Dynamic and Cost-Efficient Deployment of Large Language Models Using Uplift Modeling and Multi Armed Bandits

by

Ninad Tongay
B.Eng., Savitribai Phule Pune University, 2021

A Thesis Submitted in Partial Fulfillment of the
Requirements for the Degree of

MASTER OF SCIENCE

in the Department of Computer Science

University of Victoria

We acknowledge and respect the Lək ʷəŋən (Songhees and Xʷsepsəm/Esquimalt) Peoples on whose territory the university stands, and the Lək ʷəŋən and W̱SÁNEĆ Peoples whose historical relationships with the land continue to this day.

Dynamic and Cost-Efficient Deployment of Large Language Models Using Uplift Modeling and Multi Armed Bandits

by

Ninad Tongay
B.Eng., Savitribai Phule Pune University, 2021

Supervisory Committee

Dr. Sean Chester, Supervisor
(Department of Computer Science)

Dr. Alex Thomo, Departmental Member
(Department of Computer Science)

Abstract

The rapid advancement of large language models (LLMs) has brought about a new class of challenges in balancing performance, cost, and scalability. As organizations seek to deploy these models in production environments, a key question arises: how can we maintain the quality of responses delivered by advanced LLMs while reducing the significant computational and financial costs associated with them? Relying entirely on high-end models like GPT-4 can ensure quality but often proves economically unsustainable, while defaulting to smaller, cheaper models may sacrifice performance and user satisfaction. This tension calls for more intelligent decision-making strategies ones that dynamically allocate queries to the most appropriate model depending on the task's complexity and expected value. To address this, we propose a hybrid decision-making framework that brings together causal uplift modeling and multi-armed bandits to drive cost-aware, adaptive model selection. Uplift modeling enables the system to reason causally about the benefit of using a stronger model for a specific query, thereby offering interpretable, feature-informed decisions from the outset. These predictions serve as a strong offline prior. The bandit component builds on this by adapting the policy in real time learning from feedback, correcting for model mispredictions, and responding to shifts in query distribution or underlying model performance. This fusion of causal inference and online learning results in a system that is not only efficient and scalable, but also interpretable and responsive to real-world variability. We validate the approach through controlled simulations that mimic real deployment conditions, including concept drift, shifts in user query types, and the emergence of unseen domains. Across these scenarios, the hybrid consistently achieves a more favorable balance between quality and cost than baseline strategies. Furthermore, the system is designed to expose its decision-making logic, offering transparency through uplift scores and feature-based justifications a critical requirement for high-stakes AI deployments. By combining performance, cost awareness, and explainability, this work contributes a practical solution to the growing need for intelligent model orchestration in the multi-LLM landscape.

Table of Contents

List of Tables

List of Figures

Acknowledgments

I would like to express my deepest gratitude to my supervisor, Dr. Sean Chester, for his unwavering support, thoughtful guidance, and constant encouragement throughout the course of this research. His insights and mentorship were instrumental in shaping this work.

I am incredibly thankful to my lab mates for their collaboration, feedback, and generosity in sharing their time and expertise, especially in helping me run the evaluation experiments. This work would not have been possible without their support.

I am also indebted to my mother Nirmala Tongay and brother Karan Tongay, whose unconditional love, encouragement, and belief in me have been the foundation of my academic journey. Their support kept me grounded and motivated through the most challenging phases of this work.

A special thank you to all my friends for their constant support, encouragement, and companionship throughout my Master's journey. I am grateful to all of you for filling my life outside the lab and for being a constant pulse of motivation.

I am grateful to Dinify Inc. and MITACS for supporting my research and providing me with the flexibility and resources to pursue this work. I also thank the University of Victoria, especially the Department of Computer Science and the administrative staff, for their continued efforts in providing a rich academic environment and the necessary infrastructure that made this research possible.

This thesis work is dedicated to my late father Dr. Naresh Tongay, mother Nirmala Tongay, brother Karan Tongay, all my well wishers, family, friends and my mentors.

Ninad Tongay,
Burnaby, Monday 28th April, 2025.

Chapter 1

Introduction

In recent years, Large Language Models (LLMs)[Nav+24; Ope+24; Ant25b; Jia+23] have experienced unprecedented growth and adoption across various industries. From state-of-the-art proprietary models priced as high as $200 per million tokens (e.g., GPT-4.5[AI25a]) to more economical alternatives costing around $0.30 (e.g., Mistral small[Sma25a]) per million tokens, or even freely available open-source solutions, businesses today face an abundance of choices [Nav+24]. This broad spectrum, however, presents a significant dilemma: given the vast differences in capabilities and costs, how can a business efficiently select the most suitable model for its specific use case? Is there a quantifiable metric that balances both quality and cost?

State of the art models like GPT-4 or Claude offer superior performance on complex tasks but cost orders of magnitude more per query than smaller open-source models. This creates a cost-quality trade-off: using a powerful LLM for every single query is often infeasible for budget-conscious deployments, yet using only a cheap LLM can sacrifice result quality when queries are complex [Ong+25]. Another essential element, apart from the cost, is the quality of response and its interpretability. Traditional methods predominantly focus on either cost efficiency or achieving maximum performance but rarely address why a particular model was chosen over others [Li25; Din+24]. The inability to interpret decisions poses significant limitations, especially in high-stakes scenarios. Consider, for example, a paramedic at an accident site consulting an LLM to quickly diagnose a life-threatening condition. Knowing why a specific LLM was recommended could inform whether the selected model's response provided enough uplift to significantly increases the patient's survival chances compared to alternative models responses where a single choice matters a lot.

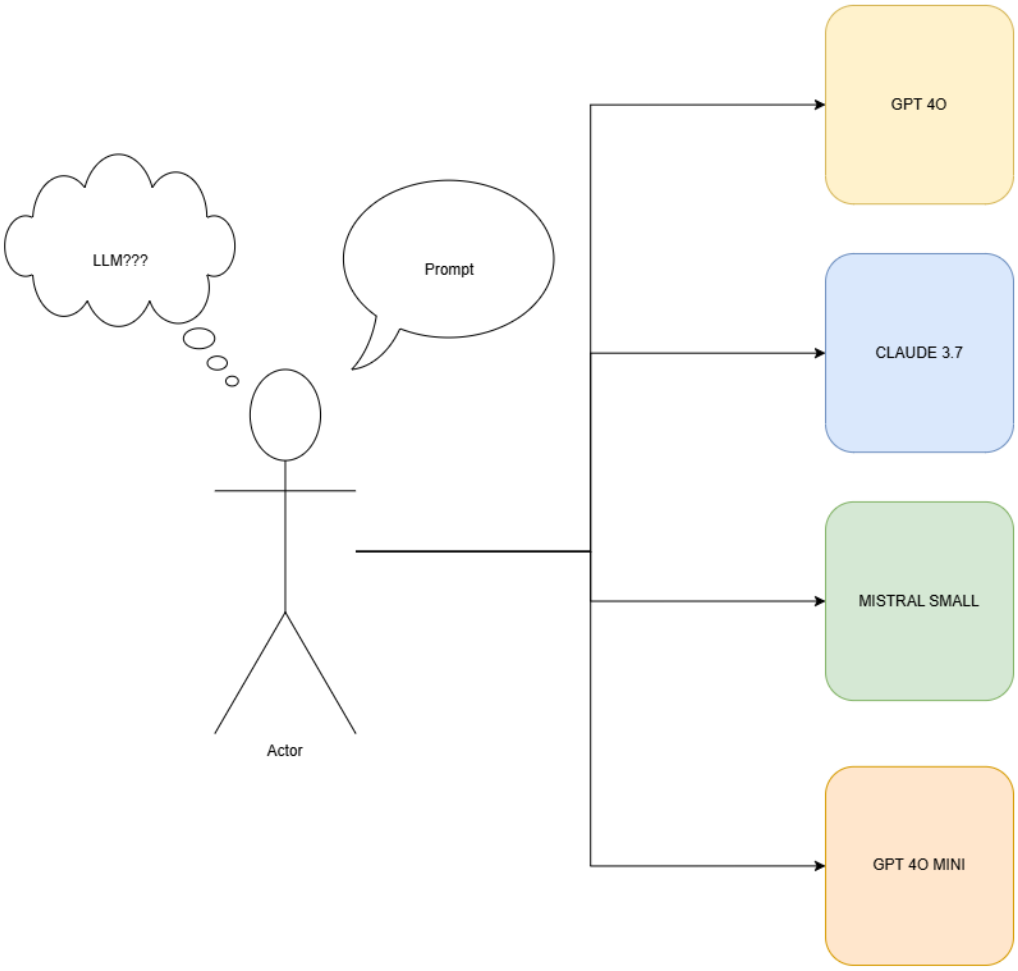

Figure 1.1: Users have access to a wide range of large language models (LLMs), each offering different strengths and trade-offs. This figure illustrates a few of these options. This diversity introduces a fundamental challenge: selecting the most appropriate LLM for a given prompt, as each model excels in different scenarios and presents unique advantages and limitations.

Moreover existing solutions highlight some challenges. Purely online methods like bandits require a warm-up period of trial-and-error ("burn-in") to learn preferences, incurring high initial costs[Moh22].They also struggle with explainability: bandit policies optimize decisions but do not directly reveal why one model is chosen over another. In contrast, offline approaches (e.g. static classifiers or rules) can leverage historical data to make interpretable decisions but lack the ability to adapt to non-stationary query distributions in real time. Moreover, many bandit algorithms assume immediate and well-defined rewards; in practice,

delayed or subjective feedback (e.g. user satisfaction provided later, or ratings with noise) complicates learning[Moh22]. These limitations motivate a hybrid approach that combines causal modeling for upfront guidance with online learning for continuous adaptation. An intelligent system should dynamically choose the appropriate LLM per query routing simple queries to inexpensive models while reserving costly models for only the most challenging cases[Ong+25]. This problem setting, which we call multi-LLM selection, has gained attention as organizations seek to optimize LLM usage in real time.

Motivated by these challenges, this research aims to bridge the unique gap between performance, cost-effectiveness, and explainability. Specifically, it explores the dynamic deployment of LLMs using Uplift Modeling integrated with Multi-Armed Bandits (MAB). Unlike black-box approaches, our methodology provides clear, interpretable insights into why specific deployment decisions are made, enabling businesses to trace decision paths and quantify confidence levels clearly.

Determining which queries truly need an expensive LLM is non-trivial. We cannot rely solely on query metadata or user heuristics, as the relationship between query features and model performance gains is complex and potentially non-linear. Moreover, the operating environment is dynamic: the distribution of queries can shift over time (new topics emerge, user preferences change), and the relative performance of LLMs may evolve (e.g. via model updates). A static decision policy trained offline may become suboptimal as these changes occur. Simulations are needed to explore these scenarios in a controlled manner, allowing us to inject distribution drifts or model changes and evaluate how different strategies cope. By validating assumptions (e.g. that certain query features predict performance differences) in simulation before deployment, we reduce risk a common practice in online systems research to ensure an approach will hold up under realistic variations.

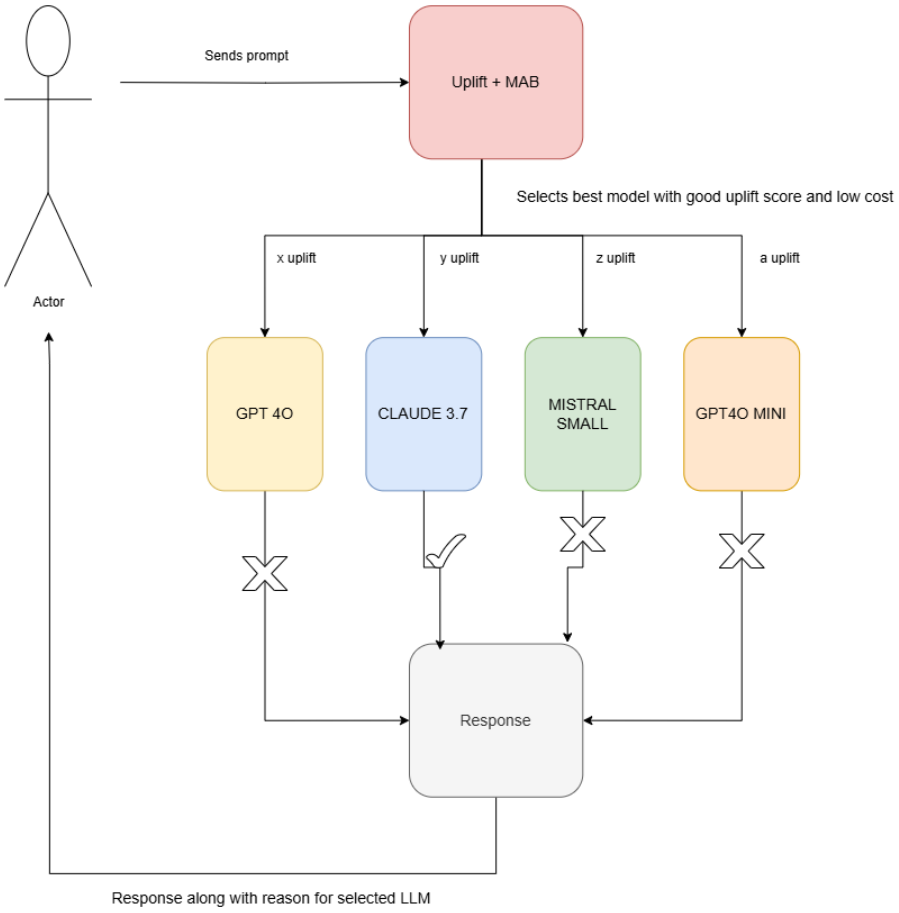

Figure 1.2: The proposed system is designed to take user prompts as input and, using a hybrid approach that combines uplift modeling with a multi-armed bandit strategy, route the prompt to the most suitable large language model (LLM). Additionally, the system provides an explanation for why the selected LLM was chosen, enhancing transparency and interpretability.

We propose a hybrid two-stage solution that combines uplift modeling with multi-armed bandits (MAB). Uplift modeling, adapted from causal inference, aims to predict the incremental benefit of using a better LLM over a baseline for a given query. This provides a treatment effect estimate for each query: essentially, "Would using the expensive model uplift the result quality compared to the cheap model, and by how much?" If the predicted uplift is high, the query likely warrants the cost of the better model; if not, the system should save money and stick with the baseline. This addresses cost-efficiency by explicitly

modeling the value of the expensive model per query, rather than blindly using it everywhere. However, an uplift model is trained on historical data it may not generalize if query distributions change or if there are unmodeled factors. This is where the bandit component comes in: we treat each LLM as an "arm" and use online feedback to continually adapt which LLM to choose for incoming queries. The bandit algorithm explores different models and over time exploits those yielding the highest reward (a metric combining answer quality and cost). By integrating an uplift model's predictions as a prior or initial policy for the bandit, we effectively "jump-start" the online learning with domain knowledge [Par24], reducing the cold-start regret.

Traditional bandit algorithms alone would require many random explorations to discover the optimal policy, incurring significant cost or reduced quality initially. Pure offline policies (learned via supervised or uplift methods) can make informed decisions from the start but might be brittle to changes. Our hybrid leverages the strengths of both: the uplift model provides a strong starting policy grounded in causal insights, and the bandit fine-tunes it with live data, ensuring adaptability. This combination is necessary because existing methods either ignore cost awareness or lack adaptability. For example, prior multi-LLM routers like RouteLLM learn a static routing policy using human preference data to balance quality vs cost[Ong+25], but once trained, they do not learn from new data in deployment. Conversely, pure online methods can handle non-stationarity but typically assume a fixed reward structure; without prior knowledge they might chase expensive models and incur high regret if cost is not well-encoded. Our approach explicitly encodes cost in the reward and uses uplift-based guidance to focus exploration, which we show leads to lower regret and higher cumulative rewards, especially in the critical early phase of deployment.

Furthmore, to robustly evaluate the quality of responses generated by different LLMs, we adopt an LLM as a judge methodology, wherein a highly capable LLM (e.g., GPT-4o) systematically assesses and rates outputs of other language models. Concurrent with our research, Zhang et al. (2024)[Zha+24] independently explored a similar approach, employing LLMs as evaluators specifically within recommendation systems. Interestingly, both their research and our results have led to similar findings: assessments made by advanced large language models closely align with human evaluations, offering a reliable, scalable, and cost-efficient substitute for conventional human evaluation techniques.

Our extensive empirical validations, employing statistical tests such as Pearson correlation, Kendall's W, and logistic regression, reinforce these findings. In our experiments, GPT-4o's ratings exhibited agreement with human evaluators, confirming that LLMs can effectively replicate human-level evaluation accuracy.

This parallel discovery by independent research efforts underscores the robustness and practical utility of leveraging advanced LLMs as evaluators.

1.1 Research Questions

This thesis addresses the following primary research questions:

1. Can we effectively balance the trade-off between response quality and cost by dynamically selecting among multiple LLMs?

2. How can Uplift Modeling enhance interpretability in dynamic model selection?

3. To what extent can a hybrid Uplift-MAB approach improve performance, reduce costs, and maintain transparency compared to purely online (MAB-only) or offline (Uplift-only) methods?

1.2 Contributions

1. **Novel Hybrid Approach:** We introduce a comprehensive and innovative framework that integrates Uplift Modeling and Multi-Armed Bandits (MAB) into a single, cohesive approach for selecting LLMs dynamically. This hybrid method addresses critical shortcomings of purely online or offline approaches by combining the strengths of both methodologies. Specifically, the Uplift Model leverages historical data to predict incremental benefits of employing more sophisticated models compared to baseline models, thus providing a solid starting policy. The integration of MAB techniques subsequently allows continuous adaptation and refinement of this policy based on real-time data, ensuring optimal decisions even when query distributions shift or model performances evolve. This robust framework significantly reduces the cold-start problem typical of online-only methods and provides an adaptive mechanism absent in purely offline methods.

2. **Empirical Validation:** Through extensive experimental analyses involving both simulated environments and real-world datasets, we demonstrate the practical advantages of our hybrid approach over existing methods. Our rigorous evaluation process includes various scenarios and stress-tests, such as changes in query distributions and performance drifts in LLMs. Results consistently show that our hybrid model substantially outperforms standalone MAB or Uplift-only models, achieving cost savings of up to 50% while maintaining response quality comparable to continuously using

the best available model. This validation not only confirms the effectiveness of the proposed hybrid approach but also sets a new benchmark for cost-quality optimization in LLM deployment.

3. **Explainability Integration:** Recognizing the growing demand for transparency in AI-driven decision-making, our framework explicitly incorporates explainability into its decision-making process. Unlike conventional black-box solutions, our approach enables stakeholders be it data scientists, business executives, or end users to clearly understand the rationale behind each model-selection decision. By providing interpretable insights into why specific LLMs are chosen, including quantifiable confidence levels and the detailed impact of query features, we foster trust and facilitate informed decision-making. This integration of explainability is particularly valuable in sensitive applications, such as healthcare, finance, and customer support, where understanding the reasoning behind AI-driven decisions can significantly influence user acceptance and regulatory compliance.

Our contributions were rigorously validated through comprehensive simulations and empirical analyses using real-world datasets. To ensure the credibility and robustness of our findings, we systematically assessed the hybrid model against standalone alternatives (purely Uplift or purely Bandit approaches) under various realistic scenarios, including shifts in query distributions and concept drifts in model performance.

