Traditional Machine Learning versus Large Language Models for Predicting Ship Turnaround Times at Brazilian Ports

Eduardo C. B. Pacheco[1], Jonnathan A. Ramos[2]

[1] Institute of Mathematics and Computer Sciences (ICMC), University of São Paulo, São Carlos, Brazil
eduardo.caruso.pacheco@usp.br
[2] Independent Researcher, Brazil
jonnathanalveskz@outlook.com

Abstract. Predicting ship turnaround time at ports is an operationally valuable regression problem on heterogeneous, text-rich tabular data. We ask whether Large Language Models (LLMs) can match purpose-built Machine Learning (ML) on this task, using a leakage-controlled pipeline over seven years of Brazilian (ANTAQ) data. On a fixed 200-berthing test set, with 20 seeds and seed-paired statistics (Wilcoxon, Cohen's d), we compare five families that read the same cases: classical ML on engineered tabular features plus multilingual sentence embeddings; frontier LLMs (Qwen3-32B, DeepSeek-V3) prompted zero-, 5- and 20-shot; a small generative LLM (Qwen2.5-1.5B) fine-tuned on 100–6000 serialized records; an encoder with a regression head fine-tuned on 100–12000 records; and a tabular foundation model (TabFM) reading the raw table in-context. Tuned XGBoost attains $R^2 \approx 0.73$. Every prompted configuration stays at $R^2 \leq 0$ and in-context examples do not help; generative fine-tuning barely clears zero, reaching only $R^2 \approx 0.10$ at 6000 examples. The encoder is the only *text* regime to rise substantially above $R^2 = 0$, reaching ≈ 0.43 at 12000 examples yet still trailing all ML. In contrast, TabFM, with no task training, matches or beats a data-matched XGBoost at every budget (100–6000; R^2 up to ≈ 0.60), leading the low-data regime, though full-data XGBoost still wins and TabFM is far costlier at inference. The predictive signal is tabular, not textual: text-only LLMs miss it, whereas a tabular foundation model recovers it in-context and engineered boosting wins at scale.

CCS Concepts: • **Computing methodologies** → **Supervised learning by regression**; *Natural language processing*.

Keywords: large language models, machine learning, maritime transportation, regression, ship turnaround time, tabular data

1. INTRODUCTION

Maritime transport carries roughly 80% of international trade by volume, and bottlenecks in ship turnaround time propagate through the logistics chain. The Brazilian National Waterway Transportation Agency (ANTAQ) publishes operational records of berthings and cargo, and several studies apply machine learning to these data with high headline numbers. Recent work [Pacheco et al. 2026] shows those numbers are inflated by structural leakage and proposes a leakage-controlled pipeline whose realistic performance ($R^2 \approx 0.6$–0.7) is our reference.

In parallel, Large Language Models (LLMs) are increasingly proposed as general predictors that ingest a record as text and emit an answer. This raises a concrete question for applied data mining: *on a real, heterogeneous, text-rich tabular regression problem, can text-based models match a purpose-built ML model?* We study ship turnaround time, where each berthing mixes categorical context (port, operation and navigation type, region), numeric cargo attributes and four high-cardinality textual fields (cargo group, containerized cargo group, origin and destination) that the reference pipeline encodes with multilingual sentence embeddings.

This work was carried out at ICMC–USP, whose computational support the authors acknowledge. The cleaned dataset (2018–2024), serialized prompts, code, per-seed predictions and metrics are released to support reproducibility.

On a single fixed 200-berthing test set and an identical protocol, we compare five families: (i) five classical ML models on tabular features *and* the cargo/route embeddings; (ii) two frontier LLMs (Qwen3-32B, DeepSeek-V3) prompted zero-, 5- and 20-shot; (iii) a small *generative* LLM (Qwen2.5-1.5B) fine-tuned to emit the number from a serialized record; (iv) the same MiniLM *encoder with a regression head* fine-tuned end-to-end on those serializations; and (v) a *tabular foundation model* (TabFM) reading the raw table in-context, with no gradient training. Every configuration runs 20 seeds, compared seed-by-seed with paired Wilcoxon tests and Cohen's d; we also report training and inference *cost*.

