

# PROJECT: Predicting Titanic Survival (Classification Problem)

## Step 1: Import Libraries

```
import pandas as pd

import numpy as np

import seaborn as sns

import matplotlib.pyplot as plt


from sklearn.model_selection import train_test_split

from sklearn.linear_model import LogisticRegression

from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
```

## Step 2: Load Dataset

```
# Load Titanic dataset

df = sns.load_dataset('titanic')

df.head()
```

## OUTPUT:

	survived	pclass	sex	age	sibsp	parch	fare	embarked	class	who	adult_male	deck	embark_town	alive	alone
0	0	3	male	22.0	1	0	7.2500	S	Third	man	True	NaN	Southampton	no	False
1	1	1	female	38.0	1	0	71.2833	C	First	woman	False	C	Cherbourg	yes	False
2	1	3	female	26.0	0	0	7.9250	S	Third	woman	False	NaN	Southampton	yes	True
3	1	1	female	35.0	1	0	53.1000	S	First	woman	False	C	Southampton	yes	False
4	0	3	male	35.0	0	0	8.0500	S	Third	man	True	NaN	Southampton	no	True



### Step 3: Data Preprocessing

**# Drop rows with missing values**

```
df = df.dropna(subset=['age', 'embarked', 'fare'])
```

**# Encode categorical variables**

```
df['sex'] = df['sex'].map({'male': 0, 'female': 1})
```

```
df['embarked'] = df['embarked'].map({'S': 0, 'C': 1, 'Q': 2})
```

**# Select features and target**

```
features = ['pclass', 'sex', 'age', 'fare', 'embarked']
```

```
X = df[features]
```

```
y = df['survived']
```

#### OUTPUT:

```
<ipython-input-3-4142678790>:5: SettingWithCopyWarning:  
A value is trying to be set on a copy of a slice from a DataFrame.  
Try using .loc[row_indexer,col_indexer] = value instead
```

See the caveats in the documentation: [https://pandas.pydata.org/pandas-docs/stable/user\\_guide/indexing.html#returning-a-view-versus-a-copy](https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy)

```
df['sex'] = df['sex'].map({'male': 0, 'female': 1})
```

```
<ipython-input-3-4142678790>:6: SettingWithCopyWarning:  
A value is trying to be set on a copy of a slice from a DataFrame.  
Try using .loc[row_indexer,col_indexer] = value instead
```

See the caveats in the documentation: [https://pandas.pydata.org/pandas-docs/stable/user\\_guide/indexing.html#returning-a-view-versus-a-copy](https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy)

```
df['embarked'] = df['embarked'].map({'S': 0, 'C': 1, 'Q': 2})
```

---

### Step 4: Split Data

**# Split into training and testing**

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```



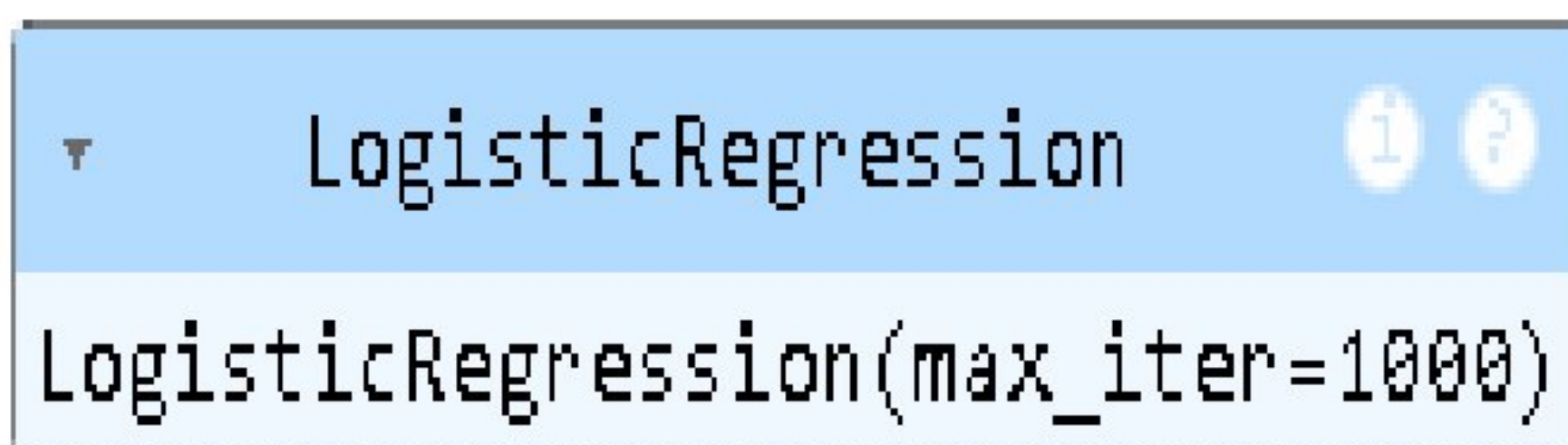
## Step 5: Build and Train Model

# Initialize and train model

```
model = LogisticRegression(max_iter=1000)
```

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

OUTPUT:



```
LogisticRegression(max_iter=1000)
```

## Step 6: Evaluate Model

# Make predictions

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

# Evaluation metrics

```
print("Accuracy:", accuracy_score(y_test, y_pred))
```

```
print("\nClassification Report:\n", classification_report(y_test, y_pred))
```

# Confusion Matrix

```
sns.heatmap(confusion_matrix(y_test, y_pred), annot=True, fmt="d",  
cmap="Blues")
```

```
plt.title("Confusion Matrix")
```


```
plt.xlabel("Predicted")
```

```
plt.ylabel("Actual")
```

```
plt.show()
```

OUTPUT:

Accuracy: 0.7902097902097902

 Classification Report:

	precision	recall	f1-score	support
0	0.76	0.91	0.83	80
1	0.85	0.63	0.73	63
accuracy			0.79	143
macro avg	0.81	0.77	0.78	143
weighted avg	0.80	0.79	0.78	143

