OLLSCOIL NA GAILLIMHE
UNIVERSITY OF GALWAY

Optimising AI/ML: Beyond Gradients

Takfarinas Saber
<takfarinas.saber@universityofgalway.ie>

UniversityofGalway.ie

OLLSCOIL NA GAILLIMHE
UNIVERSITY OF GALWAY

About Me

Dr Takfarinas Saber

Call me: **Tak**

Lecturer at University of Galway

Email: takfarinas.saber@universityofgalway.ie

BSc, MSc and PhD Computer Science

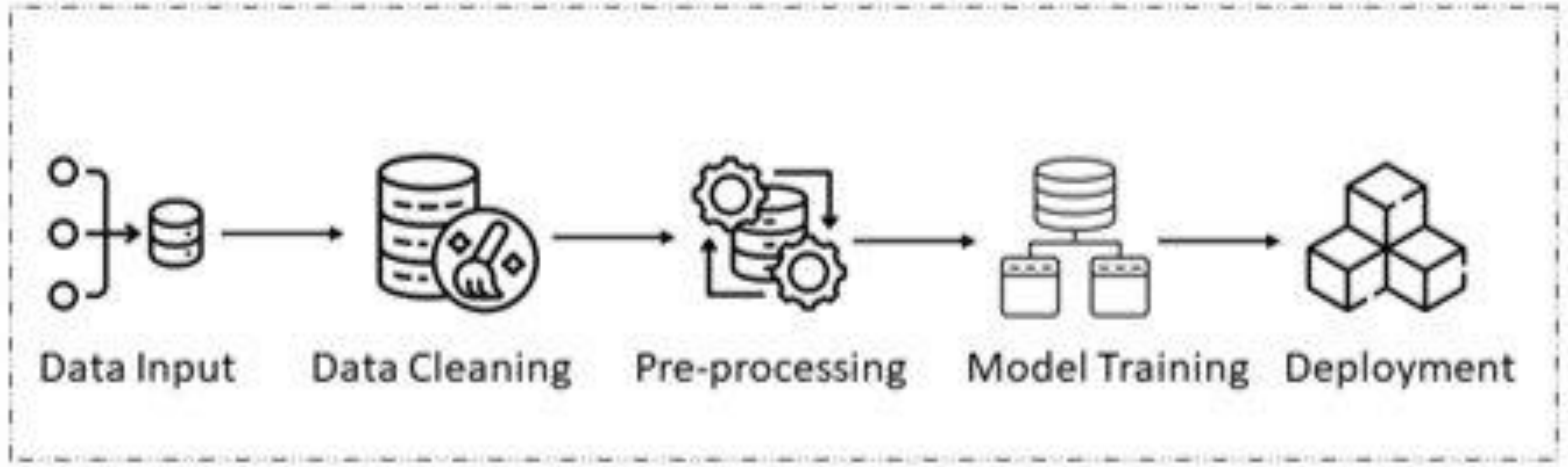
Research Areas:

- Resource Optimisation in Cloud
- Engineering, Testing and Optimisation of Software Applications

ML Pipeline

There are several ML Frameworks.

Why does your ML pipeline cost so much and run so slow?



