Fraud Detection Using a Custom Decision Tree Classifier

Vaibhav Chourasia

1. Introduction

In financial systems, detecting fraudulent transactions is a crucial task to ensure security and minimize monetary loss. Machine learning techniques play an important role in automatically identifying fraudulent behaviour based on transaction patterns. In this project, I have implemented a Decision Tree. The classifier was trained on a highly imbalanced fraud detection dataset.

2. Why Decision Tree over Random Forest

A Decision Tree is a simple, interpretable model that recursively splits data based on the most informative features. It allows us to visualize and understand how decisions are made. Although a Random Forest generally offers better generalization by combining multiple trees, I chose Decision Tree for the following reasons:

- It provides complete transparency and interpretability.
- It allows direct control over the splitting logic and hyperparameters.
- Implementation from scratch is feasible and demonstrates clear understanding of tree-based models.
- It avoids the computational cost and memory overhead of handling many trees, which is ideal for large datasets like this one.

3. Mathematical Derivation

The Decision Tree algorithm works on the principle of recursively partitioning the dataset into smaller, purer subsets. The key metric used is Information Gain, based on the Entropy measure from information theory.

Step 1: Entropy

Entropy represents the impurity or uncertainty in a dataset:

$$H(S) = -\sum_{i=1}^{c} p_i \log_2(p_i)$$

where p_i is the proportion of class i in the subset S, and c is the number of classes (here $c = 2$ for fraud and non-fraud).

Step 2: Splitting Criterion

For a feature f and threshold t, the dataset is split into two parts:

$$S_L = \{x \mid x_f \leq t\}, \quad S_R = \{x \mid x_f > t\}$$

The weighted average entropy after the split is:

$$H_{split} = \frac{|S_L|}{|S|} H(S_L) + \frac{|S_R|}{|S|} H(S_R)$$

Step 3: Information Gain

The gain from splitting on feature f at threshold t is:

$$Gain(f, t) = H(S) - H_{split}$$

The algorithm selects the feature and threshold with the highest gain, provided that the gain exceeds a minimum threshold (min_gain).

Step 4: Stopping Conditions

The recursion stops when:

- The maximum depth is reached.
- The number of samples in a node is below $min\_samples\_split$.
- All samples in the node belong to the same class.

The final leaf node predicts the majority class, and the stored probability is used for the AUC computation.

4. Dataset Description

The dataset used was a transactional dataset containing over 6 million rows and 11 columns. After balancing and cleaning, 49,278 samples remained.

- Fraudulent transactions: 8213
- Non-fraudulent transactions: 41065
- Train samples: 39,422
- Test samples: 9,856

Attributes used

Feature	Description
step	Transaction step number (time indicator)
type	Encoded transaction type
amount	Transaction amount
oldbalanceOrg	Sender's initial balance
newbalanceOrig	Sender's balance after transaction
oldbalanceDest	Receiver's initial balance
newbalanceDest	Receiver's balance after transaction
isFlaggedFraud	Binary flag (system rule trigger)
isFraud	Target variable (1 = fraud, 0 = not fraud)

5. Data Preprocessing

- Columns `nameOrig` and `nameDest` were removed.
- The categorical column `type` was label-encoded into numeric form.
- The dataset was balanced by sampling non-fraud cases to be five times the number of fraud cases.
- The data was shuffled and split into 80% training and 20% testing sets.

6. Results and Visualizations

After training with `max_depth = 5`, the model achieved the following results:

Metric	Value
Accuracy	0.9674
Precision	0.8721
Recall	0.9435
F1 Score	0.9064
AUC	0.9931

