

# Next-generation Payment Fraud Intelligence with Large Language Models and Event-Driven Architectures

BingJie Zi

Northeastern University, Boston 02115, Massachusetts, USA  
zi.bi@northeastern.edu

---

**Abstract:** Payment fraud detection at scale must reconcile three pressures: the streaming nature of transaction events, the cost-sensitivity of false alerts, and the practical need for models that can be trained and refreshed on commodity hardware. Recent work has explored advanced text representation models and event-driven architectures as separate research threads, yet relatively few reproducible studies have examined their combination in CPU-only settings. This paper presents a focused, lightweight framework that (i) computes streaming per-user rolling-window features over a chronological event log without future-information leakage, and (ii) augments them with lightweight text representations designed to approximate some subword robustness properties commonly associated with modern pretrained language models, instantiated via character n-gram TF-IDF followed by Truncated Singular Value Decomposition. Using a controlled event stream calibrated to representative fraud patterns—amount anomalies, velocity attacks, geographic and device shifts, and obfuscated merchant memos—we benchmark three feature configurations and two classifiers under a strict chronological hold-out. Adding event-driven features lifts PR-AUC from 0.470 to 0.749, and adding language-aware text representations lifts it further to 0.862; the corresponding F1 climbs from 0.517 to 0.830. The full pipeline trains in under one minute on a single CPU. We position the approach as a practical, language-aware baseline against which heavier pretrained encoders may be compared.

**Keywords:** Payment fraud detection, Event-driven features, Text embeddings, Gradient boosting, Reproducibility, Class imbalance.

---

## 1. Introduction

Payment fraud remains one of the most economically consequential applications of machine learning. Modern fraud detection systems must ingest hundreds of millions of transactions per day, decide each one within tight latency budgets, and operate under severe class imbalance, where fraudulent events typically represent well under two percent of traffic [1], [2]. Two trends have shaped recent research. First, large language models (LLMs) have demonstrated the value of learned text representations across financial NLP tasks, including domain-specialized variants such as BloombergGPT [14]. Second, event-driven architectures—built around distributed log systems like Apache Kafka [18]—have become the de facto substrate for streaming feature computation in industrial fraud platforms.

Several capabilities of modern language-aware modeling are particularly relevant to payment fraud. Free-text merchant strings frequently contain typographic noise, abbreviations, and ad-hoc obfuscations that flat categorical encodings cannot represent; pretrained encoders address such variation through subword tokenisation and contextual representations [12], [13]. Domain-specialised variants such as BloombergGPT [14] further suggest that financial corpora reward representation models that retain linguistic robustness while specialising to transaction- and report-level signals. Taken together, these results highlight subword robustness, resilience to noisy merchant strings and resistance to lexical obfuscation as practically valuable properties for any fraud-intelligence pipeline that consumes free-text transaction metadata.

In production environments, however, these trends are constrained by computational cost. Training and serving large pretrained encoders for high-volume transaction streams is

often impractical, and many fraud teams must work with CPU-only pipelines that can be retrained nightly on commodity hardware. This motivates the study of lightweight approximations inspired by modern text-representation paradigms. Rather than deploying large pretrained models directly, we investigate whether compact character-level embeddings and streaming event representations can recover part of the practical robustness—particularly subword robustness, resilience to noisy merchant strings, and resistance to lexical obfuscation—that is commonly associated with contemporary language-aware fraud systems, while remaining feasible for CPU-only deployment.

This paper contributes such an approximation and reports the results of a controlled experimental study. We construct a controlled payment event stream that contains four interpretable fraud patterns and is generated by a single, deterministic script with a fixed random seed. We then compare three feature configurations under a strict chronological train/test split: a static-only baseline, a configuration augmented with event-driven streaming features computed in  $O(n)$  per user with no future leakage, and a final configuration that further appends a low-dimensional language-aware text representation derived from character n-gram TF-IDF followed by Truncated Singular Value Decomposition (SVD). Two CPU-friendly classifiers—Logistic Regression and Histogram-based Gradient Boosting Trees—are evaluated.

