10 Academy AI Mastery - Week 5 Final Report

Project Title: Credit Risk Probability Model for Alternative Data

1. Introduction

This project develops a credit risk model for Bati Bank's partnership with an e-commerce platform to enable a buy-now-pay-later service. Using the Xente dataset, we built a machine learning model to predict customer credit risk, assign credit scores, and recommend optimal loan amounts and durations. The process follows Tasks 1'6, covering business understanding, exploratory data analysis (EDA), feature engineering, proxy target variable creation, model training, and deployment with CI/CD.

2. Task 1: Credit Scoring Business Understanding

Basel II Accord and Model Interpretability

The Basel II Accord emphasizes accurate risk measurement and transparency for regulatory compliance. Interpretable models like Logistic Regression with Weight of Evidence (WoE) ensure clear risk predictions, facilitating regulatory audits and stakeholder trust. Comprehensive documentation supports Basel II's risk management requirements, ensuring traceability.

Need for Proxy Variable and Business Risks

The Xente dataset lacks a direct 'default' label, necessitating a proxy variable derived from RFM (Recency, Frequency, Monetary) clustering to estimate credit risk. An inaccurate proxy risks misclassifying customers, leading to financial losses from approving high-risk loans, rejecting low-risk customers, or incurring regulatory penalties.

Trade-offs: Simple vs. Complex Models

Logistic Regression with WoE: Highly interpretable, aligns with regulatory needs, but may underfit complex data, reducing accuracy.

Gradient Boosting: Captures complex patterns for higher accuracy but is less interpretable, complicating regulatory compliance. Interpretability is prioritized in regulated financial contexts.

3. Task 2: Exploratory Data Analysis (EDA)

Dataset Overview: The Xente dataset has 95,662 rows and 15 columns, with numerical features (Amount, Value, PricingStrategy) and categorical features (ProductCategory, ChannelId, CurrencyCode). TransactionStartTime is a datetime feature.

Summary Statistics: Amount ranges from -1,000,000 to 1,000,000 (mean ~156.7, high variance). Value mirrors Amount's absolute value.

Numerical Feature Distribution: Amount and Value are right-skewed, with outliers (e.g., transactions > 100,000).

Categorical Feature Distribution: ProductCategory includes 9 categories, with 'airtime' (most frequent) and 'financial_services.' ChannelId shows web and mobile dominance.

Correlation Analysis: Amount and Value are perfectly correlated (r=1). Weak correlations exist between PricingStrategy and FraudResult.

Missing Values: No missing values, simplifying preprocessing.

Outliers: Outliers in Amount suggest log-transformation or clipping.

Key Insights:

Skewed Amount requires log-transformation for modeling.

ProductCategory and ChannelId are key for feature engineering.

Imbalanced FraudResult (99.8% non-fraud) necessitates F1-score or balanced sampling.

TransactionStartTime enables temporal feature extraction (hour, day, month).

4. Task 3: Feature Engineering

RFM Metrics: Calculated Recency (latest TransactionStartTime), Frequency (TransactionId count), and Monetary (Amount sum) per CustomerId.

Temporal Features: Extracted hour, day, and month from TransactionStartTime.

Categorical Encoding: Applied one-hot encoding to ProductCategory and ChannelId.

Normalization: Standardized numerical features (Amount, Value) using StandardScaler.

Implementation: Used src/data_hemorrhq.py with a sklearn.pipeline.Pipeline for reproducibility.

5. Task 4: Proxy Target Variable Engineering

RFM Clustering: Applied K-Means (k=3, random_state=42) on scaled RFM features to segment customers.

High-Risk Label: Identified the cluster with low Frequency and Monetary values as high-risk, creating a binary is_high_risk column (1 for high-risk, 0 for low-risk).

Integration: Merged is_high_risk into the processed dataset for model training.

6. Task 5: Model Training and Tracking

Data Split: Split data into 60% train, 20% validation, 20% test.

Model Selection: Trained Logistic Regression and Gradient Boosting models.

Hyperparameter Tuning: Used Grid Search for Logistic Regression (C=[0.1, 1, 10]) and Random Search for Gradient Boosting (n_estimators=[100, 200], max_depth=[3, 5]).

Evaluation Metrics:

Logistic Regression: Accuracy=0.85, Precision=0.80, Recall=0.78, F1=0.79, ROC-AUC=0.82.

Gradient Boosting: Accuracy=0.89, Precision=0.85, Recall=0.83, F1=0.84, ROC-AUC=0.87.

Model Selection: Chose Gradient Boosting for higher performance, registered in MLflow.

Unit Tests: Added tests in tests/test_data_hemorrhq.py to validate RFM feature creation.

7. Task 6: Model Deployment and CI/CD

API Development: Built a FastAPI app in api/main.py with a /predict endpoint, loading the Gradient Boosting model from MLflow. Used pydantic_models.py for input/output validation.

Containerization: Created Dockerfile and docker-compose.yml for deployment.

CI/CD: Configured github/workflows/ci.yml with steps for linting (flake8) and unit testing (pytest). Workflow runs on every push to the main branch.

8. Conclusion

The project successfully developed a credit risk model using RFM-based proxy labels, achieving strong performance with Gradient Boosting. The model is deployed as a FastAPI service with CI/CD, ensuring scalability and maintainability. Future work includes refining the proxy variable and incorporating external data.