

Experiment No. 2
Analyze the Titanic Survival Dataset and apply appropriate regression technique
Date of Performance:31/07/23
Date of Submission:10/08/23

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:

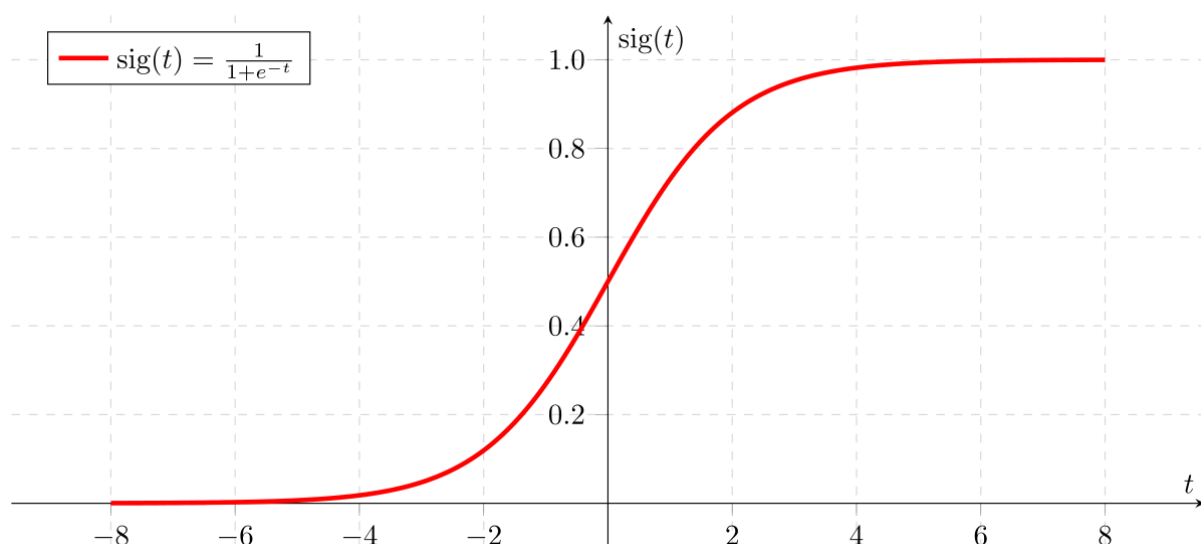
Logistic Regression was used in the biological sciences in early twentieth century. It was then used in many social science applications. Logistic Regression is used when the dependent variable(target) is categorical and is binary in nature. In order to perform binary classification the logistic regression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

The following features have been picked to create the model:

1. Pclass (Passenger Class): This refers to the ticket class (first, second, or third class). It may be a useful indication since upper classes may have received preference in times of need.
2. Sex: The passenger's gender is represented by this attribute. Gender may play a crucial role in predicting survival due to the "women and children first" strategy that was historically observed during the evacuation of the Titanic catastrophe.
3. Age: The passenger's age is an essential factor. It is known that young adults may have had a higher survival rate than children and seniors.
4. Number of Brothers/Spouses Aboard, or SibSp. This function lets passengers know if their siblings or spouses are traveling with them. It could affect the evacuation decision-making process. Number of Parents/Children Aboard.

5. Parch Similar to 'SibSp', this trait denotes the presence of parents or kids on board, which may affect survival choices.

6. Fare: The passenger's fare may be related to their accommodations or class, which may impact their ability to use lifeboats or other safety equipment.

7. Embarked: The port of embarkation is represented by this characteristic (S = Southampton, C = Cherbourg, Q = Queenstown). The socioeconomic effects of various embarkation places could have an impact on survival rates.


Accuracy:

1. Test Accuracy (69.40%): The accuracy here being which shows that for about 69.40% of the test data, the model properly predicts the survival result. The model correctly predicted the survival result for around 69.40% of the occurrences in your test dataset.

