Journal of Computer Science and Information Technology
ISSN: 3080-3586
DOI: https://doi.org/10.61424/jcsit.v3i1.953
Journal Homepage: https://bluemarkpublishers.com/index.php/JCSIT

JCSIT
BLUEMARK PUBLISHERS

| RESEARCH ARTICLE

Enterprise LLM Router: Learning Quality–Capacity–Capability Trade-offs from 2026 Model Metadata

Lily Peng
Computer Science, University Of Leeds, Leeds, WYS, UK
Corresponding Author: Lily Peng, **E-mail**: lily.peng.dev@gmail.com

| ABSTRACT

Enterprise language-model selection is a constrained decision problem, not a single leaderboard lookup. This study integrated three 2026 metadata tables covering 22 models from eight providers and evaluated benchmark quality, capability requirements, paid rate-limit capacity, and discount support. The files joined exactly on provider and model without missing cells or duplicate keys. Quality was the mean percentile rank of MMLU, HumanEval, MATH, and Arena Elo; capacity was the mean paid-RPM and paid-TPM percentile for 19 models with numeric limits; discount support averaged batch and cached-input discounts. Nested leave-one-model-out and leave-one-provider-out tests compared ridge regression, k-nearest neighbors, random forest, and a training-mean baseline. The router was evaluated on 20 capability-gated candidate sets under five utility regimes, creating 100 scenarios, followed by a 0.1-resolution weight analysis. Ridge achieved leave-one-model-out MAE 13.09 and Spearman ρ .771, but provider-held-out performance weakened to MAE 22.35 and R^2 −.082. Capability breadth produced the lowest routing regret (5.60), ahead of ridge (8.46) and observed-quality routing (10.60). The eight-model Pareto frontier exposed specialization: o1 led quality, Gemini 2.5 Flash led numeric paid capacity, and DeepSeek V3 combined high capacity with maximum discount support. The evidence favors transparent capability filtering and abstention when price, latency, deployment license, or context evidence is absent.

| KEYWORDS

Large language models; model routing; enterprise AI; multi-criteria selection; benchmark metadata; rate limits; capability constraints; provider generalization

| ARTICLE INFORMATION

ACCEPTED: 09 June 2026 **PUBLISHED:** 19 July 2026 **DOI:** https://doi.org/10.61424/jcsit.v3i1.953

1. Introduction

Organizations now choose among model families that differ in benchmark performance, tool support, access limits, and commercial mechanisms. Sending every request to one premium model is easy to operate, but it ignores task heterogeneity and can concentrate technical and vendor risk. Prior routing research has therefore treated model selection as a cost–quality optimization problem, using cascades, preference models, or learned policies to decide which model should answer a request (Chen et al., 2024; C. Li et al., 2026; Ong et al., 2025). A production router must solve a broader problem. It must first reject models that cannot perform a required operation, then balance the remaining evidence without converting a rate limit into a latency claim or a discount percentage into a dollar price.

This distinction matters because throughput, latency, and price are related but non-equivalent. Requests per minute (RPM) and tokens per minute (TPM) bound traffic admitted by an API account; they do not measure time to first

token or end-to-end response time. Serving studies show that batching, prefill scheduling, decoding, and KV-cache management alter observed latency even when the model is unchanged (Agrawal et al., 2024; Kwon et al., 2023; Zhong et al., 2024). Likewise, batch and cached-input discounts record available saving mechanisms, but realized spend also depends on token mix, cache-hit rate, output length, and contract terms. This study therefore used the terms paid-capacity and discount support throughout the empirical analysis.

The investigation addressed three questions. First, how much quality structure can be learned from capability and rate-limit metadata when a model or an entire provider is held out? Second, do learned quality estimates improve multi-objective routing over transparent single-axis and capability-breadth baselines? Third, which provider–model combinations remain efficient under changes in quality, capacity, discount, and required-feature weights? These questions extend benchmark-based routing, which predicts task–model performance from historical evaluations (Shnitzer et al., 2023), toward a compact enterprise setting where only model-level metadata are available.

The study integrated a 22-model comparison, evaluated a quality surrogate with nested held-out protocols, and tested a capability-first router over 100 scenarios plus a full weight simplex. It separated leave-one-model interpolation from stricter leave-one-provider generalization. Unified routing benchmarks require comparison with fixed and simple policies (H. Li et al., 2026); here, capability breadth outperformed the learned router on average.

2. Literature Review

Model routing predates the current LLM market. FrugalML formulated the selection of heterogeneous prediction APIs under a budget and showed that adaptive API use can improve both accuracy and expense relative to a fixed service (Chen et al., 2020). FrugalGPT extended this logic to generative models through prompt adaptation, approximation, and cascades (Chen et al., 2024). In a cascade, a low-cost model answers first and a more capable model is invoked when the response is judged insufficient (Xin, 2025). AutoMix operationalized this idea through self-verification and escalation (Aggarwal et al., 2024). These methods depend on request-level evidence, whereas the present study asked what can be learned from a small model-level metadata snapshot.

Modern routers learn from diverse signals. Benchmark-aware routing maps tasks to expected model performance (Shnitzer et al., 2023), while RouteLLM learns strong-versus-weak choices from pairwise preferences (Ong et al., 2025). Hybrid LLM predicts difficulty before generation (Ding et al., 2024), and Zooter distills model-specific rewards into a selector (Lu et al., 2024). Routoo separates performance prediction from constrained choice (Mohammadshahi et al., 2024); Select-then-route filters by task taxonomy before selection (Shah & Shridhar, 2025). MixLLM adapts routing as traffic and models change (Wang et al., 2025). These approaches span supervised, similarity-based, reinforcement-learning, generative, and cascade families (Varangot-Reille et al., 2025).

Routing evaluations also differ in their target. RouterBench emphasizes prompt–model outcome matrices and standardized multi-model comparisons (Hu et al., 2024). LLMRouterBench expands evaluation across large instance and model collections and tests whether policies beat fixed-model baselines across quality–cost operating points (H. Li et al., 2026). Pairwise human preference provides another target: Chatbot Arena estimates relative model strength from user votes (Chiang et al., 2024), while MT-Bench and related judge-based methods scale evaluation but exhibit position, verbosity, and self-enhancement biases (Zheng et al., 2023). These studies justify combining complementary measures instead of treating one benchmark as universal quality.

The four quality inputs used here represented distinct evidence. MMLU measures knowledge and problem solving across 57 subjects (Hendrycks et al., 2021a). HumanEval measures functional correctness for code generated from docstrings (Chen et al., 2021). MATH emphasizes competition-style mathematical reasoning (Hendrycks et al., 2021b). Arena Elo summarizes pairwise preference outcomes rather than task accuracy (Chiang et al., 2024). A percentile composite reduced dependence on incompatible raw scales while preserving ordinal information. Its validity was tested against the supplied overall tiers and task ranks, and its stability was checked by dropping one benchmark at a time.

Enterprise routing requires construct discipline. Official rate limits are request and token quotas that vary by tier or account (Anthropic, n.d.; Google, n.d.; OpenAI, n.d.); they can constrain queueing risk but do not reveal latency. Generation-dependent cascades can add delay, so pre-generation selection differs from post-generation escalation

(Shen et al., 2025). Advertised context length likewise differs from effective long-context performance across retrieval and reasoning tasks (Bai et al., 2024; Hsieh et al., 2024). Provider identity also cannot establish open-source status, self-hostability, or license suitability.

Finally, risk-sensitive routers need an abstention path. Conformal routing has been proposed to bound the failure rate among requests assigned to a cheaper model (Uddin & Bauer, 2026). The current metadata do not support conformal calibration because they contain no request-level successes or failures. They do support a simpler safety rule: a model was removed when a required capability was false, and it was removed from capacity-weighted comparisons when its paid limits were nonnumeric. This deterministic filtering is central to the methodology rather than an after-the-fact correction.

