Fraud Analytics Study:

### **Detecting Credit Card Fraud Transactions Using Machine Learning and Anomaly Detections**

Team Fraud

Aditi Dora, Alexis Tran, Cecilia Siek, Euniee Jiang, James Shao, Morgan Budidjaja, Stanley Toh

University of Southern California

DSO 585: Data Driven Consulting

Dr. Austin Pollok

May 12, 2025

## 

## 1. Executive Summary

This project focuses on developing a credit card fraud detection system using machine learning, trained on an existing credit card transaction dataset. Leveraging a hyperparameter-tuned XGBoost model, our approach balances the trade-off between detecting fraudulent transactions and correctly identifying legitimate purchases, while maintaining high classification accuracy for both classes.

A key limitation of the dataset is the absence of actual loss cost values from the credit card company, which constrains our ability to quantify the financial impact of fraud directly. To address this, we modeled three representative business scenarios by making informed assumptions about potential loss costs. These scenarios enabled us to fine-tune the model’s decision threshold to reflect varying levels of fraud tolerance and financial risk across different business contexts.

This strategy allowed us to tailor fraud detection performance to specific operational needs, ensuring the model aligns with real-world business outcomes. Ultimately, our solution aims to enhance fraud prevention capabilities for our target e-commerce company, minimizing financial losses while preserving the customer experience.

## 2. Business Needs and Decisions

Credit card fraud is a major financial concern in the American economy, especially for e-commerce merchants. In “2025 Credit Card Fraud Report and Statistics”, 62 million Americans had experienced at least one case of credit card fraud last year. Additionally, 63% of U.S. credit card holders have been victimized by fraud, and 51% have experienced fraud multiple times, 21% of victims had experienced recurring fraudulent charges. The median expenditure per fraudulent charge also jumped 26 percent in the previous two years, with unauthorized purchases exceeding $6.2 billion annually.

However, only 8% of fraudulent charges involved stolen or lost physical credit cards; the rest of fraudulent activities came from accessing personal data and account information remotely. E-commerce, at the core of the online payment ecosystem, suffered mostly from fraudulent actions. To understand how fraud damages e-commerce merchants, we need to understand the credit card network first.

Credit card providers like Visa and Mastercard shift liability from the credit card providers to merchants, so fraudulent transaction costs will be shifted onto merchants. When a fraudulent transaction occurs (e.g., a stolen card is used), merchants often bear the cost due to chargebacks. The 3% transaction fee charged by every credit card transaction helped offset fraud-related expenses like chargeback reimbursements and fraud prevention efforts for the credit card provider, but not the merchant. Chargeback fees ($20–$100 per dispute) further penalize merchants. This creates an incentive for merchants to adopt fraud-prevention tools like EMV chip readers and CVV verification. Credit card providers used a portion of the 3% fee to invest in fraud detection systems, like AI-based fraud monitoring and transaction verification. These systems help reduce fraud but indirectly pass the cost onto merchants via fees. Some industries (e.g., online stores, travel, adult services) face higher transaction fees (3.5%–5%) due to higher fraud risks. In summary, whenever fraud happens, e-commerce merchants have to either absorb fraud costs, or invest in fraud prevention, or pass the cost to customers through higher prices(which loses product competitiveness).In real life, merchants in the U.S. and Canada incur an average cost of $3.00 for every $1 of fraud, according to “LexisNexis® True Cost of Fraud™ Study: Ecommerce and Retail Report – U.S. and Canada Edition”.

We decided to use a machine learning model to help identify fraud transactions for our e-commerce merchant. By incorporating different machine learning models, and finding the best performing model on the e-commerce dataset, we aim to help our e-commerce merchant, by identifying fraudulent transactions in real-time before damage occurs, ultimately reducing chargebacks, protecting revenue, and enhancing customer trust.

### **Business Problem**

**How can we build a machine learning model that accurately detects fraudulent e-commerce transactions in real time, while minimizing customer friction and financial loss?**

Our goal is to create a fraud detection model that prioritizes speed, accuracy, and minimal disruption to the user experience—key pillars for online platforms. The model must identify rare fraudulent behavior from a vast pool of legitimate transactions, in a highly imbalanced setting.

### **Business Context & Generalization**

Although our dataset is drawn from an e-commerce context, the business need for accurate fraud detection extends across the financial services ecosystem. To understand how priorities shift by sector, we also examine how the model could adapt to:

1. **Enterprise Banks (e.g., JPMorgan Chase)** – Emphasize precision and real-time security
2. **Mid-sized Regional Banks** – Require cost-efficient solutions with moderate accuracy
3. **E-Commerce Platforms (our focus)** – Need lightweight, fast-response models with low false positives

