Hosted by

Petavue

OpenLLM Day

30 May, 2024

Our Sponsors

mongoDB. GOLD CAP

Our Partners

BRIGHT ACADEMY AI Brains

LLM Benchmark & Evaluation

By Jeyaraj V

Applications of LLMs

NL-to-NL

- Text Summarization
- Translation
- Chatbots & Conversational AI

NL-to-Code

- Code Generation
- Code Completion

NL-to-SQL

- Database Querying
- Business Intelligence

NL-to-Image

- Image Generation
- Image Captioning

NL-to-Audio

- Speech Synthesis
- Voice Assistants

NL-to-Actions

- Automated Workflows
- Robotic Process Automation

Other Applications

- Sentiment Analysis
- Text Classification
- Content Creation
- Legal Document Analysis

Any additional examples for application of LLMs?

Benchmarking & it’s Importance

In the context of Large Language Models (LLMs), benchmarking involves systematically assessing their performance across different platforms against a set of predefined criteria, such as Natural Language to SQL (NL-to-SQL) translation. This includes measuring key metrics such as accuracy, throughput, latency, etc.

why benchmarking is important?

- **Performance Comparison:** Reveals the most accurate, fastest, or cost-effective models for specific tasks.
- **Informed Decision-Making:** Guides organizations in choosing the best models based on their needs and constraints.
- **Resource Allocation:** Identifies models offering the best performance relative to computational and financial costs.
- **Optimizing Performance:** Provides insights for model optimization in real-world applications.

How to do Model Benchmarking

Requirements for Benchmarking:

1. **Select a Use Case -** Define the specific task (e.g., NL to SQL).
2. **Choose a Framework -** Use datasets like BIRD or Spider, which provide question-answer pairs and necessary instructions.
3. **Pass the Prompt to LLMs**:
    a. Input the prompt to the model.
    b. Measure the response against expected answers using relevant metrics.
4. **Manual Validation (if needed) -** Manually generate and validate question-answer pairs if not provided by the framework.
5. **Evaluation Script -** Utilize the eval script typically provided by the framework for accurate assessment.

Example Frameworks

- NL to NL:
    - QuAC: For conversational QA, providing question-answer pairs.
    - SQuAD: For reading comprehension tasks.
- NL to SQL:
    - BIRD: Offers complex, rich challenges for rigorous testing.
    - Spider: A robust dataset for SQL query generation tasks.

Petavue's Benchmark Study

We chose NL to SQL use case to benchmark 18 Large Langauge models. **But why NL to SQL?:**

- **High Relevance**: Crucial for data-driven fields, enabling non-technical users to access data.
- **Complexity**: Tests models' understanding of language and SQL structure.
- **Wide Applicability**: Benefits industries like finance, healthcare, and retail.
- **Innovation Potential**: Drives advancements in LLM technology.
- **Community Impact**: Provides valuable insights and tools for developers.
- NL to SQL showcases LLM capabilities, aligning with our mission of promoting open-source AI advancements.

Dataset Chosen for Benchmark

- **Complexity**: Offers a challenging and rigorous testing ground.
- **Rich Challenges**: Presents diverse scenarios for thorough evaluation.
- **Superior Testing**: Better assesses models' capabilities compared to alternatives like Spider.

Choosing BIRD ensures a comprehensive and demanding evaluation of model performance.

Our Study & Dataset Design

- A research team curated a dataset comprising **360 question-query pairs** from the BIRD's Dev dataset.
- The selection process was meticulous, aiming for comprehensive coverage of challenges.
- The pairs were categorized into **three difficulty levels**: simple, moderate, and challenging, with **120 pairs each**.
- This approach allowed for a thorough evaluation of model capabilities across various complexity levels.
- Equal distribution of difficulty levels ensures a holistic understanding and robust conclusions.

Query-Expected Response Pairs

- Question-query pairs are categorized into **three difficulty levels**: simple, moderate, and challenging.
- Each pair consists of a natural language question and the corresponding expected SQL query.

Simple Query: List down the molecule id for non carcinogenic molecules.

```
SELECT T.molecule_id FROM molecule AS T WHERE T.label = '-'
```

Moderate Query: List the city and state of members enrolled under electrical and computer engineering department.

```
SELECT city, state FROM member AS T1 INNER JOIN major AS T2 ON T2.major_id =
T1.link_to_major INNER JOIN zip_code AS T3 ON T3.zip_code = T1.zip WHERE department =
'Electrical and Computer Engineering Department' AND position = 'Member'
```

Challenging Query: For the branch which located in the south Bohemia with biggest number of inhabitants, what is the percentage of the male clients?

```
SELECT CAST(SUM(T1.gender = 'M') AS REAL) * 100 / COUNT(T1.client_id) FROM client AS T1
INNER JOIN district AS T2 ON T1.district_id = T2.district_id WHERE T2.A3 = 'south
Bohemia' GROUP BY T2.A4 ORDER BY T2.A4 DESC LIMIT 1
```

Inferencing

- Inferencing involves running the models on the selected question-query pairs to generate SQL queries from natural language questions.

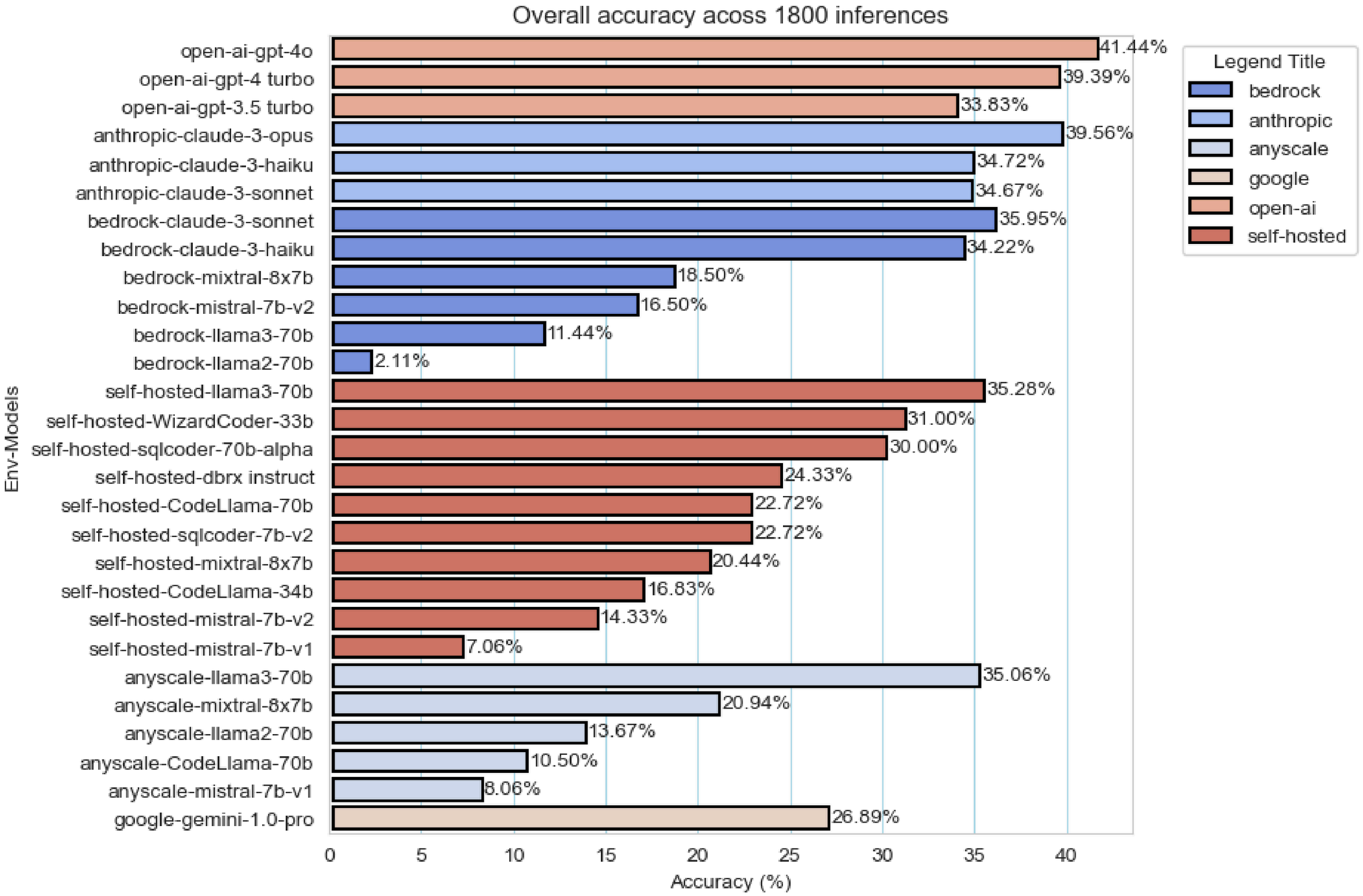
- This process is repeated across various trials with differing levels of instructions to observe how additional guidance impacts model performance.