3. Methodology

Figure 1 summarizes the workflow. The benchmark table contained four scores, three task ranks, and an overall tier; the feature table contained 15 Boolean capabilities; and the rate-limit table contained free-tier status, free and paid RPM/TPM, discounts, and minimum spend. Each table had 22 matching provider–model keys.

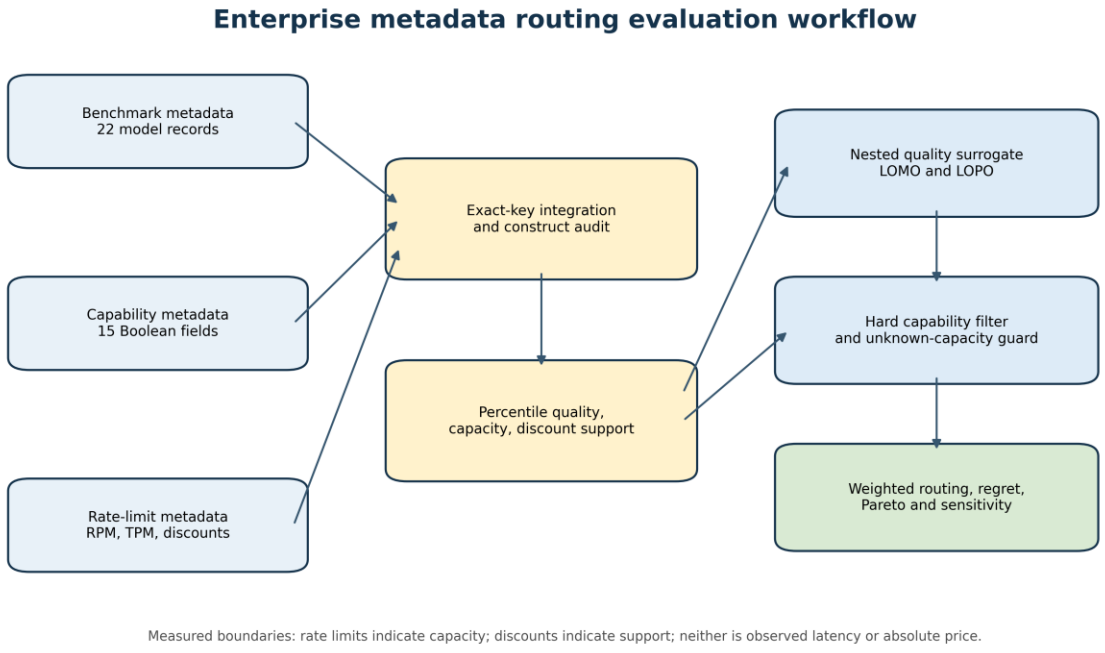

Figure 1. *Enterprise metadata routing workflow from exact-key integration through held-out quality estimation, hard capability filtering, and multi-objective evaluation.*

Table 1 reports the integration audit. The merged data had 33 columns with no blanks, duplicate rows, or duplicate keys. Paid RPM and TPM were numeric for 19 models; the three Meta records stated “varies” and remained unknown rather than zero. Five capabilities were universally true, and minimum spend was universally zero. These constant variables were excluded from learning but retained descriptively.

Table 1. Source composition, integrity checks, and merged-key coverage

Source	Rows	Cols	Missing	Dup. rows	Dup. keys	Keys matched
Benchmarks	22	10	0	0	0	22/22
Capabilities	22	17	0	0	0	22/22
Rate limits	22	10	0	0	0	22/22
Merged	22	33	0	0	0	22/22

Provider coverage and the derived quality distribution are reported in Table 2. OpenAI contributed four models; Anthropic, Google, Meta, and Mistral contributed three each; Cohere, DeepSeek, and xAI contributed two each. The data therefore represent a small cross-provider snapshot rather than a balanced panel or time series.

Table 2. Provider coverage and derived model-level constructs

Provider	Models	Q mean	Q max	T mean	D mean	Capabilities
OpenAI	4	71.45	97.73	46.05	37.50	10.75
DeepSeek	2	70.45	78.41	75.00	35.00	6.00
Anthropic	3	67.80	89.77	12.72	70.00	10.67
Google	3	62.50	88.64	83.33	25.00	14.33
Meta	3	47.54	62.50	—	0.00	7.33
xAI	2	42.33	44.32	25.00	0.00	7.00
Mistral	3	28.22	39.20	53.95	0.00	8.67
Cohere	2	10.23	14.77	82.89	0.00	10.00

Note. Q = quality; T = paid capacity; D = discount support.

Three routing constructs were calculated. For model i and benchmark b, r_ib was its empirical percentile rank among the 22 records, with higher values preferred. Composite quality was $Q_i = 100 \times \text{mean}_b(r_{ib})$ over MMLU, HumanEval, MATH, and Arena Elo. For the 19 models with numeric paid limits, capacity was $T_i = 100 \times \text{mean}[\text{pct}(\text{paid RPM}), \text{pct}(\text{paid TPM})]$. Discount support was $D_i = \text{mean}(\text{batch discount \%}, \text{cached-input discount \%})$. This final measure intentionally excluded free-tier status because a binary access flag and paid discount percentages do not share a monetary denominator. Table 3 provides the raw and derived descriptive statistics.

Table 3. Selected numeric descriptive statistics

Metric	n	Mean	SD	Min	Median	Max
MMLU	22	84.51	4.81	75.20	85.30	92.30
HumanEval	22	85.13	6.44	68.30	85.60	95.10
MATH	22	75.73	11.04	55.70	75.20	96.40
Arena Elo	22	1227.82	81.44	1050.00	1225.00	1350.00
Paid RPM	19	497.63	598.37	25.00	300.00	2000.00
Paid TPM	19	2,656,842	3,009,948	40,000	2,000,000	10,000,000
Composite quality Q	22	52.27	26.48	5.68	48.30	97.73
Paid-capacity rank T	19	52.63	27.47	5.26	53.95	88.16
Discount support D	22	22.95	27.72	0.00	12.50	70.00

Note. Q = quality; T = paid capacity; D = discount support.

Capabilities were used as hard constraints and as metadata predictors. Table 4 reports their prevalence. Five capabilities were universal, tool use appeared for 21 models, and image generation appeared for only one. The router required every selected model to satisfy all requested capabilities. A request with image generation and safety filtering therefore had one eligible record; it was routed deterministically rather than treated as a comparative win.

Table 4. Capability prevalence and variance screening

Capability	Models	Prevalence	Varies
Function Calling	22	100.0%	No
Json Mode	22	100.0%	No
Multilingual	22	100.0%	No
Streaming	22	100.0%	No
System Prompt	22	100.0%	No
Tool Use	21	95.5%	Yes
Safety Filters	14	63.6%	Yes
Batch Api	13	59.1%	Yes
Fine Tuning	13	59.1%	Yes
Vision	12	54.5%	Yes
File Upload	10	45.5%	Yes
Code Execution	8	36.4%	Yes
Embedding	5	22.7%	Yes
Web Search	5	22.7%	Yes
Image Generation	1	4.5%	Yes

Figure 2 standardizes the four benchmark values and orders models by composite quality. Figure 3 displays capability prevalence and makes the sparse differentiators visible. The composite was validated against overall tier and the three supplied domain ranks. Because lower task rank is better, negative correlations with coding, reasoning, and multilingual rank were expected. Overall tier was never used as a prediction target or feature because it was closely aligned with the benchmark-derived quality ordering.