## 3. Data Overview & Preprocessing

## **3.1 Dataset Description**

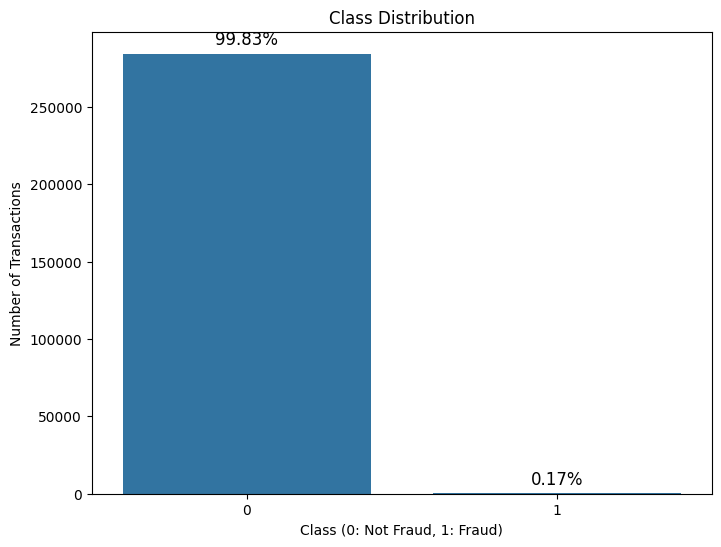
Our project uses a real-world credit card transaction dataset collected by **Worldline** and the **Université Libre de Bruxelles (ULB)**. It contains:

* **Transactions:** 284,807
* **Frauds:** 492 (0.172% of total)
* **Time Period:** Two consecutive days in September 2013
* **Region:** European cardholders
* **Features:** 30 total
  + 28 principal components (V1–V28), derived from PCA for confidentiality
  + 'Time': Seconds elapsed since the first transaction
  + 'Amount': Transaction value
  + 'Class': Target label (0 = non-fraud, 1 = fraud)

This is a **highly imbalanced dataset**, with frauds making up less than 0.2% of all records—typical of real-world fraud detection problems.

### **3.2 Data Challenges**

* **Extreme Class Imbalance**: With only 0.172% of cases labeled as fraud, a naive classifier predicting all transactions as non-fraud would still achieve 99.8% accuracy—highlighting the need for better evaluation metrics like **AUC-PR (Area Under the Precision-Recall Curve)**.
* **Anonymized Features**: PCA transformation conceals original feature meanings, which limits interpretability but helps prevent reverse-engineering of sensitive information.



## 4. Methodology

### **4.1 Metrics Considered**

Given the nature of fraud detection, we focused primarily on **precision** and **recall** as key evaluation metrics. Precision measures the proportion of detected fraud cases that were truly fraudulent, while recall measures the proportion of actual fraud cases that the model successfully identified. Both metrics are critical: precision helps minimize false positives and reduce unnecessary customer inconvenience, while recall ensures we capture as many fraudulent transactions as possible.

By default, classification models use a **prediction threshold of 0.5**, meaning transactions with a predicted fraud probability above 50% are classified as fraud. Adjusting this threshold allows us to prioritize different outcomes: **increasing** the threshold makes the model more strict, flagging only highly confident fraud cases and improving precision, while **lowering** the threshold makes the model more lenient, flagging more potential fraud but at the risk of more false positives. Depending on business priorities — whether minimizing missed fraud or minimizing customer inconvenience — we can tune the threshold accordingly. In selecting our final model, we carefully considered these trade-offs to balance operational efficiency and customer experience.

### **4.2 Model Experimentation**

To identify the best model for detecting credit card fraud in our dataset, we explored a range of machine learning models. Our goal was to choose a model that could effectively balance the trade-off between detecting fraudulent transactions and correctly classifying legitimate ones. The following models were considered:

1. **Logistic Regression**: As a baseline model, logistic regression is simple and interpretable. However, it struggles with handling imbalanced data and non-linear decision boundaries, which are crucial in fraud detection tasks where fraudulent transactions are significantly less frequent than legitimate ones.
2. **Decision Tree**: Decision trees are a non-ensemble, interpretable model. However, they are prone to overfitting, especially when the data is noisy or imbalanced. This model's performance in handling class imbalance, such as distinguishing between fraudulent and legitimate transactions, was suboptimal.
3. **Random Forest**: Random Forest, an ensemble learning technique, improves on logistic regression by building multiple decision trees and averaging their outputs to reduce overfitting. While it performs well in many scenarios, it requires significant computational power and does not naturally provide feature importance in the same intuitive way as XGBoost.
4. **Gradient Boosting**: Gradient Boosting is another ensemble method that builds trees sequentially. While it offers strong performance, it can be slower compared to XGBoost due to its computational complexity and may not scale as efficiently on larger datasets.
5. **XGBoost**: XGBoost (Extreme Gradient Boosting) is an optimized version of the gradient boosting algorithm. It builds decision trees sequentially, with each tree correcting the errors of the previous one. XGBoost is known for its high performance, speed, and ability to handle large datasets. It also includes features like regularization to prevent overfitting, and it is particularly effective for handling imbalanced datasets, making it a strong candidate for fraud detection tasks.

The following table compares the performance of each model using **Precision-Recall (PR) Score** and **Receiver Operating Characteristic (ROC) Score**:

| **Model** | **PR Score** | **ROC Score** |
| --- | --- | --- |
| Logistic Regression | 0.61 | 0.93 |
| Decision Tree | 0.55 | 0.87 |
| Random Forest | 0.87 | 0.96 |
| Gradient Boosting | 0.64 | 0.97 |
| XGBoost | 0.88 | 0.98 |

To evaluate model performance for fraud detection, we initially analyzed the Receiver Operating Characteristic (ROC) curve and its Area Under the Curve (AUC) metric. The ROC curve plots the true positive rate against the false positive rate at various threshold settings, and the AUC summarizes the model’s ability to distinguish between classes across all thresholds. A higher AUC typically reflects better model performance, with a value of 1.0 indicating perfect classification.