Data Input → Data Cleaning → Pre-processing → Model Training → Deployment

NLTK
Spark
theano
Numpy
Keras
TensorFlow
Pytorch
Scikit-learn
Pandas
mxnet

The Optimization Maturity Ladder

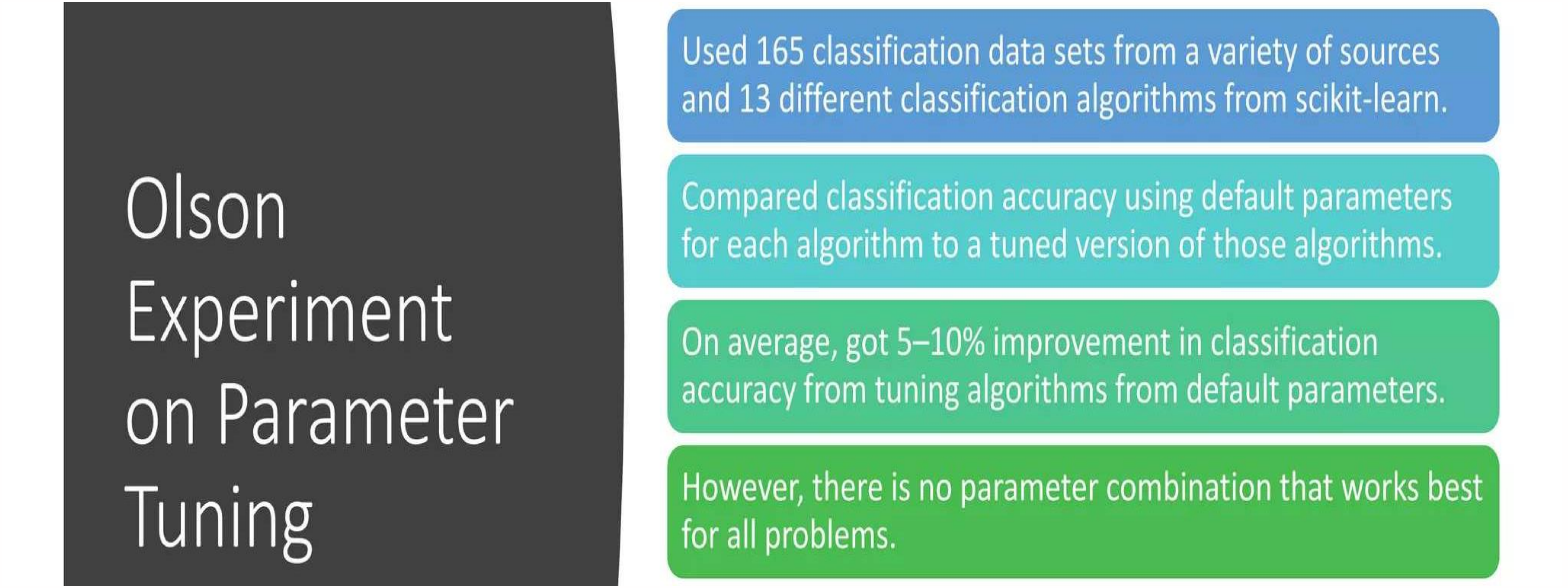
- Level 0: Using default parameters (most software engineers using ML are here)
- Level 1: Manual tuning (most ML teams are)
- Level 2: Single-component optimization
- Level 3: Pipeline-wide optimization
- Level 4: Self-optimizing autonomous MLOps

At what level are your projects?

Phase 2 (★☆☆☆☆): Single component optimization

Hyperparameter tuning (Covered)
Resource allocation
Data sampling

Hyperparameter Optimization (HPO)



Olson Experiment on Parameter Tuning

Used 165 classification data sets from a variety of sources and 13 different classification algorithms from scikit-learn.

Compared classification accuracy using default parameters for each algorithm to a tuned version of those algorithms.

On average, got 5–10% improvement in classification accuracy from tuning algorithms from default parameters.

However, there is no parameter combination that works best for all problems.

Tuning is mandatory to see improvement in ML algorithms

Olson, R.S., Cava, W.L., Mustahsan, Z., Varik, A. and Moore, J.H., 2018. Data-driven advice for applying machine learning to bioinformatics problems. In *Pacific symposium on biocomputing 2018: Proceedings of the pacific symposium* (pp. 192-203).

