.34899329, 0.72222222, 0.45833333, 0.66101695, 0.58333333],
[0.3557047 , 0.33333333, 0.125     , 0.50847458, 0.5       ],
[0.36241611, 0.61111111, 0.33333333, 0.61016949, 0.58333333],
[0.36912752, 0.38888889, 0.33333333, 0.59322034, 0.5       ],
[0.37583893, 0.55555556, 0.54166667, 0.62711864, 0.625     ],
[0.38255034, 0.16666667, 0.16666667, 0.38983051, 0.375     ],
[0.38926174, 0.63888889, 0.375     , 0.61016949, 0.5       ],
[0.39597315, 0.25      , 0.29166667, 0.49152542, 0.54166667],
[0.40268456, 0.19444444, 0.        , 0.42372881, 0.375     ],
[0.40939597, 0.44444444, 0.41666667, 0.54237288, 0.58333333],
[0.41610738, 0.47222222, 0.08333333, 0.50847458, 0.375     ],
[0.42281879, 0.5       , 0.375     , 0.62711864, 0.54166667],
[0.4295302 , 0.36111111, 0.375     , 0.44067797, 0.5       ],
[0.43624161, 0.66666667, 0.45833333, 0.57627119, 0.54166667],
[0.44295302, 0.36111111, 0.41666667, 0.59322034, 0.58333333],
[0.44966443, 0.41666667, 0.29166667, 0.52542373, 0.375     ],
[0.45637584, 0.52777778, 0.08333333, 0.59322034, 0.58333333],
[0.46308725, 0.36111111, 0.20833333, 0.49152542, 0.41666667],
[0.46979866, 0.44444444, 0.5       , 0.6440678 , 0.70833333],
[0.47651007, 0.5       , 0.33333333, 0.50847458, 0.5       ],
[0.48322148, 0.55555556, 0.20833333, 0.66101695, 0.58333333],
[0.48993289, 0.5       , 0.33333333, 0.62711864, 0.45833333],
[0.4966443 , 0.58333333, 0.375     , 0.55932203, 0.5       ],
[0.5033557 , 0.63888889, 0.41666667, 0.57627119, 0.54166667],
[0.51006711, 0.69444444, 0.33333333, 0.6440678 , 0.54166667],
[0.51677852, 0.66666667, 0.41666667, 0.6779661 , 0.66666667],
[0.52348993, 0.47222222, 0.375     , 0.59322034, 0.58333333],
[0.53020134, 0.38888889, 0.25      , 0.42372881, 0.375     ],
[0.53691275, 0.33333333, 0.16666667, 0.47457627, 0.41666667],
[0.54362416, 0.33333333, 0.16666667, 0.45762712, 0.375     ],
[0.55033557, 0.41666667, 0.29166667, 0.49152542, 0.45833333],
[0.55704698, 0.47222222, 0.29166667, 0.69491525, 0.625     ],
[0.56375839, 0.30555556, 0.41666667, 0.59322034, 0.58333333],
[0.5704698 , 0.47222222, 0.58333333, 0.59322034, 0.625     ],
[0.57718121, 0.66666667, 0.45833333, 0.62711864, 0.58333333],
[0.58389262, 0.55555556, 0.125     , 0.57627119, 0.5       ],
[0.59060403, 0.36111111, 0.41666667, 0.52542373, 0.5       ],
[0.59731544, 0.33333333, 0.20833333, 0.50847458, 0.5       ],
[0.60402685, 0.33333333, 0.25      , 0.57627119, 0.45833333],
[0.61073826, 0.5       , 0.41666667, 0.61016949, 0.54166667],
[0.61744966, 0.41666667, 0.25      , 0.50847458, 0.45833333],
[0.62416107, 0.19444444, 0.125     , 0.38983051, 0.375     ],
[0.63087248, 0.36111111, 0.29166667, 0.54237288, 0.5       ],
[0.63758389, 0.38888889, 0.41666667, 0.54237288, 0.45833333],
[0.6442953 , 0.38888889, 0.375     , 0.54237288, 0.5       ],
[0.65100671, 0.52777778, 0.375     , 0.55932203, 0.5       ],
[0.65771812, 0.22222222, 0.20833333, 0.33898305, 0.41666667],
[0.66442953, 0.38888889, 0.33333333, 0.52542373, 0.5       ],
[0.67114094, 0.55555556, 0.54166667, 0.84745763, 1.        ],
[0.67785235, 0.41666667, 0.29166667, 0.69491525, 0.75      ],
[0.68456376, 0.77777778, 0.41666667, 0.83050847, 0.83333333],
[0.69127517, 0.55555556, 0.375     , 0.77966102, 0.70833333],
[0.69798658, 0.61111111, 0.41666667, 0.81355932, 0.875     ],
[0.70469799, 0.91666667, 0.41666667, 0.94915254, 0.83333333],
[0.7114094 , 0.16666667, 0.20833333, 0.59322034, 0.66666667],
[0.71812081, 0.83333333, 0.375     , 0.89830508, 0.70833333],
[0.72483221, 0.66666667, 0.20833333, 0.81355932, 0.70833333],
[0.73154362, 0.80555556, 0.66666667, 0.86440678, 1.        ],
[0.73825503, 0.61111111, 0.5       , 0.69491525, 0.79166667],
[0.74496644, 0.58333333, 0.29166667, 0.72881356, 0.75      ],
[0.75167785, 0.69444444, 0.41666667, 0.76271186, 0.83333333],
[0.75838926, 0.38888889, 0.20833333, 0.6779661 , 0.79166667],
[0.76510067, 0.41666667, 0.33333333, 0.69491525, 0.95833333],
[0.77181208, 0.58333333, 0.5       , 0.72881356, 0.91666667],
[0.77852349, 0.61111111, 0.41666667, 0.76271186, 0.70833333],
[0.7852349 , 0.94444444, 0.75      , 0.96610169, 0.875     ],
[0.79194631, 0.94444444, 0.25      , 1.        , 0.91666667],
[0.79865772, 0.47222222, 0.08333333, 0.6779661 , 0.58333333],
[0.80536913, 0.72222222, 0.5       , 0.79661017, 0.91666667],
[0.81208054, 0.36111111, 0.33333333, 0.66101695, 0.79166667],
```

