Experiment No. 2
Analyze the Titanic Survival Dataset and apply appropriate regression technique
Date of Performance: 31/07/2023
Date of Submission: 11/08/2023

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:

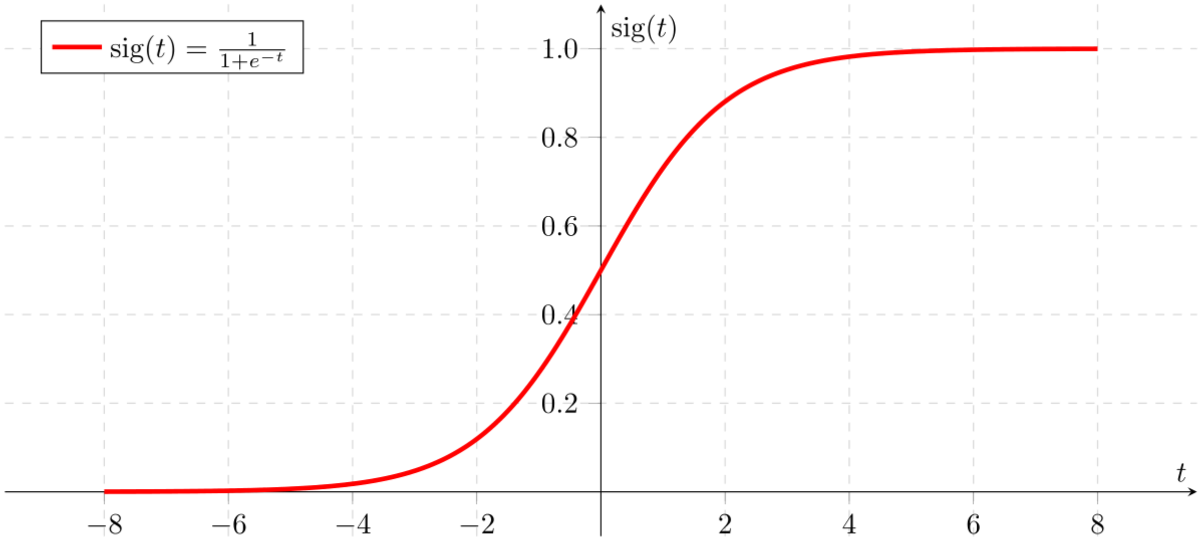
Logistic Regression was used in the biological sciences in early twentieth century. It was then used in many social science applications. Logistic Regression is used when the dependent variable(target) is categorical and is binary in nature. In order to perform binary classification the logistic regression 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. This experiment aims to predict Titanic incident survival using Logistic Regression. Starting with pandas, we analyze column values for missing data, which we subsequently fill. Utilizing seaborn, we visualize survivor distribution, influencing attribute selection.
2. Logistic Regression is then applied to predict survivors, unveiling their count distribution. To assess accuracy, pandas' metrics feature computes the model's precision. This comprehensive approach combines data analysis, visualization, and modeling to predict survival and determine model accuracy effectively.

AUC of the predictions: 0.8010869565217391

Accuracy score of the predictions: 0.8156424581005587

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

```
df = pd.DataFrame(pd.read_csv('./train.csv'))
test_data = pd.DataFrame(pd.read_csv('./test.csv'))
gender_df = pd.DataFrame(pd.read_csv('./gender_submission.csv'))
```

```
df.head()
```

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

```
for i in df.columns:
  print(i,"\t-\t", df[i].isna().mean()*100)
```

```
PassengerId     -       0.0
Survived        -       0.0
Pclass  -       0.0
Name    -       0.0
Sex     -       0.0
Age     -       19.865319865319865
SibSp   -       0.0
Parch   -       0.0
Ticket  -       0.0
Fare    -       0.0
Cabin   -       77.10437710437711
Embarked        -       0.22446689113355783
```

```
df = df.drop(["Cabin"], axis=1)
```

```
df['Age'].fillna(df['Age'].median(), inplace=True)
df['Embarked'].fillna(df['Embarked'].mode(), inplace=True)
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 11 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          891 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  Embarked     889 non-null    object
dtypes: float64(2), int64(5), object(4)
memory usage: 76.7+ KB
```

```
df = df.drop(["PassengerId", "Fare", "Ticket", "Name"], axis = 1)
```

```
from sklearn.preprocessing import LabelEncoder
```

```
cat_col= df.drop(df.select_dtypes(exclude=['object']), axis=1).columns
print(cat_col)
```

```
Index(['Sex', 'Embarked'], dtype='object')
```

```
enc1 = LabelEncoder()
df[cat_col[0]] = enc1.fit_transform(df[cat_col[0]].astype('str'))
```

```
enc2 = LabelEncoder()
df[cat_col[1]] = enc2.fit_transform(df[cat_col[1]].astype('str'))
```

```
df.head()
```

