```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.preprocessing import StandardScaler, LabelEncoder
from sklearn.metrics import (
    accuracy_score,
    confusion_matrix,
    ConfusionMatrixDisplay,
    classification_report
)
from imblearn.over_sampling import SMOTE
```

```
# Load the dataset
data = pd.read_csv('/content/loan_approval_dataset_500.csv')

# Clean column names and remove non-useful columns
data.columns = data.columns.str.strip()
data = data.drop('loan_id', axis=1)
```

```
# Step 3: Clean and encode categorical variables properly

# Strip spaces and make sure these columns are strings
data['education'] = data['education'].astype(str).str.strip()
data['self_employed'] = data['self_employed'].astype(str).str.strip()
data['loan_status'] = data['loan_status'].astype(str).str.strip()

# Fit LabelEncoders on raw string data
le_education = LabelEncoder()
le_self_employed = LabelEncoder()
le_loan_status = LabelEncoder()

data['education'] = le_education.fit_transform(data['education'])
data['self_employed'] = le_self_employed.fit_transform(data['self_employed'])
data['loan_status'] = le_loan_status.fit_transform(data['loan_status'])
```

```
# Separate the dataset into features (X) and target label (y)
X = data.drop('loan_status', axis=1)  # All columns except 'loan_status'
y = data['loan_status']               # The target column
```

```
from imblearn.over_sampling import SMOTE

# Initialize SMOTE
smote = SMOTE(random_state=42)

# Apply SMOTE to resample the data
X_resampled, y_resampled = smote.fit_resample(X, y)
```

```
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler

# Split the resampled data into training and testing sets (80% train, 20% test)
X_train, X_test, y_train, y_test = train_test_split(
    X_resampled, y_resampled, test_size=0.2, random_state=42
)

# Initialize the scaler
scaler = StandardScaler()

# Fit the scaler on training data and transform both train and test sets
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

```
from sklearn.tree import DecisionTreeClassifier

# Initialize the Decision Tree model with hyperparameters
model = DecisionTreeClassifier(
    max_depth=5,            # Limit tree depth to avoid overfitting
    min_samples_split=10,   # Minimum samples needed to split an internal node
    min_samples_leaf=5,     # Minimum samples needed in a leaf node
    random_state=42         # For reproducibility
```

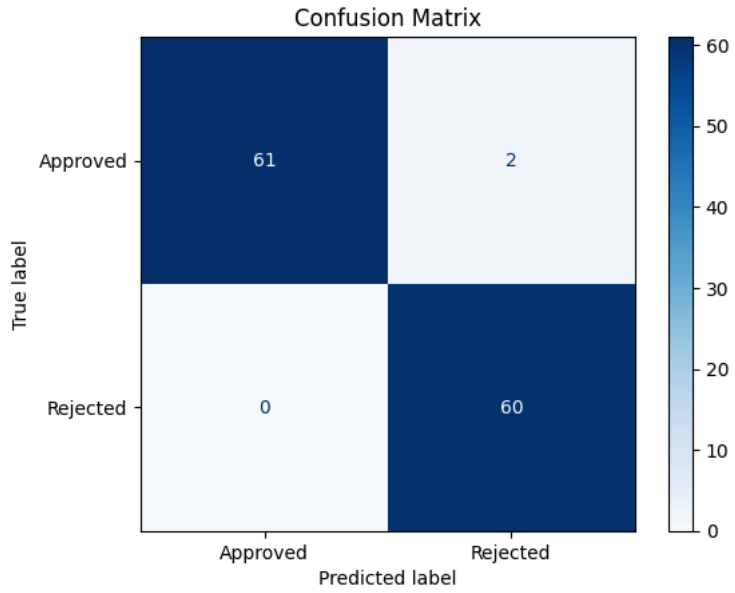
```
)

# Train (fit) the model on the training data
model.fit(X_train_scaled, y_train)
```

```
DecisionTreeClassifier(max_depth=5, min_samples_leaf=5, min_samples_split=10,
                       random_state=42)
```

