

PROJECT REPORT

CreditGuard

Credit Risk Scoring System

End-to-End Machine Learning System for Loan Default Prediction

Developed BY LOKPAVAN P

15,000

Dataset Size

21

Features

0.7705

ROC-AUC

0.4142

KS Statistic

Technology Stack

Python • scikit-learn • XGBoost • SHAP • Streamlit • pandas • matplotlib

Domain
Fintech / Credit Risk

Type
Binary Classification

Target
Loan Default (0/1)

1. Executive Summary

CreditGuard is a production-oriented credit risk scoring system built to predict the probability of loan default before disbursement. Designed to mirror the ML systems used at fintech companies like Navi, Slice, and KreditBee, it demonstrates the full data science lifecycle from business problem framing through deployment-ready serving infrastructure.

The system processes loan application data, engineers 21 domain-driven features, trains and compares two machine learning models, evaluates them on business-relevant metrics, and serves predictions through an interactive Streamlit web application with per-decision explainability.

Key Outcomes

Outcome	Detail
Model Performance	ROC-AUC 0.7705 KS Statistic 0.4142 (exceeds ≥ 0.30 threshold for production credit models)
Business Value	Optimal threshold at 0.553 minimises total credit loss cost; estimated business cost ₹1.39 Cr on validation set
Class Imbalance	15% default rate handled via class_weight='balanced' and scale_pos_weight; no synthetic oversampling used
Explainability	Exact log-odds decomposition for Logistic Regression; SHAP TreeExplainer for tree models — per-customer and global
Deployment	Streamlit app with real-time scoring, risk tier classification, and SHAP waterfall chart for every decision

2. Business Problem

2.1 Problem Statement

Every time a lending institution approves a loan, it takes on credit risk — the probability that the borrower will fail to repay. In India's booming digital lending market, fintechs process thousands of loan applications daily. Manual underwriting cannot scale; traditional rule-based systems miss complex patterns; and poor default prediction erodes profitability.

CreditGuard answers a single business question:

"Given a customer's profile at the time of loan application, what is the probability they will default — and should we approve, review, or reject this loan?"

2.2 Why This Problem Matters in Fintech

Challenge	Business Impact
15–25% default rates in unsecured lending	Directly erodes Net Interest Margin; 1% improvement in detection can save crores annually
Manual underwriting doesn't scale	Bottleneck at ₹100 Cr+ monthly disbursement volume; every hour of delay costs potential revenue
Regulatory explainability (RBI circular)	Lenders must explain credit decisions; black-box models expose institutions to regulatory risk
Asymmetric cost of errors	Missing a default (FN) costs ~₹50,000 in write-off; rejecting a good borrower (FP) costs ~₹10,000 in lost yield
Portfolio diversification	Model-driven scoring enables data-backed risk-appetite decisions across loan purpose, geography, and tenure

2.3 Business Decision Framework

The model outputs a continuous default probability score (0–1). Business teams translate this into actionable decisions via configurable risk tiers:

Tier	Threshold	Action	Rationale
LOW RISK	< 20%	Auto-approve	Below portfolio loss appetite; standard terms
MEDIUM RISK	20% – 45%	Manual underwriter review	Borderline cases requiring human judgement or additional documents
HIGH RISK	> 45%	Auto-reject / Escalate	Exceeds risk tolerance; senior credit officer review for large-ticket cases

Thresholds are configurable and should be recalibrated quarterly based on portfolio loss experience and macroeconomic conditions.

Lokpavan P

3. Dataset Overview

3.1 Data Source & Generation

The dataset is a synthetic fintech loan application dataset generated to mirror real-world lending data distributions. It contains 15,000 loan applications spanning January 2021 to December 2022, designed to replicate the kind of data available at digital lending companies in India.

Synthetic data was chosen to avoid privacy concerns while maintaining statistical realism. The default probability for each record was computed using a logistic function of known risk factors (credit score, past defaults, debt-to-income ratio, employment status) with added random noise — ensuring the dataset presents genuine ML challenges including class imbalance and correlated features.

