WHITE PAPER · MARCH 2026

Beyond gradient boosting.

Why a fine-tuned tabular foundation model outperforms legacy machine learning for SKU–store demand forecasting in retail and consumer brands.

PACX.AI · 28 MIN READ · 11 REFERENCES · PUBLISHED 4 MAR 2026

Read online · https://pacxai.com/white-papers/foundation-model-demand-forecasting

KEY FIGURES

82%

forecast accuracy at item × store level on a monthly horizon, measured out-of-time as 1 − WAPE in a live PACX deployment. Section 05

~3%

gain of the winning M5 machine-learning models over the best statistical benchmark at product–store level. Makridakis et al., 2022

$1.7T

annual global cost of inventory distortion — out-of-stocks plus overstocks. IHL Group, 2026

65.6%

of that cost comes from empty shelves rather than from excess stock. IHL Group, 2026

This paper describes the forecasting approach used in the PACX Inventory Planning platform. Performance figures are drawn from PACX deployments and internal benchmarking; third-party figures are cited to their original sources in the references.

CONTENTS

00 Executive summary

01 The cost of getting demand wrong

02 Why legacy machine learning has plateaued

03 A different starting point

04 Methodology

05 Results

06 What this means for planners

07 What a retailer needs

08 Limitations

09 Conclusion

– References

00 EXECUTIVE SUMMARY

The gains went where they matter least.

Retail has spent a decade moving demand forecasting from spreadsheets and exponential smoothing to gradient-boosted machine learning. That shift delivered real gains — but the gains were concentrated where they matter least.

The most rigorous public test of retail forecasting to date, the M5 competition on Walmart data, found that the winning machine-learning models beat the best statistical benchmark by around 40 percent at the total-company level but by only about 3 percent at the level of an individual product in an individual store — the level at which allocation, replenishment and markdown decisions are actually made.[5]

Meanwhile the cost of getting those decisions wrong has not gone away. IHL Group puts the global cost of inventory distortion — out-of-stocks plus overstocks — at roughly $1.7 trillion a year, about 6.5 percent of global retail sales, with two-thirds of that cost coming from empty shelves.[1, 2]

PACX takes a different starting point. Instead of training a gradient-boosting model from scratch on each client's history, PACX forecasts with a tabular foundation model — a transformer pre-trained on tens of millions of synthetic datasets so that it arrives already knowing how tabular data behaves — fine-tuned specifically on retail and consumer-brand data. For every item–store and snapshot date the model receives a summary of that item–store's own recent history (six months by default), the product's attributes and the calendar of the period being forecast, and predicts what it will sell over the next one or two months — built the same way at training time and at prediction time.

THE RESULT

On a live multi-store retail deployment, this approach reached 82 percent forecast accuracy at the item × store level on a monthly horizon, measured out-of-time as one minus weighted absolute percentage error (WAPE) — the hardest level in the hierarchy, and the one the M5 results showed conventional machine learning struggles to move.

WHAT THIS PAPER COVERS

Section 02

Why gradient boosting plateaus

Why gradient-boosted models plateau at the SKU–store level, and why the plateau is structural rather than a tuning problem.

Section 03

What a tabular foundation model is

What the published evidence says about it, and what retail-specific fine-tuning adds.

Section 04

The four-stage curation method

How PACX builds training examples that mirror the prediction task exactly, with a built-in guard against leakage, and how each history window becomes model inputs.

Section 05

Results, limits, and what they mean in practice

The out-of-time result and how it was measured, what it means for planners, what a retailer needs in order to run it — and what the paper does not claim.

Every inventory decision starts from a forecast.

Every inventory decision in a retail business — how much to buy, where to allocate it, when to replenish, when to mark it down — starts from a forecast. When the forecast is wrong in one direction the shelf is empty and the sale is lost; when it is wrong in the other direction the stock sits, ages, and is eventually cleared at a discount.

IHL Group's annual studies of this problem, which they call inventory distortion, estimated the combined global cost at $1.73 trillion in 2025, equivalent to about 6.5 percent of global retail sales, and $1.7 trillion in the 2026 study.[1, 2] Asia-Pacific accounts for the largest regional share at roughly $642 billion.[1]

$1.73T
global cost of inventory distortion in the 2025 study.[1]

6.5%
of global retail sales, the equivalent share.[1]

$642B
in Asia-Pacific, the largest regional share.[1]

$1.7T
global cost of inventory distortion in the 2026 study.[2]

The split matters for how forecasting should be judged. In IHL's 2026 study, out-of-stocks account for 65.6 percent of the cost and overstocks for 34.4 percent.[2] Empty shelves are the single largest component.

Figure 1. Where the cost of inventory distortion comes from: out-of-stocks 65.6 percent, overstocks 34.4 percent of the global total. Source: IHL Group, 2026 Inventory Distortion Study.[2]

A forecasting system that is accurate on aggregate but biased low on the individual items and stores that actually run out is solving the wrong problem.

The prize for closing the gap is well documented.

McKinsey & Company[4]

20–50%
lower forecast error with AI-driven forecasting in supply-chain settings.

up to 65%
fewer lost sales and less product unavailability.

5–10%
lower warehousing costs.

IHL Group[1]

2.3×
faster sales growth for retailers using AI-driven inventory management, against traditional peers.

2.5×
faster profit growth for the same retailers.

< 25%
of retailers had deployed AI or machine learning in the areas most exposed to inventory distortion.

Gartner[3]

70%
of large organisations will have adopted AI-based demand forecasting by 2030.

Data
incomplete or inaccessible data is one of the main barriers holding the rest back.

The question for a retail or consumer-brand executive, then, is not whether to use machine learning for demand forecasting. It is which kind — and whether the kind most vendors ship today can actually reach the item–store level where the money is lost.

The plateau is structural, not a tuning problem.

Gradient-boosted decision trees — XGBoost, LightGBM, CatBoost — became the default engine for retail forecasting for good reasons. They handle mixed numeric and categorical data, tolerate missing values, and train quickly.