Contributions.

—A controlled five-family comparison (tuned ML; prompted LLMs at three shot counts; generative small-LLM fine-tuning; encoder-plus-head fine-tuning; an in-context tabular foundation model) on a leakage-free ANTAQ split, with 20 seeds and seed-paired statistics.

—Evidence that all prompted regimes stay at $R^2 \leq 0$ (no gain from 5- or 20-shot) and generative fine-tuning only edges to $R^2 \approx 0.10$ at 6000 examples, while tuned XGBoost reaches $R^2 \approx 0.73$, with every gap significant at $p < 0.001$.

—A finding that an encoder with a regression head is the *only* text regime to rise substantially above $R^2 = 0$, scaling without saturating to $R^2 \approx 0.43$ and dominating the generative LLM at every budget, yet still trailing all ML models.

—Evidence that a *tabular* foundation model (TabFM), reading the raw table in-context with no training, matches or beats a data-matched XGBoost at every budget (100–6000) and so leads the low-data regime, while full-data boosting still wins overall, which locates the signal in the tabular interface rather than the text one. We also report a cost comparison (TabFM's profile is inverted: near-zero training, costly inference) and release a reproducible artifact.

2. RELATED WORK

Prior machine learning for stay-time prediction at Brazilian ports reports high headline numbers on ANTAQ ($R^2 \approx 0.83$ for turnaround regression [Rao et al. 2025], ≈74% accuracy for stay-time classification [Abreu et al. 2023]), which Pacheco et al. [Pacheco et al. 2026] audit and show to be inflated by structural leakage: a random row-level split places about 97% of test berthings in training, and under a berthing-level split the same model loses most of its R^2. We adopt their corrected, leakage-controlled pipeline as the ML reference.

LLMs have been applied to non-language tabular tasks. LIFT [Dinh et al. 2022] fine-tunes language models on serialized rows for classification and regression; TabLLM [Hegselmann et al. 2023] studies few-shot tabular classification via textual templates; and LLMTime [Gruver et al. 2023] casts numeric forecasting as next-token prediction. Encoders with regression heads (e.g. fine-tuned BERT-style models [Reimers and Gurevych 2019]) are a strong text-regression baseline often omitted from LLM-versus-tabular comparisons. A parallel line bypasses text: tabular foundation models pretrained on synthetic tasks predict in-context, and TabPFN [Hollmann et al. 2025] shows this is strongest on small tabular datasets, the regime we probe with TabFM. Our study holds test set and protocol fixed across families and adds a tuned gradient-boosting baseline with semantic embeddings, an encoder-plus-head trained on the *same* text the LLMs read, and a tabular foundation model, extending to text-based LLM predictors the observation that tree-based models still outperform deep networks on tabular data [Grinsztajn et al. 2022].

3. PROBLEM AND DATA

Let $\mathcal{B}$ be the set of berthing events. For each $b \in \mathcal{B}$ we predict two non-negative duration targets (hours): `TOperacao` (cargo operation) and `TAtracado` (total time moored), as separate regressions

on the same matrix. ANTAQ decomposes a berthing into consecutive stages (waiting to berth, $T1$; waiting for the operation to start, $T2$; cargo operation, $T3$; waiting to unberth, $T4$), so `TOperacao` $= T3$ and `TAtracado` $= T2+T3+T4$. We use the pipeline of [Pacheco et al. 2026] over ANTAQ 2018–2024 [ANTAQ 2024]: it aggregates cargo to the berthing level (summing numeric attributes, dominant value for homogeneous categoricals, mass-weighted one-hot-sum for heterogeneous ones); imputes with train-only conditional rules; encodes the four high-cardinality text fields (cargo group, containerized cargo group, origin, destination) with the multilingual encoder `paraphrase-multilingual-MiniLM-L12-v2` [Reimers and Gurevych 2019] (384 dims each, mass-weighted average); applies a semantic outlier filter; and splits 80/20 stratified by region. Absolute dates and partial-time variables are excluded as they reconstruct the targets. From the held-out 20% partition we draw a fixed test set of N=200 berthings uniformly at random (seed 42, no further stratification); the remaining $\approx$391,000 form the training set, with $\text{IDATRACACAO}_{\text{train}} \cap \text{IDATRACACAO}_{\text{test}} = \emptyset$. The split is therefore random across 2018–2024, not time-aware: the setting is interpolation within the period, not forecasting into a future one. The *same* 200 cases are scored by all systems, so every comparison is conditional on this one sample.