3.2 Dataset Statistics

15,000	17	14.75%	2 Years
Total Records	Features (Raw)	Default Rate	Date Span

3.3 Feature Descriptions

Feature	Type	Description
annual_income	Numeric	Applicant's annual income in INR (₹1L – ₹50L)
loan_amount	Numeric	Requested loan amount in INR
interest_rate	Numeric	Annual interest rate applicable (7% – 28%)
credit_score	Numeric	CIBIL/bureau credit score (300 – 900)
credit_history_length	Numeric	Years of credit history (4% missing — median imputed)
past_defaults	Numeric	Count of prior loan defaults (0–3); strongest risk predictor
debt_to_income	Numeric	Monthly EMI / Monthly Income ratio; clipped at 3.0
loan_to_income	Numeric	Loan amount / Annual income; clipped at 5.0
employment_status	Categorical	Salaried / Self-Employed / Business / Unemployed
loan_purpose	Categorical	Home / Education / Business / Vehicle / Personal / Medical
num_credit_inquiries	Numeric	Credit bureau inquiries in last 6 months (2% missing)
loan_default	Binary Target	1 = Default, 0 = No Default (15% positive class)

3.4 Key EDA Insights

Exploratory data analysis revealed the following business-critical patterns:

- Past defaults are the strongest predictor — customers with 2+ prior defaults show a default rate 4–5x higher than customers with no history of defaults.
- Credit score is inversely correlated — applicants scoring below 650 show >25% default rates, versus <8% for those above 750.
- Unemployed applicants default at approximately 3x the rate of salaried applicants, confirming income stability as a key risk factor.
- Debt-to-income (DTI) ratio above 50% is a danger zone — default rates spike non-linearly, which is better captured by XGBoost than Logistic Regression.
- Personal and Medical loans carry higher default risk than Home and Vehicle loans — likely because they lack asset backing as collateral.
- Missing data is low (<5%) and appears Missing At Random (MAR); median imputation is statistically valid.

4. Feature Engineering

4.1 Pipeline Overview

The feature engineering pipeline transforms raw application data into a model-ready feature matrix. All transformations are fitted exclusively on training data and applied to validation/test sets to prevent data leakage — a critical requirement in production ML systems.

4.2 Time-Based Train/Validation Split

Unlike standard random splitting, CreditGuard uses a time-ordered split: the earliest 80% of applications form the training set, and the most recent 20% form the validation set. This correctly simulates how the model will be deployed — always predicting on future applicants using patterns learned from the past.

Split	Records	Default Rate	Period
Training Set	12,000	14.69%	Jan 2021 – Aug 2022
Validation Set	3,000	15.00%	Aug 2022 – Dec 2022

Random splits would cause data leakage because future economic conditions, fraud patterns, or credit bureau changes would "leak" into the training set, inflating apparent performance metrics.

4.3 Missing Value Imputation

Three features contain missing values: credit_history_length (4%), employment_years (3%), and num_credit_inquiries (2%). Median imputation was chosen over mean imputation because financial features are right-skewed; the median is more robust to outliers (e.g., a CEO with 40 years of credit history should not inflate imputed values for a 25-year-old first-time borrower).

4.4 Engineered Features

Three domain-driven features were derived to capture business logic not present in the raw data:

Feature	Formula	Business Rationale
income_per_year_employed	annual_income / (employment_years + 1)	Measures earning power relative to experience; low value may indicate inflated income claims
emi_burden_score	Monthly EMI / Monthly Income (clipped at 5)	Direct measure of monthly payment stress; values >0.4 indicate over-leveraged borrowers

credit_risk_index	$(\text{past_defaults} \times 2) + (\text{inquiries} \times 0.5) - (\text{credit_score} - 600) / 100$	Composite risk signal combining bureau history, inquiry behaviour, and credit score in a single feature
-------------------	---	---

4.5 Categorical Encoding

- employment_status (4 levels): One-Hot Encoded into binary indicator columns (emp_Salaried, emp_Self-Employed, emp_Business, emp_Unemployed). Drop_first=False to preserve all signals including the emp_Unemployed flag, which is the highest-risk category.
- loan_purpose (6 levels): Frequency encoded — each category replaced by its proportion in the training set. This preserves ordinal information about loan purpose risk without creating 5 additional columns, reducing dimensionality.

4.6 Feature Scaling

StandardScaler is applied to all numeric features. This is required for Logistic Regression (which is sensitive to feature magnitude) and beneficial for gradient boosting convergence. The scaler is fitted only on training data and applied to validation/test sets.

Final feature matrix: 21 features per application (13 original numeric + 3 engineered + 4 employment OHE + 1 loan purpose frequency).

