International Journal of Scientific Research in Engineering and Management (IJSREM)
Volume: 10 Issue: 10 | Oct - 2026 SJIF Rating: 8.659 ISSN: 2582-3930

SalesSense: A Proposed Serverless Machine Learning Framework for Product Sales Classification and Business Decision Support

Gouri Dhadage[1,*], Arpita Dekate[2], Prof. Nitin Ganeshar[3], Prof. Tushar Kathane[4]

[1]Student – SY MCA, [2]Student – SY MCA [3]Asst. Prof., [4]Asst. Professor

[1,2,3,4]Yashaswi Education Society's International Institute of Management Science, Chinchwad, Pune, India

gouriidhadage@gmail.com, arpitadekate04@gmail.com, nitin.ganeshar@gmail.com, tushar.kathane@gmail.com

***Abstract*—Retail companies use machine learning to sort products into sales categories that guide stocking and promotion decisions. Running such a classifier as a cloud service raises separate questions about latency, scaling and cost. This paper proposes SalesSense, an event-driven serverless design on AWS (S3, Lambda, API Gateway, SageMaker Serverless Inference, DynamoDB, SNS, CloudWatch, IAM) that connects product-sales classification to a decision-support workflow. The design has not been deployed. We report local experiments on three public retail datasets, with naive baselines and checks that remove product scale. On the two temporal datasets, a persistence rule that repeats last week's class matched or beat every learned model: on Store Sales it scored 0.987 accuracy against 0.972 for the best model over three folds, and on Online Retail 0.602 against 0.599 over eight folds. With per-product class boundaries, the Online Retail models fell to 34–38% accuracy, close to chance. On the cross-sectional BigMart data, logistic regression reached 0.688 accuracy against 0.488 for an outlet-only rule, driven mainly by item price. High classification accuracy therefore mostly reflects product scale and recent sales, not forecasting skill. We also specify a benchmark that compares the serverless endpoint with an always-on endpoint. No latency, cold-start, throughput or cost measurements exist yet, so the paper makes no claim about operational benefits.**

***keywords*—serverless computing, Function-as-a-Service, machine learning inference, retail analytics, product sales classification, decision support, cloud architecture**

I. Introduction

Retail teams need to know quickly which products are selling well and which are not. A classifier can turn a continuous sales figure into categories such as low, medium and high. A working application needs more than the model, though. It needs data intake, input checks, inference, a readable response, access control, monitoring and a way to trigger follow-up action.

Serverless computing runs code in response to events and scales it automatically. Its behavior depends on the workload and the settings. Cold starts, platform overhead, resource limits and hard-to-repeat measurements are well documented problems.

SalesSense is a design for connecting retail data handling, ML classification and decision support in one event-driven AWS workflow. The cloud design is the main subject. The classifier is the workload we use to exercise it. We keep two things apart throughout: the local experiments, which were run, and the AWS deployment, which was not. The results therefore say something about prediction quality and nothing about latency, scaling or cost.

The paper has three contributions. First, it proposes an AWS reference architecture for product-sales classification and decision support (Section IV). Second, it gives an evaluation protocol that separates prediction quality from cloud performance, with temporal target-window purging (Section V) and a benchmark against an always-on baseline (Section VII). Third, it reports preliminary results on three public retail datasets, including naive-baseline and scale-removing checks that change how the scores should be read (Section VI). The architecture combines existing services. We do not claim a new serverless mechanism, a new classifier or a measured performance gain.

II. Background and Related Work

A. Serverless Computing

Serverless platforms hide infrastructure management and usually offer event-driven execution and pay-per-use billing. Surveys describe architecture, resource orchestration, performance, cost and security as linked concerns [15], [16], [18]. The model suits workloads whose request rate varies, but the benefit depends on the request pattern, the resource settings and the services involved.

B. Performance, Cold Starts and Reproducibility

A cold start happens when the platform has to set up an execution environment before it can run a request. A multivocal review of 112 FaaS performance studies found that most work centered on AWS Lambda and on microbenchmarks, that triggers were little studied, and that many experiments had flaws that hurt reproducibility [17]. Later reviews group cold-start mitigation into application-, checkpoint-, prediction- and cache-based approaches [19]. A trace of 85 billion requests on a production platform showed that the number and length of cold starts depend on trigger type, runtime and resource allocation [20]. A fair evaluation of SalesSense therefore has to time the whole request path and report configuration, workload, cold or warm state and the number of repetitions.