4. EXPERIMENTAL DESIGN

We report RMSE, R^2 and MAE; 20 seeds per configuration.

4.1 Regime 1: Traditional ML with embeddings

Five models on the full berthing-level matrix (ordinal-encoded categoricals, median target encoding for the mooring port, numeric and mass-weighted one-hot-sum cargo features, *and* the four 384-dim embeddings; 1,606 columns): XGBoost [Chen and Guestrin 2016] (tuned per target with Optuna), a two-layer MLP, k-NN (k=10), linear regression, and a linear SVM (SGD). Deterministic models (k-NN) replicate predictions across seeds.

4.2 Regime 2: Prompted LLMs

Qwen3-32B and DeepSeek-V3 [DeepSeek-AI 2024] via Amazon Bedrock at temperature 0.1. Each berthing is a compact JSON of all non-target fields; the model returns one number. We evaluate *zero-shot*, *5-shot* and *20-shot* (exemplars sampled uniformly at random per seed). Unparseable replies fall back to the training median (rare). We fix a single minimal prompt format, with no prompt search, chain-of-thought or similarity-based exemplar selection (see Limitations).

4.3 Regime 3: Generative small-LLM fine-tuning

We fine-tune the base (non-instruct) Qwen2.5-1.5B [Qwen Team 2025] with LoRA [Hu et al. 2022] (r=16) in `bfloat16`. Each berthing is first turned into a fluent Portuguese description (e.g. *"cargo of 26,503 tonnes, mainly mineral fuels, by cabotage from Santos to Suape, in January 2019"*) generated deterministically by DeepSeek-V3 and cached; targets are never included. The model learns to emit the number (loss masked to the answer). Conditions: untrained *base*, and fine-tuning on *100*, *500*, *1500*, *3000* and *6000* serialized records, reseeded per run.

4.4 Regime 4: Encoder with a regression head

We fine-tune the *same* multilingual MiniLM backbone that produces the ML embeddings, with a two-output linear regression head (joint [`TOperacao`, `TAtracado`]), end-to-end on the same serialized descriptions, with standardized targets. Conditions: *base* (untrained head) and fine-tuning on *100* to *12000* examples, 20 seeds.