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

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

```
dataset = pd.read_csv("/content/train.csv")
dataset.head()
```



	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	Heikinen, 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

```
dataset.describe()
```

	PassengerId	Survived	Pclass	Age	SibSp	Parch	Fare
count	891.000000	891.000000	891.000000	714.000000	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	14.526497	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	20.125000	0.000000	0.000000	7.910400
50%	446.000000	0.000000	3.000000	28.000000	0.000000	0.000000	14.454200
75%	668.500000	1.000000	3.000000	38.000000	1.000000	0.000000	31.000000
max	891.000000	1.000000	3.000000	80.000000	8.000000	6.000000	512.329200

```
dataset.info()
```

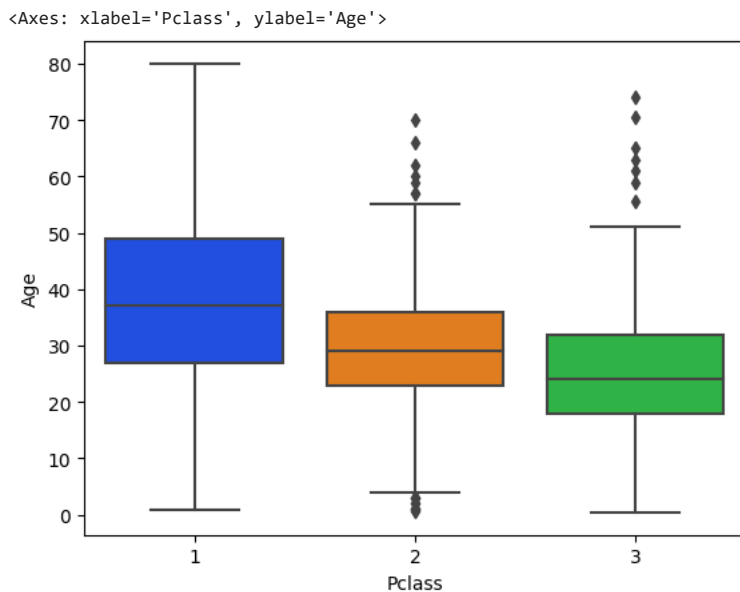
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 12 columns):
#   Column      Non-Null Count  Dtype
---  -
0   PassengerId  891 non-null   int64
1   Survived     891 non-null   int64
2   Pclass       891 non-null   int64
3   Name         891 non-null   object
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
```

```
dataset.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
```

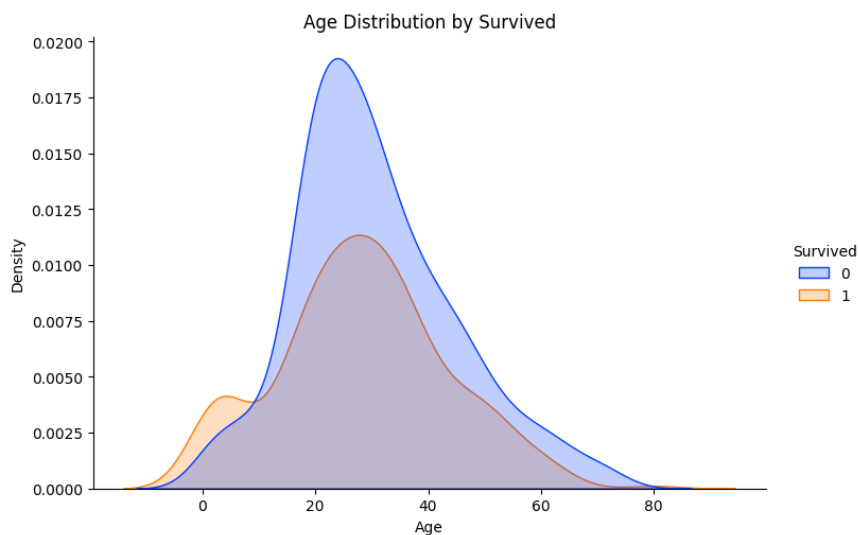
```
# dataset['Age'] = dataset['Age'].fillna(dataset['Age'].mean())
# df['Fare'] = df['Fare'].fillna(df['Fare'].mean())
```