Their high-water mark was the M5 forecasting competition, run on 42,840 Walmart sales series covering 3,049 products in 10 stores over roughly five and a half years. All five winning entries were built on gradient boosting, mostly LightGBM, and it was the first M competition in which every top method was a pure machine-learning method that beat every statistical benchmark.[5]

THE M5 COMPETITION AT A GLANCE

42,840 sales series	**3,049** products	**10** stores
~5.5 yrs of history	**5 of 5** winners on gradient boosting	

Source: Makridakis, Spiliotis and Assimakopoulos, 2022[5]

But the M5 organisers' own analysis contains the finding that should concern anyone running a store-level business. The winning entry improved on the best statistical benchmark by 22.4 percent overall. That improvement was about 40 percent at the total-company level and shrank to roughly 3 percent at the individual product–store level.[5]

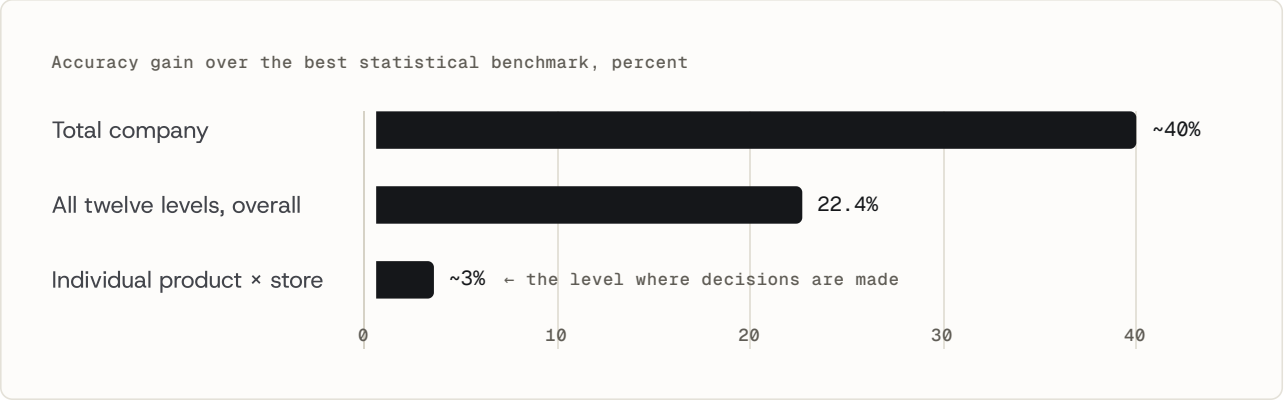

Figure 2. Accuracy gain of the winning M5 model over the best statistical benchmark, by level of the retail hierarchy. The gain collapses at the product–store level. Source: Makridakis, Spiliotis and Assimakopoulos, International Journal of Forecasting.[5]

The models were excellent at learning company-wide seasonality and calendar effects. They were barely better than exponential smoothing at predicting what one shoe, in one store, would sell next month.

Four structural reasons

There are four structural reasons for this, and none of them is fixed by better hyperparameter tuning.

1

Every model starts from zero

A gradient-boosted model knows nothing until it has seen the client's own data. It has no prior notion that sell-through decays with stock age, that a new colourway of an existing style behaves like its siblings, or that a size run in one store resembles the same size run in a similar store. Every one of those regularities has to be rediscovered from the client's history, for every client, every time the model is retrained. Retailers with two or three years of clean data can afford this; most cannot.

2

Sparse, intermittent series

At the item–store level most series are short and most months are near zero. The M5 authors noted that intermittency was the defining feature of the lower hierarchy levels and the main reason gains were so small there.[5] Tree ensembles fit these series by averaging, which systematically under-predicts the items that are about to sell out — precisely the out-of-stock component that makes up two-thirds of inventory-distortion cost.

3

The cold-start problem

A new product, a new store, or a product newly ranged into a store has no history for a per-client model to learn from. Gartner singles out the ability to learn from diverse datasets and handle new product introductions with limited history as one of the defining advantages of AI-based forecasting over conventional statistical approaches[3] — but a model trained only on one client's past cannot have that advantage, because it has never seen a launch it was not trained on.

4

Hand-built features and retraining cycles

Gradient-boosting pipelines depend on feature engineering: lags, rolling means, holiday flags, price ratios. Each is a modelling decision that must be maintained, and each is an opportunity for leakage — features that accidentally encode information from the period being predicted. The M5 winners combined hundreds of models across aggregation levels to get their result;[5] that is not an architecture most retailers can operate.

A NOTE ON TIME-SERIES FOUNDATION MODELS

Time-series foundation models such as Chronos, TimesFM and Moirai emerged as one answer, but they were built for univariate signals. Independent reviews note that most first-generation time-series foundation models handle only the sales history itself, that static categorical attributes such as style, colour or price tier remain a challenge for them, and that their optimisation objective cannot be adapted after pre-training[11] — a poor fit for retail, where the attributes of the product and the state of its stock carry much of the signal.

03 A DIFFERENT STARTING POINT

A model that arrives already knowing how tables behave.

Large language models changed software because they arrived pre-trained: they had already learned the structure of language before seeing any particular task. The same idea has now reached tabular data — the rows-and-columns data that runs retail.

What a tabular foundation model is

A tabular foundation model is a transformer that is pre-trained not on one dataset but on an enormous population of them — in the case of TabPFN, published in Nature in January 2025, on roughly 100 million synthetic datasets generated from structural causal models.[6] Through that pre-training the model learns, in general, how columns relate to targets.

