| |
|---|
| 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:**

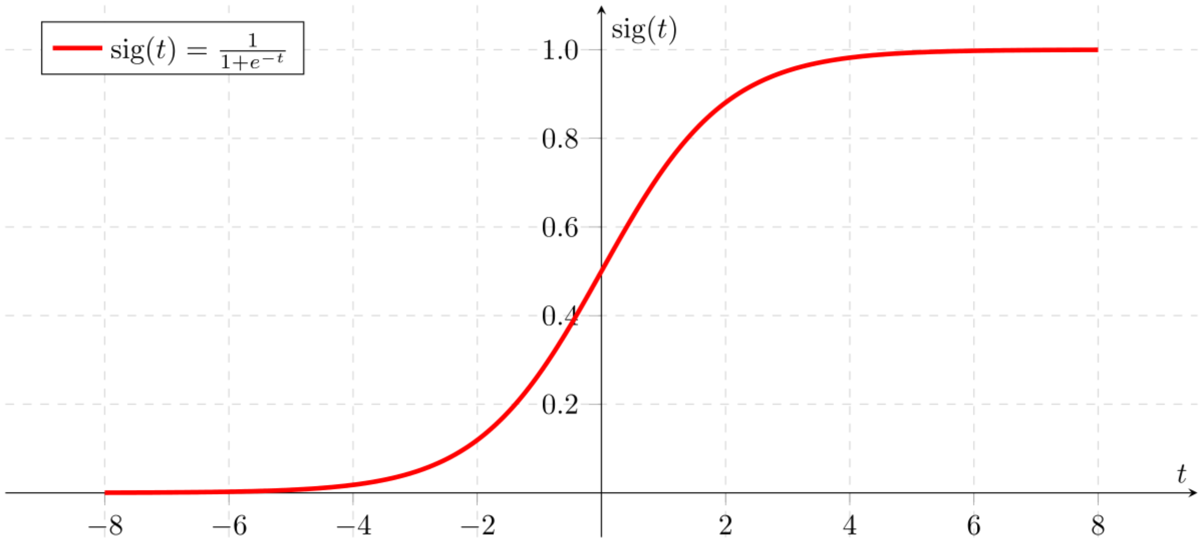
Logistic Regression was used in the biological sciences in early twentieth century. It was then used in many social science applications. Logistic Regression is used when the dependent variable(target) is categorical and is binary in nature. In order to perform binary classification the logistic regression 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, socio-economic 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.
   - Firstly the Titanic dataset was loaded and preprocessing was carried out in order to clean the data. Unwanted columns like alive, alone, embarked_town, who, adult_male & deck was dropped as they were of no use for prediction the model. Get_dummies function was used in order to convert categorical data into numeric data so the prediction can be done easily.
   - Different diagrams are plotted for the visualization and counts of number of female and males are done and also null vales are dropped from embarked column and mean values are filled in place of null values in column age.
   - Further to determine survival of a passenger the data was split into train and test data with size of 80 & 20 where Y_train contains column = Survival and X_Train consists of the remaining columns which will help us to predict the outcome.

**2.** Comment on the accuracy obtained.

Accuracy is the percentage of correct classifications that a trained machine learning model achieves, i.e., the number of correct predictions divided by the total number of predictions across all classes. The model is trained using Logistic Regression and the accuracy obtained is = 0.84 which is equivalent to approximately 84%.

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

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

```
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
```

```
df.head()
```

| | survived | pclass | sex | age | sibsp | parch | fare | embarked | class | who | ad |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 0 | 3 | male | 22.0 | 1 | 0 | 7.2500 | S | Third | man | |
| **1** | 1 | 1 | female | 38.0 | 1 | 0 | 71.2833 | C | First | woman | |
| **2** | 1 | 3 | female | 26.0 | 0 | 0 | 7.9250 | S | Third | woman | |
| **3** | 1 | 1 | female | 35.0 | 1 | 0 | 53.1000 | S | First | woman | |
| **4** | 0 | 3 | male | 35.0 | 0 | 0 | 8.0500 | S | Third | man | |

```
columns = ['alive', 'alone', 'embark_town', 'who', 'adult_male', 'deck']
data1 = df.drop(columns, axis=1)
```

```
data1.head()
```

| | survived | pclass | sex | age | sibsp | parch | fare | embarked | class |
|---|---|---|---|---|---|---|---|---|---|

```
data1
```

| | 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 |

891 rows × 9 columns

```
data1[data1['sex'].str.match("female")].count()
```

```
survived    314
pclass      314
sex         314
age         261
sibsp       314
parch       314
fare        314
embarked    312
class       314
dtype: int64
```

```
data1[data1['sex'].str.match("male")].count()
```

```
survived    577
pclass      577
sex         577
age         453
sibsp       577
parch       577
fare        577
embarked    577
class       577
dtype: int64
```