In our models, we achieved strong ROC-AUC results across the board, with XGBoost achieving the highest score of **0.98**, followed closely by Gradient Boosting (**0.97**) and Random Forest (**0.96**). These high ROC-AUC values suggested high accuracy and effective separation between fraud and non-fraud cases.

However, because fraud detection presents a **heavy class imbalance** — with legitimate transactions vastly outnumbering fraudulent ones — ROC-AUC can give an overly optimistic view of model performance. Specifically, a model can achieve a high ROC-AUC simply by correctly classifying the abundant non-fraud cases, even if it performs poorly on the rare fraud cases.

To address this, we turned to the Precision-Recall (PR) curve, which focuses more directly on the minority class (fraud). Precision measures how many detected fraud cases were actually fraud, while recall measures how many actual fraud cases were correctly detected. By focusing on precision and recall, we obtained a more meaningful evaluation of fraud detection performance.

Based on PR curve results, XGBoost again stood out, achieving the highest PR score of **0.88**, indicating strong precision and recall in detecting fraudulent transactions. Therefore, while ROC-AUC validated the overall model quality, our final model selection emphasized **PR curve performance**, leading us to select XGBoost as the best model for deployment.

### **4.3 XGBoost Model**

XGBoost was selected for the following key reasons:

1. **Handling Imbalanced Data**: One of the biggest challenges in fraud detection is the severe class imbalance between fraudulent and legitimate transactions. XGBoost provides effective handling of imbalanced datasets through its built-in support for weighted loss functions and regularization, which helps prevent overfitting to the majority class (legitimate transactions).
2. **Performance (F1, AUC)**: XGBoost consistently outperforms other models in terms of **F1 score** and **Area Under the Curve (AUC)**. These metrics are crucial in fraud detection, where both **false positives** (legitimate transactions incorrectly flagged as fraud) and **false negatives** (fraudulent transactions missed) can have significant financial consequences. XGBoost’s ability to optimize for both **precision** and **recall** made it an ideal choice for this task, ensuring a balance between detecting fraud and minimizing false alarms
3. **Interpretability via SHAP**: Understanding why the model flags a transaction as fraudulent is critical for business stakeholders. XGBoost offers interpretability through **SHAP (Shapley Additive Explanations)** values, which provide insight into which features are most influential in predicting fraud. This interpretability allows us to explain model decisions to non-technical users and ensures that the fraud detection system can be trusted, adjusted, and fine-tuned as needed.
4. **Evaluation: PR Curve > ROC for Imbalance**: Given the highly imbalanced nature of the dataset, we prioritized the Precision-Recall (PR) curve over the Receiver Operating Characteristic (ROC) curve. While the ROC curve provides useful information, it can be misleading when dealing with imbalanced datasets, as it tends to present overly optimistic results for models biased toward the majority class. The **PR curve**, in contrast, offers a clearer and more focused evaluation of model performance with respect to the minority class (fraudulent transactions), which is of primary importance in fraud detection.

By selecting XGBoost, we ensured that our model not only delivered high performance in detecting fraud but also provided valuable insights into the factors driving model decisions. This combination of high performance, interpretability, and robustness makes XGBoost a powerful tool for real-world fraud detection applications.

### **4.4 Modeling Approach**

In this section, we outline the specific steps we took to train, fine-tune, and evaluate the XGBoost model for fraud detection. This approach was designed to optimize the model's ability to accurately classify fraudulent transactions while minimizing false positives and false negatives. The following steps were followed:

1. **Baseline Model**: We began by training the XGBoost model on the raw dataset to establish a baseline performance. The initial model was trained without any tuning or resampling techniques, which allowed us to assess how well the model performed "out-of-the-box". Key evaluation metrics, such as **accuracy**, **precision**, **recall**, **F1 score**, and **AUC**, were recorded to assess the model’s basic performance and to set a starting point for improvement.
2. **Data Preprocessing**: Given the imbalanced nature of the dataset, we applied several preprocessing techniques to ensure the model could handle the skewed class distribution effectively. This included:

* **Handling Missing Data**: We used imputation techniques to fill missing values where necessary.
* **Feature Scaling**: Although XGBoost is generally robust to scaling, we ensured that features with large variance were scaled to optimize performance.
* **Class Imbalance Handling**: We explored applying the Synthetic Minority Over-sampling Technique (SMOTE) to generate synthetic samples for fraudulent transactions and experimented with weighted loss functions to give more importance to fraud cases. However, these approaches did not significantly improve performance, as the models were already effectively capturing fraud patterns without additional resampling or reweighting.

1. **Hyperparameter Tuning**: After training the baseline model, we proceeded with hyperparameter tuning to optimize the model’s performance. We used grid search in combination with cross-validation to search for the best combination of hyperparameters. The key hyperparameters that were tuned include:

* **Learning rate** : To control the step size at each iteration.
* **Maximum depth of trees**: To determine the depth of each tree.
* **Number of estimators (trees)**: To control the number of decision trees in the ensemble.
* **Subsample ratio**: To determine the fraction of data points to be used for each boosting round.