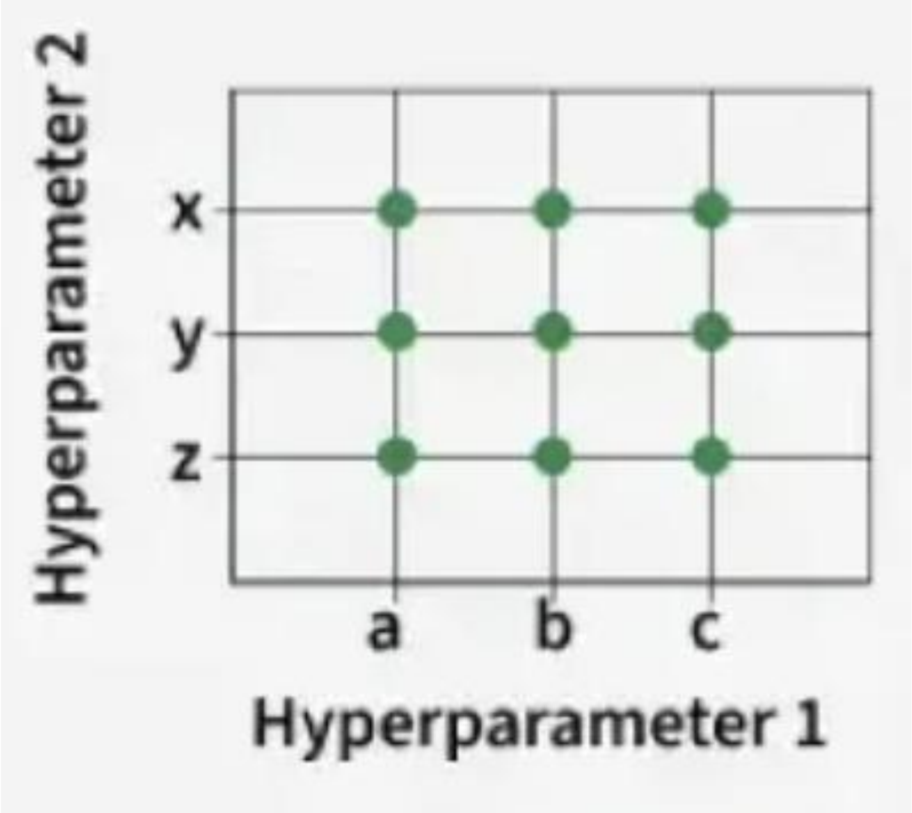
GridSearchCV

Brute-force technique for hyperparameter tuning.

It trains the model using all possible combinations of specified hyperparameter values.

It works using below steps:

- Create a grid of potential values for each hyperparameter.
- Train the model for every combination in the grid.
- Evaluate each model using cross-validation.
- Select the combination that gives the highest score.



It is slow and uses a lot of computer power which makes it hard to use with big datasets or many settings.

