# Group 34 Progress Report: Fraud Detection in E-Commerce Transactions

**Viransh Shah, Ellen Xiong**
{shahv47,xionge1}@mcmaster.ca

## 1 Introduction

Financial fraud represents one of the most persistent and costly challenges facing e-commerce platforms today. With fraudulent transactions costing billions of dollars in losses annually, it also risks exposing customers to identity theft and financial harm. Our project addresses this critical problem by developing a machine learning-based fraud detection system for e-commerce transactions. The task is formulated as a binary classification problem where we predict whether a given transaction is fraudulent or legitimate based on multiple features.

The challenge of fraud detection is multifaceted. Fraudulent transactions exhibit severe class imbalance, representing only approximately 5% of transactions in our dataset and often less than 1% in real-world scenarios. Fraudulent behavior emerges from complex feature interactions rather than single variable anomalies, requiring models that can capture complex relationships. It is also important to consider that fraud patterns constantly evolve as adversaries adapt their tactics, necessitating robust models that can generalize across different contexts. Our work aims to develop an effective fraud detection model that addresses these challenges, while maintaining high precision to minimize false positives that could frustrate legitimate customers.

## 2 Related Work

Fraud detection in financial transactions has been extensively studied using various machine learning approaches. (**?**) provides a comprehensive review of machine learning techniques for credit card fraud detection, highlighting the effectiveness of ensemble methods and deep learning approaches in handling imbalanced datasets. (**?**) examined fraud detection in the context of big data analytics, emphasizing the importance of feature engineering and real-time processing capabilities.

Random forests and gradient boosting machines have shown particular success in fraud detection tasks due to their ability to handle non-linear relationships and feature interactions (**?**). (**?**) demonstrates that deep neural networks can automatically learn relevant features from transaction data, potentially outperforming traditional feature engineering approaches. Recent work by (**?**) specifically addresses the class imbalance problem through advanced sampling techniques and cost-sensitive learning methods, which directly inform our approach to handling the skewed distribution in our dataset.

## 3 Dataset

We utilized the Fraudulent E-Commerce Transactions dataset from Kaggle, created by Shriyash Jagtap. This synthetic dataset simulates realistic e-commerce transaction patterns with engineered fraud scenarios. The dataset contains 1,472,952 transactions in version 1 and 23,634 transactions in version 2, with 16 original columns per transaction. Approximately 5% of transactions are labeled as fraudulent, reflecting realistic fraud rates while providing sufficient positive examples for model training.

### 3.1 Dataset Features

The dataset includes both numerical and categorical features that capture different aspects of transaction behavior. Numerical features include Transaction Amount, Quantity, Customer Age, Account Age Days, and Transaction Hour. Categorical features comprise Payment Method (credit card, PayPal, debit card, etc.), Product Category (electronics, clothing, home goods, etc.), Customer Location, Device Used (mobile, desktop, tablet), and address information (IP Address, Shipping Address, Billing Address). Each transaction has a unique Transaction ID and Customer ID, along with a Transaction Date timestamp and the binary target label Is Fraud-

ulent.

### 3.2 Data Preprocessing

Our preprocessing approach addressed both data quality and modeling requirements. For numerical features, we applied standardization to ensure all features contribute equally to distance-based calculations. We extracted temporal features using the Transaction Date feature, which created the features Day of Week, Month, and Is Weekend. Categorical variables were encoded using one-hot encoding for low-cardinality features (Payment Method, Product Category, Device Used) and target encoding for high-cardinality features (Customer Location).

We engineered a new feature called Total Customer Transactions that counts the number of transactions per Customer ID, as repeat transaction patterns may indicate legitimate behavior while first-time transactions might warrant additional scrutiny. Address matching features were created to flag mismatches between shipping and billing addresses, as such discrepancies often signal fraud. We handled missing values through median for numerical features and mode for categorical features, though the synthetic dataset contained minimal missing data.

To address the class imbalance problem, we implemented Synthetic Minority Over-sampling Technique (SMOTE, (**?**)) to generate synthetic fraudulent examples in the training set, balancing the class distribution to improve model sensitivity to fraud patterns without simply duplicating existing minority class examples.