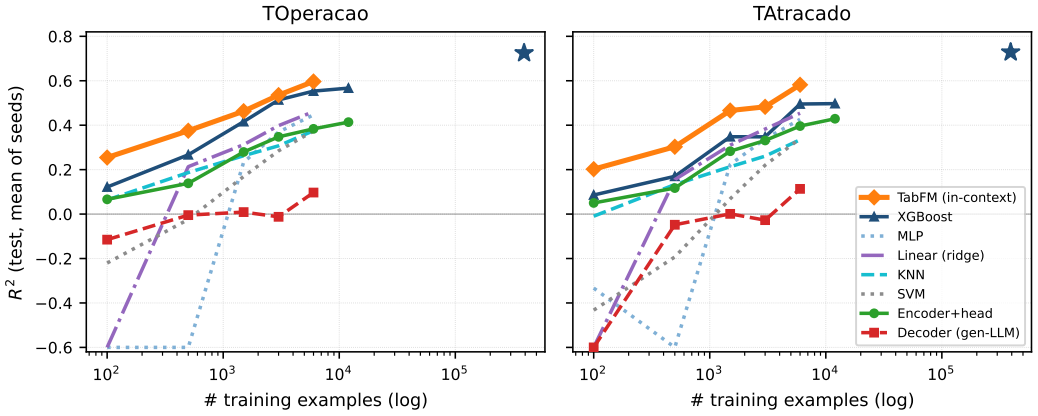

Fig. 1. Data-matched data-efficiency (R^2 vs. number of training examples, log scale) for all eight model families trained on the *same* N random berthings (100–6000, 20 seeds); $\star$ is XGBoost's full-data point (N=391,460). The tabular foundation model TabFM (bold orange) leads at every data-matched budget with zero training; full-data XGBoost ($\star$) still tops all. The text encoder catches KNN/SVM by N=6000; linear and MLP overfit catastrophically at small N in the 1606-dim feature space (clipped at -0.6 for legibility; the exact values, down to $R^2 = -9.2$, are in Table III); the generative decoder stays near 0 until a small rise at 6000.

4.5 Regime 5: Tabular foundation model (in-context)

We evaluate TabFM [Google Research 2025], a pretrained tabular foundation model that performs *in-context* learning: the N training rows are passed as context and predictions for the test cases are produced in a single forward pass, with no gradient training. It reads the *raw* tabular fields (the 74 non-target columns: origin, destination, cargo groups, mass, port, date parts, etc.), its native input, rather than the engineered 1606-dim matrix or the serialized text. We use the same data-matched budgets (100–6000 records) and the *identical* 20 seeds and random subsamples as the ML curves, so TabFM is paired seed-by-seed with the data-matched ML models.

4.6 Statistical protocol

Each configuration yields 20 per-seed metric values. Pairs of configurations are compared on RMSE seed-by-seed with the paired Wilcoxon signed-rank test [Wilcoxon 1945]; effect size is Cohen's d [Cohen 1988] (we report the pooled-SD d; $|d|$=0.8 is large). Some configurations are near-deterministic across seeds, so the pooled SD approaches zero and $|d|$ inflates (>20); read such values as "the two configurations never overlap across seeds", not as graded effect magnitudes. Wilcoxon p-values are unaffected. The test set is fixed, so seeds affect only training; ML uses seeds 0–19 and the text regimes a common 20-seed list, paired by rank.

5. RESULTS

Table I reports all configurations; Table II compares XGBoost against each text-based configuration; Figure 1 shows the data-efficiency curves (with a data-matched XGBoost curve).

5.1 ML versus text-based models

Every ML model outperforms every text-based configuration on both targets. Table II compares XGBoost against the strongest configuration of each text family: all gaps are significant at $p < 0.001$ with large pooled-d, and the RMSE deficit is 11–28 hours. Even the weakest ML model (SVM) beats the best text-based model.

Table I. Test performance (200 berthings, 20 seeds): RMSE in hours (mean ± std), mean MAE in hours and mean R^2.

