```
# Data Manipulation
import pandas as pd

# Mathmatical operation
import numpy as np

# Data visualization
import seaborn as sns

# Machine learning Algoritham
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score

# For creating a file
import joblib
```

```
"""Data Collection and Processing"""

#load data from csv file using pandas DataFrame
titanic=pd.read_csv('train.csv')
#printing data first 5 rows using head()
titanic.head()
```

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	F
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2
1	2	1	1	Cumings, Mrs. John Bradley (Florence	female	38.0	1	0	PC 17599	71.2

```
# find no of rows and column using shape attribute
titanic.shape
```

```
(891, 12)
```

```
#getting some information about the data using info()
titanic.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 12 columns):
 #   Column       Non-Null Count  Dtype  
---  ------       --------------  -----  
 0   PassengerId  891 non-null    int64  
 1   Survived     891 non-null    int64  
 2   Pclass       891 non-null    int64  
 3   Name         891 non-null    object 
 4   Sex          891 non-null    object 
 5   Age          714 non-null    float64
 6   SibSp        891 non-null    int64  
 7   Parch        891 non-null    int64  
 8   Ticket       891 non-null    object 
 9   Fare         891 non-null    float64
 10  Cabin        204 non-null    object 
 11  Embarked     889 non-null    object 
dtypes: float64(2), int64(5), object(5)
memory usage: 83.7+ KB
```

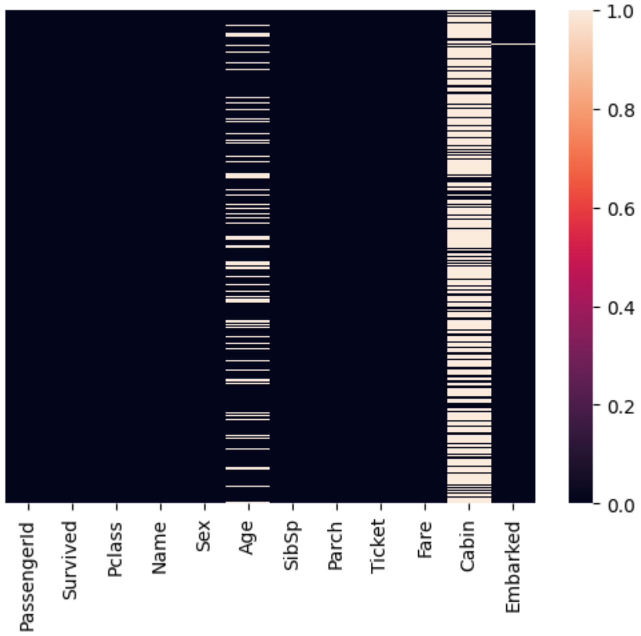
```
# check the number of each column in missing values using isnull().sum
titanic.isnull().sum()
```

```
PassengerId      0
Survived         0
```

```
Pclass           0
Name             0
Sex              0
Age            177
SibSp            0
Parch            0
Ticket           0
Fare             0
Cabin          687
Embarked         2
dtype: int64
```

```
sns.heatmap(titanic.isnull(),yticklabels=False)
# heatmap for finding null values here white portion means null value exist
```

```
<Axes: >
```



```
"""Handling the missing values"""

# Drop the Cabin column from the dataframe most of the value is missing thats why we cant find means so w
#for row we mention  axis = 0 and column axis = 1
# we droping column and assign in titanic_data
titanic_data=titanic.drop(columns='Cabin', axis=1)
```

```
# replacing in missing value in age column with mean value using fillna() means not available
titanic_data['Age'].fillna(titanic_data['Age'].mean(),inplace=True)
```

```
#finding the mode value in "Embarked " column
print(titanic_data['Embarked'].mode())

print(titanic_data['Embarked'].mode()[0])
```

```
0    S
Name: Embarked, dtype: object
S
```

```
# replacing the missing values in mode values in "Embarked" column
titanic_data['Embarked'].fillna(titanic_data['Embarked'].mode()[0],inplace=True)

# check again the number of each column in missing values using isnull().sum
titanic_data.isnull().sum()
```

```
PassengerId    0
Survived       0
Pclass         0
Name           0
Sex            0
Age            0
SibSp          0
Parch          0
```

```
Ticket         0
Fare           0
Embarked       0
dtype: int64
```

