ASSINGNMENT 2: - DATA PREPROCESSING

</h3>

- **Name: - Aryan Omkar Ashar**
- **Reg. Number - 19BAI10094**

1.Import the dataset using seaborn

In [3]:

```
import seaborn as sns
```

```
Matplotlib is building the font cache; this may take a moment.
```

In [4]:

```
print(sns.get_dataset_names())
```

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

In [5]:

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

In [6]:

```
df.head()
```

Out[6]:

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

2.Import necessary libraries

In [7]:

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

In [8]:

```
print(df)
```

```
     survived  pclass     sex   age  sibsp  parch     fare embarked   class  \
0           0       3    male  22.0      1      0   7.2500        S   Third
1           1       1  female  38.0      1      0  71.2833        C   First
2           1       3  female  26.0      0      0   7.9250        S   Third
```

```
3           1       1  female  35.0      1      0  53.1000        S   First
4           0       3    male  35.0      0      0   8.0500        S   Third
..        ...     ...     ...   ...    ...    ...      ...      ...     ...
886         0       2    male  27.0      0      0  13.0000        S  Second
887         1       1  female  19.0      0      0  30.0000        S   First
888         0       3  female   NaN      1      2  23.4500        S   Third
889         1       1    male  26.0      0      0  30.0000        C   First
890         0       3    male  32.0      0      0   7.7500        Q   Third

       who  adult_male deck  embark_town alive  alone
0      man        True  NaN  Southampton    no  False
1    woman       False    C    Cherbourg   yes  False
2    woman       False  NaN  Southampton   yes   True
3    woman       False    C  Southampton   yes  False
4      man        True  NaN  Southampton    no   True
..     ...         ...  ...          ...   ...    ...
886    man        True  NaN  Southampton    no   True
887  woman       False    B  Southampton   yes   True
888  woman       False  NaN  Southampton    no  False
889    man        True    C    Cherbourg   yes   True
890    man        True  NaN   Queenstown    no   True

[891 rows x 15 columns]
```

In [9]:

```
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 [11]:

```
df.describe()
```

Out[11]:

	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

	survived	pclass	age	sibsp	parch	fare
max	1.000000	3.000000	80.000000	8.000000	6.000000	512.329200

In [12]:

```
df.head()
```

Out[12]:

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

In [13]:

```
df.shape
```

Out[13]:

```
(891, 15)
```

In [14]:

```
df.columns
```

Out[14]:

```
Index(['survived', 'pclass', 'sex', 'age', 'sibsp', 'parch', 'fare',
       'embarked', 'class', 'who', 'adult_male', 'deck', 'embark_town',
       'alive', 'alone'],
      dtype='object')
```

3.Handling missing values if any

In [15]:

```
df
```

Out[15]:

	survived	pclass	sex	age	sibsp	parch	fare	embarked	class	who	adult_male	deck	embark_town	alive
0	0	3	male	22.0	1	0	7.2500	S	Third	man	True	NaN	Southampton	no
1	1	1	female	38.0	1	0	71.2833	C	First	woman	False	C	Cherbourg	yes
2	1	3	female	26.0	0	0	7.9250	S	Third	woman	False	NaN	Southampton	yes
3	1	1	female	35.0	1	0	53.1000	S	First	woman	False	C	Southampton	yes
4	0	3	male	35.0	0	0	8.0500	S	Third	man	True	NaN	Southampton	no
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
886	0	2	male	27.0	0	0	13.0000	S	Second	man	True	NaN	Southampton	no
887	1	1	female	19.0	0	0	30.0000	S	First	woman	False	B	Southampton	yes
888	0	3	female	NaN	1	2	23.4500	S	Third	woman	False	NaN	Southampton	no
889	1	1	male	26.0	0	0	30.0000	C	First	man	True	C	Cherbourg	yes
890	0	3	male	32.0	0	0	7.7500	Q	Third	man	True	NaN	Queenstown	no

891 rows × 15 columns

1.deleting rows with nans----1,0000...2 null values..then you can dlete the entire rows 2.deleting columns with nans 70 -80 % of my column contains null 3.replacing it with mean median and mode

In [16]:

```
df.isnull()
```

Out[16]:

	survived	pclass	sex	age	sibsp	parch	fare	embarked	class	who	adult_male	deck	embark_town	alive	alo
0	False	False	False	False	False	False	False	False	False	False	False	True	False	False	Fal
1	False	False	False	False	False	False	False	False	False	False	False	False	False	False	Fal
2	False	False	False	False	False	False	False	False	False	False	False	True	False	False	Fal
3	False	False	False	False	False	False	False	False	False	False	False	False	False	False	Fal
4	False	False	False	False	False	False	False	False	False	False	False	True	False	False	Fal
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
886	False	False	False	False	False	False	False	False	False	False	False	True	False	False	Fal
887	False	False	False	False	False	False	False	False	False	False	False	False	False	False	Fal
888	False	False	False	True	False	False	False	False	False	False	False	True	False	False	Fal
889	False	False	False	False	False	False	False	False	False	False	False	False	False	False	Fal
890	False	False	False	False	False	False	False	False	False	False	False	True	False	False	Fal

891 rows × 15 columns

In [17]:

```
df.isnull().any()
```

Out[17]:

```
survived       False
pclass         False
sex            False
age             True
sibsp          False
parch          False
fare           False
embarked        True
class          False
who            False
adult_male     False
deck            True
embark_town     True
alive          False
alone          False
dtype: bool
```

In [18]:

```
df.isnull().sum()
```

Out[18]:

```
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 [19]:

```
df
```

Out[19]:

	survived	pclass	sex	age	sibsp	parch	fare	embarked	class	who	adult_male	deck	embark_town	alive
0	0	3	male	22.0	1	0	7.2500	S	Third	man	True	NaN	Southampton	no
1	1	1	female	38.0	1	0	71.2833	C	First	woman	False	C	Cherbourg	yes
2	1	3	female	26.0	0	0	7.9250	S	Third	woman	False	NaN	Southampton	yes
3	1	1	female	35.0	1	0	53.1000	S	First	woman	False	C	Southampton	yes
4	0	3	male	35.0	0	0	8.0500	S	Third	man	True	NaN	Southampton	no
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
886	0	2	male	27.0	0	0	13.0000	S	Second	man	True	NaN	Southampton	no
887	1	1	female	19.0	0	0	30.0000	S	First	woman	False	B	Southampton	yes
888	0	3	female	NaN	1	2	23.4500	S	Third	woman	False	NaN	Southampton	no
889	1	1	male	26.0	0	0	30.0000	C	First	man	True	C	Cherbourg	yes
890	0	3	male	32.0	0	0	7.7500	Q	Third	man	True	NaN	Queenstown	no

891 rows × 15 columns

In [23]:

```
df['age'].fillna(df['age'].median(),inplace=True)
```

In [25]:

```
df
```

Out[25]:

	survived	pclass	sex	age	sibsp	parch	fare	embarked	class	who	adult_male	deck	embark_town	alive
0	0	3	male	22.0	1	0	7.2500	S	Third	man	True	NaN	Southampton	no
1	1	1	female	38.0	1	0	71.2833	C	First	woman	False	C	Cherbourg	yes
2	1	3	female	26.0	0	0	7.9250	S	Third	woman	False	NaN	Southampton	yes
3	1	1	female	35.0	1	0	53.1000	S	First	woman	False	C	Southampton	yes
4	0	3	male	35.0	0	0	8.0500	S	Third	man	True	NaN	Southampton	no
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
886	0	2	male	27.0	0	0	13.0000	S	Second	man	True	NaN	Southampton	no
887	1	1	female	19.0	0	0	30.0000	S	First	woman	False	B	Southampton	yes
888	0	3	female	28.0	1	2	23.4500	S	Third	woman	False	NaN	Southampton	no

	survived	pclass	sex	age	sibsp	parch	fare	embarked	class	who	adult_male	deck	embark_town	alive
889	1	1	male	26.0	0	0	30.0000	C	First	man	True	C	Charbourg	yes
890	0	3	male	32.0	0	0	7.7500	Q	Third	man	True	NaN	Queenstown	no

891 rows × 15 columns

In [26]:

```
df.isnull().any()
```

Out[26]:

```
survived       False
pclass         False
sex            False
age            False
sibsp          False
parch          False
fare           False
embarked        True
class          False
who            False
adult_male     False
deck            True
embark_town     True
alive          False
alone          False
dtype: bool
```

4. Seperate dependent and independent variables

In [27]:

```
x=df.iloc[:,0:3]
x
```

Out[27]:

	survived	pclass	sex
0	0	3	male
1	1	1	female
2	1	3	female
3	1	1	female
4	0	3	male
...	...	...	...
886	0	2	male
887	1	1	female
888	0	3	female
889	1	1	male
890	0	3	male

891 rows × 3 columns

In [28]:

```
y=df.iloc[:,3:]
y
```

Out[28]:

	age	sibsp	parch	fare	embarked	class	who	adult_male	deck	embark_town	alive	alone
0	22.0	1	0	7.2500	S	Third	man	True	NaN	Southampton	no	False
1	38.0	1	0	71.2833	C	First	woman	False	C	Cherbourg	yes	False
2	26.0	0	0	7.9250	S	Third	woman	False	NaN	Southampton	yes	True
3	35.0	1	0	53.1000	S	First	woman	False	C	Southampton	yes	False
4	35.0	0	0	8.0500	S	Third	man	True	NaN	Southampton	no	True
...	...	...	...	...	...	...	...	...	...	...	...	...
886	27.0	0	0	13.0000	S	Second	man	True	NaN	Southampton	no	True
887	19.0	0	0	30.0000	S	First	woman	False	B	Southampton	yes	True
888	28.0	1	2	23.4500	S	Third	woman	False	NaN	Southampton	no	False
889	26.0	0	0	30.0000	C	First	man	True	C	Cherbourg	yes	True
890	32.0	0	0	7.7500	Q	Third	man	True	NaN	Queenstown	no	True

