XGBoost Model for Fraud Detection

Crank That Bayesian Handle

July 10, 2025

Why We Chose XGBoost

We picked XGBoost because it's reliable and fast when working with tabular data like transaction records. It handles complex relationships well without needing a ton of tuning, which is handy when you want to focus on feature engineering and analysis. Plus, it's designed to deal with imbalanced data — something that's common in fraud detection — so it helps the model pay more attention to the rare fraud cases. Overall, XGBoost strikes a good balance between performance and ease of use, which made it a natural choice for this project.

Feature Engineering: Additional Features

To enhance the model's predictive power, we engineered two additional features capturing temporal and behavioral patterns of transactions on the same card:

- **Time Since Last Transaction:** For each transaction, we calculated the time elapsed since the previous transaction on the same card. This helps capture unusual transaction timing patterns that might indicate fraud.
- **Amount Difference from Last Transaction:** We computed the difference in amount between the current and the previous transaction on the same card, providing insight into abnormal spending behavior.

Data Splitting and Parameter Tuning

To make sure our model would generalize well, we split the data based on time rather than randomly shuffling it. This way, the training data always comes from earlier transactions and the test data from later ones, which better reflects how the model would perform in the real world.

For tuning, we focused on a few key parameters to strike a balance between accuracy and avoiding overfitting:

- **max_depth**: Controls how deep each tree can grow. Deeper trees can capture more complex patterns but might overfit.
- **learning_rate**: Determines how quickly the model adapts to new information. Lower rates mean slower learning but can improve performance.
- **scale_pos_weight**: Adjusts for class imbalance by weighting fraud cases more heavily during training.
- **n_estimators**: Number of trees to build. More trees usually mean better performance up to a point.