```
[0.81879195, 0.94444444, 0.33333333, 0.96610169, 0.79166667],
[0.82550336, 0.55555556, 0.29166667, 0.66101695, 0.70833333],
[0.83221477, 0.66666667, 0.54166667, 0.79661017, 0.83333333],
[0.83892617, 0.80555556, 0.5       , 0.84745763, 0.70833333],
[0.84563758, 0.52777778, 0.33333333, 0.6440678 , 0.70833333],
[0.85234899, 0.5       , 0.41666667, 0.66101695, 0.70833333],
[0.8590604 , 0.58333333, 0.33333333, 0.77966102, 0.83333333],
[0.86577181, 0.80555556, 0.41666667, 0.81355932, 0.625     ],
[0.87248322, 0.86111111, 0.33333333, 0.86440678, 0.75      ],
[0.87919463, 1.        , 0.75      , 0.91525424, 0.79166667],
[0.88590604, 0.58333333, 0.33333333, 0.77966102, 0.875     ],
[0.89261745, 0.55555556, 0.33333333, 0.69491525, 0.58333333],
[0.89932886, 0.5       , 0.25      , 0.77966102, 0.54166667],
[0.90604027, 0.94444444, 0.41666667, 0.86440678, 0.91666667],
[0.91275168, 0.55555556, 0.58333333, 0.77966102, 0.95833333],
[0.91946309, 0.58333333, 0.45833333, 0.76271186, 0.70833333],
[0.9261745 , 0.47222222, 0.41666667, 0.6440678 , 0.70833333],
[0.93288591, 0.72222222, 0.45833333, 0.74576271, 0.83333333],
[0.93959732, 0.66666667, 0.45833333, 0.77966102, 0.95833333],
[0.94630872, 0.72222222, 0.45833333, 0.69491525, 0.91666667],
[0.95302013, 0.41666667, 0.29166667, 0.69491525, 0.75      ],
[0.95973154, 0.69444444, 0.5       , 0.83050847, 0.91666667],
[0.96644295, 0.66666667, 0.54166667, 0.79661017, 1.        ],
[0.97315436, 0.66666667, 0.41666667, 0.71186441, 0.91666667],
[0.97986577, 0.55555556, 0.20833333, 0.6779661 , 0.75      ],
[0.98657718, 0.61111111, 0.41666667, 0.71186441, 0.79166667],
[0.99328859, 0.52777778, 0.58333333, 0.74576271, 0.91666667],
[1.        , 0.44444444, 0.41666667, 0.69491525, 0.70833333]])
```

In [8]:
```
x_train,x_test,y_train,y_test = train_test_split(x_scaled,y,test_size = 0.2, random_state = 43)
```

In [9]:
```
gnb = GaussianNB()
gnb.fit(x_train,y_train)
y_pred = gnb.predict(x_test)
```

In [10]:
```
cm = confusion_matrix(y_test,y_pred)
cm
```

Out[10]:
```
array([[13,  0,  0],
       [ 0,  8,  0],
       [ 0,  0,  9]])
```

In [11]:
```
tn = confusion_matrix(y_test,y_pred).ravel()
```

In [12]:
```
fp = confusion_matrix(y_test,y_pred).ravel()
```

In [13]:
```
fn= confusion_matrix(y_test,y_pred).ravel()
```

In [14]:
```
tp = confusion_matrix(y_test,y_pred).ravel()
```

In [15]:
```
print (tn, fp, fn, tp)
```

```
[13  0  0  0  8  0  0  0  9] [13  0  0  0  8  0  0  0  9] [13  0  0  0  8  0  0  0  9] [13  0  0  0  8  0  0  0
9]
```

In [16]:
```
a = accuracy_score(y_test,y_pred)
a
```

Out[16]: 1.0

In [17]:
```
e = 1 - a
e
```

Out[17]: 0.0

In [18]:
```
precision_score(y_test,y_pred,average='micro')
```

Out[18]: 1.0

In [19]:
```
recall_score(y_test,y_pred,average='micro')
```

Out[19]: 1.0

Practical No. 07

Text Analytics

1. Extract Sample document and apply following document preprocessing methods:

Tokenization, POS Tagging, stop words removal, Stemming and Lemmatization. 2. Create representation of document by calculating Term Frequency and Inverse Document Frequency.

In [31]:
```
import nltk
import re
nltk.download('punkt')
nltk.download('stopwords')
nltk.download('wordnet')
nltk.download('averaged_perceptron_tagger')
nltk.download('omw-1.4')
```

```
[nltk_data] Downloading package punkt to /home/kartik/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
[nltk_data] Downloading package stopwords to /home/kartik/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
[nltk_data] Downloading package wordnet to /home/kartik/nltk_data...
[nltk_data]   Package wordnet is already up-to-date!
[nltk_data] Downloading package averaged_perceptron_tagger to
[nltk_data]     /home/kartik/nltk_data...
[nltk_data]   Package averaged_perceptron_tagger is already up-to-
[nltk_data]       date!
[nltk_data] Downloading package omw-1.4 to /home/kartik/nltk_data...
[nltk_data]   Package omw-1.4 is already up-to-date!
```

Out[31]: True

In [32]:
```
text = "Tokenization is the first step in text analytics."
```

In [33]:
```
from nltk.tokenize import sent_tokenize
tokenized_text = sent_tokenize(text)
print(tokenized_text)
```

