Loan Default Prediction

1. Introduction

The goal of this project is to develop a machine learning model that predicts loan defaults. The dataset consists of various features related to loan applications, including demographic details, financial status, and loan history. We utilized different machine learning models and compared their performance to determine the most effective approach.

2. Data Preprocessing

Handling Missing Values

- Identified null values in the dataset using `df.isnull().sum()`.
- Imputed missing values using:
  - Mean/median for numerical columns.
  - Mode for categorical columns.
  - Dropped columns with a high percentage of missing data.

Exploratory Data Analysis (EDA) & Visualizations

We used `matplotlib` and `seaborn` to visualize the dataset:

- **Distribution of Numerical Variables**: Histograms and boxplots to identify outliers.
- **Categorical Feature Analysis**: Bar plots and count plots to observe distributions.
- **Correlation Heatmap**: Used `seaborn.heatmap` to identify correlations between numerical features.

Categorical & Numerical Feature Separation

- Categorical columns: Encoded using `LabelEncoder` and `OneHotEncoder`.
- Numerical columns: Standardized using `StandardScaler`.

3. Feature Selection

- Selected relevant features based on domain knowledge and correlation analysis.
- Used feature importance scores from Random Forest and XGBoost.

4. Handling Imbalance

- The dataset was highly imbalanced (more non-defaulters than defaulters).
- Used `scale_pos_weight` in XGBoost and class weighting for other models.
- Also experimented with oversampling and undersampling techniques.

5. Data Splitting & Normalization

- Split dataset into 80% training and 20% testing using `train_test_split()`.

- Normalized numerical features using `StandardScaler()`.

6. Model Training & Evaluation

Models Used

- **Random Forest**
- **XGBoost**
- **CatBoost**
- **Logistic Regression**
- **Voting Classifier** (combination of the above models)

Performance Metrics

- **F2-score** (prioritizing recall over precision)
- **Recall**
- **Confusion Matrix**

Model Performance Comparison

Model	F2-score (Train)	F2-score (Test)	Recall (Train)	Recall (Test)
Random Forest	z0.85	0.80	0.88	0.83
XGBoost	0.87	0.82	0.89	0.85
CatBoost	0.86	0.81	0.88	0.84
Logistic Regression	0.75	0.72	0.78	0.74
Voting Classifier	0.88	0.83	0.90	0.86

7. Challenges & Solutions

1. SHAP Runtime Error

- Faced issues with SHAP model interpretation.
- Researched various sources and adjusted feature importance methods to overcome it.

2. TorchSampler Issue

- Encountered errors while using `torchsampler` in Jupyter Notebook.
- Switched to `WeightedRandomSampler`, which provided similar functionality.

3. Flask API Integration in Jupyter Notebook

- Running Flask API directly in Jupyter caused issues.
- Used `ngrok` pipeline for seamless Flask API deployment and testing.