C. Serverless ML Inference

Serving a model on serverless infrastructure adds model loading, payload handling, batching and resource sizing. BATCH studies adaptive batching on a serverless platform and compares cost and latency against SageMaker among other systems [21]. AWS guidance describes a serverless inference pipeline in which a SageMaker Serverless Inference endpoint classifies the request [22], and AWS publishes a toolkit that benchmarks a serverless endpoint against a comparable real-time instance [23]. Serverless ML serving already exists, so SalesSense is not the first such system.

Recent reviews point the same way. A mapping study of 53 serverless ML papers lists cost, scalability, latency, cold starts, security and privacy as recurring concerns [24]. A survey of serverless inference sorts the work into four groups: SLO-aware, acceleration-aware, framework-aware and latency-aware [25]. An analytical model validated on AWS Lambda predicts response time and cold-start probability [26]. Trieu et al. ran ML workloads on open-source serverless edge frameworks under different workloads and user counts [27]. Jiang et al. compared serverless and serverful distributed training and found that FaaS can be much faster but is not significantly cheaper [28]. Together these studies are the reason we keep prediction results and operational measurements apart.

D. ML for Retail Sales

Most retail sales work frames the task as forecasting. Ma and Fildes learn features from raw sales series and use them to weight a pool of base forecasters across many products and stores [29]. Classification studies mostly address customers or orders. Chaubey et al. compare supervised classifiers for customer purchasing behavior, including stacked hybrids [30]. Al-Batah et al. classify retail sales orders by size with ten models and report near-perfect accuracy for some ensembles on a Kaggle dataset [31]. In the abstracts we read, these studies evaluate prediction alone and do not cover deployment. Scores that high also show why the split and the target definition have to be stated, as we do in Section V.

E. Serverless Analytics for Sales

One recent report is close to our setting. Kumar et al. describe an AWS serverless sales-analytics platform that uses Glue and Athena for processing, Prophet for forecasting and a conversational interface [32]. We read only its abstract and opening sections. SalesSense differs in the task (leakage-aware product-sales classification), in the use of SageMaker Serverless Inference, and in a benchmark protocol with a matched baseline.

F. Research Gap

The gap is narrower than the absence of serverless ML or serverless analytics. It is the problem of connecting a product-sales classification task, evaluated with leakage-aware temporal splits, to an auditable decision-support workflow on a SageMaker Serverless Inference architecture, and of reporting prediction quality separately from end-to-end cloud performance against a defined baseline. This gap statement is provisional. It rests on a focused pilot search, not a systematic review, so it cannot rule out earlier work that combines these pieces.

III. Research Questions

RQ1: How can a product-sales classification workflow be split into event-driven serverless components for validation, inference, storage, notification and monitoring?

RQ2: How well do selected classifiers perform under bounded, leakage-aware evaluation on the chosen retail datasets?

RQ3: Which benchmark protocol and measurements are needed to decide whether the serverless design meets latency, scalability, cold-start, reliability and cost goals in a controlled comparison?

Table I links each question to its evidence. Running the benchmark behind RQ3 is outside this study because nothing has been deployed.

TABLE I
Research Questions and Evidence

RQ	Evidence in this paper	Status
RQ1	Fig. 1, Algorithm 1, Table II	Design only
RQ2	Tables III–IX (accuracy, macro-F1)	Run; see baselines
RQ3	Section VII protocol	Specified, not run

IV. SalesSense Architecture

Fig. 1 shows the request path. S3 holds uploaded datasets and model artifacts. API Gateway receives requests. Lambda validates and normalizes inputs and calls the model. SageMaker Serverless Inference hosts the trained classifier. DynamoDB stores request metadata and outcomes. SNS sends notifications, CloudWatch collects telemetry and IAM controls permissions. This is a design. It has not been deployed.

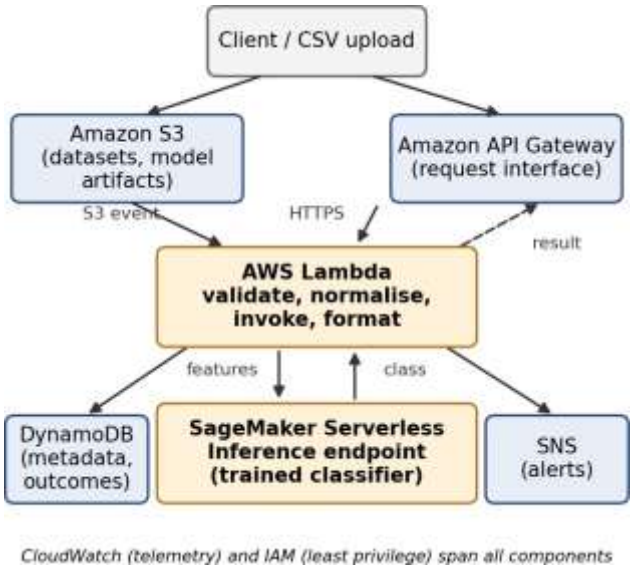


Fig. 1. Proposed SalesSense request path (design only).