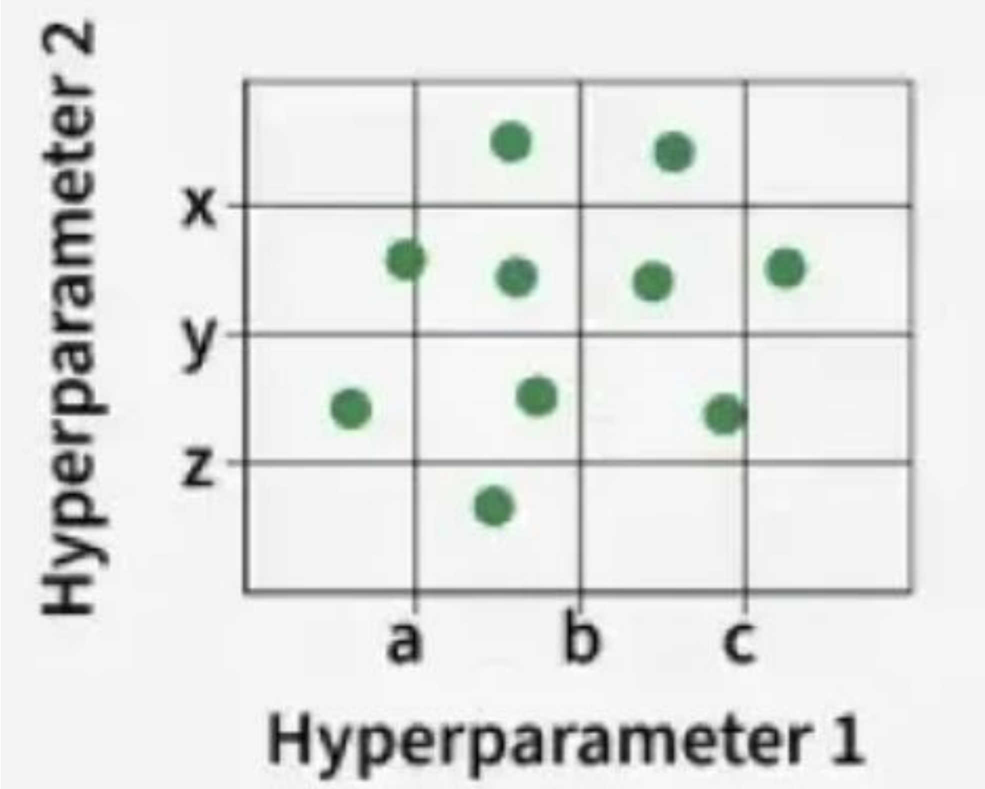
RandomizedSearchCV

Picks random combinations of hyperparameters from the given ranges.

In each iteration:

- It tries a new random combination of hyperparameter values.
- It records the model’s performance for each combination.

After several attempts it selects the best-performing set.

