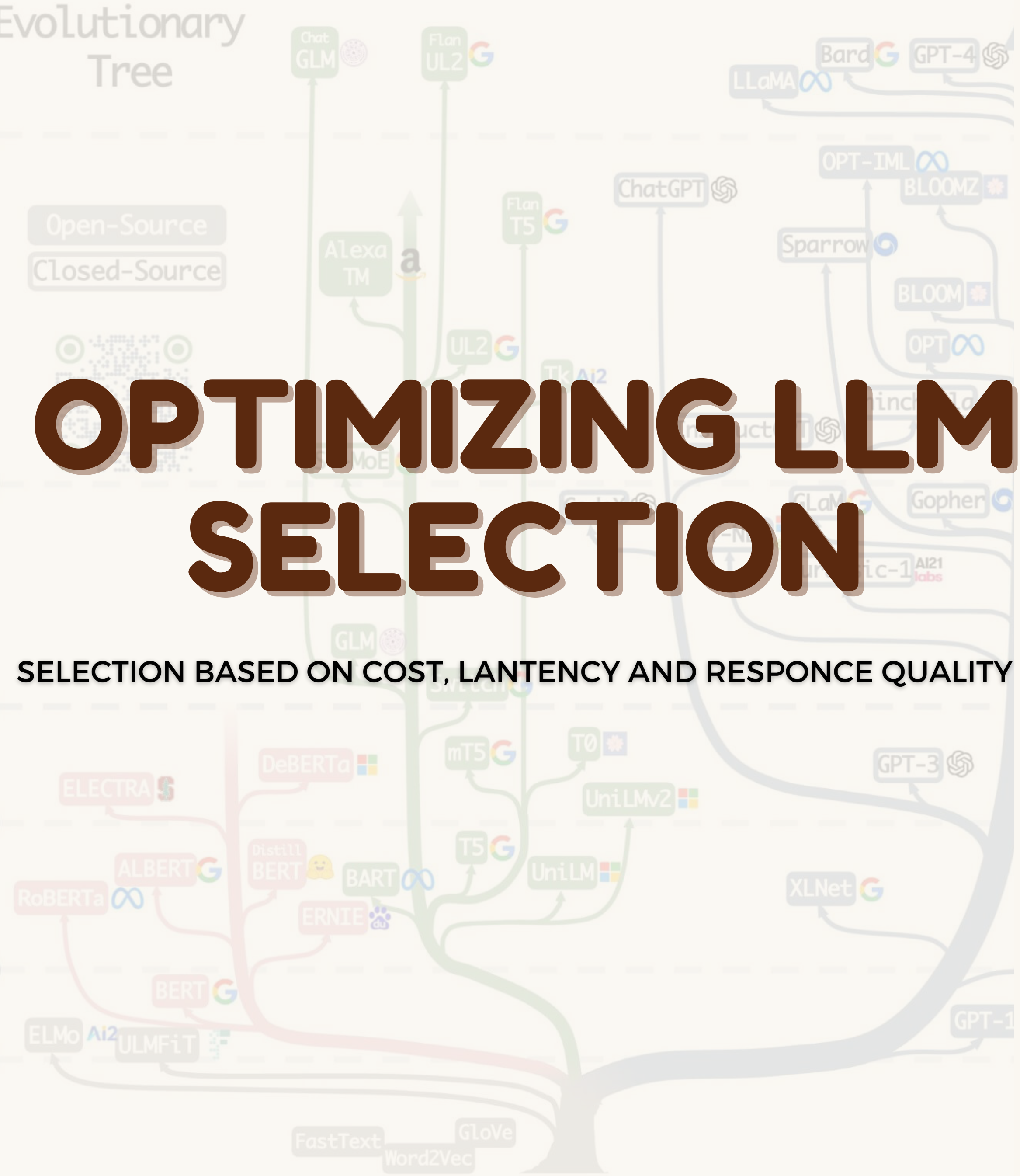


OPTIMIZING LLM SELECTION

SELECTION BASED ON COST, LANTENCY AND RESPONCE QUALITY

Bhavishya Pandit

INTRODUCTION

With the rapid growth of large language models (LLMs) across industries, choosing the right model for specific tasks has become challenging. Offer unique strengths, making evaluation on key factors essential for optimal selection.

In this guide, we address this challenge by developing a reusable codebase that enables users to systematically compare multiple LLMs.

Our approach centers on assessing cost, latency, and output effectiveness, allowing for more informed and efficient model selection. Leveraging LangChain this codebase simplifies integration and querying across different LLMs.

