LAB ASSIGMENT-5

```
import pandas as pd
import numpy as np

from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, confusion_matrix, classificati
```

Load Dataset

```
# Load Titanic training dataset
data = pd.read_csv('/content/train.csv')

# Display first 5 rows
data.head()
```

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450

Next steps: Generate code with data New interactive sheet

Select Required Features

```
# Select important features
data = data[['Survived', 'Pclass', 'Sex', 'Age', 'Fare']]

data.head()
```

	Survived	Pclass	Sex	Age	Fare
0	0	3	male	22.0	7.2500
1	1	1	female	38.0	71.2833
2	1	3	female	26.0	7.9250
3	1	1	female	35.0	53.1000
4	0	3	male	35.0	8.0500

Next steps: Generate code with data New interactive sheet

Handle Missing Values

```
# Fill missing Age values with median
data['Age'].fillna(data['Age'].median(), inplace=True)

# Check missing values
data.isnull().sum()
```

```
/tmp/ipython-input-1108450853.py:2: FutureWarning: A value is trying to be set o
The behavior will change in pandas 3.0. This inplace method will never work beca

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.met


  data['Age'].fillna(data['Age'].median(), inplace=True)
```

	0
Survived	0
Pclass	0
Sex	0
Age	0
Fare	0

dtype: int64

Encode Categorical Variable

```
# Encode Sex column (male=0, female=1)
data['Sex'] = data['Sex'].map({'male': 0, 'female': 1})

data.head()
```

	Survived	Pclass	Sex	Age	Fare
0	0	3	0	22.0	7.2500
1	1	1	1	38.0	71.2833
2	1	3	1	26.0	7.9250
3	1	1	1	35.0	53.1000
4	0	3	0	35.0	8.0500

Next steps: Generate code with `data` New interactive sheet

Split Features and Target

```
X = data.drop('Survived', axis=1)
y = data['Survived']
```

Train & Evaluate Model (70/30 Split)

```
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.30, random_state=42
)

model = LogisticRegression(max_iter=1000)
model.fit(X_train, y_train)

y_pred = model.predict(X_test)

print("70/30 Split Results")
print("Accuracy:", accuracy_score(y_test, y_pred))
print("Confusion Matrix:\n", confusion_matrix(y_test, y_pred))
print("Classification Report:\n", classification_report(y_test, y_pred))
```

```
70/30 Split Results
Accuracy: 0.7947761194029851
Confusion Matrix:
 [[134  23]
 [ 32  79]]
Classification Report:
              precision    recall  f1-score   support

           0       0.81      0.85      0.83       157
           1       0.77      0.71      0.74       111

    accuracy                           0.79       268
   macro avg       0.79      0.78      0.79       268
weighted avg       0.79      0.79      0.79       268
```

: Train & Evaluate Model (80/20 Split)

```
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.20, random_state=42
)

model = LogisticRegression(max_iter=1000)
model.fit(X_train, y_train)

y_pred = model.predict(X_test)

print("80/20 Split Results")
print("Accuracy:", accuracy_score(y_test, y_pred))
print("Confusion Matrix:\n", confusion_matrix(y_test, y_pred))
print("Classification Report:\n", classification_report(y_test, y_pred))
```

```
80/20 Split Results
Accuracy: 0.8044692737430168
Confusion Matrix:
 [[90 15]
 [20 54]]
Classification Report:
              precision    recall  f1-score   support

           0       0.82      0.86      0.84       105
           1       0.78      0.73      0.76        74

    accuracy                           0.80       179
   macro avg       0.80      0.79      0.80       179
weighted avg       0.80      0.80      0.80       179
```

Train & Evaluate Model (90/10 Split)

```
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.10, random_state=42
)

model = LogisticRegression(max_iter=1000)
model.fit(X_train, y_train)

y_pred = model.predict(X_test)

print("90/10 Split Results")
print("Accuracy:", accuracy_score(y_test, y_pred))
print("Confusion Matrix:\n", confusion_matrix(y_test, y_pred))
print("Classification Report:\n", classification_report(y_test, y_pred))
```

```
90/10 Split Results
Accuracy: 0.8555555555555555
Confusion Matrix:
 [[47  7]
 [ 6 30]]
Classification Report:
              precision    recall  f1-score   support

           0       0.89      0.87      0.88        54
           1       0.81      0.83      0.82        36

    accuracy                           0.86        90
   macro avg       0.85      0.85      0.85        90
weighted avg       0.86      0.86      0.86        90
```