A. Request Flow

- Intake: a payload arrives through the API or an S3 event.
- Validation: Lambda checks fields, types, batch size and feature schema, and returns a structured error for bad input.
- Inference: Lambda calls the SageMaker Serverless Inference endpoint with the approved features.
- Response: the predicted class and context go back to the user. Any business action follows explicit, auditable rules.
- Logging: metadata, status and latency go to DynamoDB and CloudWatch. Raw or sensitive data are not kept without need.
- Notification: SNS sends threshold or completion alerts when enabled.

The design requires least-privilege IAM policies, payload limits, retries and timeouts, idempotent event handling and defined behavior when the endpoint is down. Each evaluation records the model version, dataset version, feature schema and

inference settings. Cost figures must come from measured invocations and the prices in effect at the time.

B. Problem Definition

Take a series s with ordered records and observed value v(s,t) at origin date t. For a horizon of H records, the continuous target is

$$z(s,t) = v(s,t+1) + \ldots + v(s,t+H).$$

With q1 and q2 the one-third and two-thirds quantiles of z in the training fold, the class is Low if $z \le q1$, Medium if $q1 < z \le q2$, and High otherwise. A classifier maps the features x(s,t), known at prediction time, to a class. Quantiles come from training rows only and are applied unchanged to the test fold.

For the service, request j arrives at a_j and returns at b_j, so its latency is $\ell_j = b_j - a_j$. Let $c_j = 1$ if the request was a cold start and $u_j = 1$ if it succeeded. With N_succ successful requests and total billed charge C_total in an evaluation window, cost per successful prediction is C_total / N_succ.

C. Request Handler

Algorithm 1. Lambda handler (design)

```
parse event; take request_id as key
if request_id stored: return result
validate schema, types, batch<=B_max
  on failure return 400
x <- normalise(payload, schema v)
y <- InvokeEndpoint(x), timeout T,
  bounded retry
on endpoint error: log, return 503
label <- map(y); apply threshold rules
store id, v, model version, label,
  latency, status
if rule fires: publish SNS message
return formatted response
```

D. What Is Established and What Is New

TABLE II
COMPONENT STATUS

Element	Role	Status
S3, Lambda, API Gateway, DynamoDB, SNS, CloudWatch, IAM	Storage, compute, interface, state, alerts, telemetry, access	Existing AWS services
SageMaker Serverless Inference	Hosts the classifier	Existing service; keeps the inference tier serverless
Event-driven composition, handler logic	Validation, idempotency, response format	Proposed design
Leakage-aware protocol	Window purging, training-fold quantiles	Proposed; Section VI
Matched-baseline benchmark	Serverless vs. always-on	Specified, not run

V. DATA AND METHOD

A. Datasets

We use three public retail datasets: Store Sales Time Series (Kaggle; 3,000,888 daily rows for 54 stores and 33 product families, 1 January 2013 to 15 August 2017), UCI Online Retail (transactions aggregated by product and date) and BigMart Sales (Analytics Vidhya; 8,523 item-outlet rows from 10 outlets in 2013). Store Sales and Online Retail are used as time series with temporal folds. BigMart is cross-sectional and uses random splits.

B. Targets and Leakage Control

For Store Sales we use four stores (1, 7, 13, 19) and the first eight product families alphabetically, giving 32 series and 53,440 origin rows. The target is the sum of the next seven daily sales of a series. For BigMart the target is Item_Outlet_Sales, split into tertiles of the training portion of each random split. For Online Retail (541,909 invoice lines) we remove cancellations and rows with non-positive quantity or price, aggregate by StockCode and date (276,148 product-date rows for 3,922 products), and take the sum of the next seven observed product-date revenue records. That horizon is not seven calendar days. The cohort is the 25 products with the most product-date records. After requiring seven past and seven future records, it holds 6,981 origin rows.

Each temporal fold trains only on rows whose origin date and target end date both fall before the cutoff. Test rows are chosen by origin date inside the monthly test interval. Class boundaries are the training fold's one-third and two-thirds target quantiles, applied unchanged to the test set. This keeps test targets out of the thresholds and removes training labels whose future window crosses the cutoff.