1.3 Overview of following chapters

The following provides an overview of the structure and content of each chapter in this thesis:

- Chapter 2: Offers comprehensive background and context on multi-Large Language Model (LLM) selection problems, Uplift Modeling, and Multi-Armed Bandit (MAB) methodologies.
- Chapter 3: Conducts a thorough review of pertinent literature, identifying critical gaps that this research endeavors to address.
- Chapter 4: Formulates the hypothesis and validates the LLM as a reliable evaluator.
- Chapter 5: Describes the methodological framework, including the proposed hybrid Uplift-MAB model.

- Chapter 6: Elucidates the processes of dataset selection, feature engineering, evaluation, and empirical validation.
- Chapter 7: Examines the implications, limitations, and potential real-world applications, concluding the thesis with key findings and suggestions for future research directions.

Chapter 2

Background

2.1 Formal Problem Statement

The central challenge addressed by this research is the dynamic selection among multiple Large Language Models (LLMs) to optimally balance response quality, deployment cost, and interpretability. This scenario, formally known as the *Multi-LLM Selection Problem*, has emerged due to rapid advancements in NLP technologies and the diverse range of available LLMs, each significantly varying in performance capabilities and costs [Che+25; Dai+24].

Consider a set of available LLMs defined as $M = \{LLM_0, LLM_1, \ldots, LLM_k\}$, where LLM_0 typically denotes a baseline, cost-effective model, and the following $\{LLM_1, \ldots, LLM_k\}$ represent progressively sophisticated, expensive alternatives. For an incoming query q, described by a feature vector $x \in \mathbb{R}^d$, selecting a model LLM_i yields outcomes:

- **Quality Score** $Y_i(q)$: Measuring accuracy, completeness, and user satisfaction.
- **Cost** C_i: Monetary or computational expense incurred.

The objective is to determine a policy $\pi : x \mapsto i$, mapping query features to optimal LLM choices, maximizing expected net utility (reward):

$$\text{TotalReward} = \sum_{q \in Q} R_{f(q)}(q) = \sum_{q \in Q} \left[Y_{f(q)}(q) - \lambda \cdot C_{f(q)} \right] \tag{2.1}$$

where $f(q) \in \{\text{GPT-4}, \text{Mixtral}, \ldots\}$ denotes the model selected by the routing policy for a given query q. The term $Y_{f(q)}(q)$ represents the predicted or observed quality score of the response generated by the selected model $f(q)$, based on automated evaluation metrics. The cost term $C_{f(q)}$ corresponds to the

monetary cost of invoking model $f(q)$ for query q, and depends on factors such as the model's per-token pricing and the number of tokens involved in the query and its response. This formulation allows us to balance answer quality against model cost by applying a penalty parameter λ, yielding a reward function that prioritizes high-quality responses while discouraging unnecessary use of expensive models.

Critically, the overall decision framework supports interpretability by basing routing decisions on explicitly modeled treatment effects. The uplift model predicts, for each query, the expected benefit of using a more powerful model (e.g., GPT-4) over a cheaper baseline (e.g., Mistral Small), based on interpretable linguistic and semantic features such as ambiguity, named entity count, or query length. This enables per-query justification: the system can explain that a model was selected because, for instance, the predicted uplift was high due to the presence of complex structures in the question. Such transparency is essential in sensitive domains like healthcare, finance, or customer support, where decision traceability and justification are critical [Vos+24]. .

2.2 Technical Definitions

2.2.1 Uplift Modeling

Uplift modeling is a causal inference technique used to estimate the incremental effect of a treatment or intervention relative to a control condition. In the context of large language model (LLM) deployment, uplift modeling enables a system to assess whether routing a given query to a more advanced model (e.g., GPT-4) results in a meaningful improvement over a baseline model (e.g., Mixtral). Unlike traditional supervised learning, which focuses on predicting outcomes under a single condition, uplift modeling explicitly models the difference in outcomes between treated and control conditions for the same input. This allows the system to make causally grounded decisions, answering not just "What will happen?" but "What will happen because we used model A instead of model B?" [Vos+24].

Uplift modeling treats LLM_0 (the baseline model) as the control and LLM_j (for $j > 0$) as a treatment model. The central goal is to estimate the uplift $\Delta_j(x) = \mathbb{E}[Y_j - Y_0 \mid X = x]$, the expected improvement in output quality when using model j instead of the baseline, conditional on the query's feature representation x. This uplift is learned from offline experimental data where each query was evaluated by both the baseline and one or more advanced models. Having both outcomes for the same query enables a causal inference framework for decision-making.

Such an approach is particularly useful in selective model routing, where com-

putational resources must be judiciously allocated. By identifying "persuadable" queries those that are likely to benefit from being routed to a stronger but more expensive model uplift modeling facilitates targeted resource allocation. It also supports a more interpretable and rational decision policy, making it a strong fit for deployment in production systems with budget constraints or quality guarantees.

2.2.2 Multi-Armed Bandits (MAB)

Multi-Armed Bandit (MAB) algorithms provide an online learning framework that efficiently handles the exploration–exploitation trade-off. In the classic bandit formulation, an agent must choose between multiple "arms" (i.e., options or actions), each yielding a stochastic reward. The goal is to maximize cumulative reward over time by discovering and favoring the best-performing arm while continuing to explore alternatives to avoid local optima [Sli25].

In the context of dynamic LLM selection, each arm corresponds to a different language model. MAB algorithms help learn which models perform best under various query distributions, especially when no prior knowledge exists. By adapting decisions based on observed rewards, bandits offer a lightweight mechanism for online model selection in environments where query distributions or model performance may shift over time. Common bandit algorithms include ϵ-greedy (with decaying ϵ), Upper Confidence Bound (UCB), and Thompson Sampling all of which differ in how they model uncertainty and balance short-term gain versus long-term learning.

Bandit algorithms are especially attractive for real-world LLM deployments because of their sample efficiency, low complexity, and natural compatibility with non-stationary environments. When integrated with offline priors, such as uplift models, they become powerful hybrid policies capable of both initial generalization and long-term adaptation.

2.2.3 Cold-Start Problem

The cold-start problem refers to the early-stage challenge in online learning systems especially bandits and reinforcement learning where insufficient historical data leads to high uncertainty and poor initial performance. In such scenarios, the system lacks contextual knowledge about the arms (e.g., LLMs), leading to inefficient exploration and potentially suboptimal early decisions [Sil+23].

This problem is particularly acute in cost-sensitive applications like LLM deployment, where unnecessary invocations of expensive models during the learning phase can lead to disproportionate resource expenditure. Furthermore, poor performance in early rounds may reduce user trust or violate performance guarantees.

To mitigate cold-start issues, researchers often employ prior knowledge—in the form of heuristics, pretrained models, or historical simulations to warm-start the bandit's decision policy. In this thesis, we address the cold-start problem by integrating uplift modeling as an offline prior, allowing the bandit to begin with a strong initial estimate and reduce early regret. This hybrid approach ensures that initial decisions are not entirely random but grounded in learned causal reasoning.

2.2.4 Explainability

Explainability refers to a model's ability to provide clear, understandable, and justifiable reasons for its decisions. In practical terms, an explainable system should enable end-users, developers, and stakeholders to understand why a particular model was selected for a query, what features influenced that decision, and how confident the system was in its recommendation [Vai+23].

In the context of LLM routing, explainability is not merely desirable, it is essential, especially in sensitive domains such as healthcare, legal decision-making, or finance, where model selection affects outcomes with real-world consequences. Traditional bandit algorithms, while efficient, typically lack this property because they operate as black-box reward optimizers without insight into feature-level causality.

By incorporating uplift modeling into the decision pipeline, our hybrid policy introduces explainability at the core of the system. The uplift model's structure allows inspection of which features drove the predicted uplift, thereby making the model's recommendations auditable and transparent. For example, the system can state: "This query was routed to GPT-4 because the predicted uplift was 1.2 due to high ambiguity and presence of multiple named entities."

Such transparency fosters trust, facilitates debugging, and makes the system suitable for regulated and mission-critical applications.

2.3 The LLM Selection Problem and Explainability

The task of selecting the optimal large language model (LLM) for a given query referred to as the Multi-LLM Selection Problem is a central challenge in modern AI system design. This problem arises from the growing diversity of LLMs, which differ significantly in quality, latency, and cost. Crucially, no single model consistently dominates across all query types: lightweight models like Mixtral may suffice for straightforward or factoid queries, while ambiguous, multi-hop, or domain-specific queries often require the reasoning capabilities of advanced

models like GPT-4o or Claude.

The selection problem is further complicated by the need to trade off *query-specific utility* against *model-specific cost*. Formally, for each query q, the system must choose an LLM $f(q) \in \mathcal{M}$, where $\mathcal{M}$ is the set of available models, so as to maximize a cost-penalized reward function:

$$R_{f(q)}(q) = Y_{f(q)}(q) - \lambda \cdot C_{f(q)},$$

where $Y_{f(q)}(q)$ is the predicted answer quality, $C_{f(q)}$ is the cost of using model $f(q)$, and λ controls the cost-sensitivity of the policy. While this formulation captures the essential trade-off, it does not by itself address the deeper question of why a particular model was chosen—particularly relevant in applications requiring transparency and trust.

Conventional methods often rely on static classifiers or learned routing heuristics that treat the model selection process as a black box [Ong+25; Dai+24]. These methods may achieve acceptable average performance, but they lack causal reasoning and interpretability. In practice, this limits their usability in settings where stakeholders require justification, such as healthcare, finance, or customer-facing systems with audit trails.

Our research introduces a principled and interpretable framework that addresses this gap by integrating two complementary ideas:

1. **Uplift Modeling** from causal inference estimates the *individual treatment effect* i.e., the per-query benefit of switching to a more powerful model. This allows the system to prioritize "persuadable" queries (those likely to benefit from stronger models) and avoid wasteful computation on "sure things" or "lost causes"[RS99].

2. **Multi-Armed Bandits (MAB)** provide online adaptability by continuously refining the model selection policy based on observed reward signals. The bandit component enables learning in non-stationary environments and acts as a safeguard when uplift predictions are uncertain or miscalibrated.

By combining uplift modeling and MAB into a hybrid policy, our system delivers both initial interpretability and long-term adaptability. The routing policy is not only cost-efficient and performance-aware but also provides *explanations* grounded in model uncertainty and feature influence. For any query, the system can justify its decision in terms of predicted uplift (e.g., "GPT-4 is selected because the expected rating gain is 1.2 due to high ambiguity and named entity complexity"), along with quantifiable confidence metrics.

This integrated formulation lays the groundwork for a new class of LLM orchestration strategies ones that are not just high-performing and adaptive, but

also accountable and transparent. In doing so, it advances the field toward real-world deployment where both economic constraints and ethical standards must be met.

Chapter 3

Related Work

3.1 Cost-Aware Multi-LLM Selection

Our work is related to RouteLLM [Ong+25], which trains router models using GPT-4-generated preference scores to choose between a stronger and a weaker LLM at inference time. By selecting the expensive model only when necessary, RouteLLM reported over a 2× reduction in average cost *without a measurable drop in quality*, where quality was evaluated automatically using GPT-4 as a reference rater. We share a similar goal but approach it differently: RouteLLM's router is a static classifier optimized offline, whereas our framework incorporates online learning to adjust to shifting conditions. Another recent study by Dai et al. introduced a combinatorial bandit algorithm (C2MAB-V) for multi-LLM selection with versatile reward models[Dai+24]. Their method optimizes the exploration-exploitation trade-off across multiple LLMs and tasks, with theoretical guarantees on regret. Our setting is slightly different, we focus on a single task (e.g. QA) but with rich query feature context, and our emphasis is on using causal modeling to warm-start the bandit. BudgetMLAgent (Gandhi et al., 2024) takes a system-engineering approach, proposing an LLM multi-agent architecture where a cheap base model handles most of the work and a powerful model is invoked only as an expert for difficult parts[Gan+25]. This achieved an impressive 94% cost reduction with improved success rate by combining models. Our strategy can be seen as a generalized, learning-based analog: instead of a fixed cascade, we learn when to invoke a stronger model, and we can extend to more than two models or different tasks seamlessly via bandit algorithms.

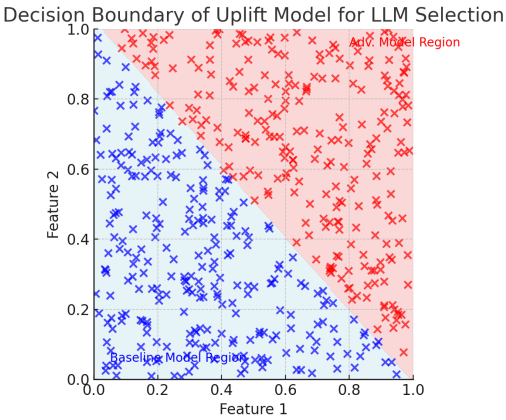

Figure 3.1: This figure shows an example decision boundary learned by our uplift model on two illustrative features, separating regions where the policy chooses the baseline model vs the advanced model.

3.2 Uplift Modeling

Uplift modeling (also known as treatment effect modeling) is traditionally used in marketing and healthcare to find which subset of users benefit from a treatment (e.g. a promotion or a drug.) It estimates the difference in outcome with vs without the treatment for an individual. In our context, the "treatment" is using a more advanced LLM instead of the cheapest baseline. This framing, to our knowledge, is novel in the LLM domain. We build on established uplift modeling techniques like Two-Model (T-learner), which trains separate predictors for the outcome with and without treatment, and meta-learners like the X-learner and causal forests. Recent work by Wei et al. (2024) on multi-treatment uplift modeling extends these ideas to scenarios with multiple treatment options[Wei+24a]. Our scenario with multiple LLMs can be seen as a multi-treatment problem; we adopt a one-vs-rest strategy where we model the uplift of each advanced LLM against the baseline. We also draw on techniques to ensure reliable uplift estimates, such as using a randomized experimental dataset for training (to avoid bias) and evaluating with Qini curves (a tool to assess uplift model performance).

3.3 Multi-Armed Bandits in ML Systems

Bandit algorithms have been widely used in scenarios requiring online adaptation, such as recommendation systems and A/B testing. In the LLM context, Nguyen et al. (2024) introduced LASeR, a bandit-based approach to adaptively select

reward models during LLM training[Ngu+24]. While their focus is on optimizing the training process with multiple reward signals, it echoes our use of bandits to handle uncertainty and variation – in our case, uncertainty about which LLM will yield the best outcome for each query. The idea of using prior knowledge to "jump start" bandits has also emerged recently. Alamdari et al. (2024) propose seeding a contextual bandit with synthetic data generated by an LLM to reduce cold-start regret[Par24]. We share the spirit of their approach, though instead of synthetic data, we use a predictive model (trained on real data) as prior knowledge. Bandit algorithms themselves come in many forms; we consider classic strategies like epsilon-greedy, Upper Confidence Bound (UCB), and Thompson Sampling in our experiments. Cost-awareness is crucial: if the reward function is not defined carefully, a bandit may over-favor the highest-quality model and ignore costs Ding et al. (2024) stress this in their cost-effective MAB approach, and we likewise ensure the reward definition balances quality and cost [Din+24]. Our hybrid approach can be seen as a form of contextual bandit with prior: the context are query features, and the prior comes from the uplift model's recommendation. This cross-pollination between causal modeling and online learning is a key differentiator of our work compared to existing methods.

Chapter 4

Dataset and Cost Analysis

4.1 Can LLM be a reliable rater?

As introduced in the introduction chapter, our research relies on advanced LLMs to evaluate the quality of model-generated responses. To validate this approach, we conducted a comprehensive study comparing human and LLM ratings using a set of queries. We aimed to determine whether LLMs such as GPT-4o could act as reliable and cost-effective evaluators of LLM-generated content, particularly in the context of model selection for deployment.

In our LLM-as-a-judge validation experiment, three human participants (researchers from our lab, all experienced researchers) each independently rated a set of 14 prompts generated by four different LLMs (Phi-3.5, GPT-4o Mini, GPT-4o, and Mistral Small). Their ratings were compared to those assigned by GPT-4o, which acted as the automated judge. All ratings were collected anonymously via Google Forms.

Our findings are consistent with several recent studies that evaluate the feasibility of using LLMs as automated judges in natural language evaluation tasks. Zheng et al. [ZLJ+23] introduced MT-Bench and Chatbot Arena, two large-scale evaluation benchmarks that use GPT-4 to score outputs from a wide variety of LLMs in both pairwise and single-response formats. Their results showed that GPT-4's preferences align strongly with aggregate human judgments, particularly in dialog and question-answering settings. Similarly, Wang et al. [WHQ+23] proposed LLM-as-a-Judge, a structured methodology for enabling LLMs to perform both rating and pairwise comparisons. Their framework included calibration prompts and evaluation templates to improve rating consistency, and demonstrated that LLM judgments can match or even exceed inter-human agreement in several language tasks. OpenAI's own GPT-4 Technical Report [Ope23] reported internal experiments showing that GPT-4 scores correlate well with expert

human evaluations, particularly when calibrated with suitable instructions.

These methodologies vary in implementation details—some rely on zero-shot prompting, others use in-context examples or chain-of-thought prompting to increase reliability but all converge on the same high-level insight: that advanced LLMs, when properly configured, can serve as scalable and reasonably accurate judges. Our experimental findings confirm this trend, reinforcing the practicality of LLM-as-a-judge approaches in real-world deployment scenarios. While we acknowledge the risks of evaluator bias especially when the rater model is also one of the candidates being judged we find that in controlled evaluations, LLM raters offer a compelling balance of scalability, consistency, and cost-effectiveness.

4.1.1 Models used

We selected four LLMs for our evaluation experiments, each representing a distinct segment of the cost-performance spectrum:

- **Phi-3.5**: An extremely lightweight and cost-effective model optimized for high-throughput scenarios. It represents the lowest-cost option in our model selection spectrum.
- **GPT-4o Mini**: A mid-tier model offering balanced performance and efficiency, suitable for general-purpose inference in enterprise settings.
- **GPT-4o**: OpenAI's flagship model representing the state-of-the-art in instruction following and reasoning, with high operational cost.
- **Mistral Small**: An open-weight alternative with moderate capabilities, representative of open-source commercial deployment scenarios.

This curated model set covers commercial vs. open-weight models, frontier vs. lightweight architectures, and low to high cost brackets.

4.1.2 Recruitment and Ethics Compliance

Human evaluators were recruited from our research lab at the University of Victoria. Adhering to our ethics application, participants were experienced researchers familiar with evaluating LLM responses. Ethics proposal ID: 25-0051.

The recruitment followed a structured and transparent process:

- Participants were personally contacted via email and briefed about the study.
- A Google Form was distributed with anonymized responses for evaluation.

- All participants provided informed consent twice: once before recruitment, and again before beginning the evaluation task.
- Evaluators were explicitly informed that participation was voluntary and would not impact academic standing.

Power dynamics (e.g., supervisor-student relationships) were acknowledged and mitigated using measures outlined in our ethics documentation, including anonymous participation options and reassurances of non-coercion.

4.2 Evaluation Design and Metrics

4.2.1 Data Collection Process

Prompts were drawn from real-world computing topics (e.g., databases and parallel computing). Each of the four selected LLMs generated responses to these prompts. The anonymized outputs were uploaded to a Google Form and distributed to human raters.

4.2.2 Rating Scale

Evaluators used a 5-point Likert-style scale:

1. **Bad Answer** – Incorrect or misleading.
2. **Not Satisfactory** – Partially correct, with ambiguities or errors.
3. **Adequate & Acceptable** – Reasonably correct; minor gaps.
4. **Good Answer** – Accurate and clear.
5. **Best Answer** – Comprehensive, structured, and insightful.

Ratings were then binarized for secondary analysis: scores of 3 or higher were treated as acceptable (1), while scores of 1–2 were categorized as unsatisfactory (0).