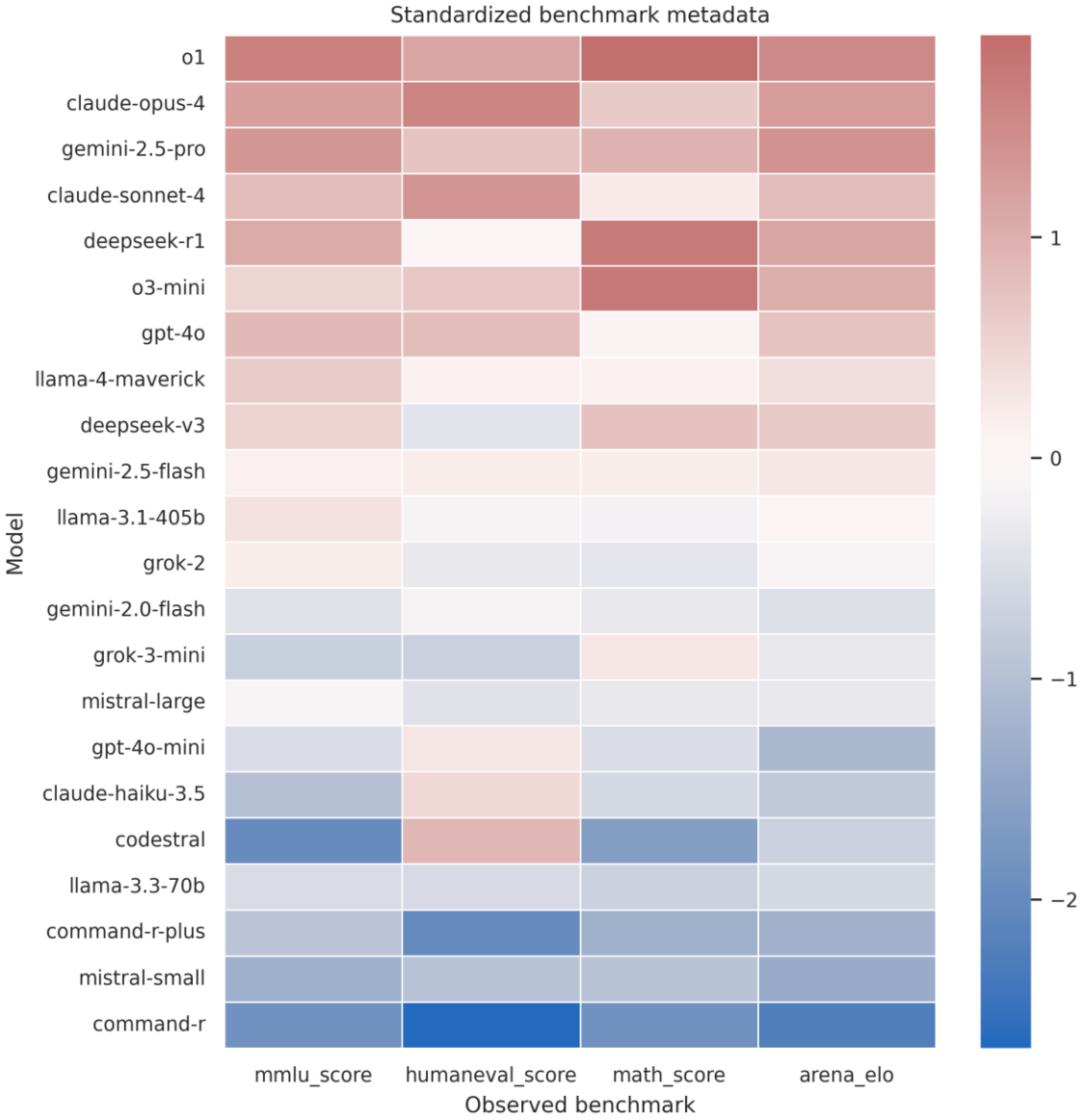

Figure 2. Standardized MMLU, HumanEval, MATH, and Arena Elo metadata ordered by composite quality.

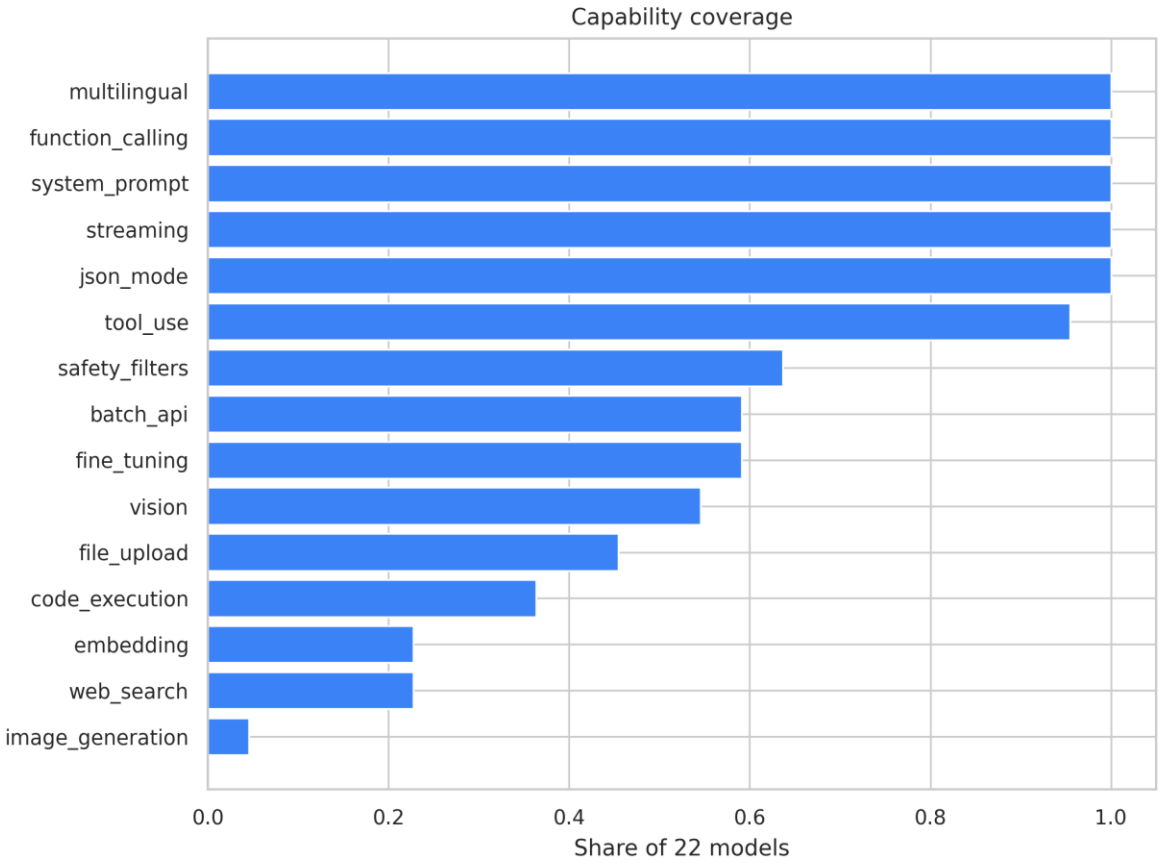

Figure 3. Prevalence of the 15 recorded capabilities across the 22 models.

Table 5 combines construct-validity and sensitivity tests. Spearman correlation measured monotonic association. A Kruskal–Wallis test assessed differences across tier groups. Five sensitivity specifications dropped one benchmark at a time or replaced percentile ranks with min–max scaling. The primary model ranking was accepted only after these checks preserved its leading model and high rank correlation.

Table 5. Composite-quality validity and index sensitivity

Analysis	Statistic	Value	p	Top	Top-5 overlap
Q vs. tier ordinal	Spearman ρ	0.957	0.0000	—	—
Q vs. coding rank	Spearman ρ	-0.861	0.0000	—	—
Q vs. reasoning rank	Spearman ρ	-0.959	0.0000	—	—
Q vs. multilingual rank	Spearman ρ	-0.432	0.0446	—	—
Tier groups (Kruskal–Wallis)	H	19.301	0.0017	—	—
Leave-one-benchmark range	Rank stability ρ	0.973–0.988	—	o1	4 in every run
Min–max alternative	Rank stability ρ	0.990	—	o1	4

The first learning experiment predicted composite quality from ten variable capabilities and seven rate-limit fields. Provider, model, benchmark values, domain ranks, and overall tier were excluded from the predictor matrix. Missing paid-limit values were median-imputed within each training fold and accompanied by missingness indicators. Ridge and k-nearest-neighbor pipelines standardized features; the random forest used the imputed values directly. Ridge alpha was selected from {0.1, 1, 10, 100} by inner cross-validation. K was selected from {3, 5, 7}. The forest used 500 trees, maximum depth 3, minimum leaf size 2, and square-root feature sampling. The fixed seed was 20260717.