Store Sales features are same-day sales, onpromotion, and store and family identifiers. BigMart features are item weight (missing values set to the median), fat content (labels harmonized), visibility, type, price (Item_MRP), outlet identifier, size (missing set to Unknown), location tier, outlet type and outlet age. Whether same-day sales are known at prediction time must be checked for any real deployment. If they are not, the features must be lagged or dropped and the experiments rerun. Online Retail features are same-day quantity, revenue, average unit price and StockCode. A second feature set uses only strictly past information: revenue and quantity summed over the seven records before the origin record, plus StockCode. All results depend on these definitions.

C. Models and Metrics

We compare a most-frequent dummy baseline, logistic regression, a decision tree and a random forest. The temporal runs use three monthly folds and three seeds (17, 42, 2026). For Store Sales the test months are May to July 2017, and a second analysis uses eight rolling monthly folds from December 2016 to July 2017. For Online Retail the test months are September to November 2011, and the rolling analysis uses eight folds from April to November 2011. BigMart uses five random 80/20 splits (seeds 11, 21, 42, 73, 101), and one variant groups the split by item. Numeric features are standardized and StockCode is one-hot encoded. Logistic regression runs up to 2,000 iterations, the tree uses scikit-learn defaults and the forest has 200 trees, or 100 for Store Sales (scikit-learn 1.8.0). No hyperparameters were tuned. We report accuracy and macro-F1. Means and standard deviations summarize the fold-seed records, and for rule-based baselines the folds. They are not confidence intervals over time. Fold-level accuracies are compared with a Wilcoxon signed-rank test. With eight folds the smallest possible two-sided p-value is 0.0078. Table III classifies the algorithms and gives the accuracy of each under the main protocol.

TABLE III
CLASSIFICATION AND COMPARISON OF THE ALGORITHMS USED

Algorithm	Family	How it predicts	Settings in this study	Strengths	Limitations	Mean accuracy		
						Store Sales	Online Retail	BigMart
Dummy (most frequent)	Chance baseline	Always predicts the most common training class	Strategy: most frequent	Sets the chance level	Ignores every feature	0.2520	0.2324	0.3451
Persistence rule	Naive time-series rule	Classifies the trailing seven-record sum with the training thresholds	No training; temporal data only	No fitting; hard to beat when series are stable	Needs history; cannot use other features	0.9868	0.6376	n/a
Series-identity rule	Lookup rule	Predicts the most common training class of the series (outlet for BigMart)	Majority class per series or outlet	Tests whether scale alone explains accuracy	Cannot handle new series; ignores time	0.9670	0.6199	0.4877
Logistic regression	Linear probabilistic model	Weighted sum of features turned into class probabilities	Numeric features standardized, categories one-hot encoded, up to 2,000 iterations, other defaults	Fast, stable, few parameters	Linear boundaries unless features are engineered	0.9539	0.6109	0.6876
Decision tree	Rule-based tree	Follows a sequence of threshold splits to a class	scikit-learn defaults, no depth limit	Readable; captures interactions	Overfits when unpruned; unstable	0.9703	0.5459	0.5886
Random forest	Ensemble of trees	Majority vote of trees trained on bootstrap samples	100 trees (Store Sales), 200 trees (Online Retail, BigMart), other defaults	Lower variance than one tree; non-linear	Harder to read; slower; more memory	0.9720	0.5710	0.6587

Accuracy under the main protocol: three temporal folds for Store Sales and Online Retail, five random splits for BigMart. The persistence rule needs a time order, so it is not defined for BigMart.

D. Validity Checks

Class boundaries are pooled across series of very different scale, so a model can score well just by recognizing which series a row belongs to. We therefore add three checks. A series-identity rule predicts the most common training class of each series (for BigMart, of each outlet). A persistence rule classifies the trailing seven-record sum, which includes the origin record, with the training thresholds. A per-series variant redefines the class boundaries as each series' own training quantiles (for BigMart, each outlet's), which removes scale. We ran the identity and persistence rules and the per-series labels on Store Sales (three folds) and Online Retail, and the identity rule and per-outlet labels on BigMart. The rolling per-series analysis was run for Online Retail only.

VI. RESULTS

A. Store Sales

Table IV gives the three main folds (May to July 2017) and eight rolling folds. The random forest and the decision tree reach 0.972 and 0.970 accuracy on the main folds, and logistic regression 0.954. Over the rolling folds the tree and the forest are indistinguishable (difference −0.0015, Wilcoxon $p = 0.148$), and the tree beats logistic regression in all 8 folds ($p = 0.0078$).

The baselines change the reading. A persistence rule that repeats the class of the trailing seven-day sum scored 0.9868 on the main folds and 0.9709 on the rolling folds, above every learned model (rolling folds: $p = 0.031$ against the tree and 0.039 against the forest, uncorrected for the number of comparisons). A rule that knows only the store-family series scored 0.9670 and 0.9646, and the tree does not differ from it ($p = 0.195$). Strictly lagged features scored about the same as same-day sales (Table VI), and calendar and promotion features alone reached 0.921 to 0.949.

