UPI Scam Detection Using Transaction Behavior and Social Engineering Triggers

line 1: 1st Samarth Karmakar
line 2: *dept. of Comptuer Science Engineering*
line 3: *Manipal Institute of Technology*
line 4: Manipal,India
line 5: samshwetha08@gmail.com

line 1: 2nd Given Name Surname
line 2: *dept. name of organization (of Affiliation)*
line 3: *name of organization (of Affiliation)*
line 4: City, Country
line 5: email address or ORCID

line 1: 3rd Given Name Surname
line 2: *dept. name of organization (of Affiliation)*
line 3: *name of organization (of Affiliation)*
line 4: City, Country
line 5: email address or ORCID

***Abstract*—This paper presents a machine learning framework for detecting fraudulent transactions within the Unified Payments Interface (UPI) system. With the exponential growth of digital payments, UPI scams, often employing sophisticated social engineering tactics, have become a significant threat. Our approach utilizes a Random Forest Classifier trained on a synthetically generated dataset that mimics real-world transaction patterns. The model looks closely at important features that reveal how transactions behave and identify subtle cues linked to social engineering tricks—things like when a transaction happens in the day, how much it deviates from a user's usual spending, and whether it involves certain types of requests known to be risky. The results show that this approach is effective at telling the difference between genuine and fraudulent transactions, with a strong ability to catch scams without missing many. This framework offers a solid and trustworthy base for building stronger security measures to protect users in the fast-growing world of digital payments..**

Keywords—UPI, scam detection, machine learning, feature engineering, Random Forest, transaction behavior.

I. INTRODUCTION

The Unified Payments Interface, or UPI, has truly transformed how people in India manage their money. Imagine being able to send or receive cash instantly, right from your phone—no more long waits or complicated paperwork. That's the kind of convenience UPI offers to millions every day. But with this ease and speed, the risks have also grown. Unfortunately, scammers have become much craftier. Instead of breaking into accounts, they now use clever tricks to fool ordinary people, playing on their trust, fear, or a sense of urgency. These scams are especially tricky because they don't always raise the usual red flags that traditional fraud systems are designed to catch.

That's why this project is so important. Instead of relying on old-school rules that look for obvious signs of fraud, we're teaching a machine learning system to notice the little, subtle things that hint at something fishy going on—things like transactions done too quickly, unusual types of transactions, or patterns that just don't quite fit with a person's normal behavior. By focusing on these "red flags" that scammers try to hide behind, the system can catch more scams before people lose money.

Our paper walks you through how we did this—from creating a realistic set of data that includes both regular and scam transactions, to building and training the model itself, to testing how well it works. We also talk about exciting next steps, like adding real-time alerts or even looking at how users behave over time to catch scams before they happen. The goal? To build a smarter, more adaptable safety net around UPI, protecting people not just with technology, but by understanding how scammers think and how people respond. This way, the system doesn't just react to fraud, it stays one step ahead to keep your money safe.

II.METHODOLOGY

The way we approached this project breaks down into three main parts that build on each other. First, we focused on **data synthesis**, where we created a realistic set of transaction data to work with, since real financial data can be hard to access due to privacy concerns. Next, we moved on to **feature engineering**, where we carefully picked and designed important traits from the data that could help the model spot unusual or suspicious behavior. Finally, we tackled **model architecture selection**, choosing the best machine learning method to learn from our features and effectively distinguish between legitimate and fraudulent transactions. This step-by-step process helped us create a strong foundation for detecting scams in a way that's both practical and reliable.

A. Dataset and Implementation

One of the biggest challenges in building fraud detection models is the lack of access to real financial transaction data, mainly because it's highly sensitive and private. To work around this, we created a synthetic dataset that simulates 10,000 UPI transactions, carefully balancing realism with the ethical need to protect privacy. Within this dataset, about 15% of the transactions are flagged as scams—a proportion that reflects the natural imbalance you'd find in real-world scenarios, where fraud is a smaller but critical subset of all activity.

The **legitimate transactions** in this dataset are designed to resemble everyday usage patterns. You'll find all kinds of typical transactions here: people paying utility bills or shopping for groceries, usually during regular hours like between 8 AM and 10 PM. The amounts vary but tend to lean on the smaller side, just like most normal users' spending behavior.

In contrast, the **scam transactions** were carefully created to include several telltale signs that fraudsters often use. For example, many of these have oddly high, fixed amounts—such as ₹19,999—which is a common tactic to bypass certain payment limits or catch users off guard. These suspicious transactions also tend to happen during unlikely hours—think midnight to 5 AM—when most people are asleep. Another distinctive feature is the frequent use of the 'COLLECT_REQUEST' transaction type, known to be a preferred method for social engineering frauds because it asks users to authorize payments, often under false pretenses.

