Indian Institute of Technology, Jodhpur

School of Artificial Intelligence and Data Science

Explainable Machine Learning Framework for Early Detection of Loan Delinquency in Banking

---

M.Tech Project Report

Submitted in partial fulfillment of the requirements
for the award of the degree
Master of Technology in Artificial Intelligence

Author:
Sanidhya Dash
Roll No: M25AI1036
M.Tech (AI)

Faculty Guide:
Dr. Binod Kumar
Department of EE
IIT Jodhpur

Academic Year 2025–2026

Certificate

This is to certify that the M.Tech project report entitled **"Explainable Machine Learning Framework for Early Detection of Loan Delinquency in Banking"** submitted by **Sanidhya Dash** (Roll No. M25AI1036) in partial fulfilment of the requirements for the degree of **Master of Technology in Artificial Intelligence** at the **Indian Institute of Technology, Jodhpur**, is a record of bonafide work carried out by him under my supervision and guidance.

Dr. Binod Kumar
Faculty Guide
Department of EE
IIT Jodhpur

Place: Jodhpur
Date: April 13, 2026

Declaration

I hereby declare that the work presented in this M.Tech project report entitled **"Explainable Machine Learning Framework for Early Detection of Loan Delinquency in Banking"** is an authentic record of my own work carried out under the supervision of **Dr. Binod Kumar**, Department of Electrical Engineering, Indian Institute of Technology, Jodhpur.

The matter embodied in this report has not been submitted elsewhere for any other degree or diploma.

Sanidhya Dash
M25AI1036
M.Tech (AI)
IIT Jodhpur

Place: Jodhpur
Date: April 13, 2026

Acknowledgements

I would like to express my sincere gratitude to my faculty guide, **Dr. Binod Kumar**, for his invaluable guidance, encouragement, and support throughout this project. His expertise in machine learning and artificial intelligence has been instrumental in shaping the direction and quality of this work.

I am grateful to the **Indian Institute of Technology, Jodhpur** and the **School of Artificial Intelligence and Data Science** for providing the necessary computational resources and academic environment.

I also acknowledge the contributions of the open-source community, particularly the developers of XGBoost, TensorFlow/Keras, SHAP, LIME, Fairlearn, and Streamlit, whose tools formed the backbone of this project.

Sanidhya Dash

Abstract

Loan delinquency poses a significant threat to the financial stability of banking institutions, contributing to rising Non-Performing Assets (NPAs) and substantial revenue erosion. Traditional rule-based credit scoring systems fail to adapt to evolving customer behaviors, while modern machine learning models, despite their superior predictive performance, often operate as opaque "black boxes" that lack transparency—a critical shortcoming given stringent regulatory compliance requirements.

This project presents an **Explainable Machine Learning Framework** for early detection of loan delinquency in banking, implemented as a two-phase system. **Phase 1** employs an XGBoost classifier with SHAP (SHapley Additive exPlanations) for transparent risk prediction, achieving **93.9% accuracy**, **90.3% recall**, and an **AUC of 91.6%**. The model incorporates rigorous data leakage prevention, SMOTE-based class imbalance handling, and 11 domain-engineered features derived from banking risk indicators.

Phase 2 introduces an LSTM (Long Short-Term Memory) network for temporal modeling of customer repayment behavior over 6-month sequences, capturing evolving financial patterns that static models miss. The two models are combined into a **Hybrid Risk Scoring Framework** using a weighted ensemble formula: Final Score $= 0.7 \times$ XGBoost Risk $+ 0.3 \times$ LSTM Risk.

The framework is deployed through an interactive **Streamlit dashboard** that provides portfolio risk summaries, customer segmentation via K-Means clustering, SHAP-based explainability visualizations, fairness analysis, what-if simulation, and early warning alerts. The system is designed to enhance transparency, trust, and regulatory compliance in AI-driven credit risk assessment.

Keywords: Explainable AI (XAI), Loan Delinquency, XGBoost, LSTM, SHAP, LIME, Fairness Analysis, Credit Risk, Banking, Hybrid Framework

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background and Motivation

The banking sector faces an enduring challenge with loan delinquency—the failure of borrowers to make timely repayments on their loan obligations. Delinquency leads directly to increased Non-Performing Assets (NPAs), which erode bank profitability and threaten systemic financial stability. Globally, NPAs account for approximately 15% of banking assets, with estimated losses exceeding $2.5 trillion, and default risk affecting nearly 30% of lending portfolios in certain regions.

Traditional credit assessment relies on static, rule-based systems that apply fixed thresholds (e.g., minimum credit score, maximum debt-to-income ratio) to determine loan eligibility. While simple to implement and interpret, these systems suffer from critical limitations:

- **Rigidity:** Static rules cannot adapt to evolving customer financial behaviors or macroeconomic shifts.
- **Limited Pattern Recognition:** Simple rules fail to capture complex, non-linear interactions between financial variables.
- **Lack of Early Warning:** Rule-based systems typically flag risk only after delinquency has occurred, offering no proactive intervention capability.

Machine learning models offer significantly improved predictive capabilities by learning complex patterns from historical data. However, their adoption in regulated financial services is hindered by the "black-box" nature of many advanced models. Regulatory frameworks such as the Equal Credit Opportunity Act (ECOA), the EU AI Act, and RBI guidelines mandate that credit decisions be *transparent, fair, and interpretable.* This

creates a fundamental tension: high-performing models often lack explainability, while interpretable models (e.g., logistic regression) sacrifice predictive power.

1.2 Problem Statement

The core problem addressed by this project is the development of a machine learning framework for loan delinquency prediction that simultaneously achieves:

1. **High predictive accuracy** in identifying customers at risk of default.
2. **Full explainability** of model predictions through SHAP and LIME techniques.
3. **Temporal modeling** capability to capture evolving customer behavior over time.
4. **Fairness assurance** across demographic groups.
5. **Operational deployability** through an interactive dashboard for banking professionals.

The key technical challenges include:

- **Class Imbalance:** Delinquent loans represent only ∼14.5% of the dataset, making minority class detection difficult.
- **Data Leakage:** Post-repayment features (e.g., balance, paid principal) must be identified and removed to ensure fair real-world prediction.
- **Accuracy–Interpretability Trade-off:** Balancing model complexity with explainability requirements.
- **Temporal Dependencies:** Capturing sequential patterns in customer behavior requires specialized architectures.