```
['Tokenization is the first step in text analytics.']
```

In [34]:
```
from nltk.tokenize import word_tokenize
tokenized_word = word_tokenize(text)
print(tokenized_word)
```

```
['Tokenization', 'is', 'the', 'first', 'step', 'in', 'text', 'analytics', '.']
```

In [35]:
```
from nltk.corpus import stopwords
stop_words = set(stopwords.words("english"))
print(stop_words)
```

```
{"haven't", 'those', 'hers', "that'll", 'were', 'won', "don't", 'you', 'did', 'shan', 'myself', 'as', 'over', "s
hould've", 'his', "wouldn't", 'of', 'hadn', 'then', 'no', 'has', 'which', 'was', 'does', 'weren', 'both', "you'v
e", 'yourself', 'we', "needn't", 're', 'when', 'some', 'again', "won't", 'on', "hasn't", 'itself', "didn't", "sh
e's", 'up', 'a', "hadn't", 'against', 'during', "mightn't", 'shouldn', "you'll", 'why', 't', 'who', 'such', 'mus
tn', 'being', 'through', 'can', 'from', 'too', 'and', 've', "shan't", 'mightn', 'wouldn', 'this', 'each', 'whom'
, "shouldn't", 'the', 'should', 'now', 'its', 'do', 'ain', 'he', 'until', 'further', 'will', 'these', 'into', 'w
hat', 'it', 'about', 'have', 'is', 'same', 'needn', 'how', 'your', "wasn't", "aren't", 'doesn', 'yours', 'once',
'doing', 'while', 'himself', 'ourselves', 'if', 'all', 'hasn', 'above', 'before', 'most', 'o', 'any', 'just', 'a
re', 'nor', 'ma', 'between', 'in', 'didn', 'had', 'other', 'y', 'to', 'been', 'they', 'after', 'be', 'don', "wer
en't", "isn't", 'down', 'our', 'under', 'my', 'isn', 'll', 'herself', "doesn't", 'me', 'at', 'by', "you'd", 'cou
ldn', 'where', 'haven', 'but', 'them', 'that', 'theirs', 'having', 'i', 'below', 'd', 'not', "it's", "mustn't",
'yourselves', 'am', 'wasn', 's', 'ours', 'she', 'only', 'him', 'themselves', 'few', "couldn't", 'very', 'for', "
you're", 'so', 'her', 'there', 'here', 'own', 'or', 'm', 'because', 'out', 'off', 'aren', 'more', 'with', 'their
', 'than', 'an'}
```

In [36]:
```
text = "How to remove stop words with NLTK library in Python?"
text = re.sub('[^a-zA-Z]',' ',text)
tokens = word_tokenize(text.lower())
filtered_text = []

for w in tokens:
    if w not in stop_words:
        filtered_text.append(w)

print("Tokenized Sentence :",tokens)
print("Filtered Sentence :",filtered_text)
```

```
Tokenized Sentence : ['howtoremovestopwordswithnltklibraryinpython']
Filtered Sentence : ['howtoremovestopwordswithnltklibraryinpython']
```

In [37]:
```
from nltk.stem import PorterStemmer
e_words = ["wait", "waiting", "waited", "waits"]
ps = PorterStemmer()
for w in e_words:
    rootWord = ps.stem(w)
print(rootWord)
```

```
wait
```

In [38]:
```
from nltk.stem import WordNetLemmatizer
wordnet_lemmatizer = WordNetLemmatizer()
text = "studies studying cries cry"
tokenization = nltk.word_tokenize(text)
for w in tokenization:
    print("Lemma for {} is {}".format(w, wordnet_lemmatizer.lemmatize(w)))
```

```
Lemma for studies is study
Lemma for studying is studying
Lemma for cries is cry
Lemma for cry is cry
```

In [39]:
```
from nltk.tokenize import word_tokenize
data = "The pink sweater fit her perfectly"
words = word_tokenize(data)
for word in words:
    print(nltk.pos_tag([word]))
```

```
[('The', 'DT')]
[('pink', 'NN')]
[('sweater', 'NN')]
[('fit', 'NN')]
[('her', 'PRP$')]
[('perfectly', 'RB')]
```

In [40]:
```
import pandas as pd
from sklearn.feature_extraction.text import TfidfVectorizer

d0 = 'Jupiter is the largest Planet'
d1 = 'Mars is the fourth planet from the sun'
string = [d0,d1]
tfidf = TfidfVectorizer()
result = tfidf.fit_transform(string)
print('Word indices:', tfidf.vocabulary_)
print('TF-IDF Values:', result)
```

