Loan Eligibility Prediction – Full Project Report

1. Introduction

Access to credit is one of the core pillars of the financial ecosystem. Financial institutions must be able to quickly assess the probability that a borrower may default on repayment. This project focuses on building an **end-to-end Machine Learning–powered Loan Eligibility and Default Prediction System** using:

- **XGBoost classifier** (final model)
- **Python & Flask backend API**
- **MySQL database integration**
- **HTML/CSS/JS frontend**
- **Google Colab for ML development**

This application predicts whether a loan applicant is **Approved** or **Rejected** and logs all predictions into a database.

---

2. Project Objectives

1. Analyze and preprocess the customer loan dataset (`Loan_default.csv`).
2. Train multiple ML models to predict loan default.
3. Select the best-performing model for deployment.
4. Build a complete backend using Flask, exposing a `/predict` API endpoint.
5. Create a minimal but user-friendly frontend for real-time prediction.
6. Store every prediction request and result in a MySQL database.
7. Deploy locally and prepare production-ready project structure.

---

3. Dataset Overview

- **Source:** `Loan_default.csv`
- **Records:** 255,347
- **Columns:** 18
- **Target Variable:** `Default` (0 = No, 1 = Yes)
- **Missing Values:** None

- **Feature Types:**
    - Numerical: Age, Income, LoanAmount, InterestRate, CreditScore, etc.
    - Categorical: Education, EmploymentType, MaritalStatus, LoanPurpose, etc.

3.1 Sample Attributes

Category	Attributes
Financial	Income, LoanAmount, InterestRate, DTIRatio
Personal	Age, CreditScore, Education
Behavioral	EmploymentType, HasDependents, HasMortgage
Loan-related	LoanTerm, LoanPurpose

---

4. Data Preprocessing

Steps performed:

4.1 Dropping Irrelevant Columns

- `LoanID` removed (does not affect prediction).

4.2 Encoding Categorical Variables

One-hot encoding applied to:

- Education
- EmploymentType
- MaritalStatus
- LoanPurpose
- HasMortgage
- HasDependents
- HasCoSigner

Example:

Education → High School Bachelor’s Master's PhD

4.3 Scaling Numerical Features

`StandardScaler` used for:

- Income
- LoanAmount
- DTIRatio
- CreditScore
- MonthsEmployed

- InterestRate

4.4 Train-test Split

- **80%** Training
- **20%** Testing
- `random_state = 42`

---

5. Exploratory Data Analysis (EDA)

5.1 Key Observations

Interest Rate

- Higher interest rate → Higher probability of default

Credit Score

- Low credit score strongly linked to default
- One of the highest-value predictors

DTIRatio (Debt-to-Income Ratio)

- Borrowers with DTI above **0.6** are high-risk.

Employment Type

- Unemployed applicants show significantly higher default likelihood.

Correlation Matrix Analysis

- No harmful multicollinearity
- Top predictors:
    - InterestRate
    - CreditScore
    - EmploymentType_Unemployed
    - DTIRatio

These findings helped define the **final feature set used in the deployed model**.

---

6. Machine Learning Models

Three models were trained:

6.1 Logistic Regression

- Accuracy: 0.8855
- Precision: 0.6007
- Recall: 0.0278
- F1-score: 0.0531

6.2 Random Forest

- Accuracy: 0.8867
- Precision: 0.6306
- Recall: 0.0475
- F1-score: 0.0883

6.3 XGBoost Classifier (Final Model)

- Accuracy: 0.8862
- Precision: 0.5495
- Recall: 0.0827
- **F1-score: 0.1438 (highest)**

Why choose XGBoost?

- Best F1-score among all models
- Best recall value (important for default detection)
- Works well with imbalanced data
- Handles non-linear patterns efficiently

---

7. Model Deployment

7.1 Saved Models

All models saved using joblib:

```
logistic_regression_model.joblib
random_forest_model.joblib
xgboost_model.joblib  ← (Used in production)
```

---

8. Backend Development (Flask API)

8.1 Key Features

- Accepts loan applicant details via JSON
- Preprocesses data exactly like training pipeline
- Runs inference with XGBoost model
- Returns probability + Approved/Rejected
- Stores every prediction into MySQL database

API Endpoint

```
POST /predict
```

API Response Example:

```
{
  "prediction": "Rejected",
  "probability": 0.72
}
```

---

9. Database Integration (MySQL)

Table: `predictions`

Stores:

- Applicant inputs
- Model prediction
- Probability score
- Timestamp

Why this is important?

✔️ Useful for dashboards
✔️ Helps banks track user behavior
✔️ Enables model monitoring (ML Ops)

---

10. Frontend Application

Features:

- Clean UI for input fields
- Dropdowns for categorical fields

- Input validation + hints
- Displays Approved/Rejected with probability
- Interacts with Flask backend using Fetch API

Example Output

```
Result: Approved
Probability of Default: 17.15%
```

---

11. End-to-End Flow

```
User Input →
Frontend →
Flask API →
Preprocessing →
XGBoost Model →
Prediction →
MySQL DB Storage →
Frontend Display
```

---

12. Project Achievements

✔️ Full ML pipeline (EDA → Training → Deployment)

✔️ Working end-to-end web application

✔️ Real-time ML predictions

✔️ Database-backed, production-ready structure

✔️ Clean modular code

✔️ GitHub-ready project layout

---

13. Limitations

- Model recall is still low (dataset is highly imbalanced)
- UI is minimal; can be expanded
- No authentication system
- No online deployment (only local)

---

14. Future Enhancements

- **Model Improvements**
  - Use SMOTE / class balancing
  - Hyperparameter tuning
  - SHAP Explainability reports
- **Frontend Enhancements**
  - Convert to React.js
  - Add data visualization dashboard
- **Backend Enhancements**
  - JWT authentication
  - Admin dashboard to see prediction history
- **Deployment**
  - Deploy backend on Render / Railway
  - Deploy frontend on Netlify / Vercel

---

15. Conclusion

This project demonstrates a full lifecycle of a Machine Learning system:

- Data Analysis
- Feature Engineering
- Model Training & Evaluation
- Backend API Development
- Frontend UI Integration
- SQL Database Logging
- Git + SSH Deployment

It proves capability in **ML engineering, data analysis, backend development, and full-stack integration**, making it highly valuable for roles in:

- Data Science
- Machine Learning
- BFSI Analytics
- AI Engineering

- Full-stack ML Developer

---

16. Appendix

Technologies Used

Category	Tools
Programming	Python, JavaScript
ML Libraries	XGBoost, Scikit-Learn, Pandas
Backend	Flask
Database	MySQL
Frontend	HTML, CSS, JS
DevOps	Git, GitHub, SSH
Model Serving	joblib