1. **Abstract**

This project develops a predictive model to assess loan default risk using machine learning techniques. Analyzing a dataset of 45,000 loan applications, we identify key factors influencing repayment behavior and build a classification system to support financial decision-making.

**2. Introduction**

**2.1 Problem Statement**

Banks lose approximately $3.5B annually to loan defaults (FDIC 2023). This project addresses:

* High manual review costs in underwriting, inconsistent risk evaluation across branches and regulatory need for explainable AI in lending

**2.2 Dataset Overview**

Source: <https://www.kaggle.com/datasets/udaymalviya/bank-loan-data/data>

* **45,000 records** of personal loans
* **14 features** including:

|  |  |  |  |
| --- | --- | --- | --- |
| Category | Feature | Description | Possible Values / Range |
| Demographics | person\_age | Age of the applicant (in years). | Numerical (e.g., 18-80) |
|  | person\_gender | Gender of the applicant. | Male, Female, Other |
|  | person\_education | Educational background. | High School, Bachelor, Master, etc. |
| Financials | person\_income | Annual income of the applicant (in USD). | Numerical (e.g., 20K−20*K*−200K) |
|  | person\_home\_ownership | Type of home ownership. | RENT, OWN, MORTGAGE |
| Employment | person\_emp\_exp | Years of employment experience. | Numerical (e.g., 0-40) |
| Loan Details | loan\_amnt | Loan amount requested (in USD). | 1K–1*K*–100K |
|  | loan\_intent | Purpose of the loan. | PERSONAL, EDUCATION, MEDICAL, etc. |
|  | loan\_int\_rate | Interest rate on the loan (percentage). | Numerical (e.g., 5%-30%) |
|  | loan\_percent\_income | Ratio of loan amount to income. | Numerical (e.g., 0.1–0.5) |
| Credit History | cb\_person\_cred\_hist\_length | Length of the applicant's credit history (in years). | Numerical (e.g., 0-30) |
|  | credit\_score / CreditScore | Credit score of the applicant. | FICO range (300-850) |
|  | previous\_loan\_defaults\_on\_file | Whether the applicant has previous loan defaults. | Yes, No |
| Additional Metrics | Income (alternative) | Annual borrower earnings. | Numerical (e.g., 20K−20*K*−200K) |
|  | LoanAmount (alternative) | Loan amount requested (alternative column name). | 1K–1*K*–100K |

**A screenshot of a computer

AI-generated content may be incorrect.**

**A screenshot of a computer

AI-generated content may be incorrect.**

[**https://www.sciencedirect.com/science/article/abs/pii/0377221786902092#BIB20**](https://www.sciencedirect.com/science/article/abs/pii/0377221786902092#BIB20)

**3. Methodology and Deployment**

**3.1 Data Preprocessing**

A preview of dataset value for the analysis.

**A screenshot of a computer

AI-generated content may be incorrect.**

Figure 1 - Data description before cleaning process.

Categorical variables frequency to understand to relation with the target variable of each one.

|  |  |  |
| --- | --- | --- |
| Step | Technique | Rationale |
| Categorical Encoding | One-hot (loan purpose) + Ordinal (education) | Algorithm compatibility |
| Data Cleansing | Filter Data | Removed data which didn´t make sense, like customer containing 144 years and more than 60 years work experience. |

**3.2 Feature Engineering**

The analysis did not yield any creatable features.

**3.3 Model Selection**

Algorithms Compared: Logistic Regression, Support Vector Machine (SVM), K-Nearest Neighbors (KNN), Decision Tree, Random Forest and Gradient Boosting (XGBoost).

**3.4 Data Splitting**

Split into Training, Validation, and Test Sets: Divide your data into three separate sets:

**Training Set:** Used to train the machine learning model.

**Validation Set:** Used to tune hyperparameters of the model and evaluate different model choices during development. This helps prevent overfitting to the training data. **Test Set:** Used for a final, unbiased evaluation of the trained model's performance on unseen data. Used *after* you have finalized your model and hyperparameters.

**3.5 Stratification (Important for imbalanced datasets)**

The target variable has imbalanced classes, use stratified splitting to ensure that each split has a representative proportion of each class.

