HINDUSTHAN INSTITUTE OF TECHNOLOGY

DEPARTMENT OF ARTIFICIAL INTELLIGENCE AND DATA SCIENCE

22AD405 – MACHINE LEARNING

LOAN APPROVAL PREDICTION USING DECISION TREE CLASSIFICATION

A MINI PROJECT REPORT

Submitted by

720823108043 - M.SAFRIN

NOV 2025

HINDUSTHAN INSTITUTE OF TECHNOLOGY

DEPARTMENT OF ARTIFICIAL INTELLIGENCE AND DATA SCINENCE

Certified that this is the bonafide for Mini Project report work done as a part of **22AD405 – MACHINE LEARNING LABORATORY** of this institution by (**M.Safrin-720823108043**), as prescribed by this Autonomous Institution for the **FIFTH** Semester during the Academic year **2025-2026.**

Place: Coimbatore
Date:

STAFF IN-CHARGE **HEAD OF THE DEPARTMENT**

Submitted for the viva voce held on _____.

` **EXAMINER**

INDEX

LOAN APPROVAL PREDICTION USING DECISION TREE CLASSIFICATION

Objective

To develop a machine learning-based system using a Decision Tree Classifier that predicts whether a loan application should be approved or rejected. The model utilizes applicant demographic, financial, and credit-related data to make accurate predictions, with additional visualization to support understanding of feature impact and model performance.

Abstact

The Loan Approval Prediction Project aims to streamline the decision-making process in financial institutions by using machine learning to assess loan applications. The dataset includes various features such as applicant income, loan amount, education, employment status, and credit history. After performing extensive data preprocessing—including handling missing values, encoding categorical variables, and normalizing numerical data—a Decision Tree Classifier is trained on the processed data.

Dataset Selection

The dataset used is the Loan Approval Prediction dataset in CSV format. It contains 614 records of loan applicants with 13 attributes. The attributes include personal details such as gender, marital status, dependents, education, and employment status, along with financial details like applicant income, co-applicant income, loan amount, loan term, credit history, and property area. The target attribute is Loan_Status**,** which indicates whether the loan application was approved or rejected.

Tools and Technologies

Programming Language: Python

Libraries: NumPy, Pandas, Scikit-learn, Matplotlib, Seaborn

IDE/Platform: Jupyter Notebook, Google Colab

Methodology

1. Data Collection

The dataset named LoanApprovalPrediction.csv was used.
It contains details of loan applicants like gender, income, dependents, loan amount, credit history, property area, and loan status (Approved/Rejected).

2. Data Preprocessing

- Converted loan amount, loan term, and credit history into numbers.
- Filled missing values:
  - Used median for loan amount and loan term.
  - Used mode for credit history and other categorical columns like gender and marital status.
- Changed `'3+'` dependents into 3 and converted to numbers.
- Used Label Encoding to convert text values (e.g., Male/Female, Yes/No) into numbers.

3. Exploratory Data Analysis (EDA)

- Checked for missing values using a heatmap.
- Plotted histograms for income and loan amount.
- Used a correlation heatmap to see relationships between features.
- Created bar charts to see how loan approval depends on gender, marital status, education, and property area.

4. Model Selection

- Chose a Decision Tree Classifier because it is easy to understand and explain.
- Limited the tree depth to 5 to avoid overfitting.

5. Training and Testing

- Split the dataset into 80% training data and 20% testing data.
- Trained the Decision Tree model on the training data.

6. Evaluation

- Checked the model performance using Accuracy, Precision, Recall, and F1-score.
- Plotted a Confusion Matrix to compare actual vs predicted results.
- Visualized the Decision Tree to understand the rules.
- Checked Feature Importance to see which features affected loan approval the most (e.g., Credit History, Income, Loan Amount).

7. Prediction

- Created a system to predict loan approval for a new applicant**.**
- Entered the applicant details (income, dependents, credit history, etc.).
- The model gave the result as either “Approved” or “Rejected”**,** and the output was shown in a simple visualization.

