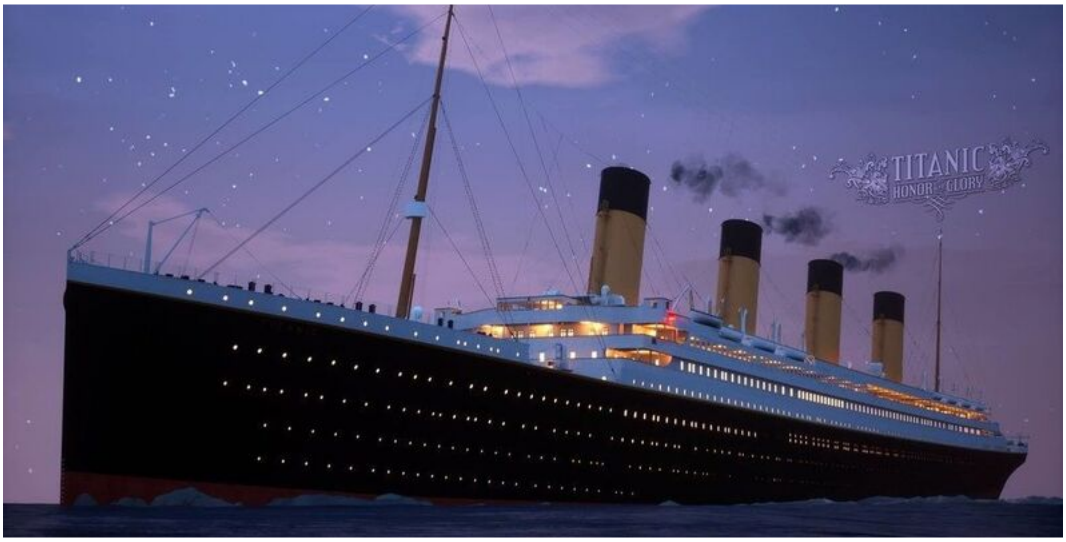
Task 2

Perform data cleaning , exploratory data analysis (EDA) on a dataset of your choice, such as the Titanic dataset from Kaggle. Explore the relationships between variables and identify patterns and trends in the data. Dataset :- https://www.kaggle.com/c/titanic/data

Machine Learning with the Titanic Dataset

Start coding or generate with AI.



Import the required libraries

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

Bringing data to on-board using pandas library

```
from google.colab import files
data=files.upload()
```

Choose Files No file chosen Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.

```
Saving titanic.csv to titanic.csv
```

Reading the csv file

```
df=pd.read_csv('titanic.csv')
df
```

	passenger_id	pclass	name	sex	age	sibsp	parch	ticket	fare
0	1216	3	Smyth, Miss. Julia	female	NaN	0	0	335432	7.7333
1	699	3	Cacic, Mr. Luka	male	38.0	0	0	315089	8.6625
2	1267	3	Van Impe, Mrs. Jean Baptiste (Rosalie Paula Go...	female	30.0	1	1	345773	24.1500
3	449	2	Hocking, Mrs. Elizabeth (Eliza Needs)	female	54.0	1	3	29105	23.0000
4	576	2	Veal, Mr. James	male	40.0	0	0	28221	13.0000
...	...	...	...	...	...	...	...	...	...
			Hipkins						

Taking shallow copy of the original dataset

```
df1=df.copy()
```

EDA

viewing the counts of record and feature

```
df1.shape
```

```
(850, 15)
```

viewing the last five feature and record using tail function

```
df1.tail()
```

	passenger_id	pclass	name	sex	age	sibsp	parch	ticket	fare
845	158	1	Hipkins, Mr. William Edward	male	55.0	0	0	680	50.000
846	174	1	Kent, Mr. Edward Austin	male	58.0	0	0	11771	29.700
			Kantor						

viewing the first five feature and record using head function

```
df1.head()
```

	passenger_id	pclass	name	sex	age	sibsp	parch	ticket	fare	cabin	e
0	1216	3	Smyth, Miss. Julia	female	NaN	0	0	335432	7.7333	NaN	
1	699	3	Cacic, Mr. Luka	male	38.0	0	0	315089	8.6625	NaN	
			Van Impe								

viewing the over all columns in a dataset

```
df1.columns
```

```
Index(['passenger_id', 'pclass', 'name', 'sex', 'age', 'sibsp', 'parch',
       'ticket', 'fare', 'cabin', 'embarked', 'boat', 'body', 'home.dest',
       'survived'],
      dtype='object')
```