The confusion matrix was:

$$\begin{bmatrix} 7981 & 228 \\ 93 & 1554 \end{bmatrix}$$

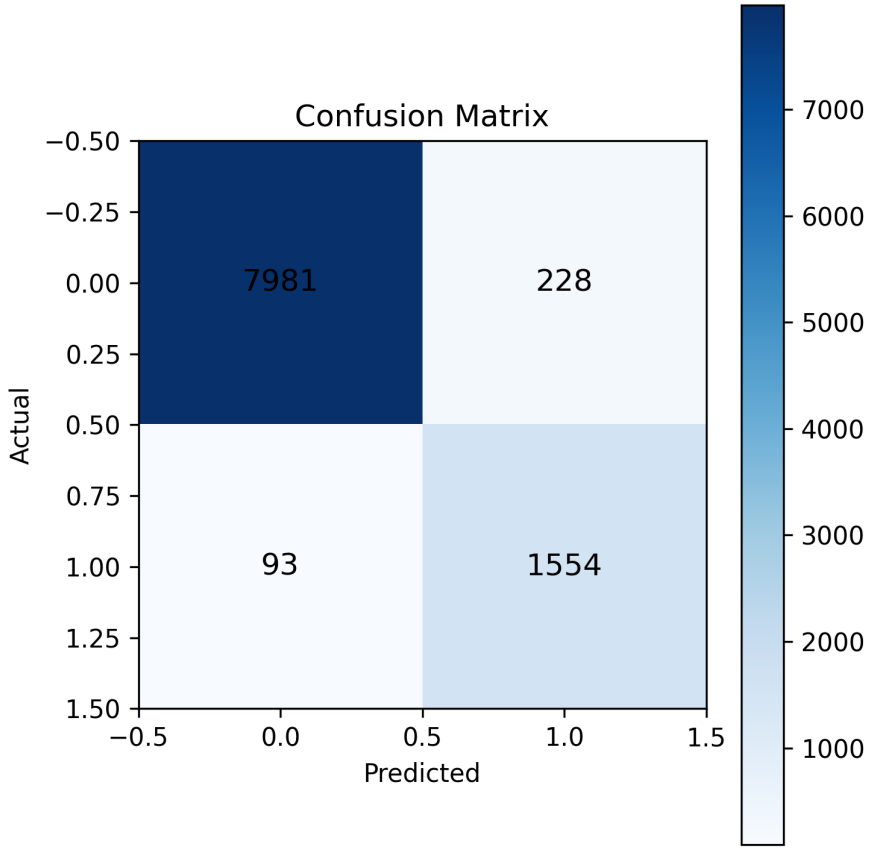

Figure 1: Confusion Matrix Visualization of Test Results

ROC Curve

The ROC curve demonstrates the trade-off between true positive and false positive rates, with an AUC close to 1 indicating excellent separation capability.