Program

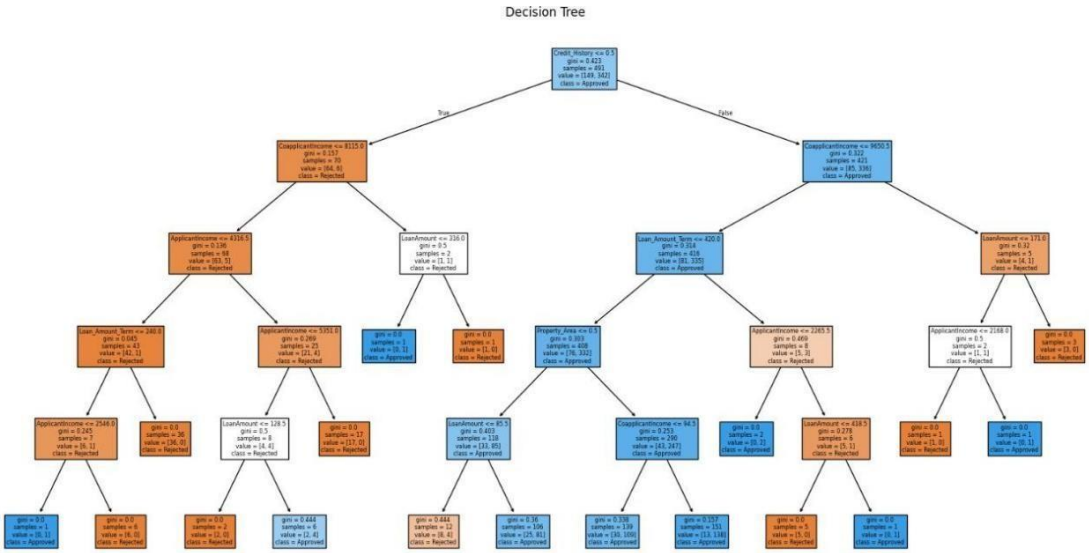
```
# ==============================
# LOAN APPROVAL PREDICTION PROJECT
# ==============================

# Import required libraries
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.preprocessing import LabelEncoder
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
from sklearn import tree
from google.colab import files

# Step 1: Upload the CSV file
print("Please upload the LoanApprovalPrediction.csv file.")
uploaded = files.upload()

# Step 2: Load the dataset
data = pd.read_csv('LoanApprovalPrediction.csv')

# Step 3: Data Preprocessing
# Convert to numeric where needed
data['LoanAmount'] = pd.to_numeric(data['LoanAmount'], errors='coerce')
data['Loan_Amount_Term'] = pd.to_numeric(data['Loan_Amount_Term'], errors='coerce')
data['Credit_History'] = pd.to_numeric(data['Credit_History'], errors='coerce')

# Handle missing numerical values
data['LoanAmount'].fillna(data['LoanAmount'].median(), inplace=True)
data['Loan_Amount_Term'].fillna(data['Loan_Amount_Term'].median(), inplace=True)
data['Credit_History'].fillna(data['Credit_History'].mode()[0], inplace=True)

# Handle missing categorical values
for col in ['Gender', 'Married', 'Self_Employed', 'Dependents']:
    data[col].fillna(data[col].mode()[0], inplace=True)

# Convert '3+' in Dependents and cast to int data['Dependents']
= data['Dependents'].replace('3+', '3')
data['Dependents'] = pd.to_numeric(data['Dependents'], errors='coerce')
```