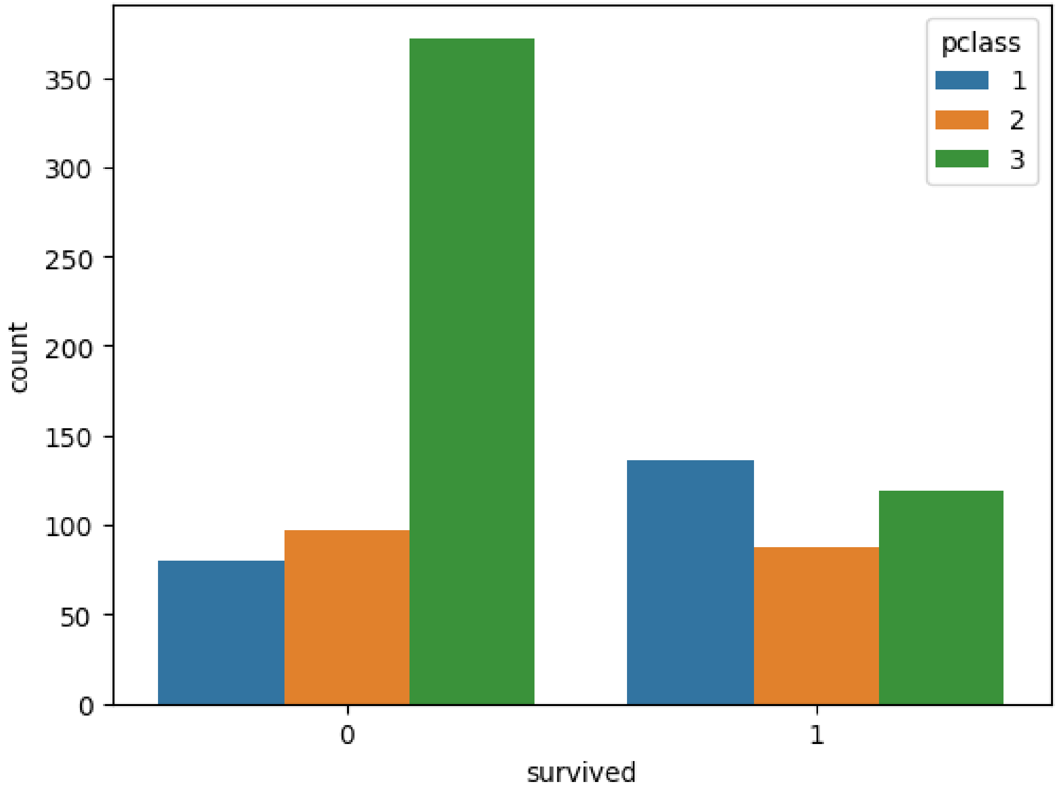
```
gender = pd.get_dummies(data1['sex'], drop_first=True)
```

```
data1['gender'] = gender
```

```
data1.drop('sex', axis=1,inplace=True)
```

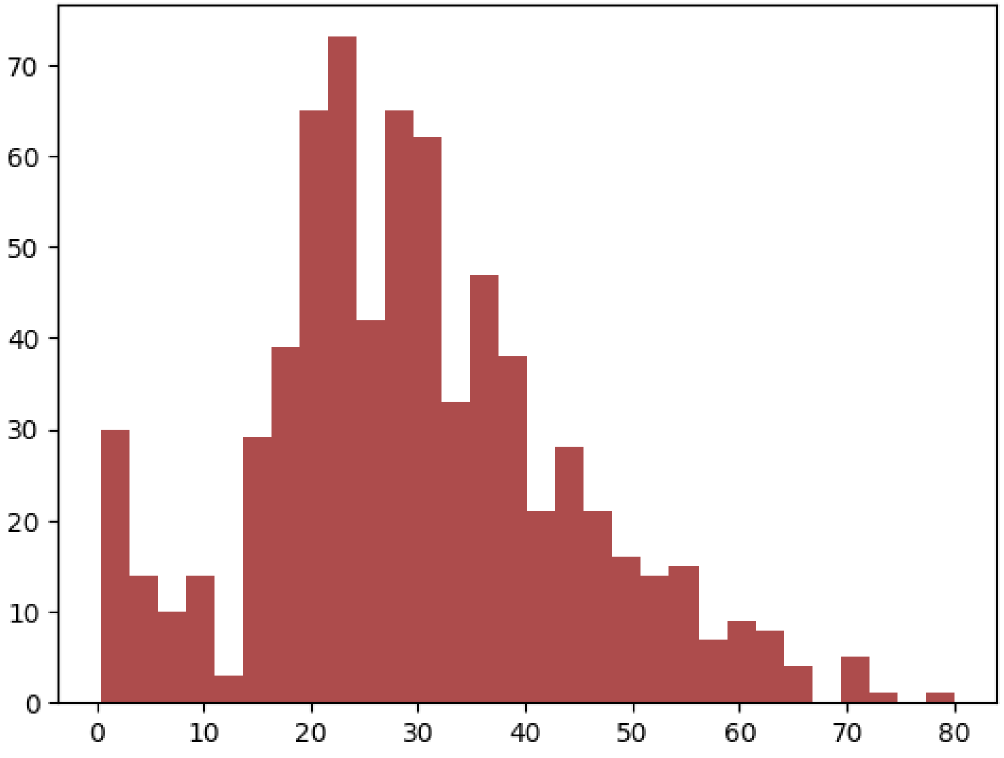
```
sns.countplot(x='survived', hue='pclass', data=data1)
```