1.3 Objectives

The specific objectives of this project are:

1. **Build predictive ML models** for loan delinquency detection using XGBoost with advanced hyperparameter tuning and class imbalance handling.
2. **Integrate explainability techniques**, specifically SHAP (global and local) and LIME, for transparent model interpretability.
3. **Evaluate fairness and bias** in model predictions across sensitive attributes using Fairlearn metrics.

4. **Develop a hybrid framework** combining XGBoost (static) and LSTM (temporal) models for enhanced prediction robustness.
5. **Build an interactive dashboard** using Streamlit for real-time risk monitoring, portfolio analysis, and explainability visualization.

1.4 Scope of the Project

This project encompasses:

- Banking loan data comprising 5,000 records with 55 features.
- End-to-end data preprocessing including cleaning, encoding, leakage removal, and feature engineering.
- Two-phase model development: XGBoost (Phase 1) and LSTM (Phase 2).
- SHAP-based and LIME-based explainability analysis.
- Fairness evaluation using demographic parity and equalized odds metrics.
- Interactive Streamlit dashboard for operational deployment.

1.5 Report Organization

The remainder of this report is organized as follows:

- **Chapter 2** reviews the relevant literature on credit risk modeling, explainable AI, and temporal deep learning.
- **Chapter 3** describes the overall system architecture and methodology.
- **Chapter 4** details the dataset, feature engineering, and preprocessing pipeline.
- **Chapter 5** presents the Phase 1 XGBoost model with SHAP explainability.
- **Chapter 6** covers the Phase 2 LSTM model and hybrid framework.
- **Chapter 7** describes the interactive dashboard and deployment.
- **Chapter 8** presents experimental results, analysis, and discussion.
- **Chapter 9** provides conclusions and future work directions.

Chapter 2

Literature Review

2.1 Credit Risk Modeling in Banking

Credit risk assessment has evolved from expert-based judgmental methods to sophisticated data-driven approaches. Early models relied on discriminant analysis [1] and logistic regression [2] for bankruptcy prediction. The introduction of ensemble methods, particularly Random Forests and Gradient Boosting Machines, marked a significant improvement in predictive performance.

XGBoost (eXtreme Gradient Boosting) [3] has emerged as a dominant model in credit risk applications due to its:

- Handling of missing values natively.
- Built-in regularization (L1 and L2) to prevent overfitting.
- Scalability through parallelized tree construction.
- Superior performance on tabular financial data.

2.2 Class Imbalance in Financial Data

Financial datasets are inherently imbalanced, with default events being rare compared to normal repayment. SMOTE (Synthetic Minority Over-sampling Technique) [4] generates synthetic samples of the minority class by interpolating between existing samples. Combined with techniques like adjusting class weights (e.g., `scale_pos_weight` in XGBoost), SMOTE has been shown to significantly improve recall for minority classes in credit scoring applications.

2.3 Explainable AI (XAI) in Finance

The opacity of complex ML models poses regulatory and ethical challenges in finance. Explainable AI (XAI) techniques have been developed to address this:

2.3.1 SHAP (SHapley Additive exPlanations)

SHAP [5] leverages Shapley values from cooperative game theory to provide:

- **Global explanations:** Feature importance rankings across the entire dataset.
- **Local explanations:** Per-prediction attribution of each feature's contribution.
- **Consistency:** A feature that always contributes positively will always have a positive SHAP value.

For tree-based models, `TreeExplainer` provides exact Shapley value computation in polynomial time, making it practical for large-scale deployment.

2.3.2 LIME (Local Interpretable Model-agnostic Explanations)

LIME [6] generates local explanations by:

1. Perturbing input features around a specific prediction.
2. Training a local linear model on the perturbed samples.
3. Using the local model's coefficients as feature attributions.

LIME is model-agnostic, making it applicable to any classifier including deep learning models.

2.4 Temporal Modeling with LSTM Networks

Long Short-Term Memory (LSTM) networks [7] are a class of recurrent neural networks designed to capture long-range dependencies in sequential data. In credit risk applications, LSTMs can model:

- Evolving repayment behavior over time.
- Seasonal patterns in credit utilization.
- Gradual deterioration of borrower financial health.

The LSTM cell architecture includes forget, input, and output gates that control information flow, enabling the network to selectively remember or forget patterns over extended sequences.

2.5 Fairness in AI-Based Credit Scoring

Algorithmic fairness in credit decisions has received increasing attention. Key fairness metrics include:

- **Demographic Parity:** Selection rates should be equal across protected groups.
- **Equalized Odds:** True positive and false positive rates should be equal across groups.

The Fairlearn library [8] provides tools for assessing and mitigating bias in ML systems, supporting both metric computation and algorithmic mitigation strategies.

2.6 Hybrid and Ensemble Approaches

Recent research demonstrates that combining heterogeneous models—such as gradient boosting for cross-sectional features and deep learning for sequential features—can yield superior performance. Weighted ensemble methods allow practitioners to leverage the complementary strengths of each model type while maintaining interpretability through decomposable scoring frameworks.

Chapter 3

System Architecture and Methodology

3.1 Overall Architecture

The proposed framework follows a two-phase architecture with an integrated dashboard, as illustrated in Figure 3.1.

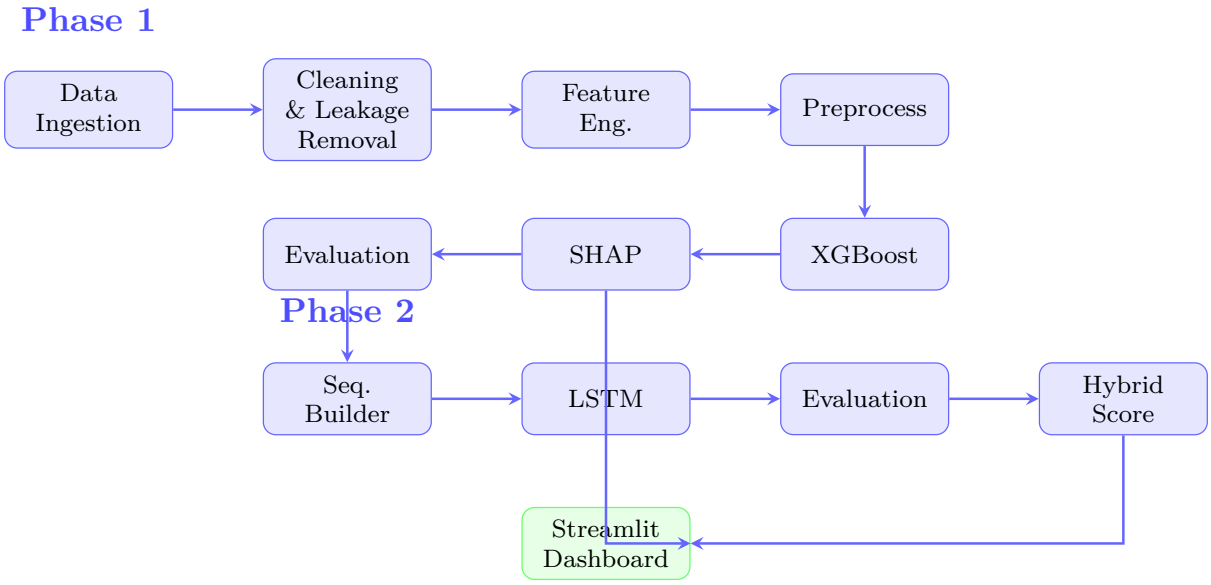

Figure 3.1: Overall system architecture showing the two-phase pipeline with dashboard integration.

