Experiment No. 2
Analyze the Titanic Survival Dataset and apply appropriate regression technique
Date of Performance:
Date of Submission:

Aim: Analyze the Titanic Survival Dataset and apply appropriate Regression Technique.

Objective: Able to perform various feature engineering tasks, apply logistic regression on the given dataset and maximize the accuracy.

Theory:

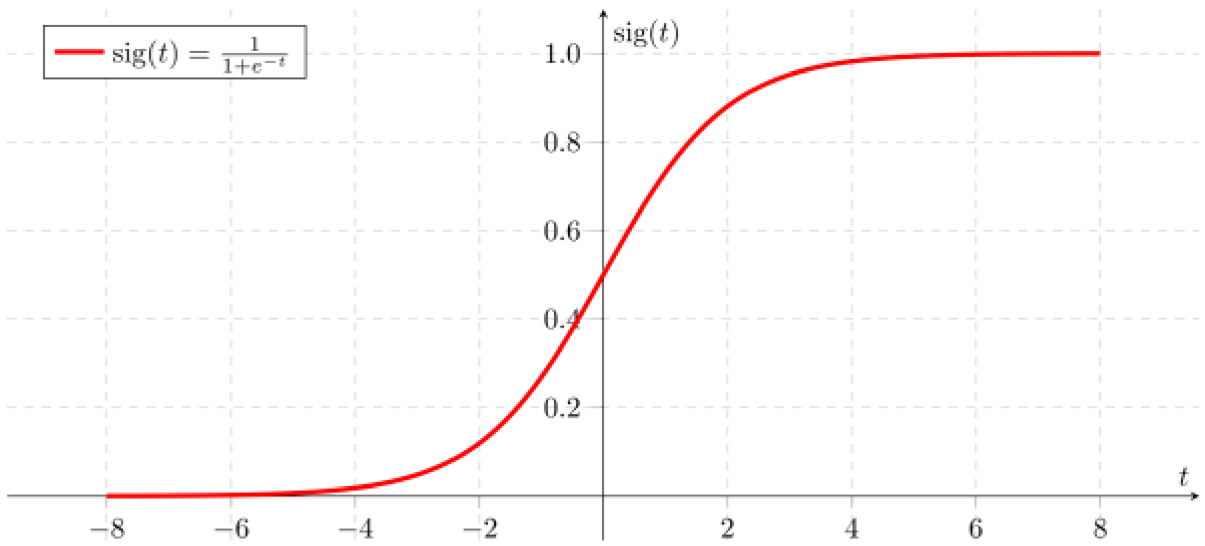
Logistic Regression was used in the biological sciences in early twentieth century. It was then used in many social science applications. Logistic Regression is used when the dependent variable(target) is categorical and is binary in nature. In order to perform binary classification the logistic regression techniques makes use of Sigmoid function.

For example,

To predict whether an email is spam (1) or (0)

Whether the tumor is malignant (1) or not (0)

Consider a scenario where we need to classify whether an email is spam or not. If we use linear regression for this problem, there is a need for setting up a threshold based on which classification can be done. Say if the actual class is malignant, predicted continuous value 0.4 and the threshold value is 0.5, the data point will be classified as not malignant which can lead to serious consequence in real time.



From this example, it can be inferred that linear regression is not suitable for classification problem. Linear regression is unbounded, and this brings logistic regression into picture. Their value strictly ranges from 0 to 1.

Dataset:

The sinking of the Titanic is one of the most infamous shipwrecks in history.

On April 15, 1912, during her maiden voyage, the widely considered "unsinkable" RMS Titanic sank after colliding with an iceberg. Unfortunately, there weren't enough lifeboats for everyone onboard, resulting in the death of 1502 out of 2224 passengers and crew.

While there was some element of luck involved in surviving, it seems some groups of people were more likely to survive than others.

In this challenge, we ask you to build a predictive model that answers the question: "what sorts of people were more likely to survive?" using passenger data (ie name, age, gender, socioeconomic class, etc).