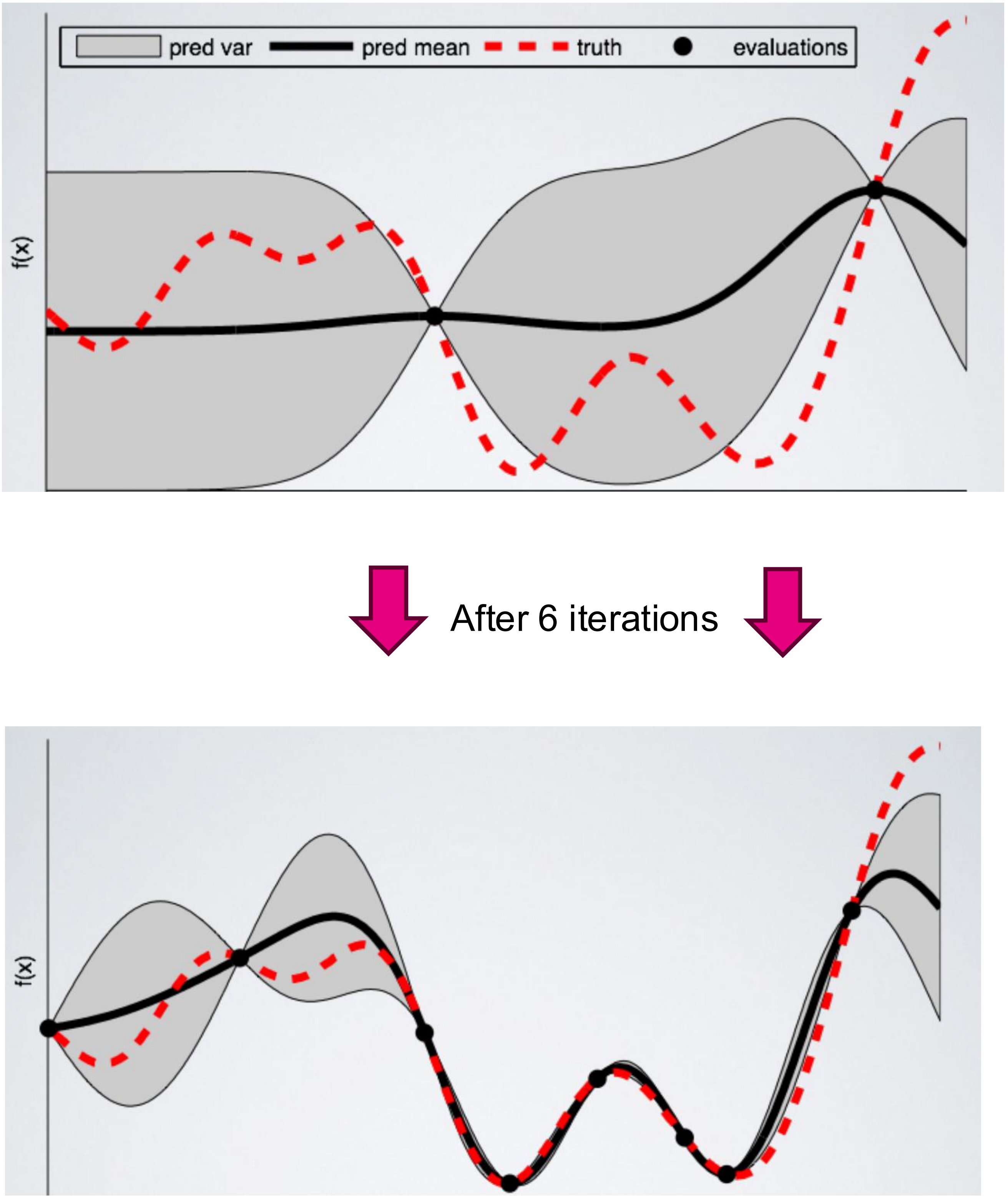


Bayesian Optimization

Grid Search and Random Search can be inefficient because they blindly try many hyperparameter combinations, even if some are clearly not useful.

Bayesian Optimization treats hyperparameter tuning like a mathematical optimization problem and learns from past results to decide what to try next:

1. Build **surrogate model** using initial evaluations of true objective function
2. Find the **hyperparameters** that perform best on the surrogate model
3. Apply these hyperparameters to the true objective function
4. Update the surrogate model incorporating the new results
5. Repeat steps 2–4 until max iterations or time is reached