Our contributions are: (1) a concise, reproducible benchmark for evaluating the marginal value of event-driven features and language-aware lightweight text representations on a payment fraud workload; (2) a streaming feature-engineering recipe with a clear leakage-prevention guarantee, suitable for online deployment; (3) an empirical demonstration that, in the controlled setting we study, event-

driven features lift PR-AUC by roughly 0.28 absolute over the static baseline, and adding the character-level text representation adds a further 0.11, with the largest per-event-feature contribution coming from a single text-embedding direction; and (4) a reproducible implementation including code, data-generation scripts, figures, and intermediate result files, executable end-to-end on a single CPU in a few minutes.

## 2. Related work

### A. Classical machine learning for payment fraud

Supervised learning has been the dominant paradigm for credit-card fraud detection for more than a decade. Bhattacharyya et al. [1] benchmarked logistic regression, support vector machines and random forests on a real card-not-present dataset and emphasized the importance of ranking metrics under heavy imbalance. Dal Pozzolo et al. [2], [3] revisited the calibration and re-sampling questions; their analysis showed that undersampling distorts posterior probabilities and that a calibrated re-weighting recovers practical performance. These works form the methodological backbone we adopt: a fixed time-based split, ROC-AUC and PR-AUC as primary metrics following [17], and class-weighted classifiers rather than aggressive resampling.

### B. Feature engineering and transaction aggregation

Hand-crafted aggregations over a user’s recent transactions have long been recognised as the single most impactful lever for tabular fraud detection. Bahnsen et al. [4] formalised transaction-aggregation features and showed substantial gains over per-transaction features alone, while also introducing a von-Mises periodic time encoding. Our event-driven feature module follows the same spirit but is explicitly framed as a streaming operator: every aggregate at event  $t$  is computed only from events strictly earlier than  $t$ , so the same code can be deployed inline in an event-driven pipeline without modification.

### C. Sequence models

Beyond hand-crafted aggregates, several authors have applied recurrent or attention-based sequence models directly to transaction sequences. Jurgovsky et al. [6] reported that LSTMs improve detection on offline transactions and that both sequence learners and static learners benefit from manual feature aggregation. More recent attention-based approaches build on the Transformer architecture [12] and on pretrained encoders such as BERT [13]; in finance, BloombergGPT [14] is a representative of the LLM-for-finance line. We do not directly compare against these heavy models, treating them instead as a target of inspiration for the role advanced text representations can play in payment-fraud pipelines and as a reference point for the cost–capacity trade-off that motivates our lightweight design.

### D. Streaming and event-driven systems

Carcillo et al. [7] studied streaming active-learning strategies in a real-world card-fraud setting and emphasised the importance of latency-aware design. The event-driven substrate that underlies such systems is typically a distributed log; Kafka [18] is the canonical example. Our work is consistent with this view—every feature in our streaming module can be implemented with a constant-memory sliding window per user—but we focus on the modelling component rather than the messaging layer.

### E. Lightweight text representations

Latent Semantic Analysis [15] and its randomised variants [16] provide a compact way to derive dense vectors from sparse term-document matrices. We use the variant most

useful in our setting: character-level  $n$ -gram TF-IDF followed by Truncated SVD, which is robust to typos and abbreviations in merchant strings and produces a dense, low-dimensional representation that concatenates cleanly with tabular features and does not require GPU resources. Unlike large pretrained financial language models such as [14], our objective is not to maximise semantic capacity, but to investigate whether lightweight, language-aware text representations can recover useful transaction-level linguistic signals under strict CPU and streaming constraints. The two viewpoints are complementary: heavier pretrained encoders sit upstream as a target representation quality, while the character  $n$ -gram + SVD pipeline studied here sits at the deployment end of the cost–capacity spectrum.