The info function highlights the total number of rows in the dataset, names of the columns, their data type, and any missing value. It is used to print the summary of a data frame. It is very important to know the data types of variables that aides in understanding the nature of data

```
df1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 850 entries, 0 to 849
Data columns (total 15 columns):
 #   Column        Non-Null Count  Dtype  
---  ------        --------------  -----  
 0   passenger_id  850 non-null    int64  
 1   pclass        850 non-null    int64  
 2   name          850 non-null    object 
 3   sex           850 non-null    object 
 4   age           676 non-null    float64
 5   sibsp         850 non-null    int64  
 6   parch         850 non-null    int64  
 7   ticket        850 non-null    object 
 8   fare          849 non-null    float64
 9   cabin         191 non-null    object 
 10  embarked      849 non-null    object 
 11  boat          308 non-null    object 
 12  body          73 non-null     float64
 13  home.dest     464 non-null    object 
 14  survived      850 non-null    int64  
dtypes: float64(3), int64(5), object(7)
memory usage: 99.7+ KB
```

The describe() method is used for calculating some statistical data like percentile, mean and std of the numerical values of the Series or DataFrame. It analyzes both numeric and object series and also the DataFrame column sets of mixed data types.

```
df1.describe()
```

	passenger_id	pclass	age	sibsp	parch	fare	bc
count	850.000000	850.00000	676.000000	850.000000	850.000000	849.000000	73.000(
mean	662.816471	2.32000	29.519847	0.522353	0.382353	34.012701	165.821‍
std	380.751936	0.83853	14.562243	1.112132	0.879511	53.705779	99.068
min	1.000000	1.00000	0.166700	0.000000	0.000000	0.000000	4.000(
25%	332.250000	2.00000	20.000000	0.000000	0.000000	7.895800	75.000(
50%	676.500000	3.00000	28.000000	0.000000	0.000000	14.108300	166.000(
75%	992.250000	3.00000	37.000000	1.000000	0.000000	31.000000	260.000(
max	1307.000000	3.00000	80.000000	8.000000	9.000000	512.329200	328.000(

Handling the missing values(DATA CLEANING)

```
df1.isnull().sum()
```

```
passenger_id      0
pclass            0
name              0
```

```
sex             0
age           174
sibsp           0
parch           0
ticket          0
fare            1
cabin         659
embarked        1
boat          542
body          777
home.dest     386
survived        0
dtype: int64
```

```
df1.isnull().mean()*100
```

```
passenger_id     0.000000
pclass           0.000000
name             0.000000
sex              0.000000
age             20.470588
sibsp            0.000000
parch            0.000000
ticket           0.000000
fare             0.117647
cabin           77.529412
embarked         0.117647
boat            63.764706
body            91.411765
home.dest       45.411765
survived         0.000000
dtype: float64
```

```
print("Percentage of missing values in age is",174/850*100)
print('Percentage of missing values in cabin is',659/850*100)
print("Percentage of missing values in Embarked is",1/850*100)
```

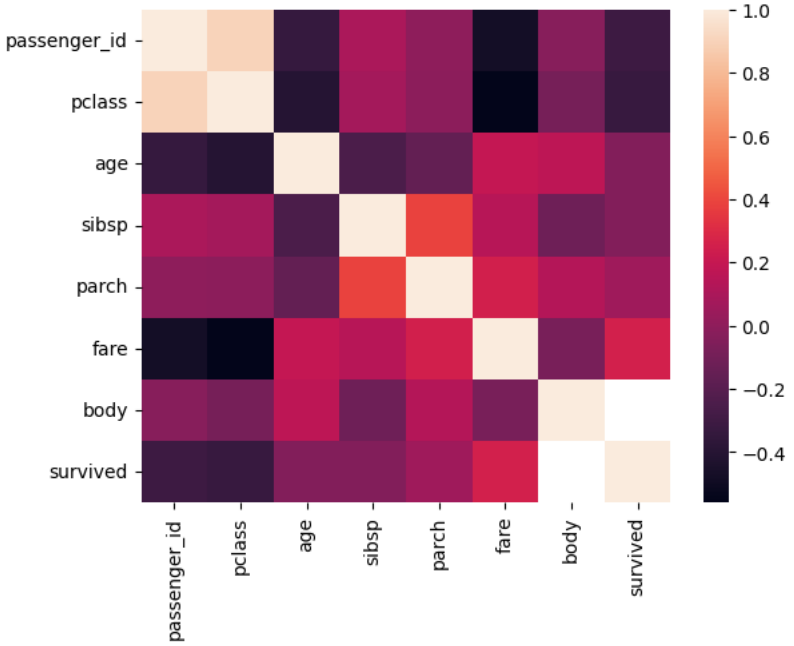
```
Percentage of missing values in age is 20.47058823529412
Percentage of missing values in cabin is 77.52941176470588
Percentage of missing values in Embarked is 0.11764705882352941
```