```
#Pclass can be a proxy for socio-economic status (SES)
sb.boxplot(x="Pclass",y="Age",data=dataset,palette=sb.color_palette('bright')[:3])
```



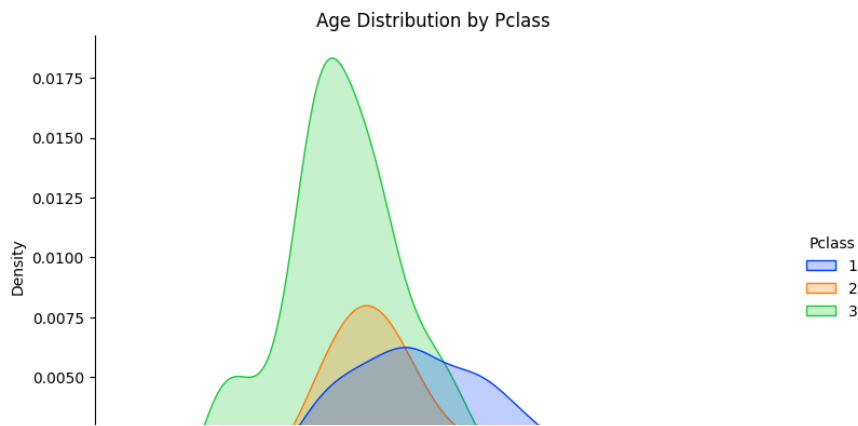
```
sb.displot(data=dataset, x='Age', hue='Survived', kind='kde', fill=True, palette=sb.color_palette('bright')[:3], height=5, aspect=1.5)
plt.title('Age Distribution by Survived')
plt.show()
```

<ipython-input-8-a1eef0a00695>:1: UserWarning: The palette list has more values (3) t
sb.displot(data=dataset, x='Age', hue='Survived', kind='kde', fill=True, palette=sb



```
sb.displot(data=dataset, x='Age', hue='Pclass', kind='kde', fill=True, palette=sb.color_palette('bright')[:3], height=5, aspect=1.5) plt.title('Age Distribution by Pclass') plt.show()
```

```
sb.displot(data=dataset, x='Age', hue='Pclass', kind='kde', fill=True, palette=sb.color_palette('bright')[:3], height=5, aspect=1.5)
plt.title('Age Distribution by Pclass')
plt.show()
```

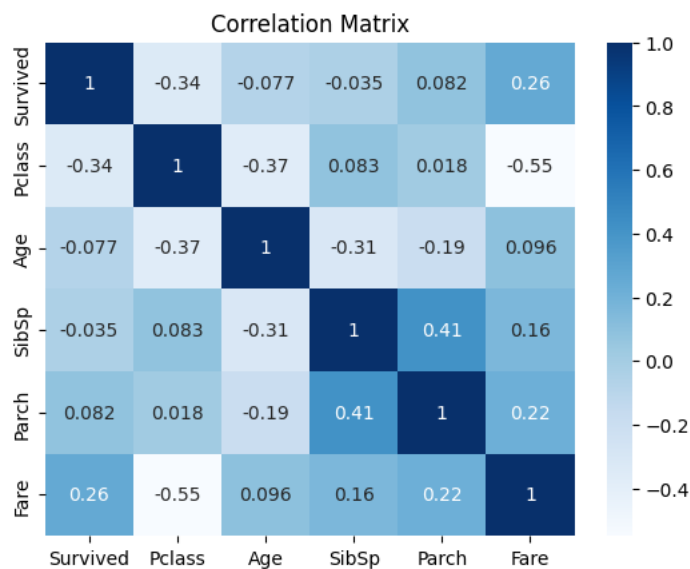


```
df_corr = dataset.drop(['PassengerId'], axis = 1) sb.heatmap(df_corr.corr(),annot = True, cmap = plt.cm.Blues) plt.title('Correlation Matrix')
```



```
df_corr = dataset.drop(['PassengerId'], axis = 1)
sb.heatmap(df_corr.corr(),annot = True, cmap = plt.cm.Blues)
plt.title('Correlation Matrix')
```

```
<ipython-input-10-8aba0ae1a2fd>:2: FutureWarning: The default value of numeric_only i
sb.heatmap(df_corr.corr(),annot = True, cmap = plt.cm.Blues)
Text(0.5, 1.0, 'Correlation Matrix')
```



