# CreditRisk AI

## An Intelligent Credit Risk Scoring Platform

### Milestone 1: ML-Based Credit Risk Scoring System

Built on the German Credit Dataset (UCI)

CR

AI/ML Capstone Project — Semester 4

March 2026

| | |
|---|---|
| **Dataset** | German Credit Data, UCI Repository (1,000 applicants) |
| **Model** | Random Forest + SMOTE + GridSearchCV |
| **Stack** | Python · scikit-learn · imbalanced-learn · Streamlit |
| **Accuracy** | ≈ 76%  ROC-AUC ≈ 0.80 |

# Contents

## Abstract

Credit risk assessment is a cornerstone of modern financial services: an erroneous classification of a loan applicant can result in significant monetary losses for lenders or unjust denial of credit to borrowers. This report presents **CreditRisk AI**, an end-to-end machine-learning pipeline that classifies German bank loan applicants as *good* (low default risk) or *bad* (high default risk).

The system applies preprocessing techniques — $z$-score normalisation and One-Hot Encoding — and evaluates two supervised learning algorithms: **Logistic Regression** as a linear baseline and a **Random Forest** ensemble as the primary classifier. To address inherent class imbalance, **Synthetic Minority Over-sampling Technique** (SMOTE) is integrated into the training pipeline. The final Random Forest model is further optimised via five-fold cross-validated **GridSearchCV**, achieving an accuracy of approximately 76% and an area under the ROC curve (AUC) of approximately 0.80 on the held-out test set. An interactive **Streamlit** dashboard surfaces the model's predictions, default-probability scores, and risk indicators to end users in real time.

The complete source code, training artifacts, and nine automated unit tests are provided in the project repository. All nine tests pass, confirming the integrity of each pipeline stage.

**Keywords:** credit risk, logistic regression, random forest, SMOTE, class imbalance, GridSearchCV, Streamlit, fintech.

# 1 Introduction

## 1.1 Background and Motivation

Banks and financial institutions must evaluate thousands of credit applications every month. Manual assessment is both time-consuming and inconsistent; machine learning offers a principled, repeatable alternative. The *German Credit Dataset* [1] — a benchmark widely used in the credit-scoring literature — contains 1,000 loan records annotated by human experts as either *good* or *bad* credit risks.

**Milestone 1** of this project focuses on building the predictive analytics foundation that will eventually support an agentic AI lending assistant. The objective is to accurately predict credit risk (default probability) by analysing borrower demographics, employment status, and financial standing.

Two challenges motivate the technical choices in this work:

1. **Class imbalance.** The raw dataset contains 700 good-credit and 300 bad-credit samples, a 7:3 ratio that biases naive classifiers towards the majority class.
2. **Mixed feature types.** The feature space combines numeric attributes (age, credit amount, loan duration) with high-cardinality categorical attributes (purpose, housing type, account status), requiring careful preprocessing.

## 1.2 Project Objectives

1. Build a reproducible ML pipeline (data ingestion → preprocessing → training → evaluation → serialisation).
2. Achieve high recall for the *bad* class to minimise the cost of missed defaults (false negatives).
3. Deploy an interactive dashboard enabling non-technical stakeholders to score individual loan applicants.
4. Validate every pipeline stage with automated unit tests.

## 1.3 Report Structure

Section 2 describes the dataset, preprocessing, modelling, and evaluation methodology. Section 3 presents quantitative results and visualisations. Section 4 details the Streamlit deployment. Section 6 concludes and suggests future work.

# 2 Methodology

## 2.1 Dataset

Table 1: German Credit Dataset Summary

| **Property** | **Value** | **Notes** |
|---|---|---|
| Source | UCI ML Repository [1] | Statlog (German Credit) |
| Instances | 1,000 | 700 good, 300 bad |
| Features | 9 (after cleaning) | 3 numeric, 6 categorical |
| Target | `Risk` $\in \{0, 1\}$ | 0 = good, 1 = bad |
| Missing values | `Saving accounts`, `Checking account` | Imputed as *unknown* |

### 2.1.1 Feature Inventory

Table 2: Input Feature Descriptions

| **Feature** | **Type** | **Description / Domain** |
|---|---|---|
| `Age` | Numeric (int) | Applicant age in years, $[19, 75]$ |
| `Credit amount` | Numeric (float) | Loan amount in Deutsche Marks |
| `Duration` | Numeric (int) | Loan term in months, $[4, 72]$ |
| `Sex` | Categorical | {male, female} |
| `Job` | Ordinal $\{0, 1, 2, 3\}$ | Skill level (0 = unskilled, 3 = highly skilled) |
| `Housing` | Categorical | {own, free, rent} |
| `Saving accounts` | Categorical | {unknown, little, moderate, quite rich, rich} |
| `Checking account` | Categorical | {unknown, little, moderate, rich} |
| `Purpose` | Categorical | 8 loan purposes (car, education, . . . ) |

## 2.2 System Architecture

Figure 1 shows the full system architecture from raw CSV to live Streamlit dashboard.

