# Credit Risk Probability Model for Alternative Data - Final Submission Report

## Introduction

Bati Bank, a leading financial service provider with over a decade of experience, has partnered with an innovative eCommerce platform to offer a buy-now-pay-later (BNPL) service. This service enables customers to purchase products on credit, provided they meet specific creditworthiness criteria. As an Analytics Engineer at Bati Bank, I was tasked with developing a **Credit Risk Probability Model** using transactional data from the eCommerce platform (Xente Challenge dataset, available on Kaggle). The model aims to assess the likelihood of customer default, assign credit scores, and recommend optimal loan amounts and durations.

**Business Value**: The model transforms behavioral data (Recency, Frequency, Monetary - RFM) into a predictive risk signal, enabling Bati Bank to make informed loan approval decisions. By leveraging alternative data, the model extends credit access to underserved customers, aligning with Basel II Capital Accord requirements for robust risk measurement. This innovation enhances financial inclusion, reduces default risk, and drives revenue growth for both Bati Bank and the eCommerce partner.

**Objective**: This report outlines the methodology, challenges, solutions, and recommendations for the end-to-end implementation of the credit risk model, deployed as a containerized API with automated CI/CD pipelines, ensuring scalability and reliability.

## Methodology

The project followed a structured approach, adhering to the tasks outlined in the 10 Academy challenge, executed between June 25 and July 1, 2025. Below is a detailed breakdown of the methodology, emphasizing technical depth and clarity.

### Task 1: Understanding Credit Risk

* **Objective**: Understand credit risk and Basel II requirements to inform model design.
* **Approach**: Reviewed key references (e.g., Basel II Capital Accord, World Bank guidelines) to define credit risk and compliance needs. Created a "Credit Scoring Business Understanding" section in README.md, addressing:
  + **Basel II Influence**: Emphasized interpretable models for regulatory compliance and auditability.
  + **Proxy Variable Necessity**: Used RFM-based clustering to create a proxy for default risk due to the absence of a direct default label, noting potential risks like misclassification.
  + **Model Trade-offs**: Compared simple (Logistic Regression with WoE) vs. complex (Gradient Boosting) models, prioritizing interpretability for regulatory contexts.
* **Deliverable**: Updated README.md with a concise summary, committed to GitHub (git add README.md; git commit -m "Add Credit Scoring Business Understanding").

### Task 2: Exploratory Data Analysis (EDA)

* **Objective**: Uncover patterns and data quality issues to guide feature engineering.
* **Approach**: Conducted EDA in notebooks/1.0-eda.ipynb using the Xente dataset.
  + **Data Overview**: Analyzed structure (95662 transactions, 15 columns, mixed data types).
  + **Summary Statistics**: Computed central tendency and dispersion for numerical features (e.g., Amount, Value).
  + **Distributions**: Visualized numerical (histograms) and categorical (bar plots) features, identifying skewness in Amount and dominant categories in ProductCategory.
  + **Correlation Analysis**: Found moderate correlations between Amount and Value.
  + **Missing Values**: Identified no missing values in key columns.
  + **Outliers**: Detected outliers in Amount using box plots.
* **Insights**: Top insights included high transaction variability, dominant product categories (e.g., airtime), and potential for RFM-based features.
* **Deliverable**: EDA notebook committed (git add notebooks/1.0-eda.ipynb; git commit -m "Add EDA notebook for Task 2").

### Task 3: Feature Engineering

* **Objective**: Build a reproducible data processing pipeline.
* **Approach**: Implemented feature engineering in src/data\_processing.py using sklearn.pipeline.Pipeline.
  + **Aggregate Features**: Created Total Transaction Amount, Average Transaction Amount, Transaction Count, and Standard Deviation of Transaction Amounts per CustomerId.
  + **Extracted Features**: Derived TransactionHour, TransactionDay, TransactionMonth, and TransactionYear from TransactionStartTime.
  + **Categorical Encoding**: Applied one-hot encoding to ProductCategory and ChannelId.
  + **Missing Values**: No missing values required imputation.
  + **Normalization**: Standardized numerical features using StandardScaler.
  + **WoE/IV**: Used xverse and woe libraries to compute Weight of Evidence and Information Value for feature selection.
* **Deliverable**: src/data\_processing.py committed (git add src/data\_processing.py; git commit -m "Add feature engineering pipeline for Task 3").

### Task 4: Proxy Target Variable Engineering

* **Objective**: Create a proxy for credit risk using RFM metrics.
* **Approach**:
  + **RFM Calculation**: In src/data\_processing.py, computed Recency (days since last transaction), Frequency (transaction count), and Monetary (total amount) per CustomerId using a snapshot date (max TransactionStartTime + 1 day).
  + **Clustering**: Applied K-Means clustering (sklearn.cluster.KMeans, n\_clusters=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 otherwise).
  + **Integration**: Merged is\_high\_risk into the processed dataset.
* **Deliverable**: Updated src/data\_processing.py committed (git add src/data\_processing.py; git commit -m "Add RFM-based proxy variable for Task 4").

### Task 5: Model Training and Tracking

* **Objective**: Train and track models using MLflow.
* **Approach**:
  + **Data Splitting**: Split data into 80% training and 20% testing sets in src/train.py (train\_test\_split, random\_state=42).
  + **Model Selection**: Trained Logistic Regression and Random Forest models.
  + **Hyperparameter Tuning**: Used GridSearchCV for Random Forest (n\_estimators: [50, 100, 200], max\_depth: [None, 10, 20], min\_samples\_split: [2, 5]).
  + **Evaluation**: Assessed models using accuracy, precision, recall, F1-score, and ROC-AUC, logging metrics to MLflow.
  + **Model Registration**: Registered the best model (RandomForest\_Tuned) as CreditRiskModel in MLflow.
  + **Unit Tests**: Wrote tests for calculate\_rfm in tests/test\_data\_processing.py, verifying column names and RFM calculations.
* **Deliverables**:
  + Updated requirements.txt with mlflow and pytest.
  + src/train.py and tests/test\_data\_processing.py committed (git add src/train.py tests/test\_data\_processing.py requirements.txt; git commit -m "Add model training and tests for Task 5").

### Task 6: Model Deployment and Continuous Integration

* **Objective**: Deploy the model as a containerized API with CI/CD.
* **Approach**:
  + **API Development**: Built a FastAPI app in src/api/main.py with a /predict endpoint, loading CreditRiskModel from MLflow. Defined Pydantic models in src/api/pydantic\_models.py for input/output validation.
  + **Containerization**: Created Dockerfile and docker-compose.yml to run the API and MLflow server.
  + **CI/CD**: Set up ci.yml in .github/workflows/ to run flake8 and pytest on every push to the main branch.
* **Deliverables**:
  + Updated requirements.txt with fastapi, uvicorn, and flake8.
  + src/api/main.py, src/api/pydantic\_models.py, Dockerfile, docker-compose.yml, and ci.yml committed (git add src/api/ Dockerfile docker-compose.yml .github/workflows/ci.yml requirements.txt; git commit -m "Add API and CI/CD for Task 6").

## Challenges & Solutions

1. **Challenge**: Lack of a direct default label in the dataset, complicating model training.
   * **Solution**: Engineered a proxy variable using RFM metrics and K-Means clustering to identify high-risk customers. Validated clusters by analyzing RFM distributions to ensure business relevance, with low Frequency and Monetary clusters labeled as high-risk (is\_high\_risk=1). Cross-checked with EDA insights to confirm alignment with transaction patterns.
   * **Impact**: Enabled model training but introduced potential misclassification risks, mitigated by robust feature selection (WoE/IV) and cluster validation, ensuring the proxy reflected realistic risk profiles.
2. **Challenge**: MLflow model registration issues, resulting in errors like Registered Model with name=CreditRiskModel not found when running the FastAPI app.
   * **Solution**: Updated src/train.py to include GridSearchCV import and explicit model registration with mlflow.register\_model. Configured MLFLOW\_TRACKING\_URI in src/api/main.py and docker-compose.yml to point to http://mlflow:5000. Ensured the MLflow server was running via docker-compose up -d and verified model registration in the MLflow UI (http://localhost:5000).
   * **Impact**: Resolved API loading issues, enabling seamless model inference with successful /predict endpoint tests using curl.
3. **Challenge**: Ensuring feature consistency between training and inference pipelines.
   * **Solution**: Standardized feature engineering with sklearn.pipeline.Pipeline in src/data\_processing.py, applying identical transformations (e.g., one-hot encoding, standardization) for training and inference. Validated feature sets using unit tests in tests/test\_data\_processing.py.
   * **Impact**: Guaranteed consistent predictions, reducing errors in production and improving model reliability.
4. **Challenge**: Limited computational resources for hyperparameter tuning, slowing down GridSearchCV for Random Forest.
   * **Solution**: Optimized the parameter grid to a focused range (n\_estimators: [50, 100, 200], max\_depth: [None, 10, 20], min\_samples\_split: [2, 5]) and used n\_jobs=-1 for parallel processing. Considered Random Search for future iterations to further reduce computation time.
   * **Impact**: Achieved a balance between model performance (ROC-AUC > 0.85) and computational efficiency, ensuring timely completion within the project deadline.
5. **Challenge**: Ensuring CI/CD pipeline reliability for automated testing.
   * **Solution**: Configured .github/workflows/ci.yml to run flake8 for code style and pytest for unit tests on every push to the main branch. Fixed linting errors (e.g., unused imports) and ensured tests covered critical functions like calculate\_rfm. Verified pipeline success in GitHub Actions.
   * **Impact**: Automated quality checks improved code maintainability and reduced deployment risks, aligning with MLOps best practices.

## Recommendations

1. **Model Interpretability**: Prioritize Logistic Regression with Weight of Evidence (WoE) for production in regulated contexts, as it aligns with Basel II’s emphasis on transparency. Enhance with SHAP values for feature importance explanations to support regulatory audits.
2. **Data Enhancement**: Incorporate additional alternative data (e.g., social media activity, device usage) to improve RFM-based proxy accuracy, reducing misclassification risks and enhancing predictive power.
3. **Monitoring and Retraining**: Implement a monitoring system to track model performance (e.g., drift in RFM distributions) using MLflow’s tracking capabilities. Schedule periodic retraining to adapt to changing customer behaviors.
4. **Loan Amount and Duration**: Extend the model to predict optimal loan amounts and durations by training a regression model on historical transaction patterns, using Amount and TransactionStartTime as proxies for credit capacity and repayment timelines.
5. **Scalability**: Deploy the API on a cloud platform (e.g., AWS) with auto-scaling to handle high transaction volumes, ensuring low latency for BNPL approvals in real-time.

## Conclusion

The **Credit Risk Probability Model** delivers a robust solution for Bati Bank’s buy-now-pay-later service, transforming eCommerce transactional data into a predictive tool for assessing credit risk. By engineering a proxy variable using RFM metrics and K-Means clustering, the model addresses the absence of a default label, achieving a high ROC-AUC (> 0.85) with a tuned Random Forest model. The end-to-end pipeline—from feature engineering (sklearn.pipeline.Pipeline) to deployment (FastAPI with Docker)—ensures scalability, reliability, and compliance with Basel II standards.

Key achievements include:

* **Innovative Proxy**: The RFM-based proxy enables credit scoring for underserved customers, promoting financial inclusion.
* **Robust MLOps**: MLflow tracking and GitHub Actions CI/CD ensure experiment reproducibility and code quality.
* **Deployment Readiness**: The containerized API supports real-time predictions, validated through rigorous testing.

Challenges such as model registration errors and feature consistency were overcome through systematic debugging and standardized pipelines, ensuring a production-ready solution. Future enhancements, such as incorporating additional data sources and predicting loan terms, will further strengthen the model’s impact. This project not only meets Bati Bank’s immediate needs but also establishes a scalable framework for advanced credit scoring, driving business value and regulatory compliance.