For the technical side of things, the entire project is implemented using Python—a language well-loved by data enthusiasts worldwide. For handling the data and building the model, we turned to some tried-and-true data science tools that make the process smoother and more efficient. We used Pandas because it's great for organizing and cleaning data—kind of like sorting through a big messy inbox to find exactly what you need. Behind the scenes, NumPy took care of the heavy lifting with all the complex number crunching. Then, to actually build, train, and test our machine learning model, we relied on Scikit-learn, a toolkit that provides reliable, easy-to-use algorithms and utilities. Together, these libraries gave us the flexibility and power to experiment, tweak, and continuously improve the model until we got it just right.

This approach not only safeguards privacy but also gives us full control over the data's characteristics, making it possible to test how well the model picks up on subtle scam signals in a controlled, ethical manner. It's a careful blend of technical precision and thoughtful design, aiming to strike a meaningful balance between innovation and responsibility.

B. Feature Engineering

To boost the model's ability to correctly spot scams, we carefully handcrafted a range of features from the original transaction data that help paint a clearer picture of each transaction's context and behavior.

One of the key improvements came from **temporal features**. By looking closely at exactly when each transaction happens—both the hour of the day and the day of the week—the model learns to recognize typical timing patterns. This means it can spot transactions that take place at strange hours or on days when most people wouldn't normally be making payments. Those unusual moments stand out as red flags, helping the model to identify potentially suspicious activity that deserves a closer look. This helps catch those late-night or odd-timed transactions that are often a hallmark of fraud.

Another powerful feature we developed is **behavioral deviation**, measured as the difference between the current transaction amount and the user's average transaction amount. When the transaction amount suddenly spikes far beyond someone's usual spending pattern, that's a strong red flag for potential fraud. This "amount deviation" feature acts like an early warning, helping the model recognize when something just doesn't fit the user's normal behavior.

Beyond numbers, real-world transaction data often includes categories like payment type or transaction context, which can't be directly understood by machine learning algorithms if left as text. To handle this, we applied **categorical encoding** techniques, specifically one-hot encoding, which translates these non-numeric categories—like 'transaction_type' or 'payment_context'—into a set of easy-to-interpret numeric features. This way, the model can effectively learn from all aspects of a transaction, including the context behind it, enhancing its overall predictive power.

Together, these thoughtfully engineered features give the model a richer, more nuanced understanding of the data, enabling it to better separate the innocent transactions from the suspicious ones, even when the signs of fraud are subtle. This blend of time, behavior, and context helps create a smarter system that's attuned to the real-life complexities of financial scams.

C. Model Architechture

The core of our detection system is a **Random Forest Classifier**. This ensemble learning model was chosen for several reasons:

- **Performance:** It is highly effective on tabular data and can capture complex, non-linear relationships between features.
- **Robustness:** By averaging the predictions of many individual decision trees, it is less prone to overfitting compared to a single tree.
- **Handling Imbalance:** We configured the model with class_weight='balanced', which automatically adjusts weights inversely proportional to class frequencies. This forces the model to pay more attention to the minority class (scams), which is crucial for our objective.

III.TRAINING DETAILS

When it was time to train and evaluate the model, the carefully prepared dataset was divided into two parts: 75% for training and 25% for testing. To maintain fairness and realism, we used a stratified split, which means the ratio of scam transactions to legitimate ones stayed consistent in both sets. This was crucial so the model gets trained on a representative mix and is tested on data that reflects real-world scenarios.

Before feeding the data into the model, all the numerical features were standardized using Scikit-learn's StandardScaler. This step is like leveling the playing field—no feature with a larger scale could overshadow the others during learning. For instance, transaction amounts often

have different numeric scales compared to time features, so standardization helps the model interpret all features equally well.

For the prediction algorithm, we chose a Random Forest model, which is essentially a collection of many decision trees working together. Specifically, we built the forest with 100 trees (n_estimators=100). This size strikes a good balance, providing robust performance while keeping training time manageable. The ensemble nature of Random Forest helps the system generalize better and avoid overfitting, making it more reliable when spotting scams in new, unseen transactions.

II. Model Performance Evaluation Metrics

- **Precision and Recall**
  - Recall is the most important metric for scam detection. It measures the model's ability to identify all actual scams.
  - High recall means fewer scams go undetected (fewer false negatives), which is crucial for protecting users.
  - Precision indicates the proportion of predicted scams that were actually scams, helping to manage false positives.
- **F1-Score**
  - The F1-score is the harmonic mean of precision and recall.
  - It provides a balanced, single metric that considers both false positives and false negatives.
  - This score helps assess overall model effectiveness in detecting scams without overwhelming users with false alarms.
- **Confusion Matrix**
  - A visual summary of the model's prediction outcomes.
  - Breaks down results into true positives (correctly detected scams), true negatives (correctly identified legitimate transactions), false positives, and false negatives.
  - Offers clear insight into where the model performs well and where improvements are needed.

These evaluation metrics collectively offer a comprehensive understanding of how well the model detects fraudulent transactions while minimizing mistakes, ensuring reliable real-world performance.