Evaluation

- The generated SQL queries are evaluated against the expected results using a custom evaluation script that measures **Execution Accuracy**.

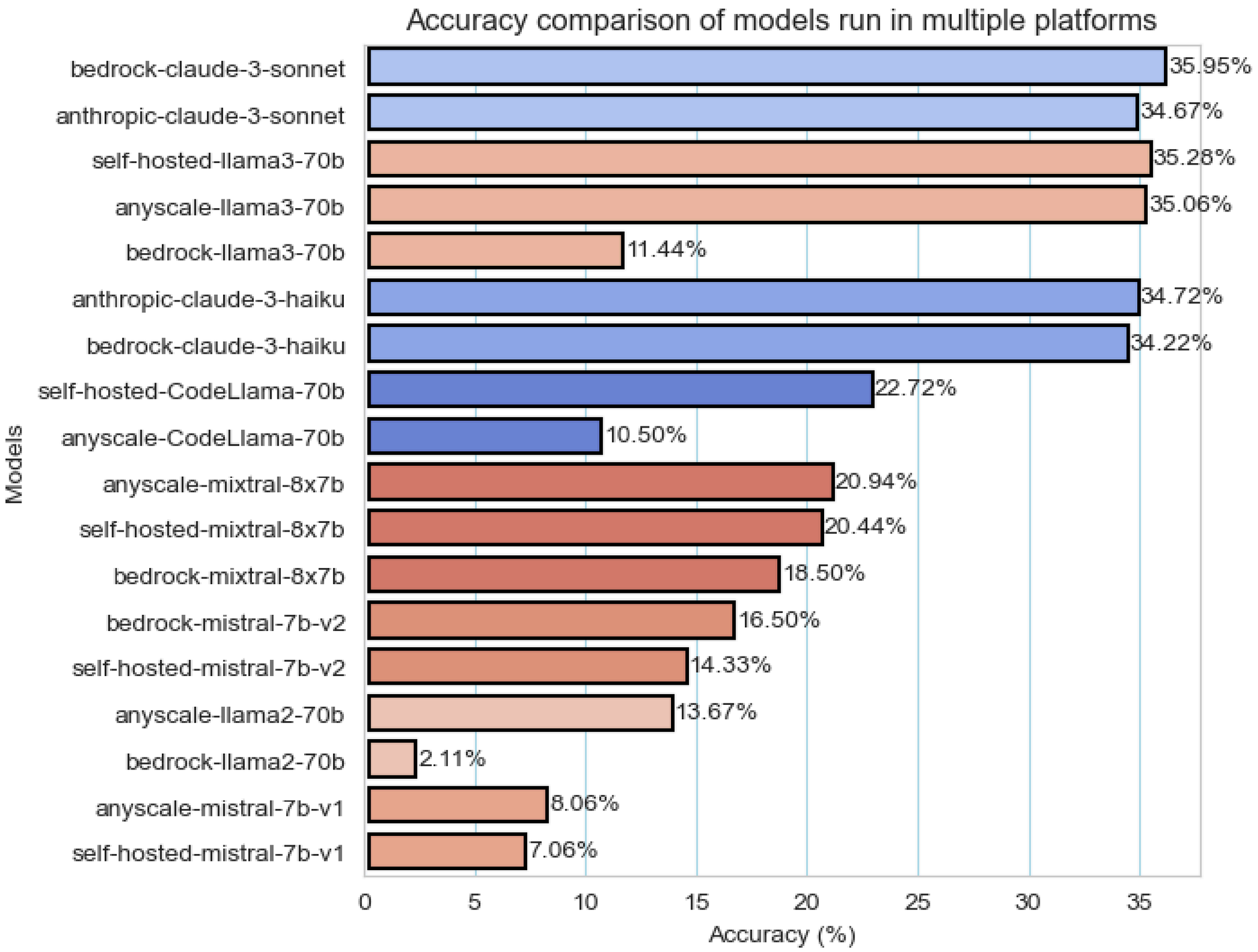
- This involves comparing the result table of the generated SQL queries with the table of expected results to check for correct outputs.

- The evaluation focuses on **overall Execution Accuracy** averaged across multiple inferences.