WHY BENCHMARKING LLM OUTPUTS

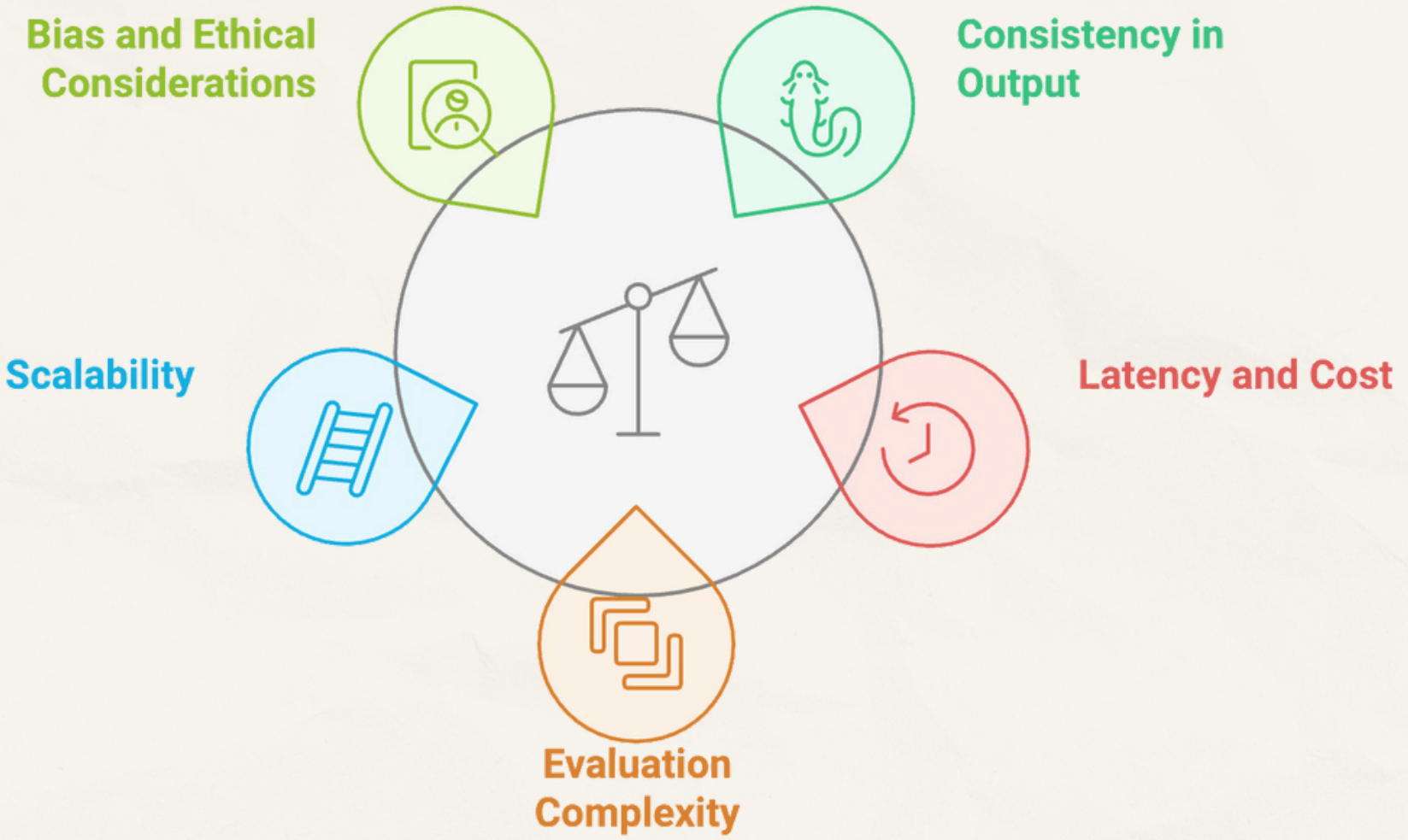
- **Task-Specific Performance**: Each LLM excels in different areas—some in creative writing, others in summarization or factual answering. Evaluating outputs by task helps in choosing the best model for each job.

- **Bias and Variability**: LLMs reflect training-data biases, potentially influencing content decisions in areas like moderation and research. Recognizing these biases is essential for fair and accurate outputs.

- **Cost Efficiency**: Some models deliver high-quality results at lower costs, making them ideal for ongoing use. Comparing cost versus quality helps optimize resource allocation.

