****

**Assessment Report**

on

**“Predict Loan Default”**

submitted as partial fulfillment for the award of

**BACHELOR OF TECHNOLOGY**

**DEGREE**

SESSION 2024-25

in

**CSE(AIML)**

By

GROUP-08

Devansh Mittal - 202401100400078

Deepak Joshi - 202401100400075

Devkesh Yadav - 202401100400081

Kartikey Kumar - 202401100400106

Prashant Tiwari - 202401100400142

**Under the supervision of**

“ABHISHEK SHUKLA SIR”

**KIET Group of Institutions, Ghaziabad**

**May, 2025**

**1. Introduction**

As digital lending continues to grow, automating credit risk evaluation has become essential. This project leverages machine learning to develop a model for predicting whether a borrower will default on a loan. Using data on credit history, income, and other applicant details, we train and evaluate a model that can help financial institutions make reliable lending decisions

**2. Problem Statement**

To predict whether a borrower will default on a loan using historical and financial features. The goal is to build a binary classifier that aids financial institutions in risk assessment.

**3. Objectives**

* Preprocess and clean the dataset for optimal model performance.
* Engineer new features that improve model accuracy.
* Train a Random Forest model with hyperparameter tuning using GridSearchCV.
* Evaluate model performance using classification metrics.
* Visualize results using confusion matrix and feature importance plots.

**4. Methodology**

**Data Collection**

* Used the dataset: train\_dataset.csv.

**Data Preprocessing**

* Missing values handled using mode (for categorical) and median (for numerical).
* Feature engineering: Added TotalIncome, LoanAmountLog, and TotalIncomeLog.
* Encoded categorical variables using LabelEncoder.

**Model Training**

* Applied train-test split with stratification (80/20).
* Used RandomForestClassifier with GridSearchCV for tuning:
  + n\_estimators: [100, 200]
  + max\_depth: [5, 10, None]
  + min\_samples\_split: [2, 5]
  + class\_weight: ['balanced']

Model Evaluation

* Accuracy, Precision, Recall, F1-Score
* Confusion matrix and feature importance visualized with Seaborn and Matplotlib

### ****5. Data Preprocessing****

* Categorical columns: 'Gender', 'Married', 'Dependents', 'Self\_Employed', 'Credit\_History', 'Loan\_Amount\_Term' filled using mode.
* Numerical column: 'LoanAmount' filled using median.
* New features:
  + TotalIncome = ApplicantIncome + CoapplicantIncome
  + LoanAmountLog = log1p(LoanAmount)
  + TotalIncomeLog = log1p(TotalIncome)
* Categorical variables encoded using LabelEncoder.
* Final features selected:
  + 'Credit\_History', 'LoanAmountLog', 'TotalIncomeLog', 'Loan\_Amount\_Term', 'Gender', 'Married', 'Education', 'Self\_Employed', 'Property\_Area', 'Dependents'

**6. Model Implementation**

We used a Random Forest Classifier optimized with GridSearchCV. Cross-validation ensured robust performance across different folds, and hyperparameters were tuned for better generalization

### ****7. Evaluation Metrics****

The following metrics were computed on the test set:

* **Accuracy**: ≈ {:.2f}%.format(accuracy\_score(y\_test, y\_pred) \* 100)
* **Precision, Recall, F1-Score**: From classification report
* **Confusion Matrix**: Visualized to show true positives, false positives, etc.

**8. Results and Analysis**

-The tuned Random Forest model showed high accuracy and balance across precision and recall.

- Confusion matrix revealed the class distribution and potential misclassifications.

-Feature importance plot indicated the most predictive features:

-Credit\_History, TotalIncomeLog, LoanAmountLog, etc

**9. Conclusion**

A well-tuned Random Forest model was developed to predict loan default risk. It demonstrated strong classification performance and interpretability. The pipeline from preprocessing to evaluation ensures reproducibility and reliability. Further improvements could involve handling class imbalance and ensembling multiple models

**10. References**

* scikit-learn documentation
* pandas documentation
* Seaborn visualization library
* Matplotlib library
* Kaggle Loan Prediction Dataset

