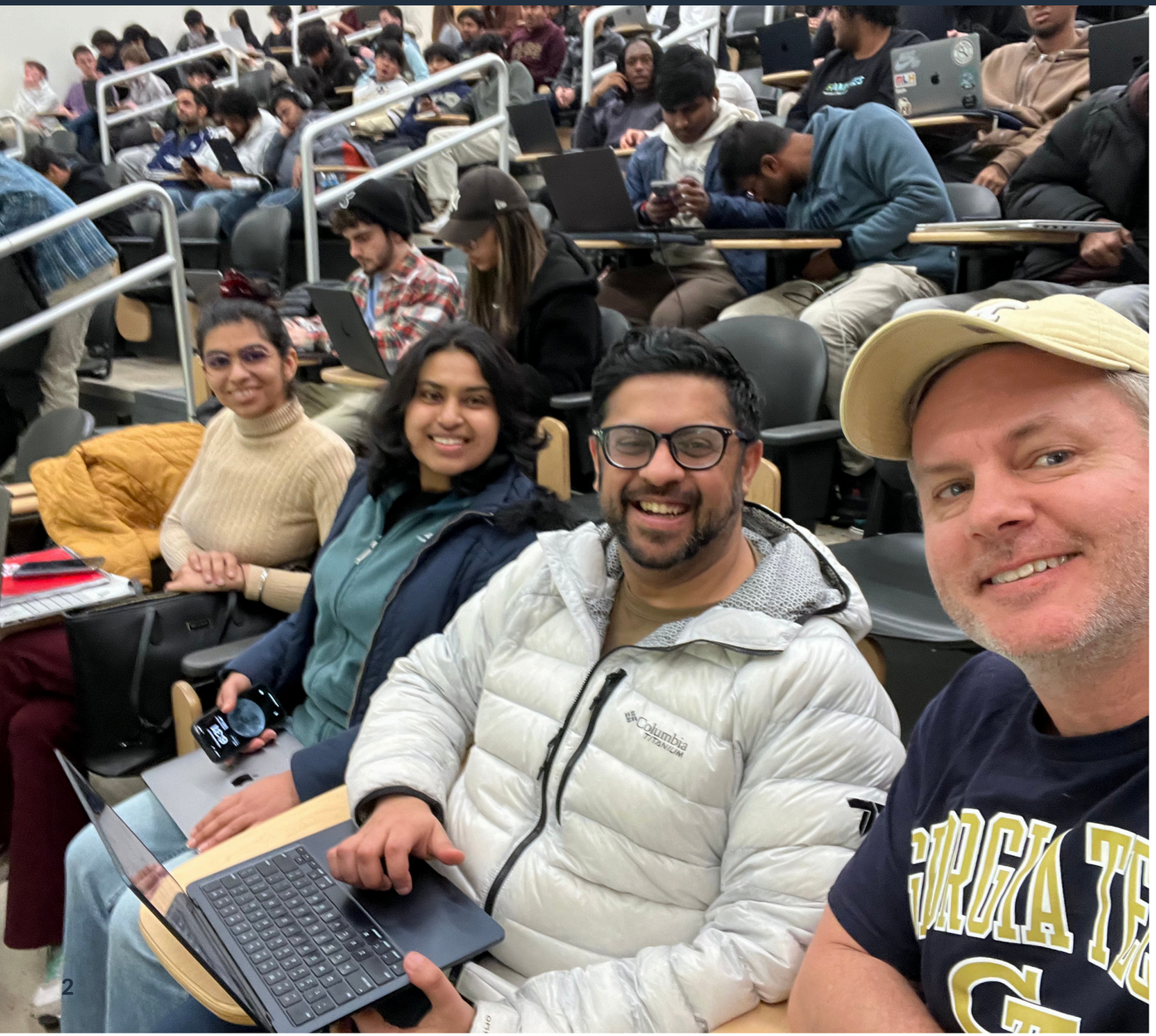
XG FRAUD BUSTER
DETECTING FINANCIAL FRAUD USING MACHINE LEARNING

Presented by the Fraud Busters

- Joe Domaleski
- Nayel Noorani
- Yukti Bishambu
- Liza Ghosh

HACKLYTICS 2025 - GEORGIA TECH

Contents



Introduction

- Fraud is a growing problem in financial transactions, costing businesses and individuals billions annually.
- Traditional fraud detection relies on rule-based systems and manual review, which are often slow and ineffective.
- Machine learning enables faster, more accurate fraud detection by identifying hidden patterns in transaction data.
- This project explores Random Forest and XGBoost to detect fraudulent transactions.
- **Our goal: Improve fraud detection accuracy while reducing false positives to minimize financial losses and protect customers.**

