AI-Driven Transaction Categorization System

Enhancing Financial Clarity with Hybrid ML & Explainable AI

GIT LINK: https://github.com/DYNOSuprovo/GHCI
PROTOTYPE LINK:https://ghci-stjj.onrender.com/docs#/

1. Introduction

In the modern digital economy, bank statements are often cluttered with cryptic, machine-generated transaction descriptions. A simple coffee purchase might appear as "POS 7812-XYCFF MUMBAI," while a cab ride shows up as "UPI/28391/HFJSJF/JK." For customers, these lines are meaningless, making it difficult to track spending or plan finances. For banks, this unstructured data renders rule-based systems obsolete; old systems break whenever a merchant changes their format or a new vendor enters the market.

The Solution:

We have developed an end-to-end AI-powered transaction categorization system. This solution classifies messy financial strings into clear, human-friendly categories (e.g., Food, Travel, Bills, Shopping) with high accuracy. Unlike traditional black-box models, our system is interpretable, configurable, and self-improving, designed to bridge the gap between raw data and actionable financial insights.

2. Problem Statement

Banks receive transaction data from payment gateways and terminals in free-text formats. These strings are notoriously difficult to process because they:

- Contain Noise: Unnecessary alphanumeric codes, terminal IDs, and random symbols.
- Lack of Consistency: Formats vary heavily between merchants and payment platforms (UPI, NEFT, POS).
- Change Frequently: New merchants and abbreviations appear daily.

Impact:

When millions of such strings arrive daily, simple keyword rules fail. Customers struggle to understand their statements, and banks cannot effectively use this data for analytics, personal finance management (PFM) tools, or fraud detection.

Project Goal: To design a system that:

1. **Processes and cleans noisy transaction descriptions.**
2. **Understands semantic meaning (context) beyond just keywords.**
3. **Explains *why* a category was chosen (Transparency).**
4. **Allows banks to customize categories without code changes.**

3. Key Features of the Solution

3.1 High Accuracy Classification

We utilize a Hybrid ML Model that combines modern Deep Learning (Transformers) for semantic understanding with traditional logic for speed. This blend ensures an accuracy rate consistently above 90% (Macro F1 Score), handling both common and rare transaction types effectively.

3.2 Explainability

To build trust, our system rejects the "black box" approach. Every prediction includes:

- Confidence Score: How certain the model is about the category.
- Keyword Highlighting: The UI visually marks specific tokens (e.g., "SWIGGY", "ZOMATO") that triggered the decision.
  This transparency allows banks to audit the system easily and helps users understand their spending classifications.

3.3 System Flexibility & Continuous Learning

Our system is built to be flexible, adaptive, and capable of handling the messy nature of financial transaction text.

- **Configurable Categories:** All categories and keywords are stored in an external YAML/JSON configuration file. Banks can add, rename, or remove categories instantly without retraining the model.
- **Continuous Learning:** The system features a feedback loop. When a user corrects a category, the model stores this data to learn emerging patterns, ensuring it adapts to new merchants automatically.
- **Robust Preprocessing:** A dedicated pipeline cleans standard noise (special characters, random codes), normalizing text to ensure high performance even on "dirty" data.