```
Word indices: {'jupiter': 3, 'is': 2, 'the': 8, 'largest': 4, 'planet': 6, 'mars': 5, 'fourth': 0, 'from': 1, 's
un': 7}
TF-IDF Values:   (0, 6) 0.3793034928087496
  (0, 4)        0.5330978245262535
  (0, 8)        0.3793034928087496
  (0, 2)        0.3793034928087496
  (0, 3)        0.5330978245262535
  (1, 7)        0.37695708675831013
  (1, 1)        0.37695708675831013
  (1, 0)        0.37695708675831013
  (1, 5)        0.37695708675831013
  (1, 6)        0.2682080718928097
  (1, 8)        0.5364161437856194
  (1, 2)        0.2682080718928097
```

Practical No, 08

Data Visualization I

1. Use the inbuilt dataset 'titanic'. The dataset contains 891 rows and contains information

about the passengers who boarded the unfortunate Titanic ship. Use the Seaborn library to see if we can find any patterns in the data. 2. Write a code to check how the price of the ticket (column name: 'fare') for each passenger is distributed by plotting a histogram.

In [1]:
```
import numpy as np
import pandas as pd
import seaborn as sb
import matplotlib.pyplot as plt
```

In [2]:
```
df = sb.load_dataset('titanic')
df.head()
```

Out[2]:

	survived	pclass	sex	age	sibsp	parch	fare	embarked	class	who	adult_male	deck	embark_town	alive	alone
0	0	3	male	22.0	1	0	7.2500	S	Third	man	True	NaN	Southampton	no	False
1	1	1	female	38.0	1	0	71.2833	C	First	woman	False	C	Cherbourg	yes	False
2	1	3	female	26.0	0	0	7.9250	S	Third	woman	False	NaN	Southampton	yes	True
3	1	1	female	35.0	1	0	53.1000	S	First	woman	False	C	Southampton	yes	False
4	0	3	male	35.0	0	0	8.0500	S	Third	man	True	NaN	Southampton	no	True

In [3]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 15 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   survived     891 non-null    int64
 1   pclass       891 non-null    int64
 2   sex          891 non-null    object
 3   age          714 non-null    float64
 4   sibsp        891 non-null    int64
 5   parch        891 non-null    int64
 6   fare         891 non-null    float64
 7   embarked     889 non-null    object
 8   class        891 non-null    category
 9   who          891 non-null    object
 10  adult_male   891 non-null    bool
 11  deck         203 non-null    category
 12  embark_town  889 non-null    object
 13  alive        891 non-null    object
 14  alone        891 non-null    bool
dtypes: bool(2), category(2), float64(2), int64(4), object(5)
memory usage: 80.7+ KB
```

In [4]:
```
print("Missing Values")
print(df.isnull().sum())
```

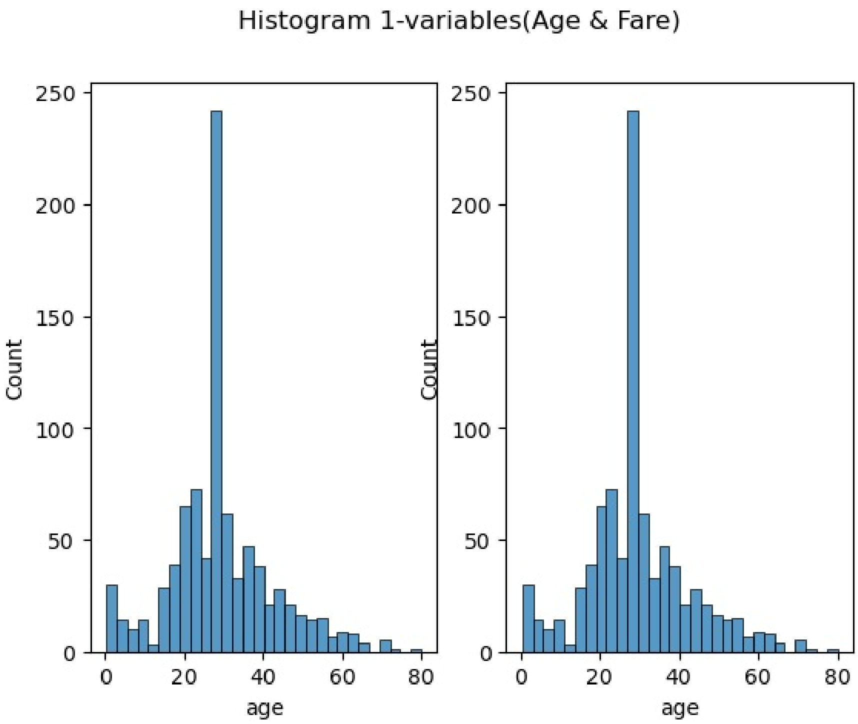
```
Missing Values
survived         0
pclass           0
sex              0
age            177
sibsp            0
parch            0
fare             0
embarked         2
class            0
who              0
adult_male       0
deck           688
embark_town      2
alive            0
alone            0
dtype: int64
```

In [5]:
```
df['age'].fillna(df['age'].median(), inplace=True)
df.isnull().sum()
```

Out[5]:
```
survived         0
pclass           0
sex              0
age              0
sibsp            0
parch            0
fare             0
embarked         2
class            0
who              0
adult_male       0
deck           688
embark_town      2
alive            0
alone            0
dtype: int64
```