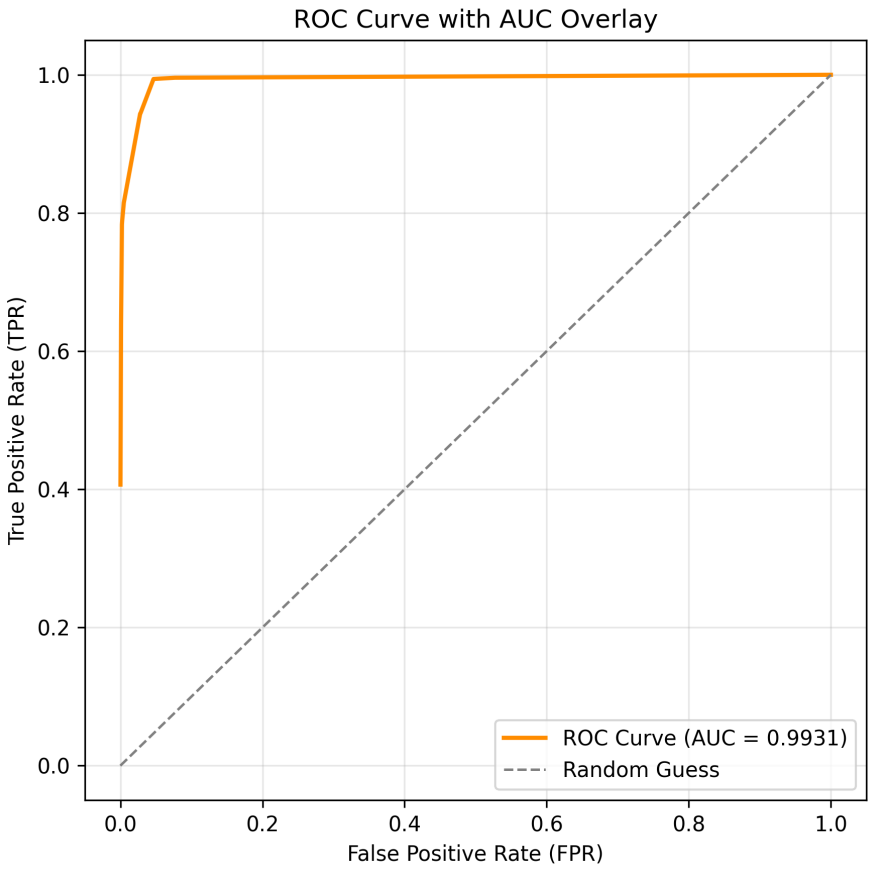

Figure 2: ROC Curve for the Custom Decision Tree Classifier

Feature Importance

Feature importance was computed based on the split frequency (weighted by depth):

Feature	**Importance**
oldbalanceOrg	2.43
amount	1.23
type	1.12
step	1.05
newbalanceDest	0.95
newbalanceOrig	0.50
oldbalanceDest	0.00
isFlaggedFraud	0.00

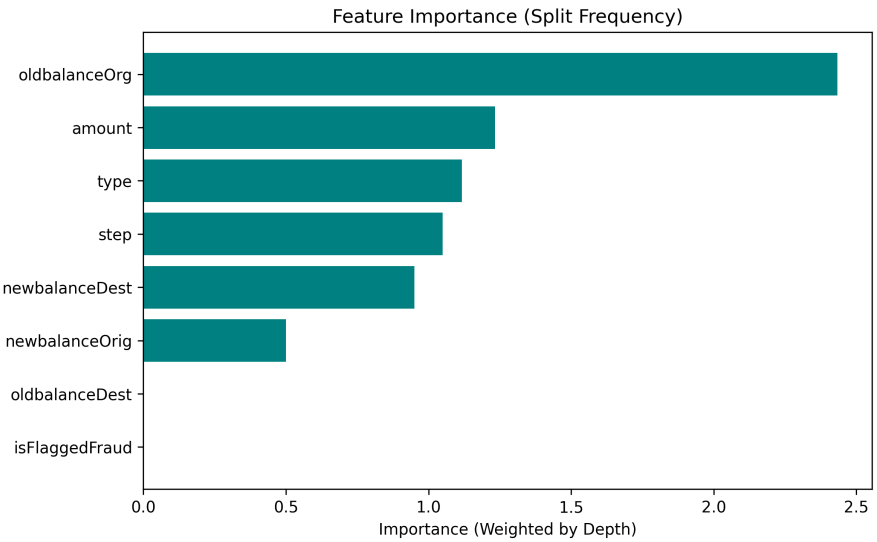

Figure 3: Feature Importance Plot

Hyperparameter Tuning (5-Fold CV)

The model was tested with various parameter combinations:

$$\texttt{max\_depth} \in \{3, 5, 7\}, \quad \texttt{min\_samples\_split} \in \{2, 5, 10\}, \quad \texttt{min\_gain} \in \{10^{-7}, 10^{-6}, 10^{-5}\}$$

The best result was obtained for:

```
max_depth = 7, min_samples_split = 10, min_gain = 1e-6
```

with the following mean metrics:

$$\text{Accuracy} = 0.9798, \quad \text{F1 Score} = 0.9394, \quad \text{AUC} = 0.9957$$

7. Inference from Results

- The model performed exceptionally well with AUC > 0.99, indicating excellent separation between fraud and non-fraud transactions.
- A high recall (0.94) means most fraudulent cases were successfully identified.
- `oldbalanceOrg` and `amount` were the most influential features, suggesting that the sender's initial balance and transaction amount strongly correlate with fraudulent activity.

- Increasing tree depth improved performance slightly without overfitting due to proper early stopping criteria.

8. Conclusion

The Decision Tree achieved strong accuracy and robustness in fraud detection. The visualizations further validated that the model not only performs well numerically but also provides interpretability and transparency.