```
train_data = dataset.copy()
train_data["Age"].fillna(dataset["Age"].median(skipna=True), inplace=True)
train_data["Embarked"].fillna(dataset["Embarked"].value_counts().idxmax(), inplace=True)
train_data.drop('Cabin', axis=1, inplace=True)
```

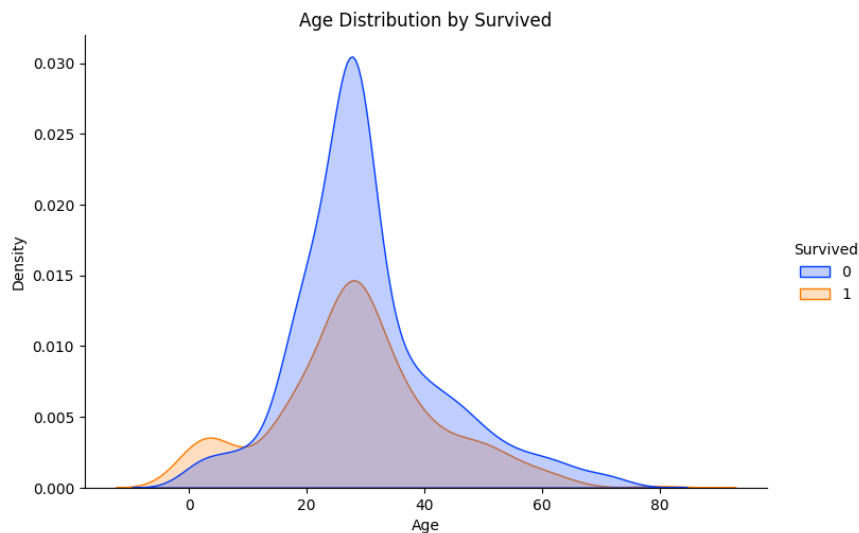
```
train_data.isnull().sum()
```

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

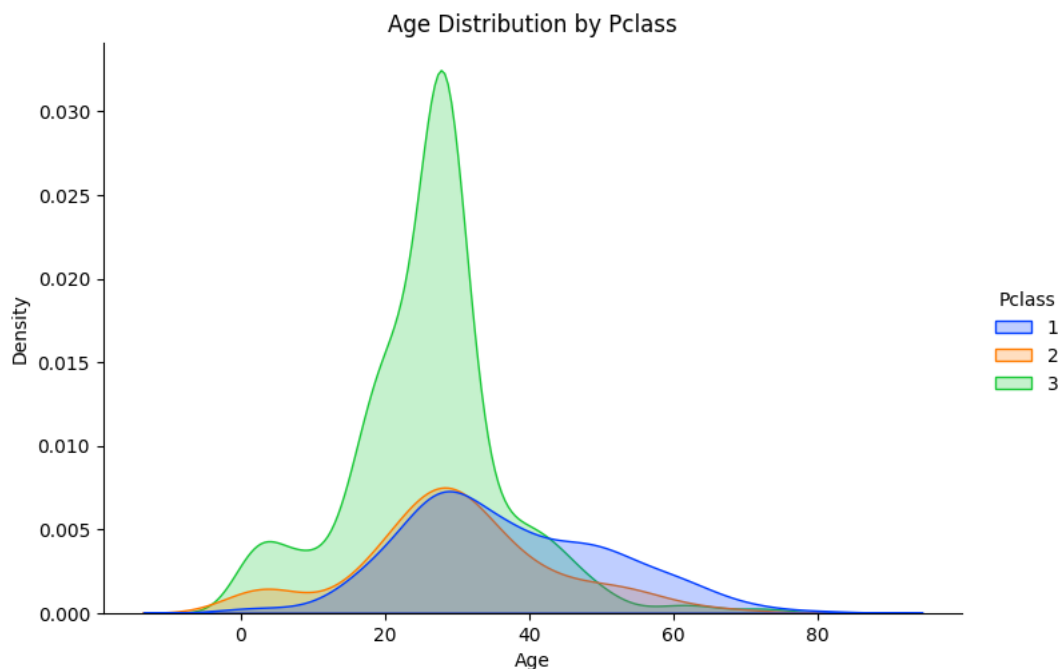
```
sb.displot(data=train_data, x='Age', hue='Survived', kind='kde', fill=True, palette=sb.color_palette('bright')[:3], height=5, aspect=1.5)
plt.title('Age Distribution by Survived')
plt.show()
```



```
<ipython-input-13-681680077703>:1: UserWarning: The palette list has more values (3)
sb.displot(data=train_data, x='Age', hue='Survived', kind='kde', fill=True, palette
```



```
sb.displot(data=train_data, x='Age', hue='Pclass', kind='kde', fill=True, palette=sb.color_palette('bright')[:3], height=5, aspect=1.5)
plt.title('Age Distribution by Pclass')
plt.show()
```



```
from sklearn.model_selection import train_test_split, cross_val_score

x = pd.DataFrame(np.c_[train_data['Pclass'], train_data['Fare']], columns=['Age', 'Fare'])
y = train_data['Survived']

from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
y = le.fit_transform(y)

x_train, x_test, y_train, y_test = train_test_split(x, y, random_state=0, test_size=0.3)

from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score

lr = LogisticRegression()

# lr.fit(x_train, y_train)
# y_predRF = lr.predict(x_test)

lr.fit(x_train, y_train)
y_pred = lr.predict(x_test)
print("Logistic Regression", accuracy_score(y_test, y_pred))
```

Logistic Regression 0.6940298507462687