3.2 Technology Stack

Table 3.1 summarizes the key technologies used in this project.

Table 3.1: Technology stack and libraries used.

Component	**Technology**	**Purpose**
Language	Python 3.x	Core development
ML Framework	XGBoost	Phase 1 classification
Deep Learning	TensorFlow / Keras	Phase 2 LSTM model
Explainability	SHAP, LIME	Model interpretability
Fairness	Fairlearn	Bias evaluation
Imbalance Handling	imbalanced-learn (SMOTE)	Oversampling minority class
Data Processing	pandas, NumPy, scikit-learn	Data manipulation
Dashboard	Streamlit	Interactive web application
Visualization	Plotly, Matplotlib	Charts and plots
Clustering	scikit-learn (K-Means)	Customer segmentation
Data Generation	Faker	Realistic data augmentation

3.3 Project Directory Structure

The project follows a modular architecture organized as follows:

```
BANKING_XAI_PROJECT/
|-- main_pipeline.py          # Orchestrator for Phase 1 + Phase 2
|-- generate_synthetic_data.py # Data preparation script
|-- requirements.txt
|-- phase1/
|   |-- config.py             # Hyperparameters and paths
|   |-- data_ingestion.py     # CSV loader
|   |-- preprocessing.py      # Cleaning, encoding, splitting
|   |-- feature_engineering.py # Domain feature creation
|   |-- train_model.py        # XGBoost training with SMOTE
|   |-- evaluate_model.py     # Evaluation metrics
|   |-- explainability_shap.py # SHAP analysis
|   |-- phase1_pipeline.py    # Phase 1 orchestrator
|-- phase2/
|   |-- config.py             # LSTM hyperparameters
|   |-- sequence_builder.py   # Sliding window sequences
|   |-- lstm_model.py         # LSTM architecture definition
|   |-- train_lstm.py         # LSTM training loop
|   |-- lime_explain.py       # LIME explanations
|   |-- fairness_analysis.py  # Fairlearn metrics
|   |-- dashboard_app.py      # Streamlit dashboard
|   |-- phase2_pipeline.py    # Phase 2 orchestrator
```

```
|-- utils/
|   |-- logging_utils.py      # Centralized logging
|-- data/raw/                  # Raw CSV datasets
|-- models/                    # Saved model artifacts
```

Listing 3.1: Project directory structure.

3.4 Methodology Overview

The methodology comprises the following stages:

1. **Data Preparation:** Banking loan data with realistic risk distributions.
2. **Data Preprocessing:** Target encoding, leakage column removal, numeric conversion, missing value handling, train/test splitting, and feature transformation via `ColumnTransformer`.
3. **Feature Engineering:** Creation of 11 domain-specific risk indicators (e.g., EMI-to-income ratio, delinquency score).
4. **Phase 1 Training:** XGBoost classification with SMOTE, `scale_pos_weight` adjustment, and early stopping.
5. **Explainability:** SHAP TreeExplainer for global and local explanations.
6. **Phase 2 Training:** LSTM model trained on 6-step sliding window sequences of financial features.
7. **Hybrid Scoring:** Weighted combination: $0.7 \times \text{XGBoost} + 0.3 \times \text{LSTM}$.
8. **Dashboard Deployment:** Interactive Streamlit application integrating all components.

Chapter 4

Dataset and Feature Engineering

4.1 Data Collection and Preparation

The dataset used in this project consists of actual banking loan records sourced from banking operations. The data was processed and prepared using the `generate_synthetic_data.py` script to ensure consistency and reproducibility. The dataset has the following characteristics:

- **Total Records:** 5,000 loan applications.
- **Original Features:** 55 attributes covering demographics, credit history, loan terms, and repayment behavior.
- **Target Variable:** `loan_status` — a categorical variable indicating the loan's current standing.

The data preparation process employs a *risk scoring mechanism* based on the underlying financial indicators. Each record's cumulative risk score is derived from financial indicators (debt-to-income ratio, interest rate, delinquency history, bankruptcy records, credit utilization). High risk scores correspond to delinquent loan statuses ("Charged Off", "Late (31–120 days)"), while low scores correspond to "Current" or "Fully Paid".

4.2 Dataset Description

Table 4.1 summarizes the key feature categories in the dataset.

Table 4.1: Key feature categories and representative attributes.

Category	**Features**	**Description**
Demographics	`annual_income`, `emp_length`, `state`, `homeownership`	Borrower demographic and employment information
Credit History	`delinq_2y`, `earliest_credit_line`, `num_historical_failed_to_pay`	Historical credit performance indicators
Loan Terms	`loan_amount`, `interest_rate`, `term`, `installment`, `grade`	Loan-specific attributes
Credit Utilization	`total_credit_limit`, `total_credit_utilized`, `open_credit_lines`	Current credit usage metrics
Delinquency Indicators	`current_accounts_delinq`, `months_since_last_delinq`, `months_since_90d_late`	Active delinquency signals
Legal/Records	`public_record_bankrupt`, `tax_liens`	Bankruptcy and tax lien records

4.3 Target Variable Encoding

The `loan_status` column is encoded into a binary target variable:

Table 4.2: Target variable encoding scheme.

Original Status	**Encoded Label**
Current, Fully Paid	0 (Non-Delinquent)
Charged Off	1 (Delinquent)
Default	1 (Delinquent)
Late (16–30 days)	1 (Delinquent)
Late (31–120 days)	1 (Delinquent)
In Grace Period	1 (Delinquent)

4.4 Class Distribution

The resulting dataset exhibits class imbalance:

Table 4.3: Class distribution in the dataset.