Variable	Definition	Key
survival	Survival	0 = No, 1 = Yes
pclass	Ticket class	1 = 1st, 2 = 2nd, 3 = 3rd
sex	Sex	
Age	Age in years	
sibsp	# of siblings / spouses aboard the Titanic	
parch	# of parents / children aboard the Titanic	
ticket	Ticket number	

fare	Passenger fare	
cabin	Cabin number	
embarked	Port of Embarkation	C = Cherbourg, Q = Queenstown, S = Southampton

Variable Notes pclass: A proxy for socio-economic status (SES)

1st = Upper, 2nd = Middle, 3rd = Lower

age: Age is fractional if less than 1. If the age is estimated, is it in the form of xx.5

sibsp: The dataset defines family relations in this way...,

Sibling = brother, sister, stepbrother, stepsister

Spouse = husband, wife (mistresses and fiancés were ignored)

parch: The dataset defines family relations in this way...

Parent = mother, father

Child = daughter, son, stepdaughter, stepson

Some children travelled only with a nanny, therefore parch=0 for them.

Code:

Conclusion:

1. What are features have been chosen to develop the model? Justify the features chosen to determine the survival of a passenger.

   The features selected include :

   Pclass: Higher class passengers may have had higher chances of survival

   Sex: Women may be given preference during the evacuation,

   Age: Age may play a role as children and elderly passengers may have been given priority,

   SibSp : Number of siblings/spouses aboard and

   Parch: no of parents/children are considered for Family presence.

   Port of departure and socioeconomic status were connected. These attributes were picked based on how they might be related to socioeconomic and survival characteristics.

2. Comment on the accuracy obtained.

   The findings were presented by the logistic regression model, which had an accuracy of approximately 0.807 on the training data and approximately 0.78 on the test data. The model learned well from the training data and performed respectively on unseen data and it successfully identified key features and correlations in the dataset.

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

from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
from sklearn.model_selection import train_test_split from sklearn import
preprocessing
```

```
df = pd.read_csv('./Titanic-Dataset.csv') df.head()
```

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

```
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
```

```
# check the number of missing values in each column
df.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
```

```
df = df.drop(columns='Cabin', axis=1)
```

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

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

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

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

```
S
```

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

```
# check the number of missing values in each column
df.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
```

```
df.describe()
```

	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

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

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

```
df['Sex'].value_counts()
```

```
male      577 female
314 Name: Sex, dtype:
int64
```

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

```
S    646
C    168
Q     77
Name: Embarked, dtype: int64
```

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

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

```
df.head()
```

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

(Florence Briggs Th...

2	3	1	3	Heikkinen, Miss. Laina	1	26.0	0	0	STON/O2. 3101282	7.9	
3	4	1	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel) 35.0	1	0	113803	53.1		
4	5	0	3	Allen, Mr. William Henry	0	35.0	0	0	373450	8.0	

```
X.head()
```

	Pclass	Sex	Age	SibSp	Parch	Fare	Embarked
0	3	0	22.0	1	0	7.2500	0
1	1	1	38.0	1	0	71.2833	1
2	3	1	26.0	0	0	7.9250	0
3	1	1	35.0	1	0	53.1000	0
4	3	0	35.0	0	0	8.0500	0

```
Y.head()
```

```
0    0
1    1
2    1
3    1
4    0
Name: Survived, dtype: int64
```

```
#Splitting the data into training data & Test data
X_train, X_test, Y_train, Y_test = train_test_split(X,Y, test_size=0.2, random_state=2)
```

```
print(X.shape, X_train.shape, X_test.shape)
```

```
(891, 7) (712, 7) (179, 7)
```

```
model = LogisticRegression()
```

```
# training the Logistic Regression model with training data
model.fit(X_train, Y_train)
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/_logistic.py:458: ConvergenceWarning: lbfg
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()
```

```
# accuracy on training data
X_train_prediction = model.predict(X_train)
```

```
training_data_accuracy = accuracy_score(Y_train, X _train_prediction)
print('Accuracy of training data : ' , training_data_accuracy)
```

```
Accuracy of training data :  0.8075842696629213
```

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

```
test_data_accuracy = accuracy_score (Y_test, X_test_prediction )
print('Accuracy score of test data : ' , test_data_accuracy )
```

```
Accuracy score of test data :  0.7821229050279329
```

0s completed at 6:16 PM