When it is given a new table it does not train in the conventional sense; it reads the labelled rows as context and predicts the unlabelled rows in a single forward pass, a mechanism called in-context learning. In the Nature paper this approach outperformed an ensemble of the strongest gradient-boosting baselines, tuned for four hours, in 2.8 seconds.[6]

~100M

synthetic datasets in TabPFN's pre-training.[6]

2.8 s

to outperform a gradient-boosting ensemble tuned for four hours.[6]

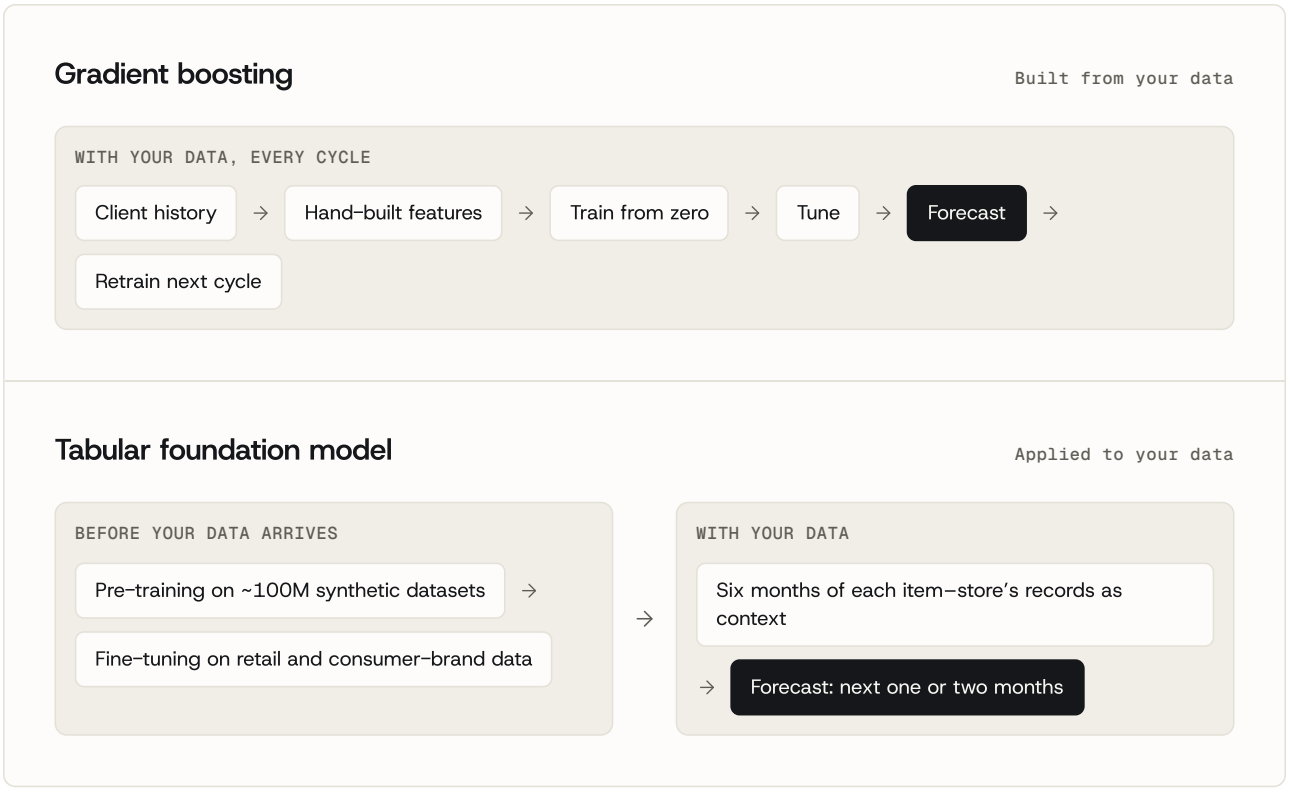

Figure 3. Two routes to a SKU–store forecast. Gradient boosting is built from the client's data, cycle after cycle; a fine-tuned tabular foundation model is applied to it, reading recent history as context.

The field is moving quickly

JANUARY 2025

TabPFN, in Nature

A tabular foundation model outperforms tuned gradient-boosting ensembles on small data — in seconds rather than hours.[6]

2025

TabPFN extended to time series

Paired with a small set of temporal features, an 11-million-parameter tabular model reaches state-of-the-art accuracy on covariate-informed forecasting and is competitive on the GIFT-Eval and fev-bench benchmarks, with no time-series-specific pre-training.[8]

FEBRUARY 2026

TabICLv2

Scales to datasets of a million rows and surpasses the previous best model on the TabArena and TALENT benchmarks without any tuning. Its authors describe gradient-boosted trees as dethroned at the top of tabular benchmarks.[7]

EARLY 2026

Sixteen models and counting

More than sixteen tabular foundation models have been released, and the state of the art is changing every few weeks.[10]

WHY THIS MATTERS FOR RETAIL, IN ONE SENTENCE

A tabular foundation model arrives already knowing how tables behave; a gradient-boosted model has to learn it from your data alone. For sparse item–store series and new launches, that prior knowledge is the difference between a forecast and a guess.

What retail-specific fine-tuning adds

General-purpose pre-training gives the model a prior about tables. It does not give it a prior about retail. PACX has fine-tuned a tabular foundation model on retail and consumer-brand data so that the model's starting assumptions match the domain:

how demand relates to stock on hand and stock age	how sell-through in the previous month predicts sell-through in the next
how movement class, price tier and product attributes such as division, style and colour interact with seasonality	how a newly launched item behaves relative to its established siblings

The result is a model that is applied to each client's data rather than built from it. When a new retailer is onboarded, the model reads that retailer's recent history as context and forecasts immediately; it does not need to rediscover the mechanics of retail from that retailer's data. When a new item is launched, its attributes and the behaviour of comparable items in the fine-tuning population give the model something to reason from on day one.

> We describe the model class here rather than the specific base model, which is an internal implementation detail that may change as the field advances. The methodology in the next section is what makes the approach work, and it is independent of the base model.