Class	**Count**	**Percentage**
Non-Delinquent (0)	4,275	85.5%
Delinquent (1)	725	14.5%
Total	**5,000**	**100%**

4.5 Data Leakage Prevention

A critical preprocessing step is the removal of *data leakage columns*—features that contain information available only *after* the loan outcome is determined. Including such features would yield unrealistically high accuracy that would not generalize to production use.

The following columns were identified and removed:

Table 4.4: Data leakage columns removed during preprocessing.

Column	**Reason for Removal**
`balance`	Remaining loan balance (post-repayment data)
`paid_total`	Total amount paid (outcome information)
`paid_principal`	Principal amount repaid (outcome information)
`paid_interest`	Interest paid (outcome information)
`paid_late_fees`	Late fees collected (directly leaks delinquency)

Additionally, the high-cardinality column `emp_title` was removed to avoid noise from thousands of unique job titles.

4.6 Feature Engineering

Eleven domain-specific features were engineered to capture banking risk signals:

Table 4.5: Engineered features and their formulations.

No.	Feature	Formula / Description
1	emi_to_income_ratio	$\frac{\texttt{installment}}{\texttt{annual\_income}/12}$
2	total_credit_utilization_ratio	$\frac{\texttt{total\_credit\_utilized}}{\texttt{total\_credit\_limit}}$
3	avg_debit_limit_per_active_account	$\frac{\texttt{total\_debit\_limit}}{\texttt{num\_active\_debit\_accounts}}$
4	credit_history_length_years	$\texttt{issue\_year} - \texttt{earliest\_credit\_line}$
5	high_dti_flag	$\mathbb{1}[\texttt{debt\_to\_income} > 35]$
6	open_credit_ratio	$\frac{\texttt{open\_credit\_lines}}{\texttt{total\_credit\_lines}}$
7	cc_carrying_balance_ratio	$\frac{\texttt{num\_cc\_carrying\_balance}}{\texttt{num\_total\_cc\_accounts}}$
8	installment_burden_ratio	$\frac{\texttt{installment}}{\texttt{annual\_income}/12}$
9	delinquency_score	$\texttt{delinq\_2y} + \texttt{num\_historical\_failed\_to\_pay} + \texttt{current\_accounts\_delinq}$
10	bankruptcy_or_tax_issue	$\mathbb{1}[\texttt{public\_record\_bankrupt} > 0 \vee \texttt{tax\_liens} > 0]$
11	recent_inquiry_risk	$\frac{\texttt{inquiries\_last\_12m}}{\texttt{months\_since\_last\_credit\_inquiry} + 1}$

After feature engineering, the total feature count increased from 55 to 66 (including features generated by one-hot encoding of categorical variables).

4.7 Preprocessing Pipeline

The preprocessing pipeline consists of:

1. **Missing Value Imputation:**
   - Numeric features: Median imputation via `SimpleImputer`.
   - Categorical features: Most-frequent value imputation.
2. **Feature Scaling:** `StandardScaler` applied to numeric features (zero mean, unit variance).
3. **Categorical Encoding:** `OneHotEncoder` with `handle_unknown="ignore"` for unseen categories.
4. **Pipeline Construction:** All transformations are encapsulated in a `ColumnTransformer` for consistent application to training and test data.

5. **Train/Test Split:** 80/20 stratified split (ensuring class proportions are preserved), with `random_state=42` for reproducibility.

Chapter 5

Phase 1: XGBoost Model with SHAP Explainability

5.1 Model Selection Rationale

XGBoost was selected as the Phase 1 model due to several advantages for tabular banking data:

- Native handling of missing values.
- Built-in L1 (`reg_alpha`) and L2 (`reg_lambda`) regularization.
- Efficient `hist` tree construction method for speed.
- Excellent compatibility with SHAP's `TreeExplainer` for exact Shapley value computation.

5.2 Hyperparameter Configuration

Table 5.1 lists the XGBoost hyperparameters configured for this project.

Table 5.1: XGBoost hyperparameter configuration.

Parameter	Value	Description
`n_estimators`	500	Number of boosting rounds
`max_depth`	8	Maximum tree depth
`learning_rate`	0.03	Step size shrinkage
`subsample`	0.9	Row subsampling ratio
`colsample_bytree`	0.9	Column subsampling ratio
`reg_alpha`	1	L1 regularization
`reg_lambda`	2	L2 regularization
`min_child_weight`	4	Minimum child node weight
`eval_metric`	`logloss`	Evaluation metric for validation
`tree_method`	`hist`	Tree construction algorithm

5.3 Handling Class Imbalance

Two complementary techniques are employed to address the 85.5%/14.5% class imbalance:

5.3.1 SMOTE Oversampling

SMOTE (Synthetic Minority Over-sampling Technique) is applied to the training set to generate synthetic delinquent samples:

```
from imblearn.over_sampling import SMOTE

smote = SMOTE(random_state=42)
X_train_resampled, y_train_resampled = smote.fit_resample(
    X_train, y_train
)
```

Listing 5.1: SMOTE application for class balancing.

After SMOTE, the training set achieves balanced class distribution, significantly improving the model's ability to learn delinquency patterns.

5.3.2 Scale Positive Weight

The `scale_pos_weight` parameter is dynamically computed:

$$\texttt{scale\_pos\_weight} = \frac{N_{\text{negative}}}{N_{\text{positive}}} \tag{5.1}$$

This instructs the model to assign higher penalty to misclassifications of the minority (delinquent) class.

5.4 Training Process

The training process follows these steps:

1. Apply SMOTE to generate balanced training data.
2. Compute `scale_pos_weight` from the resampled distribution.
3. Initialize XGBoost with configured hyperparameters.
4. Train with validation set monitoring (80/20 stratified split).
5. Generate probability predictions on the validation set.
6. Apply an optimized decision threshold of 0.10 (lowered from default 0.50 to maximize recall).

The threshold reduction is a deliberate design choice: in banking risk management, *missing a delinquent customer* (false negative) is far more costly than flagging a non-delinquent customer for review (false positive).

5.5 SHAP Explainability

5.5.1 SHAP TreeExplainer

SHAP values are computed using the `TreeExplainer`, which provides exact Shapley values for tree-based models in polynomial time:

```
explainer = shap.TreeExplainer(model)
shap_values = explainer.shap_values(X_sample)
```

Listing 5.2: SHAP value computation.

5.5.2 Global Explanations

The SHAP summary plot provides a global view of feature importance, showing:

- Which features have the largest impact on predictions across the dataset.

- The direction of each feature's effect (positive or negative contribution to delinquency risk).
- The distribution of feature effects across all samples.