1. **5-Fold Cross-Validation**: To ensure the robustness of the model, we used 5-fold cross-validation to evaluate the model’s performance across different subsets of the training data. This process helps to assess the model’s generalizability and ensure it isn’t overfitting to specific data points. Cross-validation also helped provide more reliable performance estimates, ensuring the model would perform well on unseen data.performance estimates, ensuring the model would perform well on unseen data.
2. **Evaluation Metrics**: Given the class imbalance in the dataset, we prioritized on the following evaluation metrics:

* **Precision, Recall, and F1 Score**: These metrics are particularly critical in fraud detection, where we aim to minimize both false positives and false negatives. The **F1 score** is especially important as it balances both precision and recall.
* **AUC (Area Under the Curve)**: The AUC was used to assess the model's ability to distinguish between fraudulent and legitimate transactions. A higher AUC indicates a model that can more effectively separate the two classes..
* **Precision-Recall Curve**: We prioritized the **Precision-Recall curve** over the ROC curve due to the class imbalance. The PR curve provides a clearer picture of how the model performs with respect to the minority class (fraudulent transactions), which is of primary interest in fraud detection.

1. **Model Refinement**: After evaluating the initial model’s performance, we fine-tuned it further by adjusting the **decision threshold**. By lowering or raising the threshold, we could adjust the trade-off between false positives and false negatives. For example, in scenarios where it was more critical to detect fraud at all costs (even at the cost of increasing false positives), a lower threshold was applied. This threshold optimization was integral in aligning the model with the specific needs of the e-commerce merchant and optimizing business outcomes.
2. **Final Model**: The final XGBoost model was trained using the best hyperparameters and evaluated on the test set. This final model demonstrated a strong balance between **precision** and **recall**, making it well-suited for fraud detection in real-time environments. The model was also capable of being explained through **SHAP values**, making it easier for business stakeholders to understand how and why transactions were flagged as fraudulent.

By following this approach, we ensured that the model was not only highly accurate but also optimized for the specific challenges posed by credit card fraud detection, such as imbalanced data, real-time processing needs, and the requirement for model explainability.

## 5. Application

### **5.1 Threshold Optimization**

In real-world fraud detection, a single model is not sufficient for all industries. Different businesses experience distinct cost implications from false positives (FP) and false negatives (FN), so we implemented threshold tuning to align model predictions with specific business goals. Instead of using a generic 0.5 threshold, we explored three tailored options:

* **Fraud-Minimization Model (Threshold = 0.3)**
  + Prioritized for institutions like JPMorgan Chase, where missing a fraudulent transaction can result in losses up to $250 per incident.
  + These businesses are risk-averse and can tolerate more false alerts in exchange for stronger fraud detection.
  + Lowering the threshold to 0.3 increases recall and captures more fraud cases, aligning with their high cost of FN.
* **Balanced Cost Model (Threshold = 0.5)**
  + Applied to regional banks such as Citizens or Bank RI.
  + These institutions operate with tighter margins and must balance fraud detection with customer satisfaction.
  + False negatives are costly, but so are false positives due to customer loss. A 0.5 threshold provides a middle ground.
* **Customer-Centric Model (Threshold = 0.7)**
  + Designed for e-commerce platforms and subscription services like Adobe or Etsy.
  + These businesses value customer experience and retention, where false positives (wrongly flagged transactions) lead to churn.
  + A higher threshold reduces unnecessary blocks, tolerating some fraud in favor of smoother user experiences.

We plotted **Total Cost vs. Threshold** using cost matrices for each business type, incorporating industry-specific costs:

| Business Model | Cost (FP) | Cost (FN) | Optimal Threshold | Total Cost Impact |
| --- | --- | --- | --- | --- |
| Fraud-minimization | $5 | $1000 | 0.3 | Missing fraud is expensive |
| Balanced | $50 | $500 | 0.5 | Balanced trade-off |
| Customer-centric | $100 | $200 | 0.7 | Best for customer trust |

This threshold optimization strategy demonstrates how we adapt our model to minimize real financial losses, not just improve accuracy.

**5.2 Cost Sensitive Prediction Threshold Optimization (Transaction Amount Based)**

Before implementing cost-sensitive learning during model training, we first explored **cost-sensitive threshold optimization** to minimize financial loss. In this dynamic transaction amount based approach, the model remained unchanged, and we adjusted the **prediction threshold** after training based on transaction amounts to reduce total cost. Using the validation set, we identified an optimal threshold of **0.71**, achieving a minimum total cost of **$66,360**. This method allowed us to assess whether tuning the threshold alone — without modifying the model — could effectively control fraud-related costs.

To test the stability of this threshold, we applied it to a separate test set. Using the 0.71 threshold, the total cost on the test set was **$62,535**. However, recalculating the optimal threshold for the test set revealed a new best threshold of **0.01**, leading to a lower minimum cost of **$47,142**. This significant shift demonstrated that a fixed threshold may not generalize well to new data, reinforcing the need for dynamic threshold tuning or more integrated cost-sensitive learning approaches in fraud detection systems.

### **5.3 Cost-Sensitive Learning**

While threshold tuning addresses *when* to classify a transaction as fraud, cost-sensitive learning adjusts *how* the model learns from mistakes by incorporating the actual financial consequences into the training process.

We implemented a custom losses the equal-weighted loss used in standard classification models with a dollar-weighted approach:

* In **enterprise banking**, a missed fraud (FN) incurs ~$175-$250 in direct loss, while a false positive (FP) may only cost ~$35-$40.
* In **regional banking**, the FN cost is adjusted with a 3x multiplier due to lower tolerance and fraud absorption capacity.
* In **e-commerce**, FP costs are often higher due to churn, with some platforms valuing FP penalties up to $100 per incident.