A pie chart with a blue and grey circle

AI-generated content may be incorrect.

Figure 2 – Evidence of Imbalanced data. Not Paid (74,8%) and Paid (25,2%).

**3.6 Model Selection:**

Based on the nature of your problem, the size and characteristics of your data, and your understanding of different classification algorithms, select candidate models to try.

* + 1. **Logistic Regression**

The code implements a Logistic Regression classifier, which is a linear model for binary classification that predicts probabilities of class membership. Logistic regression used for its interpretability, efficiency, and superior performance on classification problems.

* + 1. **Support Vector Machines (SVM)**

SVR is a supervised learning algorithm based on Support Vector Machines (SVM) adapted for regression tasks. It works by finding a hyperplane that best fits the data while allowing deviation (controlled by epsilon). SVR is effective in high-dimensional spaces and can model non-linear relationships using kernel functions.

* + 1. **K-Nearest Neighbors (KNN)**

KNN is a non-parametric, instance-based learning algorithm that classifies data points based on the majority vote of their nearest neighbors. It is simple, intuitive, and effective for classification tasks, particularly when the decision boundary is irregular.

* + 1. **Decision Trees**

A Decision Tree is a non-parametric, hierarchical model that splits the data into subsets based on feature values to make predictions. It is interpretable, manages both numerical and categorical data, and can model non-linear relationships without requiring feature scaling.

* + 1. **Random Forests**

The Random Forest is an ensemble learning method that constructs multiple decision trees during training and outputs the mode (classification) or mean prediction (regression) of the individual trees. It improves predictive accuracy and controls overfitting compared to a single decision tree.

* + 1. **Gradient Boosting Machines (XGBoost)**

XGBoost is an advanced gradient boosting framework that provides state-of-the-art performance for structured/tabular data. It builds decision trees sequentially, with each new tree correcting errors from previous ones, and includes built-in regularization to prevent overfitting.

**4. Model Training and Evaluation**

Each model trained according to the parameters and characteristic considered in tables below.

* 1. **Logistic Regression**

|  |  |  |  |
| --- | --- | --- | --- |
| Parameter | Value | Description | Notes |
| solver | 'liblinear' | Optimization algorithm | - Good for small/medium datasets - Supports L1 & L2 regularization - Works with scaled/unscaled data |
| penalty | 'l2' (default) | Regularization type | L2 (Ridge) regularization applied by default |
| C | 1.0 (default) | Inverse regularization strength | - Smaller = stronger regularization - Default balances fit and generalization |
| class\_weight | None (default) | Class weighting | Assumes balanced classes |
| intercept\_scaling | 1 (default) | Intercept scaling | Default used (no explicit scaling) |
| random\_state | Not set | Random seed | Results may vary between runs |

* + 1. **Key Characteristics**

**Best for**: Small-to-medium sized datasets, **Regularization**: Default L2 (can be changed to L1), **Data Prep**: Works with both scaled and unscaled data (though scale process done) and **Class Handling**: Assumes balanced classes by default.

* 1. **Support Vector Machine (SVM)**

|  |  |  |  |
| --- | --- | --- | --- |
| Parameter | Value | Description | Impact |
| Kernel | 'rbf' | Kernel type | - Non-linear mapping - Manages complex patterns - Alternatives: linear/poly/sigmoid |
| C | 1.0 | Regularization | - Lower = smoother model - Higher = precise fitting (risk overfit) |
| epsilon | 0.1 | Error tolerance | - Larger = flatter function - Smaller = tighter fit |
| gamma | 'scale' | Kernel coefficient | Auto-scaled: 1/ (n\_features \* variance) |
| shrinking | True | Optimization heuristic | Faster convergence |
| Tol | 1e-3 | Stopping tolerance | Smaller = more precise fit |

* + 1. **Key Characteristics**

**Best for**: Non-linear regression problems, **Kernel Strength**: RBF manages complex, non-linear relationships, **Flexibility**: Controlled by C (complexity) and ε (error tolerance) and **Efficiency**: Shrinking heuristic speeds up training.

