


```
In [32]: import pandas as pd
import seaborn as sns
import numpy as np
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings("ignore")
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay
data_set_name=sns.get_dataset_names()
print(data_set_name)
df = sns.load_dataset("titanic")
```

```
[ 'anagrams', 'anscombe', 'attention', 'brain_networks', 'car_crashes', 'diamonds', 'dots', 'dowjones', 'exercise', 'flights', 'fmri', 'geyser', 'glue', 'healthexp', 'iris', 'mpg', 'penguins', 'planets', 'seaice', 'taxis', 'tips', 'titanic', 'anagrams', 'anagrams', 'anscombe', 'anscombe', 'attention', 'attention', 'brain_networks', 'brain_networks', 'car_crashes', 'car_crashes', 'diamonds', 'diamonds', 'dots', 'dots', 'dowjones', 'dowjones', 'exercise', 'exercise', 'flights', 'flights', 'fmri', 'fmri', 'geyser', 'geyser', 'glue', 'glue', 'healthexp', 'healthexp', 'iris', 'iris', 'mpg', 'mpg', 'penguins', 'penguins', 'planets', 'planets', 'seaice', 'seaice', 'taxis', 'taxis', 'tips', 'tips', 'titanic', 'titanic', 'anagrams', 'anscombe', 'attention', 'brain_networks', 'car_crashes', 'diamonds', 'dots', 'dowjones', 'exercise', 'flights', 'fmri', 'geyser', 'glue', 'healthexp', 'iris', 'mpg', 'penguins', 'planets', 'seaice', 'taxis', 'tips', 'titanic']
```

```
<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
```

```
<class 'pandas.core.frame.DataFrame'>
```

```
RangeIndex: 891 entries, 0 to 890
```

```
Data columns (total 8 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	891 non-null	object
4	sibsp	891 non-null	int64
5	parch	891 non-null	int64
6	fare	891 non-null	float64
7	alive	891 non-null	object

```
dtypes: float64(1), int64(4), object(3)
```

```
memory usage: 55.8+ KB
```

In [35]: `df.info()`

```

Data columns (total 14 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 [37]: `df["deck"].value_counts(normalize=True)`
`df.drop(["deck"], axis=1)`

Out[37]:

	survived	pclass	sex	age	sibsp	parch	fare	embarked	class	who	adult_male
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
...
886	0	2	male	27.0	0	0	13.0000	S	Second	man	Tru
887	1	1	female	19.0	0	0	30.0000	S	First	woman	Fals
888	0	3	female	NaN	1	2	23.4500	S	Third	woman	Fals
889	1	1	male	26.0	0	0	30.0000	C	First	man	Tru
890	0	3	male	32.0	0	0	7.7500	Q	Third	man	Tru

891 rows × 14 columns

In [38]: `df1=df.drop(["embarked","class","who","adult_male","deck","embark_town","alone"]`
`df1['sex'].mode()[0]`

Out[38]: 'male'

In [39]: `df1['age'].mode`

Out[39]: <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 [36]: `df["sex"].value_counts(normalize=True)`

Out[36]:

	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 [34]: `df.tail()`

Out[34]:

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

In [33]: `df.head()`

Out[33]:

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

In [40]: `df1['age'].mean`

Out[40]: <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 [48]: `df1.loc[:, "sex"].mode()
df1.min();`

In [49]:

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

Out[49]:

	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	<bound method NDFrame._add_numeric_operations....		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 [51]:

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

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 8 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         891 non-null    object
4   sibsp       891 non-null    int64
5   parch       891 non-null    int64
6   fare        891 non-null    float64
7   alive       891 non-null    object
dtypes: float64(1), int64(4), object(3)
memory usage: 55.8+ KB
```

```
In [50]: def expand_string(s):
    result = ""
    i = 0
    while i < len(s):
        char = s[i]
        num = int(s[i+1])
        result += char * num
        i += 2
    return result
p = "a4b4c4d1"
print(expand_string(p))
```

aaaabbbbccccd

In []:

```
In [27]: data = ['a', 'b', 'a', 'c', 'b', 'a', 'c', 'c', 'b']
test_list=[[3,4,5],[6,2,4],[1,3,6]]
frequency = pd.Series(data).value_counts()
flattened_list = [item for sublist in test_list for item in sublist]
frequency1 = pd.Series(flattened_list).value_counts()
print(frequency)
print(frequency1)
```

```
a    3
b    3
c    3
dtype: int64
3    2
4    2
6    2
5    1
2    1
1    1
dtype: int64
```

```
In [29]: list1 = [1,2,3, 5, 4]
list2 = [3,4,5, 7, 8]
common_elements = set(list1).intersection(set(list2))
print(common_elements)
```

{3, 4, 5}

```
In [31]: list=['sohan','mahwesh','sahil']
f=[]
for i in list:
    f.append(i[0])
print(f)
```

['s', 'm', 's']


```
In [55]: list=["python","c","c++","java","python"]
         orignal=[]
         dublicate=[]
         for i in list:
             if i in orignal:
                 dublicate.append(i)
             else:
                 orignal.append(i)
         print(dublicate)
```

```
['python']
```

```
In [ ]:
```