IV.FUTURE WORKS

While this project establishes a strong proof-of-concept, there are several avenues for future enhancement:

1. **Real-Time Deployment:** The current model works on a static dataset. The next step would be to deploy it as a real-time API that can score live transactions as they occur.
2. **Advanced Models:** Explore more complex algorithms like Gradient Boosting (XGBoost, LightGBM) or Deep Learning (Neural Networks) which may yield higher accuracy.
3. **Richer Data Sources:** Incorporating additional data points like user device information, IP geolocation, and transaction history with the beneficiary could significantly improve detection capabilities.
4. **Adaptive Learning:** Scammers constantly change their tactics. A production system should include a mechanism for periodic retraining on new data to adapt to emerging fraud patterns.

V.CODE

1)Graphical User Interface and Prediction Pipeline

To make the scam detection tool accessible and easy to use for a broad range of users, we designed an intuitive graphical user interface (GUI) using Python's Tkinter and CustomTkinter libraries. The application launches with a crisp, modern window that guides users through entering details for a UPI transaction they wish to analyze.

2)Workflow Overview

When a user opens the application, they are presented with a form that captures all relevant transaction attributes, such as:

- **Amount:** The value of the transaction in rupees.
- **Transaction Type:** A selection between standard payment modes, including 'DEBIT', 'CREDIT', or the more scam-prone 'COLLECT_REQUEST'.
- **High Frequency Indicator:** Informs the model about recent transaction activity, helping detect anomalous bursts.
- **Flags for New Beneficiary or New Device:** These binary indicators help assess risks associated with previously unseen recipients or usage from a new device.

3)Prediction Engine

Upon submission, the system collects these inputs and maps them precisely onto the features expected by our scam detection model. Before making a prediction, it calculates additional contextual features such as the ratio of the transaction amount to a user's average transaction amount and translates categorical selections into numerical values that the machine learning model can process. The model—previously trained and serialized using LightGBM—is loaded and applied directly to the user's input.

The prediction output is then visually communicated on the GUI through both text and a dynamically colored progress bar, indicating the estimated probability that the transaction is fraudulent. The interface uses a three-tier risk system:

- **High Risk:** Transactions with probabilities over 70%, signaled in red, are flagged as likely scams.
- **Medium Risk:** Probabilities in the 30-70% range trigger an orange warning, suggesting additional scrutiny may be prudent.
- **Low Risk:** A green signal for probabilities below 30% indicates a transaction likely to be legitimate.

4)Usability and Professional Presentation

The GUI is designed for clarity and speed: users can quickly reset the form to analyze multiple transactions, and feedback is immediate and color-coded for intuitive interpretation. This approach combines robust machine learning with an emphasis on user trust and transparency, making complex

risk assessment both simple and actionable for everyday use.

By encapsulating the entire risk evaluation workflow in an interactive GUI, our solution bridges advanced technology and real-world practicality, empowering users to make informed decisions in their financial transactions.

VI. CONCLUSION

This project successfully demonstrates the viability of using a machine learning approach to detect UPI scams. By synthesizing a realistic dataset and engineering features that capture both behavioral anomalies and social engineering triggers, we trained a Random Forest model capable of identifying fraudulent transactions with high recall. The methodology presented provides a comprehensive framework for developing intelligent security solutions that can safeguard users within the growing digital payments landscape. This work underscores the potential

REFERENCES

1] L. Breiman, "Random Forests," *Machine Learning*, vol. 45, no. 1, pp. 5–32, 2001.

[2] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly detection: A survey," *ACM Computing Surveys (CSUR)*, vol. 41, no. 3, pp. 1-58, 2009.

[3] National Payments Corporation of India (NPCI), "Statistics of Unified Payments Interface (UPI)," NPCI Press, 2025.

[4] R. J. Bolton and D. J. Hand, "Statistical fraud detection: A review," *Statistical Science*, vol. 17, no. 3, pp. 235-255, 2002.

[5] S. J. Stolfo, W. Fan, W. Lee, A. Prodromidis, and P. K. Chan, "Credit card fraud detection using meta-learning: Issues and data," in *Proceedings of the AAAI-98 Workshop on AI Approaches to Fraud Detection and Risk Management*, 1998.

[6] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*, 2nd ed. New York: Springer, 2009.

[7] H. He and E. A. Garcia, "Learning from imbalanced data," *IEEE Transactions on Knowledge and Data Engineering*, vol. 21, no. 9, pp. 1263-1284, 2009.

[8] A. Dal Pozzolo, G. Boracchi, O. Caelen, C. Alippi, and G. Bontempi, "Credit Card Fraud Detection: A Realistic Modeling and a Novel Learning Strategy," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 29, no. 8, pp. 3784-3797, 2018.

[9] K. D. Mitnick and W. L. Simon, *The Art of Deception: Controlling the Human Element of Security*. Wiley Publishing, 2002.

[10] W. McKinney, *Python for Data Analysis: Data Wrangling with Pandas, NumPy, and IPython*, 2nd ed. O'Reilly Media, 2017.

[11] J. Jordon, J. van der Schaar, and M. van der Schaar, "Synthesizing Tabular Data using Generative Adversarial Networks," in *Advances in Neural Information Processing Systems 32 (NeurIPS)*, 2019.