Our model minimizes a weighted cost function:

**Loss** = (Cost of False Negative \* Count of False Negative) + (Cost of False Positive \* Cost of False Positive)

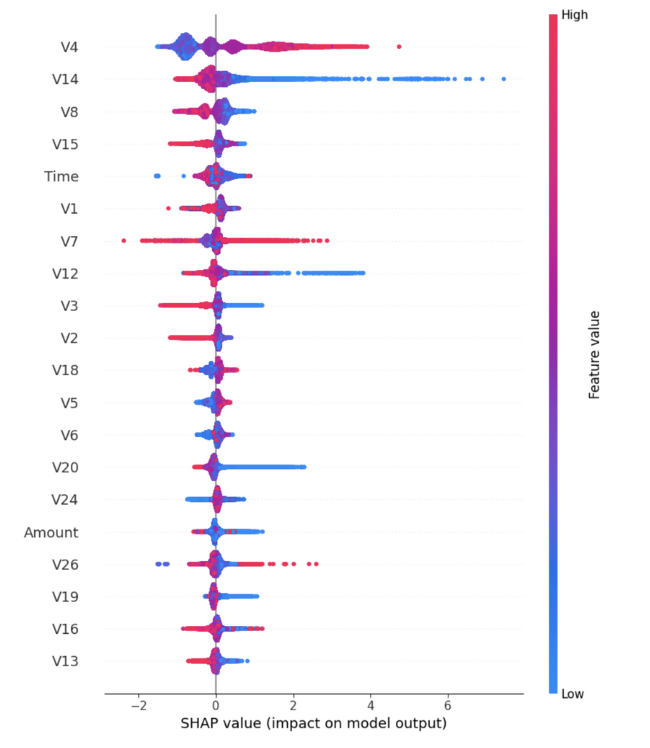
This learning strategy encourages the model to focus on minimizing business-relevant losses, not just classification errors.

By integrating this approach with XGBoost, we achieved a better alignment between fraud detection and business priorities. We also observed improved model performance under the Total Cost metric, particularly when thresholds were tuned jointly with these business-specific costs.

This method supports scalable deployment across multiple industries, each with tailored cost functions and decision thresholds.

### **5.4 SHAP Interpretability**

To move beyond traditional feature importance, we used **SHAP** (SHapley Additive Explanations) to explain why individual transactions were flagged as fraudulent. SHAP assigns each feature a value that quantifies its impact — positive or negative — on a specific prediction. Traditional feature importance only shows which variables are important on average, but SHAP goes further by:

* Explaining how each feature value pushes a specific prediction toward fraud or non-fraud.
* Capturing feature interactions and the direction of effect (positive or negative).
* Enhancing model transparency, enabling better communication with non-technical stakeholders.

### Key Insights from SHAP Analysis:

* V4 is the most influential feature in shifting predictions — potentially indicating a strong fraud signal.
* V14 and V8 also contribute significantly to fraud predictions.
* Time, V1, and V7 show moderate influence.
* Transaction Amount has relatively low predictive value, consistent with prior findings that large amounts don’t always indicate fraud.

Performance Trade-Off Full Model vs SHAP-Reduced Model: To evaluate model robustness, we compared the full model against a SHAP-based model using only the top 10 features

After identifying the top 10 most impactful features using SHAP, we retrained the model with only these inputs:

* The simplified model maintained high ROC AUC (0.9734) and improved fraud recall, catching more fraudulent cases.
* Precision dropped, increasing false positives — a trade-off that may be acceptable in real-time fraud systems where recall is prioritized.

This transaction was flagged because features like V4 and V14 showed patterns that our model has learned to associate with fraud. SHAP shows exactly how much each feature contributed, enabling transparent, case-by-case explanations. This level of interpretability allows fraud analysts to validate predictions and supports the development of early warning tools based on top-contributing features.

### 

### Impact of PCA Features:

* SHAP still helps us identify which components are most impactful (e.g., V4).
* However, PCA limits interpretability — we cannot directly tie these to real-world actions like device type or merchant category.
* Dropping less important components could still risk losing useful variance.

## 6. Results

In the e-commerce sector, the business priority shifts toward protecting the customer experience. False positives in this industry can cost over $100 per incident, largely due to downstream effects like customer churn, cart abandonment, and reputational damage. Therefore, we recommend tuning thresholds conservatively and implementing a tiered alert review system, where only medium- to high-risk transactions are escalated for manual review. This approach helps e-commerce platforms maintain customer trust, reduce churn, and ensure that fraud detection aligns with business growth and user satisfaction goals.

## Conclusion & Next Steps

Our study demonstrates the value of aligning machine learning models with real-world business costs to enhance fraud detection. By combining XGBoost with cost-sensitive learning and threshold tuning, we developed a flexible solution that adapts to varying risk profiles across industries. This approach prioritizes financial outcomes over generic accuracy metrics, resulting in more actionable, business-aligned decisions. Additionally, the use of SHAP values enhances model transparency, allowing for greater trust and interpretability by stakeholders. Overall, our solution empowers businesses to minimize fraud-related losses while preserving customer experience and operational efficiency.