5. Model Training

5.1 Model Selection Strategy

Two models were trained and compared: a Logistic Regression baseline (interpretable, fast, regulatory-friendly) and a Gradient Boosting model as the primary production candidate (handles non-linearity, interactions, and missing value patterns that LR cannot).

5.2 Class Imbalance Handling

With 14.75% defaults, the dataset has a 5.8:1 class ratio. Two approaches were used — neither relies on SMOTE or random oversampling (which can introduce spurious patterns):

- Logistic Regression: `class_weight='balanced'` — automatically scales each sample's weight inversely proportional to class frequency. Minority class (defaults) receives ~5.8x higher weight during gradient computation.
- Gradient Boosting: `sample_weight` vector passed during fit, with `scale_pos_weight = neg/pos = 5.81`. This tells the model that each default is 5.81 times more important to classify correctly than a non-default.

5.3 Hyperparameter Tuning

RandomizedSearchCV with 5-fold StratifiedKFold cross-validation and 20 random parameter combinations was used. StratifiedKFold ensures each fold maintains the 15% default rate, preventing optimistic bias from folds with fewer positives.

Parameter	Search Space	Best Value
<code>n_estimators</code>	[100, 200, 300]	100
<code>max_depth</code>	[3, 4, 5]	4
<code>learning_rate</code>	[0.05, 0.10, 0.15]	0.05
<code>subsample</code>	[0.7, 0.8, 0.9]	0.7
<code>min_samples_leaf</code>	[10, 20, 30]	10
<code>max_features</code>	['sqrt', 0.8, 1.0]	'sqrt'

Best Cross-Validation AUC (GradientBoosting): 0.7801 | Best Cross-Validation AUC (Logistic Regression): 0.7875

6. Model Evaluation

6.1 Why Accuracy is NOT the Right Metric

A naive model that predicts 'No Default' for every applicant achieves 85% accuracy — but catches zero actual defaults, allowing 100% of credit losses to pass through unchecked. Accuracy is not used as a metric in this project.

The following metrics are used instead, each serving a distinct business purpose:

- ROC-AUC: Measures the model's ability to rank-order customers by risk, regardless of threshold. An AUC of 0.77 means the model correctly identifies the higher-risk applicant in 77% of random borrower pairs.
- KS Statistic: Industry standard for credit scorecards. Measures maximum separation between cumulative default and non-default distributions across all score thresholds. KS > 0.30 is considered production-grade.
- Gini Coefficient: Equals $2 \times \text{AUC} - 1$. Widely used in Basel II/III regulatory capital models. Gini > 0.50 is considered a good credit model.
- PR-AUC (Precision-Recall): More informative than ROC under class imbalance. Shows the trade-off between catching defaults (recall) and avoiding false alarms (precision).
- Business Cost Metric: $\text{FN} \times ₹50,000 + \text{FP} \times ₹10,000$ — reflects asymmetric real-world costs and is used to find the optimal classification threshold.

6.2 Results Summary

Model	ROC-AUC	KS Stat	Gini	PR-AUC	Opt. Threshold
Logistic Regression ✓ BEST	0.7705	0.4142	0.5410	0.4222	0.553
Gradient Boosting	0.7639	0.4071	0.5279	0.4112	0.536

Logistic Regression was selected as the best model (AUC 0.7705 vs 0.7639). This is not unusual — with well-engineered features and proper scaling, LR often matches or outperforms tree models, especially on smaller datasets. It also offers regulatory advantages due to its inherent interpretability.

6.3 Confusion Matrix at Optimal Threshold (Logistic Regression)

At the business-cost-optimised threshold of 0.553:

	Predicted: No Default	Predicted: Default

Actual: No Default	TN = 1,997	FP = 553
Actual: Default	FN = 168	TP = 282

- Recall (Sensitivity): 62.7% — the model catches 282 out of 450 actual defaults in the validation set.
- Precision: 33.8% — of all flagged applications, 34% are genuine defaults. The remaining 66% are borderline cases that still benefit from underwriter review.
- The threshold 0.553 was chosen over the default 0.5 because it minimises total business cost (₹13.93 Cr estimated vs higher at other thresholds).