	TOperacao			TAtracado		
	RMSE	MAE	R^2	RMSE	MAE	R^2
Traditional ML (tabular + embeddings)						
XGBoost	19.10 ± 0.17	7.75	+0.725	20.96 ± 0.17	10.72	+0.728
MLP	20.57 ± 0.51	8.53	+0.681	23.09 ± 0.63	11.79	+0.670
KNN	22.17 ± 0.00	9.62	+0.629	23.12 ± 0.00	12.13	+0.669
Linear Reg.	25.15 ± 0.22	11.81	+0.523	27.72 ± 0.22	15.39	+0.525
SVM	26.30 ± 0.07	10.35	+0.478	28.68 ± 0.08	13.73	+0.491
Encoder + regression head (MiniLM)						
base	36.53 ± 0.00	19.74	-0.007	40.30 ± 0.00	24.69	-0.005
FT-100	35.16 ± 1.19	17.46	+0.066	39.16 ± 1.49	21.94	+0.050
FT-500	33.77 ± 1.32	16.38	+0.138	37.75 ± 1.49	20.74	+0.117
FT-1500	30.89 ± 0.96	14.77	+0.279	34.03 ± 1.01	18.36	+0.283
FT-3000	29.39 ± 0.92	14.02	+0.348	32.86 ± 1.10	18.01	+0.331
FT-6000	28.57 ± 0.91	13.20	+0.384	31.23 ± 0.97	16.70	+0.396
FT-12000	27.87 ± 0.53	12.85	**+0.414**	30.37 ± 0.71	16.25	**+0.429**
Generative small LLM (Qwen2.5-1.5B)						
base	217.0 ± 0.0	58.94	-34.54	217.7 ± 0.0	64.21	-28.33
FT-100	38.31 ± 3.29	17.61	-0.115	49.29 ± 15.1	25.16	-0.636
FT-500	36.40 ± 2.59	16.80	-0.005	41.05 ± 3.02	22.33	-0.048
FT-1500	36.15 ± 2.66	17.18	+0.009	40.11 ± 2.49	21.82	+0.001
FT-3000	36.58 ± 1.98	16.51	-0.012	40.64 ± 3.08	22.08	-0.027
FT-6000	34.52 ± 2.38	15.64	+0.097	37.80 ± 2.14	19.99	+0.113
Prompted frontier LLMs						
Qwen3-32B 0-shot	36.48 ± 0.27	18.80	-0.004	40.39 ± 0.56	22.03	-0.009
Qwen3-32B 5-shot	36.55 ± 2.73	16.99	-0.013	41.20 ± 3.86	22.41	-0.059
Qwen3-32B 20-shot	36.25 ± 2.85	15.94	+0.003	41.50 ± 4.71	21.77	-0.079
DeepSeek-V3 0-shot	36.45 ± 0.33	16.76	-0.002	42.03 ± 0.20	21.69	-0.093
DeepSeek-V3 5-shot	44.23 ± 18.9	18.96	-0.732	44.69 ± 11.5	24.49	-0.313
DeepSeek-V3 20-shot	38.18 ± 6.22	16.56	-0.128	43.34 ± 9.41	21.59	-0.214

Table II. Paired comparison of XGBoost vs. each text-based configuration. ΔRMSE = mean(config)−mean(XGBoost) in hours (positive = XGBoost better); d is pooled Cohen's d; significance from paired Wilcoxon ($^{*}p<0.05$, $^{**}p<0.01$, $^{***}p<0.001$).

	TOperacao		TAtracado	
vs. XGBoost	ΔRMSE	d	ΔRMSE	d
Encoder FT-12000	+8.8	-21.6***	+9.4	-17.8***
Gen-LLM FT-1500	+17.1	-9.0***	+19.2	-10.9***
Qwen3-32B 0-shot	+17.4	-76.7***	+19.4	-46.9***
Qwen3-32B 20-shot	+17.2	-8.5***	+20.5	-6.2***
DeepSeek-V3 0-shot	+17.3	-66.2***	+21.1	-116.1***

5.2 Prompting and fine-tuning data-efficiency

In-context examples do not help. For both frontier models, 5-shot and 20-shot are statistically indistinguishable from zero-shot (all $p > 0.1$); 20-shot ≈ zero-shot and few-shot can destabilize DeepSeek-V3 (std up to 19 h). *Generative fine-tuning barely moves above $R^2 \approx 0$:* it rescues the small model from an unusable base ($R^2 \approx -30$) but stays at $R^2 \approx 0$ through 3000 examples and only edges up to ≈ 0.10 at 6000, at or below the level of a 32B model prompted zero-shot. *The encoder head is the exception:* from a mean-predictor base it lifts R^2 monotonically and *without saturating* to ≈ 0.41–0.43 at 12000 examples (Table I), beating the generative LLM at every budget (paired $p < 0.01$). Even the 6000 → 12000 step is significant ($p < 0.05$), so the curve has not flattened, yet the encoder still approaches only the weakest ML models and remains well below XGBoost's 0.73.

Table III. Data-matched test R^2 (mean over 20 seeds) by training-set size N, all families; no value is truncated, so the collapse of the linear and MLP models at small N is visible. ML models share the engineered feature matrix; encoder/decoder read the serialized text.