Two outer protocols were run. Leave-one-model-out (LOMO) estimated interpolation to a new model record when other providers remained represented. Leave-one-provider-out (LOPO) held out every model from one provider and tested cross-vendor generalization. Performance was measured with MAE, RMSE, R^2, and Spearman ρ. A training-mean predictor supplied the non-informative baseline. A stricter Arena Elo ablation then predicted Arena Elo under LOPO from capability-only, operational-only, and combined metadata. It excluded provider and model identifiers, overall tier, all other benchmarks, and all domain ranks.

Routing followed hard filtering. Enumerating up to three variable capability requirements and collapsing duplicate eligible sets produced 20 candidate sets. Five regimes were used: quality only (1.0, 0.0, 0.0), quality focused (0.7, 0.2, 0.1), balanced (0.4, 0.4, 0.2), throughput focused (0.2, 0.7, 0.1), and discount focused (0.2, 0.1, 0.7). For eligible model i, $U_i = w_{QQ,i} + w_{TT,i} + w_{DD,i}$. Positive capacity weight excluded unknown paid limits. Regret was the observed oracle utility minus selected utility.

Learned policies substituted LOMO out-of-fold quality predictions for Q while retaining observed capacity and discounts. Comparators were observed quality, Arena, throughput, discounts, capability count, training mean, and expected random selection. Evaluation used exact oracle match and regret summaries. A 0.1-step simplex tested 66 weight combinations per candidate set. Bootstrap intervals and paired Wilcoxon tests covered the 100 capability–regime scenarios; their dependence makes these sensitivity tests, not population estimates.

4. Results and Discussion

The benchmark distribution was broad enough to create visible quality separation. As shown in Table 3, Arena Elo ranged from 1050 to 1350, MATH from 55.7 to 96.4, and the composite quality score from 5.68 to 97.73. The top three composite records were o1 (97.73), Claude Opus 4 (89.77), and Gemini 2.5 Pro (88.64). The benchmark heatmap in Figure 2 shows why a composite was useful: Claude Opus 4 was strongest on HumanEval, whereas o1 and o3-mini were especially strong on MATH. A single benchmark would have changed the middle of the ordering even though it did not change the top model.

Construct checks were strong but not independent external validation. Composite quality correlated .957 with tier ordinal ($p < .001$), −.861 with coding rank, −.959 with reasoning rank, and −.432 with multilingual rank. The tier-group Kruskal–Wallis statistic was 19.30 ($p = .0017$). Dropping any one benchmark produced Spearman correlations from .973 to .988 with the primary index; min–max scaling yielded .990. O1 remained first in every sensitivity specification, while four of the original top five remained in each alternative. These results establish internal coherence, not production accuracy, because the supplied tiers and ranks may derive from related benchmark evidence.

The capacity and feature distributions were more uneven. Paid RPM ranged from 25 to 2,000 and paid TPM from 40,000 to 10,000,000 among numeric records. Google had the highest mean provider capacity rank (83.33), followed by Cohere (82.89) and DeepSeek (75.00), while Anthropic had the highest mean discount support (70.00). Google also had the broadest mean capability count (14.33 of 15). Figure 3 shows that web search and embedding each appeared in only five models, code execution in eight, and file upload in ten. Consequently, capability filtering often mattered more than small quality-score differences.

The learned quality surrogate performed differently under interpolation and provider shift. Table 6 reports every metric. In LOMO evaluation, ridge achieved MAE 13.09, RMSE 14.94, R^2 .667, and ρ .771. KNN and random forest had MAEs of 19.69 and 19.83. Figure 4 shows that ridge preserved much of the ordering but compressed or missed several provider-specific extremes. Under LOPO, ridge MAE rose to 22.35, R^2 fell to −.082, and ρ fell to .503. The

random forest LOPO correlation was .106 and nonsignificant. Holding out a provider therefore exposed structure that a random row split would have hidden.

Table 6. Held-out quality-surrogate and Arena Elo ablation results

Target / protocol	Method	MAE	RMSE	R^2	ρ
Composite / LOMO	Ridge	13.09	14.94	0.667	0.771
Composite / LOMO	Knn	19.69	24.01	0.138	0.495
Composite / LOMO	Random Forest	19.83	23.07	0.204	0.441
Composite / LOPO	Ridge	22.35	26.91	-0.082	0.503
Composite / LOPO	Training Mean	24.08	28.07	-0.178	-0.688
Arena Elo / LOPO	Ridge Capability Only	40.75	51.92	0.574	0.704
Arena Elo / LOPO	Ridge Operational Only	74.24	86.78	-0.190	-0.265
Arena Elo / LOPO	Ridge Full Metadata	75.79	91.49	-0.322	0.359

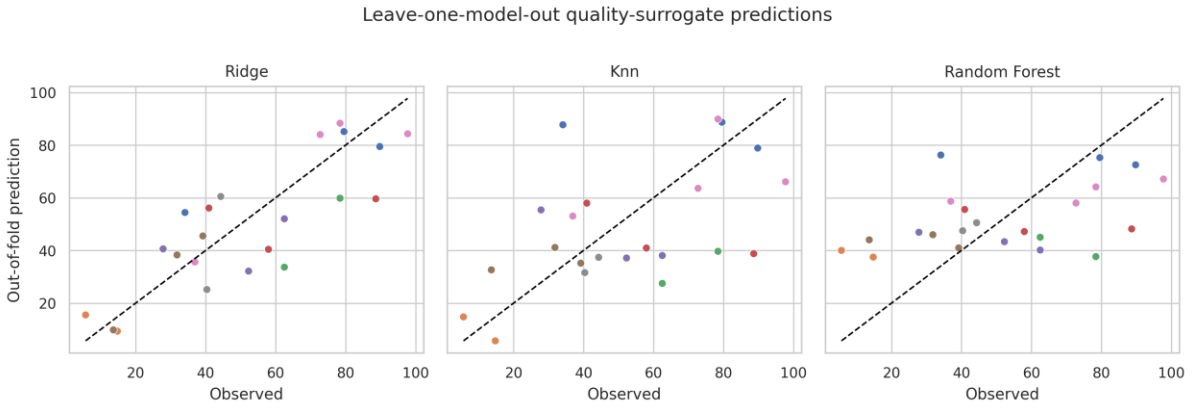

Figure 4. Leave-one-model-out composite-quality predictions for ridge, k-nearest neighbors, and random forest.

The stricter Arena Elo ablation sharpened this conclusion. Capability-only ridge achieved LOPO MAE 40.75 Elo, RMSE 51.92, R^2 .574, and ρ .704. The operational-only model produced MAE 74.24 and R^2 −.190, while combining operational and capability metadata produced MAE 75.79 and R^2 −.322. Figure 5 shows the regression toward the training mean and large errors for provider-specific clusters. Capability presence carried some cross-provider quality signal in this small sample, but rate-limit and discount fields did not generalize as quality predictors. They remained useful as decision criteria, not quality labels.