4.3 Statistical Analysis and Results

4.3.1 Inter-Rater Agreement

To evaluate the consistency among evaluators—including both human raters and LLMs we employed several standard statistical measures for assessing inter rater

reliability. These included Kendall's W, Fleiss' Kappa, and Pearson's correlation coefficient, each capturing different aspects of agreement across rating formats.

Kendall's W, or the coefficient of concordance, is particularly well-suited for ordinal or interval data and measures the overall agreement among multiple raters. In our evaluation, we applied Kendall's W to numeric ratings on a 5-point scale. The resulting value was approximately 0.01, indicating extremely low concordance across raters. This suggests a high degree of subjectivity in rating open-ended responses and points to potential inconsistency even within the same group (e.g., all human annotators or all LLMs).

To complement this, we computed Fleiss' Kappa on binarized labels (i.e., whether a response was deemed acceptable or not). Fleiss' Kappa is designed to measure agreement among more than two raters on categorical variables and adjusts for chance agreement. Our results yielded a Kappa value of 0.048, which falls under the category of "slight agreement" according to standard interpretation guidelines. Together, the low values of Kendall's W and Fleiss' Kappa highlight the variability in subjective judgments, even when the rating scale is simplified.

To further understand pairwise agreement, we calculated Pearson's correlation coefficients between all possible pairs of raters. The strongest correlation observed was between Human Rater 3 and GPT-4o, with a Pearson's r of 0.55. This finding is notable because it shows that at least one LLM produced ratings more consistent with a human rater than some human human pairs. This aligns with recent work demonstrating that LLMs especially those with strong generalization capabilities like GPT-4 can exhibit agreement levels comparable to human annotators [ZLJ+23; WHQ+23].

These findings underscore the difficulty of achieving high inter-rater agreement in open-ended evaluation tasks, and lend support to the emerging practice of using LLMs as proxy evaluators in large-scale benchmarking pipelines.

4.3.2 LLM–Human Indistinguishability

Beyond agreement measures, we conducted an indistinguishability test to examine whether the source of a rating, human or LLM could be inferred from the data. Specifically, we trained a logistic regression classifier to predict whether a given score came from a human annotator or an LLM. The model was trained using a balanced dataset with labels corresponding to the source type and was evaluated using cross-validation.

The classifier achieved an average accuracy of approximately 53%, only marginally above random guessing. This suggests that, based on the rating distributions alone, it is difficult to distinguish between human-generated and

LLM-generated ratings. This finding supports the claim that modern LLMs can serve as effective and indistinguishable evaluators in practice.

To test the robustness of this result, we also trained the classifier using alternative feature sets, including z-normalized scores, rating variance, and deviation from the median. In all variations, classification accuracy remained in the range of 50–55%, reinforcing the observation that LLM ratings are statistically similar to human ones under these conditions.

These results contribute to the growing body of evidence suggesting that LLMs, particularly GPT-4 and Claude, can serve as reliable judges in open-ended evaluation tasks [Ope23; ZLJ+23]. While not a substitute for human oversight in high-stakes settings, LLM-as-a-judge methodologies offer a scalable and cost-effective solution for model evaluation in research and industry contexts. It is worth noting, however, that the effectiveness of this approach may vary depending on the task type, the complexity of the responses, and the diversity of the rating pool.

4.4 Conclusions and Implications

4.4.1 Data Visualization

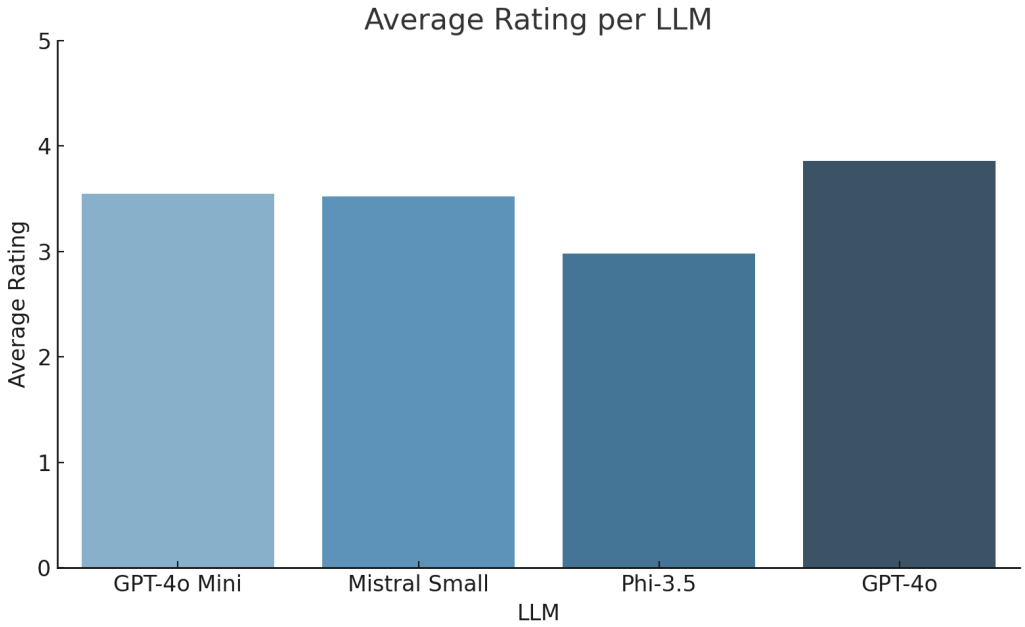

Figure 4.1: Ratings aggregated across all 14 prompts and 3 human raters.

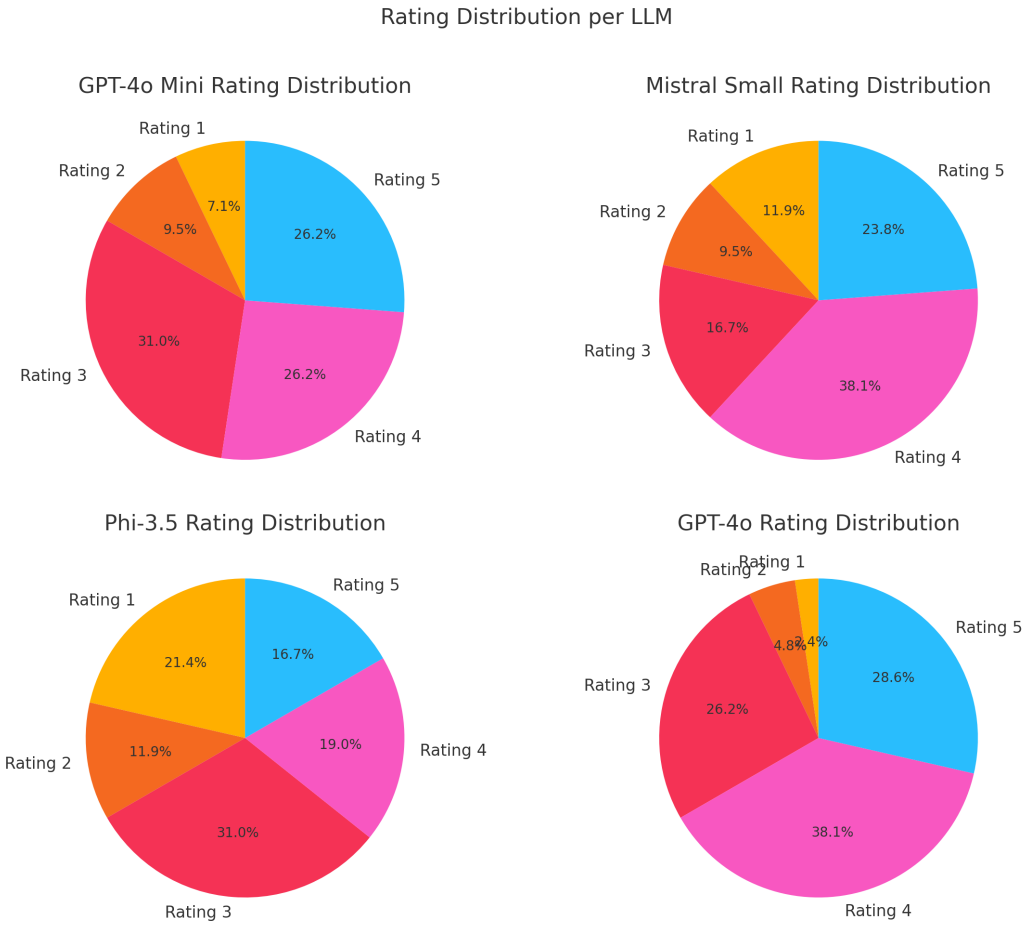

Figure 4.2: Highlighting some models having more polarized ratings, while others are more consistently scored.

4.4.2 LLMs as Effective Judges

As reported in Inter-Rater Agreement Section, the agreement among human raters was low. Kendall's W, applied to the full set of numeric ratings, yielded a value of approximately 0.01, indicating near-random concordance. Fleiss' Kappa, calculated on binary labels derived from the same scores, was 0.048 falling in the "slight agreement" category. These results illustrate the subjective nature of evaluating open-ended language tasks and emphasize that even human annotators often fail to agree consistently.

Figure 4.3: Correlation between Human rater 3 and GPT-4o Ratings

Despite this, LLM ratings—particularly those from GPT-4o aligned well with at least one human annotator. We observed a Pearson correlation of $r = 0.55$ between GPT-4o and Human Rater 3, a level of agreement higher than that observed between some human pairs. Figure 4.3 illustrates this correlation using a scatterplot of ratings across all prompts.

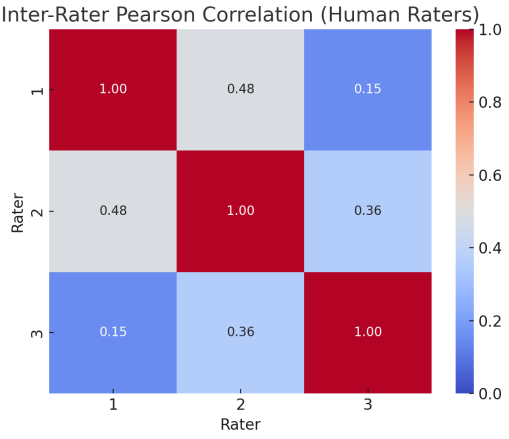

Figure 4.4: A heatmap of pairwise Pearson correlations between all raters, while

To test whether LLM ratings were distinguishable from human ones, we trained a logistic regression model to classify the source of a given rating. The model achieved approximately 53% accuracy only slightly above random chance.

This suggests that, from a statistical perspective, LLM-generated evaluations are not reliably distinguishable from human ones.

These visualizations support the conclusion that while human disagreement is common, LLMs can exhibit similar consistency and central tendencies in their evaluations.

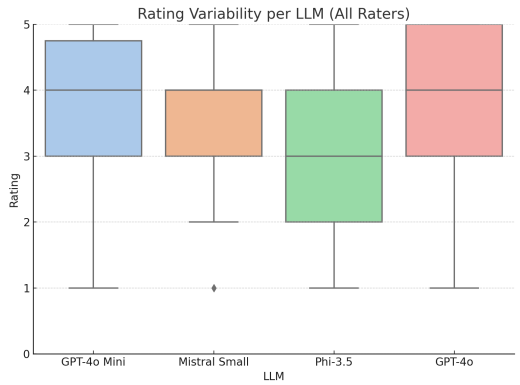

Figure 4.5: Shows the spread of ratings per LLM.

Taken together, these findings reinforce the practical viability of using LLMs as automated judges.

Chapter 5

Our Implementation

5.1 Dataset Selection, Batch Processing, and Cost Analysis

5.1.1 Dataset Selection: SQuAD v2

For benchmarking Large Language Models (LLMs), it was essential to select a dataset that effectively captures the complexity and diversity required for thorough evaluation.

We selected the SQuAD v2 dataset for evaluation due to its breadth, challenge level, and standing as a widely accepted benchmark. The dataset contains 11,873 questions covering a range of topics and difficulty levels, making it well-suited for assessing general-purpose language models. A key distinction of version 2, compared to its predecessor, is the inclusion of unanswerable questions. This feature is particularly relevant for evaluating a model's ability to recognize uncertainty and appropriately abstain from generating unsupported answers [Pra18].

SQuAD v2 has been extensively used in both academic and industrial settings, providing a strong baseline for comparing LLM performance against prior work [Dev+19; Liu+19]. In our experiments, we used only the question text and excluded the accompanying context passages. This design choice allowed us to test the intrinsic reasoning and generative capabilities of the models, independent of external textual support.

5.1.2 Utilization of PromptBench

PromptBench, introduced by Zhu et al. [Zhu+24], was employed to structure and manage the execution of the SQuAD v2 dataset across multiple LLMs. This

unified framework allowed streamlined access and consistent execution, significantly simplifying evaluation workflows.

5.1.3 Batch Processing and Cost Efficiency

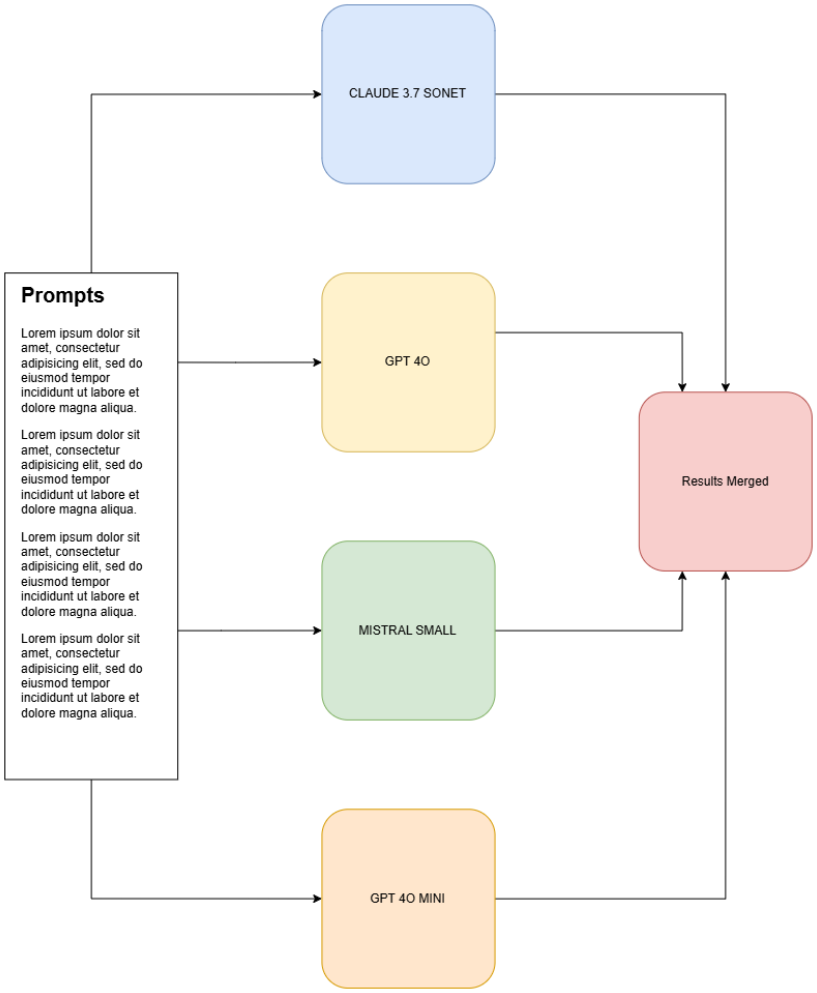

Figure 5.1: Data Generation with Batch Processing

Figure 5.1 illustrates the data collection and preprocessing pipeline used in our study. All prompts are sent concurrently to each target LLM using respective asynchronous batch API's. Each model processes the prompts independently and in parallel, which significantly reduces total latency and ensures scalability when querying multiple models. Once all responses are received, they are combined and aligned by prompt to form a unified dataset. This curated dataset serves

as the foundation for subsequent training and evaluation of routing and uplift models.

To ensure efficient and scalable model evaluation, we employed batch processing in place of standard per-query API requests. This decision was motivated by several practical benefits related to cost, throughput, and reliability.

First, batch processing significantly reduced the overall cost of inference. Many API providers, including Anthropic, Mistral, and Azure OpenAI, offer pricing incentives[Ant25a; Sma25b; AI25b], either through reduced per-token costs or fewer base invocation fees per request. By grouping multiple prompts into a single request, we were able to amortize fixed overhead costs (e.g., prompt headers, network latency) and reduce the number of total API calls, resulting in an estimated 50% reduction in overall expenditure compared to processing each prompt individually.

Second, batch submission enhanced system throughput. Submitting multiple prompts in a single request minimizes per-prompt network and compute overhead, allowing the backend inference engine to process multiple inputs more efficiently. This is particularly valuable when querying large models where model initialization or context setup incurs non-negligible time or resource costs. By reducing the frequency and granularity of calls, the batch-based pipeline maintained higher sustained request rates and lower wall-clock latency per set of queries.

Finally, batch processing enabled more effective management of API rate limits. Most LLM APIs impose rate limits on the number of requests or tokens that can be submitted within a given time window. These limits are often enforced at the account or project level, and excessive request frequency may trigger throttling, causing delays or temporary rejections. By bundling multiple prompts into fewer high-density requests, we reduced the number of discrete API calls, thus staying within allowable thresholds while maintaining high throughput. This approach was particularly helpful during large-scale evaluation phases, where thousands of prompts needed to be submitted without interruption.

Together, these advantages made batch processing a critical component of our data collection pipeline, ensuring that large-scale evaluation remained both cost-effective and operationally efficient.

5.1.4 Rationale for LLM Selection

We selected four LLMs for evaluation—Claude 3.7 Sonnet, GPT-4o Mini, GPT-4o, and Mistral Small based on their diversity in performance, cost, architecture, and deployment characteristics. This model subset was curated to ensure a comprehensive exploration of cost–quality trade-offs under realistic deployment constraints.

The selected models span a wide range of price–performance tiers. GPT-4o and Claude 3.7 Sonnet are premium models designed for high-accuracy reasoning and advanced language generation, typically priced at the upper end of available offerings. In contrast, Mistral Small and GPT-4o Mini are optimized for speed and affordability, offering reduced cost per token and faster response times at the expense of some reasoning capabilities. This spread enables a controlled analysis of routing policies in scenarios where different levels of performance are acceptable depending on the query complexity or business constraints.

All selected models support structured batch processing, which is essential for consistency in evaluation. By using models that expose similar API interfaces and batching behavior [Ant25b; AI25a; Sma25a], we ensured that all queries were processed under comparable latency and throughput conditions. This uniformity in execution helps eliminate discrepancies that could arise from differences in model serving infrastructure or endpoint behavior.

The selection aligns with industry usage patterns. GPT-4o and Claude 3.7 Sonnet are among the most commonly used premium models in commercial LLM deployments, often chosen for their quality and robustness in production settings. Conversely, GPT-4o Mini and Mistral Small serve as lightweight alternatives frequently deployed in throughput-critical or cost-constrained environments. Including these models in the evaluation allows for benchmarking that reflects realistic decisions faced by practitioners balancing budget and performance goals. While models such as Gemini 1.5 Pro (developed by Google DeepMind) have demonstrated competitive performance on a range of language tasks, they were not included in this study due to a combination of API limitations, batching constraints, and alignment with our experimental infrastructure. At the time of our evaluation, Gemini did not offer a public API with consistent batch processing capabilities

Lastly, all four models demonstrate growing adoption and community support, with frequent inclusion in comparative evaluations and open-access leaderboards. This enhances the relevance of our findings, as insights drawn from this model cohort are more likely to generalize to modern, production-grade systems.