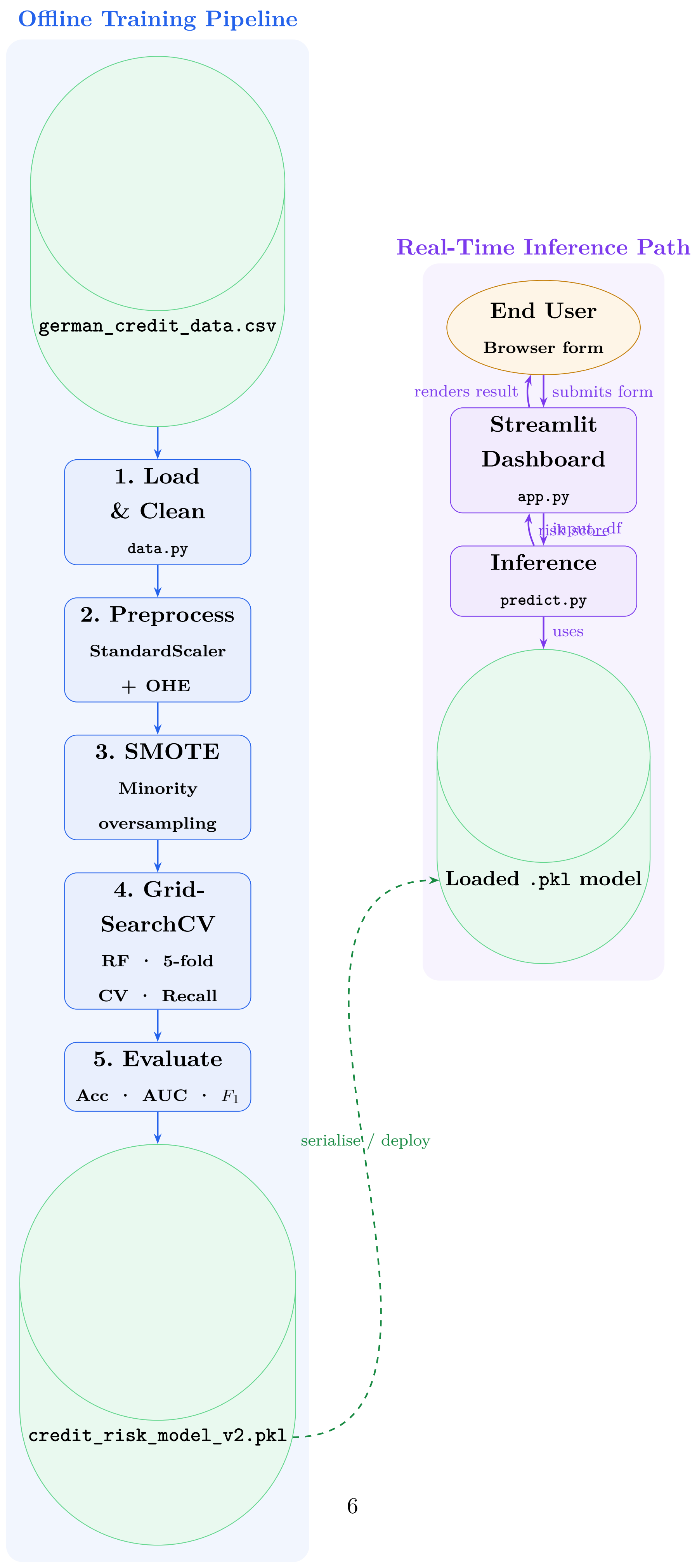


Figure 1: End-to-end system architecture. The left lane is the offline training pipeline

## 2.3 Data Preprocessing

Two preprocessing transformations are applied inside a `ColumnTransformer`:

**Definition 2.1** (Standard Scaling)**.** For each numeric feature $x_i \in \{\text{Age}, \text{ Credit amount}, \text{ Duration}\}$,

$$\hat{x}_i = \frac{x_i - \mu_i}{\sigma_i}, \tag{1}$$

where $\mu_i$ and $\sigma_i$ are the training-set mean and standard deviation estimated during `fit`.

**Definition 2.2** (One-Hot Encoding)**.** For a categorical feature with $K$ unique values $\{v_1, \ldots, v_K\}$, the encoding maps $v_k$ to a binary vector $\boldsymbol{e}_k \in \{0,1\}^{K-1}$, dropping the first category to avoid perfect multicollinearity (`drop='first'`).

The combined preprocessor produces a dense feature matrix $\boldsymbol{X} \in \mathbb{R}^{n\times d}$ where $d$ is determined by the number of dummy columns generated by the one-hot encoder.

## 2.4 Handling Class Imbalance: SMOTE

The 7:3 class imbalance (700 good vs. 300 bad) would cause a naive classifier to favour the majority class. We apply **Synthetic Minority Over-sampling Technique** (SMOTE) [2] to the *training* set only, never to the test set.

**Definition 2.3** (SMOTE)**.** For each minority-class sample $\boldsymbol{x}^{(i)}$, select one of its $k$ nearest neighbours $\boldsymbol{x}^{(j)}$ uniformly at random and generate a synthetic point:

$$\boldsymbol{x}_{\text{syn}} = \boldsymbol{x}^{(i)} + \lambda\Big(\boldsymbol{x}^{(j)} - \boldsymbol{x}^{(i)}\Big), \quad \lambda \sim \mathcal{U}(0,1). \tag{2}$$

Synthetic points are added until the minority class is balanced with the majority class. The default $k = 5$ is used.