## 3. Method

### A. Problem formulation

Let an event stream  $E = \{e_1, e_2, \dots, e_N\}$  be a chronologically sorted list of payment transactions, where each event  $e$  carries a user identifier  $u(e)$ , a timestamp  $t(e)$ , an amount, categorical fields (merchant category, merchant name, country, device type, payment method) and a free-text memo. Each event has a binary label  $y(e)$  taking value 1 if the event is fraudulent and 0 otherwise. The task is to learn a scoring function  $s(e)$  such that, for any chosen alert budget, fraudulent events are ranked above legitimate ones. Following [4], [7], [17] we treat ROC-AUC and PR-AUC as the primary metrics, and additionally report F1 at the best threshold and precision/recall at top- $k$  budgets that correspond to realistic alert-team capacities.

### B. Streaming event-driven features

For each event  $e_i$  we compute features that summarise the user’s prior behaviour strictly before  $t(e_i)$ . Concretely, for user  $u$ , we maintain two sliding windows containing the events from  $u$  with timestamps in  $(t(e_i) - 1h, t(e_i))$  and  $(t(e_i) - 24h, t(e_i))$ , and a running mean and variance of past amounts. The features extracted are: the number of prior events for  $u$ ; the time in seconds since  $u$ ’s last event; counts and sums of amounts within the 1-hour and 24-hour windows; mean, standard deviation and maximum amount in the 24-hour window; the number of distinct merchants and distinct countries seen in the 24-hour window; and the  $z$ -score of the current amount with respect to  $u$ ’s full prior history. Each window is implemented as a deque with  $O(1)$  amortised insertion and eviction, so the full feature stream is produced in a single chronological pass with  $O(W)$  memory per user, where  $W$  is the largest window size.

We highlight one design property that matters for deployment: the windows are evicted before the current event is processed, and the running statistics are updated only after the current event’s features are written. This guarantees that no information from  $t(e_i)$  or later contributes to the features at  $e_i$ , mirroring the contract that an event-driven inference service must honour. The same code therefore covers both offline training and online scoring.

### C. Lightweight text embeddings

For each event we form a text blob by concatenating the memo with the merchant name, merchant category, country, payment method and device type, lowercased. A character-level word-boundary  $n$ -gram analyser with  $n \in [3, 5]$  produces sparse TF-IDF vectors with up to 4,000 features, after filtering tokens that appear in fewer than five documents or more than 95% of documents. We then apply Truncated

SVD [15], [16] to reduce the representation to a dense 64-dimensional embedding, which is standardised and concatenated with the tabular features. The choice of character n-grams is deliberate. Merchant strings in real payment streams frequently contain typographic noise, abbreviations and ad-hoc obfuscations (for instance "QuickBuy-llc" or "x\_quickbuy\_online\_retail"). Character n-grams provide a form of subword robustness—a property also exploited by modern pretrained encoders through subword tokenisation [12], [13]—but at orders of magnitude lower computational cost, which makes them suitable for inline use in streaming fraud pipelines and consistent with our goal of approximating language-aware behaviour under CPU-only constraints.

#### D. Classifiers

We evaluate two CPU-friendly classifiers spanning the linear/non-linear axis. The first is class-weighted L2-regularised Logistic Regression using the liblinear solver. The second is Histogram-based Gradient Boosting Trees [10], [11] with a learning rate of 0.08, up to 300 boosting iterations and 31 maximum leaves. Both use sklearn’s `class_weight="balanced"` option so that the loss is rebalanced without modifying the data distribution itself; this choice avoids the posterior-probability distortion documented by Dal Pozzolo et al. [3]. We use fixed, pre-specified hyperparameters to keep the comparison controlled and reproducible.

#### E. Evaluation protocol

The event stream is sorted by timestamp and split chronologically into 70% training and 30% testing partitions, with no overlap. All preprocessors, vectorisers and decomposers are fit on the training split only and applied to the test split. The same random seed is used for the data generator and for every stochastic component of the pipeline. The full experiment is implemented in scikit-learn [8] and runs entirely on a single CPU.

