Importing libraries

In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score
```

Data Collection & Preprocessing

In [ ]:
```
# load the data form csv file to pandas dataframe
df=pd.read_csv("E:\Titanic-Dataset.csv")
df
```

In [3]:
```
# printing the first 5 rows of the dataframe (known survived,trainvied)
df.head()
```

Out[3]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	S
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN	S

In [4]:
```
df.tail()
```

Out[4]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
886	887	0	2	Montvila, Rev. Juozas	male	27.0	0	0	211536	13.00	NaN	S
887	888	1	1	Graham, Miss. Margaret Edith	female	19.0	0	0	112053	30.00	B42	S
888	889	0	3	Johnston, Miss. Catherine Helen "Carrie"	female	NaN	1	2	W./C. 6607	23.45	NaN	S
889	890	1	1	Behr, Mr. Karl Howell	male	26.0	0	0	111369	30.00	C148	C
890	891	0	3	Dooley, Mr. Patrick	male	32.0	0	0	370376	7.75	NaN	Q

In [5]:
```
# printing number of rows and columns
df.shape
```

Out[5]:
```
(891, 12)
```

In [6]:
```
# getting same information about the data
df.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
```

In [7]:
```
# check the number of missing values in each column
df.isnull().sum()
```

Out[7]:
```
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
```

Handling the missing values

In [8]:
```
# drop the "Cabin" column from the dataframe
df = df.drop(columns='Cabin', axis=1)
```

In [9]:
```
# replacing the missing values in "Age" column with mean value
df['Age'].fillna(df['Age'].mean(), inplace=True)
```

In [10]:
```
# finding the mode value of "Embarked" column
print(df['Embarked'].mode())
```
```
0    S
Name: Embarked, dtype: object
```

In [11]:
```
print(df['Embarked'].mode()[0])
```
```
S
```

In [12]:
```
# replacing the missing values in "Embarked" column with mode value
df['Embarked'].fillna(df['Embarked'].mode()[0], inplace=True)
```

In [13]:
```
# check the number of missing values in each column
df.isnull().sum()
```

Out[13]:
```
PassengerId    0
Survived       0
Pclass         0
Name           0
Sex            0
Age            0
SibSp          0
Parch          0
Ticket         0
Fare           0
Embarked       0
dtype: int64
```

In [14]:
```
# getting some statistical measures about the data
df.describe()
```

Out[14]:

	PassengerId	Survived	Pclass	Age	SibSp	Parch	Fare
count	891.000000	891.000000	891.000000	891.000000	891.000000	891.000000	891.000000
mean	446.000000	0.383838	2.308642	29.699118	0.523008	0.381594	32.204208
std	257.353842	0.486592	0.836071	13.002015	1.102743	0.806057	49.693429
min	1.000000	0.000000	1.000000	0.420000	0.000000	0.000000	0.000000
25%	223.500000	0.000000	2.000000	22.000000	0.000000	0.000000	7.910400
50%	446.000000	0.000000	3.000000	29.699118	0.000000	0.000000	14.454200
75%	668.500000	1.000000	3.000000	35.000000	1.000000	0.000000	31.000000
max	891.000000	1.000000	3.000000	80.000000	8.000000	6.000000	512.329200

In [15]:
```
# finding the number of people survived and not survived
df['Survived'].value_counts()
```

Out[15]:
```
0    549
1    342
Name: Survived, dtype: int64
```

Data Visualization

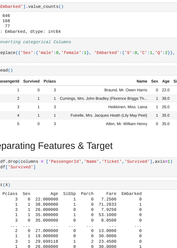
In [16]:
```
sns.set()
```

In [17]:
```
# making a count plot for "Survived" columns
sns.countplot(x='Survived', data=df)
plt.show()  # display the plot
```

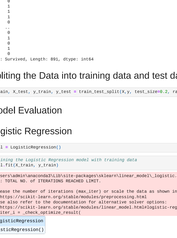
In [18]:
```
df['Sex'].value_counts()
```

Out[18]:
```
male    577
female  314
Name: Sex, dtype: int64
```

In [19]:
```
# making a count plot for 'sex' columns
sns.countplot(x='Sex', data=df)
plt.show()  # display the plot
```



In [20]:
```
# number of survivor gender wise
sns.countplot(x='Sex', hue='Survived', data=df)
plt.show()  # display the plot
```



In [21]:
```
# making a count plot for 'Pclass' columns  (Pclass=Ticket class)
sns.countplot(x='Pclass', data=df)
plt.show()  # display the plot
```


In [22]:
```
sns.countplot(x='Pclass', hue='Survived', data=df)
```

Out[22]:
```
<Axes: xlabel='Pclass', ylabel='count'>
```


Encoding the Categorical Columns

In [23]:
```
df['Sex'].value_counts()
```

Out[23]:
```
male    577
female  314
Name: Sex, dtype: int64
```

In [24]:
```
df['Embarked'].value_counts()
```

Out[24]:
```
S    646
C    168
Q     77
Name: Embarked, dtype: int64
```

In [25]:
```
# converting categorical Columns
df.replace({'Sex':{'male':0,'female':1}, 'Embarked':{'S':0,'C':1,'Q':2}}, inplace=True)
```

In [26]:
```
df.head()
```

Out[26]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Embarked
0	1	0	3	Braund, Mr. Owen Harris	0	22.0	1	0	A/5 21171	7.2500	0
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	1	38.0	1	0	PC 17599	71.2833	1
2	3	1	3	Heikkinen, Miss. Laina	1	26.0	0	0	STON/O2. 3101282	7.9250	0
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	1	35.0	1	0	113803	53.1000	0
4	5	0	3	Allen, Mr. William Henry	0	35.0	0	0	373450	8.0500	0

Separating Features & Target

In [27]:
```
X = df.drop(columns = ['PassengerId','Name','Ticket','Survived'],axis=1)
y = df['Survived']
```

In [28]:
```
print(X)
```
```
     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]
```

In [29]:
```
print(Y)
```
```
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

In [30]:
```
X_train, X_test, y_train, y_test = train_test_split(X,y, test_size=0.2, random_state=49)
```

Model Evaluation

Logistic Regression

In [31]:
```
model = LogisticRegression()
```

In [32]:
```
#training the logistic Regression model with training data
model.fit(X_train, y_train)
```
```
C:\Users\admin\anaconda3\Lib\site-packages\sklearn\linear_model\_logistic.py:460: ConvergenceWarning: lbfgs failed to converge (status=1):
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(
```

Out[32]:
```
▾ LogisticRegression
LogisticRegression()
```

Accuracy Score

In [33]:
```
# accuracy on test data
X_test_prediction = model.predict(X_test)
```

In [34]:
```
test_data_accuracy = accuracy_score(y_test, X_test_prediction)

test_data_accuracy
```

Out[34]:
```
0.8156424581005587
```

Random forest classification

In [43]:
```
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=49)

print('shape of x_train=',X_train.shape)
print('shape of y_train=',y_train.shape)
print('shape of x_test=',X_test.shape)
print('shape of y_test=',y_test.shape)

shape of x_train= (712, 7)
shape of y_train= (712,)
shape of x_test= (179, 7)
shape of y_test= (179,)
```

In [44]:
```
from sklearn.ensemble import RandomForestClassifier
```

In [45]:
```
classifier=RandomForestClassifier(n_estimators=100,criterion='gini')
classifier.fit(X_train,y_train)
```

Out[45]:
```
▾ RandomForestClassifier
RandomForestClassifier()
```

In [46]:
```
classifier.score(X_test,y_test)
```

Out[46]:
```
0.8603351955307262
```