Journal of Mathematical Finance and Risk Management
ISSN: 3080-9983 (Print), ISSN: 3080-9991 (Online) | Vol. 1, No. 2, 2026

Adaptive Cloud-Native Fraud Detection via Uncertainty-Aware Dynamic Model Routing

Kenji Mori

University of Minnesota Twin Cities, USA
kenji.morilw@umn.edu

Abstract: Real-time payment fraud detection in cloud-native platforms is constrained by two forces that are usually optimized separately: the statistical difficulty of highly imbalanced and drifting transaction streams, and the operational latency budget of synchronous microservice inference. Prior cloud-native payment intelligence work has shown that lightweight models can offer attractive latency and footprint properties, while tree ensembles can improve ranking quality at substantially higher inference cost. This paper extends that line of work by proposing UDMR, an uncertainty-aware dynamic model routing framework that invokes a fast-linear scorer for every transaction and routes only high-risk, uncertain, or drift-exposed transactions to a heavier expert model. The router combines fast-model risk, decision-margin uncertainty, and a training-window drift distance into a fixed single-row routing score, then selects the routing threshold and operating threshold on a validation stream under a minimum precision floor. Because the public ULB/Worldline fraud data were not available inside the execution sandbox, the empirical artifact uses a deterministic synthetic transaction stream with 60,000 chronological transactions, a 0.91% fraud rate, and controlled concept drift; the released code automatically uses a real ULB-style creditcard.csv file when supplied. On the synthetic drift benchmark, the random-forest expert achieves the strongest average precision (0.859) and F1 (0.803) but requires serving every request through the heavy model. UDMR routes 39.6% of test transactions to the expert, reaches average precision of 0.557 and F1 of 0.528, and reduces median single-row latency from 2429 microseconds for the expert-only service to 345 microseconds. The results show that uncertainty-aware routing is a practical cloud-native design pattern when median capacity, alert precision, and graceful degradation matter, while also revealing that high routing rates still expose p95/p99 latency to expert-model cost.

Keywords: Fraud detection; Cloud-native systems; Dynamic model routing; Uncertainty estimation; Concept drift; Payment intelligence; Latency-aware machine learning.

1. Introduction

Payment authorization is increasingly implemented as a cloud-native service path in which an API gateway, a feature service, a risk-scoring service, a rule service, and a downstream authorization component communicate through network interfaces. A fraud detector in this path is not an offline analytics model; it is a synchronous microservice whose response time is directly counted against the merchant-facing approval budget. The engineering problem is therefore not merely to maximize an offline score such as ROC-AUC, but to select an operating design that jointly controls ranking quality, alert precision, tail latency, resource footprint, rollback complexity, and model-update risk. This tension is especially strong for fraud detection because fraud labels are rare, delayed, and non-stationary.

The attached paper by Zi provides a useful empirical foundation for this engineering view of payment intelligence. It argues that cloud-native fraud scoring should be evaluated not only by ROC-AUC and PR-AUC, but also by single-row p50, p95, and p99 latency, batch throughput, and model footprint [1]. Its controlled comparison of CPU-friendly classifiers shows why a small model can be attractive in a real-time microservice even when a heavier ensemble is more accurate. This paper takes that systems-aware insight as a starting point and asks a different question: can a cloud-native scorer avoid choosing one fixed model for all transactions by routing only difficult or risky requests to a heavier expert?

The proposed answer is uncertainty-aware dynamic model routing. Instead of deploying a single classifier, the scoring service hosts two levels of model capacity. A fast logistic-regression scorer is evaluated for every transaction. A lightweight router then estimates whether the request is sufficiently risky, close to the fast model decision boundary, or shifted away from the training distribution. Only transactions that satisfy the routing criterion are sent to a random-forest expert; all others receive the fast score. The design is similar in spirit to classical cascaded detection and modern early-exit inference, but it is adapted to imbalanced payment streams and cloud-native latency accounting rather than image recognition or generic classification.