- **Generalisation Ability**:Models may vary in adapting to new scenarios. Testing across diverse tasks reveals which are most robust for real-world applications.

CHALLENGES OF COMPARING MULTIPLE LLMS



- **Consistency in Output:** LLMs may respond variably to the same query, emphasizing different aspects, making objective evaluation challenging.

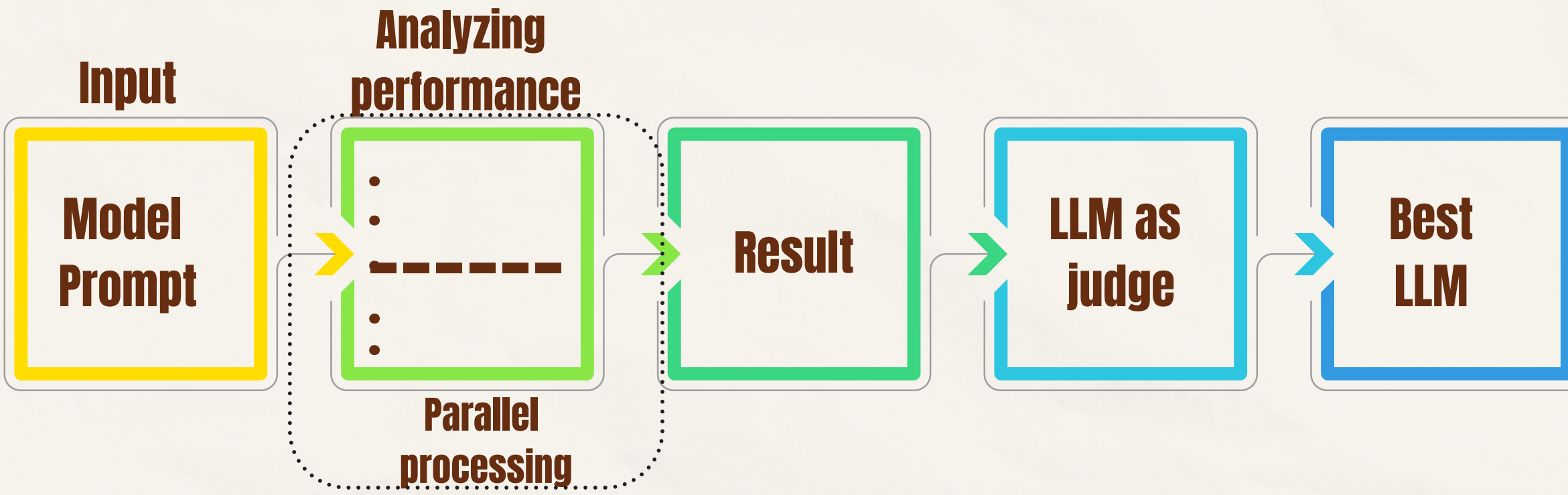
- **Latency and Cost:** Running multiple models adds to compute time and API costs, complicating large-scale testing.

- **Evaluation Complexity:** Task-appropriate metrics are essential; some require human input, others use automated metrics like BLEU or ROUGE.

- **Scalability:** Scaling infrastructure for parallel model testing with many models and tasks is resource-intensive.

- **Bias and Ethical Considerations**: LLM biases from training data can impact fairness, especially in sensitive applications, highlighting the need for careful analysis.

HOW DOES IT WORK



Input Handling: Define a function that accepts model and prompt as inputs, initiating the evaluation process for multiple language models.

Parallel Processing: Implement concurrent processing to analyze the response quality and latency of each model. Use libraries like concurrent.futures in Python to execute model evaluations simultaneously, thereby reducing overall computation time.

Result : Metrics Evaluation for each model, calculate essential metrics:

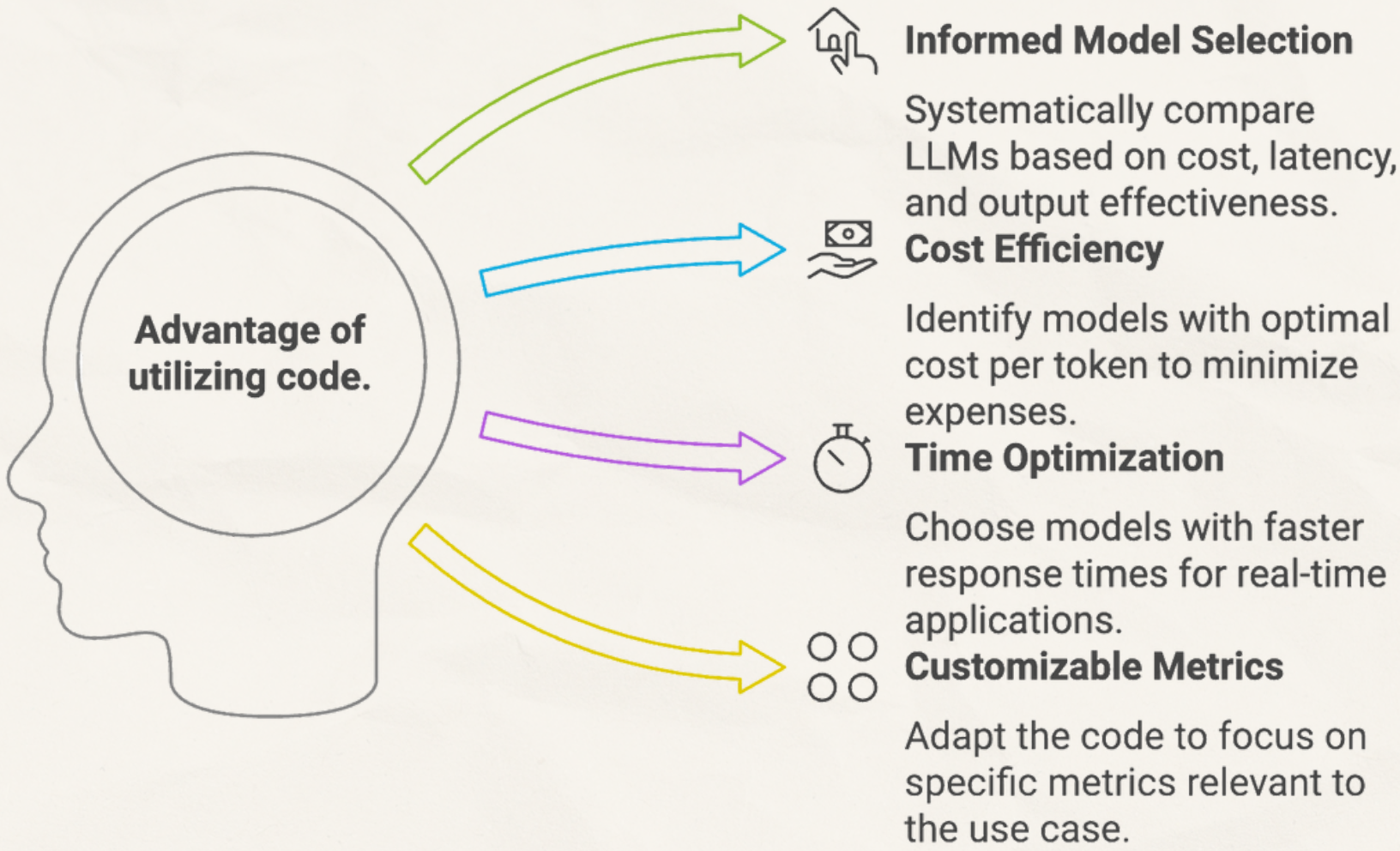
- Response Quality: Use pre-defined criteria or scoring systems to assess the informativeness and conciseness of responses.
- Latency: Measure the time taken from prompt submission to response retrieval.
- Cost: Calculate the operational cost based on token usage for each response.

LLM as Judge: Integrate a decision-making algorithm that compares the evaluated metrics. The model that meets the specified thresholds for quality, latency, and cost is selected as the best option.

Best LLM: Return the recommended model based on the evaluation results and criteria you want to optimize (for example choosing the model with the least cost)

,

BENEFITS OF THIS APPROCH



Parallel Processing:

- Built with parallel processing, the code significantly reduces the time required to compare multiple models and prompt combinations, enhancing workflow efficiency.

Ease of Integration:

- Leveraging LangChain and watsonx.ai, the codebase seamlessly integrates with a variety of open-source LLMs, allowing users to quickly deploy and test models with minimal setup.

Reusable and Scalable:

- Designed to be modular, the code can be easily reused for multiple projects and scaled to compare an increasing number of models as the ecosystem expands.

Improved Workflow:

- By streamlining the process of model comparison, the code saves time and resources, helping users quickly identify the best model for their tasks and reducing the iteration cycle for model evaluation.

LINK TO CODE IN THE COMMENTS

Follow to stay updated on Generative AI

SAVE **LIKE** **SHARE**