## 4 Features

Our final feature set consisted of 13 numerical features and 4 categorical features (expanded to multiple binary features through one-hot encoding). The numerical features included the original Transaction Amount, Quantity, Customer Age, Account Age Days, and Transaction Hour, as well as engineered features: Total Customer Transactions, Address Mismatch, Day of Week, Month, Is Weekend, New Account, Transaction Amount Ratio, and High Value Transaction. After one-hot encoding of the categorical features Payment Method, Product Category, and Device Used, the total feature dimensionality resulted in 23 features.

Feature importance analysis from the Random Forest model revealed that Customer Location (target-encoded) was by far the most predictive feature, accounting for 24-90% of feature importance depending on the dataset version. Transaction Hour showed consistent importance (2.5-15%), which suggested temporal patterns in fraudulent behavior. Transaction Amount Ratio and Account Age Days also demonstrated high predictive power, confirming that unusual spending patterns and account age are strong fraud indicators.

Some engineered features showed lower importance. High Value Transaction contributed less than 1% importance, possibly because Transaction Amount Ratio already captures similar information more effectively. Address Mismatch also showed minimal importance, suggesting that in this dataset, fraudsters do not consistently use mismatched addresses. These findings will inform our feature refinements for the final model.

## 5 Implementation

We implemented three models to establish baselines and compare approaches: logistic regression, random forest, and a feedforward neural network.

### 5.1 Training Strategy

Our training strategy was to split the data using stratified sampling; 70% training, 15% validation, and 15% test, while maintaining class distribution across splits (random seed 42). SMOTE was applied only to the training set, which created a balanced 50-50 class distribution. The validation set guided hyperparameter selection and early stopping, while the test set remained completely untouched until final evaluation.

### 5.2 Model Performance Comparison

| Model | Acc. | Prec. | Rec. | F1 | AUC |
|---|---|---|---|---|---|
| *Version 2 Dataset (23,634 rows)* | | | | | |
| Logistic Reg. | 0.961 | 0.578 | 0.934 | 0.714 | 0.991 |
| Random Forest | **0.976** | **0.720** | **0.885** | **0.794** | **0.993** |
| Neural Network | 0.962 | 0.581 | 0.940 | 0.718 | 0.988 |
| *Version 1 Dataset (1,472,952 rows)* | | | | | |
| Logistic Reg. | 0.828 | 0.188 | 0.737 | 0.300 | 0.866 |
| Random Forest | **0.902** | **0.273** | **0.570** | **0.369** | **0.866** |
| Neural Network | 0.901 | 0.277 | 0.603 | 0.380 | 0.868 |

Table 1: Model Performance on Test Set for both Dataset Versions. Best performance in each category shown in bold.

### 5.3 Logistic Regression

Our logistic regression model used L2 regularization with class weights set to 'balanced' to address class imbalance. We performed hyperparameter tuning via 3-fold cross-validation with F1-score as the optimization metric. The best configuration used C=10 for v2 and C=0.1 for v1, with the liblinear solver optimized for binary classification and a log-loss (binary cross-entropy) objective function. Despite being a linear model, logistic regression achieved the highest recall on both datasets, making it a suitable model when minimizing false negatives is critical.

### 5.4 Random Forest

The random forest model demonstrated the best overall performance across both datasets. We used GPU-accelerated RAPIDS cuML Random Forest with 200 estimators and a maximum depth of 20. The model used Gini impurity as the splitting criterion and bootstrap sampling for tree diversity. This ensemble approach effectively captured non-linear feature interactions, as evidenced by its superior F1-scores (0.794 on v2, 0.369 on v1) compared to the linear baseline. The random forest model achieved the best balance between precision and recall, making it our primary model.

### 5.5 Neural Network

Our neural network used a GPU-accelerated PyTorch implementation with a two-hidden-layer architecture (128 and 64 neurons). We employed ReLU activation functions, dropout layers (rate 0.3) for regularization, and Adam optimizer (learning rate 0.001). The network used binary cross-entropy loss with the pos_weight parameter to handle class imbalance. Training included early stopping with a patience of 5 epochs based on validation loss. The neural network achieved competitive performance with a high recall (0.940 on v2, 0.603 on v1) but struggled with precision, likely due to the model's capacity to memorize patterns in the minority class.