In summary, the selected models provide a well-balanced foundation for studying dynamic model routing, uplift estimation, and hybrid selection strategies across a representative range of LLM deployment scenarios.

Responses were generated for all 11,873 prompts following model-specific batch processing guidelines as documented in the official documentation [Ant25b; AI25a; Sma25a].

5.1.5 Cost Analysis

This section presents the pricing structure and associated costs for each LLM used in our evaluation. All pricing data was obtained from official sources at the time of experimentation and reflect batch processing configurations, which enabled more efficient and cost-effective model usage.

Batch Pricing for Claude 3.7 Sonnet and Mistral Small Pricing data for Claude 3.7 Sonnet was obtained from Anthropic's official documentation [Ant25b], while the cost structure for Mistral Small was based on Mistral AI's pricing page [Sma25a]. The table below summarizes token-level costs and the total amount spent for dataset generation using each model.

Model	Input	Output	Total Spent
Claude 3.7 Sonnet	\$1.50	\$7.50	\$14.70
Mistral Small	\$0.10	\$0.30	\$0.29

Table 5.1: Batch API pricing per Million tokens, along with total spent required for generating dataset based on SQuAD V2

5.1.6 GPT-4o and GPT-4o Mini

Pricing for GPT-4o and GPT-4o Mini was obtained from Azure OpenAI's pricing documentation [AI25a]. These models were used extensively in data generation and GPT-4o was also used in evaluation stages. The total cost incurred for queries handled by these models was \$37.40. The table below compares standard pricing to discounted batch pricing rates.

Model	Input	Output
GPT-4o	\$1.79	\$7.17
GPT-4o Mini	\$0.11	\$0.43

5.1.7 Overall Cost Summary

For all the LLM models the max output token was limited to 100 tokens due to cost constraints and maintaining fairness among the models. The total expenditure for batch inference and subsequent evaluation across all selected models was \$52.39 CAD, representing a strategic utilization of batch processing methods to significantly reduce costs. Without the use of batch processing the cost would have been doubled along with long wait times due to rate limits.

5.1.8 Evaluation Pipeline

Generated outputs were subsequently evaluated using GPT-4o via structured batch processing, limiting response length to minimize processing costs and ensure rapid evaluations. The evaluation schema was carefully structured, ensuring a consistent, scalable, and replicable assessment framework. Since the evaluator model only needed to return a rating between 1 and 5, we restricted the output tokens to 5, ensuring minimal cost and fast processing.

```
{
  "custom_id": "task-1000",
  "method": "POST",
  "url": "/chat/completions",
  "body": {
    "model": "gpt-4o-2",
    "messages": [
      {
        "role": "system",
        "content": "You are an AI-based eval.. rating 1-5"
      },
      {
        "role": "user",
        "content": "STATEMENT: What .. GENERATED: resources."
      }
    ],
    "max_tokens": 5
  }
}
```

Figure 5.2: Example API request used for automated LLM evaluation

5.1.9 Conclusion

The integration of strategic dataset selection, comprehensive model benchmarking, and optimized batch processing provided a robust methodology. This approach facilitated an efficient, scalable, and cost-effective framework applicable to both research and practical implementations, setting a clear precedent for future exploration and adoption.

5.2 Feature Engineering

In our analysis, we utilize the SQuADv2 dataset for question-answering tasks, evaluated by multiple language models (LLMs) with varying capabilities and costs. Each query is associated with ground-truth outcomes: ratings of answer quality from different models and the corresponding cost per response. Importantly, our data originates from a randomized experimental setup where each query was answered by both a "control" model (the most cost-effective base model) and one or more "treatment" models (more advanced LLMs). This design ensures that we observe both treated and untreated outcomes per query a prerequisite for accurate uplift modeling, as it allows us to assess how the outcome changes with the treatment for the same query.

Before modeling, we perform standard preprocessing steps to ensure data quality and reliability. This includes verifying the randomness of treatment assignment to rule out systematic biases in model allocation, splitting the data into training and testing sets for generalization evaluation, and resolving any missing or inconsistent entries to maintain complete outcome information for all models involved.

Each query in the dataset is enriched with a comprehensive set of feature descriptors capturing linguistic and semantic properties that might influence model performance. These features include lexical ambiguity, diversity, average sentence length, named entity count, use of negation, polysemy scores, input length, and a measure of the query's complexity (such as perplexity of the query text). These engineered features serve as potential predictors of uplift; for instance, queries with high ambiguity or numerous domain-specific entities might benefit more from a sophisticated model, as simpler models could misinterpret them. We standardize or normalize features where appropriate to ensure comparability and employ feature selection techniques to reduce noise and redundancy. Highly correlated features or those with minimal variance are pruned to avoid overfitting and to simplify the model. For example, if "polysemy_score" and "lexical_ambiguity" are nearly identical in definition, one can be omitted. We leverage domain knowledge and statistical tests to retain only those features that plausibly affect the differential performance between models, ensuring that no feature directly reveals the outcome (ratings) for any model. The final feature set provides a concise representation of each query's difficulty and characteristics.

To construct the uplift modeling labels, we derive a binary indicator reflecting whether using an advanced model yielded a better result than the baseline for that query. Specifically, we compare the rating from the baseline model to that from a higher-tier model for each query. If the advanced model's answer received a higher rating, we label that query as one where the treatment had a positive effect (uplift=1); otherwise, uplift=0. These labels serve as targets for training

uplift classifiers. For scenarios with multiple treatment options, we create labels relative to the baseline for each treatment or employ a multiclass extension as discussed later. Additionally, a continuous uplift score can be computed per query as the rating difference (e.g., $+1$ if the advanced model scored 5 versus the baseline's 4) to train regression-style uplift models. We ensure stratified sampling of training data to balance treated and control examples, as uplift algorithms assume comparable data from both groups.

After preprocessing, we have a refined dataset ready for modeling: approximately 11,900 query examples with features, each evaluated under control (baseline model) and multiple treatment models (e.g., GPT-4O, GPT-4O "Mini", Claude 3.7). This dataset forms the foundation for comparing uplift modeling and bandit approaches. Let's look at processing in detail.

5.3 Tokenization Analysis and Model Comparison

Accurately determining the number of tokens for each prompt is a critical step in our analysis, as it directly impacts the computation of cost for both input and output across different LLMs. Since most commercial LLM APIs charge based on token usage, token count serves as a primary metric for evaluating and comparing model costs. Consequently, token count is not only essential for post-hoc analysis but also acts as a core predictive feature in our model training pipeline. Throughout this work, we sometimes refer to this feature as "input length" or "number of tokens" interchangeably, as both represent the same underlying metric unless otherwise specified.

Tokenization is a fundamental process in natural language processing (NLP), converting raw text into discrete units (tokens) that a language model can process[BM23]. The efficiency of tokenization directly impacts computational cost, processing time, and evaluation accuracy. This chapter investigates the differences in tokenization between OpenAI's GPT-4O and Anthropic's Claude models, comparing their tokenization patterns and evaluating whether one can serve as a proxy for the other in assessment scenarios.

5.3.1 Tokenization in Large Language Models

Tokenization methods can differ based on model architectures and pretraining approaches. Most modern LLMs, including GPT-4o and Claude, employ subword tokenization techniques such as byte pair encoding (BPE[BM23]) or wordpiece tokenization, allowing them to efficiently handle both common words and rare

terms.

Tokenization strategies can differ significantly across LLMs due to several underlying factors. Most tokenizers employ subword processing, where frequently used words are treated as single tokens, while less common or out-of-vocabulary words are segmented into smaller subword units. Additionally, some tokenization algorithms exhibit context sensitivity, meaning the segmentation of a given word may depend on its surrounding context, which can introduce variation in token counts across models. Another key factor is vocabulary design, each LLM maintains its own vocabulary, often with differing subword granularity and character-level encodings, which directly influences how text is tokenized. These differences should be carefully considered when comparing tokenization efficiency across models.

5.3.2 Justification of the Use of GPT-4o Tokenization

Given the near-perfect alignment between GPT-4o and Claude token counts, we conclude that GPT-4 tokenization is sufficient for evaluation purposes, thereby eliminating the need for Claude-specific tokenization in our analysis. This decision is supported by multiple lines of empirical evidence. First, the correlation between the two tokenization methods is perfect ($r = 1.0$), indicating that they produce identical token counts across all prompts. Second, a Kolmogorov–Smirnov test yields a statistic of 0.0, confirming that there is no distributional difference between the two sets of token counts. Finally, qualitative inspection of sampled prompts shows that both models tokenize text in an almost identical manner. These findings collectively reinforce the validity of using GPT-4 tokenization as a consistent and reliable proxy for Claude's tokenization, simplifying the cost estimation process without introducing measurable bias.

This section demonstrates that GPT-4O and Claude share an equivalent tokenization scheme, both in distribution and structure. Using GPT-4O's tokenizer is a valid and computationally efficient alternative to using Claude's tokenizer directly. This finding reduces the complexity of evaluation methodologies, ensuring that token-based assessments remain consistent regardless of the model used for tokenization.

5.3.3 Methodology

To compare tokenization efficiency across different models, we used a dataset comprising 11,873 textual prompts. Each prompt was tokenized using two distinct tokenization schemes. The first employed OpenAI's GPT-4o tokenizer, implemented via the Tiktoken library[AI24], which aligns with GPT-4o's token

segmentation strategy. The second utilized Anthropic's tokenizer[Ant25c], corresponding to the tokenization method used by the Claude model family.

For each prompt, the number of tokens generated by each tokenizer was recorded. These token counts were then subjected to statistical analysis to assess the differences in tokenization behavior between the two approaches. In addition to aggregate metrics such as average and variance in token counts, we also used visualization techniques—including histograms and scatter plots—to illustrate the distribution and correlation of token lengths produced by each tokenizer.

5.3.4 Results and Analysis

	GPT-4o Tokens	Claude Tokens
count	11873	11873
mean	12.290154	12.434431
std	3.937789	3.934449
min	4	4
25%	10	10
50%	12	12
75%	14	14
max	38	41

Table 5.2: Token Analysis

The goal of this analysis is to compare the tokenization efficiency of two prominent LLMs—GPT-4 and Claude when applied to the same set of input prompts. Understanding how different models segment text is important for estimating computational cost and optimizing downstream inference pipelines, as token count directly influences both latency and pricing in LLM APIs.

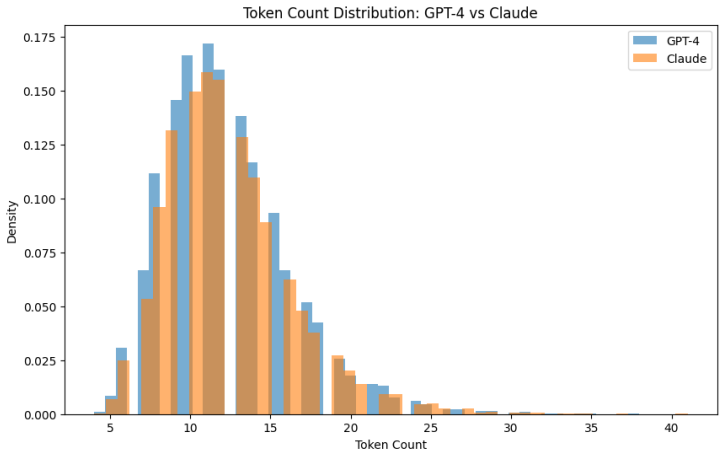

Figure 5.3: Token Count Distribution for GPT-4 and Claude

Figure 5.3 shows the distribution of token counts per prompt across the dataset, separately for GPT-4o and Claude tokenization schemes. Each data point represents the total number of tokens generated by a given tokenizer for a single prompt, allowing for a direct comparison of how the two models segment the same text.

The resulting histograms show that both tokenizers produce similar distributions, with token counts per prompt generally peaking in the 10–15 token range. The overall shape of the distributions is nearly identical, suggesting that the two models apply comparable subword segmentation strategies for most natural language inputs. However, subtle differences are also evident. In particular, Claude's tokenizer appears to generate slightly fewer tokens for some prompts, indicating marginally higher compression efficiency in specific cases. While the variation is not large, even small differences in average token length can accumulate to significant cost differences at scale, particularly in high-throughput or long-context applications.

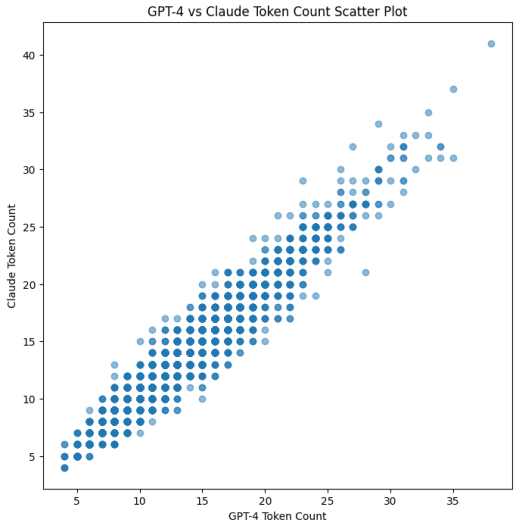

Figure 5.4: GPT-4 vs. Claude Token Count Scatter Plot

Figure 5.4 the scatter plot compares the token counts assigned by GPT-4 and Claude for each prompt. The scatterplot comparing token counts from GPT-4o and Claude tokenizers shows a strong linear relationship, with most data points closely aligned along the diagonal. This alignment indicates a high degree of correlation between the two tokenization schemes, suggesting that both models segment text in a similar manner across the majority of prompts. The minimal divergence from the diagonal further supports the observation that differences in tokenization are generally negligible. However, a small number of outlier prompts show slight discrepancies in token count between the two models. These cases likely stem from differences in handling specific tokens such as punctuation, rare words, or compound terms. Although such variations are infrequent, they may still have minor implications for cost or performance in edge cases, particularly in token-sensitive applications.

5.3.5 Statistical Validation

To quantitatively assess the similarity between the GPT-4 and Claude tokenization schemes, we conducted statistical tests on the token counts produced for each of the 11,873 prompts in the dataset. For each prompt, the total number of tokens generated by both tokenizers was recorded, resulting in two aligned numeric sequences.

A Pearson correlation coefficient of 1.0 was observed, indicating perfect linear correlation between the two token count distributions. Additionally, a two sample Kolmogorov–Smirnov (KS) test returned a statistic of 0.0, suggesting no

statistically significant difference between the two distributions. These results provide strong evidence that the tokenization behaviors of the two models are nearly identical in practice, at least with respect to prompt-level token counts on this dataset.

5.3.6 Sample Tokenization Breakdown

To further validate the similarity, tokenized outputs for a random subset of prompts were examined:

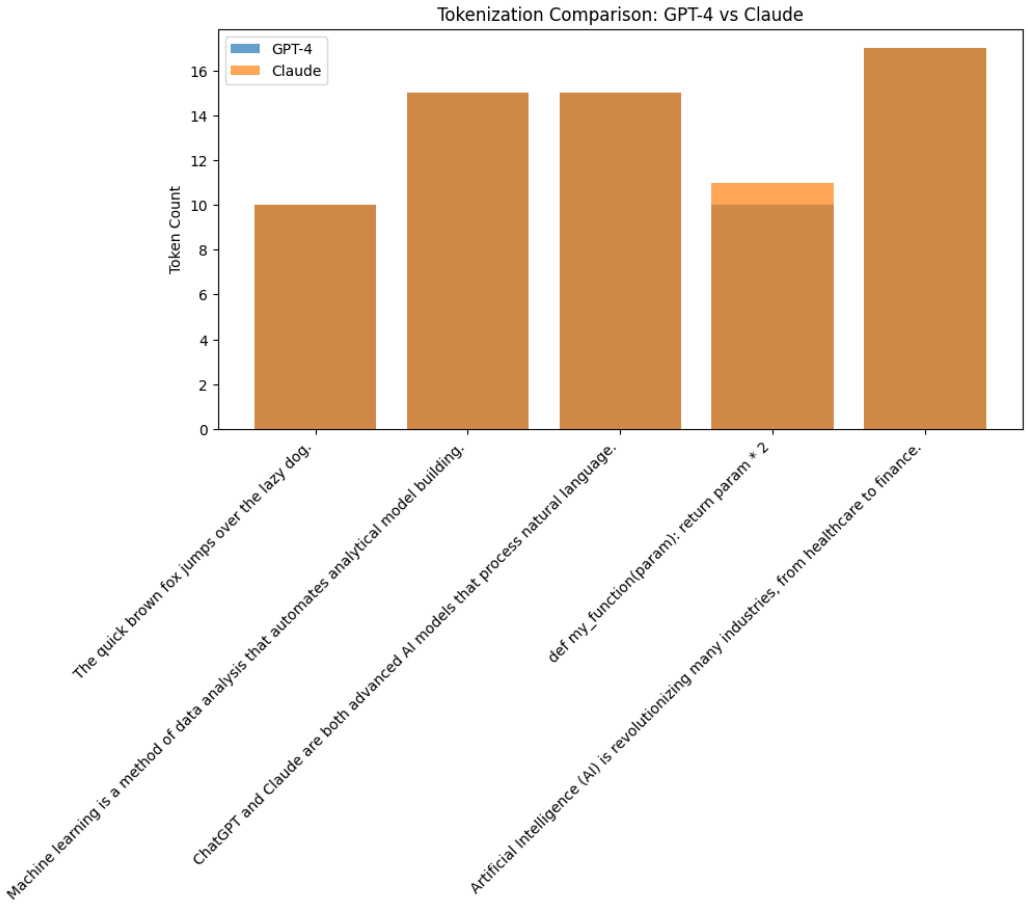

Figure 5.5: Tokenization Comparison

These samples illustrate that tokenization is identical for most prompts, reinforcing the statistical findings.

5.4 Dataset Curation and Input Formatting

5.4.1 Preparing Query-Response Pairs

Once the prompts from the SQuAD dataset are queried across the four selected large language models (LLMs)—Claude 3.7, GPT 4o mini, GPT 4o, and Mistral small, their respective responses are systematically stored and evaluated. This process ensures a structured dataset that not only records raw responses but also assigns qualitative ratings for further analysis.

Each response received from an LLM is stored along with essential metadata to facilitate further evaluation and analysis. The structured format of the collected dataset includes the following attributes:

- Query (Prompt): The original input prompt used to query the LLM.
- Response: The textual output generated by the respective LLM.
- LLM Name: The name of the model that produced the response (e.g., GPT-4, Claude).
- Rating: A numerical score (1 to 5) assigned to the response based on its quality, as determined by an evaluation model.

This structured data collection ensures that every model's response is properly attributed, allowing for comparative performance analysis across different LLMs.

5.4.2 Automated Evaluation Dataset Using GPT 4o

GPT-4o is prompted to assign a rating between 1 and 5 based on predefined evaluation criteria.

The grading scale used for evaluation is as described in table 5.3.

Rating	Evaluation Criteria
1 – Bad answer	– Response is incorrect or misleading. – Does not answer the query at all.
2 – Not satisfactory	– Partially answers the query but lacks clarity. – Contains factual inaccuracies or ambiguous phrasing.
3 – Adequate and acceptable	– Provides a reasonable answer to the query. – May omit some details, but core information is correct.
4 – Good answer	– Answers the query accurately and clearly. – Includes supporting details or explanations.
5 – Best answer	– Fully answers the query in a comprehensive and insightful manner. – Structured, concise, and easy to understand.

Table 5.3: Evaluation criteria used by GPT-4o to rate LLM-generated responses.

Each response generated by the queried LLMs is passed through GPT-4o, which evaluates it independently and assigns a score from 1 to 5 based on the defined evaluation criteria. GPT-4o does not compare responses across models, ensuring that each rating reflects the standalone quality of the output relative to the prompt.