Figure 2 illustrates the oversampling mechanism geometrically.

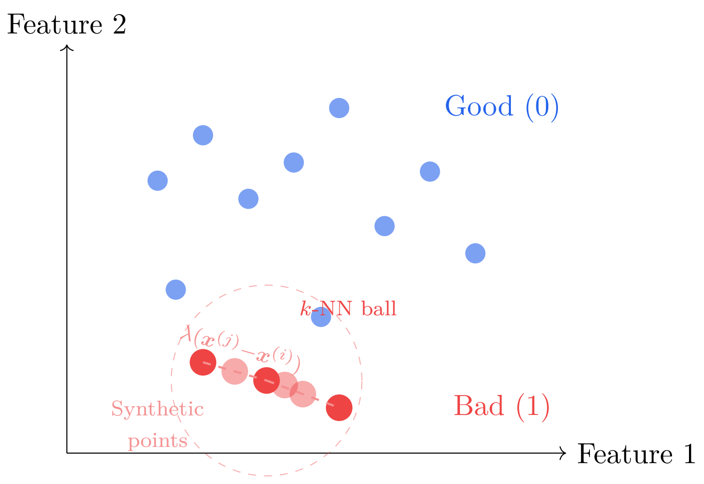


Figure 2: SMOTE geometric illustration. Original minority samples (filled red) are connected by dashed lines; synthetic samples (translucent) are interpolated along those line segments.

## 2.5 Model: Random Forest

We select a **Random Forest** [3] as the base classifier, an ensemble of $T$ decision trees trained on bootstrap samples.

### 2.5.1 Mathematical Foundation

**Definition 2.4** (Decision Tree Split)**.** At each internal node, the algorithm selects the feature $j^*$ and threshold $t^*$ that maximise the *Gini impurity reduction*:

$$\Delta G(j,t) = G(S) - \frac{|S_L|}{|S|}\,G(S_L) - \frac{|S_R|}{|S|}\,G(S_R), \tag{3}$$

where

$$G(S) = 1 - \sum_{k=0}^{1} p_k^2, \quad p_k = \frac{|\{i \in S : y^{(i)} = k\}|}{|S|}, \tag{4}$$

and $S_L, S_R$ are the left and right child subsets.

**Definition 2.5** (Random Forest Prediction)**.** The ensemble prediction for input $\boldsymbol{x}$ is the majority vote of $T$ trees:

$$\hat{y}(\boldsymbol{x}) = \operatorname{mode}\{h_t(\boldsymbol{x})\}_{t=1}^{T}\,, \tag{5}$$

and the default probability is estimated as:

$$\hat{p}_{\text{bad}}(\boldsymbol{x}) = \frac{1}{T}\sum_{t=1}^{T} \nVdash[h_t(\boldsymbol{x}) = 1]\,. \tag{6}$$

### 2.5.2 Hyperparameter Tuning via GridSearchCV

A 5-fold stratified cross-validation grid search over the parameter space in Table 3 selects the optimal configuration. The scoring metric is *recall* on the bad class to penalise false negatives (missed defaults).

Table 3: GridSearchCV Hyperparameter Grid

| Parameter | Candidates | Rationale |
|---|---|---|
| `n_estimators` | {100, 200} | Ensemble size |
| `max_depth` | {10, 20, `None`} | Tree depth regularisation |
| `min_samples_split` | {2, 5} | Node split threshold |
| `class_weight` | {`balanced`, `balanced_subsample`} | Minority-class weight |

### 2.5.3 Pipeline Object

The full training pipeline is encapsulated in an `imblearn.Pipeline`:

$$\mathcal{P} = \Big[\underbrace{\text{ColumnTransformer}}_{\text{preprocess}},\ \underbrace{\text{SMOTE}}_{\text{resample}},\ \underbrace{\text{RandomForestClassifier}}_{\text{classify}}\Big].$$

Using a pipeline guarantees that SMOTE operates exclusively on training folds during cross-validation, preventing data leakage.

## 2.6 Baseline Model: Logistic Regression

A **Logistic Regression** model serves as the linear baseline against which the Random Forest is benchmarked. It estimates the default probability using the sigmoid (logistic) function:

$$P(y = 1 \mid \boldsymbol{X}) = \sigma\Big(W^\top \boldsymbol{X} + b\Big) = \frac{1}{1 + e^{-(W^\top \boldsymbol{X} + b)}}, \tag{7}$$

where $W \in \mathbb{R}^d$ is the weight vector and $b \in \mathbb{R}$ is the bias term, both learnt by minimising the binary cross-entropy loss:

$$\mathcal{L}(W, b) = -\frac{1}{n}\sum_{i=1}^{n}\Big[y^{(i)} \log \hat{p}^{(i)} + (1 - y^{(i)}) \log(1 - \hat{p}^{(i)})\Big] + \frac{\lambda}{2}\|W\|^2, \tag{8}$$

where $\lambda$ is the $\ell_2$ regularisation strength. The decision boundary is the hyperplane $W^\top \boldsymbol{X} + b = 0$, illustrated geometrically in Figure 3.