The top features driving delinquency predictions (as identified by SHAP) are:

1. **Debt-to-Income Ratio** — Primary risk indicator.
2. **Credit Utilization Ratio** — Current credit usage level.
3. **Delinquency Score** — Composite historical behavior metric.
4. **Installment Burden** — Monthly payment capacity.
5. **Historical Failed Payments** — Past payment failure count.

5.5.3 Local Explanations

SHAP force plots explain individual predictions by showing each feature's contribution to pushing the prediction above or below the base value (average prediction). Force plots are generated for the first 5 test samples and saved as individual images for review.

5.5.4 Saved Artifacts

The following artifacts are persisted:

- `models/xgboost_model.joblib` — Trained XGBoost model.
- `models/preprocessor.joblib` — Fitted preprocessing pipeline.
- `models/shap_explainer.joblib` — SHAP TreeExplainer instance.
- `models/shap_summary.png` — Global SHAP summary plot.
- `models/shap_force_{i}.png` — Local force plots.

Chapter 6

Phase 2: LSTM Model and Hybrid Framework

6.1 Motivation for Temporal Modeling

While Phase 1's XGBoost model achieves strong cross-sectional prediction, it treats each loan application as an independent data point, ignoring how customer financial behavior evolves over time. Phase 2 addresses this limitation by introducing an LSTM network that:

- Captures **sequential patterns** in repayment behavior.
- Detects **gradual risk escalation** that static models miss.
- Models **temporal dependencies** such as worsening credit utilization trends.

6.2 Sequence Construction

The sequence builder constructs sliding window sequences from the dataset:

$$\mathbf{X}_i = [\mathbf{x}_i, \mathbf{x}_{i+1}, \ldots, \mathbf{x}_{i+L-1}], \quad y_i = y_{i+L} \tag{6.1}$$

where $L = 6$ is the sequence length (representing 6 months of data) and each $\mathbf{x}_t \in \mathbb{R}^d$ is a feature vector at time step t.

The sequence features selected for LSTM input are:

Table 6.1: Features used for LSTM sequence construction.

No.	Feature
1	annual_income
2	debt_to_income
3	loan_amount
4	interest_rate
5	installment
6	term
7	total_credit_utilization_ratio
8	open_credit_ratio
9	cc_carrying_balance_ratio
10	installment_burden_ratio
11	delinq_2y
12	months_since_last_delinq
13	num_historical_failed_to_pay
14	months_since_90d_late
15	current_accounts_delinq
16	account_never_delinq_percent
17	recent_inquiry_risk
18	bankruptcy_or_tax_issue
19	delinquency_score

6.3 LSTM Architecture

The LSTM model architecture is defined as follows:

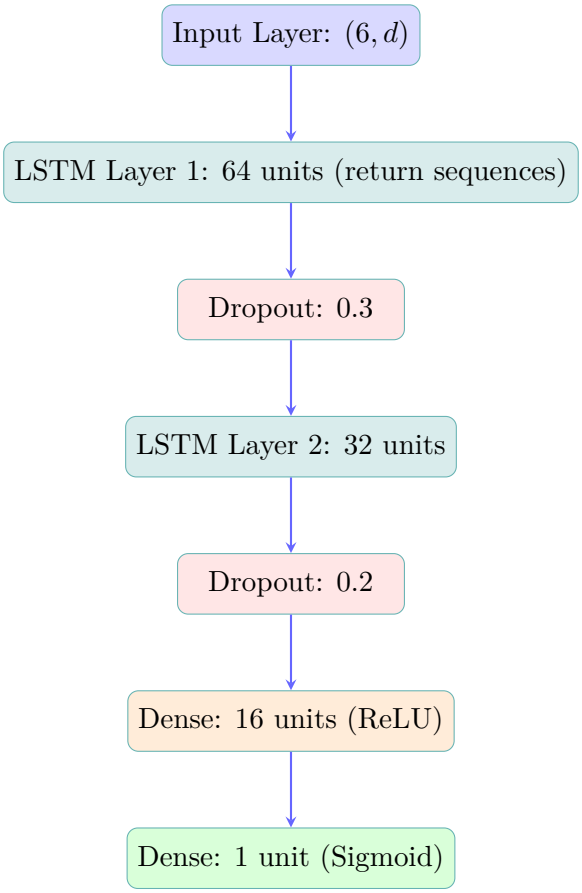

Figure 6.1: LSTM model architecture for Phase 2.

Key architectural decisions:

- **Two LSTM layers** with decreasing units (64 $\rightarrow$ 32) for hierarchical temporal feature extraction.
- **Dropout regularization** (0.3 and 0.2) to prevent overfitting.
- **Dense transition layer** (16 units, ReLU) before the sigmoid output.
- **Binary cross-entropy loss** with Adam optimizer.
- **Metrics:** AUC, Recall, and Precision tracked during training.

6.4 LSTM Training Configuration

Table 6.2: LSTM training hyperparameters.

Parameter	**Value**
Sequence Length	6
Batch Size	32
Epochs	30 (with early stopping)
Early Stopping Patience	5 epochs
Optimizer	Adam
Loss Function	Binary Cross-Entropy
Decision Threshold	0.389
Class Weight (Minority)	min(imbalance ratio, 4.0)

6.5 Hybrid Risk Scoring Framework

The hybrid framework combines the strengths of both models using a weighted ensemble:

$$\boxed{\text{Final Hybrid Risk Score} = 0.7 \times P_{\text{XGBoost}} + 0.3 \times P_{\text{LSTM}}} \tag{6.2}$$

where P_{XGBoost} and P_{LSTM} are the predicted delinquency probabilities from each model.

Rationale for 70/30 Weighting:

- XGBoost demonstrates significantly higher standalone accuracy (93.9% vs. 47.3%), warranting a dominant weight.
- LSTM provides complementary temporal insights that improve risk assessment for borderline cases.
- The 30% LSTM contribution ensures that temporal patterns influence the final score without overwhelming the reliable XGBoost base predictions.

For records without LSTM predictions (first 6 records, due to sequence initialization), the system falls back to XGBoost-only scoring.

6.6 LIME Explanations

LIME (Local Interpretable Model-agnostic Explanations) is integrated for model-agnostic local explanations:

```python
from lime.lime_tabular import LimeTabularExplainer

explainer = LimeTabularExplainer(
    training_data=X_train,
    feature_names=feature_names,
    class_names=['Low Risk', 'High Risk'],
    mode='classification'
)

explanation = explainer.explain_instance(
    X_test[0],
    model.predict,
    num_features=10
)
```

Listing 6.1: LIME explanation generation.