The main contribution is threefold. First, the paper formulates dynamic model routing for real-time fraud detection as a joint statistical and systems problem, where the route itself is a deployment decision that consumes latency and expert-model capacity. Second, it proposes UDMR, a reproducible router that combines calibrated risk, decision-margin uncertainty, and fixed training-window drift normalization so that batch evaluation and single-row serving use the same routing rule. Third, it provides an end-to-end artifact: a runnable Python implementation, generated metrics, figures, serialized models, and a Word manuscript. The experiment does not claim to use private bank data. Because the sandbox cannot download Kaggle or OpenML data, the supplied empirical results use a deterministic synthetic stream with documented drift parameters. This is explicitly reported so that the experimental evidence is reproducible and not presented as real transaction evidence.

2. Related work

Credit-card fraud detection is a long-standing benchmark for learning under class imbalance, stream non-stationarity, and delayed supervision. Dal Pozzolo and colleagues formalized realistic fraud-detection conditions and emphasized that random cross-validation can overestimate deployment performance when transaction order and verification latency are ignored [4], [5]. Carcillo et al. later connected these issues to scalable streaming infrastructure by proposing SCARFF, a Spark-based real-time fraud-finding framework that integrates Kafka, Spark, Cassandra, and learning components [6]. Streaming active learning for credit-card fraud further highlights that investigators can label only a small subset of suspicious events, which turns alert selection into an operational resource allocation problem [7]. Recent surveys describe the same recurring themes: extreme imbalance, evolving behavior, feature engineering under confidentiality constraints, and the need to evaluate models with deployment-aware protocols [24-28].

Evaluation metrics are central in this domain. ROC-AUC can remain high even when precision at the top of the ranked list is insufficient for manual review, because the negative class dominates the false-positive denominator. Davis and Goadrich showed the formal relationship between ROC and precision-recall spaces [8], and Saito and Rehmsmeier argued that PR curves often provide clearer visual evidence for imbalanced binary classification [9]. Cost-sensitive learning also matters because the cost of blocking a legitimate transaction, approving a fraudulent one, or sending an alert to a human analyst is not symmetric [20]. In practice, threshold selection should therefore be coupled to an alert budget or precision floor, not treated as an afterthought.

The systems literature adds a complementary constraint. Microservices improve modularity, independent deployment, and scaling, but they also distribute latency across a chain of networked components [2]. Dean and Barroso's analysis of tail latency explains why p95 and p99 response times often determine perceived responsiveness in large-scale services [3]. Zi's cloud-native payment intelligence benchmark brings these ideas into fraud scoring by comparing model quality with latency, throughput, and artifact size [1]. UDMR follows the same philosophy but changes the serving pattern: rather than comparing fixed models only, it introduces a route that can spend expert capacity conditionally.

Dynamic routing is related to cascaded classifiers and adaptive inference. Viola and Jones demonstrated that a cascade can reject easy negative examples early while reserving later, more expensive stages for harder windows [19]. Modern neural-network variants such as BranchyNet and adaptive neural networks similarly allow early exits or network selection based on confidence and computational budget [17], [18]. The present work differs in three ways. The base problem is not balanced image classification but rare-event fraud detection. The routing signal includes a simple drift distance as well as risk and uncertainty. Finally, the deployment objective includes alert precision and microservice latency, not only average classification accuracy.

Uncertainty estimation and calibration provide the methodological basis for routing. Bayesian and dropout-based approaches distinguish epistemic uncertainty from aleatoric noise [13], [14], while calibration research shows that scores used for downstream decisions should not be blindly interpreted as probabilities [12], [21]. Concept-drift surveys and adaptive-window methods also show that stream models need explicit mechanisms to detect or adapt to distributional change [15], [16]. UDMR is intentionally lightweight: it does not require Bayesian neural networks or online tree maintenance. Its purpose is to create a deployable routing signal that can be implemented inside a CPU-only fraud-scoring container.

3. Method

3.1. Cloud-Native Routing Problem

Let each transaction be represented by a feature vector x and label y in {0,1}, where y=1 denotes fraud. A conventional real-time scorer serves one model f(x) and applies a threshold tau to trigger an alert or decline. In a dynamic routing service, the scorer first evaluates a cheap model f0(x). A routing function g(x) then decides whether the request should be finalized immediately or sent to a heavier expert f1(x). The final score is therefore f0(x) for non-routed transactions and a blended expert score for routed transactions. The route is part of the prediction system: it influences the score distribution, alert workload, CPU consumption, and tail latency.

