In [1]:

```
import pandas as pd
import seaborn as sns
import numpy as np
```

In [2]:

```
import matplotlib.pyplot as plt
import warnings
```

In [3]:

```
warnings.filterwarnings("ignore")
```

In [6]:

```
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay
```

In [7]:

```

data_set_name=sns.get_dataset_names()
print(data_set_name)
```

```
['anagrams', 'anscombe', 'attention', 'brain_networks', 'car_crashes', 'di
amonds', 'dots', 'dowjones', 'exercise', 'flights', 'fmri', 'geyser', 'glu
e', 'healthexp', 'iris', 'mpg', 'penguins', 'planets', 'seaice', 'taxis',
'tips', 'titanic', 'anagrams', 'anagrams', 'anscombe', 'anscombe', 'attent
ion', 'attention', 'brain_networks', 'brain_networks', 'car_crashes', 'car
_crashes', 'diamonds', 'diamonds', 'dots', 'dots', 'dowjones', 'dowjones',
'exercise', 'exercise', 'flights', 'flights', 'fmri', 'fmri', 'geyser', 'g
eyser', 'glue', 'glue', 'healthexp', 'healthexp', 'iris', 'iris', 'mpg',
'mpg', 'penguins', 'penguins', 'planets', 'planets', 'seaice', 'seaice',
'taxis', 'taxis', 'tips', 'tips', 'titanic', 'titanic', 'anagrams', 'ansco
mbe', 'attention', 'brain_networks', 'car_crashes', 'diamonds', 'dots', 'd
owjones', 'exercise', 'flights', 'fmri', 'geyser', 'glue', 'healthexp', 'i
ris', 'mpg', 'penguins', 'planets', 'seaice', 'taxis', 'tips', 'titanic']
```

In [9]:

```
df = sns.load_dataset("titanic")
```

In [10]:

```
df.head()
```

Out[10]:

	survived	pclass	sex	age	sibsp	parch	fare	embarked	class	who	adult_ma
0	0	3	male	22.0	1	0	7.2500	S	Third	man	Tru
1	1	1	female	38.0	1	0	71.2833	C	First	woman	Fals
2	1	3	female	26.0	0	0	7.9250	S	Third	woman	Fals
3	1	1	female	35.0	1	0	53.1000	S	First	woman	Fals
4	0	3	male	35.0	0	0	8.0500	S	Third	man	Tru

In [11]:
```
df.tail()
```

Out[11]:

	survived	pclass	sex	age	sibsp	parch	fare	embarked	class	who	adult_n
886	0	2	male	27.0	0	0	13.00	S	Second	man	T
887	1	1	female	19.0	0	0	30.00	S	First	woman	F
888	0	3	female	NaN	1	2	23.45	S	Third	woman	F
889	1	1	male	26.0	0	0	30.00	C	First	man	T
890	0	3	male	32.0	0	0	7.75	Q	Third	man	T

In [13]:
```
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 [15]:
```
df["sex"].value_counts(normalize=True)
```

Out[15]:
```
male      0.647587
female    0.352413
Name: sex, dtype: float64
```

In [16]:
```
df.describe()
```

Out[16]:

	survived	pclass	age	sibsp	parch	fare
count	891.000000	891.000000	714.000000	891.000000	891.000000	891.000000
mean	0.383838	2.308642	29.699118	0.523008	0.381594	32.204208
std	0.486592	0.836071	14.526497	1.102743	0.806057	49.693429
min	0.000000	1.000000	0.420000	0.000000	0.000000	0.000000
25%	0.000000	2.000000	20.125000	0.000000	0.000000	7.910400
50%	0.000000	3.000000	28.000000	0.000000	0.000000	14.454200
75%	1.000000	3.000000	38.000000	1.000000	0.000000	31.000000
max	1.000000	3.000000	80.000000	8.000000	6.000000	512.329200

In [17]:
```
df["deck"].value_counts(normalize=True)
```

Out[17]:
```
C    0.290640
B    0.231527
D    0.162562
E    0.157635
A    0.073892
F    0.064039
G    0.019704
Name: deck, dtype: float64
```

In [18]:
```
df.drop(["deck"], axis=1)
```

Out[18]:

	survived	pclass	sex	age	sibsp	parch	fare	embarked	class	who	adult
0	0	3	male	22.0	1	0	7.2500	S	Third	man	
1	1	1	female	38.0	1	0	71.2833	C	First	woman	
2	1	3	female	26.0	0	0	7.9250	S	Third	woman	
3	1	1	female	35.0	1	0	53.1000	S	First	woman	
4	0	3	male	35.0	0	0	8.0500	S	Third	man	
...	...	...	...	...	...	...	...	...	...	...	
886	0	2	male	27.0	0	0	13.0000	S	Second	man	
887	1	1	female	19.0	0	0	30.0000	S	First	woman	
888	0	3	female	NaN	1	2	23.4500	S	Third	woman	
889	1	1	male	26.0	0	0	30.0000	C	First	man	
890	0	3	male	32.0	0	0	7.7500	Q	Third	man	

891 rows × 14 columns

In [20]:
```
df1 = df.drop(["embarked", "class", "who", "adult_male", "deck", "embark_to
```

In [21]:
```
df1['sex'].mode()[0]
```

Out[21]:
```
'male'
```

In [22]:
```
df1['age'].mode
```

Out[22]:
```
<bound method Series.mode of 0      22.0
1      38.0
2      26.0
3      35.0
4      35.0
       ...
886    27.0
887    19.0
888     NaN
889    26.0
890    32.0
Name: age, Length: 891, dtype: float64>
```

In [23]:
```
df1['age'].mean
```

Out[23]:
```
<bound method NDFrame._add_numeric_operations.<locals>.mean of 0      22.0
1      38.0
2      26.0
3      35.0
4      35.0
       ...
886    27.0
887    19.0
888     NaN
889    26.0
890    32.0
Name: age, Length: 891, dtype: float64>
```

In [24]:
```
df1.loc[:,"sex"].mode()
```

Out[24]:
```
0    male
Name: sex, dtype: object
```

In [25]:
```
df1.min()
```

Out[25]:
```
survived         0
pclass           1
sex         female
age           0.42
sibsp            0
parch            0
fare           0.0
alive           no
dtype: object
```

In [26]:
```
bool_series = pd.notnull(df1["sex"])
```

In [27]:

```
df1
```

Out[27]:

	survived	pclass	sex	age	sibsp	parch	fare	alive
0	0	3	male	22.0	1	0	7.2500	no
1	1	1	female	38.0	1	0	71.2833	yes
2	1	3	female	26.0	0	0	7.9250	yes
3	1	1	female	35.0	1	0	53.1000	yes
4	0	3	male	35.0	0	0	8.0500	no
...	...	...	...	...	...	...	...	...
886	0	2	male	27.0	0	0	13.0000	no
887	1	1	female	19.0	0	0	30.0000	yes
888	0	3	female	NaN	1	2	23.4500	no
889	1	1	male	26.0	0	0	30.0000	yes
890	0	3	male	32.0	0	0	7.7500	no

891 rows × 8 columns

In [28]:

```
df1.fillna(df1['age'].mean,inplace=True)
```

In [29]:

```
#Q2
ip = "a4b4c4d1"

op = ""
i = 0
while i < len(ip):
    char = ip[i]
    count = int(ip[i + 1])
    op += char * count
    i += 2


print("op =", op)
```

```
op = aaaabbbbccccd
```

In [30]:

```
#Q3
from collections import Counter


test_list = [[3, 5, 4],
             [6, 2, 4],
             [1, 3, 6]]


flattened_list = [item for sublist in test_list for item in sublist]

frequency = dict(Counter(flattened_list))


print("The original list:", test_list)
print("The list frequency of elements is:", frequency)
```

```
The original list: [[3, 5, 4], [6, 2, 4], [1, 3, 6]]
The list frequency of elements is: {3: 2, 5: 1, 4: 2, 6: 2, 2: 1, 1: 1}
```

In [32]:

```
#Q4

list1 = [1, 2, 3, 4, 5]
list2 = [4, 5, 3, 7, 8]


common_elements = [element for element in list1 if element in list2]


print("Common elements:", common_elements)
```

```
Common elements: [3, 4, 5]
```

Input list

words = ['Sohan', 'Mohan', 'Rohan']

Extracting the first letter of each word

first_letters = [word[0] for word in words]

Output the result

print("Op:", first_letters)

In [33]:

```
# Q5
words = ['Sohan', 'Mohan', 'Rohan']


first_letters = [word[0] for word in words]


print("Op:", first_letters)

```

```
Op: ['S', 'M', 'R']
```

In [34]:

```
#Q6
from collections import Counter

lst = ['pandas', 'numpy', 'flask', 'python', 'python']


counts = Counter(lst)


duplicates = [item for item, count in counts.items() if count > 1]


print("Op:", duplicates)

```

```
Op: ['python']
```

In [35]:

```
#Q7
my_string = "santosh kawade"

length = len(my_string)


print("OP: count for string is", length)

```

```
OP: count for string is 14
```

In [36]:

```
# Q8
lis = [1, 2, 5, 3, 4, 8, 9, "lis", "a"]


length = len(lis)

print("OP: count for list is", length)

```

```
OP: count for list is 9
```

In [ ]:

In [ ]: