Credit Risk Model for High-Risk Customer Identification

Executive Summary

This report details the development of a robust and interpretable credit risk modeling system, a key task within the Kifiya AI Mastery training program. Leveraging alternative transaction data, the project's core objective is to programmatically identify and predict **high-risk customer segments**. The solution encompasses comprehensive data preprocessing, advanced feature engineering (including RFM metrics and Weight of Evidence transformations), rigorous model training and hyperparameter tuning with MLflow tracking, and a full deployment setup via containerized API and CI/CD pipeline. The final model, a Gradient Boosting Classifier, achieved a high ROC AUC score of 0.9929, demonstrating excellent capability in distinguishing high-risk customers, crucial for proactive risk management and regulatory compliance in the financial sector.

1. Credit Scoring Business Understanding

Credit scoring models are vital tools in the financial industry, enabling lenders to assess the creditworthiness of borrowers and manage credit risk effectively. This section delves into key considerations for developing such a model, particularly within a regulated environment.

1.1. Influence of Basel II Accord on Model Interpretability and Documentation

The **Basel II Capital Accord** is a cornerstone of international banking regulation, designed to ensure financial institutions maintain adequate capital reserves against risks. Its three pillars profoundly influence the requirements for credit risk models:

- **Pillar 1: Minimum Capital Requirements:** Mandates that banks accurately measure and quantify credit risk (Probability of Default - PD, Loss Given Default - LGD, Exposure at Default - EAD) to determine their capital allocation. This necessitates models that are not only predictive but also **transparent** in their decision-making process. An interpretable model allows financial institutions to understand the drivers of risk, validate its assumptions, and ensure its outputs align with business realities and regulatory expectations.
- **Pillar 2: Supervisory Review Process:** Empowers regulators to critically assess a bank's internal capital adequacy assessment process (ICAAP) and the robustness of their risk models. **Interpretability** is paramount here, as it helps supervisors understand model limitations, sensitivities, and the institution's capacity to manage model risk. Black-box models are challenging to audit and justify.
- **Pillar 3: Market Discipline:** Promotes transparency through enhanced public disclosure of risk management practices. While less direct, understandable models contribute to better public confidence and market discipline.

In essence, Basel II transforms credit risk modeling from a purely predictive exercise into a critical regulatory and risk management function where **interpretability** and **thorough documentation** are paramount for trust, compliance, and sound decision-making.

1.2. Necessity of Proxy Variables for Default and Associated Business Risks

In many real-world credit datasets, a direct "default" label (e.g., a formal declaration of bankruptcy or loan charge-off) is often scarce or unavailable, especially for new customers or specific product types. Therefore, creating a **proxy variable** for default becomes a necessary step. This proxy typically involves programmatically defining "high-risk" based on observable customer behaviors that correlate with a high likelihood of future default.

For this project, we lack a direct "default" column in the raw transaction data. Instead, we defined a proxy based on **customer engagement patterns** derived from their transaction history. High-risk groups are identified as those with a high likelihood of default – typically characterized by disengagement (low frequency and low monetary value of transactions, and high recency of last transaction).

Potential Business Risks of Making Predictions Based on a Proxy:

- **Misclassification Risk:** The primary risk is that the proxy may not perfectly align with the true definition of default.
    - **False Positives (Type I Error):** Classifying a borrower as "high-risk" based on the proxy when they would not have truly defaulted. This leads to denying credit to creditworthy customers, resulting in **lost revenue** and **damaged customer relationships**.
    - **False Negatives (Type II Error):** Classifying a borrower as "low-risk" when they will genuinely default. This leads to **increased loan losses** and **undermines the bank's financial stability**.
- **Operational Misalignment:** Business processes (e.g., collections, recovery strategies) are often tied to formal default definitions. If the model's proxy-based predictions diverge significantly, it can create inefficiencies.
- **Regulatory Scrutiny:** Regulators will scrutinize the definition and justification of any proxy variable. A poorly defined proxy could lead to non-compliance issues.
- **Evolving Risk Landscape:** The effectiveness of a proxy can diminish over time as economic conditions or customer behaviors change, requiring continuous re-evaluation.

Therefore, careful consideration, robust validation, and continuous monitoring of the chosen proxy variable are crucial to mitigate these business risks.

1.3. Logical Alignment of Project Objectives with Derived Insights

The project's objective to identify high-risk customers directly supports loan approval and risk management decisions:

- **Proactive Risk Management:** By identifying disengaged (proxy high-risk) customers early, financial institutions can implement proactive strategies such as targeted interventions, personalized communication, or adjusted credit limits, potentially preventing actual defaults.
- **Informed Loan Approval:** The model's predicted probability of a customer being high-risk provides a data-driven score that loan officers can use to make more informed decisions, supplementing traditional credit checks. This allows for more precise risk-based pricing and tailored loan products.
- **Capital Allocation:** Accurate identification of high-risk segments directly feeds into Basel II's Pillar 1 requirements, enabling banks to allocate regulatory capital more efficiently and accurately against unexpected losses.
- **Customer Segmentation:** The RFM clustering not only creates the proxy but also provides valuable customer segments that can be used for differentiated strategies beyond just credit risk, such as marketing or customer service.

2. Exploratory Data Analysis (EDA) Insights

A comprehensive Exploratory Data Analysis (EDA) was performed on the raw transactional dataset (data.csv) to understand its structure, quality, and inherent patterns. Key insights derived from this phase informed the subsequent feature engineering steps:

1. **Data Volume and Structure:** The dataset comprises approximately 95,662 transaction-level records across 16 columns. This volume is manageable for local processing and provides a rich basis for deriving customer-level insights.
2. **Absence of Explicit Missing Values:** Initial checks confirmed no explicit missing values (NaNs) across any columns in the raw dataset. This simplified the initial data cleaning process.
3. **Single Currency and Country:** All transactions are uniformly in 'UGX' (Ugandan Shilling) and originate from a single CountryCode (256). This simplifies analysis by eliminating the need for currency conversion or cross-country pattern analysis within this dataset.
4. **Redundancy between 'Amount' and 'Value':** The features 'Amount' and 'Value' exhibit a near-perfect positive linear correlation (0.99). While both capture transaction magnitude, 'Amount' includes negative values (likely reversals/refunds) while 'Value' appears to be a positive magnitude. This redundancy suggests that careful handling (e.g., selecting one or deriving new features from their interplay) is needed to avoid multicollinearity.
5. **Significant Outliers in 'Amount' and 'Value':** Both 'Amount' and 'Value' distributions are

highly skewed with a substantial number of extreme outliers (very large or very negative values). These outliers were addressed through **capping (winsorization)** during preprocessing to mitigate their disproportionate impact on statistical analyses and model training, while retaining the information that extreme values might convey about risk.

6. **Categorical Nature of Identifiers and Product Information:** Features like ProviderId, ProductId, ProductCategory, and ChannelId are categorical identifiers. PricingStrategy also behaves as a categorical variable despite being numerical. These require appropriate encoding (specifically, **Weight of Evidence (WoE) transformation** for its interpretability and suitability in credit scoring contexts) to be usable by machine learning models.
7. **Temporal Feature Potential from 'TransactionStartTime':** The TransactionStartTime column, initially an object type, was identified as a critical source for extracting valuable temporal features (e.g., hour of day, day of month, month, year of the last transaction per customer). These features can reveal behavioral patterns (e.g., transaction habits at specific times or seasons) that are highly indicative of customer engagement and potential risk.
8. **Imbalanced Proxy Target Variable:** The FraudResult column, initially used as a temporary proxy for "bad outcome" during early EDA, revealed a highly imbalanced distribution (only ~0.2% fraudulent transactions). This highlighted the need for specialized techniques (e.g., stratify=y in data splitting, class_weight='balanced' in models, appropriate evaluation metrics like ROC AUC) to handle class imbalance during model training.

3. Feature Engineering

This project employs a robust, automated, and reproducible feature engineering pipeline (src/data_processing.py) to transform raw transaction data into a model-ready format. The pipeline is built using sklearn.pipeline.Pipeline and custom transformers (defined in src/transformers.py) to ensure all preprocessing steps are consistently applied, preventing data leakage and enhancing reproducibility.

The key feature engineering steps, executed sequentially within the pipeline (or as a pre-pipeline step for target generation), include:

1. **Proxy Target Variable Engineering (RFM & Clustering):**
    - **Purpose:** To programmatically create a binary is_high_risk target variable, as a direct "credit risk" column was not available in the raw data. This proxy identifies "disengaged" customers who are deemed high-risk for default.
    - **Steps:**
        - **RFM Metrics Calculation:** For each customer, Recency (days since last transaction relative to a snapshot date), Frequency (total transaction count), and Monetary (total transaction amount) were calculated from their transaction history.
        - **RFM Scaling:** RFM features were scaled using StandardScaler to ensure equal contribution during K-Means clustering.

        - **K-Means Clustering:** Customers were segmented into 3 distinct groups based on their scaled RFM profiles using KMeans with a fixed random_state for reproducibility.
        - **High-Risk Labeling:** The cluster characterized by the highest average Recency, lowest average Frequency, and lowest average Monetary value was identified as the is_high_risk segment (assigned 1), with all other customers labeled as 0.
    - **Implementation:** Handled by the generate_high_risk_target function in src/transformers.py. The resulting is_high_risk Series serves as the y for the subsequent WoE transformation and model training.
2. **Customer-Level Aggregation:**
    - **Purpose:** To shift the data granularity from individual transactions to a comprehensive customer-level view, which is essential for predicting customer default risk. This step also extracts temporal features.
    - **Features Created:**
        - total_transaction_amount: Sum of all transaction amounts for each customer.
        - average_transaction_amount: Average transaction amount per customer.
        - transaction_count: Total number of transactions for each customer (Frequency).
        - std_transaction_amount: Variability of transaction amounts per customer (NaNs filled with 0 for single-transaction customers).
        - average_pricing_strategy: Average pricing strategy used by the customer.
        - distinct_product_categories: Number of unique product categories a customer has engaged with.
        - **Temporal Features (from the *last* transaction of each customer):** last_transaction_hour, last_transaction_day, last_transaction_month, last_transaction_year.
    - **Implementation:** Handled by the CustomerAggregator custom transformer in src/transformers.py.
3. **Weight of Evidence (WoE) Transformation:**
    - **Purpose:** To convert selected numerical features (from aggregation and temporal extraction) into WoE scores. This transformation handles non-linearity, provides interpretability (coefficients directly reflect risk impact), and is particularly well-suited for Logistic Regression models in credit scoring. It also implicitly handles categorical features by binning and assigning WoE.
    - **Implementation:** Handled by the RobustWoETransformer custom transformer in src/transformers.py, which includes internal quantile-based binning for continuous numerical features (e.g., total_transaction_amount).
4. **Feature Scaling (Standardization):**
    - **Purpose:** To standardize all WoE-transformed features to have a mean of 0 and a standard deviation of 1. This ensures that features are on a similar scale, preventing any

single feature from dominating the model and improving the performance of distance-based or gradient-descent-based algorithms.
    - **Implementation:** Handled by StandardScaler from sklearn.preprocessing within the pipeline.

The final output of this phase is data/processed/model_features.csv (the transformed features, 3742 rows, 10 columns) and data/processed/is_high_risk_target.csv (the generated proxy target variable, 3742 rows), both at the customer level.

4. Model Training and Tracking

This section details the structured model training process, including experiment tracking with MLflow, hyperparameter tuning, and comprehensive model evaluation. The src/train.py script orchestrates these steps.

4.1. Methodology

- **Data Splitting:** The processed customer-level data (model_features.csv and is_high_risk_target.csv) was split into training (80% - 2993 customers) and testing (20% - 749 customers) sets using train_test_split. Crucially, stratify=y was applied to maintain the proportion of high-risk customers in both sets, vital for the imbalanced target variable.
- **Model Selection:** Three distinct machine learning classification algorithms were chosen for experimentation, representing different learning paradigms:
    - **Logistic Regression:** A linear, highly interpretable model often favored in regulated financial contexts.
    - **Random Forest Classifier:** A powerful ensemble tree-based method known for its robustness and high performance.
    - **Gradient Boosting Classifier (GBM):** Another strong ensemble method, typically offering high predictive accuracy by sequentially building trees.
- **Baseline Models:** Untuned instances of each chosen model were trained to establish a performance benchmark for comparison against their tuned counterparts.
- **Hyperparameter Tuning:** GridSearchCV was employed for systematic hyperparameter optimization for each model:
    - **Cross-Validation:** StratifiedKFold (with 5 splits and random_state=42) was used within GridSearchCV to ensure robust evaluation across different data subsets and maintain class balance in each fold.
    - **Scoring Metric:** ROC AUC (Receiver Operating Characteristic - Area Under Curve) was chosen as the primary scoring metric for optimization, as it is a robust measure for evaluating binary classifiers on imbalanced datasets.
- **MLflow Experiment Tracking:** Each training run (baseline and tuned) was meticulously tracked using MLflow. This involved logging:

- Model type and specific hyperparameters.
- Evaluation metrics (ROC AUC, Accuracy, Precision, Recall, F1-Score) on the test set.
- The model artifact itself, enabling versioning and easy retrieval.

4.2. Key Findings & Model Performance

The experiments demonstrated strong predictive capabilities across all models on the high-risk customer identification task. The performance on the test set, with ROC AUC as the primary metric, is summarized below:

Model	**Best CV ROC AUC (Training)**	**Test ROC AUC**	**Test Precision (Class 1)**	**Test Recall (Class 1)**	**Test F1-Score (Class 1)**
Logistic Regression (Baseline)	N/A	0.9722	0.99	0.86	0.92
Logistic Regression (Best Tuned)	0.9702	0.9723	0.99	0.86	0.92
Random Forest (Baseline)	N/A	0.9856	0.96	0.92	0.94
Random Forest (Best Tuned)	0.9909	0.9902	0.96	0.91	0.93
Gradient Boosting (Baseline)	N/A	**0.9929**	0.94	0.91	0.93
Gradient Boosting (Best Tuned)	0.9922	0.9908	0.93	0.91	0.92

Analysis of Model Performance:

- **Overall Strong Performance:** All models achieved very strong performance, with ROC AUC scores consistently above 0.97. This indicates that the engineered RFM-derived features are highly predictive of the is_high_risk proxy target.
- **Ensemble Models Outperform Logistic Regression:** Random Forest and Gradient Boosting consistently achieved higher ROC AUC scores compared to Logistic Regression,

demonstrating their superior ability to capture complex patterns and rank high-risk customers.

- **Tuning Impact:** Hyperparameter tuning provided notable improvements for Random Forest. For Logistic Regression, the improvement was marginal. Interestingly, the baseline Gradient Boosting model achieved the highest ROC AUC, suggesting its default parameters were already highly effective for this dataset, or the defined tuning grid did not yield further gains on the test set.
- **Precision and Recall:** While all models showed good precision and recall for the high-risk class, Logistic Regression tended towards higher precision (fewer false positives) at the cost of recall (missing more actual high-risk cases). Ensemble models generally offered a better balance.

4.3. Conclusion on Best Model

Based on the **highest Test ROC AUC score of 0.9929**, the **Gradient Boosting Classifier (Baseline) model** is identified as the best performing model for this high-risk customer identification task. This model exhibits excellent discriminative power and a strong balance of precision and recall.

4.4. Model Registration and Saving

The best performing model, the Gradient Boosting (Baseline) Classifier, has been successfully:

- **Registered in the MLflow Model Registry** under the name CreditRiskClassifier as Version 1 (Run ID: b4834c7e5f834aa18ba89115b8c115aa). This ensures robust model versioning, metadata tracking, and facilitates future deployment.
- **Saved locally to disk** at models/final_best_credit_risk_model.pkl for direct access and deployment.
- The **fitted feature engineering pipeline** (feature_engineering_pipeline.pkl) was also saved locally to ensure consistent preprocessing of new inference data.