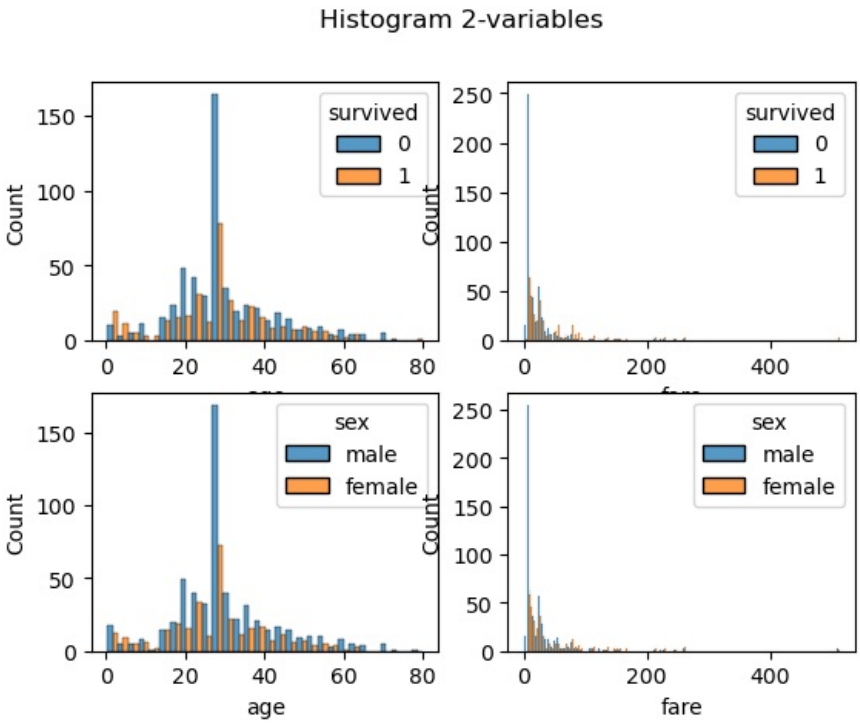
In [6]:
```
fig, axes = plt.subplots(1,2)
fig.suptitle('Histogram 1-variables(Age & Fare)')
sb.histplot(data = df, x = 'age', ax = axes[0])
sb.histplot(data = df, x = 'age', ax = axes[1])
plt.show()
```

```
/home/kartik/anaconda3/lib/python3.11/site-packages/seaborn/_oldcore.py:1119: FutureWarning: use_inf_as_na optio
n is deprecated and will be removed in a future version. Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
/home/kartik/anaconda3/lib/python3.11/site-packages/seaborn/_oldcore.py:1119: FutureWarning: use_inf_as_na optio
n is deprecated and will be removed in a future version. Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```



In [7]:
```
fig, axes = plt.subplots(2,2)
fig.suptitle('Histogram 2-variables')
sb.histplot(data = df, x = 'age', hue = 'survived', multiple='dodge', ax = axes[0][0])
sb.histplot(data = df, x = 'fare', hue = 'survived', multiple='dodge', ax = axes[0][1])
sb.histplot(data = df, x = 'age', hue = 'sex', multiple='dodge', ax = axes[1][0])
sb.histplot(data = df, x = 'fare', hue = 'sex', multiple='dodge', ax = axes[1][1])
plt.show()
```

```
/home/kartik/anaconda3/lib/python3.11/site-packages/seaborn/_oldcore.py:1119: FutureWarning: use_inf_as_na optio
n is deprecated and will be removed in a future version. Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
/home/kartik/anaconda3/lib/python3.11/site-packages/seaborn/_oldcore.py:1119: FutureWarning: use_inf_as_na optio
n is deprecated and will be removed in a future version. Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
/home/kartik/anaconda3/lib/python3.11/site-packages/seaborn/_oldcore.py:1119: FutureWarning: use_inf_as_na optio
n is deprecated and will be removed in a future version. Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
/home/kartik/anaconda3/lib/python3.11/site-packages/seaborn/_oldcore.py:1119: FutureWarning: use_inf_as_na optio
n is deprecated and will be removed in a future version. Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```



Practical No. 09

Data Visualization II

1. Use the inbuilt dataset 'titanic' as used in the above problem. Plot a box plot for

distribution of age with respect to each gender along with the information about whether they survived or not. (Column names : 'sex' and 'age') 2. Write observations on the inference from the above statistics.

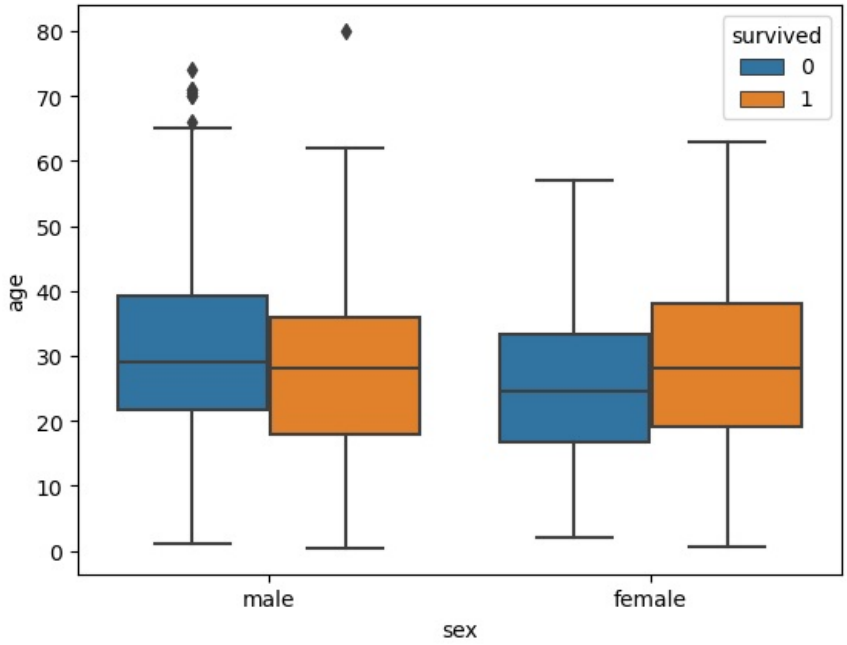
In [1]:
```
import seaborn as sb
ds = sb.load_dataset('titanic')
ds.head()
```

Out[1]:

	survived	pclass	sex	age	sibsp	parch	fare	embarked	class	who	adult_male	deck	embark_town	alive	alone
0	0	3	male	22.0	1	0	7.2500	S	Third	man	True	NaN	Southampton	no	False
1	1	1	female	38.0	1	0	71.2833	C	First	woman	False	C	Cherbourg	yes	False
2	1	3	female	26.0	0	0	7.9250	S	Third	woman	False	NaN	Southampton	yes	True
3	1	1	female	35.0	1	0	53.1000	S	First	woman	False	C	Southampton	yes	False
4	0	3	male	35.0	0	0	8.0500	S	Third	man	True	NaN	Southampton	no	True