	GRADIENT BOOSTING, TRAINED PER CLIENT	TABULAR FOUNDATION MODEL, FINE-TUNED FOR RETAIL
Starting knowledge	None. Every regularity is learned from the client's history alone.	A prior about tables from pre-training, and a prior about retail from fine-tuning.
Sparse item–store series	Fitted by averaging, which under-predicts the items about to sell out.	Read against a retail prior: how stock, sell-through and price behave for comparable items, not only this one.
New items and stores	No history, so the forecast falls back to a category average.	Forecast from attributes and comparable items on day one.
Features	Hand-built lags, rolling means and flags, maintained per client.	The same window statistics derived automatically for every client field; a new signal is one query change.
Retraining	Retrained per client and per cycle, with hyperparameter search.	Applied to each client's data; no per-client training or search.
Evaluation versus production	Separate feature pipelines can drift apart.	The same four stages run at training and at prediction time.

04 METHODOLOGY

Training examples with exactly the shape of the question.

A foundation model is only as good as the examples it is shown. Most of the engineering in PACX's forecasting pipeline is not in the model; it is in constructing training examples that have exactly the shape of the question the business will ask at prediction time.

Two layers do that work. Four declarative SQL stages, generated from fixed templates by PACX's model builder and visible as cells the client's data team can read, define what is predicted, for which item–store pairs, on which dates, with which label and which history. A feature engine then turns each history window into model inputs with the same code at training time and at prediction time.

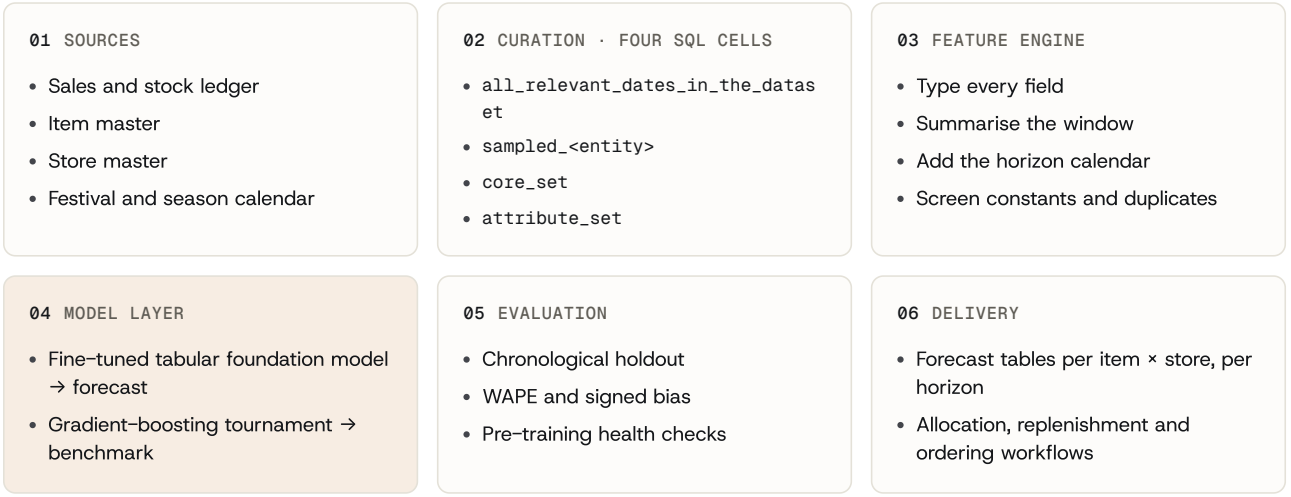

Figure 4. The forecasting pipeline end to end. Sources are curated into snapshots by four SQL cells, summarised by the feature engine, forecast by the fine-tuned foundation model and evaluated against a gradient-boosting tournament before the forecast tables reach the planning workflows.

Stage 1

Forecast calendar

`all_relevant_dates_in_the_dataset`

A grid of snapshot dates at the planning frequency, from the first to the last observed date

Produces

A complete date grid over the data range, weekly or monthly

Protects against

Missing periods read as missing data instead of zero demand

↓

Stage 2

Forecast entities

`sampled_<entity>`

Every item × store pair, paired with every grid date from its first activity onward

Produces

Item × store pairs from first activity onward

Protects against

Rows before an item existed; unexplained gaps before delistings

↓

Stage 3

Core set

`core_set`

Snapshot date plus next-horizon demand as the label, with a leakage guard on the horizon

Produces

Snapshot date + next-horizon demand label; completeness boundary

Protects against

Temporal leakage; undercounted labels at the end of the data

↓

Stage 4

Attribute set

`attribute_set`

The item–store's own records from the months strictly before each snapshot — six by default

Produces

The entity's own records strictly before the snapshot, with the product's attributes

Protects against

Any future information reaching the model input

Figure 5. The four curation cells. Stages 1–3 define what is predicted and when; stage 4 supplies the history the feature engine summarises. The same four cells run at inference time with the snapshot set to today.

STAGE 1 · ALL_RELEVANT_DATES_IN_THE_DATASET

Forecast calendar

The pipeline first scans the client's transactional ledger for its earliest and latest dates and expands that range into a complete calendar at the forecasting frequency — for a monthly objective, the first day of every month between the two; for a weekly objective, every week. This grid, rather than the dates on which transactions happened to occur, becomes the backbone of the dataset.

A month in which an item sold nothing is still a month, and it needs to be present so that zero demand is observed rather than silently absent.

STAGE 2 · SAMPLED_<ENTITY>

Forecast entities

Next, the pipeline defines what is being forecast: every combination of item and store (branch) in the ledger. For each pair it records the date of first activity and pairs the entity with every grid date from that date onward.

Two things follow. An item–store pair never has a row before it existed, so the model is never asked to learn from a period in which the product was not yet ranged. And a pair that goes quiet still has rows, so the model sees the run of zero-sale months that precedes a delisting or a stock-out rather than an unexplained gap.

STAGE 3 · CORE_SET

Core set

The core set turns entities and dates into supervised training examples. Each row is a snapshot: an item, a store and a sampled date. The label is the quantity that item sold in that store in the window immediately after the snapshot — the next one month for a one-month objective, the next two months for a two-month objective. A snapshot with no sales in that window is labelled zero, not left blank.