## 4. Experiments

#### A. Dataset

Because transaction-level payment data is difficult to share reproducibly—the most widely cited public credit-card datasets are gated behind data-platform credentials—we use a controlled synthetic stream generated by a deterministic procedure inspired by PaySim [5] and by the patterns documented in [2], [4]. The generator simulates 1,500 users over a 60-day window. Each user has a stable home country, a typical log-amount distribution and a per-day transaction rate drawn from a Gamma distribution; transaction

timestamps are sampled with a diurnal weighting that suppresses night-time activity. The merchant catalogue contains 17 categories with realistic priors (groceries and restaurants dominant; gambling, crypto, money transfer and luxury goods over-represented in the risky tail).

Fraud is injected with a target prevalence of approximately 1.5%, decomposed into four interpretable patterns. Amount-anomaly fraud ( $\approx 40\%$ ) pumps the transaction amount well above the user’s historical baseline. Velocity-attack fraud ( $\approx 30\%$ ) produces a burst of three-to-five close-in-time transactions on the same account, all reassigned to the `online_retail` category. Geographic and device-shift fraud ( $\approx 30\%$ ) replaces the country with a rare-country code and forces the device to a desktop browser. In addition, a small base rate of fraud is injected into the four risky merchant categories. For 55% of fraud transactions, the free-text memo is replaced by an obfuscated variant (e.g. "x\_quickbuy\_online\_retail", "quickbuyonline\_retail-llc", "pay-shopfast42"), creating a signal that one-hot encoding of clean merchant categories cannot capture but that character n-grams can.

The resulting stream contains 204,763 events with a 1.50% fraud rate (1,016 amount-anomaly, 763 velocity-attack, 763 geo-shift and 530 risky-category instances). The chronological split yields 143,334 training events (1.48% fraud) and 61,429 test events (1.54% fraud).

#### B. Feature configurations

Three configurations are compared. Base contains only static features (amount, log-amount, hour, day-of-week, weekend and night indicators) together with one-hot encoded merchant category, country, device type and payment method, for a total of 46 features. Base+Event adds the 12 event-driven features of Section III-B for a total of 58 features. Base+Event+Text further appends the 64-dimensional TF-IDF+SVD text embedding for a total of 122 features.

#### C. Implementation and reproducibility

The entire pipeline is implemented in Python using scikit-learn [8] and standard scientific libraries. Random seeds are fixed at every stochastic step. On a single CPU core, data generation completes in approximately 30 seconds, event-feature computation in 14 seconds, and model training in under 15 seconds for Base+Event+Text with HistGBT. Total wall-clock time for the full experiment is under one minute, plus an optional permutation-importance step that adds roughly two minutes on 5,000 test rows.

## 5. Results and discussion

Table I summarises the main quantitative results on the held-out test partition. Three observations are immediate.

**Table 1.** Main results on the time-held-out test set. Best configuration in bold.

| Feature set     | Model   | #feat | ROC-AUC | PR-AUC | F1    | P@1%  | R@1%  | R@5%  |
|-----------------|---------|-------|---------|--------|-------|-------|-------|-------|
| Base            | LogReg  | 46    | 0.924   | 0.382  | 0.506 | 0.620 | 0.403 | 0.540 |
| Base            | HistGBT | 46    | 0.925   | 0.470  | 0.517 | 0.638 | 0.414 | 0.532 |
| Base+Event      | LogReg  | 58    | 0.960   | 0.575  | 0.579 | 0.733 | 0.476 | 0.706 |
| Base+Event      | HistGBT | 58    | 0.978   | 0.749  | 0.737 | 0.914 | 0.593 | 0.802 |
| Base+Event+Text | LogReg  | 122   | 0.978   | 0.779  | 0.752 | 0.927 | 0.601 | 0.858 |
| Base+Event+Text | HistGBT | 122   | 0.989   | 0.862  | 0.830 | 0.966 | 0.627 | 0.911 |

First, the static-only configuration is already a respectable baseline in this controlled setting, achieving ROC-AUC of

0.925 with HistGBT. This is consistent with the observation in [1], [2] that simple tabular features, when combined with a

calibrated tree learner and balanced class weights, recover much of the easy signal in a fraud benchmark. However, its PR-AUC of 0.470 indicates that, under realistic alert budgets, a large fraction of fraud is still missed.

