**Fraud Detection Capstone Project Report**

**Candidate Name:** Saurabh Dolas  
**Tools Used:** Python, scikit-learn, XGBoost, SMOTE, Streamlit

**Introduction**

In the era of rapid digitization, financial fraud has evolved into a serious threat to businesses, consumers, and the banking sector. With the increase in online financial transactions, the volume and complexity of fraudulent activities have grown exponentially. This project aims to develop a robust fraud detection system using machine learning algorithms that not only detects fraudulent transactions with high precision but also evaluates the financial cost and savings implications of the model.

**Data Exploration (EDA)**

The dataset contains 11,142 transaction records with 10 key features, including transaction type, amount, sender and receiver balances (oldbalanceOrg, newbalanceOrig, etc.), and a binary label isFraud.

**Key EDA Steps:**

* **Transaction Type Distribution:** Majority of transactions were PAYMENT (≈49.5%), followed by CASH\_IN, CASH\_OUT, TRANSFER, and DEBIT.
* **Fraud Distribution:** Approximately 10.25% of the transactions were labeled as fraudulent, indicating class imbalance.
* **Visual Analysis:**
  + Countplots showed that most frauds occurred in TRANSFER and CASH\_OUT types.
  + Boxplots revealed fraudulent transactions often involved higher transaction amounts.
  + Histogram analysis of time step showed frauds concentrated in certain time windows.
* **Correlation Heatmap:** No variable other than step showed strong correlation with isFraud, ruling out major leakage.

**Conclusion:** Visual and statistical patterns indicate high-value, high-risk transactions cluster around a few types, which can be leveraged during modeling.

**Feature Engineering**

New features were engineered to enhance predictive power:

* deltaOrig = oldbalanceOrg - newbalanceOrig
* deltaDest = newbalanceDest - oldbalanceDest
* isMerchantDest: Flag based on whether the recipient is a merchant (name starts with 'M')

Categorical Encoding:

* type was one-hot encoded using pd.get\_dummies() with drop\_first=True to prevent multicollinearity.

Dropped:

* nameOrig, nameDest were dropped due to high cardinality and potential data leakage.

**Handling Class Imbalance**

Given that only ~10% of the data was fraudulent, **SMOTE (Synthetic Minority Over-sampling Technique)** was applied on the training data to synthetically balance the classes. This helps avoid model bias toward non-fraud cases.

**Model Selection and Training**

Three different classifiers were evaluated:

1. **Logistic Regression**
2. **Random Forest Classifier**
3. **XGBoost Classifier** *(best performer)*

**Data Processing:**

* Features were scaled using StandardScaler.
* 70:30 stratified train-test split ensured class proportions were preserved.

**Model Evaluation Metrics**

| **Model** | **Precision** | **Recall** | **F1-Score** | **ROC-AUC** |
| --- | --- | --- | --- | --- |
| Logistic Regression | 0.88 | 0.96 | 0.92 | 0.96 |
| Random Forest | 0.93 | 0.99 | 0.96 | 0.98 |
| XGBoost | 0.99 | 0.99 | 0.99 | 0.999 |

* **XGBoost** outperformed all other models across all metrics.
* Confusion matrices and ROC curves were plotted side-by-side to visually compare performance.
* Precision-Recall Curve for XGBoost demonstrated robustness even at low thresholds.

**Financial Impact Analysis**

To quantify the business value:

* **Total fraud amount**: ₹1.36 Billion
* **Fraud caught by model**: ₹1.361 Billion (99.97%)
* **False positive cost**: ₹181K

A **threshold tuning analysis** showed that optimal thresholds (≈0.05) minimize false negatives while keeping business cost lowest (₹18,000).

Additionally, a simulation computed total **business cost** based on:

* Missed frauds (FN) = Rs. 10,000 loss each
* Investigating false positives (FP) = Rs. 500 per case

**Business Questions Addressed**

**Model's precision and accuracy in detecting fraudulent transactions**

* In this project, the XGBoost classifier achieved a precision of 0.99 (99%), meaning that only 1% of flagged transactions were false positives. This is critical in financial systems because too many false alerts can overload investigative teams and reduce confidence in the model.
* With a recall of 0.99 (99%), XGBoost ensures that nearly all fraudulent transactions are detected. This helps prevent revenue losses and reputational damage.
* F1-Score, the harmonic mean of precision and recall, was 0.99, indicating excellent balance between catching frauds and avoiding false alarms.
* Accuracy is not the ideal metric in imbalanced datasets but was also high (>99%) due to strong model performance and class balancing using SMOTE.

**Interpretation:**The XGBoost model is extremely precise and accurate. It ensures that 99 out of every 100 flagged transactions are truly fraudulent, reducing unnecessary escalations. This high precision helps optimize fraud investigation costs and avoid customer dissatisfaction due to wrongful alerts.

**Reliability in fraud detection**

* ROC-AUC of 0.999: The Receiver Operating Characteristic - Area Under Curve (ROC-AUC) is near-perfect, signifying that the model distinguishes between fraud and non-fraud with very high confidence, regardless of decision threshold.
* Precision-Recall Curve: The curve maintained high precision even at lower recall thresholds, proving that the model maintains robustness across a wide range of probability thresholds. This ensures flexibility in operational deployment (e.g., for low- vs high-value transactions).
* Confusion Matrix Stability: Even after applying threshold tuning and SMOTE, the confusion matrix showed minimal false negatives (missed frauds) and low false positives.
* Cross-validation Consistency: Cross-validation on training folds showed consistent metrics across folds, validating model generalizability.