Because the label is an aggregate over a horizon rather than a single point, the model learns the quantity a planner needs — how many units to have on hand for the period — instead of a daily series that has to be summed afterwards.

TWO SAFEGUARDS, ENFORCED IN THE DATA

The guard against temporal leakage

1 **Strictly after, and no later than**

The label is summed only over records strictly after the snapshot date and no later than snapshot plus horizon, so no sale from the snapshot month itself can bleed into the label.

2 **The completeness boundary**

Sampled dates are capped at the last observed date minus the horizon. Any snapshot whose future window would run past the end of the data is excluded, because its label would be an undercount that the model would learn to imitate.

329

published papers across 17 fields with leakage-driven errors.[9]

This is the guard against what the reproducibility literature calls temporal leakage. Kapoor and Narayanan's survey found leakage-driven errors in 329 published papers across 17 fields, producing results they describe as wildly over-optimistic[9] — the same failure mode that produces forecasting pilots which look excellent in backtest and disappoint in production.

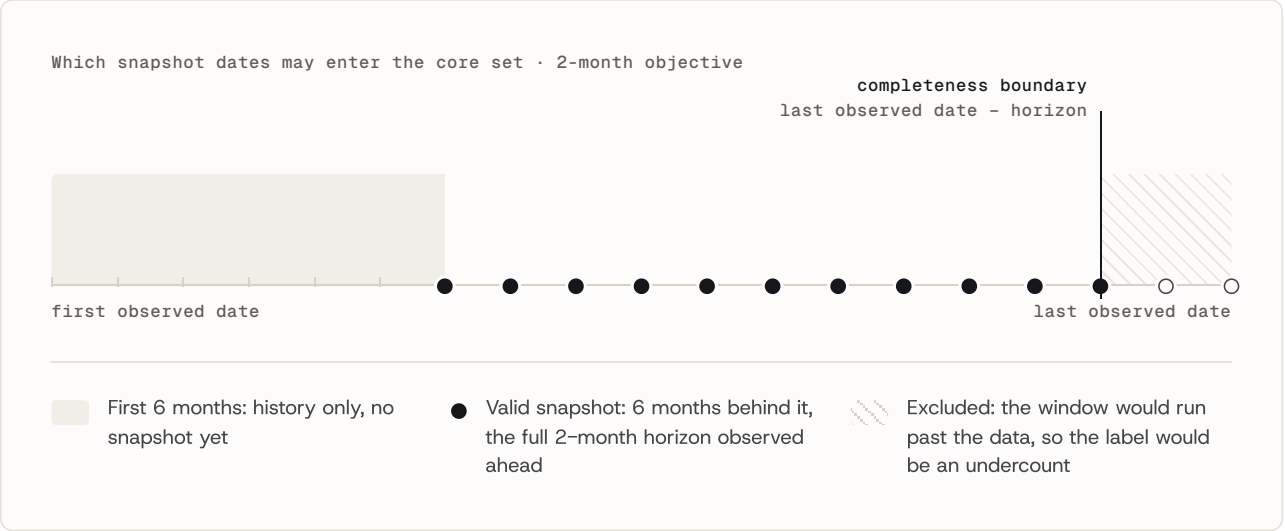

Figure 6. The completeness boundary for a two-month objective. Snapshots are sampled only where the full horizon is observed; the final months of the data are never used as snapshots, because their labels would be undercounts. The first six months supply history only.

STAGE 4 · ATTRIBUTE_SET

Attribute set

Finally, each snapshot is joined to the item–store's own records from the window strictly before the snapshot date — six months by default, adjustable per objective. This window is the raw material for the model input.

It carries the quantities and values that describe the state of the business at that moment — opening stock, stock in, sales, stock out, closing stock, stock age, sell-through and its rolling average, movement class, months at branch and consecutive zero-sale months — together with the attributes of the product: division, style and sub-style, colour, gender, size and pack, MRP and price tier, and whether the item is a new launch. Nothing from on or after the snapshot date is included. The attribute set is deliberately raw: the records themselves, not statistics. The statistics come next.

● Attribute set · six months before the snapshot — Women's block heel · Tan · 38 · Indiranagar

Month	Opening	Stock in	Sales	Stock out	Closing	Sell-through	3-mo avg	Age (mo)	Class	Zero months
M-6	18	24	11	0	31	26%	24%	2	Medium	0
M-5	31	0	9	0	22	29%	25%	3	Medium	0
M-4	22	12	14	0	20	41%	32%	4	Medium	0

Month	Opening	Stock in	Sales	Stock out	Closing	Sell-through	3-mo avg	Age (mo)	Class	Zero months
M−3	20	0	7	4	9	35%	35%	5	Medium	0
M−2	9	18	12	0	15	44%	40%	6	Fast	0
M−1	15	0	10	0	5	67%	49%	7	Fast	0

— snapshot

M+1	Label in training: units sold next month · Forecast in production	9 u

Division Women's footwear	Style Block heel	Sub-style Closed toe
Colour Tan	Gender Women	Size · pack 38 · single
MRP ₹2,499	Price tier Mid	New launch No

Illustrative reconstruction · fictional data · one snapshot of one item × store pair

Figure 7. What the attribute set holds for one snapshot: six monthly rows of one item in one store, plus the product's attributes, and the horizon that becomes the label. The feature engine summarises these rows before any model sees them. Illustrative reconstruction with fictional data.

From window to features: the feature engine

The attribute set is a stack of records per snapshot; a model needs one row per snapshot. The feature engine makes that row, on Spark, with rules that apply to every field the client provides — there is no per-client feature list to design.

