In [1]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import math
df = pd.read_csv("Titanic_trainn.csv")
df.head(10)
```

Out[1]:

	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
5	6	0	3	Moran, Mr. James	male	NaN	0	0	330877	8.4583	N
6	7	0	1	McCarthy, Mr. Timothy J	male	54.0	0	0	17463	51.8625	I

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Ca
7	8	0	3	Palsson, Master. Gosta Leonard	male	2.0	3	1	349909	21.0750	N
8	9	1	3	Johnson, Mrs. Oscar W (Elisabeth Vilhelmina Berg)	female	27.0	0	2	347742	11.1333	N
9	10	1	2	Nasser, Mrs. Nicholas (Adele Achem)	female	14.0	1	0	237736	30.0708	N

```
In [2]: 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 [3]:
```
df.isnull().sum()
```

Out[3]:
```
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
```

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

Out[4]:

	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

In [5]:
```
df.drop("Cabin",axis=1,inplace=True)     # As cabin column contain many
 NAN values so drop it first
df.dropna(inplace=True)
df.head(5)
```

Out[5]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	En
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	
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	
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	

In [6]:

```
sex = pd.get_dummies(df["Sex"],drop_first=True)                    # Creat
e dummies for sex, embark and Pclass column
embark = pd.get_dummies(df["Embarked"],drop_first=True)
Pclass = pd.get_dummies(df["Pclass"],drop_first=True)
#Drop the variables with no use and whose dummies created
df.drop(["Name","Sex","Embarked","Ticket","Pclass"],axis=1,inplace=True
)
df.head(5)
```

Out[6]:

	PassengerId	Survived	Age	SibSp	Parch	Fare
0	1	0	22.0	1	0	7.2500
1	2	1	38.0	1	0	71.2833
2	3	1	26.0	0	0	7.9250

	PassengerId	Survived	Age	SibSp	Parch	Fare
3	4	1	35.0	1	0	53.1000
4	5	0	35.0	0	0	8.0500

In [7]:

```
df = pd.concat([df,sex,embark,Pclass],axis=1)
df.head(5)
```

Out[7]:

	PassengerId	Survived	Age	SibSp	Parch	Fare	male	Q	S	2	3
0	1	0	22.0	1	0	7.2500	1	0	1	0	1
1	2	1	38.0	1	0	71.2833	0	0	0	0	0
2	3	1	26.0	0	0	7.9250	0	0	1	0	1
3	4	1	35.0	1	0	53.1000	0	0	1	0	0
4	5	0	35.0	0	0	8.0500	1	0	1	0	1

In [8]:

```
df.drop("PassengerId",axis=1,inplace=True)
df.head(5)
```

Out[8]:

	Survived	Age	SibSp	Parch	Fare	male	Q	S	2	3
0	0	22.0	1	0	7.2500	1	0	1	0	1
1	1	38.0	1	0	71.2833	0	0	0	0	0
2	1	26.0	0	0	7.9250	0	0	1	0	1
3	1	35.0	1	0	53.1000	0	0	1	0	0
4	0	35.0	0	0	8.0500	1	0	1	0	1

In [9]:

```
x = df.drop("Survived",axis=1)
y = df["Survived"]
```

In [10]:

```
from sklearn.linear_model import LogisticRegression
```

```
rissh= LogisticRegression()
rissh.fit(x,y)
```

```
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\linear_model\_logist
ic.py:762: 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[10]: LogisticRegression()
```

```
In [12]: predict = rissh.predict(x)
         from sklearn.metrics import classification_report
         classification_report(y,predict)
```

```
Out[12]: '              precision    recall  f1-score   support\n\n           0
               0.81      0.88      0.85       424\n           1       0.80
         0.70      0.75       288\n\n    accuracy                           0.81
                712\n   macro avg       0.81      0.79      0.80       712\nweig
         hted avg       0.81      0.81      0.81       712\n'
```

```
In [13]: from sklearn.metrics import confusion_matrix    # Confusion matrix
         confusion_matrix(y,predict)
```

```
Out[13]: array([[373,  51],
                [ 85, 203]], dtype=int64)
```

```
In [14]: from sklearn.metrics import accuracy_score             # Accuracy of mo
         del
         accuracy_score(y,predict)
```

```
Out[14]: 0.8089887640449438
```

In [15]:

```
print("The intercept is",rissh.intercept_)
```

```
The intercept is [3.8138798]
```

In [16]:

```
print("The coeffs of different parameters",rissh.coef_)
```

```
The coeffs of different parameters [[-0.03954496 -0.37442337 -0.0865595
3  0.00350953 -2.40146613 -0.49214919
  -0.42011897 -0.72821738 -1.97138663]]
```

In [ ]:

```

```