**11 CODE:**

**# 📊 Loan Default Prediction with Random Forest**

**# Step 1: Import 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, StratifiedKFold, GridSearchCV**

**from sklearn.preprocessing import LabelEncoder, StandardScaler**

**from sklearn.impute import SimpleImputer**

**from sklearn.ensemble import RandomForestClassifier**

**from sklearn.metrics import classification\_report, confusion\_matrix, accuracy\_score**

**sns.set(style="whitegrid")**

**# Step 2: Load Data**

**df = pd.read\_csv("/content/train\_dataset.csv")**

**# Step 3: Data Cleaning + Feature Engineering**

**# Fill categorical missing values**

**cat\_cols = ['Gender', 'Married', 'Dependents', 'Self\_Employed', 'Credit\_History', 'Loan\_Amount\_Term']**

**for col in cat\_cols:**

**df[col] = df[col].fillna(df[col].mode()[0])**

**# Fill numerical missing values**

**df['LoanAmount'] = df['LoanAmount'].fillna(df['LoanAmount'].median())**

**# New features**

**df['TotalIncome'] = df['ApplicantIncome'] + df['CoapplicantIncome']**

**df['LoanAmountLog'] = np.log1p(df['LoanAmount'])**

**df['TotalIncomeLog'] = np.log1p(df['TotalIncome'])**

**# Encode categories**

**encode\_cols = ['Gender', 'Married', 'Dependents', 'Education', 'Self\_Employed', 'Property\_Area', 'Loan\_Status']**

**for col in encode\_cols:**

**df[col] = LabelEncoder().fit\_transform(df[col])**

**# Step 4: Feature Selection**

**features = ['Credit\_History', 'LoanAmountLog', 'TotalIncomeLog', 'Loan\_Amount\_Term',**

**'Gender', 'Married', 'Education', 'Self\_Employed', 'Property\_Area', 'Dependents']**

**X = df[features]**

**y = df['Loan\_Status']**

**# Step 5: Train-Test Split with Stratification**

**X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, stratify=y, test\_size=0.2, random\_state=42)**

**# Step 6: Model Training with GridSearchCV**

**param\_grid = {**

**'n\_estimators': [100, 200],**

**'max\_depth': [5, 10, None],**

**'min\_samples\_split': [2, 5],**

**'class\_weight': ['balanced']**

**}**

**rf = RandomForestClassifier(random\_state=42)**

**grid\_search = GridSearchCV(rf, param\_grid, cv=5, n\_jobs=-1, verbose=1)**

**grid\_search.fit(X\_train, y\_train)**

**best\_rf = grid\_search.best\_estimator\_**

**# Step 7: Evaluation**

**y\_pred = best\_rf.predict(X\_test)**

**conf\_matrix = confusion\_matrix(y\_test, y\_pred)**

**report = classification\_report(y\_test, y\_pred, output\_dict=True)**

**# Step 8: Report**

**print("🔍 Best Parameters from GridSearchCV:")**

**print(grid\_search.best\_params\_)**

**print("\\n📈 Classification Report:")**

**print(classification\_report(y\_test, y\_pred))**

**print("✅ Accuracy: {:.2f}%".format(accuracy\_score(y\_test, y\_pred) \* 100))**

**# Step 9: Feature Importances**

**importances\_df = pd.DataFrame({**

**'Feature': features,**

**'Importance': best\_rf.feature\_importances\_**

**}).sort\_values(by='Importance', ascending=True)**

**# Step 10: Visualizations**

**fig, axes = plt.subplots(1, 2, figsize=(16, 6))**

**# Confusion Matrix**

**sns.heatmap(conf\_matrix, annot=True, fmt='d', cmap='YlGnBu', ax=axes[0], cbar=False)**

**axes[0].set\_title("Confusion Matrix", fontsize=14, fontweight='bold')**

**axes[0].set\_xlabel("Predicted")**

**axes[0].set\_ylabel("Actual")**

**axes[0].set\_xticklabels(['No Default', 'Default'])**

**axes[0].set\_yticklabels(['No Default', 'Default'])**

**# Feature Importance**

**# Use a color palette for variety — you can try 'husl', 'Set2', 'viridis', etc.**

**importances\_df["ColorGroup"] = importances\_df["Feature"]  # Create a dummy hue**

**sns.barplot(**

**x='Importance',**

**y='Feature',**

**hue='ColorGroup',         # Use 'Feature' as hue**

**dodge=False,              # Prevent split bars**

**data=importances\_df,**

**ax=axes[1],**

**palette='Set2',**

**legend=False              # Avoid legend since it duplicates y-axis**

**)**

**axes[1].set\_title("Feature Importances", fontsize=14, fontweight='bold')**

**axes[1].set\_xlabel("Importance", fontsize=12)**

**axes[1].set\_ylabel("Feature", fontsize=12)**

**plt.tight\_layout()**

**plt.show()**

**12 OUTPUT SNAPSHOT:**