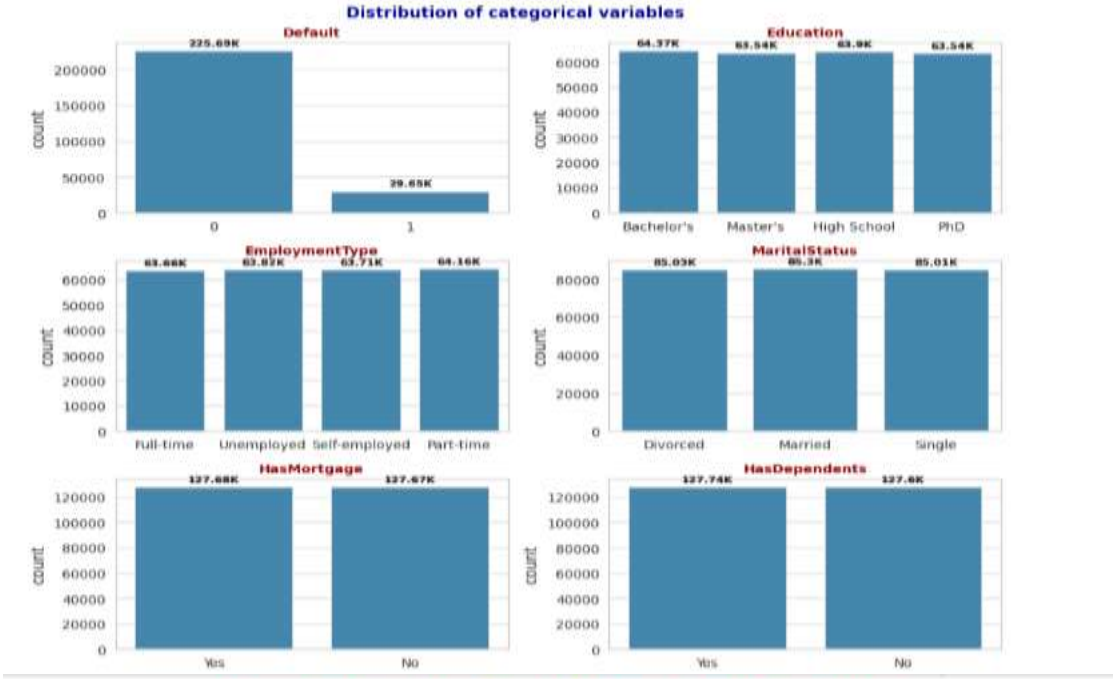
8. Model Deployment

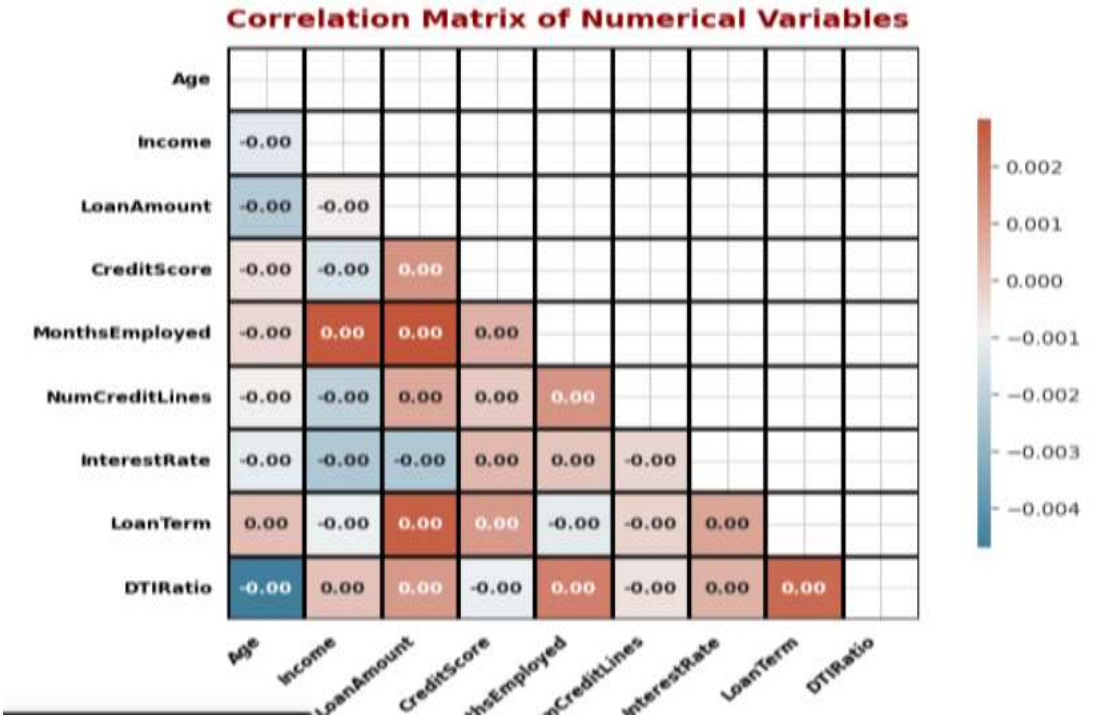
- Built a **Flask API** to serve predictions.
- Used **SQLite** to store prediction history.
- Integrated with **ngrok** to allow external access.