	N (training examples)				
`TOperacao`	100	500	1500	3000	6000
TabFM (in-ctx)	**0.255**	**0.375**	**0.463**	**0.535**	**0.597**
XGBoost	0.122	0.267	0.416	0.514	0.553
MLP	-2.10	-5.41	0.233	0.369	0.446
Linear (ridge)	-3.78	0.213	0.312	0.398	0.461
KNN	0.064	0.187	0.262	0.307	0.375
SVM	-0.22	-0.02	0.169	0.285	0.374
Encoder	0.066	0.138	0.279	0.348	0.384
Decoder (LLM)	-0.12	-0.01	0.009	-0.01	0.097
`TAtracado`	100	500	1500	3000	6000
TabFM (in-ctx)	**0.202**	**0.303**	**0.466**	**0.483**	**0.582**
XGBoost	0.086	0.170	0.348	0.348	0.495
MLP	-0.33	-9.17	0.223	0.338	0.426
Linear (ridge)	-0.81	0.155	0.310	0.383	0.454
KNN	-0.01	0.133	0.213	0.261	0.334
SVM	-0.43	-0.19	0.069	0.220	0.340
Encoder	0.050	0.117	0.283	0.331	0.396
Decoder (LLM)	-0.64	-0.05	0.001	-0.03	0.113

Data-matched comparison. To separate representation from data budget, we train *all eight families* on the same subsample sizes (100–6000 random berthings, 20 seeds, identical subsamples; Figure 1, Table III). Five findings emerge. (i) The *tabular foundation model* TabFM leads at *every* data-matched budget on both targets, significantly beating XGBoost by paired Wilcoxon at 8 of 10 size–target cells (e.g. R^2 0.26 vs. 0.12 at N=100 and 0.60 vs. 0.55 at N=6000 on `TOperacao`; only N=1500, 3000 on `TOperacao` are ties, $p \approx 0.08$), with *zero* task training, from the raw table alone. (ii) Among trained models XGBoost is the most data-efficient, reaching the encoder's N=12000 level ($R^2 \approx 0.42$) with only ≈ 1500 examples and training 25–70× faster at matched N (Table IV). (iii) At small N the high-dimensional (1606-feature) space breaks the linear and MLP models (large negative R^2 at $N \leq 500$); trees, KNN, the encoder and TabFM stay robust. (iv) The text *encoder* is competitive with standard ML at matched data: by N=6000 ($R^2 \approx 0.38$–0.40) it matches KNN and SVM. (v) The generative *decoder* remains the weakest of all. Full-data XGBoost ($\star$, all 391,460 berthings, $R^2 \approx 0.73$) still tops TabFM's best (N=6000, ≈ 0.60), but TabFM cannot ingest more context, so the residual gap is one of *data budget*, not representation: reading the raw table in-context already extracts most of the signal. A log-linear fit extrapolates that the encoder would need $\sim 4 \times 10^5$ serialized examples to match XGBoost's full-data R^2.

5.3 Computational cost

Table IV contrasts training and inference cost; local and API timings are not directly comparable in absolute terms, but orders of magnitude are. ML inference is sub-second and the encoder is the cheapest learned text model; the generative LLM is by far the most expensive while being *less* accurate, and prompted LLMs need an API round-trip per case that grows with the number of exemplars. At the largest data-matched budget (N=6000), training takes ≈ 4 s (XGBoost), ≈ 290 s (encoder) and ≈ 1170 s (generative LLM), and inference over the 200 cases ≈ 1 ms, ≈ 0.9 s and ≈ 20 s. TabFM has the *inverted* profile: no training, but in-context inference over the 200 cases grows with context from ≈ 15 s (N=100) to ≈ 570 s (N=6000), an inference cost orders of magnitude above XGBoost's.

Table IV. Computational cost. Training is one-time (per seed for fine-tuning); inference is for the full 200-case test set unless noted. ML on CPU; encoder/generative LLM on M5 Pro (MPS); prompted LLMs via Bedrock. XGBoost's data-matched training takes 1.3 s (N=100) to 4.0 s (N=6000).

