**GROUP FIVE PROJECT**

AUGUST, 2025

*DATA SCIENCE & ARTIFICIAL INTELLIGENCE*.

**LOAN APPROVAL PREDICTOR USING MACHINE LEARNING.**



By Bernice Eunice,

Eric Nzyimi,

Glorious Mwikali,

Nathan Muoki,

Joy Wangari.

**Q1 ; UNDERSTANDING THE PROBLEM AND DEFINING THE OBJECTIVES.**

The problem is to build a robust and accurate machine learning classifier that can assist financial institutions in making informed decisions regarding loan applications by predicting the Loan status based on applicant details.

**DEFINE OBJECTIVES**

1. To understand the core problem loan approval (a binary classification task) – Approved (Y) or Not Approved (N).
2. To identify the input features (applicant information) and the target variable (loan approval status).
3. To define success metrics for the model (e.g., accuracy, precision, recall, F1\_score as mentioned in the evaluation rubric).
4. To identify potential challenges (e.g., imbalance dataset, missing values, categorical features).

Loan Approval Prediction Flow.

Loan Decision (Approved/Not Approved)

ML Model Prediction (Logistic Regression/Decision Tress

Feature Processing e.g. Clean, Encode Data

Applicant Details

e.g Gender, income, Credit History...

**Q2; LOADING THE DATASET FROM KAGGLE.**

* Loan\_Train.csv is downloaded from <https://www.kaggle.com/datasets/granjithkumar/loan-approval-data-set/data> , and loaded onto Jupiter notebook.
* Loading of the python libraries

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

# ML

from sklearn.model\_selection import train\_test\_split

from sklearn.preprocessing import LabelEncoder, StandardScaler

from sklearn.linear\_model import LogisticRegression

from sklearn.tree import DecisionTreeClassifier

from sklearn.metrics import accuracy\_score, precision\_score, recall\_score, f1\_score, confusion\_matrix, classification\_report

# ML

from sklearn.model\_selection import train\_test\_split

from sklearn.preprocessing import LabelEncoder, StandardScaler

from sklearn.linear\_model import LogisticRegression

from sklearn.tree import DecisionTreeClassifier

from sklearn.metrics import accuracy\_score, precision\_score, recall\_score, f1\_score, confusion\_matrix, classification\_report

**Q3; PERFORM EXPLANATORY DATA ANALYSIS (EDA).**

This step involves understanding patterns, distributions, and the overall data.

* Checking data types and missing values

# Shape and info

print("Dataset Shape:", df.shape)

print(df.info())

# Check for null values

print("\nMissing Values:\n", df.isnull().sum())

# Summary stats

df.describe(include='all')

* Using sns.countplot() to check target balance (Loan\_Status).

# Check class balance

sns.countplot(x='Loan\_Status', data=df)

plt.title("Target Variable Distribution")

plt.show()

print(df['Loan\_Status'].value\_counts(normalize=True))

* Creating bar plot, histogram for features like Credit\_History, Education, etc.

# Fill missing categorical with mode, numerical with median

for column in df.columns:

if df[column].dtype == 'object':

df[column].fillna(df[column].mode()[0], inplace=True)

else:

df[column].fillna(df[column].median(), inplace=True)

**Q4; CLEAN AND PREPROCESS THE DATA.**

In this step, the dataset was prepared for modeling by addressing missing values, encoding categorical variables, and scaling numerical features:

**Handling Missing Values:**

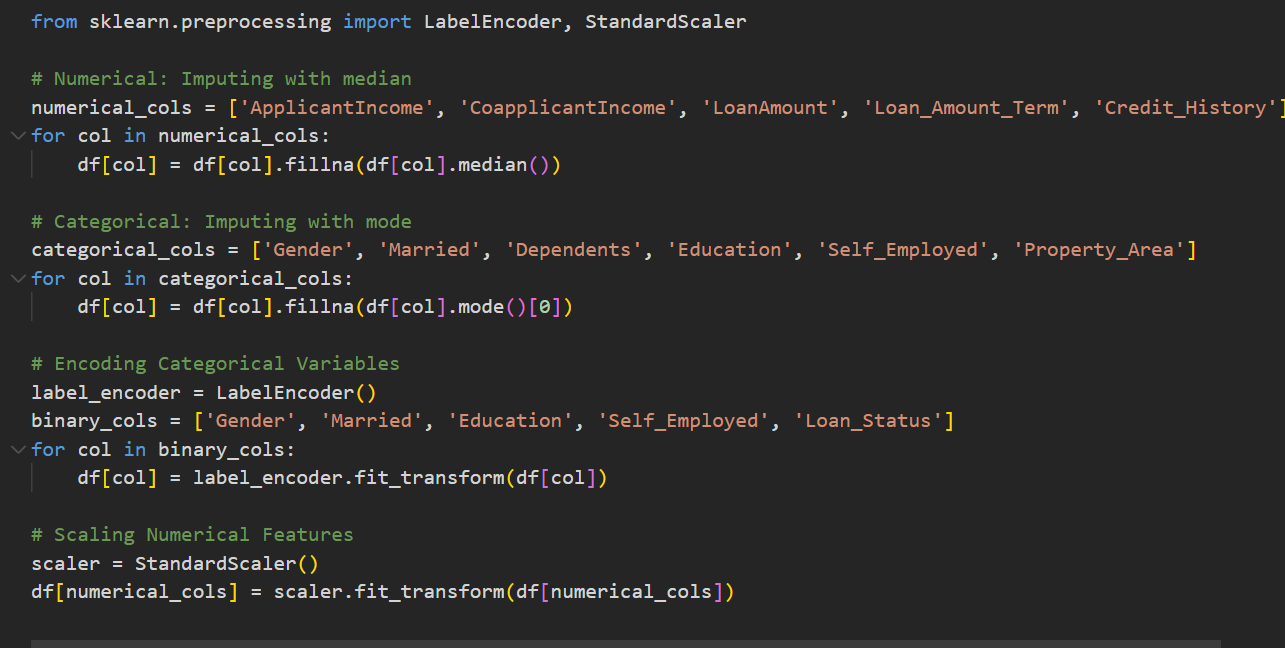
* **Numerical columns** (ApplicantIncome, CoapplicantIncome, LoanAmount, Loan\_Amount\_Term, Credit\_History) were imputed with the **median** value.
* **Categorical columns** (Gender, Married, Dependents, Education, Self\_Employed, Property\_Area) were imputed with the **mode** value.