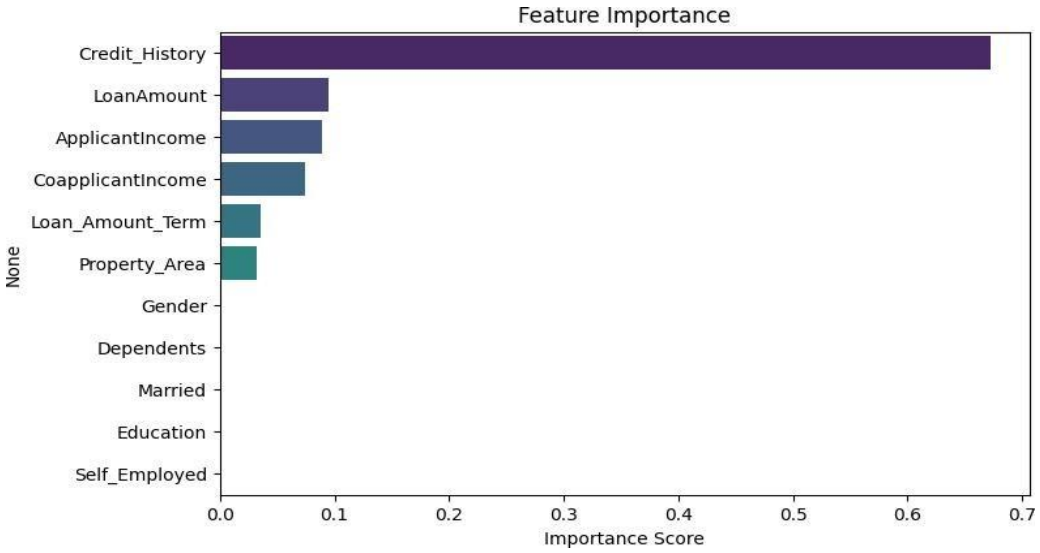
```
# Encode categorical variables
label_enc_cols = ['Gender', 'Married', 'Education', 'Self_Employed', 'Property_Area',
'Loan_Status']
le = LabelEncoder()
for col in label_enc_cols:
    data[col] = le.fit_transform(data[col])

# ==============================
# VISUALIZATION
# ==============================

# Heatmap for missing values
plt.figure(figsize=(10, 5))
sns.heatmap(data.isnull(), cbar=False, cmap='viridis')
plt.title('Missing Value Heatmap')
plt.show()

# Distribution plots
for col in ['ApplicantIncome', 'CoapplicantIncome', 'LoanAmount']:
    plt.figure(figsize=(6, 4))
    sns.histplot(data[col], bins=30, kde=True, color='skyblue')
    plt.title(f'Distribution of {col}')
    plt.xlabel(col)
    plt.ylabel("Frequency")
    plt.show()

# Correlation Heatmap - drop non-numeric columns like Loan_ID
numeric_data = data.drop(columns=['Loan_ID'])
plt.figure(figsize=(10, 6))
sns.heatmap(numeric_data.corr(), annot=True, cmap='coolwarm')
plt.title("Correlation Matrix")
plt.show()

# Bar plots for categorical features
cat_plot_cols = ['Gender', 'Married', 'Education', 'Self_Employed', 'Property_Area']
for col in cat_plot_cols:
    plt.figure(figsize=(6, 4))
    sns.countplot(x=col, hue='Loan_Status', data=data, palette='Set2')
    plt.title(f'Loan Status by {col}')
    plt.legend(title='Loan_Status', labels=['Rejected', 'Approved'])
    plt.show()

# ==============================
# MODEL TRAINING
# ==============================

# Define features and label
```

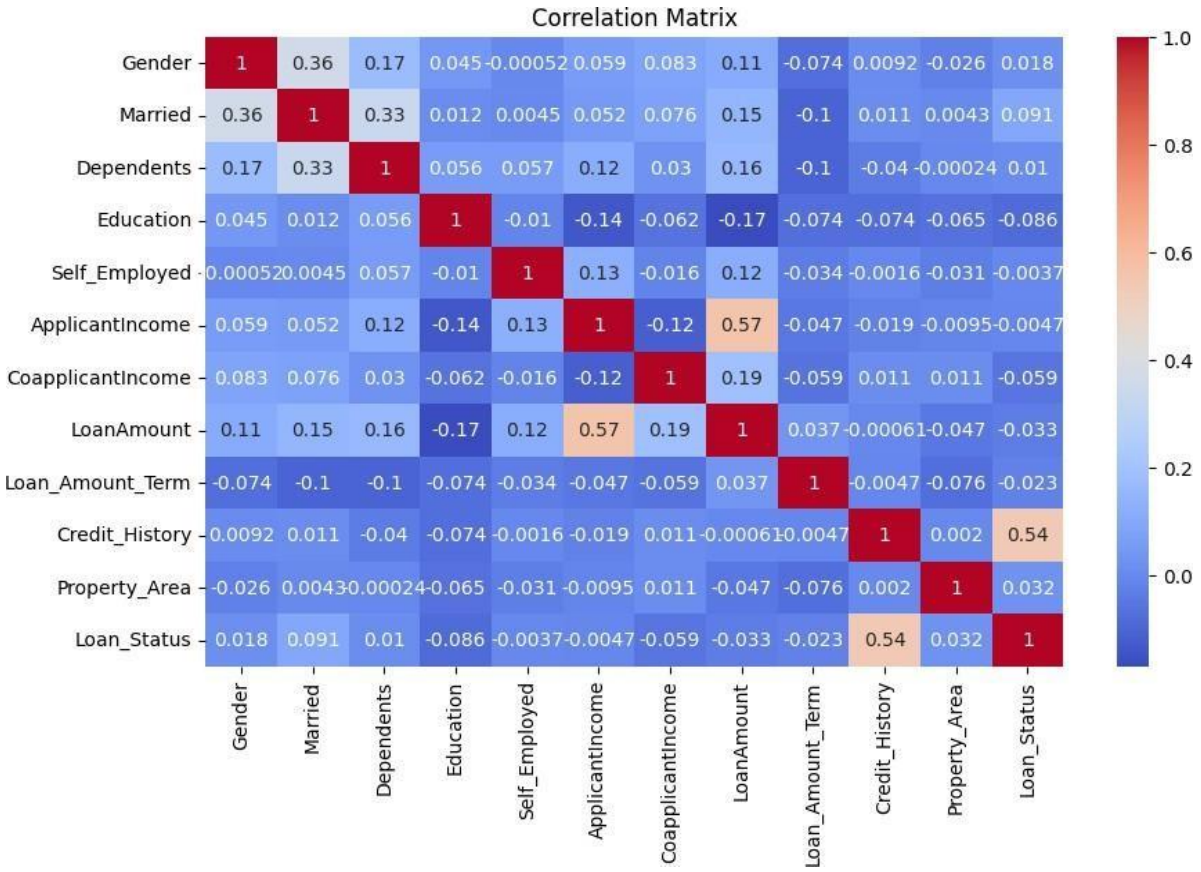
```
X = data.drop(['Loan_ID', 'Loan_Status'], axis=1)
y = data['Loan_Status']

# Handle any remaining NaNs (if any)
X.fillna(0, inplace=True)

# Split data
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Train Decision Tree model
model = DecisionTreeClassifier(max_depth=5, random_state=42)
model.fit(X_train, y_train)

# Predict
y_pred = model.predict(X_test)

# Evaluate
print(f"Accuracy: {accuracy_score(y_test, y_pred):.2f}")
print("Classification Report:")
print(classification_report(y_test, y_pred))

# Confusion Matrix
cm = confusion_matrix(y_test, y_pred)
plt.figure(figsize=(6, 4))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', xticklabels=['Rejected', 'Approved'],
yticklabels=['Rejected', 'Approved'])
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.title("Confusion Matrix")
plt.show()

# Visualize Decision Tree
plt.figure(figsize=(20, 10))
tree.plot_tree(model, feature_names=X.columns, class_names=['Rejected', 'Approved'],
filled=True)
plt.title("Decision Tree")
plt.show()

# Feature Importance
importance = pd.Series(model.feature_importances_, index=X.columns)
importance = importance.sort_values(ascending=False)
plt.figure(figsize=(8, 5))
sns.barplot(x=importance.values, y=importance.index, palette='viridis')
plt.title('Feature Importance')
plt.xlabel("Importance Score")
plt.show()

# ============================
```