"""Data Analysis"""

```
# getting statistical data measure using describe()
titanic_data.describe()
```

	PassengerId	Survived	Pclass	Age	SibSp	Parch	Fa
count	891.000000	891.000000	891.000000	891.000000	891.000000	891.000000	891.0000
mean	446.000000	0.383838	2.308642	29.699118	0.523008	0.381594	32.2042
std	257.353842	0.486592	0.836071	13.002015	1.102743	0.806057	49.6934
min	1.000000	0.000000	1.000000	0.420000	0.000000	0.000000	0.0000
25%	223.500000	0.000000	2.000000	22.000000	0.000000	0.000000	7.9104
50%	446.000000	0.000000	3.000000	29.699118	0.000000	0.000000	14.4542
75%	668.500000	1.000000	3.000000	35.000000	1.000000	0.000000	31.0000
max	891.000000	1.000000	3.000000	80.000000	8.000000	6.000000	512.3292

```
# finding the no of people survived or not using value_counts()
titanic_data['Survived'].value_counts()
#here 0 means not survived and 1 means survived
```

```
0    549
1    342
Name: Survived, dtype: int64
```

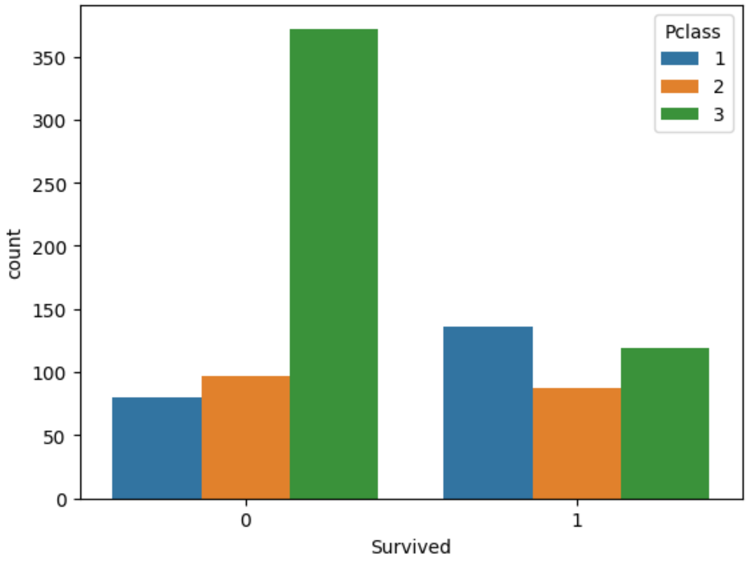
```
titanic_data['Embarked'].value_counts()
```

```
0    646
1    168
2     77
Name: Embarked, dtype: int64
```

Let`s visualize the count of survivals wrt Pclass

```
# Let`s visualize the count of survivals wrt Pclass
sns.countplot(x=titanic_data['Survived'],hue=titanic_data['Pclass'])
```

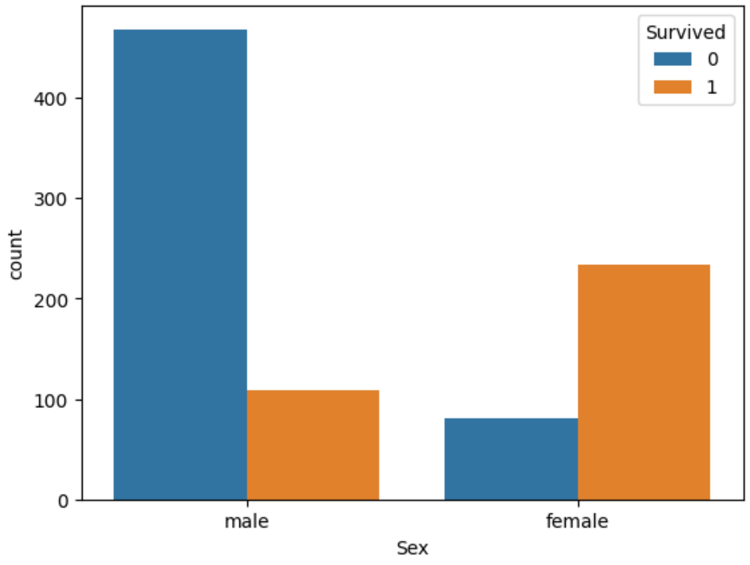
```
<Axes: xlabel='Survived', ylabel='count'>
```