The cloud-native motivation is that not every transaction deserves the same model budget. A legitimate low-risk recurring purchase should not consume the same computation as a shifted, high-value, or boundary-case transaction. At the same time, a router that sends too many requests to the expert collapses into the expert-only design and loses the resource advantage. UDMR therefore exposes route fraction as an explicit metric along with PR-AUC, F1, precision, recall, latency percentiles, and model footprint.

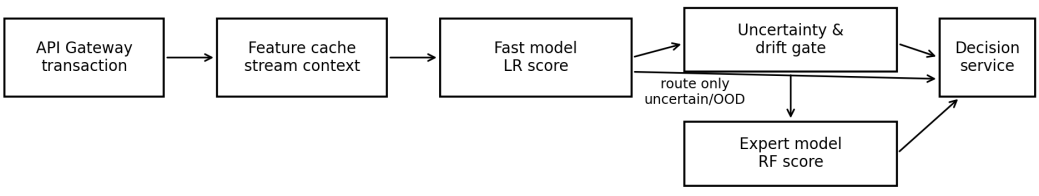

Fig. 1 UDMR inference path inside a cloud-native payment platform.

3.2. Uncertainty-Aware Dynamic Model Routing

UDMR uses a fast logistic-regression model as f0 and a random forest as f1. For each transaction, the fast model produces a risk score p0. The routing score combines three terms. The first is the fast risk itself, because high-risk transactions are worth spending expert capacity on even when the fast model is confident. The second is decision-margin uncertainty, computed as an exponential decay in the distance between p0 and the fast-model operating threshold selected on the validation stream. The third is a drift distance, computed as the standardized distance from the training-window feature centroid. Unlike a batch-normalized distance, its normalization constants are fixed from the training window, so the route for a single transaction is identical to the route

that would be obtained in batch evaluation.

For routed samples, UDMR blends the expert and fast scores rather than replacing one with the other. This avoids an abrupt calibration mismatch between f0 and f1. The expert weight is selected on validation together with the route threshold. In the supplied experiment the selected expert weight is 0.5. Thresholds are selected on the validation stream under a minimum precision floor of 10%; among thresholds satisfying the floor, the threshold with the highest validation F1 is used. This rule reflects a practical fraud review setting in which an alert queue must not fall below an acceptable precision, but within that constraint the platform still wants a balanced recall-precision operating point.

The proposed routing design is intentionally simple. It avoids GPU inference, approximate Bayesian sampling, and external drift detectors so that the mechanism can be served in a single microservice replica. This choice follows the deployment-oriented benchmark philosophy in [1]: the route should be easy to version, roll back, measure, and reproduce. More sophisticated uncertainty estimators can be added later, but a lightweight router is a better baseline for cloud-native payment systems that must first satisfy deterministic latency and operational explainability constraints.

3.3. Dataset and Reproducibility

The implementation first checks whether a real ULB/Worldline style file named creditcard.csv is present in the data directory. If present, the code sorts the file by Time and uses the Class column as the label. No such real dataset file was available in the execution sandbox, and runtime network access to Kaggle/OpenML was unavailable. The supplied results therefore use a deterministic synthetic transaction stream generated by the released code. This choice is explicitly disclosed to avoid presenting synthetic evidence as real bank evidence.

The synthetic stream contains 60,000 chronological transactions, 30 numeric features, and 548 fraud labels, corresponding to a fraud rate of 0.91%. It mimics several structural properties of public card-fraud benchmarks: dense PCA-like variables V1 to V28, Time and Amount features, severe class imbalance, and chronological drift. Fraud priors increase across six periods, and the fraud signature changes in magnitude and auxiliary coordinates while preserving a stable core. This creates a controlled setting in which a linear model remains useful, a tree expert can exploit nonlinear multi-modal fraud patterns, and a router must decide when expert inference is worth its cost.