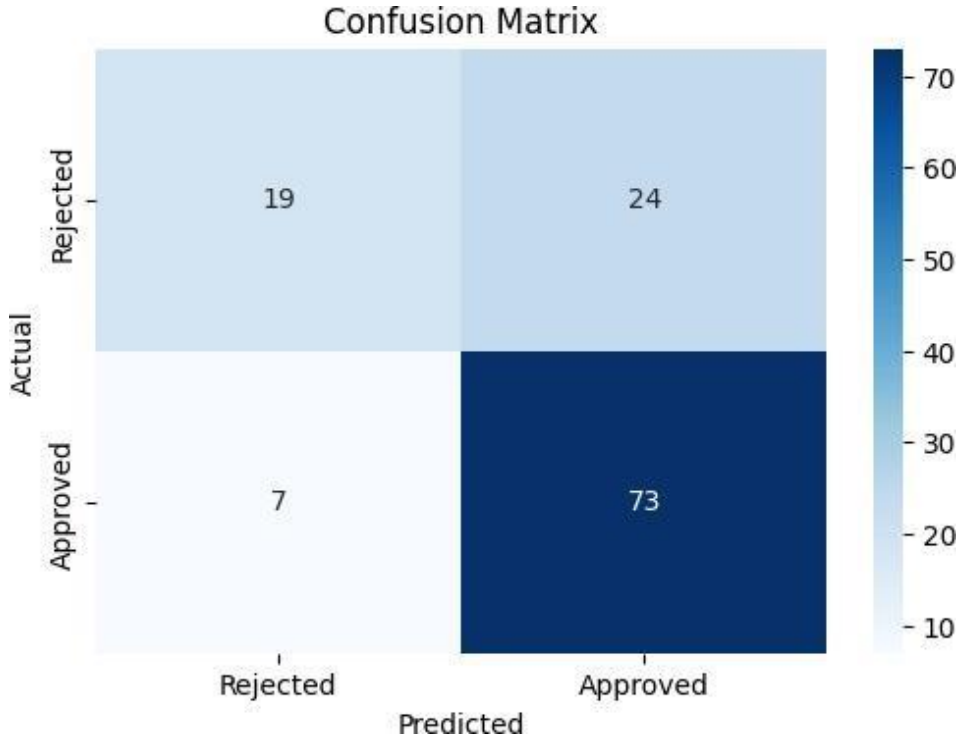
```
# PREDICT FOR NEW APPLICANT (VISUALIZED)
# ==============================

# New applicant input (encoded values must match training)
new_applicant = pd.DataFrame([{
    'Gender': 1,              # Male
    'Married': 1,             # Yes
    'Dependents': 1,          # 1 dependent
    'Education': 0,           # Graduate
    'Self_Employed': 0,       # No
    'ApplicantIncome': 4000,
    'CoapplicantIncome': 1500,
    'LoanAmount': 128,
    'Loan_Amount_Term': 360,
    'Credit_History': 1.0,
    'Property_Area': 2        # Urban
}])

# Predict loan approval
prediction = model.predict(new_applicant)[0]
result = 'Approved' if prediction == 1 else 'Rejected'
color = 'green' if prediction == 1 else 'red'

# Print result
print("\n    Loan Prediction for New Applicant:", result)

# Visualize the prediction
plt.figure(figsize=(5, 3))
plt.bar(['Loan Status'], [1], color=color)
plt.text(0, 0.5, result, fontsize=16, color='white', weight='bold', ha='center', va='center')
plt.ylim(0, 1.2)
plt.title("Loan Approval Prediction")
plt.xticks([])
plt.yticks([])
plt.box(False) plt.show()
```