Here we can see survival rate of male and female

```
sns.countplot(x=titanic_data['Sex'],hue=titanic_data['Survived'])
```

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



Here we use dictionary to change character key value to numerical

```
titanic_data.replace({'Sex':{'male':0,'female':1}, 'Embarked':{'S':0,'C':1,'Q':2}},inplace=True)
titanic_data.head()
```

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Far
0	1	0	3	Braund, Mr. Owen Harris	0	22.0	1	0	A/5 21171	7.250
1	2	1	1	Cumings, Mrs. John Bradley (Florence	1	38.0	1	0	PC 17599	71.283

"""Seprating feature and Target"""

```
"""Seprating feature and Target"""

X=titanic_data.drop(columns=['PassengerId','Name','Ticket','Survived'],axis=1)
Y=titanic_data['Survived']

print(X)

print(Y)
```

```
     Pclass  Sex        Age  SibSp  Parch     Fare  Embarked
0         3    0  22.000000      1      0   7.2500         0
1         1    1  38.000000      1      0  71.2833         1
2         3    1  26.000000      0      0   7.9250         0
3         1    1  35.000000      1      0  53.1000         0
4         3    0  35.000000      0      0   8.0500         0
..      ...  ...        ...    ...    ...      ...       ...
886       2    0  27.000000      0      0  13.0000         0
887       1    1  19.000000      0      0  30.0000         0
888       3    1  29.699118      1      2  23.4500         0
889       1    0  26.000000      0      0  30.0000         1
890       3    0  32.000000      0      0   7.7500         2

[891 rows x 7 columns]
0      0
1      1
2      1
3      1
4      0
      ..
886    0
887    1
888    0
889    1
890    0
Name: Survived, Length: 891, dtype: int64
```

```
"""Splitting the data into training data and test data"""

X_train,X_test,Y_train,Y_test = train_test_split(X,Y,test_size=0.2,random_state=2)

print(X_train.shape,Y_train.shape,X_test.shape,Y_test.shape)
```

```
(712, 7) (712,) (179, 7) (179,)
```

```
# for making decision

model = DecisionTreeClassifier()
model.fit(X_train,Y_train)
```

```
▾ DecisionTreeClassifier
DecisionTreeClassifier()
```

```
# Accuracy and training data
X_train_prediction = model.predict(X_train)
print(X_train_prediction)
```

```
[0 1 0 0 0 0 0 1 1 0 0 1 0 0 1 1 1 0 0 0 1 0 1 0 0 1 0 0 0 0 0 0 0 0 0 0 1
 0 1 0 0 0 0 1 0 0 0 1 0 1 0 0 1 0 0 0 0 0 1 0 1 0 0 1 1 0 0 1 0 0 1 0 0 1