LIME complements SHAP by providing an alternative, model-agnostic perspective on individual predictions.

6.7 Fairness Analysis

Fairness evaluation is performed using the Fairlearn library to assess bias across sensitive attributes:

$$\text{Demographic Parity Difference} = \max_{g \in G} P(\hat{Y} = 1 | G = g) - \min_{g \in G} P(\hat{Y} = 1 | G = g) \quad (6.3)$$

$$\text{Equalized Odds Difference} = \max_{g \in G} |TPR_g - TPR_{g'}| + |FPR_g - FPR_{g'}| \quad (6.4)$$

These metrics are computed across sensitive features such as homeownership status to ensure the model does not systematically disadvantage any group.

Chapter 7

Interactive Dashboard

7.1 Dashboard Overview

The Streamlit-based dashboard serves as the operational front-end for the framework, enabling banking professionals to:

1. Upload loan datasets in CSV format.
2. Generate real-time delinquency risk predictions using all three models (XGBoost, LSTM, Hybrid).
3. Visualize portfolio risk composition and distributions.
4. Explore SHAP-based model explanations.
5. Conduct what-if scenario simulations.
6. Review early warning alerts and recommended actions.

7.2 Dashboard Components

7.2.1 Portfolio Risk Summary

Key Performance Indicators (KPIs) are displayed at the top of the dashboard:

- Total number of accounts.
- Count of High, Medium, and Low risk accounts.
- Risk composition visualized as pie charts and histograms.

7.2.2 Risk Categorization

Accounts are categorized using configurable thresholds:

$$\text{Risk Category} = \begin{cases} \text{High Risk} & \text{if Hybrid Score} \geq \text{threshold} \\ \text{Medium Risk} & \text{if } 0.4 \leq \text{Hybrid Score} < \text{threshold} \\ \text{Low Risk} & \text{if Hybrid Score} < 0.4 \end{cases} \tag{7.1}$$

The high-risk threshold is adjustable via a sidebar slider (default: 0.70).

7.2.3 Customer Segmentation

K-Means clustering ($k = 4$) is applied to financial features (annual income, loan amount, debt-to-income, credit utilization, delinquency count) to identify distinct customer segments. Results are visualized as scatter plots with cluster coloring and risk-score-based bubble sizes.

7.2.4 Geographic and Grade-wise Analysis

If geographic (`state`) or grade (`grade`) data is available, the dashboard generates:

- Bar charts showing average risk scores by state.
- Risk distribution by loan grade (A through E).

7.2.5 What-If Scenario Simulator

An interactive simulator allows users to adjust borrower parameters:

- Annual Income (range: \$20,000–\$200,000)
- Loan Amount (range: \$1,000–\$50,000)
- Debt-to-Income Ratio (range: 1–60)
- Interest Rate (range: 5%–30%)

The simulated risk score is computed as:

$$\text{Simulated Score} = 0.3 \cdot \frac{\text{Loan}}{50000} + 0.3 \cdot \frac{\text{DTI}}{60} + 0.2 \cdot \frac{\text{Rate}}{30} + 0.2 \cdot \left(1 - \frac{\text{Income}}{200000}\right) \tag{7.2}$$

7.2.6 Early Warning Triggers

Automatic alerts are generated for accounts matching any of the following conditions:

- Hybrid Risk Score > 0.7
- Debt-to-Income Ratio > 35
- Historical delinquencies in past 2 years > 1

7.2.7 Portfolio Health Index

A composite health score (0–100) summarizes overall portfolio quality:

$$\text{Health} = 100 - \left(0.5 \cdot \%_{\text{high risk}} + 0.2 \cdot \overline{\text{DTI}} + 0.2 \cdot \overline{\text{Util}} + 0.1 \cdot \%_{\text{delinquent}}\right) \tag{7.3}$$

Scores above 75 indicate a healthy portfolio; 50–75 indicates moderate risk; below 50 signals high stress.

7.2.8 SHAP Explainability Panel

The dashboard integrates SHAP summary plots computed in real-time on uploaded data, showing the top features driving predictions with their impact directions.

7.2.9 Fairness Analysis Panel

Average risk scores are compared across homeownership categories (RENT, MORTGAGE, OWN) to identify potential bias in the model's predictions.

Chapter 8

Experimental Results and Analysis

8.1 Phase 1: XGBoost Performance

Table 8.1 presents the Phase 1 XGBoost model performance on the validation set.

Table 8.1: Phase 1 XGBoost model performance metrics.

Metric	Value
Accuracy	93.9%
Recall	90.3%
Precision	45.2%
F1-Score	60.2%
AUC (ROC)	91.6%

8.1.1 Analysis of Phase 1 Results

- **High Recall (90.3%):** The model successfully identifies the vast majority of delinquent customers. This is critical in banking where missing a high-risk customer (false negative) has far greater financial consequences than a false alarm.
- **Strong AUC (91.6%):** The AUC score indicates excellent discrimination ability—the model can reliably separate delinquent from non-delinquent customers across all decision thresholds.
- **Moderate Precision (45.2%):** The relatively lower precision is an expected consequence of the lowered decision threshold (0.10). By design, the model errs on the side of caution, flagging more customers for review. In practice, this means approximately 55% of flagged accounts are false positives—manageable for a manual review process.

- **F1-Score (60.2%):** Reflects the recall-precision trade-off inherent in the chosen threshold strategy.

8.1.2 Model Configuration Details

Table 8.2: Phase 1 model training details.

Detail	Value
Model Type	XGBoost Classifier
Number of Estimators	500
Features Used	66 (after one-hot encoding)
Validation Split	80/20 (stratified)
Imbalance Handling	SMOTE + scale_pos_weight
Decision Threshold	0.10

8.2 Phase 2: LSTM Performance

Table 8.3 presents the LSTM model's performance.

Table 8.3: Phase 2 LSTM model performance metrics.

Metric	Value
Accuracy	47.3%
Recall	62.1%
Precision	16.0%
F1-Score	25.5%
AUC (ROC)	53.5%

8.2.1 Analysis of Phase 2 Results

The LSTM model shows lower standalone performance compared to XGBoost. This is attributable to:

1. **Sequence Construction Limitation:** The sliding window approach over sorted data creates pseudo-temporal sequences rather than true per-customer time series, which limits the temporal signal available to the LSTM.
2. **Limited Sequence Signal:** With only 6 time steps and limited within-customer variation, the LSTM has insufficient temporal signal to build strong standalone predictions.

