Eligibility Prediction for Loan - Report

1- Introduction

Automating the loan approval process is crucial for modern lenders to improve speed and accuracy. Manual loan underwriting is time-consuming and prone to human error, whereas automation enables faster processing, reduced default risk, and lower operational costs. In fact, industry experts recommend that every step of loan approval be automated to streamline decisions and enhance efficiency. Machine learning offers a way to automatically learn patterns from historical loan data and predict whether new loan applications should be approved or not. In this project, a dataset of past loan applications is used to train classification models that can predict loan approval (Yes/No) based on applicant information. The goal is to build an accurate model to automate loan approval decisions, saving time for banks and ensuring consistent criteria in evaluating applications. (kaggle) (Github) .

2- Dataset describtion

The dataset Loan Data contains records of previous loan applications with 614 samples. Each sample includes several features about the applicant and the loan, as well as the final loan status (approved or not).
The dataset includes a variety of features related to the loan applicant and the loan itself. Applicant demographics consist of Gender, Marital Status, Number of Dependents, Education level, and whether the applicant is Self-Employed. Financial information includes ApplicantIncome, CoapplicantIncome, and Credit_History, which indicates the applicant's past credit reliability. Loan-specific details cover the LoanAmount, Loan_Amount_Term (in months), and Property_Area, categorized as Urban, Semiurban, or Rural. The target variable is Loan_Status, representing whether the applicant's loan was approved (Y) or not approved (N).

3- Data preprocessing

Before modeling, a series of essential preprocessing steps were applied to prepare the data for machine learning. To handle missing values, the dropna() function was used to remove any rows containing null entries. Although this led to a slight reduction in dataset size, it ensured that all remaining data points were complete and free of inconsistencies that could interfere with training or introduce bias. This approach was considered effective given the relatively small number of missing values. Next, categorical variables such as Gender, Married, Education, Self_Employed, and Property_Area were converted into numerical form using label encoding, where each category was mapped to a unique integer (e.g., Yes = 1, No = 0). One-hot encoding was not applied; instead, all categorical features were encoded directly to preserve simplicity. Lastly, because the dataset included features with varying scales such as income figures in the thousands and binary features standardization was performed using StandardScaler. This transformation ensured that all numeric features had a mean of 0 and a standard deviation of 1, which is particularly important for models that rely on distance calculations or gradient-based optimization.

After preprocessing, the data was split into a training set and a test set in a 70/30 ratio. The training set (70% of the data) was used to train the models, and the held-out test set (30% of the data) was used to evaluate how well the trained models generalize to unseen cases. The split was stratified by the target Loan_Status to preserve the original proportion of approved vs. not approved loans in both subsets.

One challenge with the dataset is that the classes are imbalanced: about 69% of the loans in the data were approved (Yes) while 31% were not approved (No). Class imbalance can bias a classifier to favor the majority class. To address this, SMOTE (Synthetic Minority Oversampling Technique) was applied only on the training set to balance the classes. SMOTE is an oversampling technique that generates synthetic samples for the minority class

by interpolating between real minority instances. In this case, the minority class "Loan_Status = No" was oversampled in the training data to approximately match the number of "Yes" cases. This synthetic oversampling helps the models learn the characteristics of the "No" (loan not approved) class better, without simply duplicating existing examples (which could cause overfitting). It is important to note that the test set was not oversampled or otherwise touched, so that performance evaluation remains valid and unbiased.

4- Training phase

Several machine learning classification algorithms were trained on the balanced training data. The models used in this project include:

- **K-Nearest Neighbors (KNN):** a distance-based classifier that predicts a label based on the majority vote of the k nearest training examples in feature space (with k=5 by default).
- **Support Vector Machine (SVM):** a classifier that finds the optimal hyperplane to separate classes in a high-dimensional space.
- **Decision Tree:** a binary tree-based model that splits the data on feature values to classify outcomes; the tree was grown using the default settings.
- **Random Forest:** an ensemble of many decision trees where each tree votes on the outcome; this tends to improve generalization by averaging multiple tree predictions.
- **Naive Bayes:** a probabilistic classifier based on Bayes' theorem with an assumption of feature independence (Gaussian Naive Bayes was used for continuous features).
- **Artificial Neural Network (ANN):** a multi-layer perceptron (feed-forward neural network) with one hidden layer (implemented via MLPClassifier).
- **Logistic Regression:** a linear model that estimates the probability of the positive class (loan approved) with a logistic function; it was trained with regularization.

All models were trained on the **balanced training set** (after applying SMOTE) to give them roughly equal exposure to approved and non-approved loan examples. No extensive hyperparameter tuning was performed; default parameters were used unless mentioned otherwise.

performance table

Model	Accuracy	Macro Precision	Macro Recall	Macro F1-Score
KNN	67%	61%	61%	61%
SVM	75%	72%	64%	65%
Decision Tree	69%	63%	62%	62%
Random Forest	81%	82%	72%	74%
Naive Bayes	81%	89%	68%	71%
ANN	73%	68%	66%	67%
Logistic Regression	80%	80%	70%	72%