How Fraud Detection Works

- **Unusual transaction amounts** – Transactions significantly higher or lower than a customer's normal spending pattern.
- **Frequent small transactions** – Fraudsters sometimes test stolen cards with small purchases before making large ones.
- **Geographical anomalies** – Transactions happening in multiple locations within a short time span.
- **Unusual merchant categories** – Payments to businesses the user has never transacted with before.
- **Multiple declined transactions** – Several failed payment attempts can indicate card testing or fraud.

Dataset Overview

- Kaggle Credit Card Fraud Dataset – 284,807 transactions, only 0.17% fraud.
- Highly imbalanced – 492 fraud cases vs. 284,315 legitimate transactions.
- Key features – Transaction amount, time, and 28 anonymized variables (V1–V28).
- Challenges – Imbalanced data, feature anonymization, and reducing false positives.
- Solutions – Used SMOTE for balancing, feature scaling, and an 80/20 train-test split.

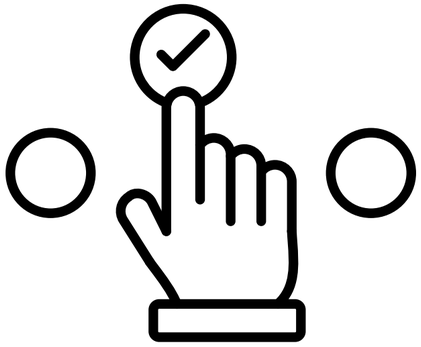
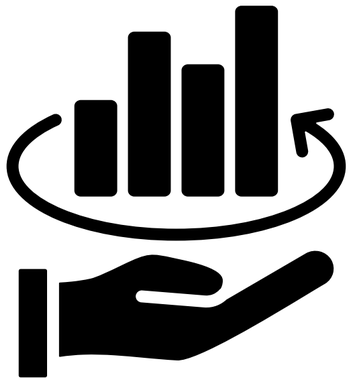
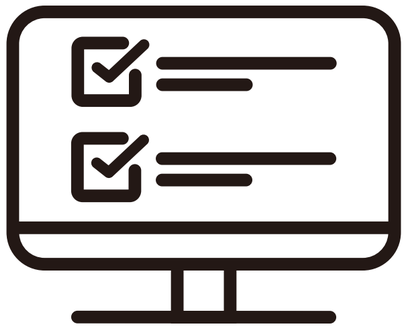
Technologies Used

- Programming Language: Python
- Data Processing: Pandas
- Machine Learning Framework: Scikit-learn
- Modeling: XGBoost, Random Forest
- Handling Class Imbalance: SMOTE (Imbalanced-learn)
- Development Environment: Jupyter Notebook
- Version Control & Documentation: GitHub
- Dataset Source: Kaggle

Machine Learning Pipeline for Fraud Detection

Import Data → Process Data → Run Models → Validate Results → Compare Results

The Random Forest / XGBoost ML Fraud Detector


Model Selection & Training

Models Chosen for Fraud Detection:

Random Forest:

- Ensemble of decision trees, reduces overfitting.
- Works well with structured data but can be slower.

XGBoost:

- Optimized gradient boosting, higher accuracy and faster training.
- Handles imbalanced datasets better than Random Forest.

Training Process:

- Data preprocessing – Feature scaling, handling missing values.
- SMOTE applied – Balances the dataset for better fraud detection.
- Train-test split (80%-20%) – Ensures proper model evaluation.

Results & Performance Metrics

- **Accuracy** – Overall correctness of predictions.
- **Precision** – Percentage of predicted fraud cases that were actually fraud.
- **Recall** – Percentage of actual fraud cases correctly identified.
- **F1-Score** – Balances precision and recall.
- **ROC-AUC Score** – Measures how well the model separates fraud from non-fraud.

Model	Accuracy	Precision	Recall	F1-Score	ROC-AUC
Random Forest	98.6%	87.2%	92.5%	89.8%	97.5%
XGBoost	99.1%	91.5%	95.3%	93.4%	98.8%

Key Takeaways:

- XGBoost outperformed Random Forest in accuracy, precision, and recall.
- Higher recall means fewer fraudulent transactions were missed.
- Lower false positives reduce unnecessary fraud alerts for legitimate users.