891 rows × 12 columns

In [30]:

```
df["class"].value_counts()
```

Out[30]:

```
Third     491
First     216
Second    184
Name: class, dtype: int64
```

In [31]:

```
df["who"].value_counts()
```

Out[31]:

```
man      537
woman    271
child     83
Name: who, dtype: int64
```

In [34]:

```
from collections import Counter as c
c(df["class"])
```

Out[34]:

```
Counter({'Third': 491, 'First': 216, 'Second': 184})
```

In [35]:

```
data1=df.copy()
```

5.Encode the columns which are categorical

In [37]:

```
from sklearn.preprocessing import LabelEncoder
```

In [41]:

```
le=LabelEncoder()
print("before label encoding",c(df["embark_town"]))
```

```
df['Country']=le.fit_transform(df['embark_town'])
print("after label encoding",c(df["embark_town"]))
```

```
before label encoding Counter({'Southampton': 644, 'Cherbourg': 168, 'Queenstown': 77, na
n: 2})
after label encoding Counter({'Southampton': 644, 'Cherbourg': 168, 'Queenstown': 77, nan
: 2})
```

In [42]:

```
data1
```

Out[42]:

	survived	pclass	sex	age	sibsp	parch	fare	embarked	class	who	adult_male	deck	embark_town	alive
0	0	3	male	22.0	1	0	7.2500	S	Third	man	True	NaN	Southampton	no
1	1	1	female	38.0	1	0	71.2833	C	First	woman	False	C	Cherbourg	yes
2	1	3	female	26.0	0	0	7.9250	S	Third	woman	False	NaN	Southampton	yes
3	1	1	female	35.0	1	0	53.1000	S	First	woman	False	C	Southampton	yes
4	0	3	male	35.0	0	0	8.0500	S	Third	man	True	NaN	Southampton	no
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
886	0	2	male	27.0	0	0	13.0000	S	Second	man	True	NaN	Southampton	no
887	1	1	female	19.0	0	0	30.0000	S	First	woman	False	B	Southampton	yes
888	0	3	female	28.0	1	2	23.4500	S	Third	woman	False	NaN	Southampton	no
889	1	1	male	26.0	0	0	30.0000	C	First	man	True	C	Cherbourg	yes
890	0	3	male	32.0	0	0	7.7500	Q	Third	man	True	NaN	Queenstown	no

891 rows × 15 columns

In [43]:

```
data1['embark_town']=data1['embark_town'].replace(['Southampton','Cherbourg','Queenstown'
],
                                                  ['0','1','2'])
```

In [44]:

```
data1['embark_town']
```

Out[44]:

```
0       0
1       1
2       0
3       0
4       0
      ..
886     0
887     0
888     0
889     1
890     2
Name: embark_town, Length: 891, dtype: object
```

In [45]:

```
from sklearn.compose import ColumnTransformer
from sklearn.preprocessing import OneHotEncoder
```

In [46]:

```
#name of the column transformer,the transform to apply,the column indices to apply to it
ct=ColumnTransformer([("one",OneHotEncoder(),[0])],remainder="passthrough")
```

In [47]:

```
x
```

Out[47]:

	survived	pclass	sex
0	0	3	male
1	1	1	female
2	1	3	female
3	1	1	female
4	0	3	male
...	...	...	...
886	0	2	male
887	1	1	female
888	0	3	female
889	1	1	male
890	0	3	male

891 rows × 3 columns

In [48]:

```
x.shape
```

Out[48]:

```
(891, 3)
```

In [49]:

```
x=ct.fit_transform(x)
x
```

Out[49]:

```
array([[1.0, 0.0, 3, 'male'],
       [0.0, 1.0, 1, 'female'],
       [0.0, 1.0, 3, 'female'],
       ...,
       [1.0, 0.0, 3, 'female'],
       [0.0, 1.0, 1, 'male'],
       [1.0, 0.0, 3, 'male']], dtype=object)
```

In [50]:

```
x.shape
```

Out[50]:

```
(891, 4)
```

6.Splitting into test and train

In [52]:

```
from sklearn.model_selection import train_test_split
```

```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.3,random_state=0)
```

In [53]:

```
print(x_train.shape)
print(x_test.shape)
print(y_train.shape)
print(y_test.shape)
```

```
(623, 4)
(268, 4)
(623, 12)
(268, 12)
```

In [ ]:

```

```