In [1]:
```
#Loading the Data

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

from sklearn import metrics
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split

titanic = pd.read_csv("Titanic.csv")

titanic.head()
```

Out[1]:

	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	Heikkinen, 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
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN	S

In [2]:
```
# Drop features that do not seem to add any value to our model
titanic.drop(['PassengerId', 'Name', 'Ticket', 'Cabin'], axis=1, inplace=True)
```

In [3]:
```
# Create categorical dummies for the embarkment ports
ports = pd.get_dummies(titanic.Embarked, prefix='Embarked')
ports.head()
```

Out[3]:

	Embarked_C	Embarked_Q	Embarked_S
0	0	0	1
1	1	0	0
2	0	0	1
3	0	0	1
4	0	0	1

In [4]:
```
titanic = titanic.join(ports)
titanic.drop(['Embarked'], axis=1, inplace=True)
```

In [5]:
```
#Transform gender names to binaries
titanic.Sex = titanic.Sex.map({'male': 0, 'female': 1})
```

In [6]:
```
#Replace missing values
titanic[pd.isnull(titanic).any(axis=1)]
```

Out[6]:

	Survived	Pclass	Sex	Age	SibSp	Parch	Fare	Embarked_C	Embarked_Q	Embarked_S
5	0	3	0	NaN	0	0	8.4583	0	1	0
17	1	2	0	NaN	0	0	13.0000	0	0	1
19	1	3	1	NaN	0	0	7.2250	1	0	0
26	0	3	0	NaN	0	0	7.2250	1	0	0
28	1	3	1	NaN	0	0	7.8792	0	1	0
...	...	...	...	...	...	...	...	...	...	...
859	0	3	0	NaN	0	0	7.2292	1	0	0
863	0	3	1	NaN	8	2	69.5500	0	0	1
868	0	3	0	NaN	0	0	9.5000	0	0	1
878	0	3	0	NaN	0	0	7.8958	0	0	1
888	0	3	1	NaN	1	2	23.4500	0	0	1

177 rows × 10 columns

In [7]:
```
titanic.Age.fillna(titanic.Age.mean(), inplace=True)
```

In [8]:
```
#Train and Test split
y = titanic.Survived.copy()
X = titanic.drop(['Survived'], axis=1)
```

In [9]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25, random_state=123)
```

In [11]:
```
model = LogisticRegression()
model.fit(X_train, y_train)
```

```
C:\Users\rainy\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn\linear_model\_logistic.py:458: 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[11]:
```
▼ LogisticRegression

LogisticRegression()
```

In [12]:
```
y_pred = pd.Series(model.predict(X_test))
y_test = y_test.reset_index(drop=True)
z = pd.concat([y_test, y_pred], axis=1)
z.columns = ['True', 'Prediction']
z.head()
```

Out[12]:

	True	Prediction
0	1	1
1	0	0
2	0	1
3	0	0
4	0	0

In [13]:
```
#To evaluate the entire test set, we can use the metrics module from the scikit-learn package.
print("Accuracy:", metrics.accuracy_score(y_test, y_pred))
print("Precision:", metrics.precision_score(y_test, y_pred))
print("Recall:", metrics.recall_score(y_test, y_pred))
```

```
Accuracy: 0.8071748878923767
Precision: 0.7594936708860076
Recall: 0.7142857142857143
```

In [ ]:
```

```