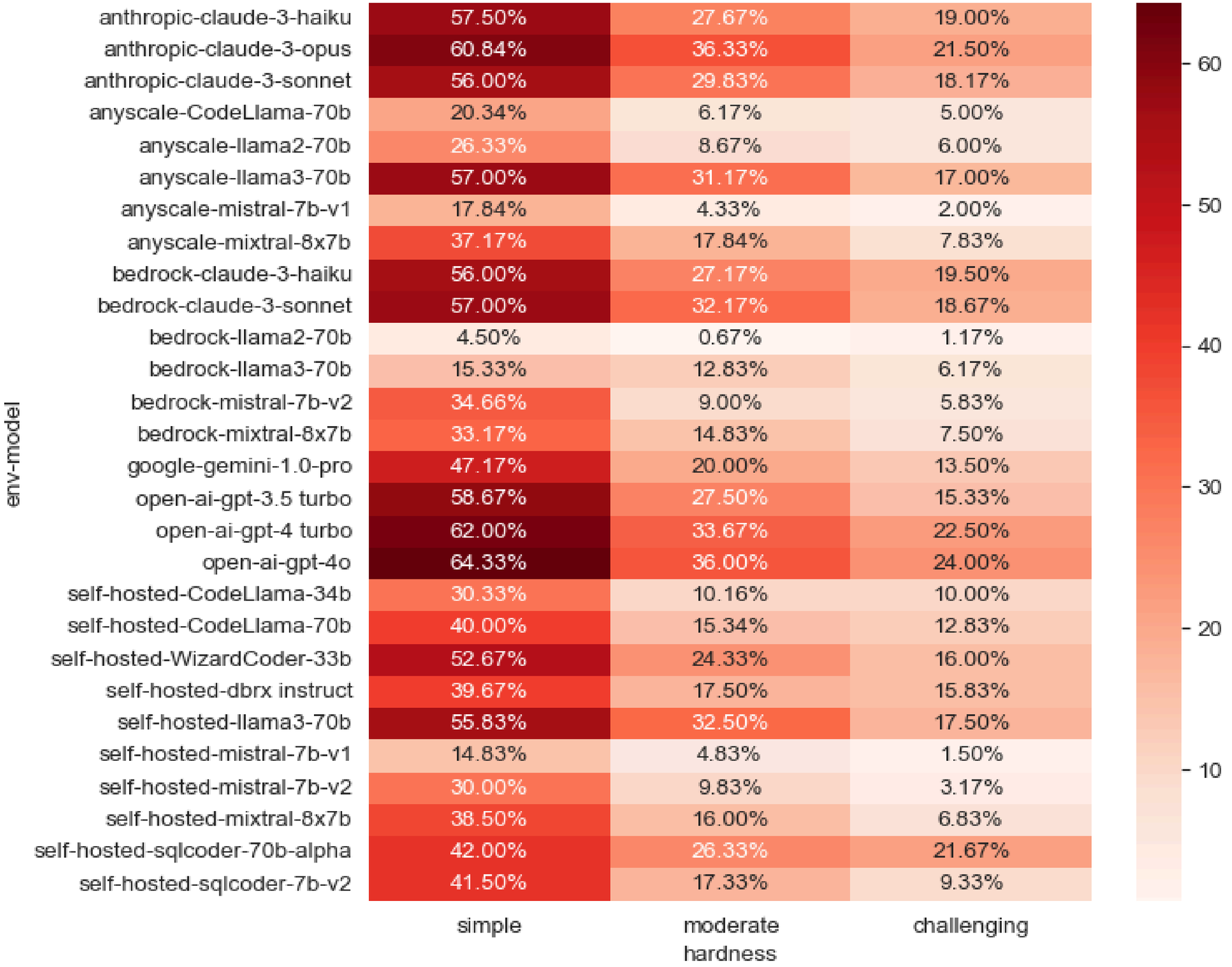
Overall Accuracy

Accuracy based on Platform in which they are run



Overall Accuracy

Accuracy comparison of models in multiple platforms



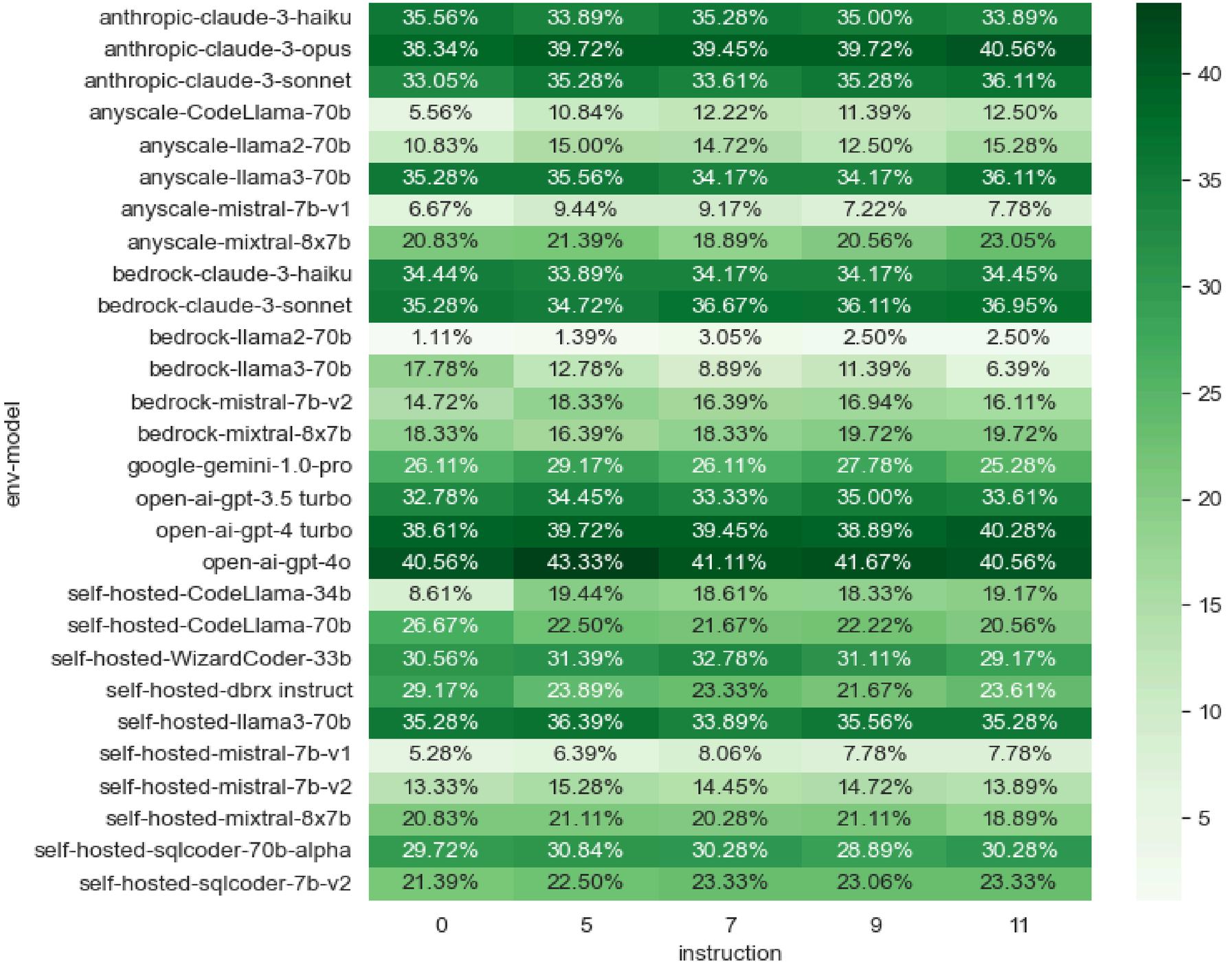
Impact of Difficulty Level of Questions on Model Accuracy

Hardness Levels:

(1) Simple

(2) Moderate

(3) Challenging



Impact of Number of Instructions on Model Accuracy

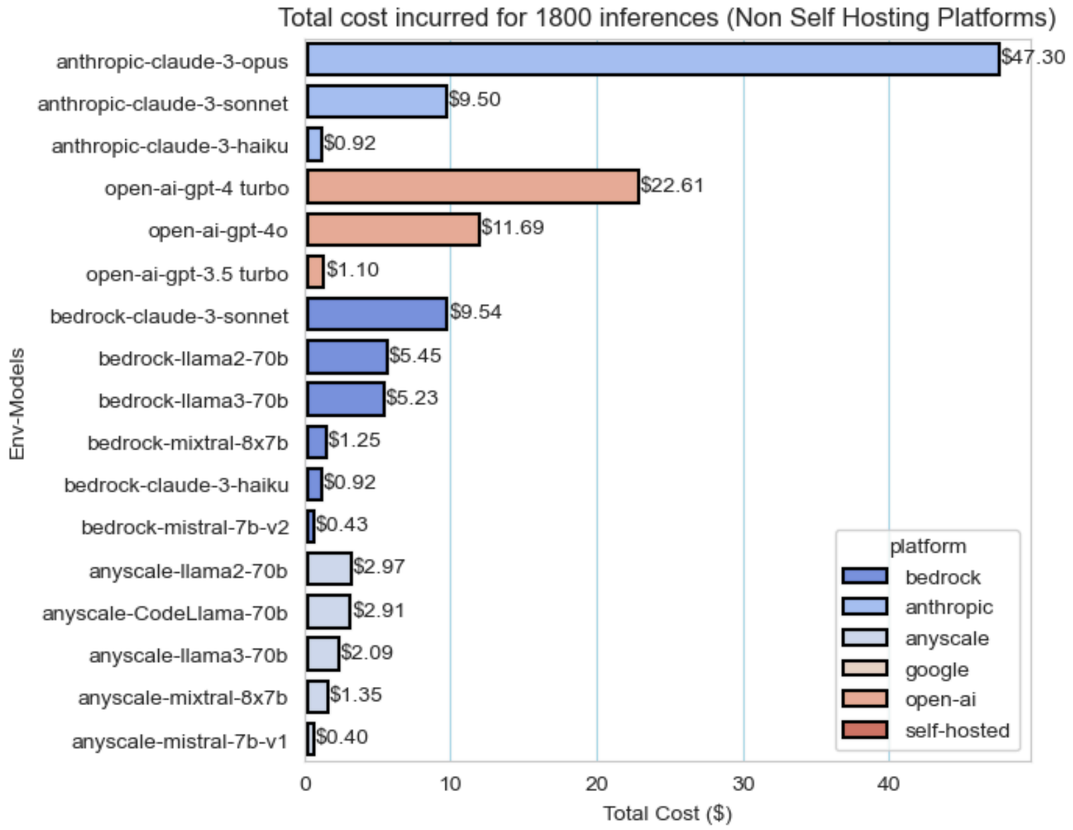
Overall Accuracy summary

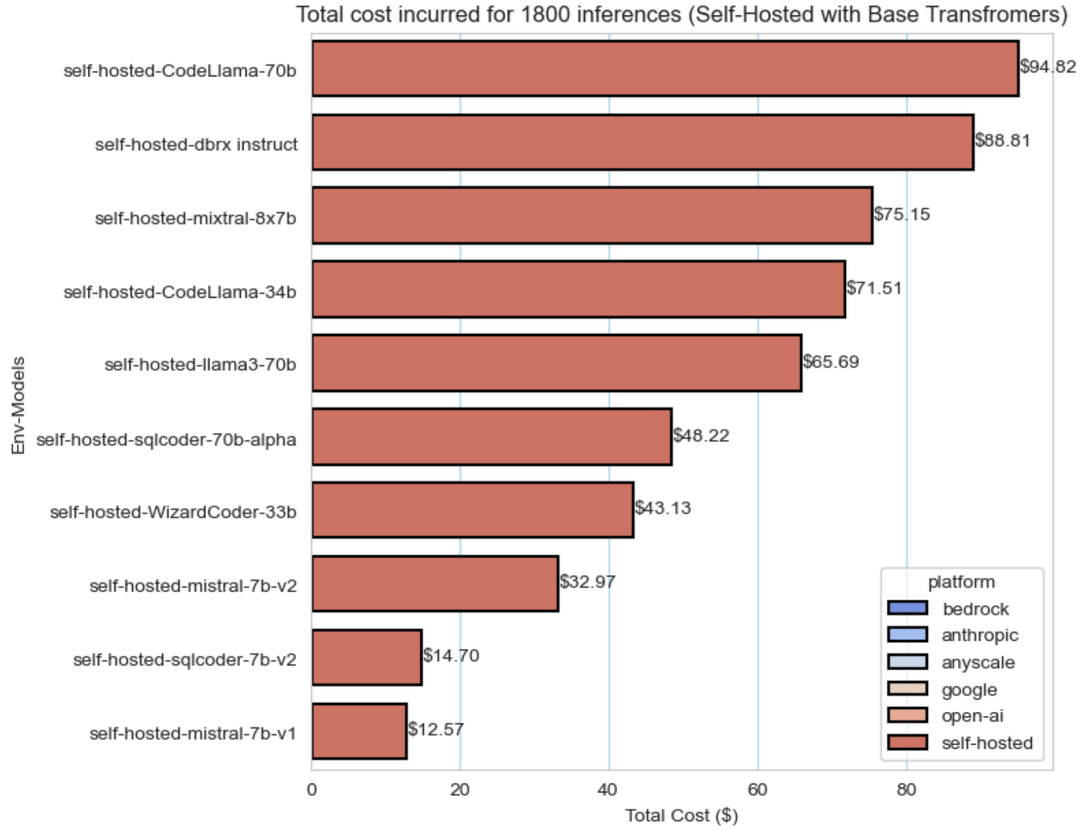
- **Top Performing Models**: The highest accuracy was achieved by OpenAI's GPT-4 Omni (41.44%), Anthropic's Claude3-Opus (39.56%), and OpenAI's GPT-4 Turbo (39.39%).
- **Next Best Models**: Claude3-Sonnet had 35.95% accuracy on Bedrock and 34.67% on Anthropic, while Claude3-Haiku had 34.22% on Bedrock and 34.72% on Anthropic.
- **OpenAI's GPT 3.5 Turbo**: This model achieved an accuracy of 33.83%.
- **Llama3-70B**: Consistent accuracy of 35% was noted for this model in both Self-hosted and Anyscale environments.
- **Self-hosted Models**: Wizardcoder 33B and SQLcoder-70B-alpha achieved accuracies of 31% and 30%, respectively.