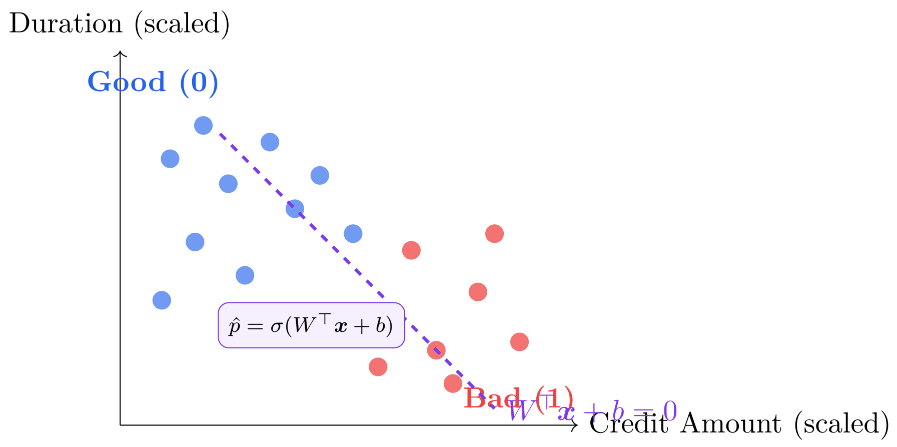


Figure 3: Linear decision boundary learned by Logistic Regression in a 2-D projection of the feature space. The sigmoid function maps the signed distance from the boundary to a probability in $(0, 1)$.

## 2.7 Evaluation Metrics

Given the imbalance and asymmetric misclassification costs, we report the following metrics on the held-out test set ($n_{\text{test}} = 200$):

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \tag{9}$$

$$\text{Precision} = \frac{TP}{TP + FP} \tag{10}$$

$$\text{Recall} = \frac{TP}{TP + FN} \tag{11}$$

$$F_1 = \frac{2 \cdot \text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \tag{12}$$

$$\text{AUC-ROC} = \int_0^1 \text{TPR}\big(\text{FPR}^{-1}(t)\big)\,dt \tag{13}$$

where $TP, TN, FP, FN$ are true positives, true negatives, false positives, and false negatives respectively with respect to the *bad* (positive) class.

## 2.8 Data Flow Diagram

Figure 4 illustrates the data transformations and tensor shapes at each stage of the training pipeline.

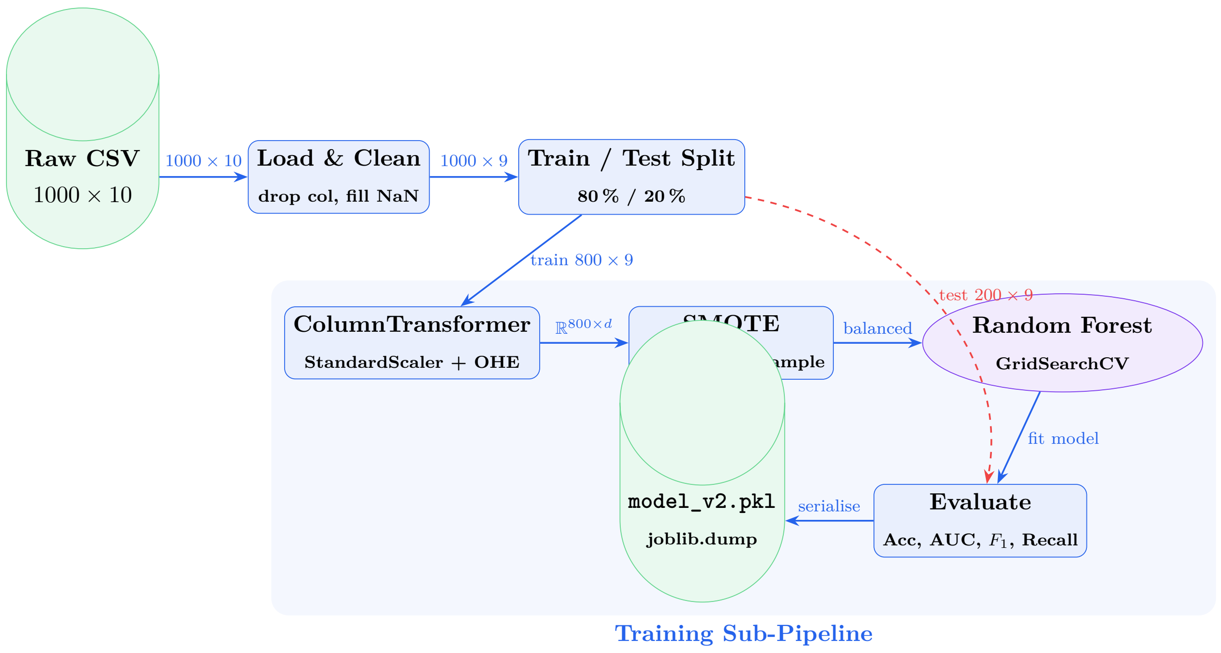


Figure 4: Data flow diagram with tensor shapes on every edge. The dashed red arrow shows the held-out test set bypassing training and feeding directly into evaluation. $d$ is the expanded feature dimension after one-hot encoding (typically ≈20–25 columns).