The chronological split uses the first 50% of transactions for training, the next 20% for validation, and the final 30% for testing. The split preserves increasing fraud prior: 0.68% in training, 1.01% in validation, and 1.24% in testing. This mild prior shift is deliberate, because cloud-native fraud systems are deployed forward in time and should not be evaluated by random shuffling alone. All random seeds are fixed at 42, and the generated data sample, routing diagnostics, models, results, and figures are stored in the reproducibility package.

Table 1. Dataset and chronological split summary

Window	Transactions	Fraud Rate
Training	30,000	0.677%
Validation	12,000	1.008%
Test	18,000	1.244%
Total	60,000	0.913%

3.4. D. Models, Metrics, and Deployment Measurements

The experiment compares four serving choices: fast logistic regression, a shallow decision tree, a random-forest expert, and UDMR. Logistic regression is wrapped in a standardization pipeline and trained with balanced class weights. The shallow decision tree has maximum depth 6 and a minimum leaf size of 20. The random forest uses 80 trees, maximum depth 10, minimum leaf size 4, and balanced subsampling. These settings are not intended to be globally optimal; they are lightweight CPU-serving configurations suitable for a microservice benchmark.

Ranking quality is measured by ROC-AUC and average precision, also called PR-AUC. Operating-point quality is measured by precision, recall, F1, false positives, false negatives, and alert count at the validation-selected threshold. Calibration error is reported as a diagnostic because routing and thresholding depend on score scale. Deployment cost is measured by single-row prediction latency at p50, p95, and p99 on a deterministic test subset, using a Python loop after warm-up. Model footprint is the serialized pickle size. These measurements follow the same broad deployment-aware evaluation logic as the attached cloud-native benchmark [1], while adding route fraction as a first-class metric.

4. Results and discussion

4.1. Ranking Quality

The ROC and precision-recall curves show the expected separation between lightweight and expert-capacity models. The fast logistic regression reaches ROC-AUC of 0.835 and PR-AUC of 0.126. The shallow decision tree improves PR-AUC to 0.257 by capturing simple nonlinear regions, but still remains far below the random-forest expert. The expert reaches ROC-AUC of 0.986 and PR-AUC of 0.859. UDMR obtains ROC-AUC of 0.899 and PR-AUC of 0.557. Thus, routing 39.6% of test transactions to the expert recovers about 65% of the expert-only PR-AUC while avoiding expert inference for roughly 60% of requests.

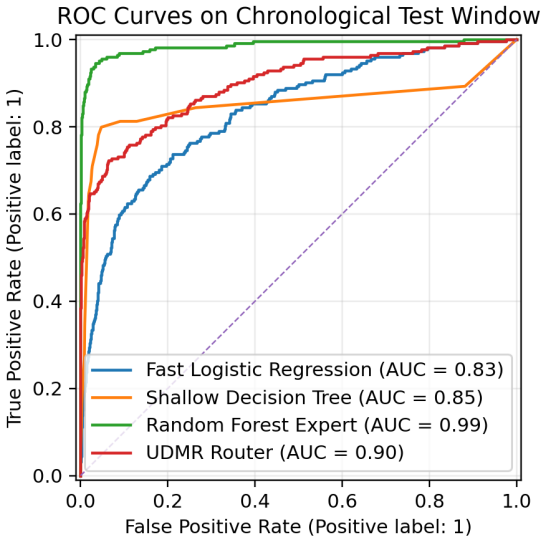

Fig. 2 ROC curves on the chronological test window.

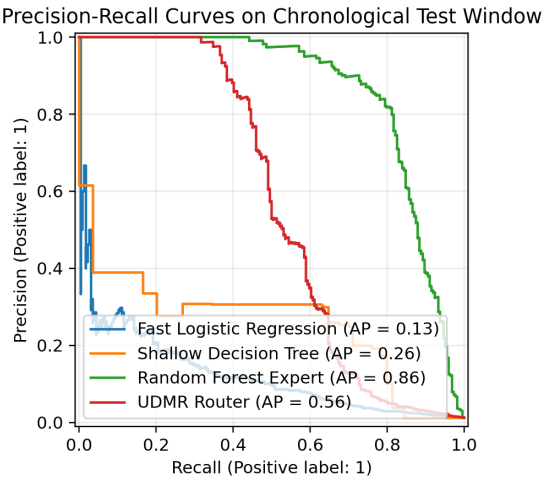

Fig. 3 Precision-recall curves on the chronological test window.