```
from sklearn.metrics import accuracy_score

# Use the trained model to predict the labels for the test set
y_pred = model.predict(X_test_scaled)

# Calculate accuracy by comparing predicted and true labels
accuracy = accuracy_score(y_test, y_pred)

print(f"Model Accuracy: {accuracy:.2f}")
```

```
Model Accuracy: 0.98
```

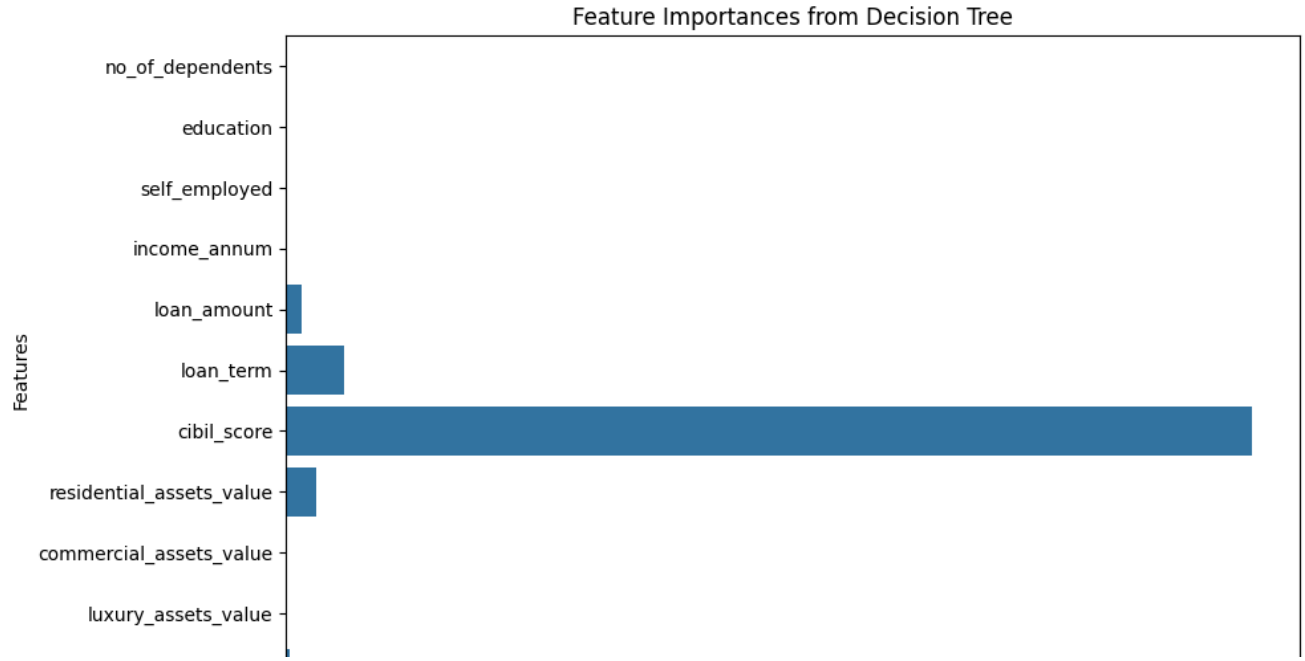
```
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay, classification_report
import matplotlib.pyplot as plt

# Compute the confusion matrix
cm = confusion_matrix(y_test, y_pred)

# Create and display the confusion matrix plot with labels
disp = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=le_loan_status.classes_)
disp.plot(cmap=plt.cm.Blues)
plt.title("Confusion Matrix")
plt.show()

# Print the classification report (precision, recall, f1-score for each class)
print("\nClassification Report:")
print(classification_report(y_test, y_pred, target_names=le_loan_status.classes_))
```




```
Classification Report:
              precision    recall  f1-score   support

    Approved       1.00      0.97      0.98        63
    Rejected       0.97      1.00      0.98        60

    accuracy                           0.98       123
   macro avg       0.98      0.98      0.98       123
weighted avg       0.98      0.98      0.98       123
```

```
import matplotlib.pyplot as plt
import seaborn as sns

# Get feature importances from the model
```

```
importances = model.feature_importances_
features = X.columns

# Create a bar plot showing the importance of each feature
plt.figure(figsize=(10, 6))
sns.barplot(x=importances, y=features)
plt.title("Feature Importances from Decision Tree")
plt.xlabel("Importance Score")
plt.ylabel("Features")
plt.tight_layout()
plt.show()
```