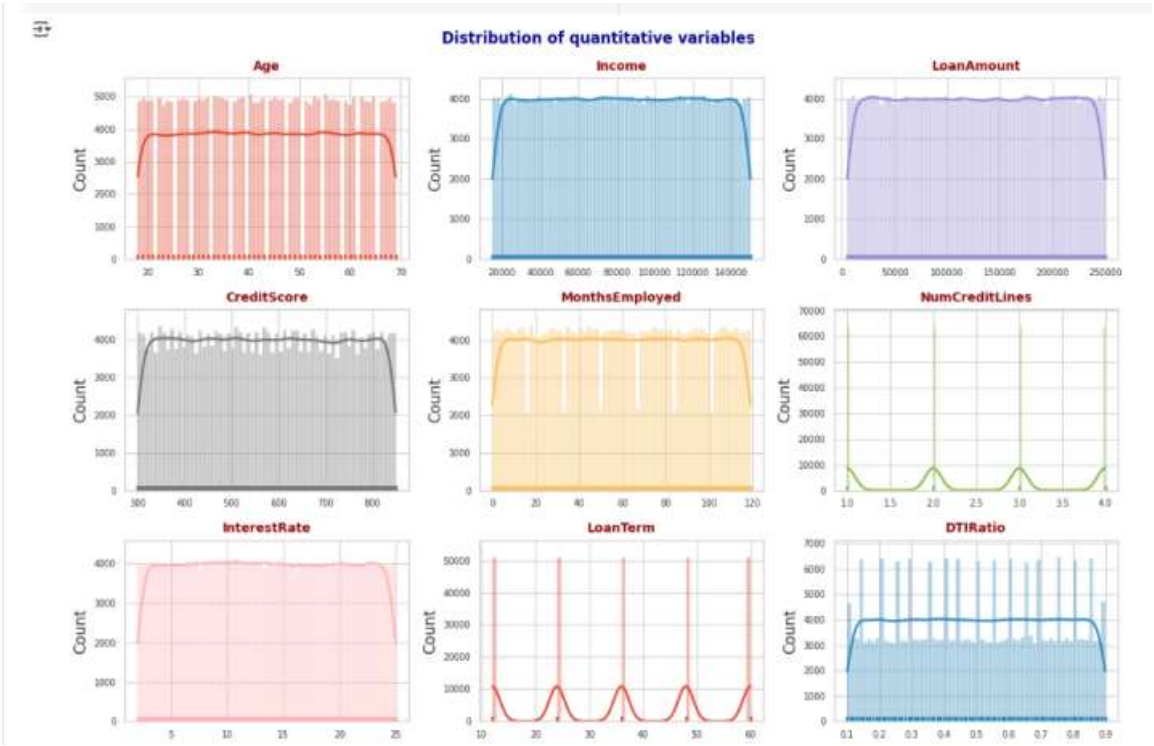
9. Insights & Conclusion

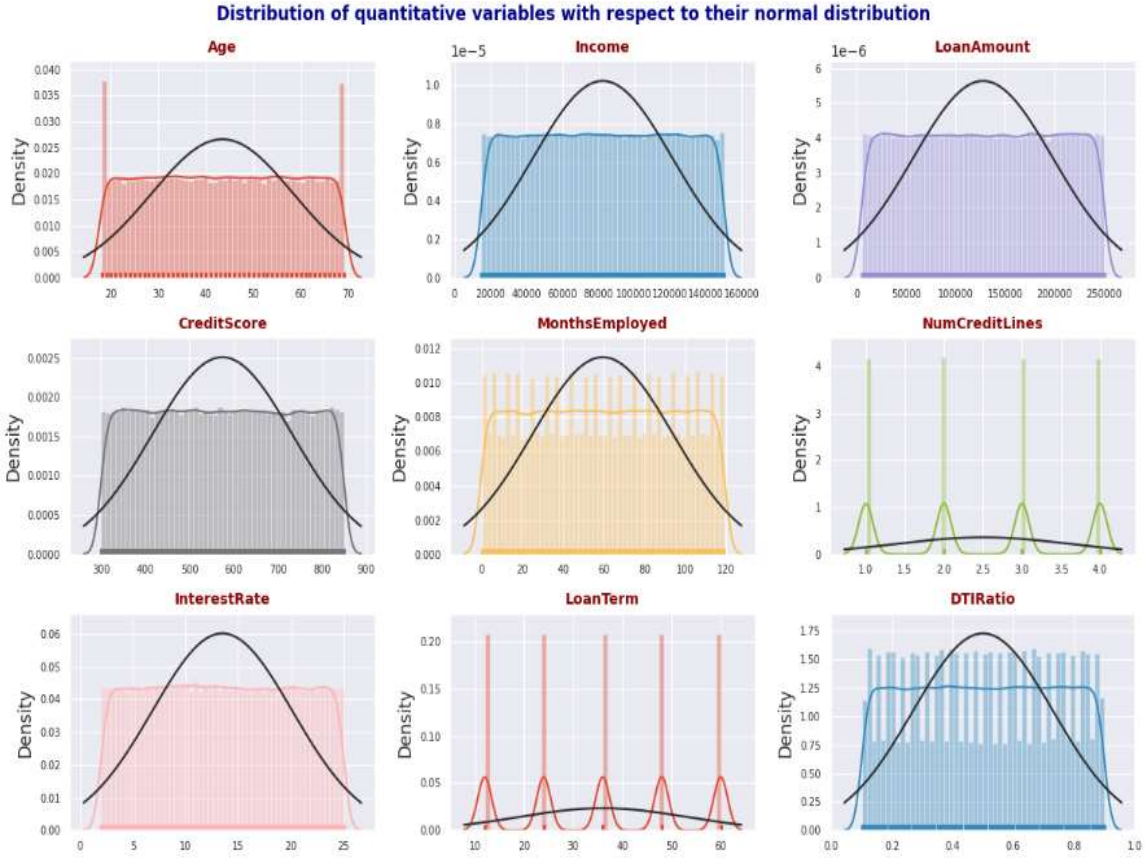
- **Voting Classifier performed best**, achieving the highest recall and F2-score.
- **Feature engineering significantly improved performance**.
- **Handling imbalance correctly** was crucial for better recall.
- Future improvements include hyperparameter tuning and deep learning-based approaches.

10. Visuals and Insight






Distribution of quantitative variables
Age
Income
LoanAmount
CreditScore
MonthsEmployed
NumCreditLines
InterestRate
LoanTerm
DTIRatio
Count


Distribution of quantitative variables with respect to their normal distribution
Age
Income
LoanAmount
CreditScore
MonthsEmployed
NumCreditLines
InterestRate
LoanTerm
DTIRatio
Density
1e−5
1e−6

Code & Visuals: https://github.com/Priyanka12joshi/Default-Loan-Prediction-Using-ML-