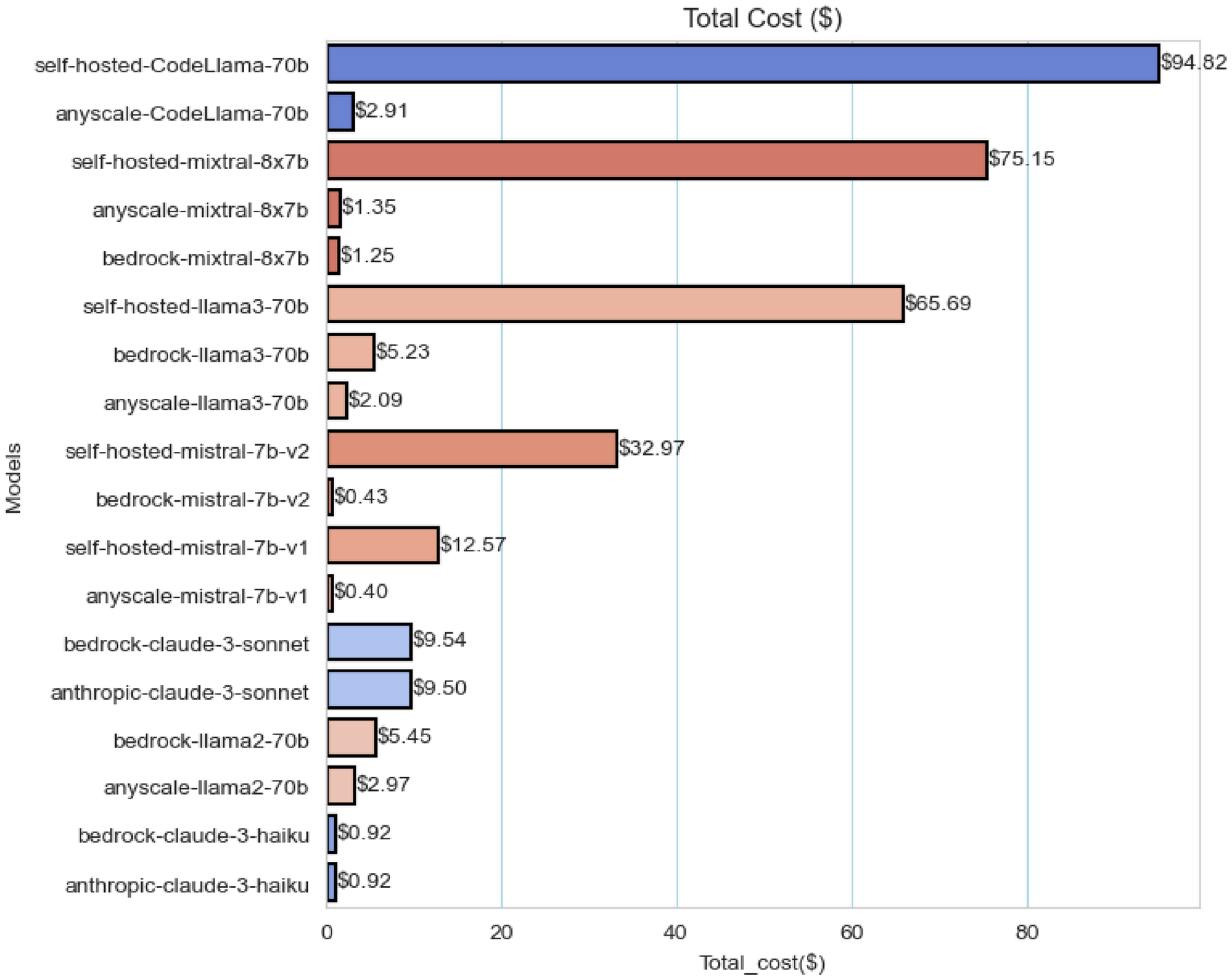
Total Cost

Total Cost based on Platform in which they are run

- Non self hosted platforms influenced by Input and Output tokens.
- Self-hosted influenced by inference time (amount of time taken to run an inference).