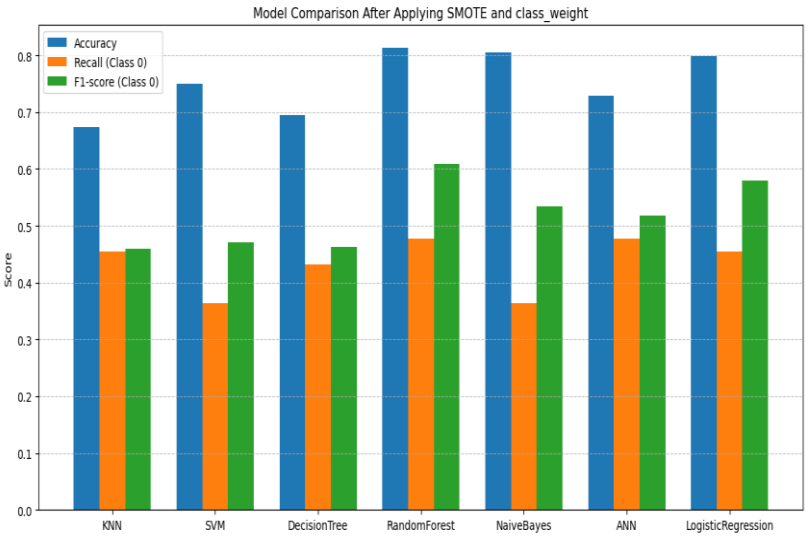
The performance table summarizes the results of all trained models based on key evaluation metrics. Among the models, **Random Forest** and **Naive Bayes** stood out as top performers, both achieving an accuracy of **81%**. Random Forest had the best balance overall with strong macro precision, recall, and F1-score. **Logistic**

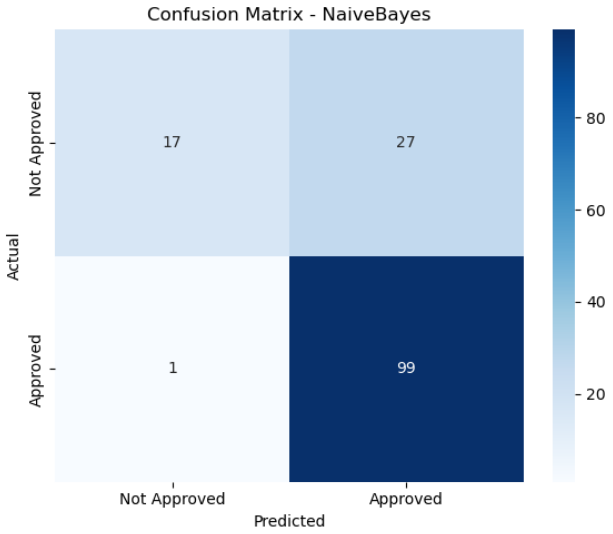
Regression also performed well, reaching 80% accuracy with solid recall and F1. On the other hand, **KNN** showed the weakest performance, with the lowest accuracy (**67%**) and lower macro averages across all metrics. This highlights that while simple models like KNN may struggle, ensemble and probabilistic methods can offer more reliable results for this classification task.

5- Visualization

A confusion matrix is a table used to describe the performance of a classification model by comparing the predicted values to the actual labels. It shows the number of correct and incorrect predictions made by the model, broken down by each class. For the Naive Bayes model, the confusion matrix reveals that it correctly predicted 17 "Not Approved" loans and 99 "Approved" loans. However, it also misclassified 27 "Not Approved" cases as "Approved," and only 1 "Approved" case was incorrectly classified. This indicates that while Naive Bayes was highly accurate in identifying approved loans, it struggled more with detecting rejections.

To better visualize and compare the performance across all models, **a bar plot** was generated showing Accuracy, Recall (Class 0), and F1-score (Class 0) for each classifier. Random Forest stood out with the highest scores overall, particularly in F1-score, making it the most balanced and reliable model. Naive Bayes also performed well, especially in precision, but its lower recall shows it missed many rejections. Models like SVM and ANN had solid, balanced performance, while KNN and Decision Tree lagged behind. The bar plot provides a clear, visual summary of each model's strengths, helping identify which ones offer the best trade-off between accuracy and sensitivity to the minority class.





6- Conslusion

This project focused on automating the loan approval process using machine learning models applied to a real-world dataset of past loan applications. After handling missing values, encoding categorical variables, and applying feature scaling, class imbalance was addressed using SMOTE to balance the training data. Seven classification models were trained and evaluated: KNN, SVM, Decision Tree, Random Forest, Naive Bayes, ANN, and Logistic Regression.

The models were assessed using accuracy, recall, and F1-score especially focusing on class 0 (not approved loans). Random Forest and SVM delivered strong, balanced performance, while Naive Bayes stood out with the highest accuracy and F1-score for identifying rejected loans. ANN achieved the best recall for class 0, making it most sensitive to loan rejections.

KNN showed the weakest performance overall, while Logistic Regression and Decision Tree produced moderate results. The use of SMOTE significantly improved minority class detection across all models, especially ANN and SVM. One key takeaway is that data balancing and preprocessing play a critical role in classification success. The study also highlighted that more complex models are not always better Naive Bayes, despite its simplicity, outperformed deeper trees and even ANN in some aspects. In real applications, the model choice should reflect business priorities: minimizing false approvals may favor ANN or Random Forest, while maximizing overall accuracy might favor Naive Bayes or SVM.

In conclusion, this project provided valuable experience in the end-to-end machine learning pipeline—from data cleaning and transformation to model training and evaluation. With the models achieving around 80% accuracy, it's clear that machine learning can effectively support faster and more consistent loan decisions. Further performance could be enhanced through model tuning or by including additional features like credit scores.