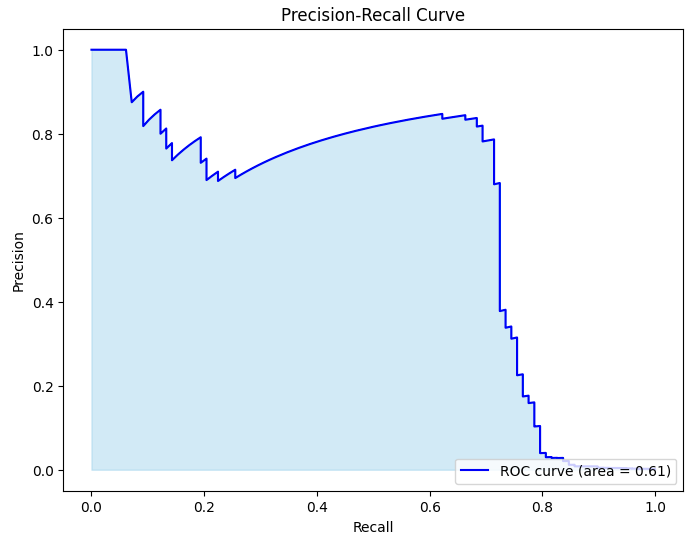
To move from prototype to production, the next phase involves integrating our model into a live transaction environment with access to streaming or batch transaction data. We will pilot the solution with a selected business partner (e.g., an e-commerce merchant) to validate cost performance in a real-world setting. Model retraining pipelines should be established to ensure continuous adaptation to evolving fraud patterns, with thresholds recalibrated quarterly. Additionally, customizing cost matrices for individual clients will further improve alignment with their risk tolerance and business goals. Finally, building dashboards that visualize model outputs, SHAP explanations, and business impact metrics will support informed decision-making by fraud and risk management teams.

## **Appendix**

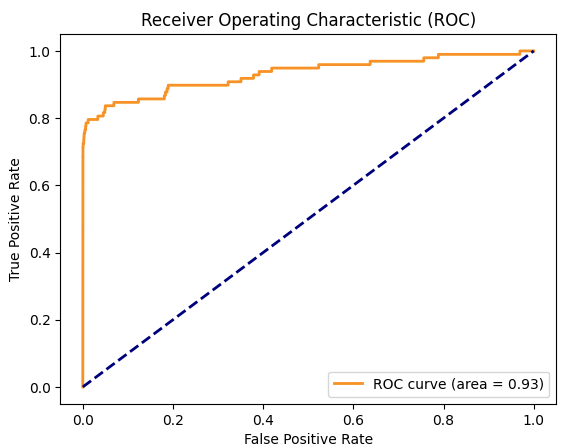
1. Dataset: [Kaggle - Credit Card Fraud](https://www.kaggle.com/datasets/mlg-ulb/creditcardfraud)
2. Completed Code Files:
3. [Logistic Regression](https://colab.research.google.com/drive/1BCwH_K2CCHoutycZya4WLSjXqK6yNlSI?usp=sharing)
4. [Decision Tree](https://colab.research.google.com/drive/1OEjZ_JebM5OYO94UvKlv2MFHoBIYQ4bg?usp=sharing)
5. [Random Forest](https://colab.research.google.com/drive/1RHDTMeIDMawlRL16pgiexF0nippf74qf?usp=sharing)
6. [XGBoost](https://colab.research.google.com/drive/1XXKsOsX3-pP1t4qkfBKSzzGx-tj9hh2g?usp=sharing)
7. [Cost Sensitive Prediction Threshold Optimization (Dynamic Transaction Amount Based)](https://colab.research.google.com/drive/1NBc1rEaQzAUtmOrBPaeZPDuO2DPeLBxU?usp=sharing)
8. [PR Curve and ROC Curves](https://colab.research.google.com/drive/1OEjZ_JebM5OYO94UvKlv2MFHoBIYQ4bg?usp=sharing)
9. LexisNexis® True Cost of Fraud™ Study: Ecommerce and Retail Report – U.S. and Canada Edition - <https://risk.lexisnexis.com/insights-resources/research/us-ca-true-cost-of-fraud-study>
10. When false positives spiked, company abandoned fraud prevention tools - <https://www.jpmorgan.com/insights/payments/analytics-and-insights/cnp-fraud-prevention-combat-chargebacks?utm_source=chatgpt.com>
11. How AI Transformed Financial Fraud Detection: A Case Study of JP Morgan Chase - <https://medium.com/%40jeyadev_needhi/how-ai-transformed-financial-fraud-detection-a-case-study-of-jp-morgan-chase-f92bbb0707bb>
12. Midwestern Top 100 Regional Bank Reduces False Negatives, Financial Losses, Operating Expenses and Reputational Risk With OASIS - <https://blog.argodata.com/midwestern-top-100-regional-bank-reduces-false-negatives-financial-losses-operating-expenses-and-reputational-risk-with-oasis?utm_source=chatgpt.com>
13. Mobile Fraud Costs Hit $4.61 Per Dollar for North American E-commerce Merchants - <https://mobileidworld.com/mobile-fraud-costs-hit-4-61-per-dollar-for-north-american-e-commerce-merchants/?utm_source=chatgpt.com>
14. Optimal Cost Matrix Example