**Interpretation:**The model is highly reliable across transaction scenarios. It offers consistent classification results even when fraud patterns are subtle or transaction volumes fluctuate. Such reliability is crucial for banking and fintech institutions to deploy the model in production environments with minimal oversight.

**Potential losses due to model errors**

In fraud detection, every False Negative (FN) — a missed fraud — represents a direct financial loss. Each False Positive (FP) — a legitimate transaction wrongly flagged — represents a cost in investigation time and customer dissatisfaction. To evaluate these, a custom business cost simulation was conducted:

* Total fraud value in dataset: ₹1.36 Billion
* Fraud amount detected by model: ₹1.361 Billion
* Fraud detection rate: 99.97%
* False Positive cost per transaction: ₹500
* False Negative loss per missed fraud: ₹10,000

At the optimal threshold (≈0.05) derived via tuning:

* False Negatives were reduced to just a few cases, keeping potential missed fraud loss under ₹10,000.
* False Positives, though slightly higher, resulted in an investigation cost of only ₹181K, which is minimal compared to the savings from preventing fraudulent payouts.

**Interpretation:**At optimal threshold settings, the model minimizes total financial impact. It detects nearly all frauds, saving ₹1.361 billion, while limiting operational costs from false alarms. This trade-off between risk and cost is favorable, making the model both economically efficient and operationally viable.

**Model Deployment & Testing**

To ensure the practical usability of the fraud detection model, a Streamlit-based web application was developed and deployed. This interface allows users to upload a transaction dataset and instantly receive predictions on whether each transaction is fraudulent or not, based on the trained XGBoost model.

**Deployment Highlights:**

* Built using Streamlit for rapid prototyping and deployment.
* Accepts CSV file uploads from users.
* Performs feature engineering (same as training logic) in real-time.
* Applies the saved XGBoost model and scaler to generate predictions.
* Displays key metrics such as predicted fraud count, fraud probability, and fraud rate.
* Includes interactive threshold tuning and business cost simulation features.
* Allows download of results for further review or investigation.

**Functional Test Performed**

To validate the deployment setup, the application was tested using the same dataset on which the model was originally trained. This step ensured that:

* The feature transformation pipeline behaves as expected.
* Predictions generated through the deployed app match those from the training phase.
* summaries and business impact metrics reflect accurate results based on real model output.

**Screenshot of Deployment**

The figure below shows the web interface for uploading the transaction file within the deployed app

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

Figure: Fraud Detection App - File Upload Interface

**Conclusion**The fraud detection pipeline designed in this project demonstrates a comprehensive application of data science principles, from EDA and feature engineering to model tuning and financial cost simulation. XGBoost emerged as the most effective model with near-perfect performance metrics and minimal business cost.

To extend the solution into a real-world context, the final model was successfully deployed via a Streamlit-based application. This app allows users to upload transaction files, visualize fraud insights, run predictions in real-time, and simulate business cost impact under different decision thresholds.

The deployment was functionally validated using the same dataset from model training, confirming the pipeline’s robustness and reliability.

This end-to-end solution is scalable, interpretable, and production-ready, making it well-suited for fraud risk teams in banking and fintech domains.

**Future Scope and Enhancements**

While the current fraud detection framework is robust and effective, several improvements can further enhance its accuracy, scalability, and real-world deployment potential. The following directions outline meaningful extensions for future work:

**1. Temporal Pattern Modeling**

Currently, transactions are treated as independent events. Incorporating **sequential or time-series modeling** (e.g., using RNNs or time-aggregated features) can capture behavioral shifts over time, such as sudden spikes in transfer frequency, inactivity followed by large transactions, or inconsistent geolocation trails.

**2. Behavioral and Network Features**

Future iterations can extract **user behavior patterns**, such as average transaction size per user or transaction timing. Additionally, **graph-based features** (e.g., transaction networks between users or devices) can uncover fraud rings or intermediary mule accounts, improving detection of coordinated frauds.

**3. Hybrid Model Architecture**

Combining **unsupervised anomaly detection** (e.g., Isolation Forests, Autoencoders) with supervised classifiers like XGBoost can help detect novel fraud tactics that do not follow past patterns — especially useful in a constantly evolving fraud landscape.

**4. Model Explainability and Governance**

Adoption in financial institutions demands transparency. Leveraging **SHAP (SHapley Additive Explanations)** or **LIME** can offer per-transaction insights into why a decision was made, aiding human investigators and satisfying regulatory requirements.

**5. Cost-Aware Threshold Tuning**

Rather than using a fixed 0.5 threshold, future models should explore **dynamic thresholding** based on transaction amount or account risk profile. This allows for minimizing expected business cost (missed fraud vs false alarms) more effectively.

**6. Real-Time Model Deployment**

The current model is batch-based. Integrating with **real-time data pipelines** (via Apache Kafka or Spark Streaming) can facilitate instant fraud detection and prevention before funds are disbursed.

**7. External Data Enrichment**

Integrating additional data sources, such as:

* Customer profiles (e.g., credit score, KYC status)
* Historical fraud watchlists
* Device/browser fingerprinting  
  can enhance prediction by capturing more holistic signals beyond transaction metadata.

**8. AutoML and Hyperparameter Optimization**

Future efforts can automate model tuning through **Bayesian optimization** or tools like **AutoML frameworks** (e.g., H2O, TPOT) to discover better-performing models with minimal manual intervention.

**9. Model Retraining and Monitoring**

To stay relevant, the model should be **continuously monitored and retrained** with fresh data. Implementing a feedback loop where flagged frauds are validated and incorporated into future training cycles can reduce drift and maintain performance.