Regime	Training	Inference (200 cases)
XGBoost (ML)	182 s (+9 s embed.)	3 ms
Encoder + head	33–380 s (100–12000)	$\approx$ 0.9 s
Generative LLM (1.5B)	600–1170 s (3000–6000)	$\approx$ 20 s
Prompted LLM (32B)	none	0.6–2.0 s *per case*
TabFM (in-context)	none ($\approx$ 0.2 s context)	15–570 s (100–6000)

6. DISCUSSION

Three points stand out. First, *the tabular interface is decisive, not model size*: a 1.5B model fine-tuned on 1500 examples and a 32B model prompted with 20 examples both stall at $R^2 \approx 0$, while tuned trees reach 0.73 and, most tellingly, TabFM beats a data-matched XGBoost at every budget from the *raw* columns. Casting the record as free text (as the LLMs do) discards structure that both engineered features and a pretrained tabular transformer exploit. Second, *a regression formulation helps and scales*: the text regime that predicts a number with a supervised head improves monotonically with data, whereas the one that generates the number as tokens does not; the encoder nonetheless remains text-only and never matches the tabular models. The two regimes differ at once in architecture (encoder vs. decoder), adaptation (full fine-tuning vs. LoRA) and task formulation (head vs. token generation), so we cannot isolate a single cause: the evidence supports only that the regime differing along these axes did markedly better. An ablation holding the backbone fixed, with a regression head on the same decoder, is left to future work. Third, *cost compounds the accuracy gap*: the generative LLM is the most expensive and least accurate, TabFM buys its accuracy with heavy in-context inference, while ML is both cheap and best at scale.

Limitations. Representations are deliberately asymmetric, as each family is used in practice: ML uses engineered features, the text regimes read raw fields as text, and TabFM the raw table. The test set has 200 fixed, leakage-free cases drawn at random rather than temporally: all statistics are conditional on it, and its size limits power in the tails. Prompted LLMs use one minimal prompt format with random exemplars; chain-of-thought or similarity-based exemplar selection might raise those numbers, so the prompting result characterises this format, not prompting in general. Seeds are paired by rank. We explored one model per text family and one tabular foundation model under modest budgets; other backbones, hybrid text-plus-tabular models, and further tabular foundation models are left to future work.

Practical implications and future work. With a documented schema and abundant records, tuned gradient boosting is the pragmatic choice: most accurate and, at milliseconds per inference, by far the cheapest. When labels are scarce, a tabular foundation model is compelling, with higher accuracy and no training, if its heavier inference is acceptable; text-only models remain attractive only for undocumented or shifting schemas. Promising directions: hybrid models feeding the fine-tuned encoder's embeddings alongside the engineered features into one boosting model, other tabular foundation models, and predictive uncertainty.

7. CONCLUSION

On a leakage-controlled seven-year ANTAQ benchmark for ship turnaround time, traditional ML with semantic embeddings decisively outperforms text-based models across prompting and fine-tuning: tuned XGBoost reaches $R^2 \approx 0.73$ while no prompted configuration beats a mean predictor and generative fine-tuning only barely does ($R^2 \leq 0.11$ at 6000 examples), every gap significant at $p < 0.001$. An encoder with a regression head is the only text regime to extract real signal (R^2 up to 0.43,

still rising at 12000 examples) and dominates the generative LLM at equal data and far lower cost, yet still trails all ML. Crucially, the failure is one of *interface*, not of foundation models: TabFM, reading the raw table in-context with no task training, matches or beats a data-matched XGBoost at every budget and leads the low-data regime, though full-data boosting still wins overall and at a fraction of its inference cost. For heterogeneous operational-time regression, our evidence favors purpose-built tabular models (engineered boosting at scale, a tabular foundation model when data are scarce) over text-only models used as direct predictors.

8. CODE AND DATA

Code, serialized prompts, per-seed predictions, metrics and figures are released at `https://github.com/eduardocbpacheco/antaq-ml-vs-llm` [Pacheco and Ramos 2026]; the cleaned ANTAQ dataset (2018–2024) and the leakage-controlled pipeline are on Zenodo (DOI `10.5281/zenodo.20549160`) [Pacheco and Louzada 2026; Pacheco et al. 2026].