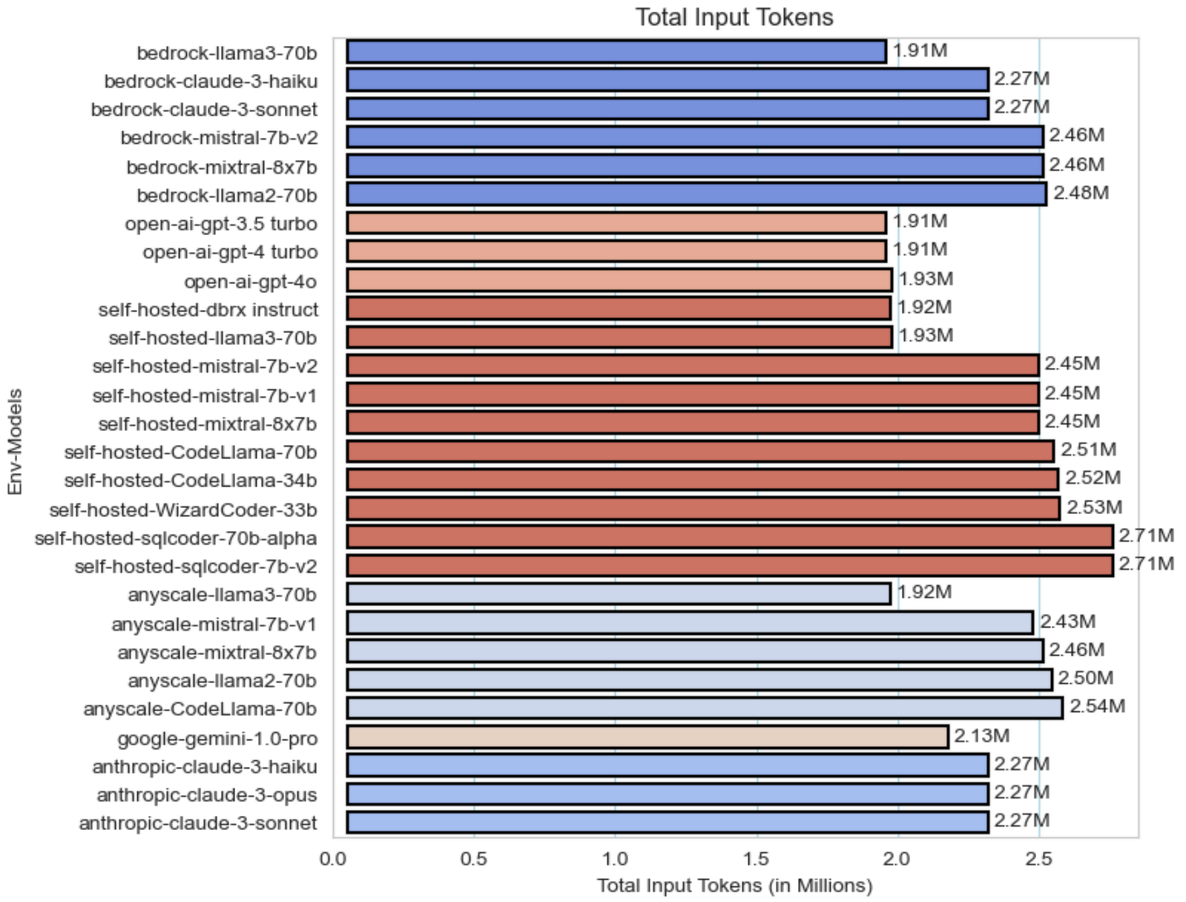


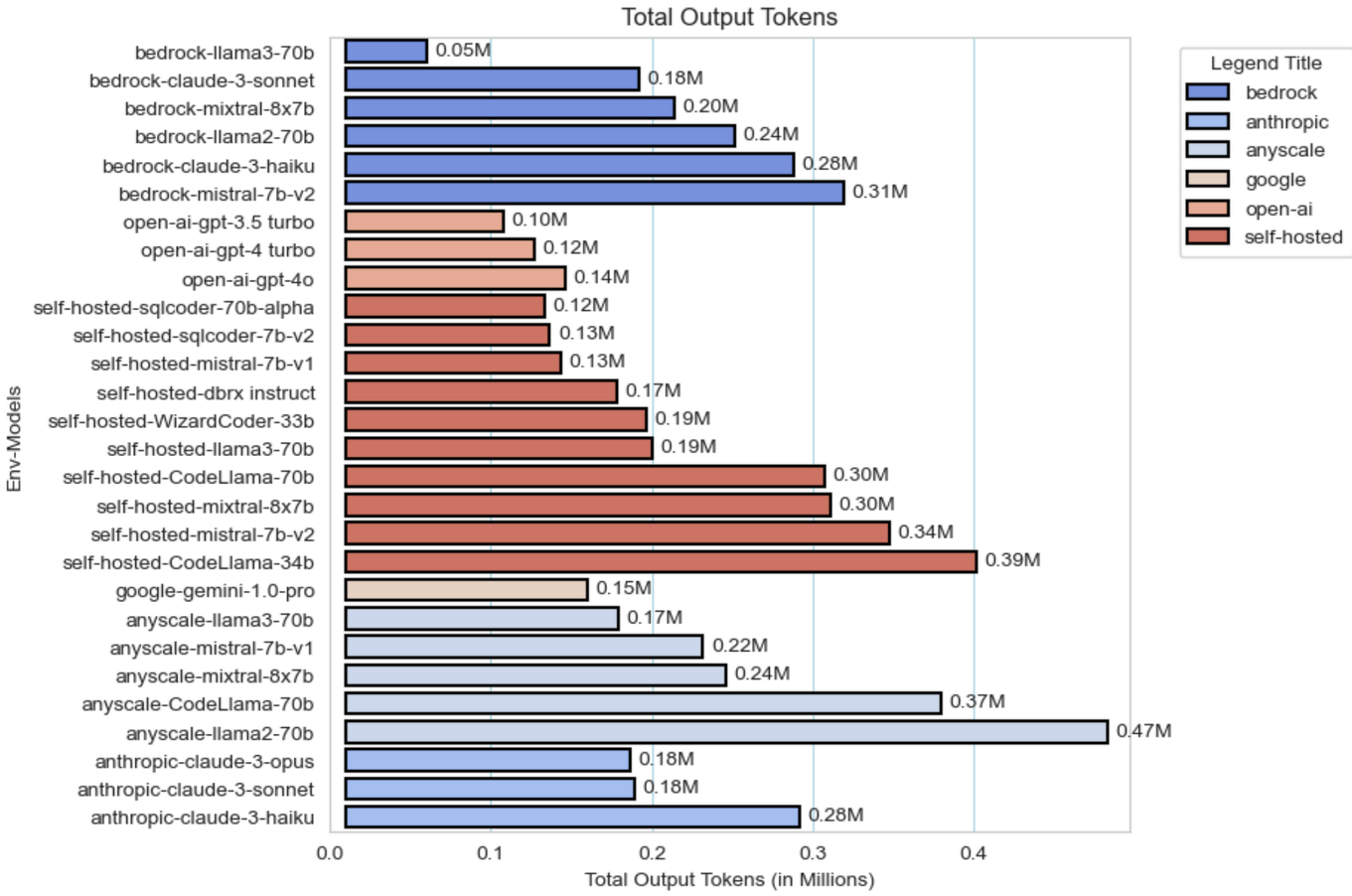


Total Cost

Cost comparison of models run in multiple platforms

Total number of input tokens and total output tokens used for running all models across 1800 inferences.





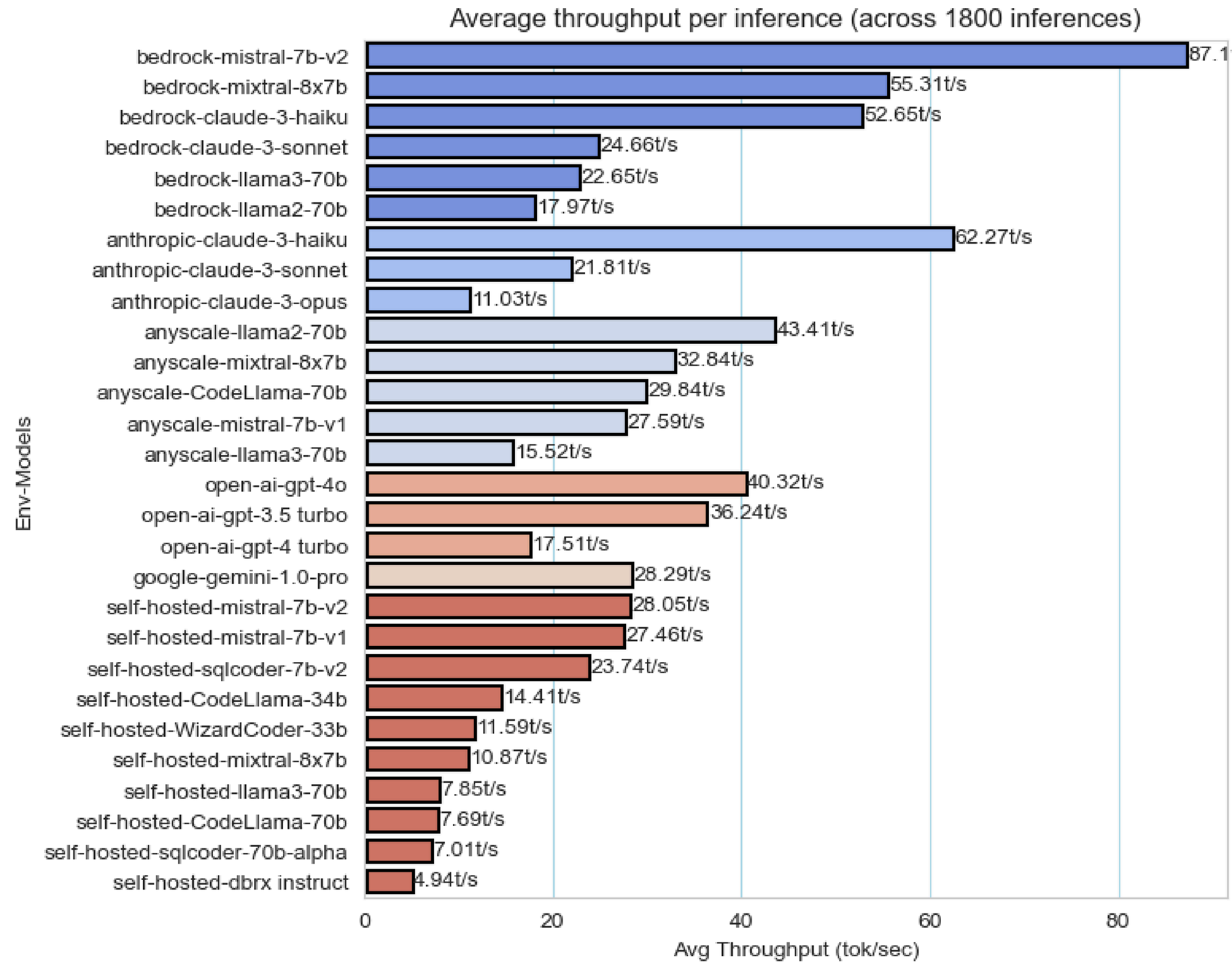
Overall Cost Summary

- **Cost Calculation**: The cost for non self-hosted models is based on the total number of input and output tokens, with differences in tokenization and output token generation affecting overall costs. For self-hosted models, the cost is determined solely by the duration the server is used for inference.

- **Most Expensive Models**: Among non self-hosted models, Anthropic's Claude3 Opus is the most expensive ($47.3), followed by GPT-4 Turbo ($22.61) and GPT-4 Omni ($11.69).