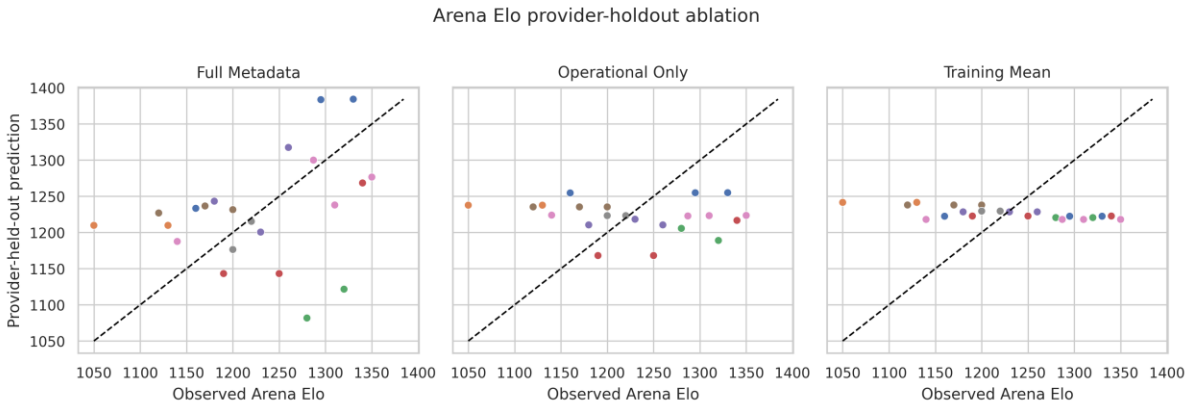

Figure 5. Provider-held-out Arena Elo predictions under full metadata, operational-only metadata, and the training-mean baseline.

The multi-objective frontier in Figure 6 and Table 7 contained eight models: o1, Claude Opus 4, Gemini 2.5 Pro, Claude Sonnet 4, DeepSeek R1, GPT-4o, DeepSeek V3, and Gemini 2.5 Flash. No one of these was dominated simultaneously on quality, numeric paid capacity, and discount support. O1 occupied the maximum-quality extreme; Gemini 2.5 Flash occupied the maximum-capacity extreme; DeepSeek V3 combined a 75.00 capacity score with the maximum 70.00 discount support. Claude Opus 4 and Claude Sonnet 4 remained on the frontier because their 70.00 discount support compensated for low capacity ranks. This is the core rationale for routing: the leading model depends on the operating point.

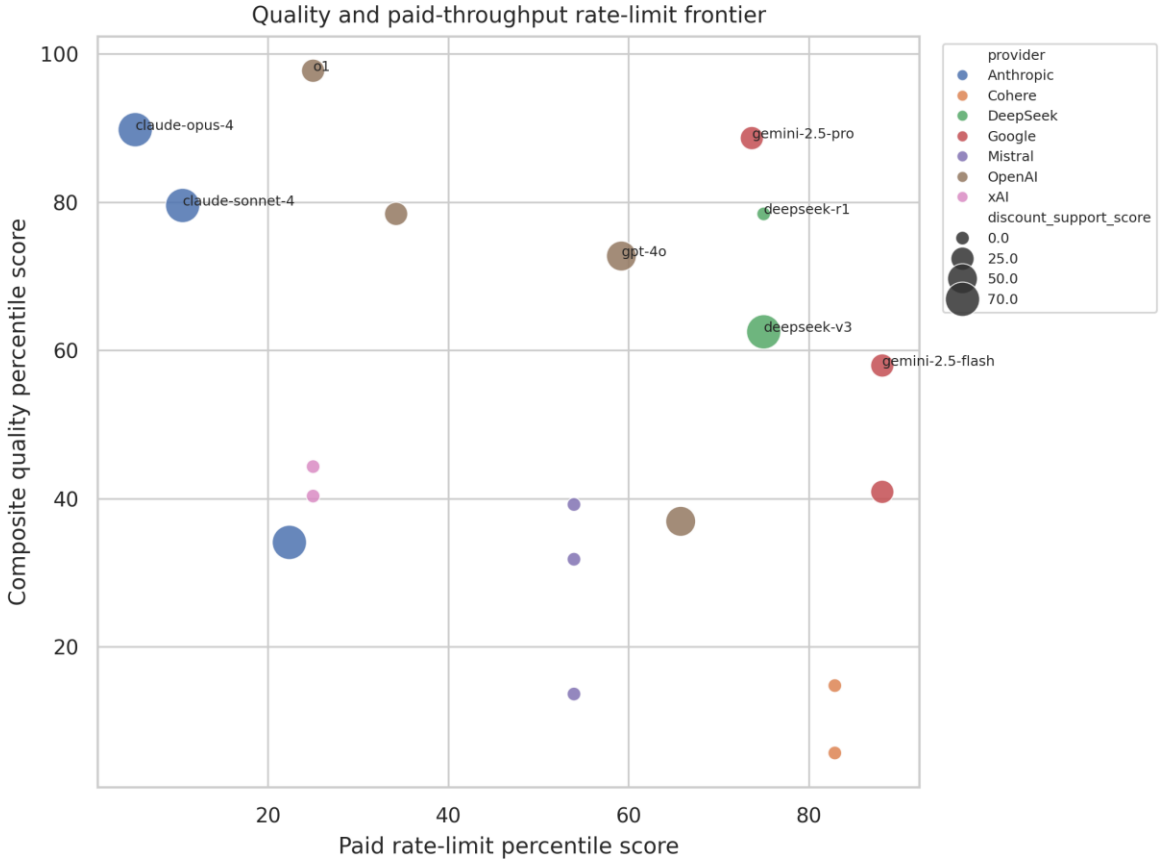

Figure 6. Quality and numeric paid-capacity frontier; marker area represents paid discount support.

Table 7. Non-dominated models on quality, paid capacity, and discount support

Provider	Model	Q	T	D
OpenAI	o1	97.73	25.00	25.00
Anthropic	claude-opus-4	89.77	5.26	70.00
Google	gemini-2.5-pro	88.64	73.68	25.00
Anthropic	claude-sonnet-4	79.55	10.53	70.00
DeepSeek	deepseek-r1	78.41	75.00	0.00
OpenAI	gpt-4o	72.73	59.21	50.00
DeepSeek	deepseek-v3	62.50	75.00	70.00
Google	gemini-2.5-flash	57.95	88.16	25.00

Note. Q = quality; T = paid capacity; D = discount support.

Table 8 defines the five routing regimes, and Table 9 compares policies across all 100 scenarios. The capability-count baseline had the lowest overall mean regret at 5.60, a median of 0, and an exact oracle match rate of .51. Ridge ranked second with mean regret 8.46, median regret 7.15, and exact match .09. Observed-quality-only and Arena-only routing each had mean regret 10.60, while expected random selection had 23.00. Ridge beat expected random decisively, but it did not beat the transparent capability-breadth heuristic. A paired Wilcoxon comparison gave $p = .0135$ for ridge versus capability count, with capability count lower by 2.86 utility points on average. Ridge and observed-quality-only were not distinguishable in this scenario set ($p = .328$).

Table 8. Utility-weight regimes used in the routing evaluation

Regime	wQ	wT	wD	Emphasis
Quality only	1.0	0.0	0.0	Observed benchmark quality
Quality focused	0.7	0.2	0.1	Quality with capacity guard
Balanced	0.4	0.4	0.2	Joint operating point
Throughput focused	0.2	0.7	0.1	Paid RPM/TPM capacity
Discount focused	0.2	0.1	0.7	Batch/cache discount support

Note. Q = quality; T = paid capacity; D = discount support.

Table 9. Overall router performance across 100 capability–weight scenarios

Method	Mean regret	Median	P90	Relative	Exact
Feature Count	5.60	0.00	24.89	0.079	0.51
Ridge Router	8.46	7.15	18.18	0.106	0.09
Observed Quality Only	10.60	3.37	36.26	0.150	0.40
Discount Only	12.47	8.39	47.16	0.161	0.24
Knn Router	14.22	13.30	43.94	0.180	0.07
Throughput Only	16.89	18.58	39.77	0.215	0.20
Random Forest Router	22.32	13.30	51.70	0.272	0.02
Random Expected	23.00	22.38	35.73	0.298	0.13

Note. Q = quality; T = paid capacity; D = discount support.

Figure 7 decomposes regret by weight regime. Single-axis policies won when the oracle used that same single axis: observed quality won the quality-only regime, throughput won the throughput-focused regime, and discount support nearly matched the discount-focused oracle. Capability count had zero mean regret in the balanced and quality-focused regimes because the broadest eligible models also occupied favorable quality–capacity positions in these candidate sets. This result should not be generalized to arbitrary traffic; it demonstrates that a small metadata table can make a simple dominance pattern look stronger than a learned policy.