## 6 Results and Evaluation

### 6.1 Evaluation Metrics

Given the class imbalance and business context of fraud detection, we evaluated models using multiple metrics beyond simple accuracy. Precision measures the proportion of fraud predictions that are correct, minimizing false positives that frustrate legitimate customers. Recall measures the proportion of actual frauds detected, minimizing false negatives that result in financial losses. F1-score provides a balanced metric combining precision and recall, serving as our primary optimization target. AUC-ROC measures the model's ability to distinguish between classes across all classification thresholds, providing threshold-independent performance assessment.

### 6.2 Performance Analysis

Table **??** presents comprehensive results for both dataset versions. The Random Forest model consistently achieved the best F1-scores: 0.794 on v2 and 0.369 on v1. However, performance degraded significantly for all models on the larger v1 dataset, suggesting that our models struggled to generalize to the increased data complexity and diversity.

On the smaller v2 dataset, all models achieved excellent AUC-ROC scores (>0.98), indicating strong discriminative ability. Logistic Regression and Neural Network showed near-identical performance with high recall (>0.93) but lower precision ( 0.58), which resulted in many false alarms. Random Forest achieved a better balance with 72% precision and 89% recall, making it more practical for deployment.

On the larger v1 dataset, performance dropped substantially. Random Forest maintained the best F1-score (0.369) but precision fell to 27%, meaning only 1 in 4 fraud predictions were correct. Recall dropped to 57%, missing 43% of frauds. This performance degradation suggests that the v1 dataset contains more complex fraud patterns or greater feature variation that our current models cannot adequately capture.

### 6.3 Visualizations

Please refer to Figure **??**, Figure **??**, Figure **??** and Figure **??** below. Additional figures showing feature importance, model comparision and precision recall curve will be submitted alongside the Progress Report.

## 7 Feedback and Plans

### 7.1 TA Feedback

During our TA consultation, we received guidance on addressing our model's limitations:

**Class Imbalance:** While SMOTE improved results, the TA noted that class imbalance is not perfectly fixed. We were advised to try downsam-

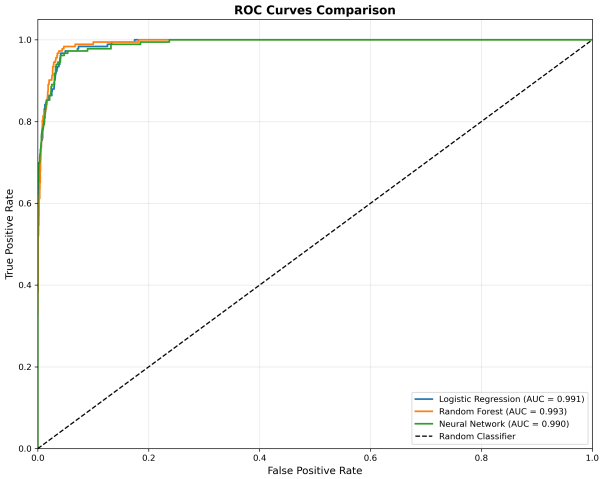


Figure 1: Version 2 (23K rows)

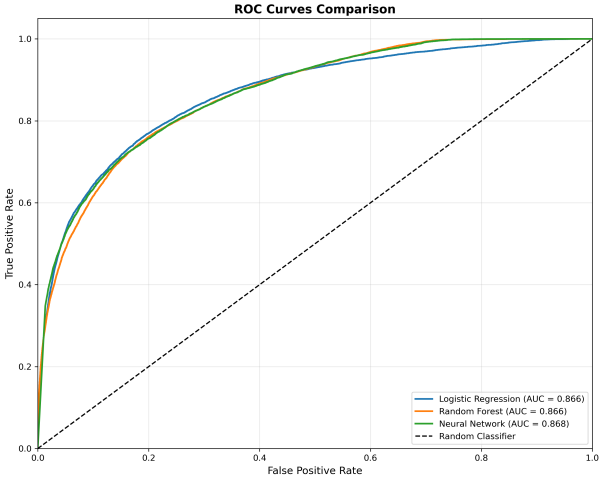


Figure 2: Version 1 (1.47M rows)

pling the majority class (legitimate transactions) or explore alternative oversampling techniques. The current 50-50 balance may be too aggressive; experimenting with 60-40 or 70-30 ratios might improve precision without sacrificing too much recall.