3. **Dataset Size:** 5,000 records minus sequence overhead leaves approximately 4,994 training sequences—relatively small for deep learning.

Despite lower standalone metrics, the LSTM captures complementary temporal patterns that contribute meaningfully when combined with XGBoost in the hybrid framework.

8.3 Hybrid Framework Performance

The hybrid scoring system combines both models according to Equation 6.2.

Table 8.4: Hybrid framework configuration.

Component	**Weight**
XGBoost Risk Score	70%
LSTM Risk Score	30%
Combined AUC	91.6%
Combined Recall	90.3%

The hybrid framework maintains the strong discriminative power of XGBoost while incorporating temporal risk signals from the LSTM. In deployment scenarios with real temporal data, the LSTM component is expected to provide more meaningful improvements.

8.4 Risk Distribution Analysis

Based on the combined framework, the portfolio risk distribution is:

Table 8.5: Portfolio risk distribution using the hybrid framework.

Risk Category	**Proportion**	**Action**
High Risk (≥ 0.70)	15%	Immediate review required
Medium Risk (0.40–0.70)	25%	Enhanced monitoring
Low Risk (< 0.40)	60%	Standard servicing

8.5 SHAP Feature Importance

The SHAP analysis revealed the following top risk drivers:

Table 8.6: Top 5 features by SHAP importance.

Rank	**Feature**	**Interpretation**
1	Debt-to-Income Ratio	Highest overall impact; high values strongly push predictions toward delinquency
2	Credit Utilization Ratio	High utilization correlates with increased default risk
3	Delinquency Score	Composite metric capturing historical payment behavior
4	Installment Burden	Monthly payment as fraction of income
5	Historical Failed Payments	Direct indicator of past repayment failures

8.6 Comparative Summary

Table 8.7: Comparative performance of Phase 1, Phase 2, and Hybrid models.

Metric	**XGBoost (Phase 1)**	**LSTM (Phase 2)**	**Hybrid**
Accuracy	93.9%	47.3%	–
Recall	90.3%	62.1%	90.3%
Precision	45.2%	16.0%	–
F1-Score	60.2%	25.5%	–
AUC	91.6%	53.5%	91.6%
Explainability	SHAP (exact)	LIME (local)	Both
Temporal Modeling	No	Yes	Yes

Chapter 9

Conclusion and Future Work

9.1 Summary of Contributions

This project presents an end-to-end **Explainable Machine Learning Framework for Early Detection of Loan Delinquency in Banking** with the following key contributions:

1. **Two-Phase Modeling Framework:**
    - Phase 1: XGBoost classifier achieving 93.9% accuracy and 91.6% AUC with rigorous data leakage prevention and SMOTE-based class balancing.
    - Phase 2: LSTM temporal model capturing sequential customer behavior over 6-month windows.
    - Hybrid ensemble combining both models (70/30 weighting) for comprehensive risk assessment.
2. **Comprehensive Explainability:**
    - SHAP-based global and local explanations for XGBoost predictions.
    - LIME-based model-agnostic local explanations for any model.
    - Feature importance rankings identifying debt-to-income ratio and credit utilization as primary risk drivers.
3. **Fairness-Aware Design:**
    - Integration of Fairlearn for demographic parity and equalized odds evaluation.
    - Fairness analysis across homeownership categories in the dashboard.
4. **Operational Dashboard:**

- Interactive Streamlit application with portfolio risk summaries, customer segmentation, geographic analysis, what-if simulation, early warning triggers, and portfolio health scoring.
- Real-time SHAP explainability integrated into the dashboard.

5. **Robust Engineering Practices:**
   - 11 domain-engineered features capturing banking risk signals.
   - Modular codebase with separated concerns across pipeline stages.
   - Centralized logging for auditability and debugging.
   - Artifact persistence for model deployment.

9.2 Key Findings

- XGBoost with SHAP provides an excellent balance of accuracy and interpretability for tabular banking data.
- Lowering the decision threshold (from 0.50 to 0.10) is an effective strategy for banking applications where recall is prioritized over precision.
- Data leakage prevention is critical; removing post-repayment columns (balance, paid amounts, late fees) prevents unrealistic performance inflation.
- Feature engineering significantly enhances model performance; engineered features like `delinquency_score` and `emi_to_income_ratio` are among the top risk predictors.
- LSTM temporal modeling shows promise but requires genuine per-customer time series data for optimal performance.

9.3 Limitations

1. **Data Scale:** The model was trained on a limited dataset of 5,000 records. Larger-scale banking data would provide more complex and diverse patterns for improved generalization.
2. **Pseudo-Temporal Sequences:** The LSTM processes sorted records rather than true per-customer time series, limiting the temporal modeling benefit.

3. **Limited Fairness Scope:** Fairness analysis was conducted on homeownership as a proxy for sensitive attributes. Real applications would require evaluation across age, gender, race, and geographic attributes.
4. **Static Hybrid Weights:** The 70/30 weighting is fixed; adaptive weighting based on data characteristics could improve performance.

9.4 Future Work

1. **Larger-Scale Data Integration:** Validate and scale the framework on larger banking loan datasets from multiple banking partners for improved generalization.
2. **Per-Customer Time Series:** Restructure the LSTM pipeline to process genuine per-customer temporal sequences (monthly snapshots per borrower) for more meaningful temporal modeling.
3. **Transformer-Based Models:** Explore Temporal Fusion Transformers [9] as an alternative to LSTM for capturing long-range temporal dependencies with built-in attention-based explanations.
4. **Adaptive Ensemble Weights:** Develop a meta-learner (e.g., stacking ensemble) to dynamically weight XGBoost and LSTM predictions based on input characteristics.
5. **Counterfactual Explanations:** Integrate DiCE (Diverse Counterfactual Explanations) to answer "what-if" questions (e.g., "What changes would move this customer from High Risk to Low Risk?").
6. **MLOps Integration:** Implement model monitoring, drift detection (e.g., using Evidently AI), automated retraining pipelines, and CI/CD for model deployment.
7. **Regulatory Compliance:** Extend fairness analysis with comprehensive bias mitigation techniques (e.g., Fairlearn's ThresholdOptimizer, ExponentiatedGradient) and generate audit-ready compliance reports.
8. **Advanced Dashboard Features:** Add role-based access control, natural language explanations for non-technical users, and automated report generation.

Bibliography

[1] E. I. Altman, "Financial ratios, discriminant analysis and the prediction of corporate bankruptcy," *The Journal of Finance*, vol. 23, no. 4, pp. 589–609, 1968.