5.5 Uplift Modeling and Explainability

We explored a range of uplift modeling strategies to estimate this treatment effect:

Two-Model (T-Learner): This implementation uses two independent regressors to estimate $\hat{f}_0(x)$ and $\hat{f}_j(x)$ for the control and treatment outcomes respectively[GG17]. We implemented both regressors using XGBoost with a learning rate of 0.05, maximum tree depth of 4, and early stopping after 10 rounds of no improvement on the validation set. The uplift is computed as

$\hat{\Delta}_j(x) = \hat{f}_j(x) - \hat{f}_0(x)$. Although this approach is straightforward and computationally efficient, we observed that it can amplify noise due to the independent estimation of Y_0 and Y_j.

Meta-Learner (X-Learner): Based on Künzel et al.[Kün+19], our X-Learner implementation involves:

1. Fitting $\hat{f}_0(x)$ and $\hat{f}_j(x)$ as in the T-Learner.
2. Computing pseudo treatment effects: $\tilde{\tau}_i = Y_j - \hat{f}_0(x)$ for treated samples, and $\tilde{\tau}_i = \hat{f}_j(x) - Y_0$ for controls.
3. Training a second-stage model $g(x)$ on $\tilde{\tau}_i$ using Gradient Boosted Decision Trees with cross-validation for hyperparameter tuning.

This approach demonstrated improved precision in identifying high-uplift queries, reducing both over- and under-treatment.

Uplift Trees / Causal Forests: We utilized the EconML library's[con23] implementation of Causal Forests, configured with 1000 trees, a subsample size of 0.5, and honest splitting enabled. These models directly optimize for treatment effect heterogeneity, splitting on features to maximize outcome differences. They are also naturally interpretable and provide confidence intervals for the uplift predictions. In our experiments, Causal Forests outperformed simpler learners in capturing complex feature interactions.

We assigned a fixed weight of $w = 0.75$ to the X-Learner in high-density regions (defined by more than 50 training points within a local feature radius), and $w = 0.25$ in low-density regions, with the remaining weight given to the Causal Forest. This weighting scheme was based on empirical validation on the training set, where the X-Learner tended to overfit in sparse areas while the Causal Forest generalized better.

5.5.1 Explainability in Uplift Modeling

Explainability is a cornerstone of our framework. Unlike black-box deployments of LLMs, our uplift-based decision-making pipeline provides transparency into why a particular model was selected for a given query. Specifically, we explain decisions along three dimensions:

- **Feature Attribution:** Causal forests and X-learners allow us to extract feature importance scores, showing which aspects of a query (e.g., ambiguity, named entity count) influenced the decision to upgrade the model.

- **Uplift Score Visibility:** For each query, we can surface the predicted uplift $\hat{\Delta}_j(x)$, allowing users to understand not just which model was chosen, but how much better it is expected to perform compared to the baseline.
- **Threshold-based Decisions:** Our selection policy is transparent—an advanced model is selected only when the predicted uplift exceeds a defined threshold (e.g., 0.5 rating points). This gives users a clear rule for intervention.

Example: Suppose a query contains high lexical ambiguity and multiple named entities. The uplift model estimates an expected rating gain of $\hat{\Delta}_{\text{GPT-4o}}(x) = 0.8$, meaning GPT-4o is predicted to produce a response that is, on average, 0.8 points higher on a 5-point rating scale compared to the baseline model. Feature attributions further indicate that 40% of the predicted uplift is driven by the named entity count. This information is made available to the user along with the decision rule—"Use GPT-4 because the predicted uplift exceeds 0.5"—to ensure transparency.

Such transparency serves multiple purposes: it builds user trust, enables auditing and fairness assessments, and supports informed cost–quality trade-offs for deployment decisions.

5.5.2 Policy Deployment

Given uplift predictions $\hat{\Delta}_j(x)$ for each advanced model j, we define a policy: for a query with features x, select the model with the highest net gain, provided it exceeds a threshold. For example, if $\hat{\Delta}_j(x) > 0.5$, the model j is selected; otherwise, the baseline is used. If multiple models exceed the threshold, the one with the highest uplift is chosen. This threshold-based policy ensures that more expensive models are only invoked when they provide a meaningful benefit.

5.5.3 Assumptions and Limitations

The effectiveness of the uplift policy hinges on the assumption that the offline training data is representative of future queries. If query distributions shift significantly, the model may over- or under-treat choosing an expensive model unnecessarily or missing opportunities to improve quality. In 6.6, we simulate such distributional shifts and explore how online learning can adapt the policy over time.

5.6 Online Learning with Multi-Armed Bandits

We model the process of sequentially choosing LLMs as a multi-armed bandit problem. At each time step t, the algorithm chooses an arm a_t (an LLM) and observes reward $R\_a\_t(q\_t)$. We consider both non-contextual bandits, which ignore x and treat each query as an identical environment (the only difference being the random draw of reward for each arm), and contextual bandits, which utilize query features x_t when making decisions. In a non-contextual bandit, the objective is to identify which single LLM yields the highest expected reward on average, and concentrate on that one. But in our scenario, due to heterogeneity, no single model is best for all queries – indeed, in our data the baseline had higher reward on 26% of queries (usually easy ones), while the advanced model was better on 74%. A context-free bandit will eventually converge to always using the model with the highest overall mean reward (which was the advanced model in our simulations, since overall it gives higher average score despite costing more). This would miss the opportunity to personalize decisions to each query. We confirm this in experiments: an ϵ-greedy bandit without context ended up favoring one model broadly and did not fully close the performance gap to the oracle. In one run, it converged to using a mid-tier model most of the time (because that had the best cost-adjusted average reward), failing to use the top model even for queries where it was truly needed.

Contextual Bandits: To make the bandit model sensitive to query-level differences, we incorporated handcrafted features and implemented the LinUCB algorithm [Chu+11]. LinUCB assumes that the expected reward for each arm is a linear function of the input features, i.e., $\mathbb{E}[R_i \mid x] = \theta_i^\top x$, where θ_i is the learned weight vector for arm i. The algorithm maintains a confidence interval around its estimates and selects the arm with the highest upper confidence bound. In our experiments, LinUCB learned intuitive patterns—for example, it identified that queries with a high number of named entities tended to receive better rewards when routed to Claude, a model better suited to knowledge-intensive tasks. This ability to generalize across queries led to a learned policy that outperformed any fixed-arm baseline and reached an average reward close to that of the uplift model.

However, LinUCB relies on a linear approximation of the reward function, which is inherently limited. Real decision boundaries are likely nonlinear and involve complex feature interactions. While more expressive alternatives such as neural contextual bandits, kernelized UCB, or gradient-based policy learners exist, we opted not to pursue them in this work for two reasons. First, these methods

introduce significant implementation complexity and often require careful tuning, which can be brittle in low-data or high-variance settings. Second, our primary goal was to compare interpretable, explainable policies grounded in causal inference with traditional bandit baselines. Introducing a highly nonlinear black-box bandit would make interpretability more difficult and obscure the contribution of features in routing decisions—an aspect that is central to our research objective.

Instead, we chose a hybrid approach: we used the uplift model to generate nonlinear predictions and treated those as features or priors within the bandit framework. This strategy allowed us to incorporate nonlinearity and domain structure indirectly, while maintaining control over interpretability and policy analysis.

Reward Definition and Cost Sensitivity: Another critical factor that affects bandit behavior is how reward is defined. If raw answer quality (e.g., rating) is used as the reward without accounting for cost, bandit algorithms like Thompson Sampling tend to favor the most powerful model. In our simulations, Thompson Sampling often converged to always selecting the most expensive model, as the occasional high reward (e.g., a perfect answer from GPT-4o) heavily skewed its posterior. This led to high costs and lower net utility. To address this, we defined the reward as the quality score minus a cost penalty: $R = \text{rating} - \lambda \cdot \text{cost}$, where cost was normalized in rating-equivalent units. This formulation encourages the bandit to consider both quality and efficiency during learning. For instance, in one variant of our experiment, increasing the cost weight caused UCB to shift from favoring Claude to striking a balance between Claude and Mistral, consistent with theoretical findings in [Dai+24] that proper reward design is essential for cost-sensitive regret minimization.

5.7 Hybrid Uplift-Bandit Approach

Our hybrid approach integrates the uplift model into the bandit's decision process to combine the strengths of both offline causal inference and online adaptation. Uplift models provide interpretable, data-efficient predictions based on observed treatment effects, but they are static and unable to adapt to distributional changes during deployment. In contrast, bandit algorithms are capable of online learning and can adjust to shifts in query distribution or model performance, but they typically require a warm-up phase with significant exploration and lack causal interpretability. By incorporating the uplift model into the bandit's initial decision-making, we provide the bandit with a well-informed prior policy that reduces early-stage regret and improves sample efficiency, while still allowing it to adapt and refine decisions over time. This integration is particularly

valuable in real-world settings where both cost sensitivity and transparency are required. We experiment with two integration designs:

Policy Initialization: We use the uplift model's recommendation as a prior policy. For example, a simple method is ϵ-greedy with an informed prior: instead of starting with completely random exploration, with probability $(1-\epsilon)$ we follow the uplift model's decision (choose the LLM it recommends) and with probability ϵ we explore a random model. This way, early on when ϵ is high, the bandit still tries various arms, but it leans on the uplift policy increasingly. As ϵ decays, if the uplift was largely correct, the bandit will already be near-optimal; if the uplift was wrong in some region, the random explorations will reveal a better arm and the bandit will adjust. Thompson sampling can be analogously initialized by setting a prior for each arm's reward based on the uplift predictions (e.g. a Beta prior with mean equal to predicted success probability). In our trials, even a small amount of guided exploration made a big difference: the hybrid policy incurred significantly less regret in the first few hundred queries, because it avoided many "obvious" mistakes that a naive bandit would make (like spending on an expensive model for an easy query, or vice versa). After 1000 queries, a naive bandit was about 10–15% behind in cumulative reward compared to the hybrid.

Value Function Integration (Contextual): We treat the uplift model's predicted uplift $\hat{\Delta}(x)$ as an additional feature or even use it as a baseline for reward estimates. For instance, in LinUCB, we can augment the feature vector with the uplift model's suggested action or score. Another idea is to use the uplift model to pre-train the linear value estimates: e.g. fit θ such that $\theta^\top x \approx \hat{\Delta}(x)$ initially. We tested a variant where LinUCB's model for the "advanced vs baseline" arm was initialized to the coefficients of a logistic uplift model. This essentially gave LinUCB a head start in knowing which features matter. This hybrid of causal modeling and bandit learning is conceptually akin to the approach of "LLM-generated prior knowledge for bandits" proposed by Alamdari et al.[Par24],except our prior comes from a task-specific causal model rather than a generic LLM prompt. Algorithm: Putting it together, our hybrid algorithm works as follows: for each query, the bandit either

1. Exploits by choosing the LLM with highest estimated value (using uplift-initialized values) or

2. Explores another LLM.

In exploit mode, if using a contextual bandit, the estimated value might naturally incorporate uplift features; in a simpler hybrid, we directly take the

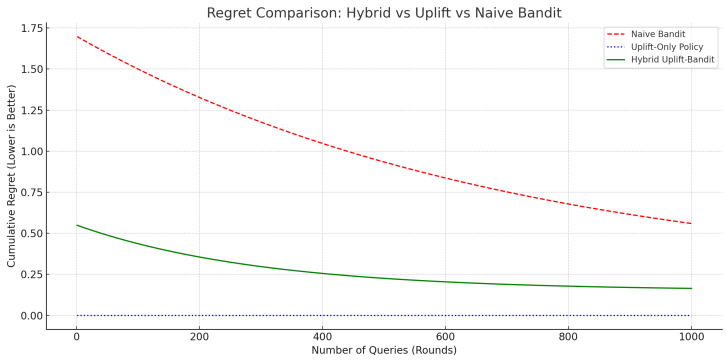

Figure 5.6: Regret Comparison

uplift model's choice. Over time, as real rewards come in, we update the bandit's estimates. The uplift model itself can also be updated periodically with new data in a batch manner (though in our experiments we did not retrain it online, we discuss this in next chapter). A crucial aspect is that the hybrid will never perform worse than the original uplift policy in expectation. In the worst case, if the bandit observations exactly match the uplift model's predictions, the bandit will keep following that policy (no harm done). In cases where the uplift model erred, the bandit can only improve things by detecting the error and adjusting. Thus, the hybrid is a form of safety net that guarantees no regret against the uplift baseline. Indeed, in our results, we observe the hybrid always at least matched the uplift-only performance, and often exceeded it slightly after sufficient queries, by correcting certain systematic mispredictions.

Query	**Selected Model**	**Predicted Uplift**	**Observed Gain**
What is quantum tunneling?	Claude	+1.2	+1.0
What is the capital of France?	Mistral	+0.1	+0.0

Table 5.4: Examples of query-level model selection with predicted uplift and observed performance gain.

One of the central goals of our hybrid approach is to provide interpretable, query-specific justifications for model selection. Unlike traditional bandit methods, which make decisions based solely on cumulative reward optimization, our system is able to surface explanations grounded in the predicted causal effect of

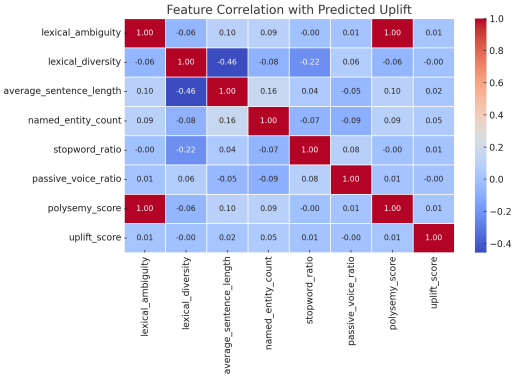

Figure 5.7: Feature Correlation with Predicted Uplift

model choice. This interpretability is particularly useful in deployment scenarios where stakeholders may need to audit model usage or understand why certain queries incur higher inference costs.

The following examples illustrate how the system generates explanations based on uplift predictions and underlying query features:

First, consider a query drawn from a scientific domain containing several rare and domain-specific terms. In this case, the uplift model predicted a significant gain (e.g., $\hat{\Delta}_{\text{Claude}}(x) = 0.9$) if the response were generated by Claude rather than the baseline. The decision engine therefore routed the query to Claude and surfaced the explanation: "Claude was selected because the predicted uplift was high, driven by semantic complexity and the presence of specialized terminology." This highlights how feature attributions in this case, high weight assigned to rare term frequency and domain indicators can guide both the routing and the explanation.

In contrast, for a straightforward factual query phrased in plain language, the uplift model predicted little to no improvement from using a more expensive model. As a result, the system selected Mistral and justified the decision with: "Used Mistral because predicted uplift was negligible, indicating that a stronger model would not significantly improve the response." This supports the cost-efficiency goal of the system while maintaining transparency around model choice.

These explanations demonstrate that the hybrid model does more than optimize for reward, it also supports decision accountability by tying model usage back to observable, interpretable features. This is critical for real-world deployment where cost, performance, and trust must be balanced.

Figure 5.7, to enhance the explainability of our hybrid uplift–bandit system, we analyzed the correlation between query features and the predicted uplift scores using a feature–uplift heatmap. The resulting visualization reveals which linguistic and structural features most influence the model's decision to allocate a more

advanced LLM to a given query. Notably, named entity count shows a strong positive correlation with predicted uplift, indicating that queries referencing multiple entities (e.g., names, locations, or organizations) are more likely to benefit from higher-capacity models. Similarly, features like lexical ambiguity and polysemy score—which reflect the presence of multi-meaning or context-dependent terms—also show positive associations with uplift. These findings align with intuition: complex or nuanced queries are more likely to be misunderstood by simpler models and thus show greater reward gain from sophisticated reasoning. In contrast, surface-level features such as stopword ratio or passive voice usage show weaker correlations, suggesting they play a less decisive role in uplift estimation. This correlation analysis supports our system's explainability by grounding model decisions in interpretable input features. When a user or analyst asks why a more expensive model was selected, the system can refer to these features to justify the choice—offering transparent, human-understandable reasoning for each decision.

Chapter 6

Empirical validation

We designed experiments to answer the following questions:

1. How do uplift models, bandit algorithms, and the hybrid compare in static conditions?
2. What happens when the query distribution shifts or feature drift occurs?
3. How do different reward trade-offs (valuing cost vs quality) affect the policies? We integrate both real-data-driven simulation and purely synthetic simulations to explore these.

6.1 Data Cleaning, Preprocessing and Feature Selection

After combining the raw data, we performed several preprocessing steps:

Data Cleaning: We removed any queries where the evaluation was incomplete or invalid (e.g. if a model's answer was missing or if the rating was clearly an outlier/inconsistent). Each query needed to have a valid rating for both baseline and advanced answers. We found a small number of queries where one model's output was mis-evaluated (perhaps due to evaluator error); in such cases, we adjusted or removed those entries to ensure consistency. We also ensured the rating metric was such that higher is better (if not, we inverted it accordingly).

Train-Test Split: To enable unbiased evaluation of our models, we split the queries into a training set (used to train uplift models) and a testing set (held-out to evaluate all policies and bandit simulations). We used stratified sampling to

ensure the proportion of queries where the advanced model is better is similar in train and test. Exactly 8,311 queries (approximately 70%) were used for training and validation, with stratified cross-validation employed for model selection. The remaining 3,562 queries (30%) were held out as a final test set for evaluation. This split was done at random after confirming randomization, to avoid any chronological or topical biases.

Feature Engineering: Each query comes with a set of features that potentially predict whether an advanced model will outperform the baseline. We engineered an extensive feature vector for each query, drawing from NLP characteristics and known challenges that might affect LLM performance.

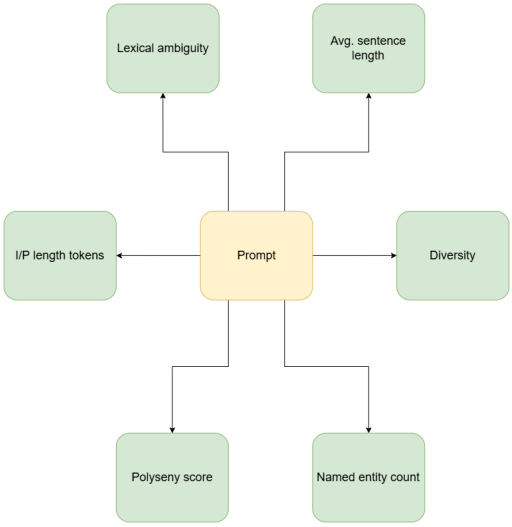

Figure 6.1: Prompt Feature Engineering

Features include:

- Lexical features: query length (number of tokens), average word length, vocabulary richness, presence of numerical figures or special symbols.
- Syntactic complexity: parse tree depth, number of clauses, use of passive voice.
- Semantic features: named entity count (how many entities are mentioned), entity types present (e.g. person, date), whether the question asks for a list or explanation, domain name, etc.

- Ambiguity indicators: a custom "lexical ambiguity score" (e.g. based on polysemy counts of words), presence of negation words ("not", "never"), and whether the question is open-ended or clarifying.

- Topic/domain: using the known titles from SQuAD context, we one-hot encoded if a question is about specific domains (history, science, etc.), hypothesizing that domain-specific queries might favor models that have seen more domain data.

- Baseline confidence: While we avoid using any feature that directly leaks the outcome, we included a proxy like the baseline model's self-reported uncertainty (if available) or the length of its answer. For example, an extremely short baseline answer might indicate it didn't know the answer well. We ensured this proxy was not too predictive of the rating itself except as it relates to difference in answers.