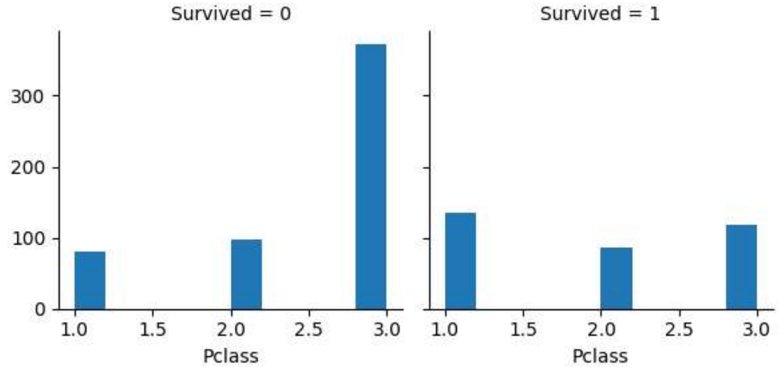
	Survived	Pclass	Sex	Age	SibSp	Parch	Embarked
0	0	3	1	22.0	1	0	2
1	1	1	0	38.0	1	0	0
2	1	3	0	26.0	0	0	2
3	1	1	0	35.0	1	0	2
4	0	3	1	35.0	0	0	2

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 7 columns):
 #   Column    Non-Null Count  Dtype  
---  ------    --------------  -----  
 0   Survived  891 non-null    int64  
 1   Pclass    891 non-null    int64  
 2   Sex       891 non-null    int64  
 3   Age       891 non-null    float64
 4   SibSp     891 non-null    int64  
 5   Parch     891 non-null    int64  
 6   Embarked  891 non-null    int64  
dtypes: float64(1), int64(6)
memory usage: 48.9 KB
```

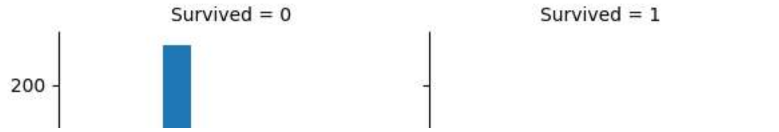
```
sns.FacetGrid(df, col= 'Survived').map(plt.hist,'Pclass')
```

```
<seaborn.axisgrid.FacetGrid at 0x7a5d1990d540>
```



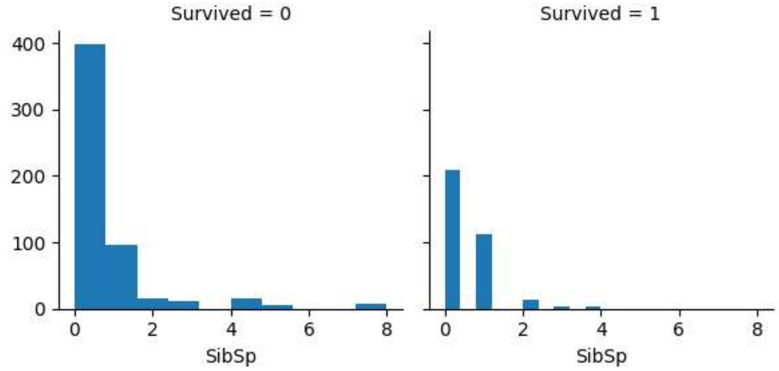
```
sns.FacetGrid(df, col='Survived').map(plt.hist, 'Age')
```

```
<seaborn.axisgrid.FacetGrid at 0x7a5d199fa140>
```



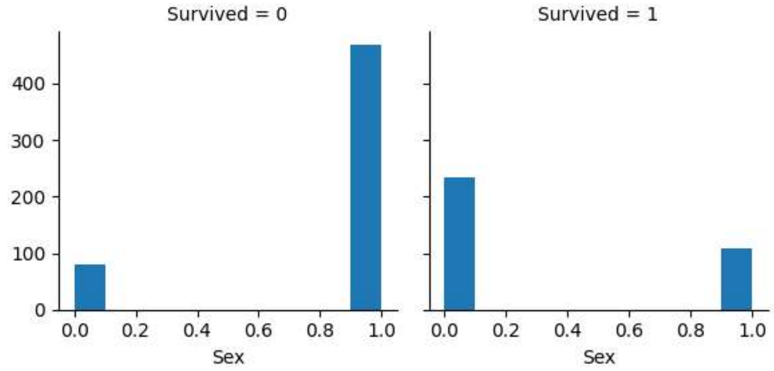
```
sns.FacetGrid(df, col='Survived').map(plt.hist, 'SibSp')
```

```
<seaborn.axisgrid.FacetGrid at 0x7a5d198bfb80>
```



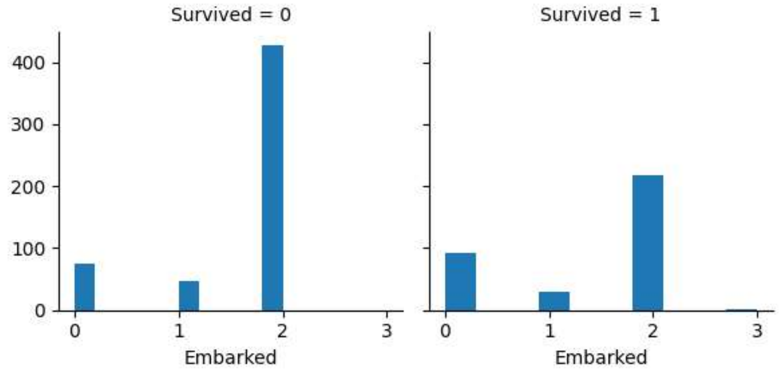
```
sns.FacetGrid(df, col='Survived').map(plt.hist, 'Sex')
```