In [6]:
```
sb.boxplot(x='sex',y='age',data=ds, hue='survived')
```

Out[6]:
```
<Axes: xlabel='sex', ylabel='age'>
```



In [ ]:
```
```

Practical No. 10

Data Visualization III Download the Iris flower dataset or any other dataset into a DataFrame. (e.g., https://archive.ics.uci.edu/ml/datasets/Iris). Scan the dataset and give the inference as:

1. List down the features and their types (e.g., numeric, nominal) available in the dataset.
2. Create a histogram for each feature in the dataset to illustrate the feature distributions.
3. Create a box plot for each feature in the dataset.
4. Compare distributions and identify outliers.

In [6]:
```
import seaborn as sb
ds = sb.load_dataset('iris')
ds.head()
```

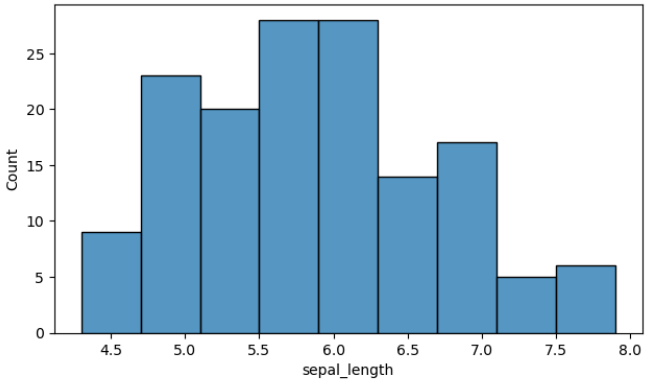
Out[6]:

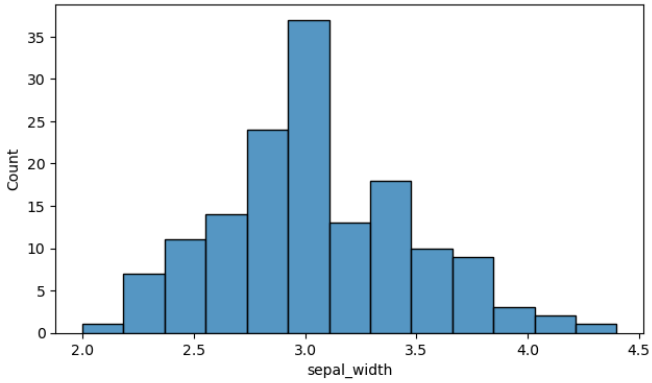
	sepal_length	sepal_width	petal_length	petal_width	species
0	5.1	3.5	1.4	0.2	setosa
1	4.9	3.0	1.4	0.2	setosa
2	4.7	3.2	1.3	0.2	setosa
3	4.6	3.1	1.5	0.2	setosa
4	5.0	3.6	1.4	0.2	setosa

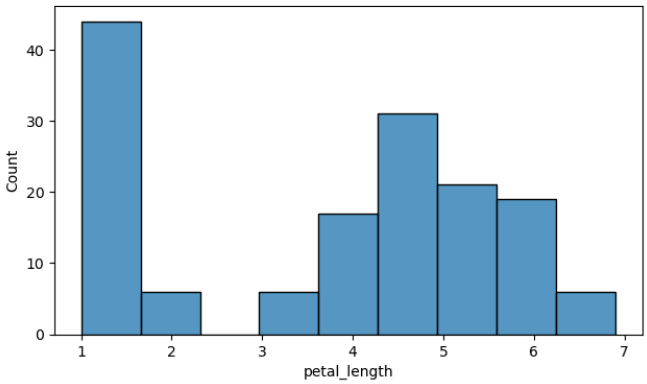
In [13]:
```
import matplotlib.pyplot as plt
fig, axes = plt.subplots(2,2,figsize = (16,9))
sb.histplot(ds['sepal_length'], ax = axes[0,0])
sb.histplot(ds['sepal_width'], ax = axes[0,1])
sb.histplot(ds['petal_length'], ax = axes[1,0])
sb.histplot(ds['petal_width'], ax = axes[1,1])
```