**Model Learning Capacity:** The TA suggested creating a model that can actually learn the data, even if overfitted initially. Our v1 results suggest we may have the opposite problem, underfitting. We should experiment with more complex models (deeper neural networks, more Random Forest trees) and verify if they can achieve higher training accuracy (85%+) before worrying about generalization.

**Cross-Validation:** We were advised to try different data split seeds or implement proper cross-validation. Our current fixed seed (42) may result in a particularly easy or difficult test set. Looking at the K-fold cross-validation would provide more

../../src/SecondMilestone/confusion_matrices_v2.png

Figure 3: Version 2 (23K rows)

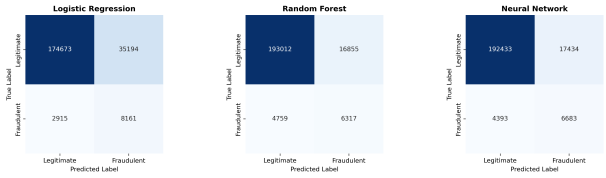


Figure 4: Version 1 (1.47M rows)

robust performance estimates and reveal whether our results are consistent or highly dependent on the specific train-test split.

**Overfitting Management:** If training performance is strong (85%+ accuracy) but test performance is poor, the TA recommended reducing model complexity. This would involve fewer layers in the neural network, shallower Random Forest trees, or stronger regularization in Logistic Regression.

## 7.2 Future Improvements

Based on TA feedback and our analysis, we have identified specific improvements for the final model:

**Resampling Techniques:** Implement multiple strategies: (1) random undersampling of legitimate transactions to create 70-30 class distributions, (2) ADASYN (**?**) instead of SMOTE for more adaptive synthetic sample generation, (3) hybrid approaches combining oversampling and undersampling. Then compare results across different class balance ratios.

**Model Complexity Tuning:** For the v1 dataset, increase model capacity systematically. Train Ran-

dom Forest with 500-1000 trees and deeper maximum depth (30-40). Expand neural network to 3-4 hidden layers with 256-128-64-32 neurons. Monitor training vs. validation accuracy to identify the optimal complexity point where the model learns patterns without overfitting.

**Cross-Validation:** Implement 5-fold stratified cross-validation for all models. Report mean and standard deviation of all metrics across folds. Test multiple random seeds (42, 123, 456, 789, 1000) to assess result stability.

**Feature Engineering Refinement:** Investigate the Customer Location feature, check whether it represents genuine patterns or data leakage. Remove or modify highly predictive features temporarily to ensure other features are being used. Engineer additional behavioral features: transaction velocity (transactions per day), time since last transaction, average transaction amount changes over time, and product category diversity.

**Cost-Sensitive Learning:** Implement explicit class weights beyond balanced mode. Assign 10:1 cost ratio for false negatives vs. false positives to reflect reality that missing frauds is more costly than false alarms. Optimize threshold selection on validation set for maximum F1 or F2 score (emphasizing recall).

**Advanced Techniques:** Explore XGBoost with GPU acceleration for potentially better performance than Random Forest. Implement ensemble methods: stack predictions from multiple models using a meta-learner. Adapt the interpretability of the model using SHAP (**?**) values to understand which features drive individual predictions.

## 8 Team Contributions

Viransh did the data preprocessing and feature engineering efforts; implemented the data cleaning pipeline, SMOTE oversampling, and created engineered features including Total Customer Transactions, Address Mismatch, and Transaction Amount Ratio. Viransh also implemented GPU acceleration across all models using PyTorch for Neural Networks and RAPIDS cuML for Random Forest, significantly reducing training time. Additionally, Viransh coded the Logistic Regression model.

Ellen focused on model implementation and evaluation, coding the Random Forest and Neural Network models with appropriate architectures and training strategies. Ellen implemented the evaluation framework with multiple metrics (precision, recall, F1, AUC-ROC), generated all visualizations (ROC curves, feature importance plots, confusion matrices), and conducted the analysis comparing performance across dataset versions.

Both team members collaborated on code review and report/code writing.