```
def predict_loan_status():
    print("\nEnter the following details for loan prediction:")

    try:
        no_of_dependents = int(input("Number of dependents (0-5): ").strip())
        education_input = input("Education (Graduate/Not Graduate): ").strip()
        self_employed_input = input("Self Employed (Yes/No): ").strip()
        income_annum = float(input("Annual Income: ").strip())
        loan_amount = float(input("Loan Amount: ").strip())
        loan_term = int(input("Loan Term (in years): ").strip())
        cibil_score = int(input("CIBIL Score (300-900): ").strip())
        residential_assets_value = float(input("Residential Assets Value: ").strip())
        commercial_assets_value = float(input("Commercial Assets Value: ").strip())
        luxury_assets_value = float(input("Luxury Assets Value: ").strip())
        bank_asset_value = float(input("Bank Asset Value: ").strip())
    except ValueError:
        print("Invalid input type! Please enter numeric values where required.")
        return

    # Validate categorical inputs
    if education_input not in le_education.classes_:
        print(f"Invalid Education value! Allowed: {list(le_education.classes_)}")
        return
    if self_employed_input not in le_self_employed.classes_:
        print(f"Invalid Self Employed value! Allowed: {list(le_self_employed.classes_)}")
        return

    # Prepare input data as DataFrame
    input_data = pd.DataFrame({
        'no_of_dependents': [no_of_dependents],
        'education': le_education.transform([education_input]),
        'self_employed': le_self_employed.transform([self_employed_input]),
        'income_annum': [income_annum],
        'loan_amount': [loan_amount],
        'loan_term': [loan_term],
        'cibil_score': [cibil_score],
        'residential_assets_value': [residential_assets_value],
        'commercial_assets_value': [commercial_assets_value],
        'luxury_assets_value': [luxury_assets_value],
        'bank_asset_value': [bank_asset_value]
    })
```

```
    # Scale input data using the trained scaler
    input_scaled = scaler.transform(input_data)

    # Make prediction
    prediction = model.predict(input_scaled)
    result = le_loan_status.inverse_transform(prediction)[0]

    print(f"\nPrediction: The loan is {result}")
```

```
212145010000# Run the interactive loan status prediction
predict_loan_status()
```

```
Enter the following details for loan prediction:
Number of dependents (0-5): 3
Education (Graduate/Not Graduate): Not Graduate
Self Employed (Yes/No): Yes
Annual Income: 20000
Loan Amount: 100000
Loan Term (in years): 10
CIBIL Score (300-900): 450
Residential Assets Value: 10000
Commercial Assets Value: 5000
Luxury Assets Value: 2000
Bank Asset Value: 1000

Prediction: The loan is Rejected
```

LOAN APPROVED TEST CASES

TEST-CASE-1:

Number of dependents (0-5): 1

Education (Graduate/Not Graduate): Graduate

Self Employed (Yes/No): No

Annual Income: 120000

Loan Amount: 50000

Loan Term (in years): 5

CIBIL Score (300-900): 750

Residential Assets Value: 100000

Commercial Assets Value: 50000

Luxury Assets Value: 20000

Bank Asset Value: 30000

TEST-CASE:2

Number of dependents (0-5): 0

Education (Graduate/Not Graduate): Graduate

Self Employed (Yes/No): No

Annual Income: 150000

Loan Amount: 40000

Loan Term (in years): 4

CIBIL Score (300-900): 780

Residential Assets Value: 120000

Commercial Assets Value: 60000

Luxury Assets Value: 25000

Bank Asset Value: 50000

LOAN REJECTED TEST CASES

TEST-CASE-1:

Number of dependents (0-5): 3

Education (Graduate/Not Graduate): Not Graduate

Self Employed (Yes/No): Yes

Annual Income: 20000

Loan Amount: 100000

Loan Term (in years): 10

CIBIL Score (300-900): 450

Residential Assets Value: 10000

Commercial Assets Value: 5000

Luxury Assets Value: 2000

Bank Asset Value: 1000

TEST-CASE:2

Number of dependents (0-5): 2

Education (Graduate/Not Graduate): Not Graduate

Self Employed (Yes/No): Yes

Annual Income: 25000

Loan Amount: 90000

Loan Term (in years): 8

CIBIL Score (300-900): 480

Residential Assets Value: 12000

Commercial Assets Value: 8000

Luxury Assets Value: 3000

Bank Asset Value: 1500