TABLE IV
STORE SALES, GLOBAL CLASS BOUNDARIES

Method	3-fold		8-fold	
	Acc.	F1	Acc.	F1
Persistence	0.9868	0.9868	0.9709	0.9709
Random forest	0.9720	0.9721	0.9618	0.9619
Decision tree	0.9703	0.9704	0.9602	0.9604
Series identity	0.9670	0.9666	0.9646	0.9646
Logistic reg.	0.9539	0.9533	0.9258	0.9257
Dummy	0.2520	0.1342	0.2709	0.1420

Learned models and the dummy: mean over fold-seed runs. Rule baselines: mean over folds.

Table V uses each series' own training quantiles as class boundaries. The learned models fall to 0.48–0.57 accuracy, still above the one-in-three level, but persistence is higher at 0.65 and the series-identity rule drops to chance. Part of the signal in Table IV is therefore real, and most of it is scale and recent sales.

TABLE V
STORE SALES, PER-SERIES CLASS BOUNDARIES, 3 FOLDS

Method	Accuracy	Macro-F1
Persistence	0.6524	0.6316
Random forest	0.5683	0.5378
Decision tree	0.5541	0.5311
Logistic regression	0.4824	0.4515
Series identity	0.2717	0.2023
Dummy	0.2541	0.1351

Table VI compares feature sets on the three main folds. Same-day sales is not required, because strictly lagged sales scores as well. Calendar and promotion features alone are lower for the tree and forest but not for logistic regression, which again suggests that series identity carries much of the signal.

TABLE VI
STORE SALES FEATURE SETS, ACCURACY, 3 FOLDS

Feature set	Tree	Forest	Log. reg.
Same-day sales	0.9703	0.9720	0.9539
Lagged sales	0.9721	0.9696	0.9597
Calendar + promo	0.9211	0.9405	0.9492

B. Online Retail

Table VII gives the main protocol (three folds, September to November 2011) and the eight-fold rolling analysis. Under the main protocol logistic regression is the best learned model (0.6109 accuracy), ahead of the forest and the tree. Across the eight rolling folds it beat both in every fold (mean accuracy difference +0.039 against the forest and +0.065 against the tree, Wilcoxon $p = 0.0078$ for each).

The baselines change the reading. The persistence rule scored 0.6376 on the three main folds and 0.6019 on the rolling folds. The series-identity rule scored 0.6199 and 0.6002. Logistic regression is statistically indistinguishable from both (rolling folds: difference −0.003 and −0.002, won in 4 of 8 folds, $p = 1.00$ and 0.95). The forest is worse than persistence ($p = 0.023$). Replacing same-day features with strictly past ones raised logistic regression to 0.6189 over the rolling folds, which is still within the spread of the baselines.

TABLE VII
ONLINE RETAIL, GLOBAL CLASS BOUNDARIES

Method	3-fold		8-fold	
	Acc.	F1	Acc.	F1
Persistence	0.6376	0.6209	0.6019	0.5845
Series identity	0.6199	0.6121	0.6002	0.5950
Logistic reg.	0.6109	0.6020	0.5985	0.5938
Random forest	0.5710	0.5665	0.5592	0.5548
Decision tree	0.5459	0.5400	0.5334	0.5281
Dummy	0.2324	0.1253	0.2314	0.1248

Learned models and the dummy: mean over fold-seed runs (3 folds × 3 seeds; 8 folds × 3 seeds). Rule baselines: mean over folds.

Table VIII removes product scale by using each product's own training quantiles as class boundaries. Every learned model then drops to between 0.34 and 0.38 accuracy over the rolling folds, near the one-in-three level of a balanced three-class problem. Persistence is the only method clearly above that level (0.4671). Most of the accuracy in Table VII therefore comes from product scale.

TABLE VIII
ONLINE RETAIL, PER-PRODUCT CLASS BOUNDARIES, 8 ROLLING FOLDS

Method	Accuracy	Macro-F1
Persistence	0.4671	0.4151
Logistic regression	0.3791	0.3294
Random forest	0.3506	0.3400
Decision tree	0.3433	0.3355
Series identity	0.3113	0.2513
Dummy	0.2443	0.1292

C. BigMart

BigMart has no time dimension, so the persistence rule does not apply. Logistic regression is the best model with a random split (0.6876 accuracy, 0.6880 macro-F1), well above an outlet-identity rule (0.4877), and grouping the split by item changes little (Table IX). With per-outlet class boundaries, logistic regression still reaches 0.5982 against 0.3453 for the identity rule, so the models do learn item-level signal here. Item price carries most of it: price alone gives 0.5446 for logistic regression, and without price the best model (a tree, 0.5105) is only level with the outlet rule (0.4877). Default trees overfit. A depth of 4 or 5 raised the tree to about 0.69, and a forest of depth 10 reached 0.6865, but those depths were chosen on the same splits, so they are optimistic.