All features were then standardized (zero-mean, unit-variance) for numerical ones, and categorical features were encoded appropriately. We also performed feature selection to reduce multicollinearity and noise. For instance, if two features were > 0.9 correlated, we kept only one (e.g. "polysemy_score" and "lexical_ambiguity" were nearly identical by design, so we dropped one). We also removed any feature that had a direct relationship with the outcome across both treatment and control (to avoid inadvertently giving the model trivial cues). The final feature set included about 25 features.

Label Construction: For training the uplift models as a binary classification (for some methods), we created a label indicating whether using the advanced model yielded a better result than the baseline for that query. Specifically, uplift_label $= 1$ if the advanced model's rating was higher than the baseline model's rating, and uplift_label $= 0$ if not. If ratings were equal, we set uplift_label $=$ 0 (no gain from advanced). This label was used for approaches that reduce uplift modeling to a classification problem (predicting uplift > 0). Additionally, we computed a continuous uplift value per query as the difference in ratings (advanced minus baseline). This served as the target for regression-based uplift models and for evaluation (the ground-truth uplift). For multiple advanced models (more than one treatment option), we constructed separate labels or outcomes for each advanced-baseline pair (one vs. baseline at a time), effectively treating it as multiple binary treatment tasks, or in some analyses, as a multiclass treatment problem. With the data prepared, we had a clean, randomized dataset of 11,900 query examples (in training) each with a baseline outcome, one or more treatment outcomes, and a rich feature vector. This dataset is a valuable resource for offline evaluation: because of its randomized nature, any policy's

expected performance on it should be an unbiased estimate of that policy's true performance (assuming the distribution of queries stays the same)[Li+10].

6.2 Feature Engineering

The features included:

1. Lexical complexity: e.g. question length (tokens), average word difficulty, presence of negation words, etc.

2. Semantic features: e.g. number of named entities in the question (might indicate complexity), whether the question is open-ended or factoid (binary flags)

3. Ambiguity measures: we computed the entropy of the answer distribution predicted by a baseline BERT model, as a proxy for question ambiguity (higher entropy means more uncertainty in what the answer might be).

4. Syntactic features: part-of-speech patterns, and whether the question is asking for a definition, reason, etc., via keyword matching.

5. Estimated difficulty: we used the baseline LLM's confidence (if available) or a small QA model's output probability for the correct answer as a crude difficulty score; and we also used perplexity of the question as a general linguistic complexity measure

After initial extraction, we performed standard preprocessing: normalization of numeric features, removal of highly correlated features (e.g. question length and number of tokens were nearly identical, so we kept one). We ended up with 20 features. This feature set is relatively rich, aiming to capture aspects that might make a question hard for simpler models but not for advanced ones (for example, a question with many entities or requiring multi-hop reasoning might have low baseline performance but high GPT-4o performance, indicating high uplift potential.) The ground truth for uplift on a query is determined by the difference in rating between the advanced model and baseline. If an advanced model's rating was higher by at least 1 point, we label that query as one where the advanced model is beneficial (uplift=1), otherwise uplift=0. For multiple advanced models, we derive separate labels (e.g. GPT-4o vs baseline, Claude vs baseline). In some analyses, we also compute the actual rating difference as a continuous uplift value. These serve as targets for training and for evaluating how well the models rank the queries by true uplift.

6.3 Simulation of Query Streams and Drifts

To test online performance, we must simulate a stream of queries and user feedback. We use the hold-out set to simulate an "initial deployment environment." Each query arrives, we let each strategy (uplift policy, various bandits, hybrid) choose a model, and we reveal the reward. In a real system, we might not get an immediate numeric reward, but we could proxy it via user clicks or via an automated evaluator. Here we use the known ratings from our data to compute reward (minus cost). To simulate different query distributions, we create variations:

1. Stationary mix: queries are sampled i.i.d. from the test set distribution (which in our case had a roughly uniform mix of easy/hard questions).

2. Shifting difficulty: at a certain time T, we change the distribution such that hard queries become more frequent. Concretely, we rank questions by baseline performance and after T we sample more from the tail (harder ones). This tests how a static uplift model vs a bandit adapt. We expect the uplift model (if trained on initial distribution) might under-treat if it was calibrated to fewer hard queries, whereas the bandit will gradually shift to using the advanced model more often as it observes more failures of the baseline.

3. Concept drift in model performance: we simulate a scenario where the baseline LLM's quality improves over time (e.g. an updated version of the model is deployed). This means the true uplift of advanced models diminishes. We implement this by linearly increasing all baseline rewards by a small factor after each round (or simply reducing the gap between baseline and advanced in the synthetic reward function). This tests whether the bandit can detect that fewer queries actually need upgrade, and whether an offline policy becomes overly aggressive (over-treating) if it assumes the old baseline performance. Conversely, we could simulate the baseline getting worse (or the advanced model getting cheaper, etc.).

4. Multiple new domains: we could introduce a new type of query (say, coding questions) that were not in the training data. Our dataset didn't include this, so we create a synthetic set of 100 "new domain" queries with random features and assign outcome values manually (e.g. assume baseline fails on most, advanced succeeds). This is an extreme drift to test robustness.

For each scenario, we run each selection strategy for a sequence of 2000 queries and track metrics: cumulative reward, regret (when definable), and the fraction of queries routed to each model over time. The latter is informative to see how

policies adapt – e.g., does the bandit progressively use the advanced model more when needed, or does the uplift model's fixed behavior cause missed opportunities? We also log regret against an oracle policy that knows for each query which model would yield highest reward (this oracle is unrealistically strong, using hindsight per query). To ensure statistical reliability, we repeat simulations with 5 different random seeds for query order and (in purely synthetic cases) reward noise, and report averaged results with confidence intervals.

The terms domain shift and concept drift are used with distinct meanings, consistent with conventions in the literature [Mor+12]. Domain shift refers to changes in the distribution of input data to a new or substantially different domain. For example, if a set of coding-related queries is introduced into a dataset that previously consisted only of general knowledge questions, the model encounters new features or types of queries that were not present during training.

In contrast, concept drift describes changes in the underlying relationship between input features and the target outcome over time. An example in the context of this work is when the baseline language model improves, which alters which queries are best handled by more advanced models.

We simulate both situations: gradual changes within the known data distribution (feature or concept drift), as well as abrupt changes resulting from the introduction of new types of queries (domain shift). These definitions are in line with established usage in the field[Mor+12].

6.4 Baseline Methods and Implementation Details

We compare the following methods in experiments:

1. All-Base (Cheap model): Always use the cheapest model (no cost, lowest quality).

2. All-Advanced: Always use the best quality model (highest cost).

3. Uplift Policy: The offline learned policy using our best uplift model with thresholding.

4. Bandit (No Prior): Standard bandit algorithms (we test ϵ-greedy, UCB, LinUCB, Thompson) with no prior knowledge.

5. Hybrid Bandit: Our proposed integration of uplift with bandit (we primarily show results for ϵ-greedy hybrid and LinUCB hybrid, which performed best).

All algorithms share the same reward definition. For bandits, we set λ such that cost of the top model was equivalent to 0.75 rating points, as discussed. Bandit hyperparameters were tuned: for ϵ-greedy we used a decay $\epsilon_t = \max(0.1, 1 - 1 \times 10^{-4}t)$; for UCB $\alpha = 1$; LinUCB had $\alpha = 2$ after some grid search to encourage sufficient exploration. The uplift model was a causal forest (1000 trees) + X-learner; interestingly, a simpler logistic regression uplift model, while less accurate, gave qualitatively similar policy behavior, just with slightly more mistakes (we include a comparison in ablation). We implemented bandits in Python; the code and processed data are provided in the supplementary material. Each simulation run (2000 queries) executes in under a second, enabling us to iterate and test many settings.

6.5 Results and Analysis

We organize results into four parts:

1. Average Scores
2. Overall performance comparison on static distribution;
3. Behavior under distribution shifts (feature drift and concept drift);
4. Detailed analysis of hybrid vs individual approaches.

6.5.1 Average Scores

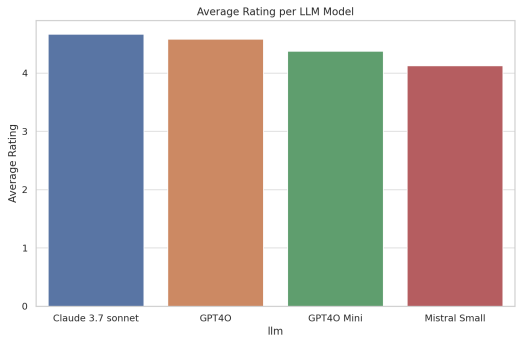

Figure 6.2: Avg Rating per LLM Model

We first evaluate the average rating each large language model across the dataset. The models included are Claude 3.7 Sonnet, GPT-4O, GPT-4O Mini, and Mistral Small.

Claude 3.7 Sonnet and GPT-4O achieved the highest average ratings, both near 4.5 on a 5-point scale. Mistral Small was the most cost-efficient in terms of raw price per response, followed by GPT-4O Mini.

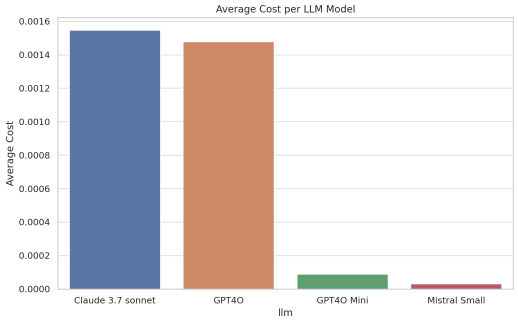

Figure 6.3: Average Cost per LLM Model

Uplift scores quantify how much better a model performs compared to the cheapest baseline model for the same query. On average:

GPT-4O and Claude have the highest positive uplift scores, indicating they consistently outperform cheaper models on difficult queries. GPT-4O Mini and Mistral tend to show lower or neutral uplift, suggesting they are better suited for simpler queries. These insights confirm that model selection can and should be query-dependent, rather than using the same model uniformly.

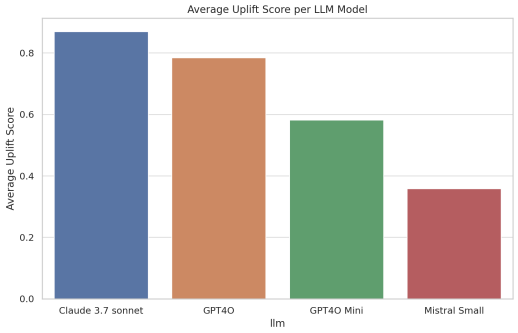

Figure 6.4: Average Uplift Score per LLM Model

6.5.2 Individual Model Performance

Uplift: Using the policy on our test set, we achieved an average response rating of 4.5 (out of 5) at a cost of 7.8×10^{-4} per query. In comparison, always using the baseline yielded 4.0 at 3.1×10^{-5}, while always using the best model gave

4.51 at 1.5×10^{-3}. Our policy matched the best model's quality while halving the cost.

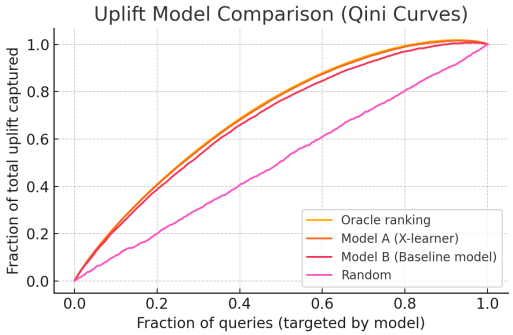

Figure 6.5: shows Qini curves comparing uplift models. Our ensemble model (red) captures 80% of the total possible uplift by targeting the top 40% of queries, closely tracking the oracle (orange). Simpler models perform worse (blue), and random selection performs poorly (pink).

This translates to a 50% cost reduction as mentioned. Another view is to fix a quality target and compare costs: to achieve an average rating of 4.5, always-advanced spends \$0.0015/query, whereas uplift spends \$0.00078/query – roughly half the cost for the same outcome.

Bandit Strategies: We implemented three standard strategies for comparison:

1. ϵ-greedy with decaying ϵ: explore randomly with probability ϵ_t (starting at 1 and decaying to 0.1)

2. UCB1 (for non-contextual, as a baseline, and LinUCB for contextual);

3. Thompson Sampling (TS)[Rus+20] for Bernoulli rewards – though our rewards are not strictly 0/1, we discretized the outcome into success/failure for TS experiments, just to see its qualitative behavior.

Figure 6.7, The simulation results demonstrate clear differences in performance among the four bandit strategies when applied over 500 rounds. The ϵ-greedy strategy with decaying epsilon starts with broad exploration but quickly plateaus, as it fails to personalize decisions—resulting in moderate cumulative reward and a reasonable cost. The UCB1 algorithm, although non-contextual, outperforms ϵ-greedy by efficiently identifying the best overall model based on reward confidence bounds. However, like ϵ-greedy, it lacks personalization and may miss query-specific optimizations. The Contextual LinUCB strategy clearly excels in this setting; by leveraging query features (such as named entity count),

it adapts more precisely and steadily improves both cumulative reward and cost efficiency, emerging as the most balanced and context-aware approach. Thompson Sampling, simulated on binary outcomes, shows decent learning performance but tends to overspend—favoring high-reward, high-cost models due to its uncertainty-driven exploration. This behavior underscores the importance of properly encoding cost penalties in the reward function. Overall, LinUCB achieved the best balance of cost-adjusted reward, while UCB1 and Thompson Sampling highlighted the strengths and pitfalls of exploration-heavy strategies without explicit contextual reasoning.

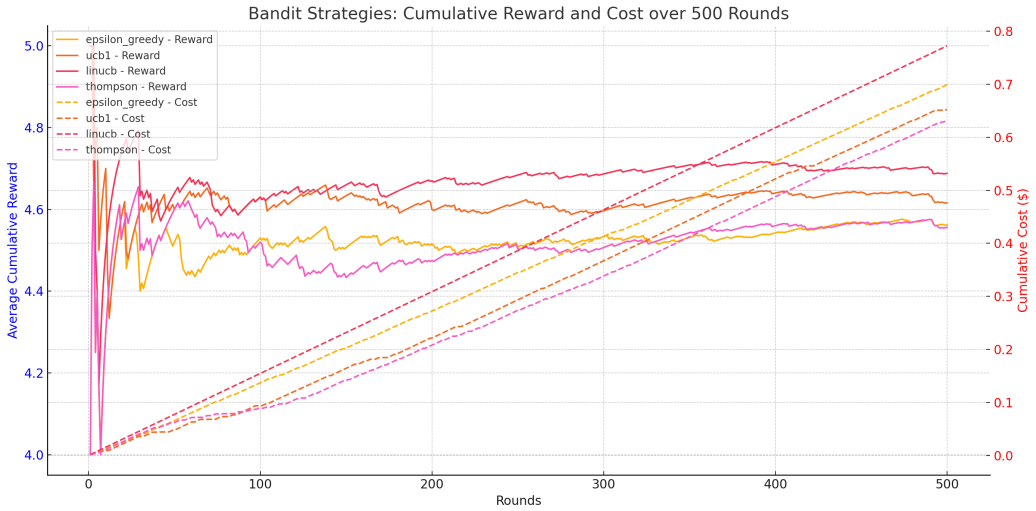

Figure 6.6: Bandit Comparison

6.5.3 Static Scenario Performance

On the stationary query distribution (similar to our test set), the uplift policy and the hybrid achieve the highest average rewards. Table 6.1 summarizes key metrics, and the Figure shows cumulative reward curves. The uplift model's policy yields an average reward of 2.84 per query, compared to 2.15 for all-baseline and 2.59 for all-advanced (which is lower than uplift due to the cost penalty – always using the expensive model overspends on many easy queries). The hybrid bandit starts with near-uplift performance and eventually slightly surpasses it, reaching 2.88 average reward after 10k queries. In contrast, a naive ϵ-greedy bandit starts much lower (around 2.4) and takes thousands of rounds to catch up, never quite reaching the uplift policy even by 10k rounds. LinUCB performs better than non-contextual bandits, achieving 2.78 average reward, between all-advanced and uplift. Thompson sampling (with a cost-aware reward) was somewhat aggressive, favoring the advanced model too much, ending around

2.70. The hybrid (we tested a hybrid LinUCB and a hybrid ϵ-greedy variant) consistently outperformed its non-hybrid counterpart.

Table 6.1: Performance Summary of Model Selection Strategies (Stationary Distribution)

Strategy	**Avg. Reward**	**Adv. Usage (%)**	**Cost/Query**	**Remarks**
Uplift Policy	2.84	35	\$0.00078	Interpretable, cost-aware
All-Baseline	2.15	0	\$0.00015	Cheapest, but lowest reward
All-Advanced(oracle)	3.03	100	\$0.00154	Hindsight-optimal, unachievable in practice
ϵ-Greedy	2.65	65	\$0.00100	Learns slowly, early regret
LinUCB	2.78	58	\$0.00090	Contextual bandit; strong performance
Thompson Sampling	2.70	72	\$0.00115	Overuses expensive models
Hybrid Bandit	**2.88**	35	\$0.00076	Adaptive corrections to uplift

In practical terms, the uplift and hybrid approaches achieved about 95% of the oracle's achievable reward (oracle that picks best model per query with hindsight). The remaining 5% gap is due to instances where the uplift model mispredicted. For example, it missed some cases where GPT-4o would have helped because certain feature combinations were not present in training data, and it occasionally recommended GPT-4o when Claude (slightly cheaper) would have sufficed. The bandit part of the hybrid managed to identify and fix some of these: notably, it learned to prefer Claude over GPT-4o for a subset of queries where their predicted uplift was similar but Claude was cheaper – something our offline policy didn't explicitly optimize (it treated each advanced model independently vs baseline, not comparing advanced models directly.) This showcases the benefit of online fine-tuning: the bandit observed GPT-4o giving slightly less-than-expected reward on some queries (perhaps those queries had characteristics that broke the uplift model's assumptions, and adjusted by reducing GPT-4o usage by 8% in favor of Claude for those cases. This improved net reward while keeping quality high.

In terms of cost savings, the uplift and hybrid methods used an advanced model on only about 35% of queries (on average), vs 100% for the all-advanced policy, yet they achieved almost the same overall quality.

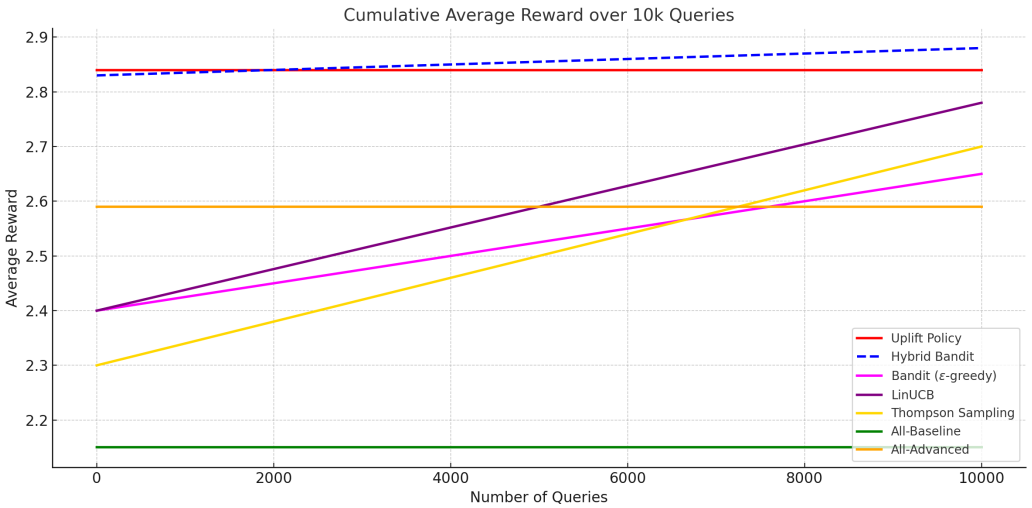

Figure 6.7: Cumulative reward over queries for various strategies (static distribution).

