```python
# 1. Import Libraries
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder, StandardScaler
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
```

```python
# 2. Load Dataset
data = pd.read_csv('loan_data.csv')  # Replace with your dataset file
```

```python
# 3. Handle Missing Values
data['Gender'].fillna(data['Gender'].mode()[0], inplace=True)
data['Married'].fillna(data['Married'].mode()[0], inplace=True)
data['Dependents'].fillna(data['Dependents'].mode()[0], inplace=True)
data['Self_Employed'].fillna(data['Self_Employed'].mode()[0], inplace=True)
data['Credit_History'].fillna(data['Credit_History'].mode()[0], inplace=True)
data['LoanAmount'].fillna(data['LoanAmount'].median(), inplace=True)
data['Loan_Amount_Term'].fillna(data['Loan_Amount_Term'].mode()[0], inplace=True)
```

Show hidden output

```python
# 4. Encode Categorical Features
cols = ['Gender', 'Married', 'Education', 'Self_Employed', 'Property_Area', 'Loan_Status', 'Dependents']
le = LabelEncoder()
for col in cols:
    data[col] = le.fit_transform(data[col])
```

```python
# 5. Feature Selection
X = data.drop(['Loan_ID', 'Loan_Status'], axis=1)
y = data['Loan_Status']
```

```python
# 7. Train-Test Split
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0.2, random_state=42)
```

```python
# 9. Evaluate Model
y_pred = model.predict(X_test)

print("Accuracy:", accuracy_score(y_test, y_pred))
print("Classification Report:\n", classification_report(y_test, y_pred))
print("Confusion Matrix:\n", confusion_matrix(y_test, y_pred))
```

```python
# 10. Predict New Sample (scaled)
new_input = pd.DataFrame([{
    'Gender': 1,  # Male
    'Married': 1,
    'Dependents': 0,
    'Education': 0,
    'Self_Employed': 0,
    'ApplicantIncome': 4000,
    'CoapplicantIncome': 1500,
    'LoanAmount': 130,
    'Loan_Amount_Term': 360,
    'Credit_History': 1.0,
    'Property_Area': 2  # Urban
}])

# Scale new input
new_input_scaled = scaler.transform(new_input)

# Predict
pred = model.predict(new_input_scaled)
print("Prediction:", "Not Defaulted" if pred[0] == 1 else "Defaulted")
```