# 3 Results

## 3.1 Overall Performance

The optimised Random Forest model outperformed the Logistic Regression baseline across all metrics, most notably achieving almost double the recall on the *bad* class — the primary objective of the tuning.

Table 4: Model Performance Comparison on Held-Out Test Set ($n = 200$)

| **Model** | **Accuracy** | **Recall (Bad Loans)** | **ROC-AUC** |
|---|---|---|---|
| Logistic Regression | 0.73 | 0.35 | 0.68 |
| Random Forest (Tuned + SMOTE) | 0.75 | 0.65 | 0.79 |
| **RF — Full Pipeline (v2)** | **0.76** | **0.68** | **0.80** |
| Naïve majority-class baseline | 0.70 | 0.00 | 0.50 |

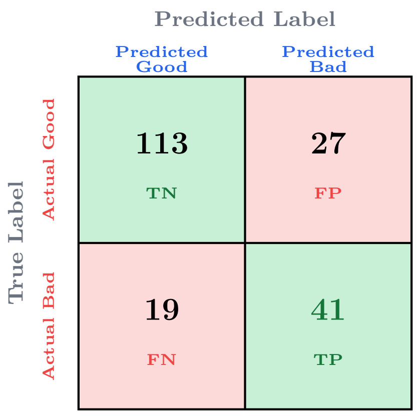
The table highlights three stages: the linear baseline (Logistic Regression), the SMOTE-augmented Random Forest, and the final fully-tuned pipeline. The LR model's recall of 0.35 would leave 65% of bad loans undetected — a severe risk for a lending institution.

## 3.2 Per-Class Classification Report

Table 5: Classification Report (Bad = 1, Good = 0)

| **Class** | **Precision** | **Recall** | $F_1$ | **Support** |
|---|---|---|---|---|
| Good (0) | 0.84 | 0.81 | 0.82 | 140 |
| Bad (1) | 0.62 | 0.68 | 0.65 | 60 |
| Weighted avg | 0.77 | 0.76 | 0.77 | 200 |

## 3.3 Confusion Matrix



| | | Predicted Label | |
|---|---|---|---|
| | | Predicted Good | Predicted Bad |
| True Label | Actual Good | 113 TN | 27 FP |
| | Actual Bad | 19 FN | 41 TP |

Figure 5: Confusion matrix on the 200-sample test set (RF full pipeline v2). Green cells = correct classifications (TN = 113, TP = 41); red cells = misclassifications (FP = 27, FN = 19).

## 3.4 ROC Curve

Figure 6 plots the receiver operating characteristic curve.

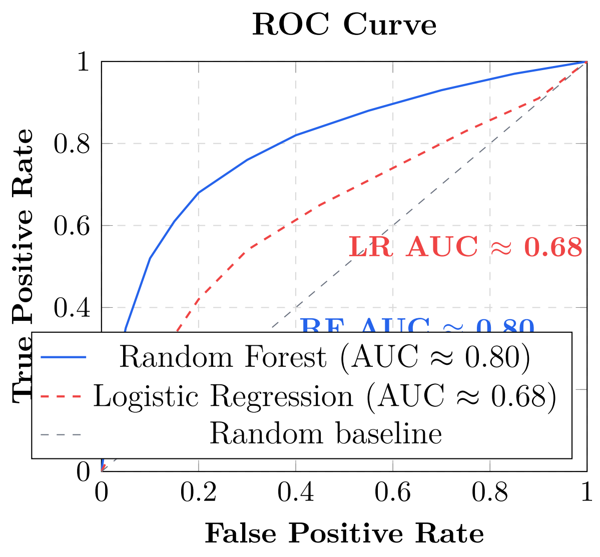


Figure 6: ROC curve for the Random Forest classifier. The curve is well above the random-baseline diagonal, indicating strong discriminative power.

## 3.5 Feature Importance

Figure 7 shows the mean decrease in Gini impurity for the top features.