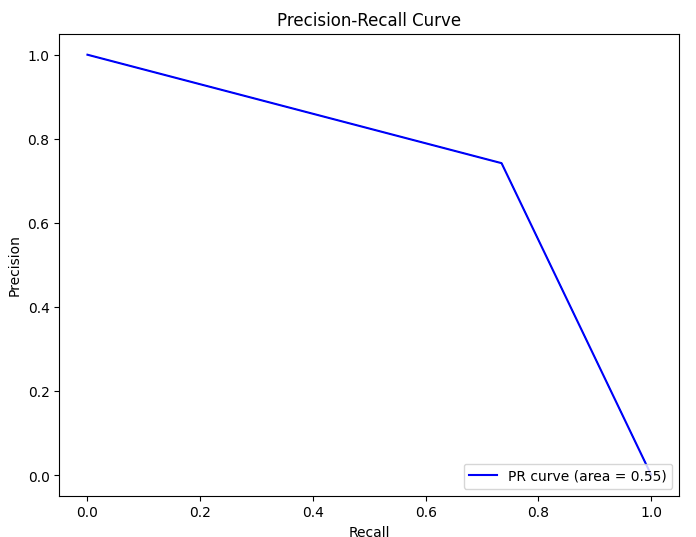
| Use Case | TP Cost | TN Cost | FP Cost | FN Cost |
| --- | --- | --- | --- | --- |
| JPMorgan Chase | $0 | $0 | $35-40 | $175-350 |
| Regional Bank | $0 | $0 | $25-30 | $525-750 |
| E-commerce Merchant | $0 | $0 | $45-60 | $15-40 + ops |

1. Logistic Regression Precision-Recall Curve

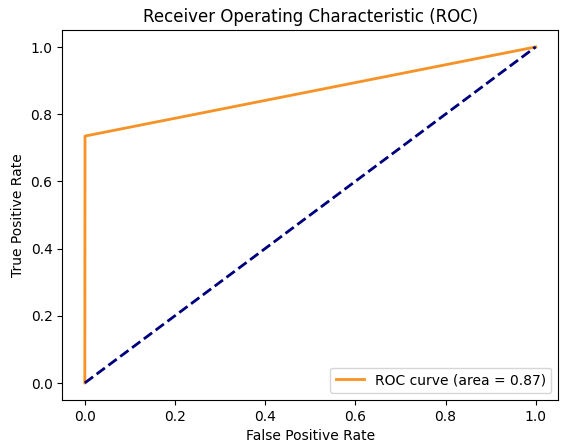


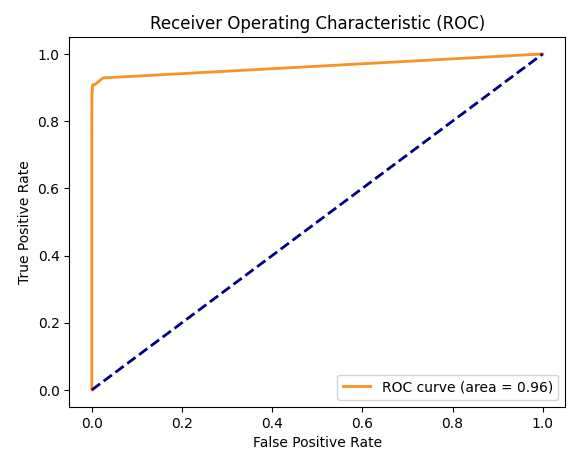
Logistic Regression Receiver Operating Characteristic Curve

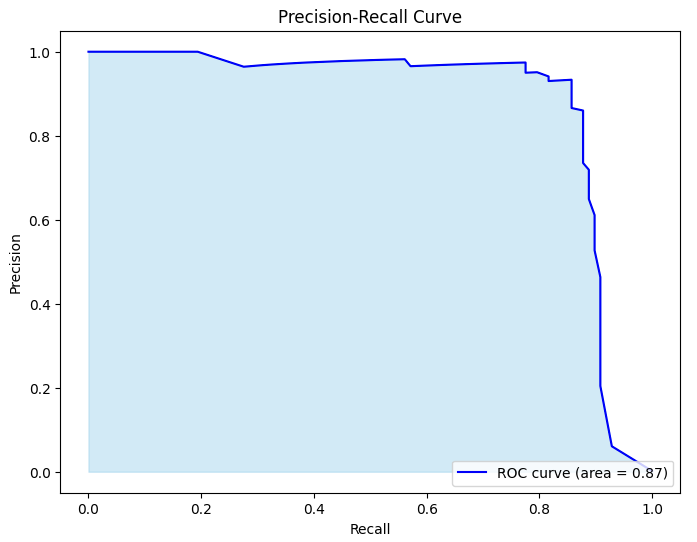


1. Decision Tree Precision-Recall Curve

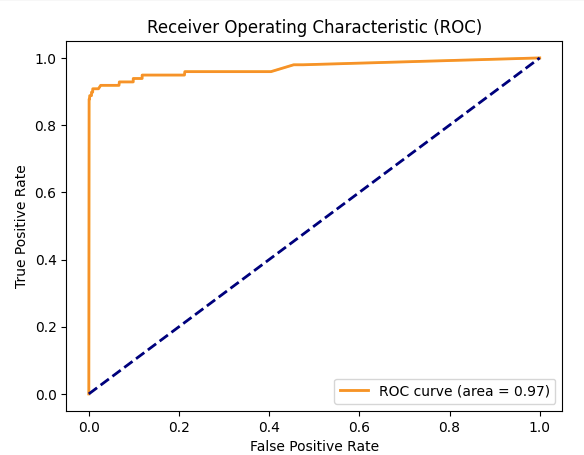
Decision Tree Receiver Operating Characteristic Curve