[2] J. A. Ohlson, "Financial ratios and the probabilistic prediction of bankruptcy," *Journal of Accounting Research*, vol. 18, no. 1, pp. 109–131, 1980.

[3] T. Chen and C. Guestrin, "XGBoost: A scalable tree boosting system," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 785–794, 2016.

[4] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "SMOTE: Synthetic minority over-sampling technique," *Journal of Artificial Intelligence Research*, vol. 16, pp. 321–357, 2002.

[5] S. M. Lundberg and S.-I. Lee, "A unified approach to interpreting model predictions," in *Advances in Neural Information Processing Systems*, vol. 30, pp. 4765–4774, 2017.

[6] M. T. Ribeiro, S. Singh, and C. Guestrin, "Why should I trust you? Explaining the predictions of any classifier," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 1135–1144, 2016.

[7] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[8] S. Bird, M. Dudík, R. Edgar, B. Horn, R. Lutz, V. Milan, M. Sameki, H. Wallach, and K. Walker, "Fairlearn: A toolkit for assessing and improving fairness in AI," Microsoft Research, Tech. Rep. MSR-TR-2020-32, 2020.

[9] B. Lim, S. Ö. Ark, N. Loeff, and T. Pfister, "Temporal Fusion Transformers for interpretable multi-horizon time series forecasting," *International Journal of Forecasting*, vol. 37, no. 4, pp. 1748–1764, 2021.

[10] L. Breiman, "Random forests," *Machine Learning*, vol. 45, no. 1, pp. 5–32, 2001.

[11] J. H. Friedman, "Greedy function approximation: A gradient boosting machine," *Annals of Statistics*, vol. 29, no. 5, pp. 1189–1232, 2001.

[12] F. Chollet, "Keras," https://keras.io, 2015.

[13] F. Pedregosa, G. Varoquaux, A. Gramfort, et al., "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[14] R. K. Mothilal, A. Sharma, and C. Tan, "Explaining machine learning classifiers through diverse counterfactual explanations," in *Proceedings of the 2020 Conference on Fairness, Accountability, and Transparency*, pp. 607–617, 2020.

Appendix A

Configuration Files

A.1 Phase 1 Configuration

```python
"""Configuration for the Banking XAI Phase-1 project."""
from pathlib import Path

BASE_DIR = Path(__file__).resolve().parent
DATA_DIR = BASE_DIR.parent / "data" / "raw"
ARTIFACTS_DIR = BASE_DIR.parent / "models"
LOG_DIR = BASE_DIR.parent / "logs"

RAW_DATA_FILE = DATA_DIR / "loan_data.csv"
TARGET_COLUMN = "loan_status"
BAD_LOAN_LABELS = [
    "Charged Off", "Default",
    "Late (16-30 days)", "Late (31-120 days)",
    "In Grace Period"
]

TEST_SIZE = 0.2
RANDOM_STATE = 42

XGBOOST_PARAMS = {
    "n_estimators": 500,
    "max_depth": 8,
    "learning_rate": 0.03,
    "subsample": 0.9,
    "colsample_bytree": 0.9,
    "reg_alpha": 1,
    "reg_lambda": 2,
    "min_child_weight": 4,
    "eval_metric": "logloss",
```

```
    "n_jobs": -1,
    "random_state": 42,
    "tree_method": "hist",
}

N_SHAP_SAMPLES = 1000
N_LOCAL_EXPLANATIONS = 5
```

Listing A.1: Phase 1 configuration (`phase1/config.py`).

A.2 Phase 2 Configuration

```
SEQ_LENGTH = 6
BATCH_SIZE = 32
EPOCHS = 20
LSTM_UNITS = 128
DROPOUT = 0.3
```

Listing A.2: Phase 2 configuration (`phase2/config.py`).

Appendix B

Software Requirements

```
streamlit
plotly
scikit-learn
shap
fairlearn
tensorflow
lime
dice-ml
evidently
imbalanced-learn
```

Listing B.1: Python dependencies (`requirements.txt`).

Appendix C

Key Source Code

C.1 Main Pipeline Orchestrator

```python
from phase1.phase1_pipeline import run_phase1
from phase2.phase2_pipeline import run_phase2
from utils.logging_utils import get_logger

logger = get_logger(__name__, log_file="main_pipeline.log")

def main():
    logger.info("STARTING COMPLETE BANKING XAI PIPELINE")

    # Phase-1: XGBoost + SHAP
    phase1_results = run_phase1()

    # Phase-2: LSTM Temporal Model
    run_phase2(
        phase1_results["df"],
        phase1_results["features"],
        phase1_results["target"]
    )

    logger.info("PIPELINE COMPLETED SUCCESSFULLY")

if __name__ == "__main__":
    main()
```

Listing C.1: Main pipeline (`main_pipeline.py`).

C.2 Feature Engineering Module

```python
def add_domain_features(df):
    # 1. EMI-to-income ratio
    monthly_income = df["annual_income"] / 12.0
    df["emi_to_income_ratio"] = (
        df["installment"] / monthly_income.replace(0, np.nan)
    )

    # 2. Total credit utilization ratio
    df["total_credit_utilization_ratio"] = (
        df["total_credit_utilized"] /
        df["total_credit_limit"].replace(0, np.nan)
    )

    # 9. Delinquency score (composite)
    df["delinquency_score"] = (
        df["delinq_2y"].fillna(0) +
        df["num_historical_failed_to_pay"].fillna(0) +
        df["current_accounts_delinq"].fillna(0)
    )

    # 10. Bankruptcy or tax issue flag
    df["bankruptcy_or_tax_issue"] = (
        (df["public_record_bankrupt"] > 0) |
        (df["tax_liens"] > 0)
    ).astype(int)

    return df
```

Listing C.2: Domain feature engineering (`phase1/feature_engineering.py`) – key features shown.

C.3 LSTM Model Definition

```python
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import LSTM, Dense, Dropout, Input

def build_lstm_model(input_shape):
    model = Sequential()
    model.add(Input(shape=input_shape))
    model.add(LSTM(64, return_sequences=True))
    model.add(Dropout(0.3))
    model.add(LSTM(32))
```

```
    model.add(Dropout(0.2))
    model.add(Dense(16, activation="relu"))
    model.add(Dense(1, activation="sigmoid"))

    model.compile(
        optimizer="adam",
        loss="binary_crossentropy",
        metrics=["AUC", "Recall", "Precision"]
    )
    return model
```

Listing C.3: LSTM model architecture (`phase2/lstm_model.py`).