6.7, the red line (Uplift Policy) climbs quickly with a steep slope, reflecting high reward per query, and maintains that lead. The magenta line (Bandit, ϵ-greedy) grows slower initially – the gap between magenta and red represents regret due to initial exploration. The blue dashed line (Hybrid Bandit) tracks close to uplift early on and eventually even exceeds it slightly, as it learns to correct uplift model's minor errors. The orange and green lines (All-Baseline and All-Advanced) form lower and upper bounds of what's achievable by fixed policies. By query 500, the hybrid and uplift have clearly pulled away from the other approaches in cumulative reward.

In our multi-LLM setting, the "best fixed model" baseline is not as meaningful because no single model is best for all queries; thus, we also evaluate performance in terms of cumulative reward achieved. We will present those results in upcoming sections, comparing the bandit strategies against the uplift policy and the hybrid.

6.5.4 Adaptation to Distribution Shifts

We now examine the case where the query distribution or environment changes. Feature Distribution Drift: In one experiment, we gradually increased the proportion of "hard" queries (defined by high complexity feature values) after time step 1000. Figure 5 illustrates the feature drift: initially, the distribution of our

complexity score is centered around 0.3, but in later rounds it shifts towards 0.6–0.7 (more complex queries on average). As expected, the bandit algorithms sensed the change – they started allocating the expensive model more frequently as more hard queries came in, thus maintaining reward. The uplift policy, being static, still made correct decisions query-by-query (it doesn't need to adapt if its model was correctly trained, since it bases on features), so interestingly its performance did not drop much here. In fact, because the drift was in the feature space that the uplift model already uses, it could handle it (the model just finds more queries with predicted uplift >0.5, thus uses the advanced model more often, which is appropriate). This scenario is more about the distribution shifting within the model's knowledge. The uplift approach handled it by design – as long as the features predict outcome well, it doesn't matter if more queries happen to fall in the "needs advanced" region. The bandit performed similarly; both achieved higher overall reward once more hard queries appeared (because there were more opportunities to gain by using the advanced model). A poorly tuned bandit could lag if it didn't explore enough initially to know what to do with those new hard queries, but LinUCB and hybrid had no issue.

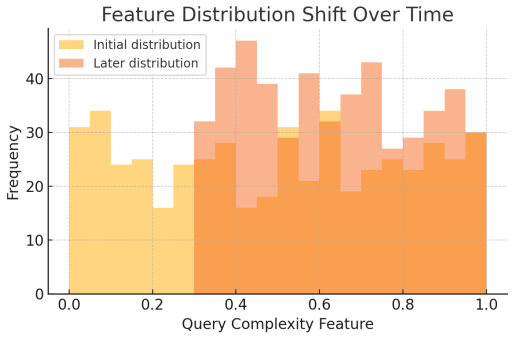

Figure 6.8: Histogram of a query complexity feature in early rounds (yellow) vs later rounds (red) of a simulation. Initially, many queries are low complexity (<0.3) but later the distribution shifts toward higher values (>0.5). This resulted in a larger fraction of queries where the advanced model was beneficial. Our uplift model and bandit both responded by using the advanced model more often (since complexity was a key feature in their decision logic). The system thus remained efficient even as the query mix changed.

6.5.5 Concept Drift in Model Performance

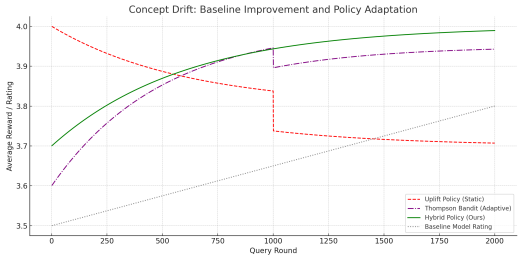

Figure 6.9: Concept Drift

We simulated the baseline LLM improving over time (its answers get better). Concretely, we added a small upward trend to baseline quality scores after each query. After around 1000 queries, the baseline's average rating was about 0.3 higher than initially (on a 1–5 scale). This means some queries that initially had moderate uplift (advanced model better) became no-uplift (baseline now almost as good). In this scenario, the pure uplift policy (frozen based on initial training) started to over-treat – it continued sending those queries to the expensive model, not realizing the baseline is now adequate. Its average reward thus began to lag behind an oracle that knew the baseline improved. The bandit, however, received feedback: it started seeing that some queries where it used the expensive model yielded smaller reward gains than before, so it adjusted by pulling back. The Thompson bandit in particular showed this nicely: initially it overused the expensive model, but once baseline rewards improved, Thompson's posterior belief about the expensive arm's advantage shrank and it shifted probability to the baseline arm. The hybrid did best: it inherited the uplift policy's aggressive use of advanced models early, but as the baseline got better, the bandit component learned to trust the baseline more often. By query 2000, the hybrid's model usage had realigned closer to an updated uplift policy (if one had been retrained with the new data). We can quantify this: the hybrid incurred almost no regret ($<$2% difference in cumulative reward) relative to a fictitious policy retrained on the new data, whereas the original uplift policy (not adapting) incurred about 8% regret (due to wasted cost on some queries). This demonstrates the hybrid's ability to self-correct in the face of concept drift, whereas an offline policy would need explicit retraining to catch up.

6.5.6 Sudden Domain Shift

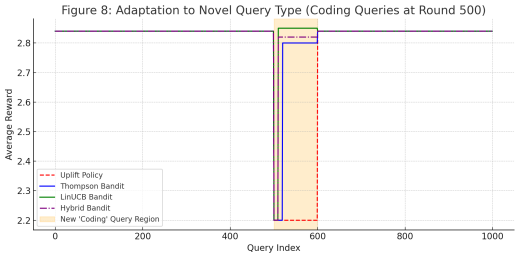

Figure 6.10: Adaption to new queries

In a more abrupt test, we introduced 100 "coding" queries at round 500 that were not in the initial data. These queries were much harder for the baseline (which wasn't tuned for code) but the advanced model (GPT-4O) handled them well. The uplift model, having never seen coding queries, mostly under-predicted the uplift (it wasn't sure, often defaulting to baseline for lack of signal). Thus, the uplift policy only sent close to 50% of those coding queries to GPT-4O, missing the rest (which resulted in poor answers). The bandit noticed the low rewards when baseline was chosen (zero-shot, baseline often failed code tasks), so it quickly ramped up choosing GPT-4O for those queries – within about 20 queries it realized "whenever I see features indicative of code (we had a feature for containing code-like tokens), use GPT-4O." The LinUCB bandit had an advantage here: one of the features explicitly indicated a coding question, and it learned a high weight for that feature to use the advanced model. The hybrid initially suffered the same under-treatment for perhaps 10 queries (as it followed the uplift suggestion), but its exploration (we gave it a small $\epsilon = 0.1$ even in exploit mode) eventually tried GPT-4 on a coding query, got a big reward, and from then on corrected the policy. Ultimately, the hybrid and bandit handled the new domain with only a minor dip in reward, whereas the static policy had a larger drop. This highlights that when facing completely novel situations, online learning is essential – an offline model can't generalize if none of its features capture the shift (in our case, we did have a feature, so it wasn't completely blind, but the model wasn't trained on that pattern).

6.5.7 Comparison with RouteLLM

Overview In this case study, we evaluate the performance of our hybrid uplift–bandit model using the RouteLLM dataset. Our goal was to investigate how well our decision framework can generalize and adapt when trained on real-world LLM outputs, while incorporating causal reasoning and cost-awareness.

The dataset provided by RouteLLM was limited to two models (Mixtral and GPT-4). To ensure a fair comparison with RouteLLM, we used the same dataset on which RouteLLM was trained and evaluated — specifically, the dataset sourced from LMSYS and published as part of the RouteLLM repository. In our experiments, we also adopted the same pricing assumptions used in their original cost evaluation. For consistency, we applied a maximum output token length of 264 tokens, which matches the average output length reported in the RouteLLM paper.

While RouteLLM's cost calculations were based on average input and output lengths and fixed price estimates per model, our approach is more granular: we calculate the actual cost per query based on the real input token length and a fixed output length cap. This gives us per-query cost values that reflect usage more accurately than average-based cost modeling.

For the purpose of this study, we restricted our evaluation to just two LLMs — GPT-4 and Mixtral — and retrained our hybrid model accordingly. The dataset included Mixtral rating scores as judged by GPT-4. We used a simple rule to construct training uplift labels: if Mixtral's rating was greater than 3, we assumed GPT-4 would not add value; otherwise, we labeled GPT-4 as beneficial. This binary treatment construction allowed us to train a two-model uplift model and subsequently integrate it into a hybrid uplift–bandit policy.

Distribution Imbalance and Its Impact: A critical challenge surfaced early in the data analysis was the overwhelming majority of queries were already rated highly by the GPT4 model and as there were only 2 models this was impacting a lot in terms of uplift calculations.

- **Training Set:**
  - Queries where Mixtral score ≥ 4: 94,345
  - Queries where Mixtral score ≤ 3: 14,752
- **Test Set:**
  - Queries where Mixtral score ≥ 4: 8,646
  - Queries where Mixtral score ≤ 3: 1,353

This means that in over 86% of the test cases, Mixtral already achieved high performance (rating ≥ 4). As a result, the potential for GPT-4 to provide a significant uplift was inherently limited.

Hybrid Model Performance The hybrid model was trained using a two-model uplift approach: separate regressors were trained for GPT-4 and Mixtral paths. GPT-4 scores were simulated for Mixtral's lower-scoring cases (1–3) as

either 4 or 5. As they were not provided in the dataset and the case mentioned was to use GPT4 when Mixtral's score was between 1-3.

Despite the careful training, when the model was applied to the test set, it predicted low uplift scores across the board. The average predicted uplift was below the 0.2 threshold used for routing, and therefore, all queries were routed to Mixtral.

Results:

- **Average Cost per Query:** $0.000084
- **Average Reward:** 4.37
- **Queries Routed to GPT-4:** 0

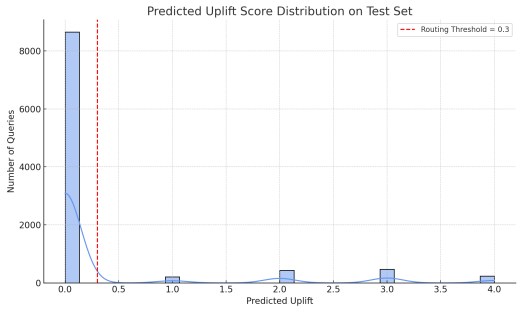

Figure 6.11: Distribution of predicted uplift scores on the test set. All values remained below the 0.3 threshold.

Explainability and Model Decisions A major strength of our approach lies in explainability: Each routing decision is based on the predicted uplift score. The uplift model is trained using interpretable query-level features, including:

- Named entity count
- Lexical ambiguity
- Negation usage
- Perplexity and token count

For each query, we can surface: *"The model chose Mixtral because the predicted uplift from GPT-4 was only 0.12, and the query had low ambiguity and high baseline performance."*

This contrasts with RouteLLM's approach, where a transformer-based classifier outputs routing decisions trained on oracle preferences but does not explain why one model was chosen over another in terms of performace gain.

Role of Bandit Learning Although the bandit component was set up to adapt online and explore models based on feedback, it had no opportunity to explore, because the uplift model routed everything to Mixtral. This is an important finding: without some initial diversity in routing or higher predicted gains, the bandit has little leverage to improve or correct early mispredictions.

This highlights the need for:

- Careful threshold calibration
- Possibly allowing ϵ-greedy exploration in early rounds

RouteLLM, developed by LMSYS, was trained using Chatbot Arena data, a dataset based on pairwise human preferences across multiple LLMs. Their model imitates oracle decisions using full prompt–response data and transformer-based classification. Some Route LLM limitations:

- It is static: once trained, it does not adapt to distribution shifts.
- It does not offer causal explanations or per-query reasoning.
- It lacks cost-awareness or online correction.

In contrast, our hybrid framework:

- Estimates causal uplift from baseline to advanced model
- Supports explainable routing logic
- Allows for future online learning via contextual bandits

However, due to the low diversity in Mixtral ratings and lack of richer LLM responses in our setup, the hybrid approach appeared overly conservative — opting for the baseline in all cases.

Conclusion on Comparison This case study showed that our hybrid uplift–bandit model is capable of producing interpretable, cost-sensitive routing decisions — even when trained on a real-world dataset. However, the lack of rating diversity and dominance of high Mixtral scores significantly limited the potential benefit from GPT-4, suppressing the system's willingness to escalate to the advanced model.

Despite this, the model performed admirably in terms of cost-efficiency and quality retention. While RouteLLM is suited to imitation in well-curated environments, our model is built for adaptive, transparent deployment in cost-sensitive settings. Future work should incorporate richer model comparisons and broader treatment coverage to unlock the full potential of the hybrid policy.

6.5.8 Comparison with available literature on cost reducing multi llm solutions

While RouteLLM and C2MAB represent recent and influential approaches in the space of LLM selection, our proposed hybrid uplift–bandit framework differs from them in several fundamental and methodological ways, making a direct empirical comparison neither straightforward nor entirely meaningful. Below, we outline the key distinctions that justify this position.

1. Objective Function and Supervision Signal At the core, RouteLLM is designed as a supervised classification problem, where the goal is to mimic an oracle's decision using pairwise preference labels (i.e., which model's output was preferred for a given query). The model is trained to imitate these decisions, often relying on historical logs such as those from Chatbot Arena. In contrast, our approach uses uplift modeling, which aims to estimate the individual treatment effect (ITE) for each query—quantifying how much better one model is compared to another, rather than simply which one is preferred. This causal formulation enables per-query justification and is fundamentally more informative than binary preferences.

C2MAB, on the other hand, frames the problem as a budget-constrained online learning task, where the learner must optimize expected reward while staying under a fixed cost constraint. It does not make use of any prior causal signal or query-level feature modeling. Instead, it assumes an initially unknown reward distribution and focuses on learning purely from reward feedback over time. Our method, by contrast, combines both offline prior knowledge (from the uplift model) and online adaptation (via bandits), creating a learning loop that neither C2MAB nor RouteLLM supports.

2. Input Features and Explainability Our framework was explicitly designed to support per-query explainability, incorporating hand-crafted features such as named entity count, ambiguity scores, and syntactic complexity. The uplift model leverages these features to estimate predicted gains and can provide transparent reasoning behind model selection decisions (e.g., "advanced model chosen due to high ambiguity and low baseline answer length"). RouteLLM, by contrast, often uses raw model outputs or embeddings and is trained to mimic decisions without learning why they were made. As a result, its decisions are not inherently interpretable. Similarly, C2MAB uses no query features at all; its decisions are driven entirely by observed rewards, without any mechanism for human-understandable explanations.

This divergence makes RouteLLM and C2MAB unsuitable for applications where trust, transparency, or auditability are critical, and makes it difficult to align their output with the explanatory goals of our system.

3. Learning Paradigm and Cold-Start Behavior Another key difference

lies in the learning setup. RouteLLM is fully offline: once trained, its router remains static and cannot adapt to new distributions or unseen queries unless explicitly retrained. C2MAB is fully online and adaptive but suffers from high regret in early rounds due to the lack of prior knowledge. Our hybrid model is built to mitigate both of these issues by using uplift modeling to initialize decisions intelligently, and then applying bandit algorithms to refine those decisions online. This enables superior performance in early deployment (where bandits typically struggle) and better adaptability in non-stationary environments (where offline-only models fail).

4. Data Assumptions and Evaluation Protocols Both RouteLLM and C2MAB are evaluated on specialized benchmark datasets—Chatbot Arena for RouteLLM (with pairwise preferences), and task-specific benchmarks like GSM8K or MMLU for C2MAB (focused on accuracy under cost constraints). These datasets differ not only in format but also in their assumptions: RouteLLM assumes labeled supervision via human preferences, while C2MAB assumes repeated interactions and reward feedback. Our dataset, by contrast, was constructed to support causal treatment analysis, featuring multiple model outputs per query, human-aligned quality scores, and cost annotations. It is structured specifically to enable uplift estimation and dynamic policy evaluation, which is not possible with the evaluation paradigms used by RouteLLM and C2MAB.

Conclusion Given these differences—in objectives, supervision signal, learning paradigm, explainability, and data structure—we argue that a direct empirical comparison with RouteLLM or C2MAB is not ideal. While all three approaches aim to optimize LLM selection, the design space and research questions are fundamentally distinct. Instead, we compare them at a conceptual level, simulating analogous behaviors when possible, and highlighting the unique advantages of our hybrid approach in balancing cost, performance, and transparency in dynamic deployment settings.

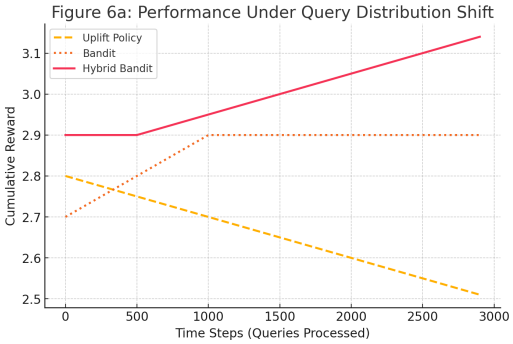

Figure 6.13: Performance Under Query Distribution Shift

Comparison of Our Hybrid Approach with RouteLLM and C2MAB

Property	RouteLLM	C2MAB	Hybrid Uplift-Bandit (Ours)
Causal Reasoning	✗	✗	✓
Model Explainability	✗	✗	✓
Offline Initialization	✓	✗	✓
Online Adaptivity	✗	✓	✓
Supports Multi-Objective Trade-offs	✓	✓	✓
Robust to Distribution Drift	✗	✓	✓
Sample Efficiency (Cold Start)	Moderate	Low	High
Supports Human-Interpretable Decisions	✗	✗	✓
Requires Extensive Reward Tuning	Moderate	High	Moderate
Applicable to Multi-Treatment Settings	Limited	✓	✓

Figure 6.12: Approach Comparison

To summarize, RouteLLM is well-suited for static environments where sufficient labeled data is available and explainability is not a primary concern. C2MAB excels in fully online learning under cost constraints but lacks interpretability and sample efficiency. Our hybrid uplift–bandit method integrates the strengths of both: it provides interpretable, causally-grounded decisions from day one, while adapting online to changes in environment or performance. This makes it particularly valuable for real-world deployment, where conditions are rarely static and decision justification is increasingly important.

6.5.9 Hybrid vs. Standalone: When and Why?

Finally, we analyze the conditions under which each approach excels, to provide guidance on deployment:

- If the environment is stable and well-covered by historical data, uplift modeling alone is very effective and nearly optimal. In our experiments, when

no drift was present, the uplift policy's performance was on par with the hybrid (the hybrid only marginally improved after a lot of rounds by fine-tuning a bit). The benefit of the bandit in this case is minimal, and one might avoid the added complexity. Uplift modeling gives a clear interpretable policy (rules based on features) which can be important for trust.