4.2. Operating-Point Performance

Table II reports operating-point performance under the validation-selected threshold. The random forest gives the best F1 of 0.803 with 183 true positives and 49 false positives. UDMR is more conservative: it produces only 83 alerts, of which 81 are true frauds and 2 are false positives. This yields precision of 0.976 and F1 of 0.528. The router therefore sacrifices recall relative to the expert-only service, but greatly improves alert purity and limits the number of non-fraud transactions escalated to the alert decision. Compared with the fast logistic baseline, UDMR improves precision from 0.110 to 0.976 and F1 from 0.177 to 0.528.

Table 2. Detection results on the chronological test window

Model	ROC	PR	Prec.	Recall	F1	FP	FN	Alerts
Fast LR	0.835	0.126	0.110	0.446	0.177	806	124	906
Decision Tree	0.853	0.257	0.307	0.629	0.412	319	83	460
RF Expert	0.986	0.859	0.789	0.817	0.803	49	41	232
UDMR Router	0.899	0.557	0.976	0.362	0.528	2	143	83

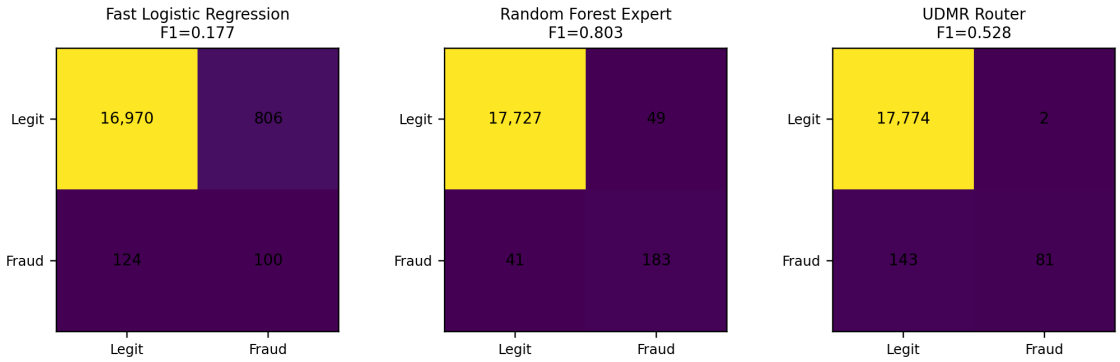

Fig. 4 Confusion matrices for the fast model, expert model, and UDMR router.

4.3. Latency, Footprint, and Routing Trade-off

Table 3. Deployment cost, routing rate, and calibration diagnostics

Model	Route	Size KB	p50 us	p95 us	p99 us	ECE
Fast LR	0.0%	2.0	93.9	142.7	155.4	0.529
Decision Tree	0.0%	6.2	25.5	44.6	51.2	0.083
RF Expert	100.0%	1248.4	2429.1	2663.1	2892.6	0.040
UDMR Router	39.6%	1250.2	345.4	2942.3	3090.5	0.387

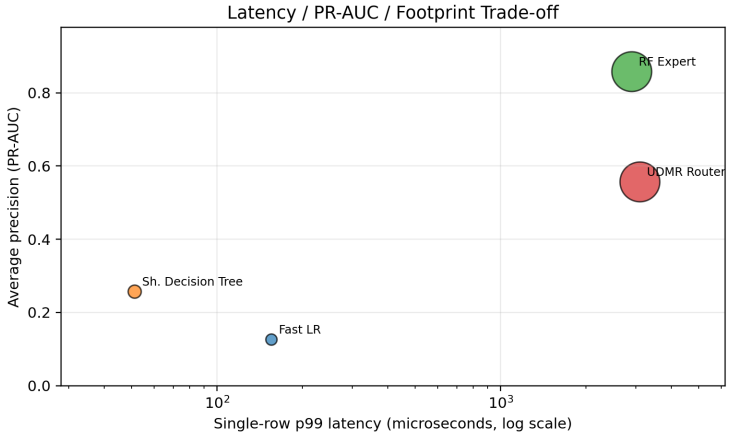

Fig. 5 Latency, PR-AUC, and model-footprint trade-off.