REFERENCES

Abreu, L. R., Maciel, I. S. F., Alves, J. S., Braga, L. C., and Pontes, H. L. J. A decision tree model for the prediction of the stay time of ships in Brazilian ports. *Engineering Applications of Artificial Intelligence* 117 (1): 105634, 2023.

ANTAQ. Estatístico Aquaviário. https://web.antaq.gov.br/anuario/, 2024.

Chen, T. and Guestrin, C. XGBoost: A Scalable Tree Boosting System. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. San Francisco, USA, pp. 785–794, 2016.

Cohen, J. *Statistical Power Analysis for the Behavioral Sciences*. Lawrence Erlbaum Associates, Hillsdale, USA, 1988.

DeepSeek-AI. DeepSeek-V3 Technical Report. arXiv:2412.19437, 2024.

Dinh, T., Zeng, Y., Zhang, R., Lin, Z., Gira, M., Rajput, S., yong Sohn, J., Papailiopoulos, D., and Lee, K. LIFT: Language-Interfaced Fine-Tuning for Non-Language Machine Learning Tasks. In *Advances in Neural Information Processing Systems*. New Orleans, USA, pp. 11763–11784, 2022.

Google Research. TabFM: A Tabular Foundation Model (v1.0.0). https://huggingface.co/google/tabfm-1.0.0-pytorch, 2025.

Grinsztajn, L., Oyallon, E., and Varoquaux, G. Why do tree-based models still outperform deep learning on typical tabular data? In *Advances in Neural Information Processing Systems*. New Orleans, USA, pp. 507–520, 2022.

Gruver, N., Finzi, M., Qiu, S., and Wilson, A. G. Large Language Models Are Zero-Shot Time Series Forecasters. In *Advances in Neural Information Processing Systems*. New Orleans, USA, pp. 19622–19635, 2023.

Hegselmann, S., Buendia, A., Lang, H., Agrawal, M., Jiang, X., and Sontag, D. TabLLM: Few-shot Classification of Tabular Data with Large Language Models. In *Proceedings of the International Conference on Artificial Intelligence and Statistics*. Valencia, Spain, pp. 5549–5581, 2023.

Hollmann, N., Müller, S., Purucker, L., Krishnakumar, A., Körfer, M., Hoo, S. B., Schirrmeister, R. T., and Hutter, F. Accurate predictions on small data with a tabular foundation model. *Nature* 637 (8045): 319–326, 2025.

Hu, E. J., Shen, Y., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., Wang, L., and Chen, W. LoRA: Low-Rank Adaptation of Large Language Models. In *Proceedings of the International Conference on Learning Representations*. Online, pp. 1–13, 2022.

Pacheco, E. C. B. and Louzada, F. ANTAQ Ship-Turnaround Dataset and Reproduction Artifacts. Zenodo, https://doi.org/10.5281/zenodo.20549160, 2026.

Pacheco, E. C. B., Martins, R. P., Gualberto, A. S., Germano, J. V. S., Neto, M. M., and Louzada, F. Methodological Pitfalls in Predicting Ship Turnaround Time at Brazilian Ports: An Empirical Audit and a Reproducible Pipeline. Manuscript under review, 2026.

Pacheco, E. C. B. and Ramos, J. A. antaq-ml-vs-llm: code, metrics and predictions. https://github.com/eduardocbpacheco/antaq-ml-vs-llm, 2026.

Qwen Team. Qwen2.5 Technical Report. arXiv:2412.15115, 2025.

Rao, A. R., Wang, H., and Gupta, C. Predictive Analysis for Optimizing Port Operations. *Applied Sciences* 15 (6): 2877, 2025.

Reimers, N. and Gurevych, I. Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*. Hong Kong, China, pp. 3982–3992, 2019.

Wilcoxon, F. Individual Comparisons by Ranking Methods. *Biometrics Bulletin* 1 (6): 80–83, 1945.