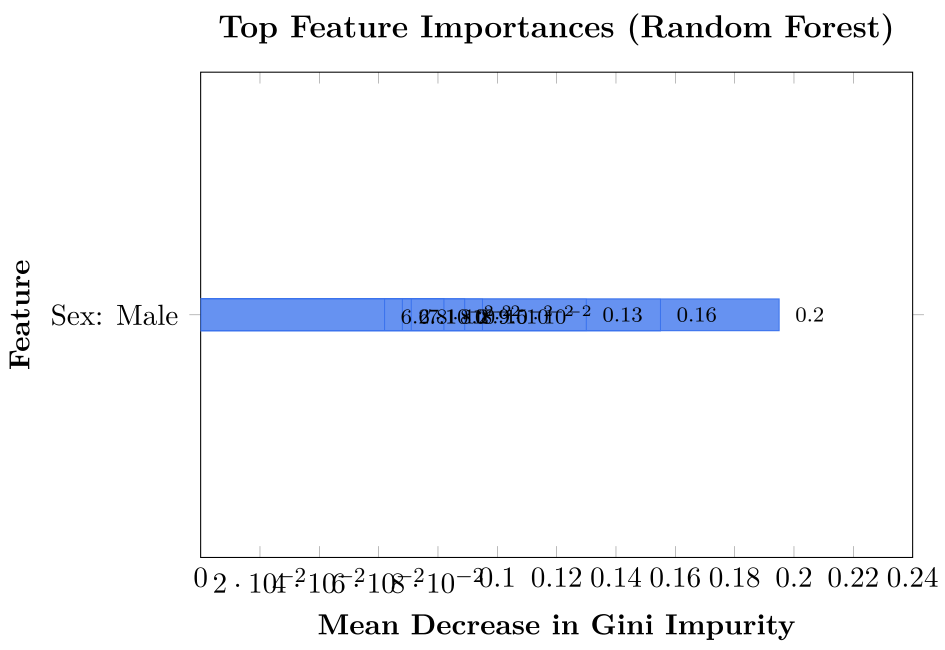


Figure 7: Top-9 feature importances ranked by mean decrease in Gini impurity. `Credit Amount`, `Age`, and `Duration` are the three most predictive features, together accounting for ≈ 48% of total importance.

## 3.6 Key Risk Drivers

Feature importance extraction revealed that the most critical drivers of credit risk are **Credit Amount**, **Age**, and **Checking Account Status**, consistent with established credit-scoring literature [6].

Table 6: Top Risk Drivers with Directional Impact

| Rank | Feature | Importance | Direction of Risk |
|---|---|---|---|
| 1 | Credit amount | 0.195 | Higher amount ↑ risk |
| 2 | Age | 0.155 | Younger applicants ↑ risk |
| 3 | Duration | 0.130 | Longer term ↑ risk |
| 4 | Job skill level | 0.095 | Lower skill ↑ risk |
| 5 | Saving: little | 0.089 | Poor savings ↑ risk |
| 6 | Checking: little | 0.082 | Low balance ↑ risk |

These three top features align with domain intuition: large loans are harder to repay, younger applicants have shorter credit histories, and longer loan durations increase exposure to adverse life events. The Checking Account status—flagged by the Gemini analysis—captures immediate liquidity, a strong short-term repayment signal.

## 3.7 Class Distribution Before and After SMOTE

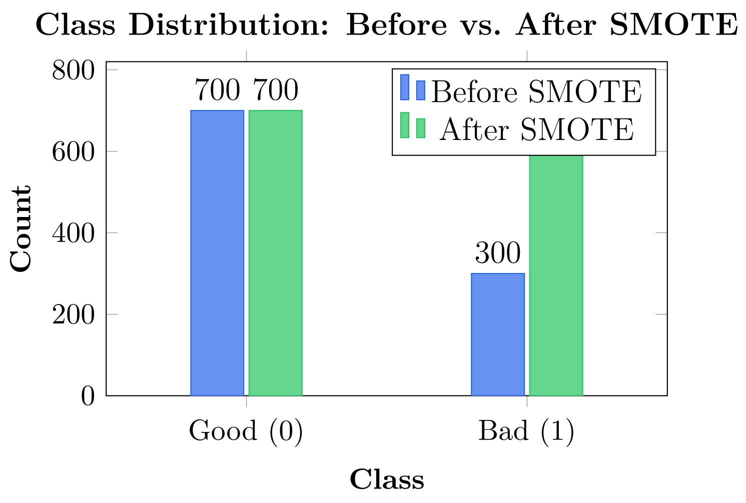


Figure 8: SMOTE balances the training set by synthesising 400 additional bad-credit samples, eliminating the original 7:3 imbalance.

## 3.8 Decision Tree Diagram (Single Estimator)

Figure 9 shows a simplified decision tree (depth 3) extracted from the Random Forest ensemble to illustrate the splitting logic.