Handling missing values

It is observed that the variable cabin has highest missing value which is of 77%. Then Age has missing value of 20.47% and Embarked has 0.117%.

```
sns.heatmap(df.corr())
```

```
<ipython-input-87-aa4f4450a243>:1: FutureWarning: The default value of numeric_only i
  sns.heatmap(df.corr())
<Axes: >
```




```
df1['survived'].unique()
```

```
array([1, 0])
```

```
df1['cabin'].unique()
```

```
array([nan, 'C82', 'D15', 'C50', 'E33', 'B57 B59 B63 B66', 'E34', 'C83',
       'C125', 'B82 B84', 'B96 B98', 'B51 B53 B55', 'C132', 'C31', 'C68',
       'B94', 'F E46', 'C126', 'D34', 'B28', 'C65', 'B52 B54 B56', 'D17',
       'C86', 'C7', 'E25', 'E17', 'D56', 'A9', 'B26', 'C91', 'B5', 'D',
       'A23', 'C22 C26', 'C124', 'E63', 'B35', 'B18', 'C6', 'B49', 'B19',
       'G6', 'D35', 'C23 C25 C27', 'C62 C64', 'B73', 'E12', 'B41', 'A20',
       'B69', 'C78', 'A14', 'D26', 'C51', 'B79', 'E52', 'C49', 'C104',
       'B61', 'D43', 'D48', 'C123', 'A7', 'D47', 'D49', 'B80', 'A29',
       'F38', 'B58 B60', 'A34', 'D33', 'D20', 'F G63', 'T', 'A6',
       'C55 C57', 'B38', 'E49', 'F4', 'E121', 'C85', 'C87', 'B86', 'B102',
       'C111', 'D22', 'B22', 'B77', 'B50', 'B30', 'E36', 'F33', 'E60',
       'B39', 'C118', 'D21', 'F2', 'E39 E41', 'E101', 'A32', 'D38',
       'C101', 'C54', 'C106', 'C32', 'D36', 'E50', 'D40', 'C30', 'C52',
       'C130', 'C80', 'F G73', 'E10', 'B10', 'B24', 'C2', 'D28', 'E44',
       'D30', 'B20', 'C28', 'C97', 'E8', 'D11', 'B101', 'D37', 'F E57',
       'E24', 'A24', 'C46', 'E31', 'C93', 'C39', 'B37'], dtype=object)
```

```
# Dropping of 'Cabin' variable as it has highest missing values.
df1.drop(columns='cabin',axis=1)
```

	passenger_id	pclass	name	sex	age	sibsp	parch	ticket	fare
0	1216	3	Smyth, Miss. Julia	female	28.0	0	0	335432	7.7333
1	699	3	Cacic, Mr. Luka	male	38.0	0	0	315089	8.6625
2	1267	3	Van Impe, Mrs. Jean Baptiste (Rosalie Paula Go...	female	30.0	1	1	345773	24.1500
3	449	2	Hocking, Mrs. Elizabeth (Eliza Needs)	female	54.0	1	3	29105	23.0000
4	576	2	Veal, Mr. James	male	40.0	0	0	28221	13.0000