1

Type every field

Each column is classified from its data type and cardinality: continuous measure, numeric category, category, or date. Free-text identifiers with thousands of distinct values are dropped, so a barcode or invoice number can never masquerade as a signal.

```
sales → continuous · price_tier → category · txn_date → date · invoice_no → dropped
```

2

Summarise the window

For each item × store and snapshot, the records in the window collapse into statistics: sum, mean, minimum, maximum, spread and median for measures; most frequent value and distinct count for categories; and, for dates, the recency of each record relative to the snapshot and the mix of weekdays, months and quarters. A record count says how much history the entity has.

```
sum(sales) · avg(closing_stock) · max(stock_age) · mode(movement_class) · min(days_since_record)
```

3

Add the calendar of the horizon

The months being forecast are known in advance, so the festivals and season that fall inside the horizon are added as flags and counts — Diwali or Pongal in the target window is a fact about the future the model is allowed to know.

```
target_festival_count · target_is_diwali · target_season
```

4

Screen the result

Constant columns are removed, and where two features move together almost perfectly one is dropped. The feature table that remains is the model input — and the same code produces it again, from the same four cells, at prediction time.

```
variance ≤ 0.01 → drop · |correlation| > 0.95 → keep one
```

Figure 8. The feature engine, applied to every field the ledger carries. One rule set turns the six-month window into one row per item × store and snapshot; the same rules run at prediction time.

> External drivers that a client holds — a promotion calendar, competitor prices, weather — enter through the attribute query as ordinary fields and are summarised the same way. The festival and season calendar is the one external source PACX maintains itself.

WHY THE WINDOW IS SUMMARISED AUTOMATICALLY, NOT HAND-BUILT

A conventional pipeline asks a data scientist to decide which lags, rolling means and flags to build for each client, and to maintain them. PACX derives one family of statistics for every field the ledger carries — level, spread, extremes, recency and mix — so the model input is complete by construction and identical at training and prediction time. Adding a client field is a change to one query, not a modelling project. The retail knowledge comes from the fine-tuned foundation model, not from hand-crafted features.

Training and prediction are the same operation

The reason to build the training set this way is symmetry. At training time the model sees six months of history before a historical snapshot and learns to map it to what came next. At prediction time the snapshot is simply today: the same four cells run, the same window is assembled, the same feature engine summarises it, and the model outputs the next one or two months of demand for every item–store pair.

There is no separate feature pipeline for inference and no gap between how the model was evaluated and how it is used.

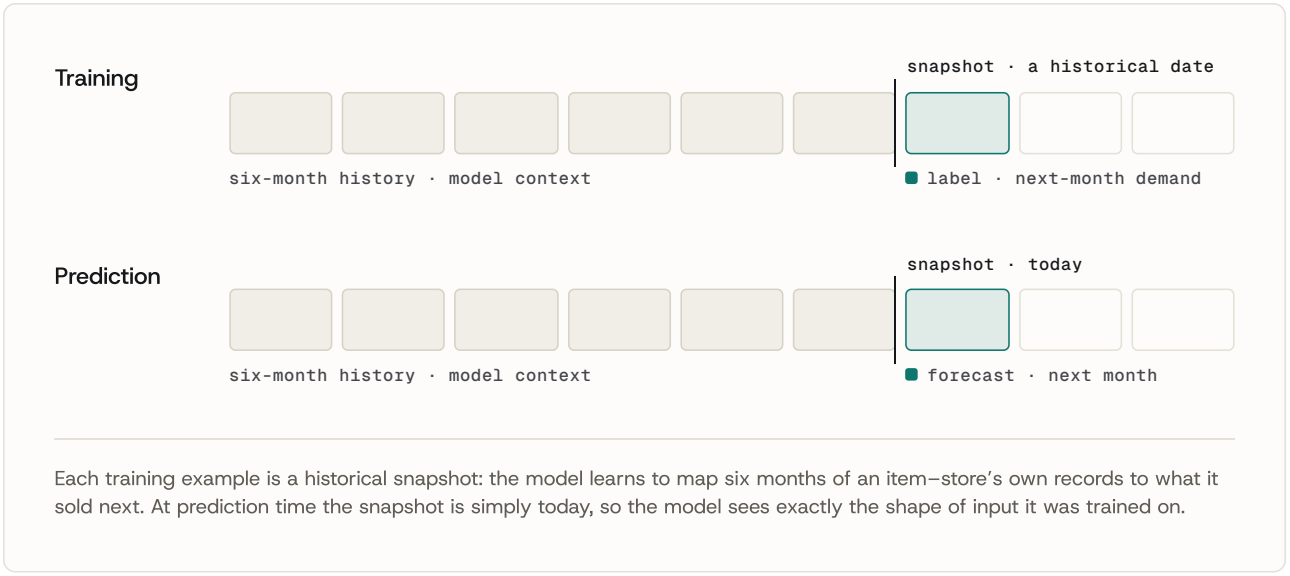

Figure 9. The prediction task has exactly the shape of the training examples. Each snapshot's six-month history is the model's input; the horizon after it is the label in training and the forecast in production.

The horizon and the frequency are first-class design choices rather than post-processing steps. A one-month objective and a two-month objective are separate core sets with separate labels, so a merchandiser who plans replenishment monthly and a buyer who commits purchase orders two months out each receive a forecast that was trained on exactly their question. A weekly objective is built the same way, on a weekly grid.

STAGE	CELL	WHAT IT PRODUCES	WHAT IT PROTECTS AGAINST
1 Forecast calendar	`all_relevant_dates_in_the_dataset`	A complete date grid over the data range, weekly or monthly	Missing periods read as missing data instead of zero demand
2 Forecast entities	`sampled_<entity>`	Item × store pairs from first activity onward	Rows before an item existed; unexplained gaps before delistings
3 Core set	`core_set`	Snapshot date + next-horizon demand label; completeness boundary	Temporal leakage; undercounted labels at the end of the data
4 Attribute set	`attribute_set`	The entity's own records strictly before the snapshot, with the product's attributes	Any future information reaching the model input

05 RESULTS

Accuracy at the level where the shelf is stocked.

The approach was deployed for a multi-store footwear and apparel retailer whose sales and stock ledger is organised by item and branch, with product attributes covering division, style, sub-style, heel type, colour, gender, pack size and MRP. Forecasts were generated for every active item × store pair on a one-month horizon and evaluated out-of-time — that is, against months the model had not seen, using the same completeness boundary applied in training.