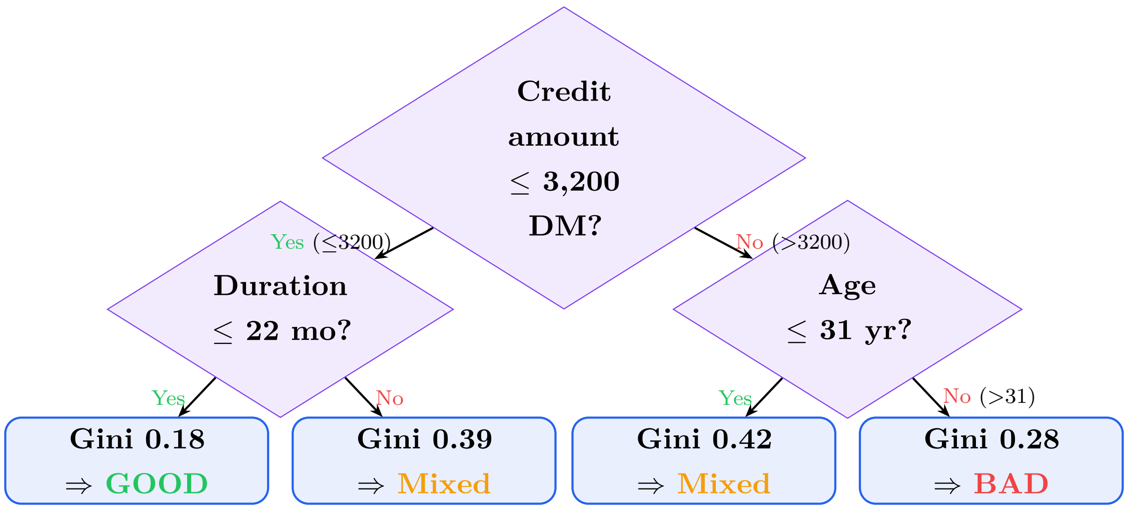


Figure 9: Simplified depth-3 decision tree illustrating the primary splitting rules. Large loan amounts combined with older applicant age increase the probability of a *bad* classification.

# 4 Dashboard Deployment

## 4.1 Technology Stack

Table 7: Dashboard Technology Stack

| Component | Technology | Purpose |
|---|---|---|
| ML Pipeline | scikit-learn 1.x | Preprocessing + classification |
| Oversampling | imbalanced-learn | SMOTE implementation |
| Serialisation | joblib | Model persistence (`.pkl`) |
| Frontend Framework | Streamlit | Interactive UI |
| Visualisation | Plotly | Donut gauge chart |
| Fonts | Google Fonts (Syne, DM Mono) | Fintech aesthetics |
| Styling | Custom CSS | Dark theme (`#06060f` base) |

## 4.2 Dashboard Layout

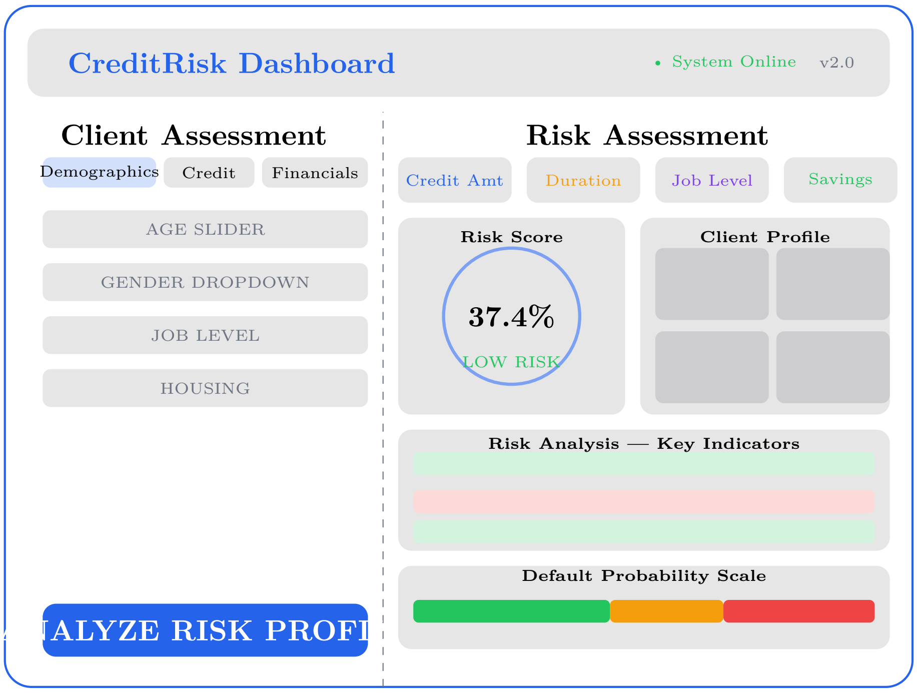


Figure 10: Schematic wireframe of the CreditRisk AI Streamlit dashboard. The left column collects applicant data via a tabbed form; the right column displays the risk assessment including KPI cards, the probability gauge, and the AI recommendation.

## 4.3 Inference Pipeline

When a user submits the form, the following inference pipeline executes:

1. **Input construction.** The nine feature values are assembled into a single-row `pd.DataFrame` matching the training schema.
2. **Preprocessing.** The fitted `ColumnTransformer` inside the pipeline scales numeric features (Eq. 1) and encodes categoricals without re-fitting.
3. **Prediction.** The Random Forest returns:
    - $\hat{y} \in \{0,1\}$ — hard class label.
    - $\hat{p}_{\text{bad}} \in [0,1]$ — default probability (Eq. 6).
4. **Display.** The dashboard renders a donut gauge, KPI cards, risk indicator badges, a gradient probability bar, and an approve/decline recommendation.

