In [1]:
```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, classification_report, confusion
```

In [2]:
```
data = pd.read_csv("C:\\Users\\Lenovo\\Downloads\\Churn_Modelling.csv")
data
```

Out[2]:

	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	
0	1	15634602	Hargrave	619	France	Female	42	2	
1	2	15647311	Hill	608	Spain	Female	41	1	8
2	3	15619304	Onio	502	France	Female	42	8	15
3	4	15701354	Boni	699	France	Female	39	1	
4	5	15737888	Mitchell	850	Spain	Female	43	2	12
...	...	...	...	...	...	...	...	...	
9995	9996	15606229	Obijiaku	771	France	Male	39	5	
9996	9997	15569892	Johnstone	516	France	Male	35	10	5
9997	9998	15584532	Liu	709	France	Female	36	7	
9998	9999	15682355	Sabbatini	772	Germany	Male	42	3	7
9999	10000	15628319	Walker	792	France	Female	28	4	13

10000 rows × 14 columns

In [3]:
```
data = data.drop(columns=['RowNumber', 'CustomerId', 'Surname'])
data
```

Out[3]:

	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard
0	619	France	Female	42	2	0.00	1	1
1	608	Spain	Female	41	1	83807.86	1	0
2	502	France	Female	42	8	159660.80	3	1
3	699	France	Female	39	1	0.00	2	0
4	850	Spain	Female	43	2	125510.82	1	1
...	...	...	...	...	...	...	...	...
9995	771	France	Male	39	5	0.00	2	1
9996	516	France	Male	35	10	57369.61	1	1
9997	709	France	Female	36	7	0.00	1	0
9998	772	Germany	Male	42	3	75075.31	2	1
9999	792	France	Female	28	4	130142.79	1	1

10000 rows × 11 columns

In [4]:
```
data = pd.get_dummies(data, columns=['Geography', 'Gender'], drop_first=True
```

In [5]:
```
X = data.drop(columns=['Exited'])
```

In [6]:
```
y = data['Exited']
```

In [7]:
```
X
```

Out[7]:

	CreditScore	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	Est
0	619	42	2	0.00	1	1	1	
1	608	41	1	83807.86	1	0	1	
2	502	42	8	159660.80	3	1	0	
3	699	39	1	0.00	2	0	0	
4	850	43	2	125510.82	1	1	1	
...	...	...	...	...	...	...	...	
9995	771	39	5	0.00	2	1	0	
9996	516	35	10	57369.61	1	1	1	
9997	709	36	7	0.00	1	0	1	
9998	772	42	3	75075.31	2	1	0	
9999	792	28	4	130142.79	1	1	0	

10000 rows × 11 columns

In [8]:
```
y
```

Out[8]:
```
0       1
1       0
2       1
3       0
4       0
       ..
9995    0
9996    0
9997    1
9998    1
9999    0
Name: Exited, Length: 10000, dtype: int64
```

In [9]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, ran
```

In [10]:
```
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
```

In [11]:
```
model = LogisticRegression(random_state=42)
model.fit(X_train, y_train)
```

Out[11]:
```
LogisticRegression(random_state=42)
```
In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [12]:
```
y_pred = model.predict(X_test)
y_pred
```

Out[12]:
```
array([0, 0, 0, ..., 0, 0, 0], dtype=int64)
```

In [13]:
```
accuracy = accuracy_score(y_test, y_pred)
print(f"Accuracy: {accuracy:.2f}")
```

```
Accuracy: 0.81
```

In [14]:
```
print(classification_report(y_test, y_pred))
print("Confusion Matrix:")
print(confusion_matrix(y_test, y_pred))
```

```
              precision    recall  f1-score   support

           0       0.83      0.96      0.89      1607
           1       0.55      0.20      0.29       393

    accuracy                           0.81      2000
   macro avg       0.69      0.58      0.59      2000
weighted avg       0.78      0.81      0.77      2000

Confusion Matrix:
[[1543   64]
 [ 314   79]]
```