HEADLINE RESULT

82%

forecast accuracy at the item × store level on a monthly horizon, measured as 1 − WAPE (weighted absolute percentage error) on out-of-time data.

Level	Horizon
Item × store	Next month
Metric	**Evaluation**
1 − WAPE	Out-of-time

How the metric is computed

$$\text{accuracy} = 1 - \Sigma\,|\text{actual} - \text{forecast}| \div \Sigma\,\text{actual}$$

Errors are summed across every item–store pair and divided by total actual volume, so each pair counts in proportion to what it actually sold.

How the figure was measured

The number is only as good as the protocol behind it. PACX's evaluation harness applies the same rules to every model it trains, including the gradient-boosting baseline the foundation model is compared against.

01 **Chronological holdout, no shuffle**

Snapshot rows are sorted by date and the most recent tenth is held out. The cut is rounded to a whole date, so no day is split between training and test, and test dates are checked to fall strictly after the last training date.

02 **Health checks before training**

The data must pass checks for sample volume, a usable target (not null, constant or all zero), duplicate item–store–date rows, drift between the training and test periods, and leakage — any feature almost perfectly correlated with the

target is treated as a copy of it and the run is stopped.

03 **One primary metric, bias alongside**

WAPE is the selection and reporting metric. Signed weighted error (bias), mean absolute error, root-mean-square error and R^2 are recorded next to it, so a model that is accurate on average but consistently low on the items that run out cannot hide.

04 **Retrain, then serve**

After evaluation the production model is fitted on the full history and the same four cells and feature engine produce its inputs at prediction time.

Three points of context make this figure meaningful.

1

The level

Item × store is the bottom of the retail hierarchy, the level at which the M5 competition found gradient-boosted models improved on simple statistical benchmarks by only about 3 percent.[5] Accuracy at category or company level is routinely higher because errors cancel in aggregation; accuracy at item × store is what determines whether the shelf is stocked.

2

The metric

WAPE weights each item–store's error by its volume, so the figure cannot be inflated by predicting zero for the long tail of slow movers — the trick that makes many SKU-level accuracy claims look better than they are. It is the metric a planner's replenishment logic actually depends on.

3

The comparison

On the same data, with the same four-stage curation, the same feature engine and the same out-of-time evaluation, the fine-tuned foundation model substantially outperformed the gradient-boosting baseline PACX previously deployed — a tournament of CatBoost and LightGBM candidates with squared-error and Tweedie objectives, up to 1,500 trees with early stopping and Bayesian hyperparameter search, selected on validation WAPE. The foundation model needed no per-client hyperparameter search, which is also why the time from data connection to first forecast is measured in days rather than months.

What accuracy at this level buys is easiest to see through the two halves of inventory distortion. Better item–store forecasts mean fewer items under-allocated to the stores where they would have sold — the out-of-stock component to which IHL attributes 65.6 percent of the cost — and fewer over-allocated to stores where they will age into markdown.[2] Because the same forecast feeds the allocation, replenishment and ordering workflows, and the pricing and

markdown decisions downstream of them, the improvement propagates through every decision that depends on it.

06 WHAT THIS MEANS FOR PLANNERS

Five things that change for a planning team.

The practical consequences of the approach are worth stating separately from the accuracy figure, because several of them change how a planning team operates.

01

New launches and new stores are forecastable from day one.

Because the model carries retail priors from fine-tuning and reads product attributes as context, an item with no history is forecast from what it is and what comparable items did, rather than defaulted to a category average.

02

The forecast is at the level decisions are made.

Item × store, monthly, for a one- or two-month horizon that matches the replenishment or buying cycle — no top-down disaggregation of a category forecast.

03

No feature-engineering backlog.

Adding a new signal — a promotion calendar, a competitor price, weather — is a change to the attribute query. The feature engine derives its statistics automatically; there is no lag to design and no retraining project to schedule around it.

04

Backtests mean something.

The completeness boundary and the strictly-before window are enforced in the data, not in a modelling convention, and every run must pass leakage and drift checks before training. The accuracy a planner sees in evaluation is the accuracy they should expect in production.

05

Faster onboarding.

The model is applied, not built. A retailer with six months of clean item–store history can be forecast; a retailer with three years benefits from more context but does not have to wait for it.

07 WHAT A RETAILER NEEDS

Deliberately modest data requirements.

The data requirements are deliberately modest and correspond to what most retailers and consumer brands already hold in their ERP or point-of-sale systems.

INPUT	DESCRIPTION
Sales and stock ledger	Periodic (daily or monthly) records per item and store: opening stock, receipts, sales quantity and value, transfers out, closing stock. Six months of history is the minimum; longer history improves context.
Item master	Product attributes: division, category, style, colour, gender, size or pack, price and price tier, launch date. Whatever attributes the business uses to describe the assortment.
Store master	Branch identifiers and any attributes the business uses to group stores (format, region, tier).
Forecast objectives	The horizons the business plans on — typically next month and next two months — each of which becomes its own core set.

Where it runs

The four curation cells and the feature engine run on Databricks — Spark SQL and Spark — so the pipeline scales to tens of millions of item–store snapshots without leaving the platform the data sits on. In PACX's managed deployment the client's data lands in a dedicated, encrypted tenant space and is processed there.

The four cells are visible in the model builder as ordinary SQL, so the client's data team can read and audit exactly what was sampled, how it was labelled and which history it saw. Forecasts are written back as tables and surfaced in the PACX planning workspace, where they feed the allocation, replenishment and ordering workflows and supply the demand layer for the wider decision chain — merchandise financial planning, assortment and size curves, pricing and promotions, and markdowns.

Runs on

Databricks — Spark SQL and Spark

Pipeline

Four SQL cells and one feature engine, identical at training and prediction

Minimum history	Six months of item–store records
Output	Forecast tables per item × store, per horizon

08 LIMITATIONS

What this paper does not claim.

A white paper that only argues for its own approach is marketing. These are the boundaries of what the evidence above supports, stated so that a reader can weigh the claim rather than take it.