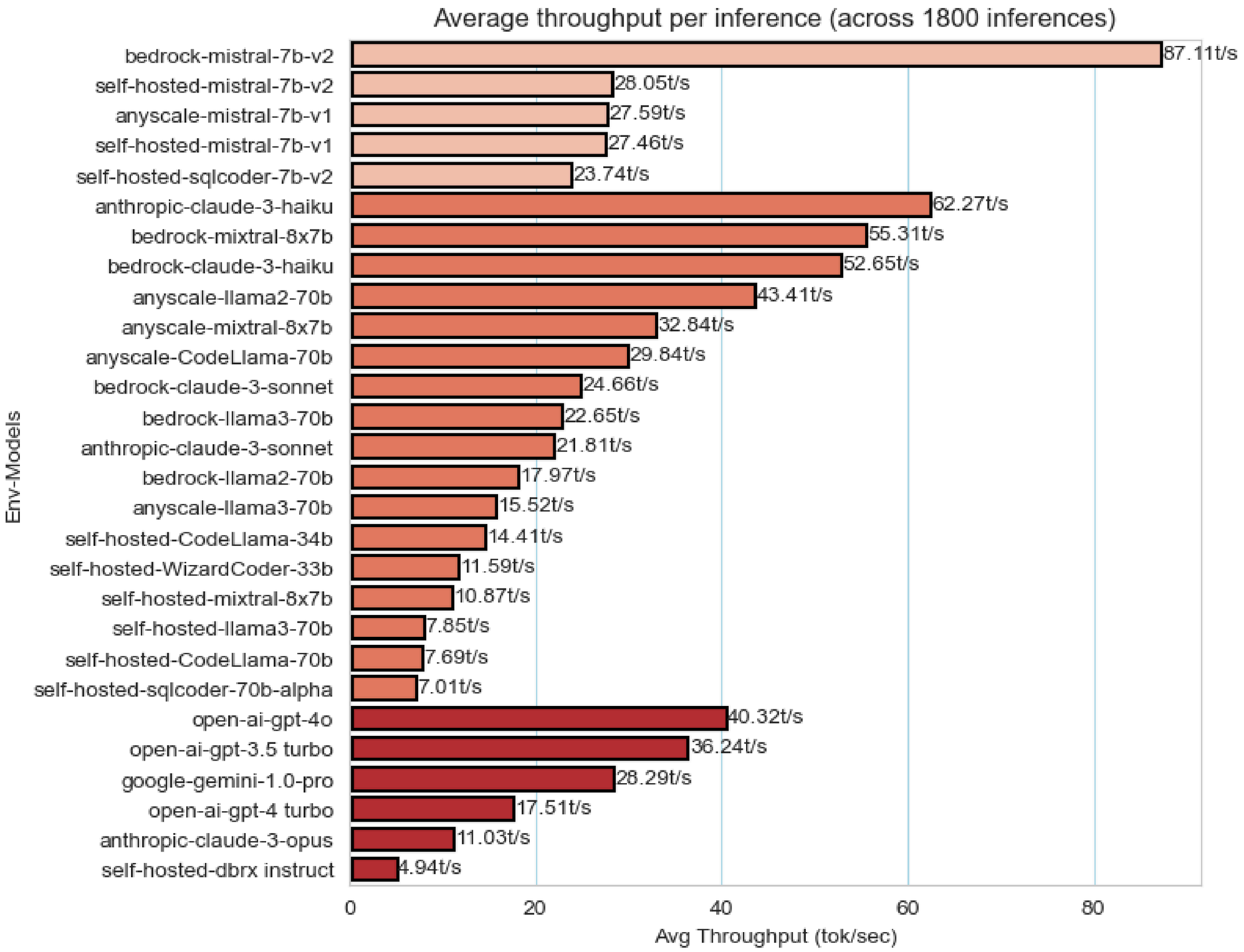
- **Self-Hosted Model Costs**: Costs for self-hosted models are higher due to use of base transformers for inference. For example, CodeLlama-70b and DBRX Instruct have the highest latencies (21.11 sec and 17.78 sec, respectively), leading to higher costs.

- **Platform Optimizations**: Platforms like Anyscale and Bedrock reduce latency through optimizations and inference engines, resulting in lower latency and costs compared to self-hosted setups, where inherent versions without optimizations are used.