Output & Results






Correlation Matrix
Gender 1 0.36 0.17 0.045 -0.00052 0.059 0.083 0.11 -0.074 0.0092 -0.026 0.018
Married 0.36 1 0.33 0.012 0.0045 0.052 0.076 0.15 -0.1 0.011 0.0043 0.091
Dependents 0.17 0.33 1 0.056 0.057 0.12 0.03 0.16 -0.1 -0.04 -0.00024 0.01
Education 0.045 0.012 0.056 1 -0.01 -0.14 -0.062 -0.17 -0.074 -0.074 -0.065 -0.086
Self_Employed -0.00052 0.0045 0.057 -0.01 1 0.13 -0.016 0.12 -0.034 -0.0016 -0.031 -0.0037
ApplicantIncome 0.059 0.052 0.12 -0.14 0.13 1 -0.12 0.57 -0.047 -0.019 -0.0095 -0.0047
CoapplicantIncome 0.083 0.076 0.03 -0.062 -0.016 -0.12 1 0.19 -0.059 0.011 0.011 -0.059
LoanAmount 0.11 0.15 0.16 -0.17 0.12 0.57 0.19 1 0.037 -0.00061 -0.047 -0.033
Loan_Amount_Term -0.074 -0.1 -0.1 -0.074 -0.034 -0.047 -0.059 0.037 1 -0.0047 -0.076 -0.023
Credit_History 0.0092 0.011 -0.04 -0.074 -0.0016 -0.019 0.011 -0.00061 -0.0047 1 0.002 0.54
Property_Area -0.026 0.0043 -0.00024 -0.065 -0.031 -0.0095 0.011 -0.047 -0.076 0.002 1 0.032
Loan_Status 0.018 0.091 0.01 -0.086 -0.0037 -0.0047 -0.059 -0.033 -0.023 0.54 0.032 1
Gender
Married
Dependents
Education
Self_Employed
ApplicantIncome
CoapplicantIncome
LoanAmount
Loan_Amount_Term
Credit_History
Property_Area
Loan_Status
1.0
0.8
0.6
0.4
0.2
0.0


Confusion Matrix
Actual
Rejected
Approved
19
24
7
73
Rejected
Approved
Predicted
70
60
50
40
30
20
10

Conclusion

In this mini-project, we developed a Loan Approval Prediction system using a Decision Tree Classifier. The dataset was cleaned, missing values were handled, and categorical data was encoded for training. Through EDA, we found that features like Credit History, Applicant Income, and Loan Amount play a major role in loan approval decisions.

Future Work

- Use other machine learning models like Random Forest, Logistic Regression, and XGBoost to improve accuracy.
- Apply cross-validation and hyperparameter tuning to make the model more robust.
- Add new features such as applicant's age, employment history, and savings for better predictions.
- Build a web-based interface so that users can enter loan details and instantly get predictions.
- Deploy the model using Flask/Django or Streamlit for real-world applications.

References

1. Kaggle – Loan Prediction Datasets: https://www.kaggle.com/datasets
2. Scikit-learn Documentation: https://scikit-learn.org
3. Pandas Documentation: https://pandas.pydata.org
4. Seaborn Documentation: https://seaborn.pydata.org