```
<seaborn.axisgrid.FacetGrid at 0x7a5d197e8b20>
```



```
sns.FacetGrid(df, col='Survived').map(plt.hist, 'Embarked')
```

```
<seaborn.axisgrid.FacetGrid at 0x7a5d196a2f50>
```



```
X = df.drop(['Survived'], axis=1)
y = df['Survived']
```

```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2)
```

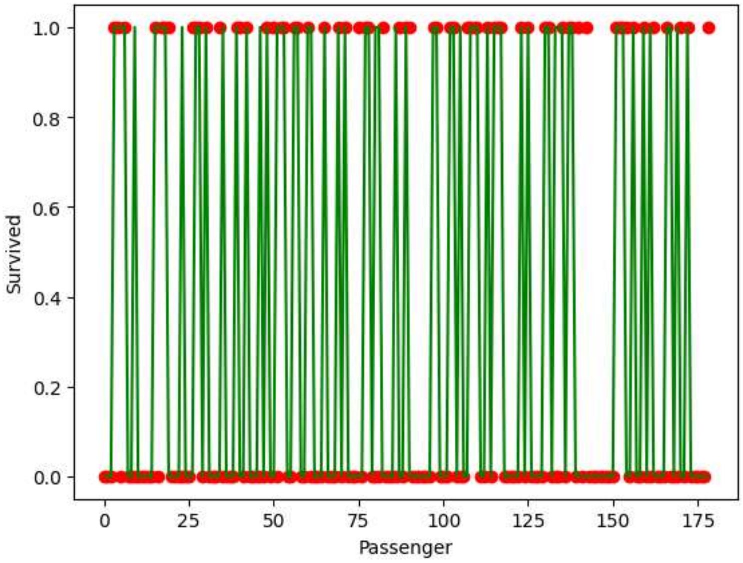
```
from sklearn.linear_model import LogisticRegression
model = LogisticRegression()
model.fit(X_train, y_train)
```

```
▾ LogisticRegression
LogisticRegression()
```

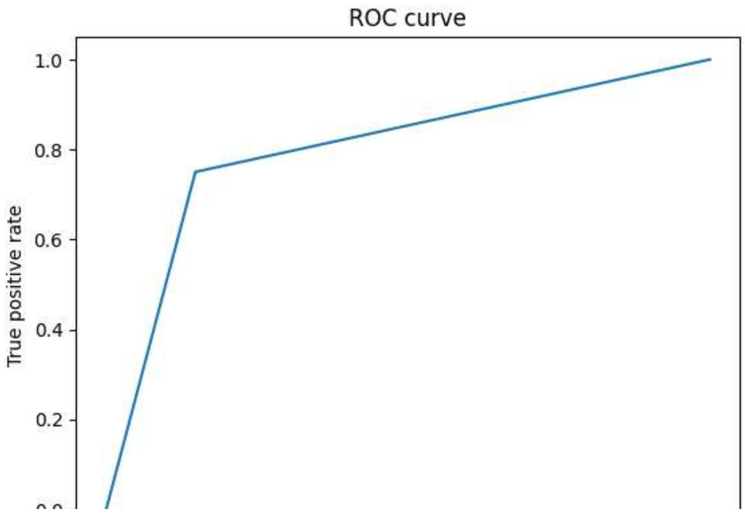
```
y_pred = model.predict(X_test)
pred_df = pd.DataFrame({'Actual': y_test, 'Predicted': y_pred})
pred_df.head()
```

	Actual	Predicted
145	0	0
45	0	0
57	0	0
677	1	1
506	1	1

```
plt.scatter([i for i in range(len(X_test["Age"]))], y_test, color='red')
plt.plot([i for i in range(len(X_test["Age"]))], y_pred, color='green')
plt.ylabel('Survived')
plt.xlabel('Passenger')
plt.show()
```



```
from sklearn import metrics
fpr, tpr, thresholds = metrics.roc_curve(y_test, y_pred, pos_label=1)
plt.plot(fpr, tpr)
plt.xlabel('False positive rate')
plt.ylabel('True positive rate')
plt.title('ROC curve')
plt.show()
```



```
print("AUC of the predictions: {0}".format(metrics.auc(fpr, tpr)))
print("Accuracy score of the predictions: {0}".format(metrics.accuracy_score(y_pred, y_test)))
```

```
AUC of the predictions: 0.8010869565217391
Accuracy score of the predictions: 0.8156424581005587
```

✓ 0s completed at 00:26