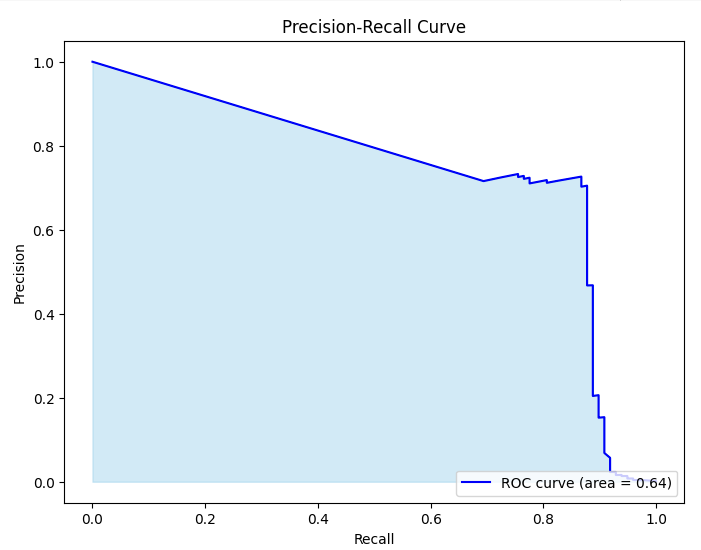


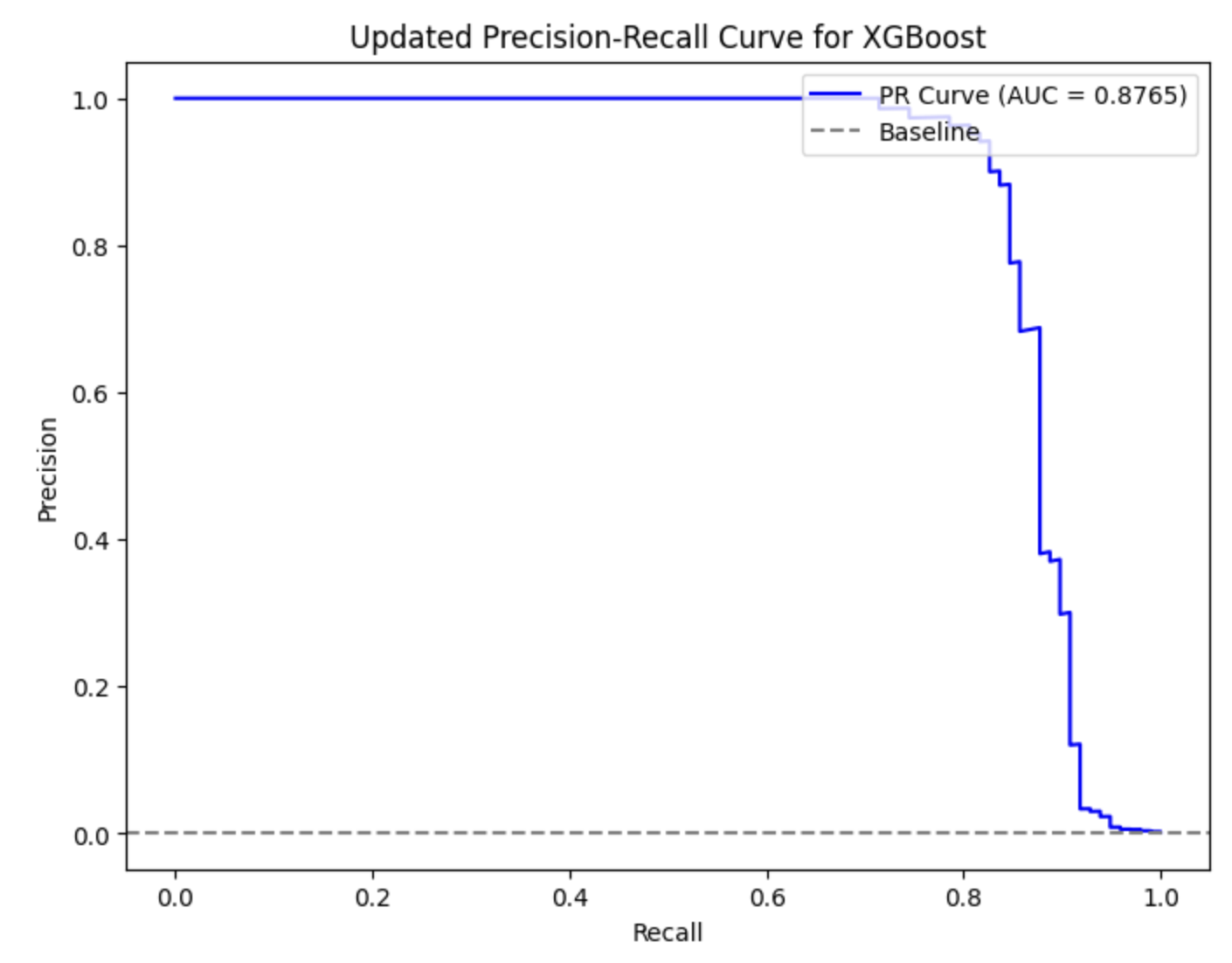
1. Random Forest Precision-Recall Curve

Random Forest Receiver Operating Characteristic Curve

1. Gradient Boosting Precision-Recall Curve



Gradient Boosting Receiver Operating Characteristic Curve

1. XGBoost Precision-Recall Curve

XGBoost Receiver Operating Characteristic Curve