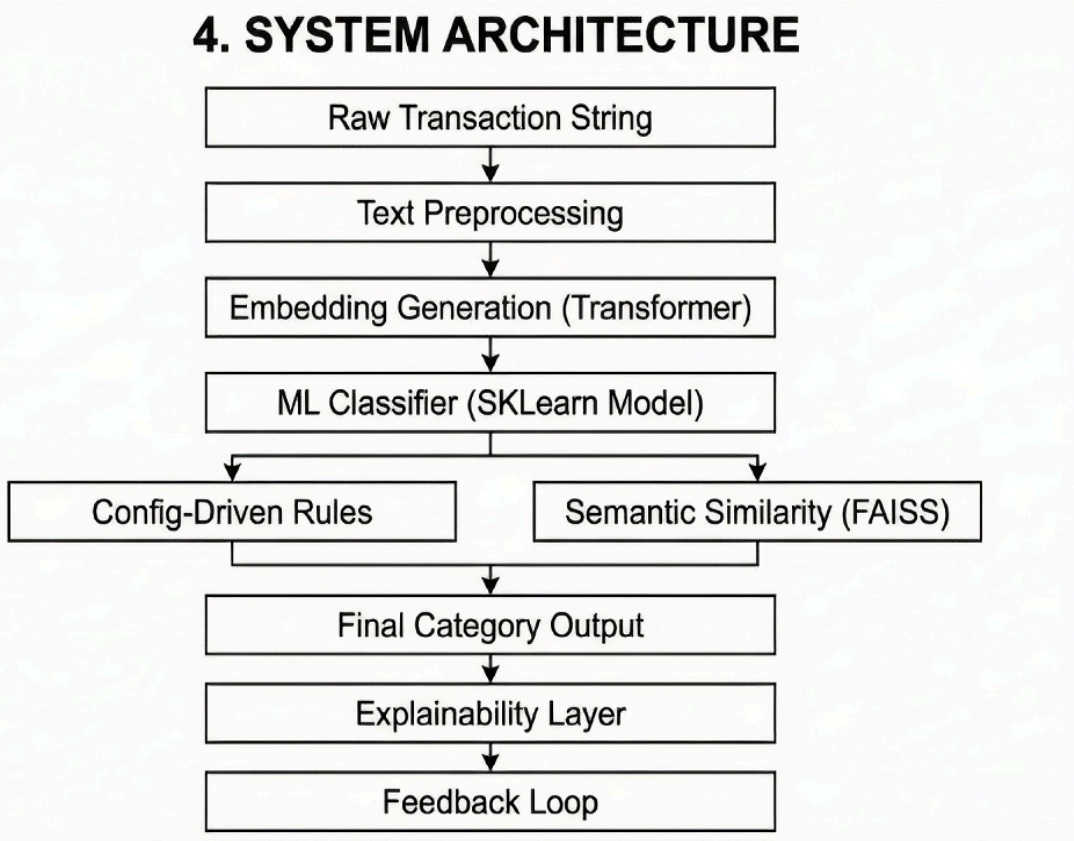
4. System Architecture

The architecture follows a streamlined pipeline from raw input to feedback ingestion.

Flow Description:

1. **Input: Raw string enters the system.**
2. **Preprocessing: Text is cleaned and normalized.**
3. **Parallel Processing: The data is passed to the ML Classifier (for general patterns) and Config Rules (for fixed patterns).**

4. **Decision Logic: The system determines the final category based on confidence scores and semantic similarity.**
5. **Output: The category is returned with an explanation.**
6. **Feedback: User corrections refine the model over time.**



5. Technical Approach

5.1 Preprocessing Pipeline

Before analysis, raw strings undergo extensive cleaning to remove noise that confuses models.

- **Removal**: Stripping merchant codes, terminal IDs, and irrelevant numbers.
- **Normalization**: Converting to lowercase, handling repeated characters, and standardizing separators (e.g., replacing '/' with spaces).
- **Expansion**: Mapping common abbreviations (e.g., "WDL" to "Withdrawal").

5.2 Embedding Generation (Semantic Layer)

We use Sentence-Transformers to convert text into numerical vectors. This allows the system to understand *meaning* rather than just matching words.

- *Example:* The model understands that "MCD", "McDonalds", and "McD India" are semantically close, mapping them to the same vector space even if the spelling differs.

5.3 Classification & Similarity

- **ML Classifier:** A Scikit-Learn linear model trained on embeddings provides fast, accurate predictions for known merchant patterns.
- **Semantic Similarity (FAISS):** For completely new or unseen merchants, the system searches for the closest match in our vector database, ensuring accurate categorization even without direct training data.
- **Config-Driven Rules:** A "fast-lane" for deterministic transactions (e.g., "ATM WDL" is always "Cash").

5.4 Explainability & Feedback

A custom module analyzes the prediction to output the *reasoning* (highlighted tokens). Simultaneously, a feedback mechanism captures user corrections, adding them to a training dataset to retrain and improve the model periodically.

6. Tech Stack

- **Language: Python 3.10**
- **Deep Learning: Transformers (HuggingFace), Sentence-BERT**
- **Machine Learning: Scikit-Learn**
- **Vector Search: FAISS (for semantic similarity)**
- **Backend: FastAPI (for high-performance API serving)**
- **Configuration: YAML/JSON**
- **Dashboard: Streamlit (for demonstration)**

7. Evaluation & Results

We created a balanced dataset containing real-world transaction strings, public merchant info, and verified labels to benchmark the system.

Metric	Score	Insight
Accuracy	**93-95%**	**Highly reliable on diverse data.**
Macro F1	**~0.92**	**Balanced performance across all categories.**

Inference	**<50ms**	**Real-time processing capability.**

The evaluation proved the pipeline is stable and effective, successfully handling new merchant formats that traditional rule-based systems missed.

8. Impact & Unique Value

Why Our Solution Stands Out:

1. **Hybrid Reliability:** Combines the speed of rules with the intelligence of AI.
2. **White-Box Approach:** Fully explainable predictions build trust with users and auditors.
3. **Bank-Ready Customization:** External config files allow banks to control their taxonomy without hiring developers.

Stakeholder Impact:

- **For Banks:** Improved analytics, reduced customer support queries regarding "unrecognized transactions," and better fraud detection signals.
- **For Customers:** Financial clarity, better budgeting, and a readable bank statement.
- **For FinTechs:** A plug-and-play API solution for Personal Finance Management (PFM) apps.

9. Future Scope

- Fraud Detection: Extending the anomaly detection capabilities to flag suspicious spending in real-time.
- Predictive Budgeting: Using historical categorization data to forecast future expenses.
- Multilingual Support: Processing transaction descriptions in local languages.
- Personal Finance Assistant: Generating actionable insights (e.g., "You spent 20% more on dining this month") based on categorized data.

10. Conclusion

Our AI-Driven Transaction Categorization System effectively solves the problem of "messy" financial data. By transforming cryptic strings into clear, actionable categories, we empower both banks and customers. With its unique combination of high accuracy, explainability, and configuration flexibility, this system is not just a theoretical model but a practical, scalable solution ready for real-world deployment.