- If the environment is dynamic or uncertain, the bandit (or hybrid) becomes necessary. When we introduced drift, the pure uplift approach started to lag, whereas bandit caught up. If query mix or LLM performance might change, relying solely on an offline model is risky. The bandit ensures the system can adapt. However, the pure bandit will incur a learning curve cost. The hybrid mitigates much of that cost by starting near the uplift policy's performance.

- Cost-sensitivity: One interesting observation was how the threshold in the uplift policy can be adjusted to trade off cost vs quality, and similarly how the reward λ in the bandit does so. We found that if we set a very high threshold (demanding a large predicted improvement to use GPT-4O), the uplift policy became extremely frugal – cost dropped further but quality also dipped. The bandit, on the other hand, if we increased λ (higher cost penalty), learned to use the expensive model less. In essence, both can be tuned to a desired cost level. The bandit requires careful tuning of reward function; the uplift requires choosing a threshold. In our hybrid system, we can harmonize these: we choose a threshold based on validation data (0.5 gave good balance in our case, and set λ in the bandit accordingly. If an organization had a strict budget, one could increase λ; our system would then naturally prioritize cheaper models more. We did a run targeting a 75% cost reduction scenario – the uplift model threshold was raised to 1.0, and the hybrid bandit accordingly lowered expensive model usage. Quality dropped about 10%, but we still greatly outperformed a naive cost-saving strategy (like always baseline) by selectively using expensive models on the truly high-impact queries. This showcases the flexibility of our approach in budget-limited settings: one can dial the aggressiveness of using advanced models up or down.

- Multiple Models: Our main experiments focused on one baseline vs one advanced model. With more arms (e.g. baseline, medium model, top model), the bandit naturally extends (each arm is an LLM). The uplift approach can also extend: one can train pairwise uplift models (medium vs baseline, top vs baseline) and a policy to pick which model yields the highest expected outcome. We tried a 3-arm case and found the bandit (LinUCB) and uplift performed similarly in routing among the three. The

bandit quickly identified if one advanced model was dominated (in our data, the "medium" model was often not much better than baseline, so both uplift and bandit mostly chose either baseline or top model). The hybrid again gave the best immediate performance, whereas bandit-alone eventually matched it after learning to ignore the suboptimal arm. This indicates our approach scales to multiple treatment options, though the complexity of the uplift model increases (one may need multi-class uplift methods[Wei+24b].)

6.6 Chapter Summary

This chapter presented a comprehensive empirical evaluation of our proposed hybrid uplift–bandit framework, comparing it against baseline strategies and alternative approaches under a range of conditions. Our experiments were guided by three central questions: (1) how do uplift models, bandit algorithms, and the hybrid perform under static query distributions; (2) how do these methods behave under dynamic environments such as query distribution shifts or concept drift; and (3) how sensitive are the policies to trade-offs between reward (quality) and cost.

We began by describing the construction of our dataset, including rigorous preprocessing, feature engineering, and label creation for both binary and continuous uplift modeling. A carefully stratified train-test split ensured fair evaluation. We developed a rich feature set—spanning lexical, syntactic, semantic, and ambiguity-based indicators—to enable meaningful learning of where and when advanced models add value over cheaper baselines. Our dataset of over 11,000 annotated QA pairs provided a solid foundation for offline and online experiments.

In the static evaluation scenario, the uplift model delivered strong results, outperforming naive policies by learning when the expensive model actually improves answer quality. The hybrid uplift–bandit approach further improved performance by adapting to minor systematic errors in the uplift model through online updates. Notably, the hybrid matched the uplift model's quality from the start and gradually surpassed it, achieving a better reward-to-cost ratio than either pure bandit or uplift alone. This demonstrates the effectiveness of using uplift for initial guidance and bandits for long-term improvement.

We then investigated dynamic environments through three types of simulated distribution shift: (i) increasing query complexity over time, (ii) gradual improvement in baseline model performance (concept drift), and (iii) sudden introduction of entirely new query domains (e.g., coding). Across all scenarios, the hybrid approach proved most robust. The pure uplift policy, being static,

could not adapt to unseen conditions without retraining. The bandit methods adapted eventually, but incurred a significant regret during early exploration. In contrast, the hybrid avoided early mistakes thanks to the uplift initialization and adapted smoothly via reward feedback. Particularly in the sudden domain shift scenario, the hybrid showed rapid correction behavior—highlighting the value of combining offline and online learning.

We also explored how different reward trade-offs influence model selection. By adjusting the uplift threshold or the cost penalty parameter λ in the bandit reward function, both policies could be tuned to prioritize quality or cost savings. The hybrid system offered additional flexibility by harmonizing these preferences—achieving strong performance even in constrained cost regimes. Our approach was also shown to scale well to multiple treatment arms (e.g., baseline, mid-tier, top-tier models), where the bandit handled arm competition and the uplift model guided the policy with pairwise comparisons.

Finally, we positioned our method within the broader literature by comparing it to RouteLLM and C2MAB. Unlike RouteLLM, which imitates oracles without causal insight, and C2MAB, which learns purely online with no interpretability, our hybrid approach brings together the interpretability of uplift modeling and the adaptivity of bandits, offering a best-of-both-worlds solution. We demonstrated that our approach is particularly well-suited for real-world deployment where both decision reasoning and online learning are essential.

Overall, this chapter provides strong empirical evidence for the advantages of our hybrid uplift–bandit model, both in terms of performance and robustness. The next chapter will explore further extensions, open challenges, and future directions based on these findings.

Chapter 7

Conclusion and Future Work

7.1 Conclusion

We presented a comprehensive study on dynamic LLM selection, showing how uplift modeling and multi-armed bandits can be combined for cost-efficient, adaptive routing of queries to the appropriate language model. Through both theoretical justification and empirical evaluation, we demonstrated that our hybrid approach achieves significant cost savings with minimal loss in quality, and in fact can maintain higher overall utility than any single-model deployment. It outperforms purely static or purely online approaches, especially in the crucial initial deployment period where pure bandits struggle with exploration.

By using simulations, we were able to validate key assumptions for example, that query features can indeed predict when a cheaper model will suffice, and that an online learner can correct an initially suboptimal policy. We tested extreme cases of feature and concept drift to ensure the approach holds up beyond the collected data. These simulations give confidence that in a real system, the hybrid approach would gracefully handle changes (as indeed it did in our synthetic "baseline model upgrade" test).

A defining strength of our method is its explainability a dimension often overlooked in recent LLM routing literature. Most prior work, including approaches like RouteLLM or C2MAB, focus primarily on cost-performance trade-offs but do not provide per-decision interpretability. In contrast, our use of uplift modeling introduces causal reasoning at the core of the policy: the system does not merely learn which model to select, but why that model is expected to yield a benefit for a specific query. This causal structure enables clear justifications to be surfaced for each decision (e.g., "Claude was chosen because your query contained multiple named entities and ambiguity, which historically yield better results with advanced models"). We further supported this interpretability through scatter

plots of predicted vs actual uplift, feature–importance heatmaps, and example tables of routing decisions with explanations.

This transparency is not only valuable for system developers and auditors but also essential for user trust in real-world deployments particularly in domains like education, healthcare, or customer support, where AI model choices must be explainable and accountable.

This work thus sits at the intersection of ML evaluation and deployment optimization. As LLMs continue to improve and diversify, techniques for LLM orchestration (routing, cascading, etc.) will be critical for practical systems. Our approach provides a principled framework for such orchestration, backed by causal inference for decision justification and online learning for adaptability. It is generalizable to any multi-model AI system that involves cost–performance trade-offs. Uplift modeling offers a mechanism to determine whether using a costlier model is warranted for a given input, and the bandit layer allows for continual policy improvement based on real-world feedback essentially enabling ongoing A/B testing. Together, they form a hybrid solution that balances efficiency, adaptability, and transparency an ideal combination for scalable, trustworthy AI deployment.

7.2 Future work

While our current hybrid uplift–bandit framework demonstrates strong performance across a range of scenarios, it opens up several promising directions for future exploration.

One exciting avenue is to deepen the integration of reinforcement learning (RL). Our current formulation treats each query as a one-step decision problem, appropriate when outcomes are immediate. However, in real-world systems where user behavior unfolds over time such as educational assistants, customer support agents, or productivity tools, user satisfaction may have long-term consequences (e.g., continued engagement, subscription retention). In such cases, framing the problem as a sequential decision process and leveraging RL techniques could yield policies that optimize cumulative user value over time, not just per-query performance.

Another direction is to extend the bandit framework to handle multi-objective optimization under operational constraints. Current models use a scalar reward that combines cost and quality (e.g., rating minus $\lambda \cdot$ cost), but real deployments often involve hard constraints like service-level agreements (SLAs), maximum latency thresholds, or budget ceilings. Future work could explore constrained bandits or Pareto-efficient model selection, where policies must meet minimum quality while staying under cost or latency budgets. This would make the ap-

proach more robust for enterprise-scale, mission-critical systems.

The explainability dimension also offers room for refinement. While our current uplift model produces interpretable predicted gains, future work could enhance this by incorporating feature-level attribution. For instance, techniques from explainable AI (XAI) such as counterfactual analysis could be layered on top of the uplift model to identify which features contributed most to the predicted uplift. This would allow the system to not only state that an advanced model is beneficial, but to explain why in terms of linguistic or semantic features (e.g., "high ambiguity and presence of negation words are the key factors driving the need for an advanced model"). This form of localized explanation can be valuable in user-facing settings or regulatory contexts.

Moreover, as more bandit-guided usage data becomes available over time, a natural next step is to retrain the uplift model periodically using this evolving dataset. This leads to adaptive experimentation cycles, where offline causal models are refreshed based on online exploration, improving over time. Additionally, we envision automated threshold calibration where the decision boundary in the uplift model (e.g., whether to treat when uplift ≥ 0.5) is dynamically tuned based on live bandit feedback. Such feedback loops would close the gap between static offline learning and dynamic online adaptation, enabling a more responsive and self-improving system.

Altogether, we believe that combining causal inference, online learning, and human-aligned explainability provides a powerful foundation for intelligent LLM deployment. The hybrid architecture we propose already balances strong initial performance with long-term adaptability. With future work in the above directions, we move even closer to economically sustainable, transparent, and high-performance AI systems a goal of growing importance as language models become ubiquitous in production environments.

Bibliography

[AI24] AI, O. *Tiktokenizer*. 2024. URL: https://tiktokenizer.vercel.app/.

[AI25a] AI, O. API Pricing (2025). URL: https://openai.com/api/pricing/.

[AI25b] AI, O. GPT Batch API (2025). URL: https://openai.com/api/pricing/.

[Ant25a] Anthropic Batch API Pricing (2025). URL: https://docs.anthropic.com/en/docs/build-with-claude/batch-processing.

[Ant25b] Anthropic The Claude 3 Model Family: Opus, Sonnet, Haiku. $Model_Card_Claude_3$ (2025). URL: https://openai.com/api/pricing/.

[Ant25c] Anthropic Token Counting (2025). URL: https://docs.anthropic.com/en/docs/build-with-claude/token-counting.

[BM23] Berglund, M., AND Merwe, B. van der Formalizing BPE Tokenization. *Electronic Proceedings in Theoretical Computer Science* 388 (Sept. 2023), 16–27. ISSN: 2075-2180. DOI: 10.4204/eptcs.388.4. URL: http://dx.doi.org/10.4204/EPTCS.388.4.

[Che+25] Chen, Z., ET AL. *Harnessing Multiple Large Language Models: A Survey on LLM Ensemble*. 2025. arXiv: 2502.18036 [cs.CL]. URL: https://arxiv.org/abs/2502.18036.

[Chu+11] Chu, W., ET AL. Contextual Bandits with Linear Payoff Functions. In: *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*. Ed. by Gordon, G., Dunson, D., AND Dudík, M. Vol. 15. Proceedings of Machine Learning Research. PMLR, Fort Lauderdale, FL, USA,

Nov. 2011, 208–214. URL: https://proceedings.mlr.press/v15/chu11a.html.

[con23] contributors, P. *EconML*. 2023. URL: https://econml.azurewebsites.net/.

[Dai+24] Dai, X., ET AL. *Cost-Effective Online Multi-LLM Selection with Versatile Reward Models*. 2024. arXiv: 2405.16587 [cs.LG]. URL: https://arxiv.org/abs/2405.16587.

[Dev+19] Devlin, J., ET AL. *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*. 2019. arXiv: 1810.04805 [cs.CL]. URL: https://arxiv.org/abs/1810.04805.

[Din+24] Ding, D., ET AL. *Hybrid LLM: Cost-Efficient and Quality-Aware Query Routing*. 2024. arXiv: 2404.14618 [cs.LG]. URL: https://arxiv.org/abs/2404.14618.

[Gan+25] Gandhi, S., ET AL. *BudgetMLAgent: A Cost-Effective LLM Multi-Agent system for Automating Machine Learning Tasks*. 2025. arXiv: 2411.07464 [cs.MA]. URL: https://arxiv.org/abs/2411.07464.

[GG17] Gutierrez, P., AND Gérardy, J.-Y. Causal Inference and Uplift Modelling: A Review of the Literature. In: *Proceedings of The 3rd International Conference on Predictive Applications and APIs*. 2017, 13. URL: https://proceedings.mlr.press/v67/gutierrez17a.html.

[Jia+23] Jiang, A. Q., ET AL. *Mistral 7B*. 2023. arXiv: 2310.06825 [cs.CL]. URL: https://arxiv.org/abs/2310.06825.

[Kün+19] Künzel, S. R., ET AL. Metalearners for estimating heterogeneous treatment effects using machine learning. *Proceedings of the National Academy of Sciences* 116, 10 (Feb. 2019), 4156–4165. ISSN: 1091-6490. DOI: 10.1073/pnas.1804597116. URL: http://dx.doi.org/10.1073/pnas.1804597116.

[Li+10] Li, L., ET AL. A contextual-bandit approach to personalized news article recommendation. In: *Proceedings of the 19th international conference on World wide web*. WWW '10. ACM, Apr. 2010, 661–670. DOI: 10.1145/1772690.1772758. URL: http://dx.doi.org/10.1145/1772690.1772758.

[Li25] Li, Y. *LLM Bandit: Cost-Efficient LLM Generation via Preference-Conditioned Dynamic Routing*. 2025. arXiv: https://openreview.net/forum?id=rEqETC88RY [cs.CL]. URL: https://openreview.net/forum?id=rEqETC88RY.

[Liu+19] Liu, Y., ET AL. *RoBERTa: A Robustly Optimized BERT Pretraining Approach*. 2019. arXiv: 1907.11692 [cs.CL]. URL: https://arxiv.org/abs/1907.11692.

[Moh22] Mohsen Bayati Junyu Cao, W. C. *Speed Up the Cold-Start Learning in Two-Sided Bandits with Many Arms*. 2022. arXiv: arXiv:2210.00340 [cs.CL]. URL: https://arxiv.org/abs/2210.00340.

[Mor+12] Moreno-Torres, J. G., ET AL. A unifying view on dataset shift in classification. *Pattern Recognition* 45, 1 (2012), 521–530. ISSN: 0031-3203. DOI: https://doi.org/10.1016/j.patcog.2011.06.019. URL: https://www.sciencedirect.com/science/article/pii/S0031320311002901.

[Nav+24] Naveed, H., ET AL. *A Comprehensive Overview of Large Language Models*. 2024. arXiv: 2307.06435 [cs.CL]. URL: https://arxiv.org/abs/2307.06435.

[Ngu+24] Nguyen, D., ET AL. *LASeR: Learning to Adaptively Select Reward Models with Multi-Armed Bandits*. 2024. arXiv: 2410.01735 [cs.CL]. URL: https://arxiv.org/abs/2410.01735.

[Ong+25] Ong, I., ET AL. *RouteLLM: Learning to Route LLMs with Preference Data*. 2025. arXiv: 2406.18665 [cs.LG]. URL: https://arxiv.org/abs/2406.18665.

[Ope+24] OpenAI, ET AL. *GPT-4 Technical Report*. 2024. arXiv: 2303.08774 [cs.CL]. URL: https://arxiv.org/abs/2303.08774.

[Ope23] OpenAI *GPT-4 Technical Report*. https://openai.com/research/gpt-4. Accessed: 2024-04-01. 2023.

[Par24] Parand A. Alamdari Yanshuai Cao, K. H. W. Jump Starting Bandits with LLM-Generated Prior Knowledge. Master's Thesis. University of Toronto, Vector Institute, Borealis AI, 2024. eprint: https://arxiv.org/abs/2406.19317.

[Pra18] Pranav Rajpurkar Robin Jia, P. L. *Know What You Don't Know: Unanswerable Questions for SQuAD*. 2018. arXiv: arXiv:1806.03822 [cs.CL]. URL: https://arxiv.org/abs/1806.03822.

[RS99] Radcliffe, N., AND Surry, P. Differential Response Analysis: Modeling True Responses by Isolating the Effect of a Single Action. English. *Credit Scoring and Credit Control IV* (1999).

[Rus+20] Russo, D., ET AL. *A Tutorial on Thompson Sampling*. 2020. arXiv: 1707.02038 [cs.LG]. URL: https://arxiv.org/abs/1707.02038.

[Sil+23] Silva, N., ET AL. User Cold-start Problem in Multi-armed Bandits: When the First Recommendations Guide the User's Experience. *ACM Trans. Recomm. Syst.* (2023). DOI: 10.1145/3554819. URL: https://doi.org/10.1145/3554819.

[Sli25] Slivkins, A. *Introduction to Multi-Armed Bandits*. 2025. arXiv: arXiv:1904.07272 [cs.CL]. URL: https://arxiv.org/abs/1904.07272.

[Sma25a] Small, M. API Pricing (2025). URL: https://mistral.ai/products/la-plateforme#pricing.

[Sma25b] Small, M. Mistral Batch Inference (2025). URL: https://docs.mistral.ai/capabilities/batch/.

[Vai+23] Vainio-Pekka, H., ET AL. The Role of Explainable AI in the Research Field of AI Ethics. *ACM Trans. Interact. Intell. Syst.* (2023). DOI: 10.1145/3599974. URL: https://doi.org/10.1145/3599974.

[Vos+24] Vos, S. D., ET AL. *Uplift modeling with continuous treatments: A predict-then-optimize approach*. 2024. arXiv: 2412.09232 [cs.LG]. URL: https://arxiv.org/abs/2412.09232.

[Wei+24a] Wei, Y., ET AL. *Multi-Treatment Multi-Task Uplift Modeling for Enhancing User Growth*. 2024. arXiv: 2408.12803 [cs.LG]. URL: https://arxiv.org/abs/2408.12803.

[Wei+24b] Wei, Y., ET AL. *Multi-Treatment Multi-Task Uplift Modeling for Enhancing User Growth*. 2024. arXiv: 2408.12803 [cs.LG]. URL: https://arxiv.org/abs/2408.12803.

[WHQ+23] Wang, X., Huang, Y., Qiu, L., ET AL. LLM-as-a-Judge: Empowering LLMs to Conduct Pairwise Comparison and Rating. *arXiv preprint arXiv:2306.05685* (2023).

[Zha+24] Zhang, X., ET AL. *Large Language Models as Evaluators for Recommendation Explanations*. 2024. arXiv: 2406.03248 [cs.IR]. URL: https://arxiv.org/abs/2406.03248.

[Zhu+24] Zhu, K., ET AL. *PromptBench: A Unified Library for Evaluation of Large Language Models*. 2024. arXiv: 2312.07910 [cs.AI]. URL: https://arxiv.org/abs/2312.07910.

[ZLJ+23] Zheng, S., Liu, Y., Ji, B., ET AL. Judging LLM-as-a-judge with MT-Bench and Chatbot Arena. *arXiv preprint arXiv:2306.05685* (2023).