**Encoding Categorical Variables:**

* Applied **Label Encoding** to binary categorical columns (Gender, Married, Education, Self\_Employed, Loan\_Status) to convert them into numeric format.

**Scaling Numerical Features:**

* Used **StandardScaler** to standardize the numerical columns for models sensitive to feature scale, such as Logistic Regression.



**Q5; PERFORM FEATURE SELECTION.**

The goal was to retain the most relevant features for prediction and remove unnecessary ones:

1. **Exploratory Checks:**
   * Plotted **histograms** of numerical features to understand distributions.
   * Generated a **correlation heatmap** to identify relationships between numerical variables and detect multicollinearity. No high correlations (> 0.8) were observed.
2. **Removing Irrelevant Columns:**
   * Dropped Loan\_ID as it does not contribute to the prediction task.
3. **Preparing Features and Target Variable:**
   * Defined X as all features except Loan\_Status, and y as the target variable (Loan\_Status).



**Q6; SPLIT THE DATASET INTO TRAINING AND TESTING SETS**.

*Dividing the data to train and test sets.*

* Training data helps the model **learn patterns.**
* Testing data checks if the model can use those patterns on new data
* **Features (X)** → the things we’ll use to make the decision (like income, education, etc.)
* **Target (y)** → the decision we’re trying to predict (loan approved = Y or N)

***from sklearn.model\_selection import train\_test\_split***

The dataset is split into 80% training and 20% testing and a random state of 42 is used to make the split repeatable and predictable.

**Q7; TRAIN TWO MODELS: LOGISTIC REGRESSION AND DECISION TREE.**

***Logistic Regression***

It’s a statistical model used to predict **binary outcomes** like if the loan was approved or not.

It works by drawing a boundary and predicting whether data falls into one of two categories.

***from sklearn.linear\_model import LogisticRegression.***

***-Train the model model= logisticregression()***

***Model.fit(x\_train,y\_train)***

***-Make predictions y\_pred= model.preict(x\_test)***

***-Evaluate the model to check the performance***.

*Decision Tree*

It uses yes /no questions to make predictions like will the loan be approved or not.

It uses different features like the credit history, income, education etc in the sample data to make a prediction.

***from sklearn.tree import DecisionTreeClassifier***

* + ***Train the decision tree model.***
  + ***Make predictions.***
  + ***Evaluate the model.***

**Q8 ; EVALUATE THE MODELS USING ACCURCY, PRECISION, RECALL, F1\_SCORE, CONFUSION MATRIX.**

Once we’ve trained the models (Logistic Regression and Decision Tree), we need to measure how well the models perform using the key evaluation metrics.

Accuracy– To determine the Proportion of total correct predictions

Precision– To show How many predicted "yes" loans were actually correct

F1-Score – A balanced score between Precision and Recall

Recall – Determines how many actual "yes" loans were predicted correctly

​

Confusion **Matrix** – A table showing:

* TP (True Positive)
* TN (True Negative)
* FP (False Positive)

**Q9; TEST THE MODELS ON THE RESERVED TEST SET.**

After training and evaluating, we check how well the model performs on the unseen data ie (30% of the original data). This shows how well the model generalizes to new data.

We use predict(X\_test) to make predictions.

We re-calculate Accuracy, Precision, Recall, and F1-score.

This confirms that the model learns patterns and prevents overfitting.

Q 10 ; CONCLUSION AND RECOMMENDATIONS.

***CONCLUSION.***

This project aimed to predict loan approvals using machine learning. After cleaning and preparing the data, we built and evaluated two models: **Logistic Regression** and **Decision Tree Classifier**.

Both models were able to capture the key patterns in the data. However, **Logistic Regression performed slightly better**, especially in terms of precision and F1-score.

The models showed that features such as Credit History, Applicant Income, and Loan Amount were critical in determining loan eligibility.

The confusion matrix showed that the models had a good balance between correctly approving and rejecting loans, minimizing errors.

***RECCOMMENDATIONS.***

1. Logistic Regression is suitable in production environments due to its better overall performance and lower risk of overfitting.
2. Improve Data Quality: prioritize data quality, by data cleaning the original dataset in order to improve accuracy.
3. Monitor Performance Over Time: Regularly retrain models with new data to maintain relevance and accuracy
4. Add More Features: Including more financial variables like employment duration or savings may improve prediction performance.
5. Consider Model Ensembles: For future improvements, try ensemble methods like Random Forest or Gradient Boosting for potentially better results.