Filling NAN values with 'mean','median'and 'mode' values

```
df1['age']=df1['age'].fillna(df1['age'].median())
```

```
#df1['cabin']=df1['cabin'].fillna(df1['cabin'].mode().iloc[0])
df1['embarked']=df1['embarked'].fillna(df1['embarked'].mode().iloc[0])
```

```
df1.dropna(subset=['fare'],how='any')
```

	passenger_id	pclass	name	sex	age	sibsp	parch	ticket	fare
0	1216	3	Smyth, Miss. Julia	female	28.0	0	0	335432	7.7333
1	699	3	Cacic, Mr. Luka	male	38.0	0	0	315089	8.6625
2	1267	3	Van Impe, Mrs. Jean Baptiste (Rosalie Paula Go...	female	30.0	1	1	345773	24.1500
3	449	2	Hocking, Mrs. Elizabeth (Eliza Needs)	female	54.0	1	3	29105	23.0000
4	576	2	Veal, Mr. James	male	40.0	0	0	28221	13.0000
...	...	...	...	...	...	...	...	...	...
			Hipkins,						

```
columns_to_drop = ['boat', 'body', 'home.dest']
df2=df1.drop(columns=columns_to_drop)
```

```
df3=df2.dropna(subset=['fare'],how='any')
```

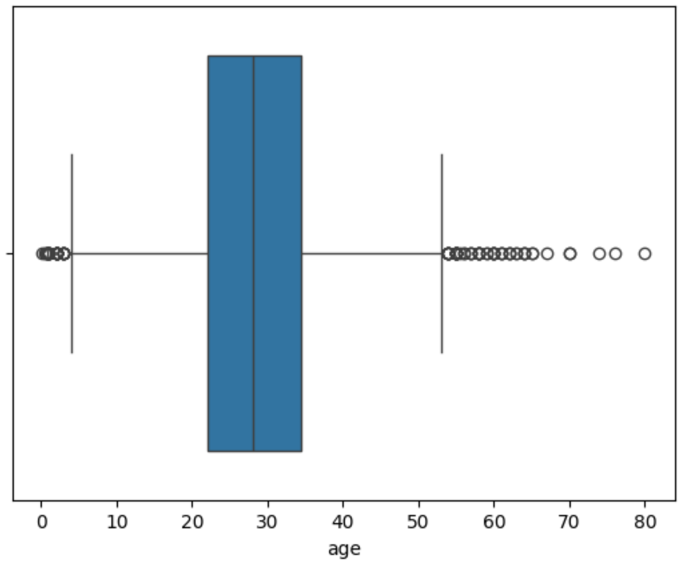
```
df3.isnull().sum()
```

```
passenger_id    0
pclass          0
name            0
sex             0
age             0
sibsp           0
parch           0
ticket          0
fare            0
cabin           0
embarked        0
survived        0
dtype: int64
```

DATA VISUALIZATION AND OUTLIER DETECTION

```
sns.boxplot(x=df3['age'])
```

```
<Axes: xlabel='age'>
```

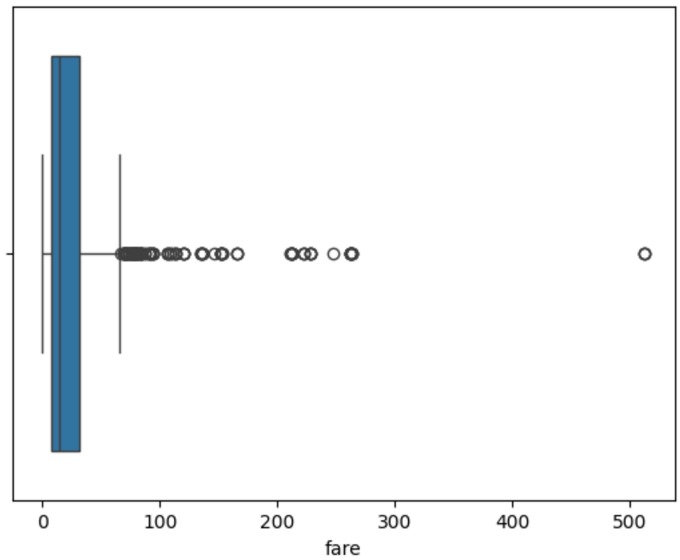



```
df3.age.mean()
```

```
29.17186890459364
```