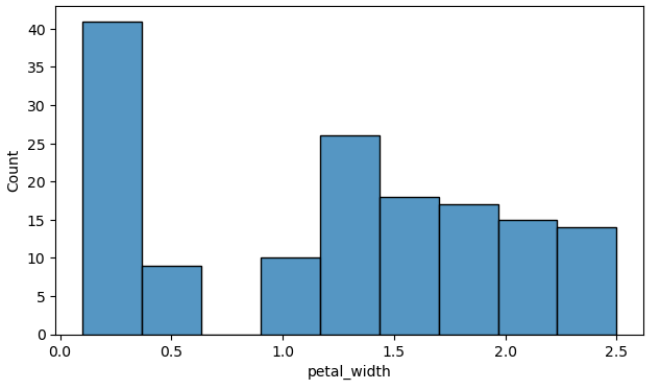
```
/home/kartik/anaconda3/lib/python3.11/site-packages/seaborn/_oldcore.py:1119: FutureWarning: u
se_inf_as_na option is deprecated and will be removed in a future version. Convert inf values
to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
/home/kartik/anaconda3/lib/python3.11/site-packages/seaborn/_oldcore.py:1119: FutureWarning: u
se_inf_as_na option is deprecated and will be removed in a future version. Convert inf values
to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
/home/kartik/anaconda3/lib/python3.11/site-packages/seaborn/_oldcore.py:1119: FutureWarning: u
se_inf_as_na option is deprecated and will be removed in a future version. Convert inf values
to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
/home/kartik/anaconda3/lib/python3.11/site-packages/seaborn/_oldcore.py:1119: FutureWarning: u
se_inf_as_na option is deprecated and will be removed in a future version. Convert inf values
to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```

Out[13]:
```
<Axes: xlabel='petal_width', ylabel='Count'>
```



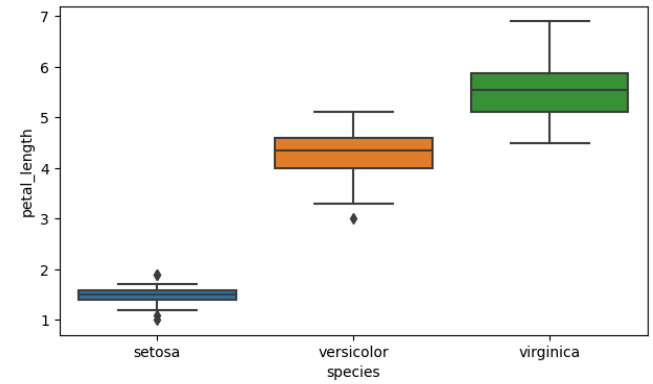


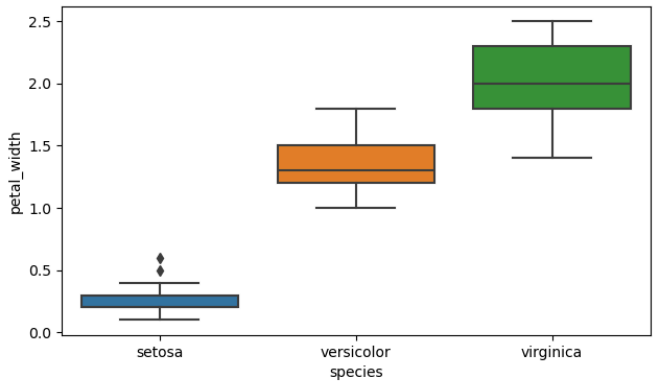


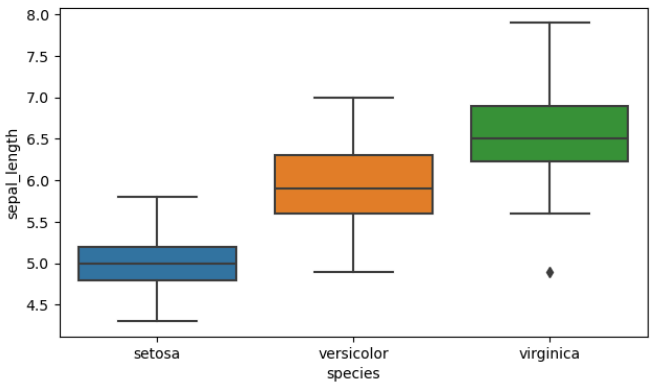


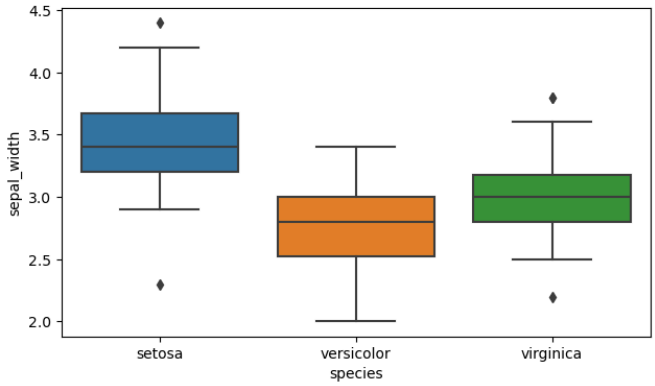
In [17]:
```
import matplotlib.pyplot as plt
fig, axes = plt.subplots(2,2,figsize=(16,9))
sb.boxplot(x='species',y='petal_length',data = ds, ax=axes[0,0])
sb.boxplot(x='species',y='petal_width',data = ds, ax=axes[0,1])
sb.boxplot(x='species',y='sepal_length',data = ds, ax=axes[1,0])
sb.boxplot(x='species',y='sepal_width',data = ds, ax=axes[1,1])
```

Out[17]: <Axes: xlabel='species', ylabel='sepal_width'>









Practical No. 14

Use the following dataset "covid_vaccine_statewise.csv" and Perform the following data analytics.

1. Describe the dataset.
2. No. of persons by state vaccinated for first dose.
3. No. of persons by state vaccinated for second dose.
4. No. of males vaccinated.
5. No. of females vaccinated.

In [1]:
```
import pandas as pd
```

In [4]:
```
df = pd.read_csv("covid_vaccine_statewise.csv")
```

In [6]:
```
df.head()
```

Out[6]:

	Updated On	State	Total Doses Administered	Sessions	Sites	First Dose Administered	Second Dose Administered	Male (Doses Administered)	Female (Doses Administered)	Transgender (Doses Administered)	..
0	16/01/2021	India	48276.0	3455.0	2957.0	48276.0	0.0	NaN	NaN	NaN	..
1	17/01/2021	India	58604.0	8532.0	4954.0	58604.0	0.0	NaN	NaN	NaN	..
2	18/01/2021	India	99449.0	13611.0	6583.0	99449.0	0.0	NaN	NaN	NaN	..
3	19/01/2021	India	195525.0	17855.0	7951.0	195525.0	0.0	NaN	NaN	NaN	..
4	20/01/2021	India	251280.0	25472.0	10504.0	251280.0	0.0	NaN	NaN	NaN	..

5 rows × 24 columns

In [7]:
```
df.shape
```

Out[7]:
```
(7845, 24)
```

In [8]:
```
df.describe()
```

Out[8]:

	Total Doses Administered	Sessions	Sites	First Dose Administered	Second Dose Administered	Male (Doses Administered)	Female (Doses Administered)	Transgender (Doses Administered)	Admin
count	7.621000e+03	7.621000e+03	7621.000000	7.621000e+03	7.621000e+03	7.461000e+03	7.461000e+03	7461.000000	7.621(
mean	9.188171e+06	4.792358e+05	2282.872064	7.414415e+06	1.773755e+06	3.620156e+06	3.168416e+06	1162.978019	1.044(
std	3.746180e+07	1.911511e+06	7275.973730	2.995209e+07	7.570382e+06	1.737938e+07	1.515310e+07	5931.353995	4.452
min	7.000000e+00	0.000000e+00	0.000000	7.000000e+00	0.000000e+00	0.000000e+00	2.000000e+00	0.000000	0.000(
25%	1.356570e+05	6.004000e+03	69.000000	1.166320e+05	1.283100e+04	5.655500e+04	5.210700e+04	8.000000	0.000(
50%	8.182020e+05	4.547000e+04	597.000000	6.614590e+05	1.388180e+05	3.897850e+05	3.342380e+05	113.000000	1.185
75%	6.625243e+06	3.428690e+05	1708.000000	5.387805e+06	1.166434e+06	2.735777e+06	2.561513e+06	800.000000	7.579
max	5.132284e+08	3.501031e+07	73933.000000	4.001504e+08	1.130780e+08	2.701636e+08	2.395186e+08	98275.000000	6.236

8 rows × 22 columns

In [9]:
```
df.describe(include='object')
```

Out[9]:

	Updated On	State
count	7845	7845
unique	213	37
top	16/01/2021	Delhi
freq	37	213

In [10]:
```
avg_firstdose = df["First Dose Administered"].astype("float").mean(axis = 0)
print("Average of First Dose:", avg_firstdose)
```

