Technical Narrative for FarrmWise AI Lender

Authors: Cedrouseroll Omondi, Lawrence Mungai, Hillary Munyole, Judith Makokha, Ian Ocholla

1. Introduction

This document outlines the architecture, methods, assumptions, and scalability considerations of the loan eligibility prediction system developed during the AI for Startups Hackathon. The system integrates a machine learning model with a Nuxt 4 frontend and a Flask backend to evaluate and explain farmer loan applications. It is designed for scalability, interpretability, and real-time response.

2. Problem Statement

Rural farmers often lack access to structured credit scoring systems. Traditional banking models fail to incorporate behavioral, agronomic, and community factors. Our solution uses a machine learning pipeline to predict loan eligibility based on diverse, localized data and generates transparent explanations for decisions.

3. Data Pipeline and Preprocessing

3.1 Input Data

The dataset comprises:

* Demographic attributes (age, gender, region)
* Financial indicators (income, assets, repayment history)
* Agricultural data (farm size, crop type, rainfall forecast)
* Behavioral patterns (mobile advisory usage, past default)

3.2 Cleaning and Transformation

* Stringified list values (e.g., ['Small Business', 'Remittance']) are parsed into readable text.
* Features are one-hot encoded.
* Missing values are imputed with zeros.
* StandardScaler is applied to normalize feature distributions.

4. Modeling Approach

4.1 Model Choice

We use *XGBoostClassifier*, chosen for its:

* High performance with tabular data
* Support for handling missing values
* Interpretability via feature importance

4.2 Training and Evaluation

* Data is split 80/20 into training and test sets.
* Evaluation metrics: accuracy, F1-score, precision, and recall.
* Typical accuracy exceeds 90%, with strong balance across classes.

4.3 SHAP Explanations

* SHAP (SHapley Additive exPlanations) is used to interpret predictions.
* We store SHAP bar and beeswarm plots for top explanations per farmer.

5. Flask API Backend

5.1 Predict Endpoint
The /predict route in Flask:

* Accepts JSON payloads representing a loan applicant
* Returns a structured response including:

    * Credit_score (float)
    * Eligibility (Approved/Rejected)
    * Reasoning (list of textual explanations)
    * Explanation_data (features and weights)
    * Plots (SHAP visualizations)
    * Recommended_limit
    * GPT_output (natural language recommendation)

5.2 Explanation Pipeline

The Flask server:

    * Loads the trained model using joblib

* Preprocesses the payload to match training schema
* Applies SHAP or fallback LIME explanations
* Generates plots saved to /static/plots/

6. Nuxt Frontend API Handler

The server/api/loan.ts route:

* Uses axios to POST to Flask server
* Accepts name, loan_amount_requested, and duration
* Receives full prediction response
* Constructs a new object with:
    * Farmer name, region, status
    * Model prediction and AI-generated recommendation
    * Full explanation_data for frontend visualization

Example Entry:

```ts
{
  name: 'Jane Akinyi',
  region: 'Busia',
  score: 0.9993,
  risk: 'Low',
  request: '80,000',
  status: 'Approved',
  model_prediction: {
    credit_score: 0.9993,
    eligibility: 'Approved',
    explanation_data: {
      features: ['region <= 2.00', 'age <= 0.19'],
      weights: [0.1, 0.05]
    },
    reasoning: [...],
    gpt_output: 'Farmer Jane Akinyi is Approved for a loan...'
  },
  ai_recommendation: 'Farmer Jane Akinyi is Approved...'
}
```

7. Assumptions and Design Considerations

7.1 Assumptions

* Behavioral and climate features have predictive power.
* Local regions are mapped numerically for model compatibility.
* Farmers are grouped into eligibility tiers based on score thresholds.

7.2 Security & Integrity

* Assumes Flask server runs in a secure, firewall-protected environment.
* Farmer input data is sanitized before model evaluation.
* No Personally Identifiable Information is persisted without consent.

8. Scalability Strategy

8.1 Model Hosting

* XGBoost model hosted in Flask can be containerized with Docker
* Can scale horizontally with load balancers (e.g., NGINX, AWS ELB)

8.2 Frontend API

* Nuxt 4 API routes are serverless-deployable (e.g., Vercel, Netlify Functions)
* Stateless API ensures distributed deployment without shared memory

8.3 Data & Insights

* SHAP plots and explanation features can be stored in Redis or PostgreSQL
* Farmers' historical requests can be version-controlled via MongoDB or DynamoDB

8.4 Monitoring & Improvements

* Logs for prediction latency, feature drift, and SHAP failures are recorded
* Support for retraining with updated data through scheduled Airflow pipelines

9. Future Work

* Integration of satellite imagery and Normalized Difference Vegetation Indices features
* Voice-based data collection via Interactive Voice Response
* Agent dashboards with interactive explanation graphs
* Automated retraining using monthly batch ingestion

10. Conclusion

The system demonstrates how AI can augment rural financial decision-making using transparent, interpretable machine learning. By combining robust modeling with real-time explanations and a scalable architecture, the platform can support credit access for underbanked populations across regions.