 0 1 0 0 1 0 1 0 0 0 1 0 0 0 0 0 0 0 1 1 0 0 0 1 0 1 1 1 0 0 0 1 0 1 0 0 0
 1 1 0 0 1 0 0 1 0 1 1 0 0 1 0 1 0 1 1 1 0 1 1 1 1 1 1 0 0 1 1 1 0 1 1 0 0
 0 1 0 0 0 0 1 0 0 0 0 1 0 0 0 1 1 0 0 0 0 0 0 0 0 1 1 1 0 0 1 1 1 0 1 1 1
 1 0 0 0 0 0 0 0 0 0 1 0 1 1 1 1 1 0 0 0 0 0 0 1 0 0 1 1 1 1 0 0 0 0 0 0 0
 0 1 0 0 0 1 0 0 1 1 0 0 1 1 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 1 0 1 0 0 0
 0 0 0 0 0 0 1 0 1 0 0 0 0 0 1 0 0 0 0 1 0 1 0 0 1 1 1 0 0 1 0 0 0 0 1 0 1
 0 1 0 0 0 0 0 0 0 1 1 1 0 0 0 0 0 1 1 1 0 1 1 0 0 0 0 0 0 0 0 0 1 1 0 0 0
 0 1 1 1 0 0 0 0 1 0 0 0 0 1 0 0 0 1 0 0 1 0 1 1 0 1 0 0 0 1 1 0 1 1 1 0 1
 0 0 1 0 0 0 1 1 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 1 0 1 0 1 1 1 0 1 1 0 0 0
 0 1 1 1 0 0 1 1 0 0 1 0 1 0 0 0 0 1 1 0 0 0 1 0 0 0 0 0 1 0 0 1 0 1 1 0 0
 1 1 0 0 0 1 0 0 0 0 0 0 0 0 1 1 1 1 1 0 0 1 0 1 0 1 1 0 1 0 0 0 1 1 0 1 0
 0 0 0 0 0 0 0 1 1 1 0 0 0 1 0 0 1 0 1 0 0 0 0 1 1 0 1 1 0 0 0 1 1 1 0 1 0
 0 1 0 0 1 0 0 1 0 0 0 1 1 0 0 0 0 0 1 0 0 0 0 0 1 1 1 1 1 0 1 1 0 1 1 0 0
 0 0 0 0 0 1 0 0 0 0 0 0 0 0 1 0 1 1 1 0 0 1 0 0 1 0 0 0 0 0 1 1 1 0 0 0 1
 0 0 1 0 0 0 0 0 1 0 1 0 1 0 0 1 1 0 0 1 1 0 1 1 1 1 0 0 0 0 1 1 0 0 0 1 1
 0 0 0 0 0 0 0 0 0 0 0 1 0 0 1 1 1 0 0 0 1 0 0 0 1 1 0 0 0 0 0 1 1 0 0 0 0
 0 0 0 1 0 1 0 0 0 1 1 1 1 1 1 0 1 1 0 0 0 1 1 0 0 0 0 1 0 1 1 0 1 1 0 0 0
 0 0 0 1 0 0 0 0 0]
```

```
training_data_accuracy = accuracy_score(Y_train,X_train_prediction)
print('Accuracy score of training data :',training_data_accuracy)
```

```
Accuracy score of training data : 0.9859550561797753
```

```
#Accuracy on test data
X_test_prediction = model.predict(X_test)
print(X_test_prediction)
```

```
[0 0 1 0 1 0 0 0 0 1 0 1 1 0 0 0 1 0 0 0 1 1 0 0 0 1 1 1 0 0 1 0 0 0 0 1 1
 0 0 0 0 0 1 0 0 0 1 0 0 0 0 1 0 0 0 0 0 0 0 0 1 1 0 0 1 1 1 0 0 0 1 0 0 0
 1 1 1 1 1 1 1 0 0 1 1 1 0 0 1 0 0 1 0 0 0 1 0 1 0 1 1 0 1 1 0 1 1 1 1 0 1
 0 0 0 1 1 0 1 1 0 1 0 0 0 0 0 0 1 0 0 0 1 0 1 0 1 0 0 0 0 0 1 1 1 1 1 0 0
 1 0 0 1 0 0 1 0 1 1 1 0 1 0 1 0 1 1 0 0 1 0 0 1 1 1 0 0 0 0 1]
```

```
testing_data_accuracy = accuracy_score(Y_test,X_test_prediction)
print('Accuracy score of testing data :',testing_data_accuracy)
```

```
Accuracy score of testing data : 0.776536312849162
```

For creating file using joblib

```
# Here we create file which is use for predicting a persion in django framework

file='job_modell.sav'
joblib.dump(model,file)
```

```
['job_modell.sav']
```