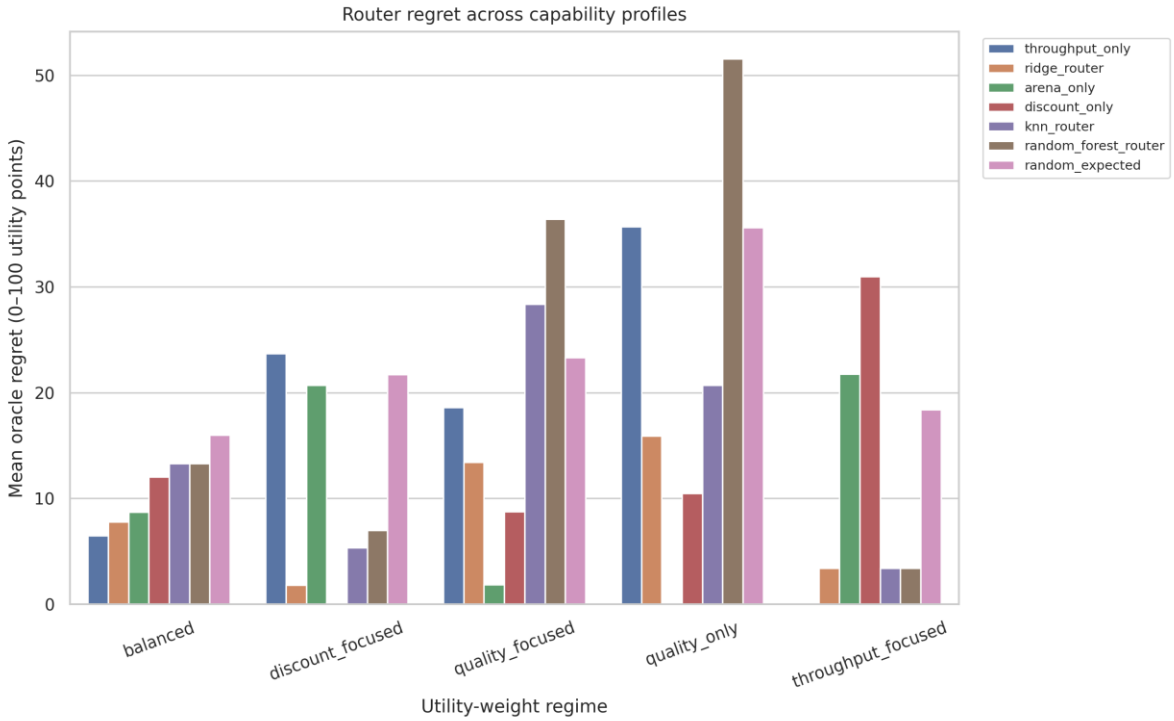

Figure 7. Mean oracle regret by router and utility-weight regime across capability-gated profiles.

Provider selection was concentrated rather than uniform. Figure 8 shows that throughput-only routing selected Gemini 2.5 Flash for every capability profile where it remained eligible. Quality-only routing split primarily between o1 and Gemini 2.5 Pro after feature filtering. Discount-focused routes moved toward DeepSeek and Anthropic

models. This concentration is operationally useful because it reveals where a nominally multi-provider router still creates vendor dependence.

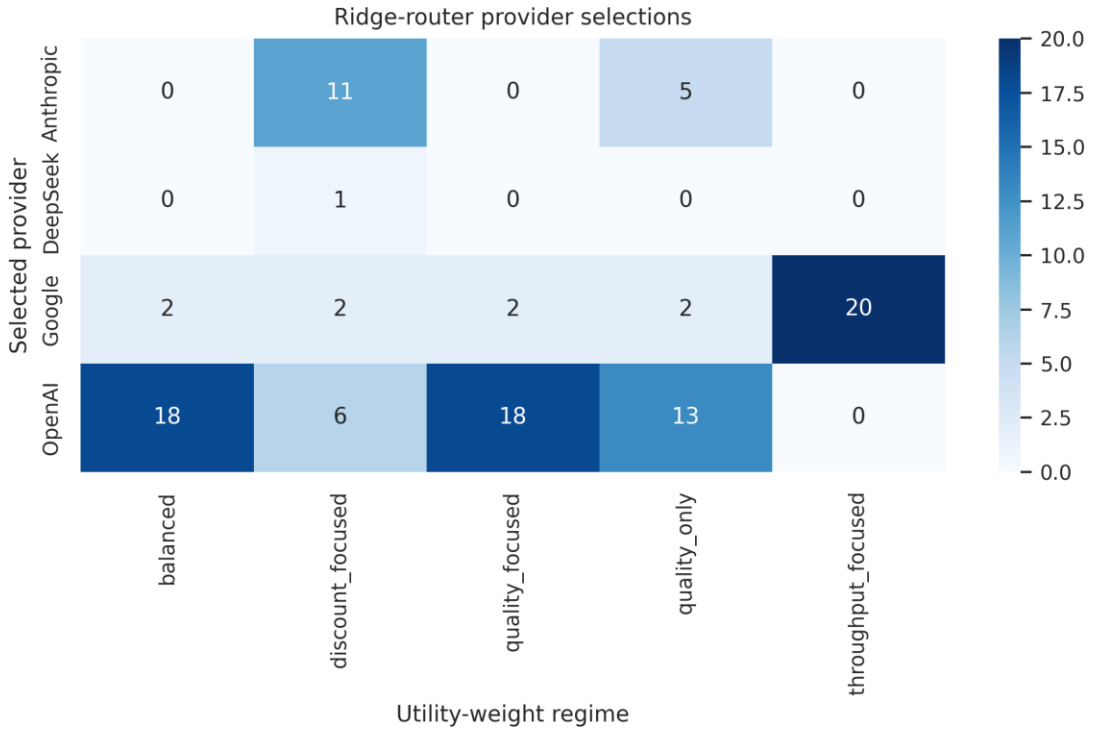

Figure 8. Provider selection shares produced by routing policies under the five weight regimes.

Named enterprise profiles in Table 10 translate the utility function into concrete decisions. With safety and tool use required, Gemini 2.5 Pro led the general assistant profile at utility 75.35, followed by o1 and GPT-4o. Gemini 2.5 Pro also led the complex-reasoning, coding-and-execution, vision/document, web-retrieval, batch, and fine-tuning profiles because it combined high quality, numeric capacity, and broad features. Gemini 2.5 Flash led the high-throughput profile at 75.80. DeepSeek V3 led discount-supported processing at 69.00. Image generation returned Gemini 2.5 Pro as the sole eligible record, so that result is a constraint outcome rather than comparative evidence.

Table 10. Constraint-aware outcomes for named enterprise profiles

Profile	Q/T/D	Eligible	Selected model	Utility
General enterprise assistant	0.60/0.25/0.15	14	Google / gemini-2.5-pro	75.35
Complex reasoning	0.75/0.15/0.10	8	Google / gemini-2.5-pro	80.03
Coding and execution	0.65/0.25/0.10	7	Google / gemini-2.5-pro	78.53
Vision and document analysis	0.50/0.30/0.20	9	Google / gemini-2.5-pro	71.42
Web-grounded retrieval	0.50/0.30/0.20	5	Google / gemini-2.5-pro	71.42

Profile	Q/T/D	Eligible	Selected model	Utility
Batch analytics	0.30/0.40/0.30	12	Google / gemini-2.5-pro	63.56
Fine-tuned extraction	0.40/0.30/0.30	7	Google / gemini-2.5-pro	65.06
High-throughput service	0.20/0.70/0.10	14	Google / gemini-2.5-flash	75.80
Discount-supported processing	0.20/0.10/0.70	18	DeepSeek / deepseek-v3	69.00
Image generation	0.50/0.20/0.30	1	Google / gemini-2.5-pro	66.56

Note. Q = quality; T = paid capacity; D = discount support.