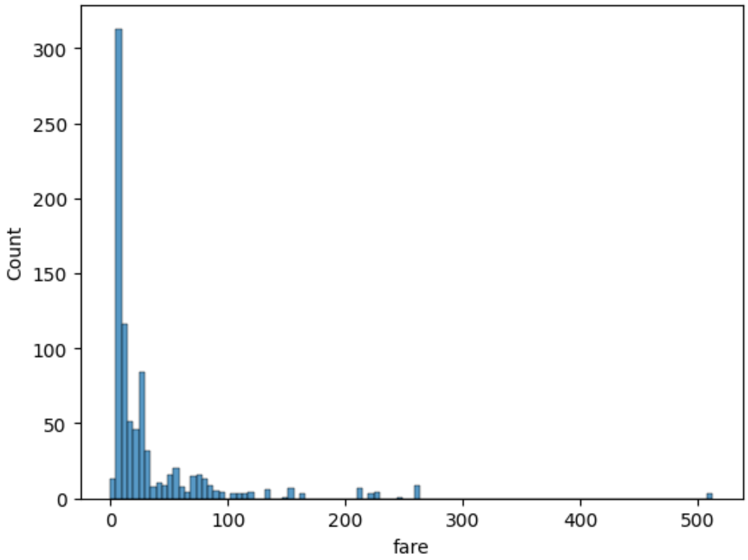
```
sns.boxplot(x = df3['fare'])
```

```
<Axes: xlabel='fare'>
```




```
# Histogram shows the outlier presence of numerical variable by right or left skew
sns.histplot(df3['fare'])
```

```
<Axes: xlabel='fare', ylabel='Count'>
```




```
print('Skewness of Fare-',df3['fare'].skew())
print('Skewness of Age-',df3['age'].skew())
```

```
Skewness of Fare- 4.3088922688863045
Skewness of Age- 0.5811314382672461
```

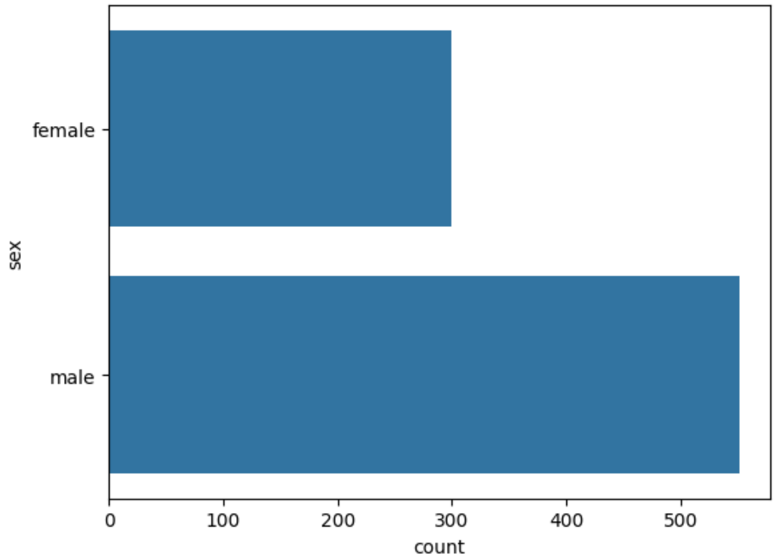
The skewness range is between -1 to 1. Here the fare variable is rightly skewed which indicates the presence of more outliers.

```
Q1 = df['fare'].quantile(0.25)
Q3 = df['fare'].quantile(0.75)
IQR = Q3 - Q1
Outlier_Fare = df[(df['fare']<Q1-1.5*IQR)|(df['fare']>Q3+1.5*IQR)]
Outlier_Fare.head()
```

	passenger_id	pclass	name	sex	age	sibsp	parch	ticket	fare	emba
10	313	1	Widener, Mr. Harry Elkins	male	27.0	0	2	113503	211.5000	
11	43	1	Bucknell, Mrs. William Robert	female	60.0	0	0	11813	76.2917	

```
print(sns.countplot(df['sex']))
```

```
Axes(0.125,0.11;0.775x0.77)
```