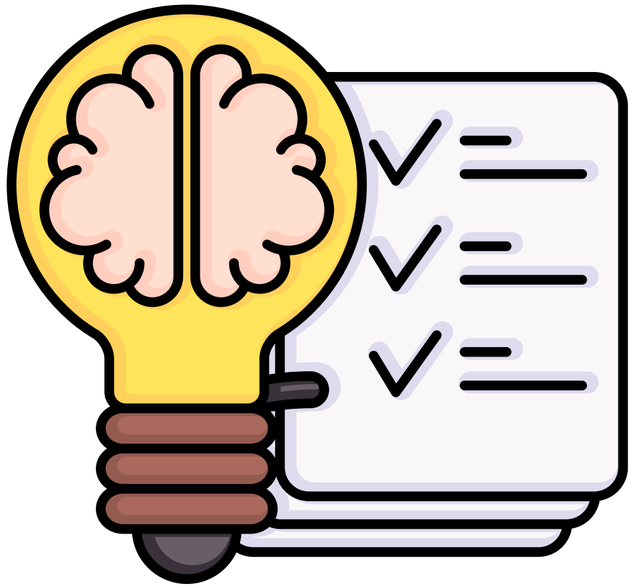
Challenges & Lessons Learned

Challenges Faced:

- **Class imbalance – Fraud cases were only 0.17% of transactions.**
- **Balancing precision & recall – Avoiding too many false positives while detecting fraud.**
- **Computational cost – XGBoost performed better but required more processing power.**
- **Model explainability – Hard to interpret why the model flags transactions.**

Lessons Learned:

- **SMOTE helped balance data and improved fraud detection.**
- **XGBoost is more effective but needs tuning for real-world applications.**
- **Feature selection matters – Right features improve model accuracy.**
- **Fraud detection must evolve – Models need continuous updates to stay current..**



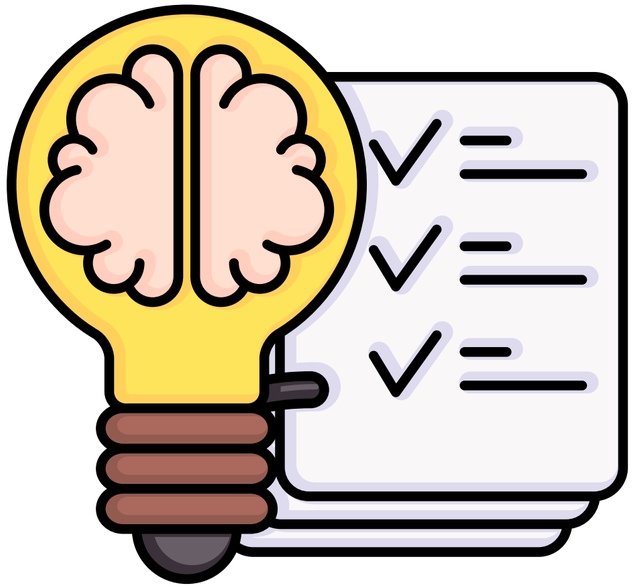
Challenges & Lessons Learned

Challenges Faced:

- **Class imbalance – Fraud cases were only 0.17% of transactions.**
- **Balancing precision & recall – Avoiding too many false positives while detecting fraud.**
- **Computational cost – XGBoost performed better but required more processing power.**
- **Model explainability – Hard to interpret why the model flags transactions.**

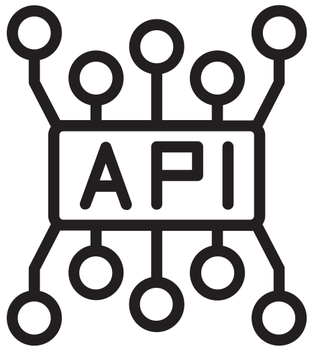
Lessons Learned:

- **SMOTE helped balance data and improved fraud detection.**
- **XGBoost is more effective but needs tuning for real-world applications.**
- **Feature selection matters – Right features improve model accuracy.**
- **Fraud detection must evolve – Models need continuous updates to stay current..**



Future Work & Next Steps

- Real-time fraud detection – Deploy model for live transaction monitoring.
- Explainability & transparency – Use SHAP values to interpret fraud predictions.
- Adaptive learning – Continuously update the model with new fraud patterns.
- Scalability – Optimize for large-scale financial systems.
- API integration – Connect the model with banking and e-commerce platforms.



THANK YOU!