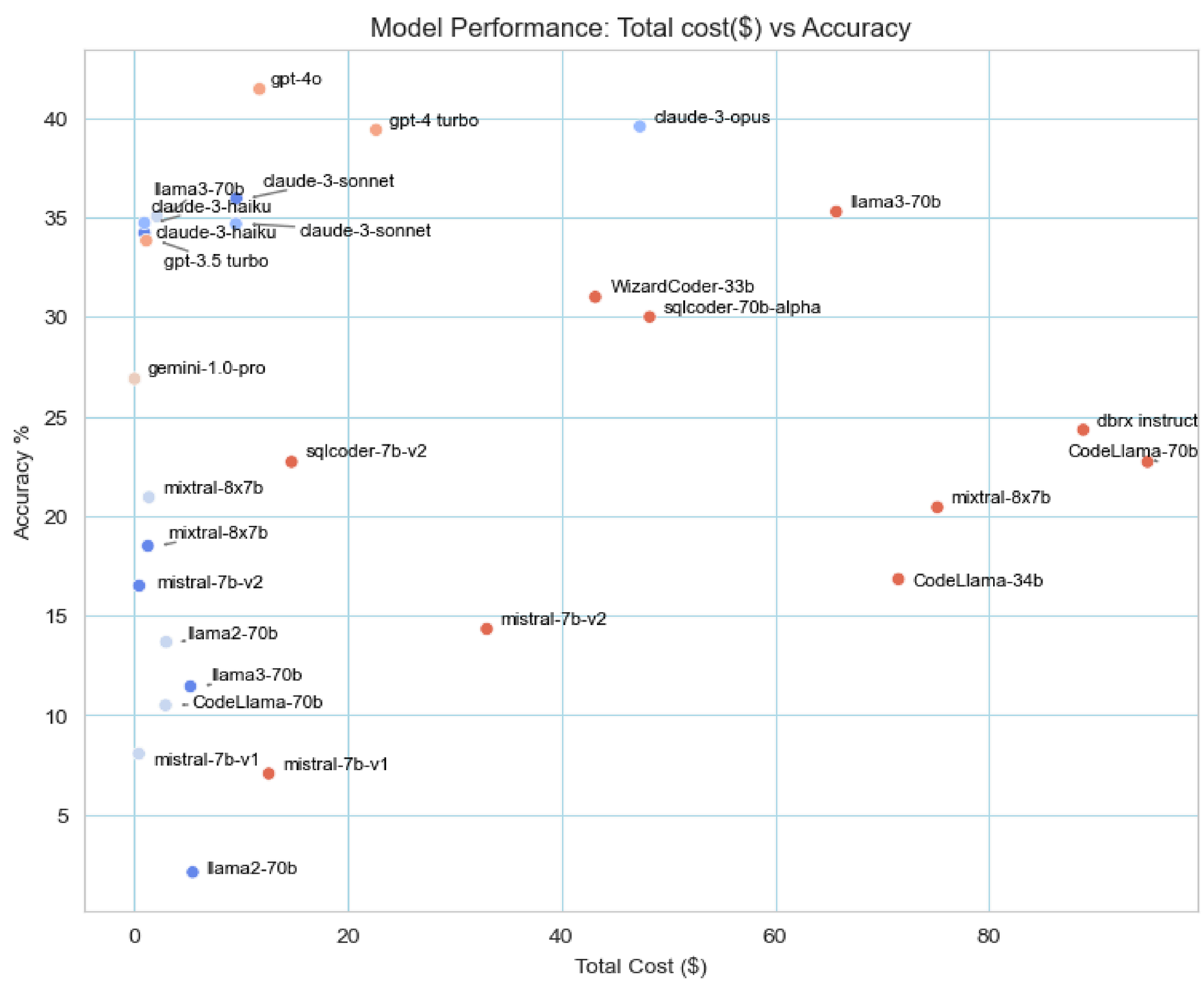
Throughput

Average throughput per inference of all models across 1800 inferences

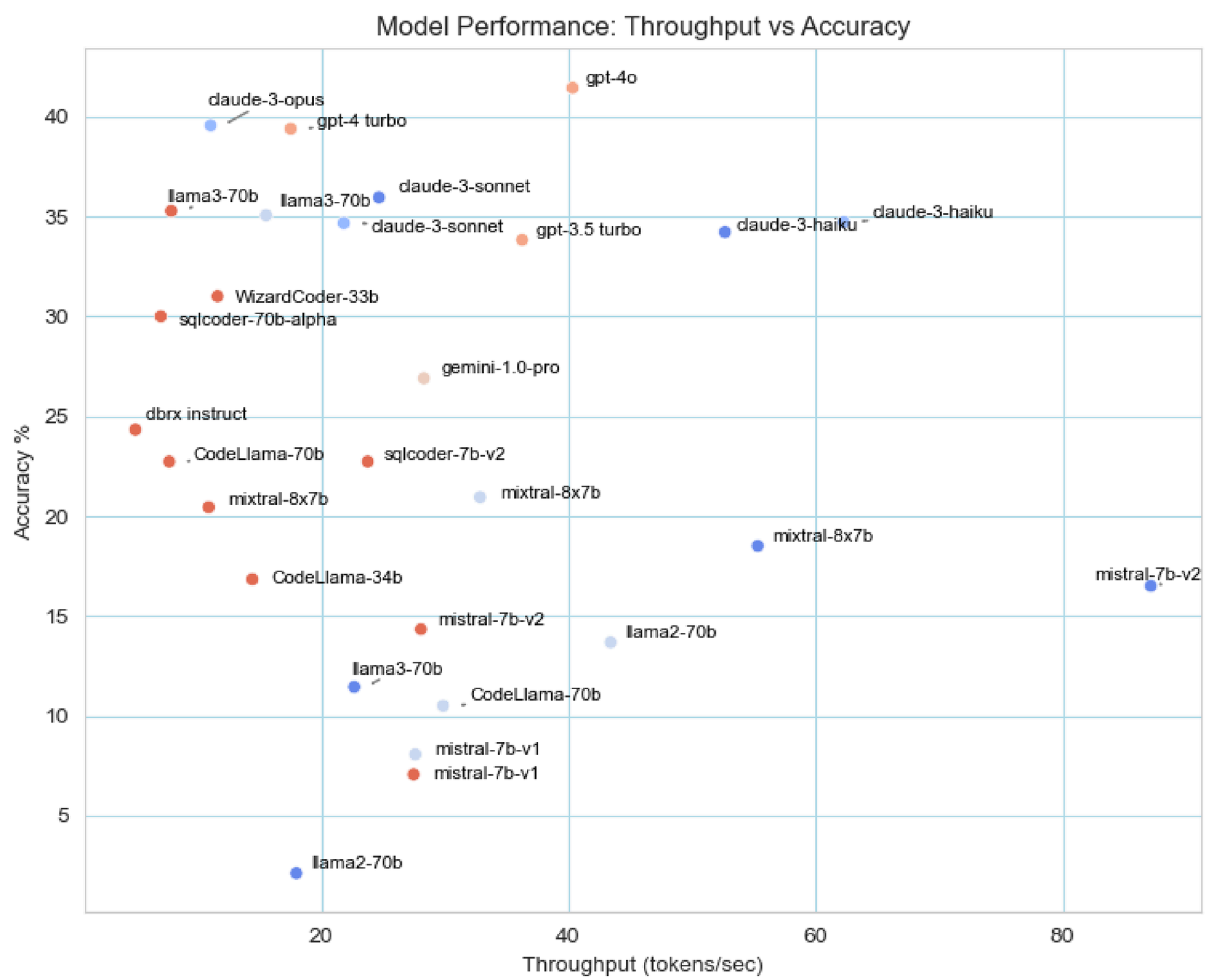


Throughput

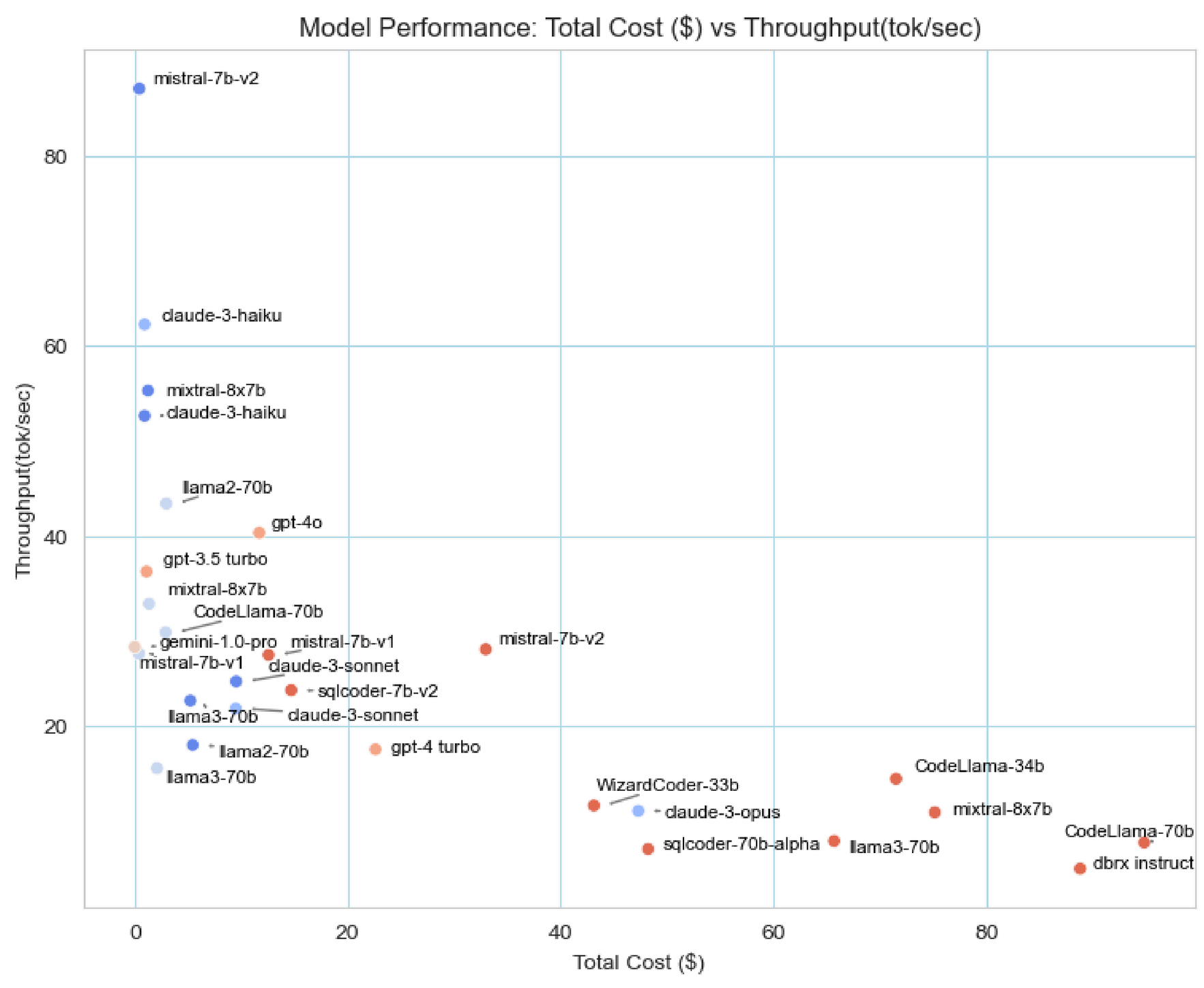
Throughput of all models across different platforms grouped by model size.



Total Cost vs Accuracy



Throughput vs Accuracy



Total Cost vs Throughput

PETAVUE BENCHMARK RESEARCH GIT HUB REPOSITORY

https://github.com/petavue/NL2SQL-Benchmark

See you all in the next session!!

Follow Us on LinkedIn

petavue.com

linkedin.com/company/petavue