Figure 9 presents the 0.1-step sensitivity slice at discount weight .1 for the ridge router. Regret was lowest when quality weight was small and capacity weight was large, because the operational metadata directly identify capacity. Regret rose when quality dominated, reaching 13.4 points at weights (.7, .2, .1). Table 11 summarizes the best policy within each named regime. The weight analysis confirms that model recommendations are conditional: o1 is defensible when benchmark quality dominates and required features permit it; Gemini Flash is defensible when numeric throughput dominates; Gemini Pro is favored when capability breadth and balanced utility matter; DeepSeek V3 and Anthropic models become attractive when discount support receives more weight.

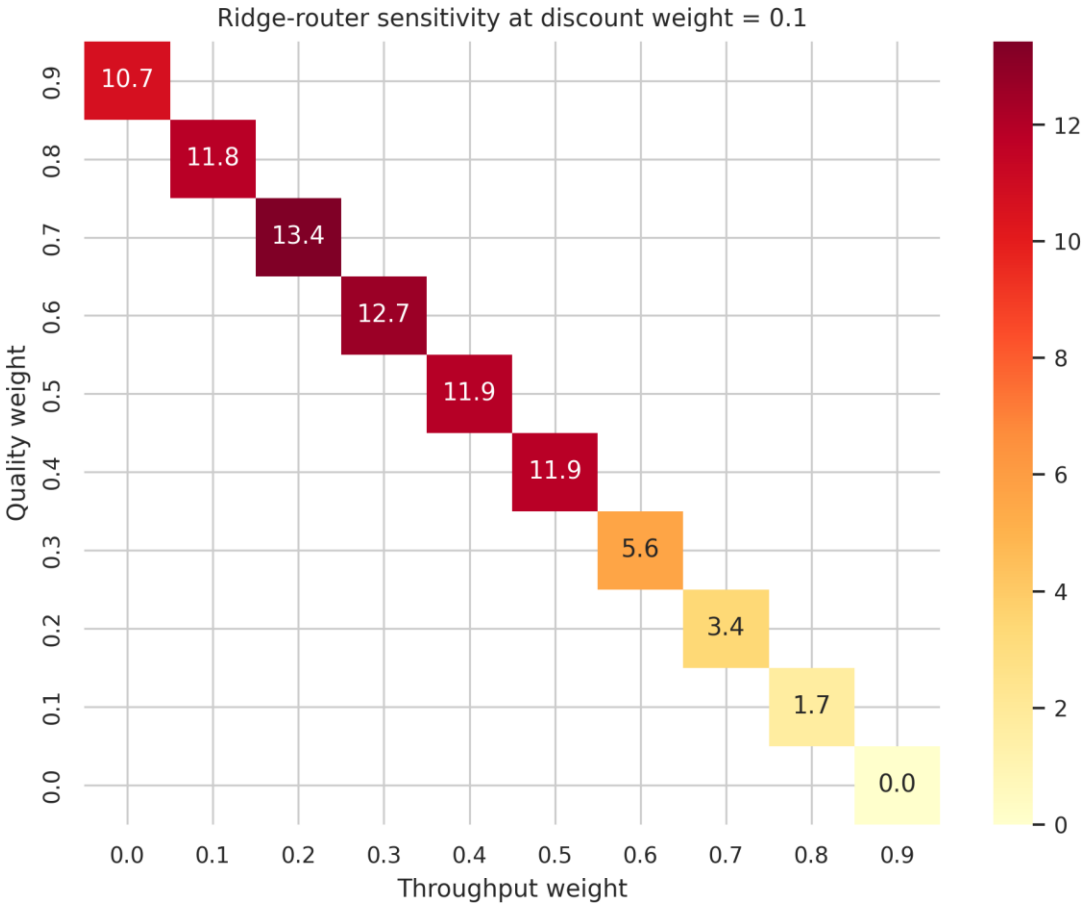

Figure 9. Ridge-router regret along the quality–capacity trade-off when the discount-support weight is 0.1.

Table 11. Best and second-best policies within each utility regime

Regime	Best policy	Regret	Exact	Second policy	Regret
Quality Only	Arena Only	0.00	1.00	Observed Quality Only	0.00
Quality Focused	Feature Count	0.00	1.00	Arena Only	1.86
Balanced	Feature Count	0.00	1.00	Throughput Only	6.48
Throughput Focused	Throughput Only	0.00	1.00	Knn Router	3.41
Discount Focused	Discount Only	0.15	0.90	Ridge Router	1.80

Note. Q = quality; T = paid capacity; D = discount support.

The router abstained from an "open-source" route because the data contain no license, weights, hosting, hardware, or deployment-cost fields. It likewise abstained on exact budget, context fit, and latency SLA. Context claims need behavioral evidence (Bai et al., 2024; Hsieh et al., 2024), while RPM/TPM cannot replace latency traces (Agrawal et al., 2024; Zhong et al., 2024). Production use should add token prices, traffic distributions, latency traces, effective-context tests, privacy controls, and verified licenses.

The study covers 22 models at one time point. Binary capabilities do not express reliability, benchmark aggregates are not request-level outcomes, and the dependent scenarios are not observed demand. Three models lack numeric

paid limits; no absolute cost was estimated; and provider effects weakened LOPO performance. The resulting architecture therefore combines learned scores with hard constraints, transparent weights, monitoring, and abstention. Risk calibration would additionally require request-level data such as those used by conformal gates (Uddin & Bauer, 2026).

5. Conclusions

This study completed a full metadata-based evaluation of an enterprise LLM router across 22 models, eight providers, 20 distinct capability-gated candidate sets, five operating regimes, and a complete weight sensitivity grid. The data supported three defensible constructs: benchmark quality, paid rate-limit capacity, and discount support. They did not support measured latency, dollar price, context-window effectiveness, or deployment-license claims.

The central result is that transparent structure mattered more than model complexity. Ridge regression predicted composite quality reasonably when one model was held out, but its accuracy deteriorated when an entire provider was unseen. Capability-only metadata predicted provider-held-out Arena Elo better than rate-limit metadata, and the capability-count heuristic achieved lower routing regret than every learned policy. The Pareto and named-profile analyses nevertheless produced useful conditional guidance: o1 for maximum benchmark quality, Gemini 2.5 Pro for balanced capability-constrained work, Gemini 2.5 Flash for high numeric throughput, and DeepSeek V3 or Anthropic models when paid discount support is emphasized.

The practical design is therefore a capability-first, multi-objective router with explicit uncertainty boundaries. It should filter infeasible models, score only measured dimensions, expose weights, monitor vendor concentration, and abstain when required metadata are absent. Adding request-level outcomes, real prices, latency traces, effective-context tests, and verified license fields would permit a later router to answer the remaining cost, SLA, and self-hosting questions directly.

References

[1] Aggarwal, P., Madaan, A., Anand, A., Potharaju, S. P., Mishra, S., Zhou, P., Gupta, A., Rajagopal, D., Kappaganthu, K., Yang, Y., Upadhyay, S., Faruqui, M., & Mausam. (2024). AutoMix: Automatically mixing language models. In *Advances in Neural Information Processing Systems* (Vol. 37). https://arxiv.org/abs/2310.12963

[2] Agrawal, A., Kedia, N., Panwar, A., Mohan, J., Kwatra, N., Gulavani, B., Tumanov, A., & Ramjee, R. (2024). Taming throughput-latency tradeoff in LLM inference with Sarathi-Serve. In *18th USENIX Symposium on Operating Systems Design and Implementation* (pp. 117–134). https://www.usenix.org/conference/osdi24/presentation/agrawal

[3] Anthropic. (n.d.). *Rate limits*. Retrieved July 17, 2026, from https://platform.claude.com/docs/en/api/rate-limits

[4] Bai, Y., Lv, X., Zhang, J., Lyu, H., Tang, J., Huang, Z., Du, Z., Liu, X., Zeng, A., Hou, L., Dong, Y., Tang, J., & Li, J. (2024). LongBench: A bilingual, multitask benchmark for long context understanding. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics* (pp. 3119–3137). https://doi.org/10.18653/v1/2024.acl-long.172