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




```
plt.hist(data1['age'],bins=30,color='darkred',alpha=0.7)
```

```
(array([30., 14., 10., 14.,  3., 29., 39., 65., 73., 42., 65., 62., 33.,
        47., 38., 21., 28., 21., 16., 14., 15.,  7.,  9.,  8.,  4.,  0.,
         5.,  1.,  0.,  1.]),
 array([ 0.42      ,  3.07266667,  5.72533333,  8.378     , 11.03066667,
        13.68333333, 16.336     , 18.98866667, 21.64133333, 24.294     ,
        26.94666667, 29.59933333, 32.252     , 34.90466667, 37.55733333,
        40.21      , 42.86266667, 45.51533333, 48.168     , 50.82066667,
        53.47333333, 56.126     , 58.77866667, 61.43133333, 64.084     ,
        66.73666667, 69.38933333, 72.042     , 74.69466667, 77.34733333,
        80.        ]),
 <BarContainer object of 30 artists>)
```



```
data1.head()
```

| | survived | pclass | age | sibsp | parch | fare | embarked | class | gender |
|---|---|---|---|---|---|---|---|---|---|
| **0** | 0 | 3 | 22.0 | 1 | 0 | 7.2500 | S | Third | 1 |
| **1** | 1 | 1 | 38.0 | 1 | 0 | 71.2833 | C | First | 0 |
| **2** | 1 | 3 | 26.0 | 0 | 0 | 7.9250 | S | Third | 0 |
| **3** | 1 | 1 | 35.0 | 1 | 0 | 53.1000 | S | First | 0 |
| **4** | 0 | 3 | 35.0 | 0 | 0 | 8.0500 | S | Third | 1 |

```
change = {'First':1 ,'Second':2,'Third':3}
data1['class'] = data1['class'].replace(change)
```

```
change1 = {'C':1 ,'Q':2,'S':3}
data1['embarked'] = data1['embarked'].replace(change1)
```

```
data1.head()
```

| | survived | pclass | age | sibsp | parch | fare | embarked | class | gender |
|---|---|---|---|---|---|---|---|---|---|
| **0** | 0 | 3 | 22.0 | 1 | 0 | 7.2500 | 3.0 | 3 | 1 |
| **1** | 1 | 1 | 38.0 | 1 | 0 | 71.2833 | 1.0 | 1 | 0 |
| **2** | 1 | 3 | 26.0 | 0 | 0 | 7.9250 | 3.0 | 3 | 0 |
| **3** | 1 | 1 | 35.0 | 1 | 0 | 53.1000 | 3.0 | 1 | 0 |
| **4** | 0 | 3 | 35.0 | 0 | 0 | 8.0500 | 3.0 | 3 | 1 |

```
column_name = 'embarked'
data1 = data1.dropna(subset = [column_name],axis = 0)
```

```
data1['age'].fillna(data1['age'].mean() , inplace=True)
```

```
<ipython-input-140-b5d6b2d9217e>:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexi
  data1['age'].fillna(data1['age'].mean() , inplace=True)
```

```
x=data1.iloc[:,1:]
y=data1.iloc[:,0]
```

```
x
```

| | pclass | age | sibsp | parch | fare | embarked | class | gender |
|---|---|---|---|---|---|---|---|---|
| **0** | 3 | 22.000000 | 1 | 0 | 7.2500 | 3.0 | 3 | 1 |
| **1** | 1 | 38.000000 | 1 | 0 | 71.2833 | 1.0 | 1 | 0 |
| **2** | 3 | 26.000000 | 0 | 0 | 7.9250 | 3.0 | 3 | 0 |
| **3** | 1 | 35.000000 | 1 | 0 | 53.1000 | 3.0 | 1 | 0 |
| **4** | 3 | 35.000000 | 0 | 0 | 8.0500 | 3.0 | 3 | 1 |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... |
| **886** | 2 | 27.000000 | 0 | 0 | 13.0000 | 3.0 | 2 | 1 |

```
y
```

```
0      0
1      1
2      1
3      1
4      0
      ..
886    0
887    1
888    0
889    1
890    0
Name: survived, Length: 889, dtype: int64
```

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

```
X_train, X_test, Y_train , Y_test = train_test_split(x , y,test_size = 0.2 , random_state=1)
```

```
model = LogisticRegression()
```

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

```
(711, 8) (711,)
```

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

```
▾ LogisticRegression
LogisticRegression()
```

```
y_pred = model.predict(X_test)
```

```
accuracy = accuracy_score(Y_test,y_pred)
print(f"Accuracy:{accuracy:.2f}")
```

```
Accuracy:0.84
```

0s completed at 1:21 AM