* 1. **K-Nearest Neighbors**

|  |  |  |  |
| --- | --- | --- | --- |
| Parameter | Value | Description | Impact/Alternatives |
| n\_neighbors | 5 | Number of neighbors | - Smaller k: More sensitive to noise - Larger k: Smoother boundaries |
| weights | 'uniform' | Neighbor weighting | - 'distance': Closer neighbors have more influence |
| algorithm | 'auto' | Nearest neighbor search method | Auto-selects from: - ball\_tree - kd\_tree - brute |
| metric | 'minkowski' (p=2) | Distance metric | - p=2: Euclidean (default) - p=1: Manhattan |
| leaf\_size | 30 (default) | Tree leaf size | Affects tree construction speed |
| p | 2 (implied) | Power parameter | 2 for Euclidean, 1 for Manhattan |

**4.3.1 Key Characteristics**

**Distance-Based**: Relies entirely on feature space distances, **No Training Phase**: Stores all training data (lazy learner), **Scale Sensitive**: Requires feature scaling for superior performance and **Non-Parametric**: Makes no assumptions about data distribution.

* 1. **Decision Tree**

|  |  |  |  |
| --- | --- | --- | --- |
| Parameter | Value | Description | Impact/Alternatives |
| random\_state | 42 | Random seed | Ensures reproducible results |
| criterion | 'gini' | Split quality measure | - 'gini': Faster, works well for most cases - 'entropy': More computationally intensive |
| max\_depth | None | Tree depth limit | - None: Grows until purity or min\_samples - Can set integer value to prevent overfitting |
| min\_samples\_split | 2 | Minimum samples to split node | - Higher values prevent overfitting - Typical range: 2-20 |
| min\_samples\_leaf | 1 | Minimum samples per leaf | - Higher values create smoother boundaries |
| ccp\_alpha | 0 | Cost complexity pruning | - 0: No pruning - >0: Prunes to reduce overfitting |

**4.4.1 Key Characteristics**

**Interpretability**: Simple to visualize and explain, **Non-Parametric**: Makes no assumptions about data distribution, **Feature Importance**: Provides native feature importance scores and **No Scaling Needed**: Works with raw feature values.

* 1. **Random Forest**

|  |  |  |  |
| --- | --- | --- | --- |
| Parameter | Value | Description | Impact/Considerations |
| n\_estimators | 1000 | Number of trees | - Higher = better performance but slower - 100-500 often sufficient |
| random\_state | 42 | Random seed | Ensures reproducible results |
| criterion | 'gini' | Split quality measure | - 'gini': Faster computation - 'entropy': Potentially better splits |
| max\_depth | None | Maximum tree depth | - None: Grows until pure - Can limit to control complexity |
| min\_samples\_split | 2 | Minimum samples to split | - Higher prevents overfitting |
| min\_samples\_leaf | 1 | Minimum leaf samples | - Higher creates smoother boundaries |
| bootstrap | True | Bootstrap sampling | - True: Better generalization - False: Use entire dataset |
| max\_features | 'auto' (√n) | Features per split | - Lower reduces correlation between trees |

**4.5.1 Key Characteristics**

**Ensemble Method**: Combines multiple decision trees, **Robustness**: Manages noise and outliers well, **Feature Importance**: Provides native feature rankings and **Parallelizable**: Trees built independently.

* 1. **Gradient Boosting (XGBoost)**

|  |  |  |  |
| --- | --- | --- | --- |
| Parameter | Value | Description | Impact/Recommendation |
| n\_estimators | 100 | Number of boosting rounds | - Start with 50-200 range - Combine with early\_stopping\_rounds |
| learning\_rate | 0.1 | Step size shrinkage | - Lower = more robust but slower - Typical range: 0.01-0.3 |
| max\_depth | 3 | Tree depth limit | - Shallower trees prevent overfitting - Range 2-8 common |
| random\_state | 42 | Random seed | Ensures reproducibility |
| objective | 'binary:logistic' | Loss function | Standard for binary classification |
| booster | 'gbtree' | Model type | Tree-based (alternatives: gblinear, dart) |
| gamma | 0 | Minimum loss reduction | - Higher = more conservative - Try 0-5 range |
| subsample | 1 | Row sampling | - Lower values prevent overfitting - Try 0.6-1.0 |
| colsample\_bytree | 1 | Column sampling | - Feature subsampling - Try 0.5-1.0 |
| reg\_alpha | 0 | L1 regularization | - Adds feature selection |
| reg\_lambda | 1 | L2 regularization | - Default mild regularization |

* + 1. **Key Characteristics**

**Gradient Boosting**: Sequential tree building correcting errors, **Built-in Regularization**: Multiple parameters control overfitting, **Feature Importance**: Provides native feature rankings and **Manages Missing Values**: Automatically learns imputation.