TABLE IX
BIGMART, MEAN ACCURACY OVER 5 RANDOM SPLITS

Method	Random	By item	Per outlet
Logistic regression	0.6876	0.6900	0.5982
Random forest	0.6587	0.6596	0.5701
Decision tree	0.5886	0.5960	0.5117
Outlet identity	0.4877	0.4885	0.3453
Dummy	0.3451	0.3244	0.3482

VII. BENCHMARK PROTOCOL (NOT YET RUN)

No cloud measurement is reported here. This section fixes the plan in advance so later results can be read against it. Numeric settings are proposals to be finalized against endpoint limits and budget.

A. Systems

System S is the proposed path: API Gateway, Lambda and SageMaker Serverless Inference. System B uses the same API Gateway and Lambda code in front of a SageMaker real-time endpoint on one fixed instance type. Both use the same model artifact, container, payload schema and region. Keeping the Lambda tier identical isolates the effect of the hosting mode, following the serverless-versus-real-time comparison in AWS's own toolkit [23]. We record region, memory size, concurrency limits, instance type, SDK and container versions and the model hash for every run. The real-time endpoint is not serverless and serves only as the comparator.

B. Workloads

- Sequential single-record requests for warm latency without queueing.
- Cold-start trials after fixed idle intervals. Cold status comes from provider telemetry where available, otherwise from a documented latency rule.
- Steady load at several concurrency levels (proposed: 1, 5, 10, 25, 50) and one burst profile.
- Payloads of 1, 10 and 100 records (proposed).

S and B run in interleaved, randomized order over several days. Warm-up requests are discarded and counted. Each condition is repeated as independent runs, and the run is the unit of analysis. Before any latency comparison, a prediction-agreement check confirms that S and B return identical classes on a held-out set.

C. Metrics and Analysis

We measure client-side end-to-end latency and, where possible, split it into API Gateway, Lambda and endpoint parts using CloudWatch. Latency is summarized by median, p95 and

p99. We report p99 only for conditions with enough requests (we propose at least 1,000). Throughput is successful requests per second. Error rate counts client errors, server errors, throttles and timeouts. Cost per successful prediction follows Section IV-B and uses billing line items when available, or a price calculation that states its date and region. Idle cost is reported separately.

Latencies are skewed, so the plan uses medians with percentile-bootstrap confidence intervals, the Hodges–Lehmann shift between systems, a Mann–Whitney U test on run-level summaries and Cliff's delta as effect size, with Holm correction across metrics. A claim that S improves a metric would need a measured difference that meets this plan.

VIII. Discussion and Limitations

The baselines are the main result of the study. On both temporal datasets, a rule that repeats last week's class does as well as or better than the best learned model, and a rule that only knows the product does almost as well. With per-product class boundaries, the Online Retail models sit near chance. The classifiers therefore mostly recover how large each product is and how it sold recently. This does not make classification useless for decision support, but it means the target definition matters, and pooled quantile labels can make a weak model look strong.

BigMart is different. Its models beat the outlet rule and still learn above the per-outlet baseline, mostly from item price. The rankings do not carry over between datasets: logistic regression led on Online Retail and BigMart, and the tree and forest led on Store Sales, where they were indistinguishable from each other. We do not claim that one model is better in general.

The architecture gives a concrete way to combine inference with managed event handling and monitoring, but its advantages are hypotheses until measured. Earlier work gives reasons for caution in both directions. Serverless training was not significantly cheaper than IaaS in one controlled study [28], and cold-start cost depends on runtime, trigger and resource allocation [20]. Section VII is built to produce this evidence for the SalesSense workload.

Main limitations:

- Small cohorts (four stores, eight families, 25 products) do not represent the full datasets.
- The cohort rule, test months and model settings for Store Sales and Online Retail were fixed for this analysis. Earlier runs of the same experiments gave somewhat different numbers, for example logistic regression on Store Sales at 0.9186 instead of 0.9539. The values reported here come from the scripts released with the paper.
- No hyperparameters were tuned. Depth-limited trees on BigMart were chosen on the same splits and are optimistic. The Store Sales forest uses 100 trees, and the other datasets use 200.
- Store Sales has eight rolling folds, and seeds do not add independent time periods. P-values are not corrected for multiple comparisons.
- The rolling per-series analysis was run for Online Retail only.
- Class boundaries are quantiles of continuous targets. Their business meaning needs stakeholder review.
- Online Retail uses seven observed records, not seven calendar days.
- No AWS deployment or operational benchmark exists.
- The related-work search was a focused pilot, not a systematic review.

