```python
# Import necessary libraries
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from sklearn.impute import SimpleImputer
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, classification_report

# Load the dataset
data = pd.read_csv('/content/loanapprovalpredict.csv')  # Replace
'LoanApprovalPrediction.csv' with your dataset filename

# Data preprocessing
# Encode categorical features using LabelEncoder
label_encoder = LabelEncoder()
data['Gender'] = label_encoder.fit_transform(data['Gender'])
data['Married'] = label_encoder.fit_transform(data['Married'])
data['Dependents'] = label_encoder.fit_transform(data['Dependents'])
data['Education'] = label_encoder.fit_transform(data['Education'])
data['Self_Employed'] = label_encoder.fit_transform(data['Self_Employed'])
data['Property_Area'] = label_encoder.fit_transform(data['Property_Area'])
data['Loan_Status'] = label_encoder.fit_transform(data['Loan_Status'])

# Define the features (X) and the target variable (y)
```


```python
# Import necessary libraries
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from sklearn.impute import SimpleImputer
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, classification_report

# Load the dataset
data = pd.read_csv('/content/loanapprovalpredict.csv')  # Replace
'LoanApprovalPrediction.csv' with your dataset filename
```

```
# Data preprocessing
# Encode categorical features using LabelEncoder
label_encoder = LabelEncoder()
data['Gender'] = label_encoder.fit_transform(data['Gender'])
data['Married'] = label_encoder.fit_transform(data['Married'])
data['Dependents'] = label_encoder.fit_transform(data['Dependents'])
data['Education'] = label_encoder.fit_transform(data['Education'])
data['Self_Employed'] = label_encoder.fit_transform(data['Self_Employed'])
data['Property_Area'] = label_encoder.fit_transform(data['Property_Area'])
data['Loan_Status'] = label_encoder.fit_transform(data['Loan_Status'])

# Define the features (X) and the target variable (y)
X = data[['Gender', 'Married', 'Dependents', 'Education', 'Self_Employed',
'ApplicantIncome', 'CoapplicantIncome', 'LoanAmount', 'Loan_Amount_Term',
'Credit_History', 'Property_Area']]
y = data['Loan_Status']

# Handle missing values using SimpleImputer
imputer = SimpleImputer(strategy='mean')  # You can choose a different
strategy ('mean', 'median', 'most_frequent', etc.)
X = imputer.fit_transform(X)

# Split the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)

# Create and train the logistic regression model
model = LogisticRegression()
model.fit(X_train, y_train)

# User input for prediction
print("Please enter the following details to check loan sanction status:")
gender = int(input("Gender (0 for Female, 1 for Male): "))
married = int(input("Married (0 for No, 1 for Yes): "))
dependents = int(input("Dependents (0, 1, 2, 3+): "))
education = int(input("Education (0 for Not Graduate, 1 for Graduate): "))
self_employed = int(input("Self Employed (0 for No, 1 for Yes): "))
applicant_income = float(input("Applicant Income: "))
coapplicant_income = float(input("Coapplicant Income: "))
```

```
loan_amount = float(input("Loan Amount: "))
loan_amount_term = float(input("Loan Amount Term: "))
credit_history = int(input("Credit History (0 or 1): "))
property_area = int(input("Property Area (0 for Urban, 1 for Semiurban, 2 for
Rural): "))

user_input = [[gender, married, dependents, education, self_employed,
applicant_income, coapplicant_income, loan_amount, loan_amount_term,
credit_history, property_area]]

# Make prediction on the user input
user_prediction = model.predict(user_input)

if user_prediction[0] == 1:
    print("Congratulations! Your loan will be sanctioned.")
else:
    print("Sorry, your loan will not be sanctioned.")

# Make predictions on the test data
y_pred = model.predict(X_test)

# Evaluate the model
accuracy = accuracy_score(y_test, y_pred)
classification_rep = classification_report(y_test, y_pred)

print(f'Accuracy: {accuracy}')
print(f'Classification Report:\n{classification_rep}')
```

Output

```
/usr/local/lib/python3.10/dist-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(
Please enter the following details to check loan sanction status:
Gender (0 for Female, 1 for Male): 1
Married (0 for No, 1 for Yes): 1
Dependents (0, 1, 2, 3+): 1
Education (0 for Not Graduate, 1 for Graduate): 1
Self Employed (0 for No, 1 for Yes): 0
Applicant Income: 6000
Coapplicant Income: 2000
Loan Amount: 200
Loan Amount Term: 360
Credit History (0 or 1): 1
Property Area (0 for Urban, 1 for Semiurban, 2 for Rural): 0
Congratulations! Your loan will be sanctioned.
Accuracy: 0.7886178861788617
Classification Report:
              precision    recall  f1-score   support

           0       0.95      0.42      0.58        43
           1       0.76      0.99      0.86        80

    accuracy                           0.79       123
   macro avg       0.85      0.70      0.72       123
weighted avg       0.83      0.79      0.76       123
```

Output-2:

```
/usr/local/lib/python3.10/dist-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(
Please enter the following details to check loan sanction status:
Gender (0 for Female, 1 for Male): 1
Married (0 for No, 1 for Yes): 0
Dependents (0, 1, 2, 3+): 1
Education (0 for Not Graduate, 1 for Graduate): 0
Self Employed (0 for No, 1 for Yes): 0
Applicant Income: 9000
Coapplicant Income: 5000
Loan Amount: 250000
Loan Amount Term: 240
Credit History (0 or 1): 0
Property Area (0 for Urban, 1 for Semiurban, 2 for Rural): 1
Sorry, your loan will not be sanctioned.
Accuracy: 0.7886178861788617
Classification Report:
              precision    recall  f1-score   support

           0       0.95      0.42      0.58        43
           1       0.76      0.99      0.86        80

    accuracy                           0.79       123
   macro avg       0.85      0.70      0.72       123
weighted avg       0.83      0.79      0.76       123
```

Output-3:

```
/usr/local/lib/python3.10/dist-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(
Please enter the following details to check loan sanction status:
Gender (0 for Female, 1 for Male): 1
Married (0 for No, 1 for Yes): 1
Dependents (0, 1, 2, 3+): 2
Education (0 for Not Graduate, 1 for Graduate): 1
Self Employed (0 for No, 1 for Yes): 1
Applicant Income: 20000
Coapplicant Income: 15000
Loan Amount: 350000
Loan Amount Term: 480
Credit History (0 or 1): 1
Property Area (0 for Urban, 1 for Semiurban, 2 for Rural): 0
Sorry, your loan will not be sanctioned.
Accuracy: 0.7886178861788617
Classification Report:
              precision    recall  f1-score   support

           0       0.95      0.42      0.58        43
           1       0.76      0.99      0.86        80

    accuracy                           0.79       123
   macro avg       0.85      0.70      0.72       123
weighted avg       0.83      0.79      0.76       123
```