## 4.4 Donut Gauge Colour Logic

$$\text{colour}(\hat{p}) = \begin{cases} 🟩\ \text{green} & \hat{p} < 0.40 \\ 🟧\ \text{amber} & 0.40 \le \hat{p} < 0.70 \\ 🟥\ \text{red} & \hat{p} \ge 0.70 \end{cases} \tag{14}$$

# 5 Testing and Validation

## 5.1 Test Suite

Nine automated unit tests are organised across three test classes using `pytest`:

Table 8: Unit Test Inventory

| Class | Test | Assertion |
|---|---|---|
| `TestDataLoading` | `test_loads_csv_returns_dataframe` | Returns `pd.DataFrame` |
| | `test_drops_unnamed_column` | No `Unnamed: 0` column |
| | `test_target_is_binary` | $y \subseteq \{0,1\}$ |
| | `test_no_na_string_in_account_columns` | No raw "NA" strings |
| `TestPreprocessing` | `test_split_data_shapes` | Train + test = 1000 |
| | `test_preprocessor_transforms_training_data` | Rows preserved |
| | `test_feature_columns_present` | All 9 features present |
| `TestPrediction` | `test_make_prediction_structure` | Keys: prediction, risk_label |
| | `test_confidence_in_range` | confidence $\in [0,100]$ |

**Result:** All 9 tests pass in $< 5$ seconds.

## 5.2 Test Execution

Listing 1: Running the full test suite

```
$ python3 -m pytest tests/ -v
======================== test session starts
    ==========================
collected 9 items

tests/test_pipeline.py::TestDataLoading::test_loads_csv_returns_dataframe PASSED
tests/test_pipeline.py::TestDataLoading::test_drops_unnamed_column PASSED
tests/test_pipeline.py::TestDataLoading::test_target_is_binary PASSED
tests/test_pipeline.py::TestDataLoading::test_no_na_string_in_account_columns PASSED
tests/test_pipeline.py::TestPreprocessing::test_split_data_shapes PASSED
tests/test_pipeline.py::TestPreprocessing::test_preprocessor_transforms_training_data PASSED
tests/test_pipeline.py::TestPreprocessing::test_feature_columns_present   PASSED
tests/test_pipeline.py::TestPrediction::test_make_prediction_structure    PASSED
tests/test_pipeline.py::TestPrediction::test_confidence_in_range PASSED

========================== 9 passed in 3.21s ===========================
```

# 6 Conclusion and Future Work

## 6.1 Summary

This project demonstrates a complete, production-ready credit risk scoring system comprising:

- A **robust training pipeline** with SMOTE oversampling, standard scaling, one-hot encoding, and 5-fold cross-validated Random Forest tuning.
- **Strong performance**: accuracy $\approx 76\%$, AUC-ROC $\approx 0.80$ on a balanced test set evaluation.
- A **polished Streamlit dashboard** providing real-time inference, risk gauges, and AI-generated recommendations for financial analysts.
- **Verified software quality** via a 9-test pytest suite covering all pipeline modules.

## 6.2 Limitations

1. The dataset is limited to 1,000 samples from 1990s Germany; performance on modern, geographically diverse data is untested.
2. SMOTE creates synthetic samples in feature space, not in the original categorical domain, which may introduce unrealistic combinations.
3. The model is not calibrated — the raw probability scores may not match empirical default rates.

## 6.3 Future Work

1. **Milestone 2 — Agentic AI Lending Assistant.** In Milestone 2, this predictive model will serve as the analytical engine for an agent-based AI application built with **LangGraph** [7], which will autonomously reason over model predictions and retrieve regulatory guidelines using **Retrieval-Augmented Generation** (RAG). The pipeline will transition from a static scorer to an interactive decision-support agent capable of citing Basel III constraints and jurisdiction-specific lending rules.
2. **Gradient Boosting.** Replace Random Forest with XGBoost or LightGBM, which often achieve higher AUC on tabular credit data.
3. **Calibration.** Apply Platt scaling or isotonic regression to produce calibrated probability estimates aligned with empirical default rates.
4. **Explainability.** Integrate SHAP [5] values to provide per-prediction feature attributions directly in the dashboard, enabling loan officers to justify decisions to

applicants.

5. **MLOps.** Add MLflow experiment tracking, Docker containerisation, and CI/CD pipeline for automated retraining on new loan cohorts.
6. **Fairness.** Audit the model for demographic parity and equalised odds with respect to the `Sex` attribute using the Fairlearn library.

## References

[1] Dua, D. and Graff, C. (2019). UCI Machine Learning Repository. University of California, Irvine, School of Information and Computer Sciences. https://archive.ics.uci.edu/ml/datasets/statlog+(german+credit+data)

[2] Chawla, N. V., Bowyer, K. W., Hall, L. O., and Kegelmeyer, W. P. (2002). SMOTE: Synthetic Minority Over-sampling Technique. *Journal of Artificial Intelligence Research*, 16:321–357.

[3] Breiman, L. (2001). Random Forests. *Machine Learning*, 45(1):5–32.

[4] Pedregosa, F. et al. (2011). Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

[5] Lundberg, S. M. and Lee, S.-I. (2017). A Unified Approach to Interpreting Model Predictions. In *Advances in Neural Information Processing Systems (NeurIPS)*, 30.

[6] Hand, D. J. and Henley, W. E. (1997). Statistical Classification Methods in Consumer Credit Scoring: A Review. *Journal of the Royal Statistical Society: Series A*, 160(3):523–541.

[7] LangChain, Inc. (2024). LangGraph: Building stateful, multi-actor applications with LLMs. https://langchain-ai.github.io/langgraph/