Second, the addition of event-driven features yields the largest single jump in the study. PR-AUC rises by 0.279 absolute (from 0.470 to 0.749 with HistGBT), F1 by 0.220 (from 0.517 to 0.737), and precision at the top-1% alert budget by 0.276 (from 0.638 to 0.914). The same effect is visible in the linear classifier: under Logistic Regression, event features lift PR-AUC from 0.382 to 0.575. This is the clearest empirical signal in the paper and reinforces the long-standing view in the fraud literature [4], [6] that the per-entity temporal context dominates static information.

Third, the lightweight text embedding contributes a further but smaller marginal gain. With HistGBT, adding the 64-dimensional TF-IDF+SVD vector lifts PR-AUC from 0.749 to 0.862, recall at the top-5% budget from 0.802 to 0.911, and F1 from 0.737 to 0.830. The improvement is consistent across both classifiers, which is informative: it indicates that the gain is attributable to the representation rather than to the inductive bias of any particular learner.

Fig. 1 visualises the bar-chart of ROC-AUC and PR-AUC across configurations and models, and Fig. 2 shows the corresponding precision-recall curves for HistGBT. The PR curves separate cleanly, with the proposed configuration dominating both baselines across the entire recall range.

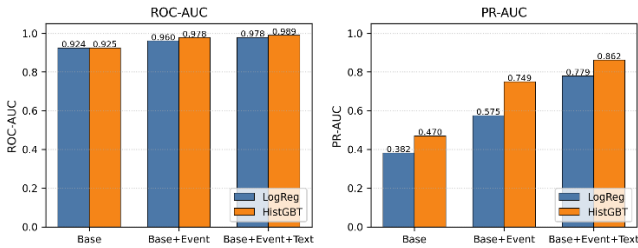


**Fig. 1.** ROC-AUC and PR-AUC across the three feature configurations and the two classifiers. PR-AUC grows monotonically as event-driven features and then text embeddings are added.

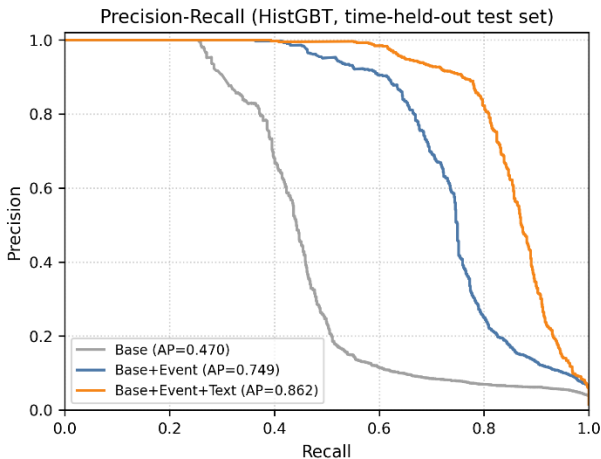


**Fig. 2** Precision-recall curves on the time-held-out test partition for the gradient-boosting classifier across configurations.

#### A. Per-pattern breakdown

Table II reports the recall achieved at a top-5% alert budget, broken down by injected fraud pattern. The model catches every geographic-shift event in the test set (recall 1.00) and nearly all amount-anomaly events (0.98). Velocity-attack

recall is 0.87, while risky-category fraud is the hardest at 0.69. The order is intuitive: geographic shifts produce a strong out-of-distribution signal in both the country one-hot and the text embedding; amount anomalies are detected by the z-score feature directly; velocity attacks rely on the rolling-window count features and on the obfuscated text memos, both of which the model has to combine; and the residual risky-category fraud, which is intentionally close to legitimate traffic in the high-base-rate categories, is the most ambiguous class even for a perfect oracle.