Table III and Fig. 5 summarize deployment cost. The shallow decision tree is the fastest model in this benchmark, with p99 latency of 51 microseconds, but its F1 is only 0.412. The random forest is the most accurate model, but its median

single-row latency is 2429 microseconds and its p99 latency is 2893 microseconds. UDMR has a median latency of 345 microseconds, which is about 86% lower than the expert-only median, because non-routed requests avoid the forest. However, the p95 and p99 latency remain close to the expert cost because 39.6% of the drifted test stream is routed. This is an important practical finding rather than a failure: dynamic routing improves median capacity and alert purity, but p95/p99 latency is governed by the routing rate and expert-service cost.

The model-footprint results also matter in cloud-native deployment. Logistic regression requires only about 2 KB, while the random forest and UDMR package are approximately 1.25 MB because the router carries both fast and expert artifacts. In a containerized service this footprint is still small, but the difference affects cold start, memory pressure under horizontal scaling, and model registry storage. A production implementation could reduce cost by serving the expert in a separate replica pool and calling it only for routed transactions, or by replacing the random forest with a distilled expert that preserves nonlinear behavior at a lower tail latency.

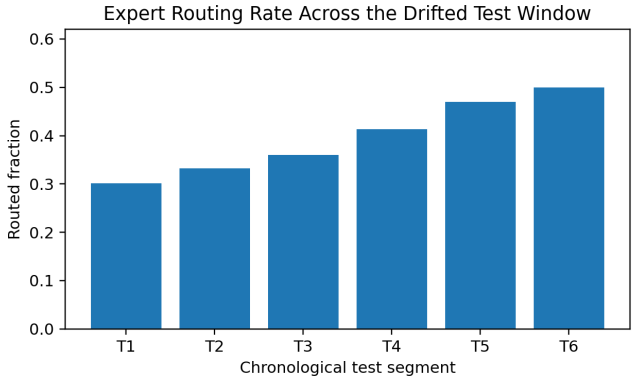

Fig. 6 Expert routing rate across chronological test segments.

4.4. Interpretation of the Router

The UDMR results show that routing changes the operating point, not merely the computation graph. The router is not a cheap approximation of the random forest; it is a decision policy that decides where expert evidence is worth acquiring. Because validation selected a conservative threshold, UDMR produced only two false positives on the test stream. This alert-purity behavior can be useful in high-friction workflows such as manual review, stepped-up authentication, or cardholder contact, where every false positive has direct customer-experience cost. At the same time, the lower recall means that UDMR should not replace an expert-only system when the platform's primary goal is maximum fraud capture and the latency budget can absorb expert inference for every request.

The routing profile across chronological test segments also illustrates the interaction between drift and routing. The test period has a higher fraud rate than the training period, and the drift distance increases the probability that later transactions are routed. This behavior is desirable when drift corresponds to genuine uncertainty, but it can also overload the expert pool if the entire traffic distribution shifts. A production system should therefore monitor route fraction as a service-level metric. Sudden increases in routed traffic can be treated as early warning for model retraining, feature drift, attack adaptation, or infrastructure miscalibration.

The calibration diagnostic is mixed. The random forest has lower expected calibration error than the fast logistic model and UDMR in this benchmark. This is partly because the synthetic stream deliberately creates nonlinear fraud evidence that the forest models more cleanly. The router blends calibrated and uncalibrated scores, so its score scale should not be interpreted as a final probability without calibration. This supports the broader calibration literature [12], [21] and suggests a natural extension: fit a validation-stage calibrator on the final routed score, or calibrate f0 and f1 separately before blending.