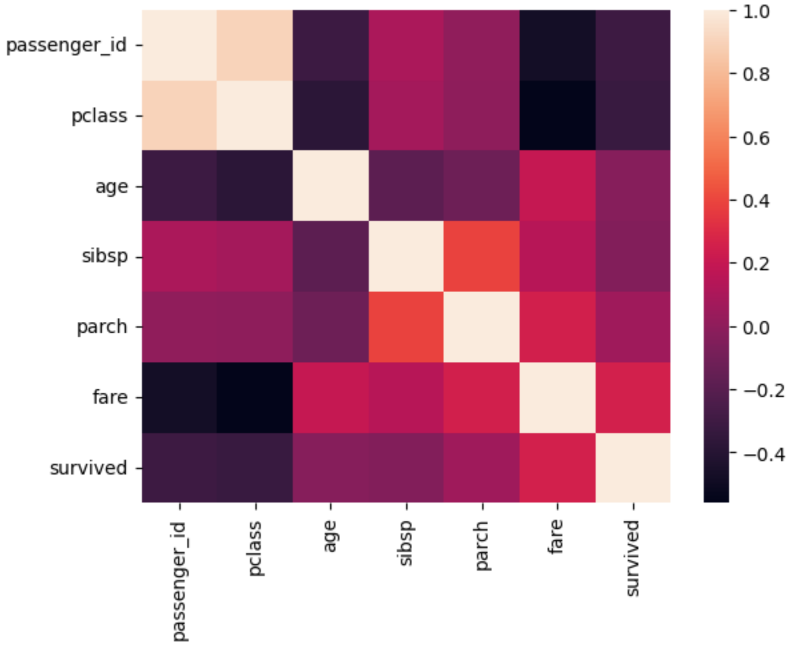



```
print('Male',(df['sex']=='male').sum()/850*100)
print('Female',(df['sex']=='female').sum()/850*100)
```

```
Male 64.8235294117647
Female 35.17647058823529
```

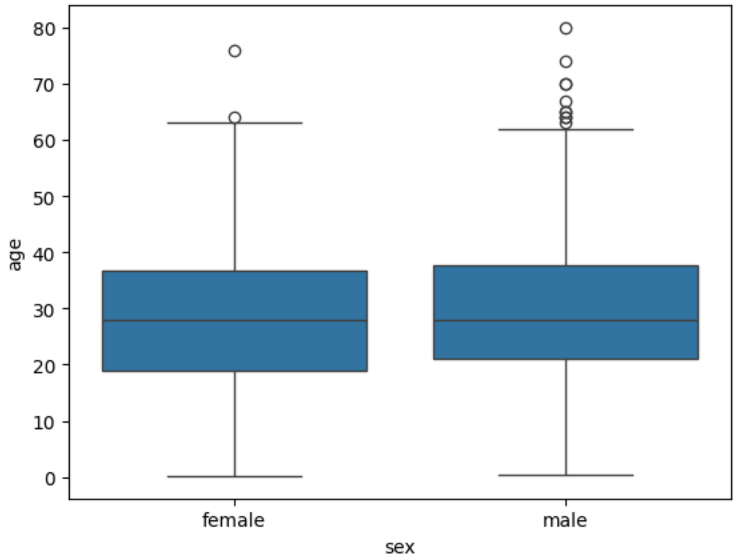
```
sns.heatmap(df3.corr())
```

```
<ipython-input-60-39c919fe9a67>:1: FutureWarning: The default value of numeric_only i
  sns.heatmap(df3.corr())
<Axes: >
```




```
sns.boxplot(y=df['age'],x=df['sex'])
```

```
<Axes: xlabel='sex', ylabel='age'>
```

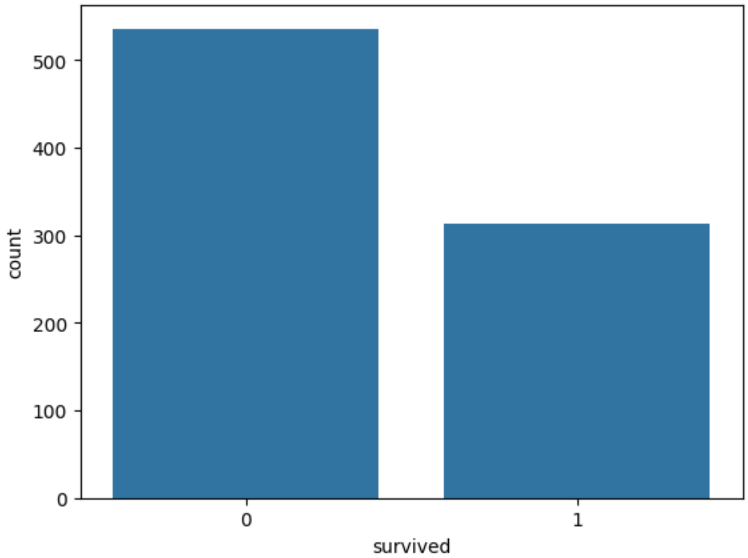



```
y=df3['survived']
y
```

```
0      1
1      0
2      0
3      1
4      0
      ..
845    0
846    0
847    1
848    0
849    0
Name: survived, Length: 849, dtype: int64
```