IX. Conclusion

We proposed SalesSense, an event-driven serverless architecture for product-sales classification and decision support, and specified a benchmark against an always-on baseline. Experiments on three public retail datasets support only preliminary statements about prediction quality. On the temporal datasets, naive baselines matched or beat the learned models, and on Online Retail per-product class boundaries brought every learned model close to chance. On BigMart the models learn real signal, mostly from item price. The evidence does not support claims about latency, elasticity, cold starts, reliability, scalability or cost.

The next steps are to deploy the architecture and run the Section VII benchmark, consider target definitions that remove product scale, test on larger cohorts and the full datasets, and release the data provenance and code. No formal hypothesis is tested in this study.

Acknowledgment

[Funding, conflict-of-interest, author-contribution, data-availability and AI-use statements to be completed by the authors. Data sources: Kaggle Store Sales Time Series Forecasting competition; UCI Online Retail dataset; Analytics Vidhya BigMart Sales Prediction. Check each source terms before release.]

References

[1] S. Batrel, A. M. Brahme, and J. Gaikwad, "A comprehensive overview of data mining algorithms," Library Progress Int., 2024. [Online]. Available: https://www.researchgate.net/publication/384562785
[2] P. Wadkar and S. D. Mundhe, "Comparative study of different machine learning algorithms used for credit card fraud detection," Int. J. Res. Finance Manage., 2025. [Online]. Available: https://www.researchgate.net/publication/396213597
[3] S. R. Misal, T. K. Kathane, and S. D. Mundhe, "A study on credit card fraud detection using machine learning," 2020. [Online]. Available: https://www.researchgate.net/publication/376833594
[4] S. R. Misal, P. Wadkar, and S. D. Mundhe, "Comparative analysis of different machine learning algorithms used in breast cancer prediction," 2023. [Online]. Available: https://www.researchgate.net/publication/376833481
[5] N. Mandhane and P. Wadkar, "Sentiment analysis of Amazon reviews datasets using machine learning and deep learning algorithms," Int. J. Res. Publ. Rev., 2025. [Online]. Available: https://www.researchgate.net/publication/396697030
[6] N. Pachpor, A. M. Brahme, A. Labade, et al., "Rainfall forecasting and analysis using machine learning models," conference paper, 2025. [Online]. Available: https://www.researchgate.net/publication/403151930
[7] N. P. Ganeshar, "Multiple disease prediction using machine learning," conference paper, 2024. [Online]. Available: https://www.researchgate.net/publication/388743082
[8] T. M. Sonawane and V. N. Ugare, "Docker containers: Revolutionizing software deployment and infrastructure management," conference paper, 2025. [Online]. Available: https://www.researchgate.net/publication/398858555
[9] V. N. Ugare, "Challenges in seamless remote connectivity between on-premise systems and cloud platforms," conference paper, 2026. [Online]. Available: https://www.researchgate.net/publication/408539621
[10] S. Batrel, A. M. Brahme, and J. Gaikwad, "An examination of impact of cloud computing and Kubernetes on data availability: An empirical approach," Afr. J. Biol. Sci., 2024. [Online]. Available: https://www.researchgate.net/publication/380574477