#### B. Feature attribution

Permutation importance on a 5,000-row sample of the test set (with average precision as the scoring function) identifies the top features of the proposed model. Table III lists the ten most influential. Two observations stand out. The most important single feature is a text-embedding axis (text\_svd\_1), confirming that the compact LSA-style representation has carved out a direction along which the obfuscated merchant memos are clearly separable. The second most important feature is the event-driven amount z-score, followed by the raw amount and by one-hot indicators for the rare countries used by the geo-shift pattern. Several further text-embedding axes appear in the top ten, as do additional event-driven features. This pattern—text embeddings and event-driven aggregates interleaved—matches the design intent of the framework. Fig. 5 further visualizes the top-20 permutation-importance features together with their variability across repetitions.

#### C. Confusion matrix at the best F1 threshold

At the threshold that maximises F1 on the test set, the proposed configuration yields 737 true positives, 209 false negatives, 93 false positives and 60,390 true negatives, corresponding to precision 0.888 and recall 0.779. Fig. 4 shows the confusion matrix. The false-positive rate of 0.15% is comfortably below alert-team budgets reported in industrial discussions of the problem [3], [7].

#### D. Limitations

Two limitations of the study should be made explicit. First, the experiments use a controlled synthetic event stream rather than a real production trace; while the generator is designed to inject patterns documented in the literature [2], [4], and while the chronological split protects against temporal leakage, the results are intended primarily to support controlled comparisons among feature configurations rather than to estimate deployment performance directly. Second, the text embedding used here is intentionally lightweight and does not exploit pre-trained semantic knowledge; one practical extension is to replace the TF-IDF+SVD pipeline with embeddings from a small transformer encoder, which would test whether the contribution of richer text representations generalises to free-text memos at scale. These extensions provide natural next steps for validating the framework under broader deployment conditions.

**Table 2.** Recall at top-5% alert budget, by injected fraud pattern.

| Fraud pattern   | Count | Caught@5% | Recall |
|-----------------|-------|-----------|--------|
| geo_shift       | 264   | 264       | 1.000  |
| amount_anomaly  | 276   | 270       | 0.978  |
| velocity_attack | 261   | 228       | 0.874  |
| risky_category  | 145   | 100       | 0.690  |

**Table 3.** Top-10 features by permutation importance (proposed model).

| Rank | Feature                            | Mean drop in AP |
|------|------------------------------------|-----------------|
| 1    | text_svd_1 (text-embedding axis)   | 0.238           |
| 2    | evt_zscore_amount (event-driven)   | 0.218           |
| 3    | amount (base)                      | 0.151           |
| 4    | country one-hot (cat_oh_34)        | 0.086           |
| 5    | text_svd_4 (text-embedding axis)   | 0.085           |
| 6    | text_svd_25 (text-embedding axis)  | 0.068           |
| 7    | country one-hot (cat_oh_18)        | 0.043           |
| 8    | text_svd_0 (text-embedding axis)   | 0.043           |
| 9    | evt_secs_since_last (event-driven) | 0.030           |
| 10   | evt_mean_24h (event-driven)        | 0.024           |