1. **Model Training and Evaluation**
   1. **Evaluate on the Validation Set:** Evaluate the performance of each trained model on the validation set using appropriate classification metrics.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Model | Accuracy | F1-Score | ROC-AUC | Precision | Recall | PR-AUC |
| Logistic Regression | 0.7935 | 0.2007 | 0.7740 | 0.7172 | 0.1167 | 0.5358 |
| KNN | 0.8330 | 0.5824 | 0.8200 | 0.6555 | 0.5240 | 0.6631 |
| Decision Tree | 0.9007 | 0.7795 | 0.8609 | 0.7698 | **0.7893** | 0.8030 |
| Random Forest | **0.9272** | **0.8246** | **0.9722** | **0.8884** | 0.7693 | **0.9246** |
| XGBoost | 0.9198 | 0.8089 | 0.9688 | 0.8603 | 0.7633 | 0.9154 |

* 1. **Evaluation Metrics**

Random Forest dominates in most metrics (Accuracy, F1-Score, ROC-AUC, Precision and PR-AUC).

The confusion matrix used for analysis of the best model is the Randon Forest matrix contains: True Negative 5104 (TN), False Positive 145 (FP), False Negative 346 (FN) and True Positive 1154 (TP). The dataset is imbalanced, Negative Class (Majority): 5,249 instances versus Positive Class (Minority): 1,500 instances. What means (~78% Negative vs. 22% Positive).

**5.2.1 Model Strengths**

**High Accuracy (92.7%):** The model is correct **93% of the time**, making it reliable for general use. *Business Impact:* Reduces operational costs by minimizing manual verification needs.

**High ROC-AUC (97,2%):** Near-perfect ability to distinguish between classes (**97.2%**). *Business Impact:* Trustworthy for ranking/scoring applications.

**High Precision (88.8%): 89% of predicted positives are correct** (low false alarms). *Business Impact:* Ideal for: **Spam filtering** (few legitimate emails marked as spam). **Fraud detection** (minimizes unnecessary transaction blocks).

**Strong PR-AUC (92,5%):** Maintains high precision even as recall increases, critical for imbalanced data. *Business Impact:* Robust in scenarios where the positive class is rare (e.g., disease screening).

**5.2.2 Key Weaknesses & Risks**

**Moderate Recall (76.9%): 23% of actual positives are missed** (346 FN in your confusion matrix). *Business Risk:* Unacceptable in high-stakes domains

**Class Imbalance Bias: Accuracy (97.2%) > Recall (76.9%)** indicates a bias toward the majority class. This means the model has high precision in predicting negatives but only moderate precision for positive cases. **Business Risk**: Undetected positives could lead to a failure in identifying people who are likely to be good payers.

The importance analysis using Randon Forest, which shows what factors impact most on the results.

A graph with blue squares

AI-generated content may be incorrect.

Figure 4 – Feature Importance in Randon Forest

Three main factors are: Whether the applicant has previous loan defaults (Yes or No), ratio of loan amount to income and interest rate on the loan (percentage).

* 1. **Further Improvements**
* Cross-Validation on training set for hyperparameter tuning.
* Focused on F1 (imbalance) + ROC-AUC (ranking performance).
* Prioritize recall improvement (reduce FN) even at the cost of more FP.

Melhorias

**XGBoost**

Scale\_pos\_weight = 3.5

A close-up of a text

AI-generated content may be incorrect.

Balanceamento amostras

* Undersample , oversample ou SMOTE
* Regularização -> Ridge, Lasso ou ElasticNet

A screenshot of a computer program

AI-generated content may be incorrect.