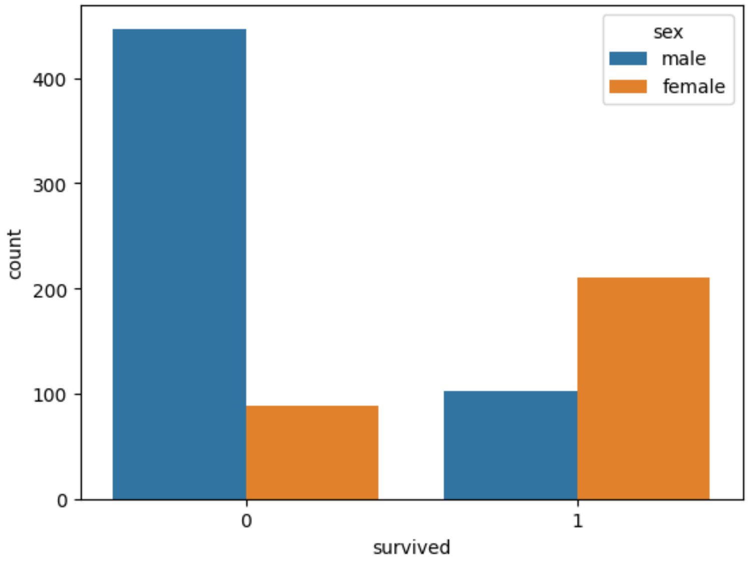
```
sns.countplot(x='survived',data=df3)
```

```
<Axes: xlabel='survived', ylabel='count'>
```



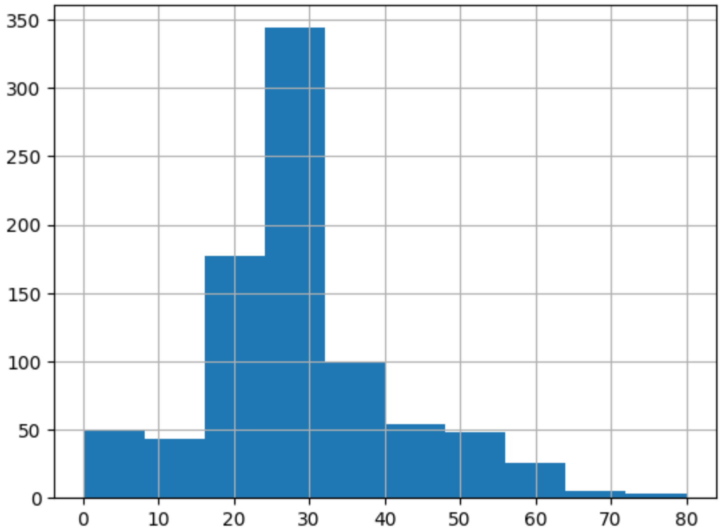

```
sns.countplot(x='survived',hue='sex',data=df3)
```

```
<Axes: xlabel='survived', ylabel='count'>
```




```
df3['age'].hist()
```

```
<Axes: >
```