5. Model Deployment and Continuous Integration

This section outlines the setup for deploying the trained model as a containerized API and establishing a CI/CD pipeline to ensure code quality and automated testing.

5.1. API Development (src/api/)

A REST API has been developed using **FastAPI** to serve real-time credit risk predictions:

- **src/api/pydantic_models.py**: Defines PredictionRequest and PredictionResponse Pydantic models for strict data validation of incoming requests and outgoing responses, ensuring data integrity at the API boundary.
- **src/api/main.py**:

- Initializes the FastAPI application.
- On startup, it loads the final_best_credit_risk_model.pkl (with a fallback to local .pkl if MLflow Registry is inaccessible) and the feature_engineering_pipeline.pkl.
- Exposes a /predict endpoint that:
  - Accepts new raw customer transaction data (as a list of transactions) validated by PredictionRequest.
  - Internally applies the loaded feature_engineering_pipeline to transform the raw data into the model-ready format.
  - Uses the loaded ML model to predict the risk probability and binary high-risk label.
  - Returns the prediction in the PredictionResponse format.

5.2. Containerization

The entire service is containerized using Docker for consistent and isolated deployment environments:

- **Dockerfile**: Defines the Docker image for the credit_risk_api service. It sets up the Python environment, installs dependencies from requirements.txt, copies the application code, and specifies uvicorn as the server to run the FastAPI application.
- **docker-compose.yml**: Orchestrates the Docker service for local development and testing. It defines the credit_risk_api service, builds its image from the Dockerfile, maps host port 8000 to container port 8000, and mounts the project directory (including data/ and models/) into the container to enable access to processed data and saved models.

5.3. Continuous Integration (CI) with GitHub Actions

A CI pipeline is configured using GitHub Actions to automate testing and ensure code quality on every push to the main branch:

- **.github/workflows/ci.yml**: Defines the workflow.
  - **Trigger:** Activated on push events to the main branch.
  - **Python Setup:** Sets up a Python 3.9 environment.
  - **Dependency Installation:** Installs all project dependencies from requirements.txt.
  - **Code Linting:** Runs flake8 on the src/ directory to check for code style violations and potential programming errors.
  - **Unit Testing:** Executes pytest on the tests/ directory to run all defined unit tests for the data processing and custom transformers.
  - **Build Failure:** The workflow is configured to fail if either the linter or any unit tests fail, ensuring that only high-quality, verified code is integrated.

6. Conclusion and Future Work

This project successfully developed a robust credit risk probability model leveraging alternative transaction data, demonstrating a full end-to-end machine learning workflow from data

understanding to deployment. The use of RFM-based proxy labeling, WoE transformation, and MLflow tracking ensures interpretability, reproducibility, and effective experiment management, which are critical in regulated financial environments.

Future Work:

- **Refine Proxy Target:** Explore more sophisticated methods for defining the "default" proxy, potentially incorporating external data (e.g., actual loan repayment status if available) or more complex behavioral indicators.
- **Advanced Feature Engineering:** Investigate additional aggregate features (e.g., time-series features like rolling averages of transaction amounts over different windows), or interaction features.
- **Model Explainability (XAI):** For the Gradient Boosting Model, implement and visualize SHAP or LIME values to enhance its interpretability, making its "black-box" nature more transparent for business and regulatory stakeholders.
- **Production MLflow Tracking:** Set up a remote MLflow Tracking Server (e.g., on AWS S3 or Azure Blob Storage) to centralize experiment tracking and model registry across different environments (local, Colab, Docker).
- **API Scalability:** Implement more advanced API deployment strategies (e.g., Kubernetes, serverless functions) for higher scalability and reliability in a production environment.
- **Monitoring:** Establish continuous monitoring for model performance (e.g., drift detection, accuracy over time) and data quality in production.

This project serves as a strong foundation for building and deploying interpretable credit risk models in the FinTech domain.