01

One deployment, one category.

The 82 percent figure comes from a single live deployment in footwear and apparel. Other categories, channels and data qualities will produce different numbers. PACX publishes per-client results only with the client's consent and with the measurement context attached.

02

An aggregate metric.

WAPE at item × store is volume-weighted across the whole assortment. It says how good the forecast is for the business, not how good it is for any single item. Bias is tracked alongside it for exactly this reason, and planners should read individual forecasts with their history, not the headline figure, in view.

03

Six months, and attributes, are the floor.

A retailer needs at least six months of item–store history. A new item with no history is forecast from its attributes and from comparable items; an item with neither has little to be forecast from.

04

External drivers are what the client supplies.

The only external signal PACX maintains itself is a festival and season calendar. Weather, macroeconomic indicators, competitor prices and marketing activity improve a forecast only when the client provides them as fields in the ledger.

05

The base model will change.

This paper describes a model class, not a fixed artefact. As the field advances the base model will be replaced, and the result reported here should be re-validated on the client's own data when it is.

06

Not independently audited.

The performance figures are PACX's own, from deployments and internal benchmarking. The evaluation protocol is described in full so that a prospective client can reproduce it on their own history before relying on it.

FIVE QUESTIONS TO ASK OF ANY FORECASTING CLAIM

The same questions apply to this paper. A vendor who cannot answer all five is reporting a number, not a result.

1. At what level of the hierarchy was accuracy measured — item × store, or a category total where errors cancel?
2. Which metric — volume-weighted WAPE, or a MAPE that a long tail of zero forecasts can flatter?
3. Was the evaluation out-of-time, on periods the model never saw, with no shuffling across dates?
4. How were labels near the end of the data handled — excluded, or allowed to undercount?
5. Are the features at prediction time produced by the same code, from the same inputs, as the features at training time?

09 CONCLUSION

Where the next generation of models earns its keep.

Gradient boosting was the right answer to retail forecasting in 2020. It is still a good answer at the category and company level. But the evidence from the largest public retail forecasting benchmark is that its advantage almost vanishes at the item–store level, and it is at the item–store level that retailers lose $1.7 trillion a year to empty shelves and ageing stock.

Tabular foundation models change the starting point. Pre-trained on tens of millions of datasets and fine-tuned on retail and consumer-brand data, the model PACX deploys arrives already understanding how demand relates to stock, sell-through, price and product attributes, and reads each client's recent history as context rather than learning it from scratch. Combined with a data-curation method that makes training examples exactly the shape of the prediction task — and that enforces the guard against leakage in the data itself — this approach reached 82 percent accuracy at the item × store level on a monthly horizon in live deployment.

For retail and consumer-brand executives, the implication is that SKU–store forecasting is no longer the place where machine learning runs out of road. It is the place where the next generation of models earns its keep.

ABOUT PACX

PACX is an AI-native retail planning platform for demand forecasting, allocation and ordering, with connected recommendations, configurable constraints, staged approvals and auditable actions.

Its demand forecasting is built on a fine-tuned tabular foundation model and runs natively on the client's data platform, using the four-stage method described in this paper.

SOURCES

References

[1] IHL Group, "Retail Inventory Crisis Persists Despite $172 Billion in Improvements", *Analyst Corner*, September 2025. https://www.ihlservices.com/news/analyst-corner/2025/09/retail-inventory-crisis-persists-despite-172-billion-in-improvements/

[2] IHL Group, "The 2026 Inventory Distortion Study and key research findings", 2026. https://www.ihlservices.com/news/ihl-research-findings/

[3] Gartner, "Gartner Predicts 70% of Large Organizations Will Adopt AI-Based Supply Chain Forecasting to Predict Future Demand by 2030", *Press release*, 16 September 2025. https://www.gartner.com/en/newsroom/press-releases/2025-09-16-gartner-predicts-70-percent-of-large-orgs-will-adopt-ai-based-supply-chain-forecasting-to-predict-future-demand-by-2030

[4] McKinsey & Company, "AI-driven operations forecasting in data-light environments". https://www.mckinsey.com/capabilities/operations/our-insights/ai-driven-operations-forecasting-in-data-light-environments

[5] S. Makridakis, E. Spiliotis and V. Assimakopoulos, "M5 accuracy competition: Results, findings, and conclusions", *International Journal of Forecasting, vol. 38, no. 4*, 2022. https://www.sciencedirect.com/science/article/pii/S0169207021001874

[6] N. Hollmann et al., "Accurate predictions on small data with a tabular foundation model", *Nature, vol. 637*, January 2025. https://www.nature.com/articles/s41586-024-08328-6

[7] TabICL authors, "TabICLv2: A better, faster, scalable, and open tabular foundation model", *arXiv:2602.11139*, February 2026. https://arxiv.org/abs/2602.11139

[8] S. B. Hoo, S. Müller, D. Salinas and F. Hutter, "From Tables to Time: Extending TabPFN-v2 to Time Series Forecasting", *arXiv:2501.02945*, 2025. https://arxiv.org/abs/2501.02945

[9] S. Kapoor and A. Narayanan, "Leakage and the reproducibility crisis in machine-learning-based science", *Patterns, vol. 4, no. 9*, 2023. https://arxiv.org/abs/2207.07048

[10] C. Molnar, "The state of tabular foundation models (2026)", *Mindful Modeler*, February 2026. https://mindfulmodeler.substack.com/p/the-state-of-tabular-foundation-models

[11] AI Horizon Forecast, "Time series foundation models: A deep dive into strengths and limitations". https://aihorizonforecast.substack.com/p/time-series-foundation-models-a-deep

Sources last reviewed 4 Sep 2026

CITE THIS PAPER

PACX.ai (2026). Beyond Gradient Boosting: Why a fine-tuned tabular foundation model outperforms legacy machine learning for SKU–store demand forecasting in retail and consumer brands. White paper, March 2026. https://pacxai.com/white-papers/foundation-model-demand-forecasting