7. Explainability

7.1 Why Explainability is Mandatory in Credit Lending

The Reserve Bank of India (RBI) and global regulatory frameworks require that automated credit decisions be explainable and auditable. A model that says "rejected" without justification exposes the lender to:

- Regulatory penalties for opaque decision-making
- Fair lending litigation if protected classes are disproportionately affected
- Customer complaints that cannot be resolved without a clear explanation
- Operational risk if model errors cannot be diagnosed

7.2 Explanation Method: Log-Odds Decomposition

For the Logistic Regression model, CreditGuard uses exact log-odds decomposition — not an approximation. The log-odds of default can be decomposed as:

$$\text{log-odds} = \text{intercept} + \sum (\text{coefficient}_i \times \text{scaled_feature}_i)$$

Each term $\text{coefficient}_i \times \text{scaled_feature}_i$ is the exact contribution of feature i to the log-odds of default. Positive values increase default risk; negative values decrease it. This decomposition is mathematically exact (not an approximation like LIME) and directly auditable by regulators.

7.3 Global Feature Importance

Across all 3,000 validation customers, the top risk drivers by mean absolute log-odds contribution are:

#	Feature	Mean Contribution	Business Interpretation
1	<code>debt_to_income</code>	0.539	Monthly payment burden relative to income — strongest single predictor
2	<code>credit_risk_index</code>	0.340	Composite engineered feature capturing bureau history holistically
3	<code>credit_score</code>	0.311	CIBIL score — validated third-party creditworthiness signal
4	<code>past_defaults</code>	0.277	Prior default history — most intuitive risk flag for underwriters
5	<code>loan_to_income</code>	0.259	Loan size relative to annual income — measures repayment capacity

7.4 Per-Customer Explanation (Example)

For every prediction, the Streamlit app generates a waterfall chart showing how each feature pushed the risk score up (red) or down (blue) for that specific customer. Example for a high-risk applicant:

- credit_risk_index: +1.136 — composite of 2 past defaults and 6 recent inquiries significantly increases risk
- emp_Unemployed: +0.801 — unemployment flag is a strong categorical risk driver
- credit_score: +0.543 — low CIBIL score of 540 increases default probability
- loan_to_income: -0.387 — relatively moderate loan amount partially mitigates risk
- debt_to_income: -0.378 — DTI ratio is within acceptable range, offsetting some risk

This per-feature breakdown enables underwriters to understand, challenge, and override model decisions with documented rationale — a key requirement for responsible AI in lending.

8. System Architecture

8.1 Project Structure

File / Directory	Purpose
data/raw/loan_data.csv	Raw synthetic dataset (15,000 records, 17 columns)
data/processed/	Train and validation CSVs after feature engineering
notebooks/eda.ipynb	Exploratory Data Analysis with business insights and visualisations
src/feature_engineering.py	Full preprocessing pipeline: imputation, encoding, scaling, time-based split
src/train_model.py	Model training: LR baseline + GradientBoosting with RandomizedSearchCV
src/evaluate.py	Business-metric evaluation: AUC, KS, Gini, threshold optimisation, all plots
src/predict.py	Single/batch prediction + log-odds decomposition explainability
src/artifacts/	Saved models (.pkl) and preprocessing pipeline for serving
app.py	Streamlit web application with interactive UI, risk tier display, and explanation chart
reports/	Auto-generated evaluation plots: ROC curves, KS curves, score distributions, SHAP charts

8.2 Data Flow

Raw CSV → **feature_engineering.py** → **train_model.py** → **evaluate.py** → **predict.py** → **app.py**

Each stage saves its outputs (processed CSVs, model artifacts, evaluation reports) so the pipeline can be resumed at any point. The preprocessing pipeline is serialised as preprocessing.pkl, ensuring that the exact same transformations applied during training are applied at inference time.

8.3 Streamlit Application

The Streamlit app provides a no-code interface for credit analysts and underwriters. Features include:

- Sidebar form for entering all customer application details
- Real-time computation of derived financial metrics (DTI, EMI, loan-to-income) before scoring
- Instant default probability score with progress bar and colour-coded risk tier badge
- Business recommendation (Approve / Review / Reject) based on the configurable threshold

- Feature contribution waterfall chart showing the top 12 drivers for the specific prediction
- Explanation method label for transparency (log-odds decomposition for LR; SHAP for tree models)

Lokpavan P

9. Production Deployment Strategy

9.1 Serving Architecture

In a production fintech environment, the scoring system would be deployed as a microservice:

Layer	Implementation
API Layer	FastAPI container exposing POST /score endpoint; accepts JSON application payload, returns probability + risk tier + SHAP values
Feature Service	Pulls bureau data (CIBIL API), bank statement features (account aggregator), and computes derived features using the serialised preprocessing.pkl pipeline
Model Registry	MLflow tracks all model versions, hyperparameters, and metrics; champion model is deployed, challenger runs in shadow mode on 5% of traffic
Audit Log	Every prediction stored in PostgreSQL: input features, output score, risk tier, explanation values, timestamp, underwriter override (if any)
Monitoring	Grafana dashboards for score distribution drift, approval rate trends, actual default rates by vintage (as loans mature)

9.2 Model Monitoring & Drift Strategy

Credit risk models degrade over time due to changes in borrower behaviour, macroeconomic conditions, and product offerings. The following monitoring framework is recommended:

Risk	Metric	Frequency	Action Trigger
Data drift	PSI on key features	Weekly	PSI > 0.20 → alert and investigate
Concept drift	Actual vs predicted default rate	Monthly	Deviation > 20% from expected → retrain candidate
Model staleness	AUC on settled loan cohort	Monthly	AUC drops > 3% from baseline → trigger retraining
Feature failure	API availability, null rates	Real-time	Bureau API outage → fallback to rule-based scorecard

Retraining uses a rolling 18-month window with recent vintages overweighted (last 6 months at 2x weight). New models undergo 30-day shadow deployment before promotion to champion.

10. Future Improvements

Improvement	Business Value	Complexity
Bureau feature integration (CIBIL API)	+5–8% AUC improvement; trade-line level data unavailable in synthetic dataset	Medium
Bank statement analysis	Detects income misrepresentation; behavioural signals unavailable from application form alone	High
Vintage-based survival modelling	Time-to-default as target instead of binary; better for long-tenure loans	High
Ensemble stacking (LR + GBM + CatBoost)	+2–3% AUC; diversity of model architectures reduces variance	Medium
Fairness audit by demographic group	RBI and SEBI increasingly scrutinise disparate impact on protected groups	Medium
WOE / IV binning transformation	Basel-compliant scorecard format; easier for risk teams to interpret and adjust	Medium
Real-time streaming (Kafka + Flink)	Sub-100ms decision latency for embedded lending at point of sale	High
A/B testing framework	Rigorous champion-challenger testing on live traffic with statistical significance testing	Medium

11. Technology Stack

Category	Tool / Library	Usage in Project
Language	Python 3.11+	All pipeline components
Data	pandas, numpy	Data loading, manipulation, feature engineering
ML	scikit-learn	Logistic Regression, GradientBoosting, StandardScaler, RandomizedSearchCV, metrics
ML	XGBoost	Primary boosting model (installed separately; GBM fallback if unavailable)
Explainability	SHAP	TreeExplainer for XGBoost; log-odds decomposition native fallback for LR
Visualisation	matplotlib, seaborn	EDA plots, ROC/KS curves, score distributions, feature contribution charts
App	Streamlit	Interactive web UI for credit analysts and underwriters
Serialisation	pickle	Model and preprocessing pipeline persistence for serving
Math	scipy	expit (sigmoid) function for log-odds to probability conversion

12. Conclusion

CreditGuard demonstrates a complete, production-oriented credit risk scoring system built to the standard expected of a Data Scientist at a fintech lending company. The project addresses each layer of the problem:

- Business understanding: Clear problem framing with asymmetric cost awareness and regulatory context
- Data quality: Realistic missing value handling, time-based splits, and domain-appropriate imputation
- Feature engineering: Three engineered features capturing domain logic unavailable in raw application data
- ML rigour: Proper imbalance handling, cross-validated hyperparameter tuning, and multiple evaluation metrics
- Explainability: Mathematically exact per-decision explanations satisfying regulatory requirements
- Deployment thinking: Serving architecture, monitoring strategy, and failure mode planning

The model achieves ROC-AUC of 0.7705 and KS Statistic of 0.4142 on a held-out validation set — metrics consistent with production credit scorecards in India's digital lending market. With bureau feature integration and bank statement analysis, AUC improvements of 5–8% are achievable, bringing performance in line with industry-leading models.

This project is structured for easy extension: adding new features, swapping in XGBoost or CatBoost, integrating with a real bureau API, or deploying behind a FastAPI service all require changes to single, well-isolated modules. The separation of feature engineering, training, evaluation, and serving concerns reflects production-grade software design.

CreditGuard v1.0

Built as a Data Science I Portfolio Project | Fintech Credit Risk Domain