[11] T. K. Kathane and S. D. Mundhe, "Stateful database cluster migration across multi-cloud ecosystems," YASHOMANTHAN, 2026. [Online]. Available: https://www.researchgate.net/publication/408534828
[12] T. M. Sonawane and V. N. Ugare, "Secured data communication in green cloud computing through blockchain and IoT integration," conference paper, 2026. [Online]. Available: https://www.researchgate.net/publication/399825833
[13] S. R. Misal, A. M. Brahme, and S. D. Mundhe, "Big data analytics: Applications, tools, challenges and future innovations," 2023. [Online]. Available: https://www.researchgate.net/publication/376833474
[14] K. S. Kumar, V. Gupta, Y. Patil, G. Jindal, N. R. Cheepurupalli, and M. M. Bhanushali, "Impact of AI on operations management," in Recent Trends in Engineering and Science for Resource Optimization and …, 2025. [Online]. Available: https://scholar.google.com/citations?hl=en&user=Ini2yjoAAAAJ
[15] H. B. Hassan, S. A. Barakat, and Q. I. Sarhan, "Survey on serverless computing," J. Cloud Comput., vol. 10, Art. 39, 2021, doi: 10.1186/s13677-021-00253-7.
[16] H. Shafiei, A. Khonsari, and P. Mousavi, "Serverless computing: A survey of opportunities, challenges, and applications," ACM Comput. Surv., vol. 54, no. 11s, Art. 239, 2022, doi: 10.1145/3510611.
[17] J. Scheuner and P. Leitner, "Function-as-a-Service performance evaluation: A multivocal literature review," J. Syst. Softw., vol. 170, Art. 110708, 2020, doi: 10.1016/j.jss.2020.110708.
[18] Z. Li et al., "The serverless computing survey: A technical primer for design architecture," ACM Comput. Surv., vol. 54, no. 10s, Art. 220, 2022, doi: 10.1145/3508360.
[19] A. Ebrahimi, M. Ghobaei-Arani, and H. Saboohi, "Cold start latency mitigation mechanisms in serverless computing: Taxonomy, review, and future directions," J. Syst. Archit., vol. 151, Art. 103115, 2024, doi: 10.1016/j.sysarc.2024.103115.
[20] A. Joosen et al., "Serverless cold starts and where to find them," in Proc. 20th Eur. Conf. Comput. Syst. (EuroSys), 2025, doi: 10.1145/3689031.3696073.
[21] A. Ali, R. Pinciroli, F. Yan, and E. Smirni, "BATCH: Machine learning inference serving on serverless platforms with adaptive batching," in Proc. Int. Conf. High Perform. Comput., Netw., Storage Anal. (SC20), 2020, doi: 10.1109/SC41405.2020.00073.
[22] Amazon Web Services, "Pattern 1: Serverless ML inference pipeline," AWS Prescriptive Guidance. Accessed: Oct. 3, 2026. [Online]. Available: https://docs.aws.amazon.com/prescriptive-guidance/latest/agentic-ai-serverless/pattern-serverless-ml-inference.html
[23] Amazon Web Services, "Introducing the Amazon SageMaker Serverless Inference Benchmarking Toolkit," AWS Machine Learning Blog. Accessed: Oct. 3, 2026. [Online]. Available: https://aws.amazon.com/blogs/machine-learning/introducing-the-amazon-sagemaker-serverless-inference-benchmarking-toolkit/
[24] A. Barrak, F. Petrillo, and F. Jaafar, "Serverless on machine learning: A systematic mapping study," IEEE Access, vol. 10, pp. 99337–99352, 2022, doi: 10.1109/ACCESS.2022.3206366.
[25] A. Aslani and M. Ghobaei-Arani, "Machine learning inference serving models in serverless computing: A survey," Computing, vol. 107, Art. 47, 2025, doi: 10.1007/s00607-024-01377-9.
[26] N. Mahmoudi and H. Khazaei, "Performance modeling of serverless computing platforms," IEEE Trans. Cloud Comput., vol. 10, no. 4, pp. 2834–2847, 2022, doi: 10.1109/TCC.2020.3033373.
[27] Q. L. Trieu, B. Javadi, J. Basilakis, and A. N. Toosi, "Performance evaluation of serverless edge computing for machine learning applications," in Proc. IEEE/ACM 15th Int. Conf. Utility Cloud Comput. (UCC), 2022, pp. 139–144.
[28] J. Jiang et al., "Towards demystifying serverless machine learning training," in Proc. Int. Conf. Manage. Data (SIGMOD), 2021.
[29] S. Ma and R. Fildes, "Retail sales forecasting with meta-learning," Eur. J. Oper. Res., vol. 288, no. 1, pp. 111–128, 2021, doi: 10.1016/j.ejor.2020.05.038.
[30] G. Chaubey, P. R. Gavhane, D. Bisen, and S. K. Arjaria, "Customer purchasing behavior prediction using machine learning classification techniques," J. Ambient Intell. Humaniz. Comput., 2022, doi: 10.1007/s12652-022-03837-6.
[31] M. S. Al-Batah, M. S. Alzboon, and H. Zureigat, "Predictive mathematical modeling and classification of retail sales orders using AI machine learning techniques," J. Comput. Sci., vol. 22, no. 3, pp. 878–885, 2026, doi: 10.3844/jcssp.2026.878.885.
[32] A. Kumar, N. Janani, M. Aashiq, and R. Mahesh, "SalesForge – Product sales analytics & demand forecasting system," Int. J. Res. Appl. Sci. Eng. Technol., vol. 14, no. IV, Apr. 2026, doi: 10.22214/ijraset.2026.79504.