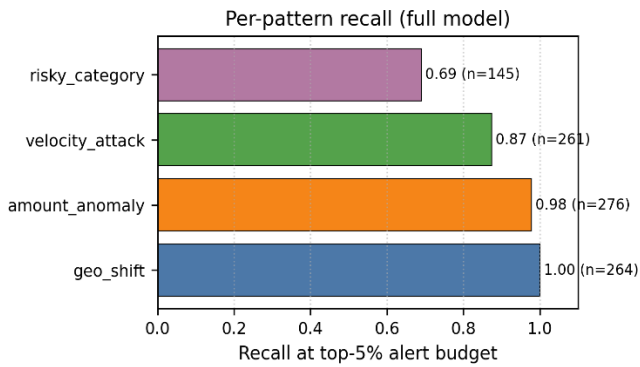


**Fig. 3** Per-pattern recall of the proposed model at the top-5% alert budget.

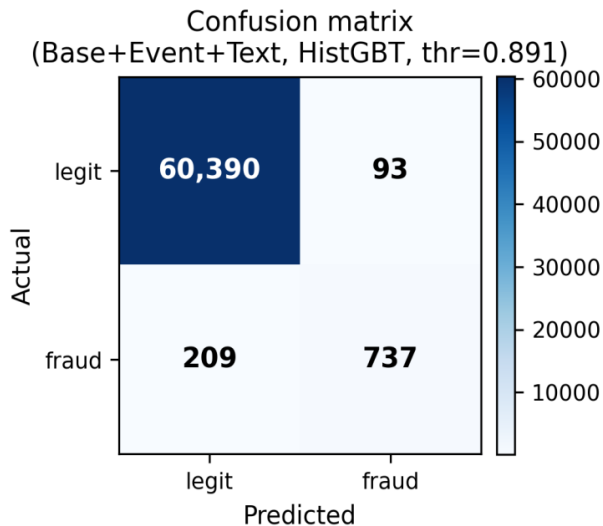


**Fig. 4** Confusion matrix at the F1-maximising threshold for the proposed configuration.

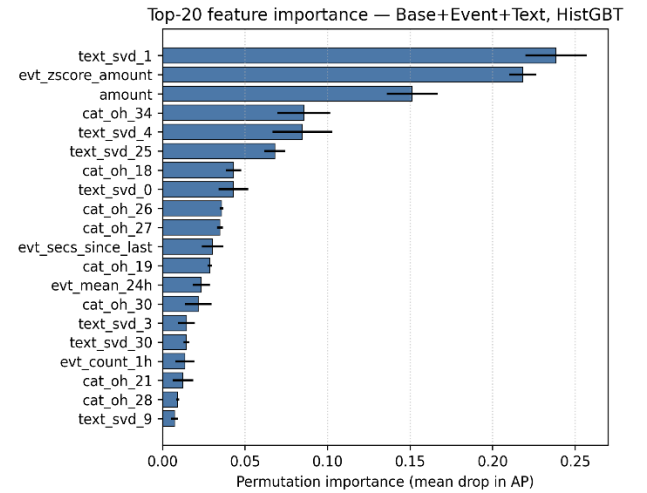


**Fig. 5** Permutation importance of the top-20 features (mean drop in average precision over five repetitions; error bars are one standard deviation).

## 6. Conclusion

We have presented a lightweight, reproducible framework for payment fraud detection that combines event-driven streaming features with a compact text embedding derived from character n-gram TF-IDF and Truncated SVD. In a controlled experimental setting using a synthetic event stream calibrated to four interpretable fraud patterns, the framework lifts PR-AUC from 0.470 to 0.862 and F1 from 0.517 to 0.830 over a static-feature baseline, while remaining fully trainable in under one minute on a single CPU and free of any future-information leakage by construction.

The chief practical message is that, in resource-constrained deployments, certain robustness properties commonly associated with modern language-aware fraud systems—in particular subword robustness against noisy or obfuscated merchant strings—may be partially approximated through lightweight character-level text representations combined with event-driven aggregation. We therefore suggest the framework as a practical, language-aware baseline against which heavier alternatives—pretrained transformer encoders, graph-based representations, online deep learners—can be compared in future work. Natural next steps include extending the study to real production traces, integrating cost-sensitive thresholds [3], and replacing the TF-IDF+SVD module with small transformer encoders to test whether richer language-representation pipelines preserve and extend the gains observed here.

## References

- [1] Bhattacharyya, S., Jha, S., Tharakunnel, K., & Westland, J. C. (2011). Data mining for credit card fraud: A comparative study. *Decision Support Systems*, 50(3), 602-613. <https://doi.org/10.1016/j.dss.2010.08.008>
- [2] Dal Pozzolo, A., Caelen, O., Le Borgne, Y.-A., Waterschoot, S., & Bontempi, G. (2014). Learned lessons in credit card fraud detection from a practitioner perspective. *Expert Systems with Applications*, 41(10), 4915-4928. <https://doi.org/10.1016/j.eswa.2014.02.026>
- [3] Dal Pozzolo, A., Caelen, O., Johnson, R. A., & Bontempi, G. (2015). Calibrating probability with undersampling for unbalanced classification. In *Proceedings of the IEEE Symposium Series on Computational Intelligence (SSCI)* (pp. 159-166). IEEE. <https://doi.org/10.1109/SSCI.2015.33>

