Research Report

Overview

This report demonstrates that asymptotic improvements to throughput and average latency for tabular XGBoost models is possible, with no cost to precision and recall.

Dataset

- Credit Card Transaction Dataset on Kaggle
- The dataset contains 284807 transactions and 28 anonymized features, and the task is to classify each transaction as fraud or not. Fraud is rare within the dataset (only 492 instances).

Methods

Given the data and the predictions, we analyze the data to find rules that when evaluated, only contain non-fraud data.

At runtime, if the transaction is detected as part of this group, we output "Not Fraud", otherwise we run the model as usual and use its prediction.

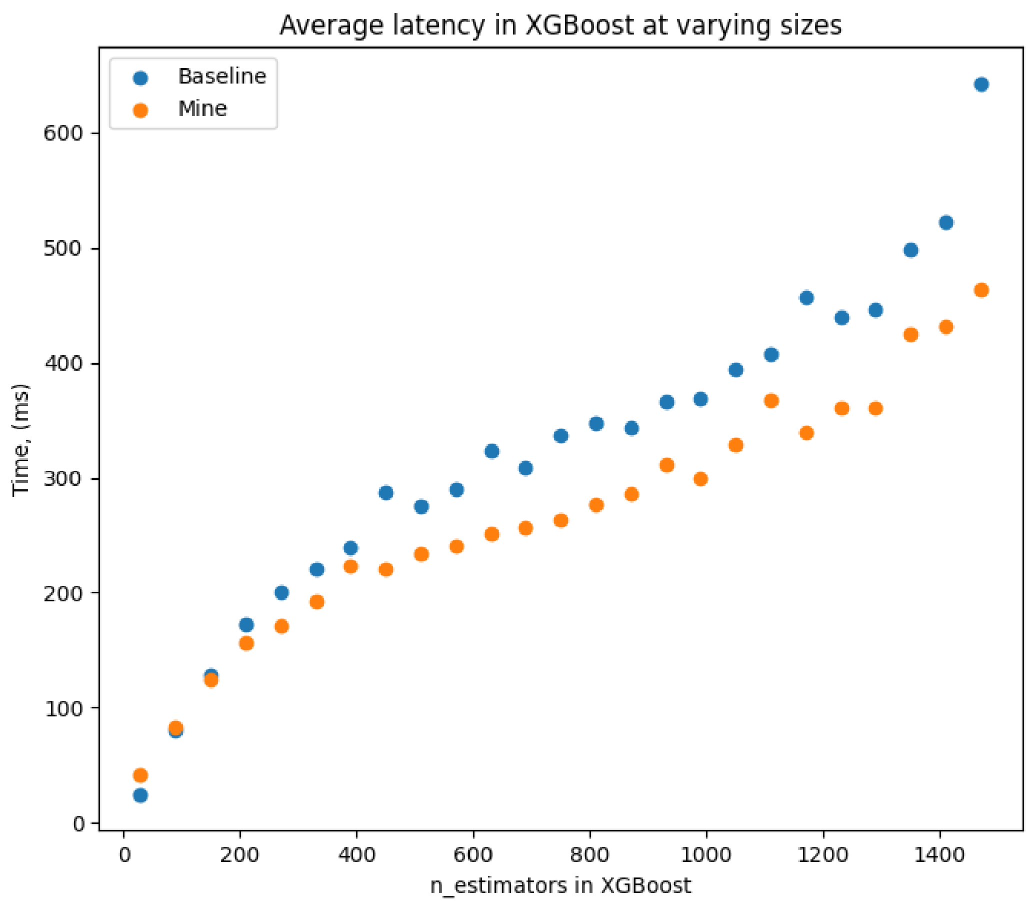
This approach's efficacy in reducing latency depends on the time complexity of the model, how many transactions the rule can quickly detect as not fraud, and how quickly the rule can operate.

Results

We trained an `XGBoost` model on this dataset from the library `xgb`, with n_estimators ranging from 30 to 1500 with a step size of 60 [30, 90, 150, ..., 1470].

In all observed cases [30, 1500] with a step size of 60, we observed consistent or improved precision and observed consistent recall (See Table 1 in Appendix.).

Note that the below times are over all the transactions (284807 transactions) of them.



Chosen arbitrarily to show specifics of speed-ups, (not from the above experiment), is n_estimators = 400:

Metric	**Baseline**	**Experimental**	**Change**
Time (ms)	235.906 ms	199.115 ms	-15.6% latency
Time (us) per trans.	0.8 us	0.7 us.	- 15.6% latency
Queries per second (QPS)	1.207M	1.430M	+18.4% throughput
Precision	0.934	0.934	--
Recall	0.685.	0.685	--

Appendix

Table 1: shows how across n_estimators, precision and recall are maintained.

n_estimators	**baseline_precision**	**experimental_precision**	**baseline_recall**	**experimental_recall**	**recall_diff**	**precision_diff**
30	0.926829	0.938272	0.612903	0.612903	0	0.0114423
90	0.920455	0.931034	0.653226	0.653226	0	0.0105799
150	0.922222	0.932584	0.669355	0.669355	0	0.010362
210	0.923077	0.933333	0.677419	0.677419	0	0.0102564
270	0.933333	0.933333	0.677419	0.677419	0	0
330	0.933333	0.933333	0.677419	0.677419	0	0
390	0.933333	0.933333	0.677419	0.677419	0	0
450	0.933333	0.933333	0.677419	0.677419	0	0
510	0.933333	0.933333	0.677419	0.677419	0	0
570	0.933333	0.933333	0.677419	0.677419	0	0
630	0.933333	0.933333	0.677419	0.677419	0	0
690	0.933333	0.933333	0.677419	0.677419	0	0
750	0.933333	0.933333	0.677419	0.677419	0	0
810	0.933333	0.933333	0.677419	0.677419	0	0
870	0.934066	0.934066	0.685484	0.685484	0	0
930	0.933333	0.933333	0.677419	0.677419	0	0
990	0.934066	0.934066	0.685484	0.685484	0	0
1050	0.933333	0.933333	0.677419	0.677419	0	0
1110	0.934066	0.934066	0.685484	0.685484	0	0
1170	0.934066	0.934066	0.685484	0.685484	0	0
1230	0.934066	0.934066	0.685484	0.685484	0	0
1290	0.934066	0.934066	0.685484	0.685484	0	0
1350	0.934066	0.934066	0.685484	0.685484	0	0
1410	0.934066	0.934066	0.685484	0.685484	0	0
1470	0.934066	0.934066	0.685484	0.685484	0	0