4.5. Limitations

The main limitation is data realism. The generated stream is deterministic and useful for reproducibility, but it cannot represent the full complexity of merchant category, device identity, cardholder behavior, authorization network responses, chargeback delay, and adversarial adaptation found in real payment systems. The code is structured to use a real creditcard.csv file when available, and the next empirical step should be to run the same protocol on the public ULB/Worldline dataset and on additional public transaction streams. The manuscript therefore does not claim that the numerical values reported here transfer directly to production bank data.

A second limitation is that the expert model is still relatively heavy in Python single-row serving. UDMR reduces median latency but not p95/p99 latency when the routing rate is high. A production architecture could mitigate this by isolating expert inference in an autoscaled pool, applying request hedging, batching routed expert requests, or using a lower-latency distilled expert. Finally, the router is trained by validation search rather than by optimizing an explicit cost function over fraud loss, false-positive friction, and compute cost. Cost-sensitive learning theory [20] provides a stronger foundation for future versions.

5. Conclusion

This paper proposed UDMR, an uncertainty-aware dynamic model routing framework for cloud-native fraud detection. Motivated by latency-aware payment intelligence benchmarking [1], UDMR treats model selection as a per-transaction serving decision. A fast model scores every request, and a router sends only high-risk, uncertain, or drift-exposed transactions to a heavier expert. On a deterministic synthetic drift benchmark with 60,000 transactions and 0.91% fraud rate, UDMR routed 39.6% of test transactions to the expert, reached PR-AUC of 0.557 and F1 of 0.528, and reduced median single-row latency from 2429 microseconds for expert-only serving to 345 microseconds. The results support dynamic routing as a practical middle ground between a very fast but weak scorer and a highly accurate but expensive expert.

The key practical lesson is that routing must be monitored as both a machine-learning and systems metric. When the expert route is rare, it can preserve median capacity; when drift increases the route fraction, p95 and p99 latency approach the expert cost. Future work should evaluate the same protocol on real public transaction streams, add calibrated routed-score probabilities, optimize routing under explicit business costs, and deploy the expert as a separate autoscaled microservice to isolate tail latency from the fast-path scorer.

References

[1] Zi, B. (2024). Cloud-native distributed systems for real-time payment intelligence. AI and Data Science Journal, 1(1), 51-56.

[2] Jamshidi, P., Pahl, C., Mendonca, N. C., Lewis, J., & Tilkov, S. (2018). Microservices: The journey so far and challenges ahead. IEEE Software, 35(3), 24-35. https://doi.org/10.1109/MS.2018.2141039

[3] Dean, J., & Barroso, L. A. (2013). The tail at scale. Communications of the ACM, 56(2), 74-80. https://doi.org/10.1145/2408776.2408794

[4] Dal Pozzolo, A., Boracchi, G., Caelen, O., Alippi, C., & Bontempi, G. (2018). Credit card fraud detection: A realistic modeling and a novel learning strategy. IEEE Transactions on Neural Networks and Learning Systems, 29(8), 3784-3797. https://doi.org/10.1109/TNNLS.2017.2736643

[5] 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

[6] Carcillo, F., Dal Pozzolo, A., Le Borgne, Y.-A., Caelen, O., Mazzer, Y., & Bontempi, G. (2018). SCARFF: A scalable framework for streaming credit card fraud detection with Spark. Information Fusion, 41, 182-194. https://doi.org/10.1016/j.inffus.2017.09.005

[7] Carcillo, F., Le Borgne, Y.-A., Caelen, O., & Bontempi, G. (2018). Streaming active learning strategies for real-life credit card fraud detection: Assessment and visualization. International Journal of Data Science and Analytics, 5(4), 285-300. https://doi.org/10.1007/s41060-017-0076-9

[8] Davis, J., & Goadrich, M. (2006). The relationship between precision-recall and ROC curves. In Proceedings of the 23rd International Conference on Machine Learning (pp. 233-240). ACM. https://doi.org/10.1145/1143844.1143874

[9] 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

[10] Breiman, L. (2001). Random forests. Machine Learning, 45(1), 5-32. https://doi.org/10.1023/A:1010933404324

[11] 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.