```
Average of First Dose: 7414415.300354284
```

In [11]:
```
df["First Dose Administered"].fillna(value = avg_firstdose, inplace=True)
df.head()
```

Out[11]:

	Updated On	State	Total Doses Administered	Sessions	Sites	First Dose Administered	Second Dose Administered	Male (Doses Administered)	Female (Doses Administered)	Transgender (Doses Administered)	..
0	16/01/2021	India	48276.0	3455.0	2957.0	48276.0	0.0	NaN	NaN	NaN	..
1	17/01/2021	India	58604.0	8532.0	4954.0	58604.0	0.0	NaN	NaN	NaN	..
2	18/01/2021	India	99449.0	13611.0	6583.0	99449.0	0.0	NaN	NaN	NaN	..
3	19/01/2021	India	195525.0	17855.0	7951.0	195525.0	0.0	NaN	NaN	NaN	..
4	20/01/2021	India	251280.0	25472.0	10504.0	251280.0	0.0	NaN	NaN	NaN	..

5 rows × 24 columns

In [12]:
```
avg_seconddose = df["Second Dose Administered"].astype("float").mean(axis = 0)
print("Average of Second Dose:", avg_seconddose)
```

```
Average of Second Dose: 1773755.2436688098
```

In [13]:
```
df["Second Dose Administered"].fillna(value = avg_seconddose, inplace = True)
df.head()
```

Out[13]:

	Updated On	State	Total Doses Administered	Sessions	Sites	First Dose Administered	Second Dose Administered	Male (Doses Administered)	Female (Doses Administered)	Transgender (Doses Administered)	..
0	16/01/2021	India	48276.0	3455.0	2957.0	48276.0	0.0	NaN	NaN	NaN	..
1	17/01/2021	India	58604.0	8532.0	4954.0	58604.0	0.0	NaN	NaN	NaN	..
2	18/01/2021	India	99449.0	13611.0	6583.0	99449.0	0.0	NaN	NaN	NaN	..
3	19/01/2021	India	195525.0	17855.0	7951.0	195525.0	0.0	NaN	NaN	NaN	..
4	20/01/2021	India	251280.0	25472.0	10504.0	251280.0	0.0	NaN	NaN	NaN	..

5 rows × 24 columns

In [15]:
```
first_dose = df.groupby('State')[['First Dose Administered']].sum()
first_dose.head()
```

Out[15]:

State	First Dose Administered
Andaman and Nicobar Islands	6.091235e+07
Andhra Pradesh	1.277347e+09
Arunachal Pradesh	9.349147e+07
Assam	6.300867e+08
Bihar	1.514989e+09

In [16]:
```
first_dose = df.groupby('State')[['Second Dose Administered']].sum()
first_dose.head()
```

Out[16]:

State	Second Dose Administered
Andaman and Nicobar Islands	1.476109e+07
Andhra Pradesh	3.694601e+08
Arunachal Pradesh	2.257485e+07
Assam	1.414313e+08
Bihar	2.814331e+08

In [17]:
```
male = df["Male(Individuals Vaccinated)"].sum()
print("The Total Number of Male Individuals Vaccinated are :", int(male))
```

```
The Total Number of Male Individuals Vaccinated are : 7138698858
```

In [18]:
```
female = df["Female(Individuals Vaccinated)"].sum()
print("The Total Number of Female Individuals Vaccinated are :", int(female))
```

```
The Total Number of Female Individuals Vaccinated are : 6321628736
```