[5] Chen, L., Zaharia, M., & Zou, J. (2020). FrugalML: How to use ML prediction APIs more accurately and cheaply. In *Advances in Neural Information Processing Systems* (Vol. 33, pp. 10685–10696). https://arxiv.org/abs/2006.07512

[6] Chen, L., Zaharia, M., & Zou, J. (2024). FrugalGPT: How to use large language models while reducing cost and improving performance. *Transactions on Machine Learning Research*. https://openreview.net/forum?id=cSimKw5p6R

[7] Chen, M., Tworek, J., Jun, H., Yuan, Q., Pinto, H. P. de O., Kaplan, J., Edwards, H., Burda, Y., Joseph, N., Brockman, G., Ray, A., Puri, R., Krueger, G., Petrov, M., Khlaaf, H., Sastry, G., Mishkin, P., Chan, B., Gray, S., . . . Zaremba, W. (2021). Evaluating large language models trained on code [Preprint]. *arXiv*. https://doi.org/10.48550/arXiv.2107.03374

[8] Chiang, W.-L., Zheng, L., Sheng, Y., Angelopoulos, A. N., Li, T., Li, D., Zhu, B., Zhang, H., Jordan, M. I., Gonzalez, J. E., & Stoica, I. (2024). Chatbot Arena: An open platform for evaluating LLMs by human preference. In *Proceedings of the 41st International Conference on Machine Learning* (pp. 8359–8388). https://proceedings.mlr.press/v235/chiang24b.html

[9] Ding, D., Mallick, A., Wang, C., Sim, R., Mukherjee, S., Ruhle, V., Lakshmanan, L. V. S., & Awadallah, A. H. (2024). Hybrid LLM: Cost-efficient and quality-aware query routing [Preprint]. *arXiv*. https://arxiv.org/abs/2404.14618

[10] Google. (n.d.). *Rate limits*. Retrieved July 17, 2026, from https://ai.google.dev/gemini-api/docs/rate-limits

[11] Hendrycks, D., Burns, C., Basart, S., Zou, A., Mazeika, M., Song, D., & Steinhardt, J. (2021a). Measuring massive multitask language understanding. In *International Conference on Learning Representations*. https://openreview.net/forum?id=d7KBjmI3GmQ

[12] Hendrycks, D., Burns, C., Kadavath, S., Arora, A., Basart, S., Tang, E., Song, D., & Steinhardt, J. (2021b). Measuring mathematical problem solving with the MATH dataset. In *Advances in Neural Information Processing Systems* (Vol. 34, pp. 3057–3074). https://proceedings.neurips.cc/paper/2021/hash/be83ab3ecd0db773eb2dc1b0a17836a1-Abstract.html

[13] Hsieh, C.-P., Sun, S., Kriman, S., Acharya, S., Rekesh, D., Jia, F., Zhang, Y., & Ginsburg, B. (2024). RULER: What's the real context size of your long-context language models? In *First Conference on Language Modeling*. https://arxiv.org/abs/2404.06654

[14] Hu, Q. J., Bieker, J., Li, X., Jiang, N., Keigwin, B., Ranganath, G., Keutzer, K., & Upadhyay, S. K. (2024). RouterBench: A benchmark for multi-LLM routing system [Preprint]. *arXiv*. https://arxiv.org/abs/2403.12031

[15] Kwon, W., Li, Z., Zhuang, S., Sheng, Y., Zheng, L., Yu, C. H., Gonzalez, J. E., Zhang, H., & Stoica, I. (2023). Efficient memory management for large language model serving with PagedAttention. In *Proceedings of the 29th Symposium on Operating Systems Principles*. https://doi.org/10.1145/3600006.3613165

[16] Li, C., Liu, G., & Zhao, Z. (2026). Cost-aware LLM-style routing for AIOps log analysis: Log parsing, anomaly detection, fault diagnosis, and incident summarization on LogEval task files. *Journal of Technology Informatics and Engineering, 5*(2), 91–103. https://doi.org/10.51903/jtie.v5i2.538

[17] Li, H., Zhang, Y., Guo, Z., Wang, C., Tang, S., Zhang, Q., Chen, Y., Qi, B., Ye, P., Bai, L., Wang, Z., & Hu, S. (2026). LLMRouterBench: A unified benchmark for robust and efficient LLM routing [Preprint]. *arXiv*. https://arxiv.org/abs/2601.07206

[18] Lu, K., Yuan, H., Lin, R., Lin, J., Yuan, Z., Zhou, C., & Zhou, J. (2024). Routing to the expert: Efficient reward-guided ensemble of large language models. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies* (pp. 1964–1974). https://doi.org/10.18653/v1/2024.naacl-long.109

[19] Mohammadshahi, A., Shaikh, A. R., & Yazdani, M. (2024). Routoo: Learning to route to large language models effectively [Preprint]. *arXiv*. https://arxiv.org/abs/2401.13979

[20] Ong, I., Almahairi, A., Wu, V., Chiang, W.-L., Wu, T., Gonzalez, J. E., Kadous, M. W., & Stoica, I. (2025). RouteLLM: Learning to route LLMs from preference data. In *International Conference on Learning Representations*. https://openreview.net/forum?id=8sSqNntaMr

[21] OpenAI. (n.d.). *Rate limits*. Retrieved July 17, 2026, from https://developers.openai.com/api/docs/guides/rate-limits

[22] Shah, S., & Shridhar, K. (2025). Select-then-route: A two-stage framework for efficient LLM routing. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing: Industry Track* (pp. 425–441). https://doi.org/10.18653/v1/2025.emnlp-industry.28

[23] Shen, Y., Liu, Y., Huang, Z., Yin, R., Zheng, X., & Huang, X. (2025). SATER: Cost-aware and latency-efficient LLM routing through speculative answer generation. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing* (pp. 10515–10529). https://doi.org/10.18653/v1/2025.emnlp-main.531

[24] Shnitzer, T., Ou, A., Silva, M., Soule, K., Sun, Y., Solomon, J., Thompson, N., & Yurochkin, M. (2023). Large language model routing with benchmark datasets [Preprint]. *arXiv*. https://doi.org/10.48550/arXiv.2309.15789

[25] Uddin, I., & Bauer, A. (2026). Conformal LLM routing with distribution-free safety guarantees. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics: Student Research Workshop* (pp. 791–799). https://doi.org/10.18653/v1/2026.acl-srw.70

[26] Varangot-Reille, C., Bouvard, C., Gourru, A., Ciancone, M., Schaeffer, M., & Jacquenet, F. (2025). Doing more with less: A survey on routing strategies for resource optimisation in large language model-based systems [Preprint]. *arXiv*. https://arxiv.org/abs/2502.00409

[27] Wang, X., Liu, Y., Cheng, W., Zhao, X., Chen, Z., Yu, W., Fu, Y., & Chen, H. (2025). MixLLM: Dynamic routing in mixed large language models. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies* (pp. 10912–10922). https://doi.org/10.18653/v1/2025.naacl-long.545

[28] Xin, Q. (2025). Hybrid cloud architecture for efficient and cost-effective large language model deployment. *Journal of Information Systems and Informatics, 7*(3), 2182–2195. https://doi.org/10.51519/journalisi.v7i3.1170

[29] Zheng, L., Chiang, W.-L., Sheng, Y., Zhuang, S., Wu, Z., Zhuang, Y., Lin, Z., Li, Z., Li, D., Xing, E. P., Zhang, H., Gonzalez, J. E., & Stoica, I. (2023). Judging LLM-as-a-judge with MT-Bench and Chatbot Arena [Preprint]. *arXiv*. https://arxiv.org/abs/2306.05685

[30] Zhong, Y., Liu, S., Chen, J., Hu, J., Zhu, Y., Liu, X., Jin, X., & Zhang, H. (2024). DistServe: Disaggregating prefill and decoding for goodput-optimized large language model serving. In *18th USENIX Symposium on Operating Systems Design and Implementation* (pp. 193–210). https://www.usenix.org/conference/osdi24/presentation/zhong-yinmin