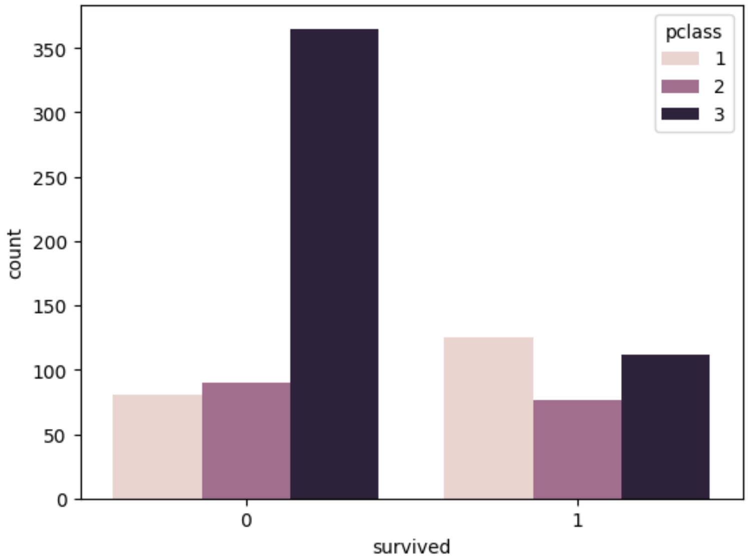



```
df3.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 849 entries, 0 to 849
Data columns (total 12 columns):
 #   Column        Non-Null Count  Dtype  
---  ------        --------------  -----  
 0   passenger_id  849 non-null    int64  
 1   pclass        849 non-null    int64  
 2   name          849 non-null    object 
 3   sex           849 non-null    object 
 4   age           849 non-null    float64
 5   sibsp         849 non-null    int64  
 6   parch         849 non-null    int64  
 7   ticket        849 non-null    object 
 8   fare          849 non-null    float64
 9   cabin         849 non-null    object 
 10  embarked      849 non-null    object 
 11  survived      849 non-null    int64  
dtypes: float64(2), int64(5), object(5)
memory usage: 86.2+ KB
```

```
sns.countplot(x='survived',hue='pclass', data=df3)
```

```
<Axes: xlabel='survived', ylabel='count'>
```



X and Y split

```
x=df3.drop(columns=['survived'],axis=1)
```

```
y=df3['survived']
```

Label encoding the object into integer

```
from sklearn.preprocessing import LabelEncoder
le=LabelEncoder()
x['sex']=le.fit_transform(x['sex'])
x['embarked']=le.fit_transform(x['embarked'])
x['name']=le.fit_transform(x['name'])
x['ticket']=le.fit_transform(x['ticket'])
x['cabin']=le.fit_transform(x['cabin'])
x['embarked']=le.fit_transform(x['embarked'])
```

Performing standard scaler method for better understanding of the machine to imporve and learn

```
from sklearn.preprocessing import StandardScaler
scale=StandardScaler()
scale.fit_transform(x)
```

```
array([[ 1.45648373,  0.81221282,  1.27732623, ..., -0.48961009,
        -0.35138222, -0.59678841],
       [ 0.09689551,  0.81221282, -1.22827218, ..., -0.47229821,
        -0.35138222,  0.6300236 ],
       [ 1.59060172,  0.81221282,  1.4653483 , ..., -0.18375144,
        -0.35138222,  0.6300236 ],
       ...,
       [-0.51320984, -0.38081259, -0.05109055, ..., -0.14928419,
        -0.35138222,  0.6300236 ],
       [ 1.18298823,  0.81221282,  0.77457157, ..., -0.3770475 ,
        -0.35138222,  0.6300236 ],
       [-0.62365994, -0.38081259, -0.46800904, ..., -0.39148649,
        -0.35138222,  0.6300236 ]])
```

Train test split

```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.2,random_state=74)
```

As this dataset comes under classifier,here we are going to apply three various classifier models such as LOGISTIC REGRESSION,KNN,SVC

Import and build LogisticRegression

```
from sklearn.linear_model import LogisticRegression
Log = LogisticRegression()
```

```
Log.fit(x_train,y_train)
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/_logistic.py:458: Conver
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
▾ LogisticRegression
LogisticRegression()
```

```
y_pred = Log.predict(x_test)
```

```
from sklearn.metrics import confusion_matrix,recall_score,accuracy_score,precision_score,f1_score
cm = confusion_matrix(y_test,y_pred)
print('Confusion',cm)
recall = recall_score(y_test,y_pred)
print('Recall',recall)
accuracy = accuracy_score(y_test,y_pred)
print('Accuracy',accuracy)
precision = precision_score(y_test,y_pred)
print('Precision',precision)
F1score = f1_score(y_test,y_pred)
print('F1 score',F1score)
```

```
Confusion [[99 16]
 [17 38]]
Recall 0.6909090909090909
Accuracy 0.8058823529411765
Precision 0.7037037037037037
F1 score 0.6972477064220184
```

Import and build KNN Algorithm

```
from sklearn.neighbors import KNeighborsClassifier
classifi=KNeighborsClassifier(n_neighbors=5,p=2)
classifi.fit(x_train,y_train)
```