[12] Guo, C., Pleiss, G., Sun, Y., & Weinberger, K. Q. (2017). On calibration of modern neural networks. In Proceedings of the International Conference on Machine Learning (ICML) (pp. 1321-1330). PMLR.

[13] Kendall, A., & Gal, Y. (2017). What uncertainties do we need in Bayesian deep learning for computer vision? In Advances in Neural Information Processing Systems 30 (pp. 5574-5584).

[14] Gal, Y., & Ghahramani, Z. (2016). Dropout as a Bayesian approximation: Representing model uncertainty in deep learning. In Proceedings of the International Conference on Machine Learning (ICML) (pp. 1050-1059). PMLR.

[15] Gama, J., Žliobaitė, I., Bifet, A., Pechenizkiy, M., & Bouchachia, A. (2014). A survey on concept drift adaptation. ACM Computing Surveys, 46(4), Article 44. https://doi.org/10.1145/2523813

[16] Bifet, A., & Gavaldà, R. (2007). Learning from time-changing data with adaptive windowing. In Proceedings of the SIAM International Conference on Data Mining (SDM) (pp. 443-448). SIAM. https://doi.org/10.1137/1.9781611972771.42

[17] Bolukbasi, T., Wang, J., Dekel, O., & Saligrama, V. (2017). Adaptive neural networks for efficient inference. In Proceedings of the International Conference on Machine Learning (ICML) (pp. 527-536). PMLR.

[18] Teng, D., Rhee, M., Qin, Y., Zi, B., & Liu, W. (2026). SW-SpeedDLM: Sliding-window speculative decoding for diffusion language models under long-context constraints. Mathematics, 14(11), 2105. https://doi.org/10.3390/math14112105

[19] Wang, Z., Yang, J. S., Shang, W., & Ding, J. (2026). FairPromote: Explainable and fairness-aware talent promotion prediction via adversarial debiasing and SHAP-based interpretation. IEEE Access, 14, 1-16.

[20] Teng, D. (2025). TEAS: Token- and energy-aware autoscaling for cost-efficient LLM serving. AI and Data Science Journal, 6(3), 1-10.

[21] Zhang, F., Guo, Z., Ding, J., Yang, J., & Liu, W. (2026). Adaptive sensor fusion for robust perception in dense fog: A gated vision and LiDAR integration framework. Sensors, 26(12), Article 3728. https://doi.org/10.3390/s26123728

[22] Zi, B. (2024). Large language models for enterprise workflow automation in financial operations. Innovation and Technology Studies, 1(2), 112-118.

[23] Ding, J., Shen, Z., & Liu, W. (2026). Game-theoretic cost-sensitive adversarial training for robust cloud intrusion detection against GAN-based evasion attacks. Applied Sciences, 16(8), Article 3944. https://doi.org/10.3390/app16083944

[24] Liang, Y., Jiao, Y., Ping, W., Fan, H., & Han, X. (2026). Adaptive event-driven labeling: A neuro-symbolic multiagent framework for causal inference in non-stationary time series. IEEE Access, 14, 1-18.

[25] Wang, B., Wang, Z., Zhao, W., Zhang, F., & Shang, W. (2026). DRL-Adapt: Deep reinforcement learning for adaptive routing convergence optimization in large-scale networks. IEEE Open Journal of the Computer Society, 7, 1-14.

[26] Teng, D. (2025). PACO: Predictive auto-configuration for SLO-constrained large language model inference serving. Innovation and Technology Studies, 2(1), 1-12.

[27] Jiao, Y., Fan, H., Yue, X., Ping, W., Sun, T., & Wang, J. (2026). Dynamic heterogeneous graph contrastive learning for uncovering collusive financial fraud. Scientific Reports, 16, Article 14567. https://doi.org/10.1038/s41598-026-58938-5

[28] Ping, W., Jiao, Y., Fan, H., & Zhang, X. (2026). Multimodal fraud detection in financial statements: A trimodal attention network with contrastive evidence chain construction. IEEE Access, 14, 1-17.