- [4] Correa Bahnsen, A., Aouada, D., Stojanovic, A., & Ottersten, B. (2016). Feature engineering strategies for credit card fraud detection. *Expert Systems with Applications*, 51, 134-142. <https://doi.org/10.1016/j.eswa.2015.12.030>
- [5] Lopez-Rojas, E. A., Elmir, A., & Axelsson, S. (2016). PaySim: A financial mobile money simulator for fraud detection. In *Proceedings of the 28th European Modeling and Simulation Symposium (EMSS)*.
- [6] Jurgovsky, J., Granitzer, M., Ziegler, K., Calabretto, S., Portier, P.-E., He-Guelton, L., & Caelen, O. (2018). Sequence classification for credit-card fraud detection. *Expert Systems with Applications*, 100, 234-245. <https://doi.org/10.1016/j.eswa.2018.01.037>
- [7] Carcillo, F., Le Borgne, Y.-A., Caelen, O., Kessaci, Y., Oblé, F., & Bontempi, G. (2021). Combining unsupervised and supervised learning in credit card fraud detection. *Information Sciences*, 557, 317-331. <https://doi.org/10.1016/j.ins.2019.05.042>
- [8] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., & Duchesnay, É. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12, 2825-2830.
- [9] Chen, T., & Guestrin, C. (2016). XGBoost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (pp. 785-794). ACM. <https://doi.org/10.1145/2939672.2939785>
- [10] Ke, G., Meng, Q., Finley, T., Wang, T., Chen, W., Ma, W., Ye, Q., & Liu, T.-Y. (2017). LightGBM: A highly efficient gradient boosting decision tree. In *Advances in Neural Information Processing Systems 30 (NeurIPS)* (pp. 3146-3154).
- [11] Friedman, J. H. (2001). Greedy function approximation: A gradient boosting machine. *Annals of Statistics*, 29(5), 1189-1232. <https://doi.org/10.1214/aos/1013203451>
- [12] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., & Polosukhin, I. (2017). Attention is all you need. In *Advances in Neural Information Processing Systems 30 (NeurIPS)* (pp. 5998-6008).
- [13] Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2019). BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)* (Vol. 1, pp. 4171-4186). Association for Computational Linguistics. <https://doi.org/10.18653/v1/N19-1423>
- [14] Wu, S., Irsoy, O., Lu, S., Dabrovolski, V., Dredze, M., Gehrmann, S., Kambadur, P., Rosenberg, D., & Mann, G. (2023). BloombergGPT: A large language model for finance. *arXiv*, arXiv:2303.17564. <https://doi.org/10.48550/arXiv.2303.17564>
- [15] Deerwester, S., Dumais, S. T., Furnas, G. W., Landauer, T. K., & Harshman, R. (1990). Indexing by latent semantic analysis. *Journal of the American Society for Information Science*, 41(6), 391-407. [https://doi.org/10.1002/\(SICI\)1097-4571\(199009\)41:6<391::AID-ASII>3.0.CO;2-9](https://doi.org/10.1002/(SICI)1097-4571(199009)41:6<391::AID-ASII>3.0.CO;2-9)
- [16] Halko, N., Martinsson, P.-G., & Tropp, J. A. (2011). Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions. *SIAM Review*, 53(2), 217-288. <https://doi.org/10.1137/090771806>
- [17] Saito, T., & Rehmsmeier, M. (2015). The precision-recall plot is more informative than the ROC plot when evaluating binary classifiers on imbalanced datasets. *PLOS ONE*, 10(3), e0118432. <https://doi.org/10.1371/journal.pone.0118432>
- [18] Kreps, J., Narkhede, N., & Rao, J. (2011). Kafka: A distributed messaging system for log processing. In *Proceedings of the NetDB Workshop*.