After 6 iterations

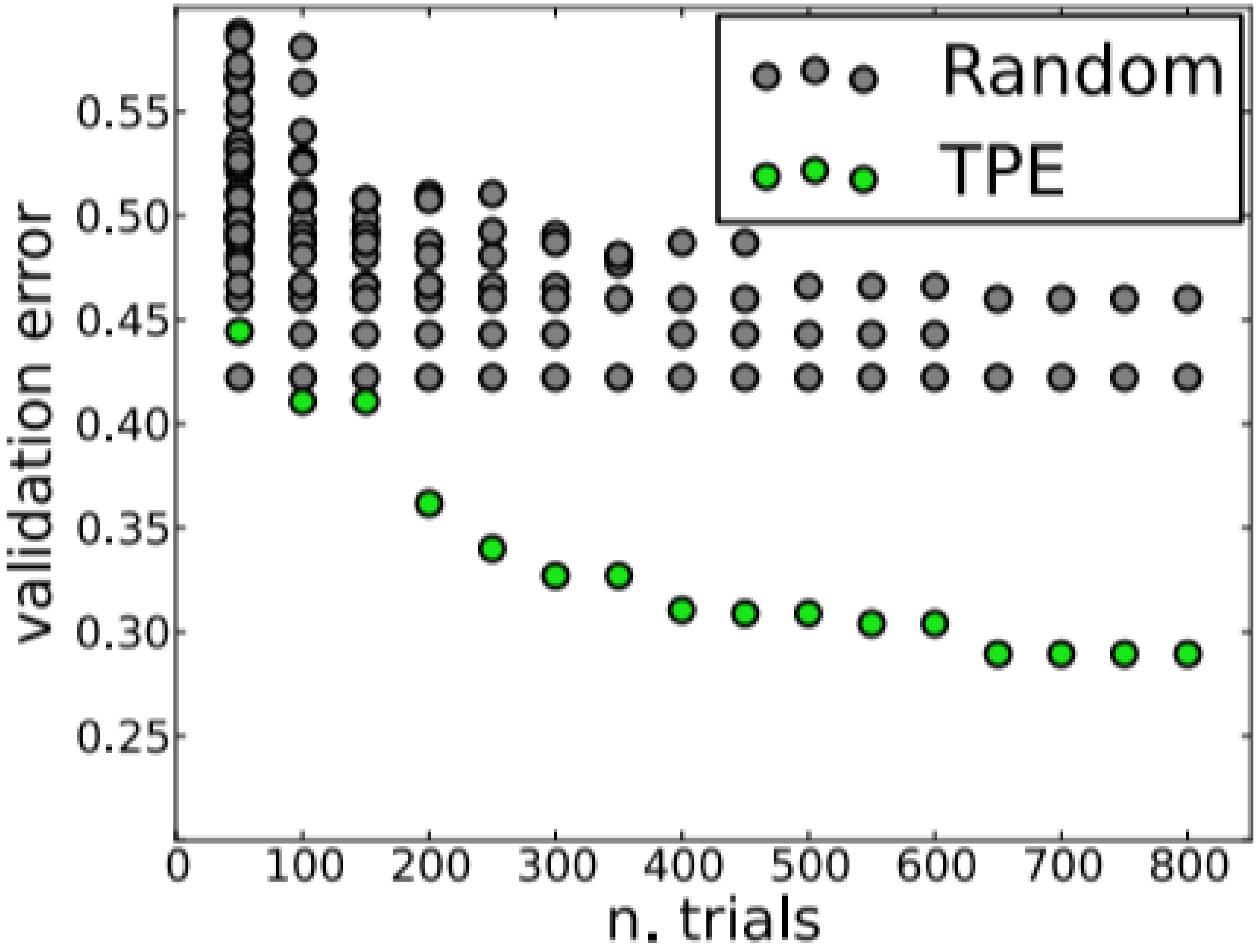
Bayesian Optimization vs Random Search

Common surrogate models used in Bayesian optimization include:

- Gaussian Processes
- Random Forest Regression
- Tree-structured Parzen Estimators (TPE)

Validation error for hyperparameter optimization of an image classification neural network [1] with:

- Random search in grey
- Bayesian Optimization (using the Tree Parzen Estimator or TPE) in green.



[1] Bergstra, J., Yamins, D. and Cox, D., 2013, February. Making a science of model search: Hyperparameter optimization in hundreds of dimensions for vision architectures. In *International conference on machine learning* (pp. 115-123). PMLR.

Advantages of Hyperparameter tuning

- **Improved Model Performance**:
  - Finding the optimal combination of hyperparameters can significantly boost model accuracy and robustness.
- **Reduced Overfitting and Underfitting**:
  - Tuning helps to prevent both overfitting and underfitting resulting in a well-balanced model.
- **Enhanced Model Generalizability**:
  - By selecting hyperparameters that optimize performance on validation data the model is more likely to generalize well to unseen data.
- **Optimized Resource Utilization**:
  - With careful tuning resources (e.g., computation time and memory) can be used more efficiently avoiding unnecessary work.
- **Improved Model Interpretability**:
  - Properly tuned hyperparameters can make the model simpler and easier to interpret.

Challenges in Hyperparameter Tuning

- **Dealing with High-Dimensional Hyperparameter Spaces:**
  - The larger the hyperparameter space the more combinations need to be explored. This makes the search process computationally expensive and time-consuming especially for complex models with many hyperparameters.

- **Handling Expensive Function Evaluations:**
  - Evaluating a model's performance can be computationally expensive, particularly for models that require a lot of data or iterations.

- **Auxiliary problems:**
  - Resource Allocation for Single Training Job
  - Optimal subset from large dataset
  - Maximize model performance with minimum data

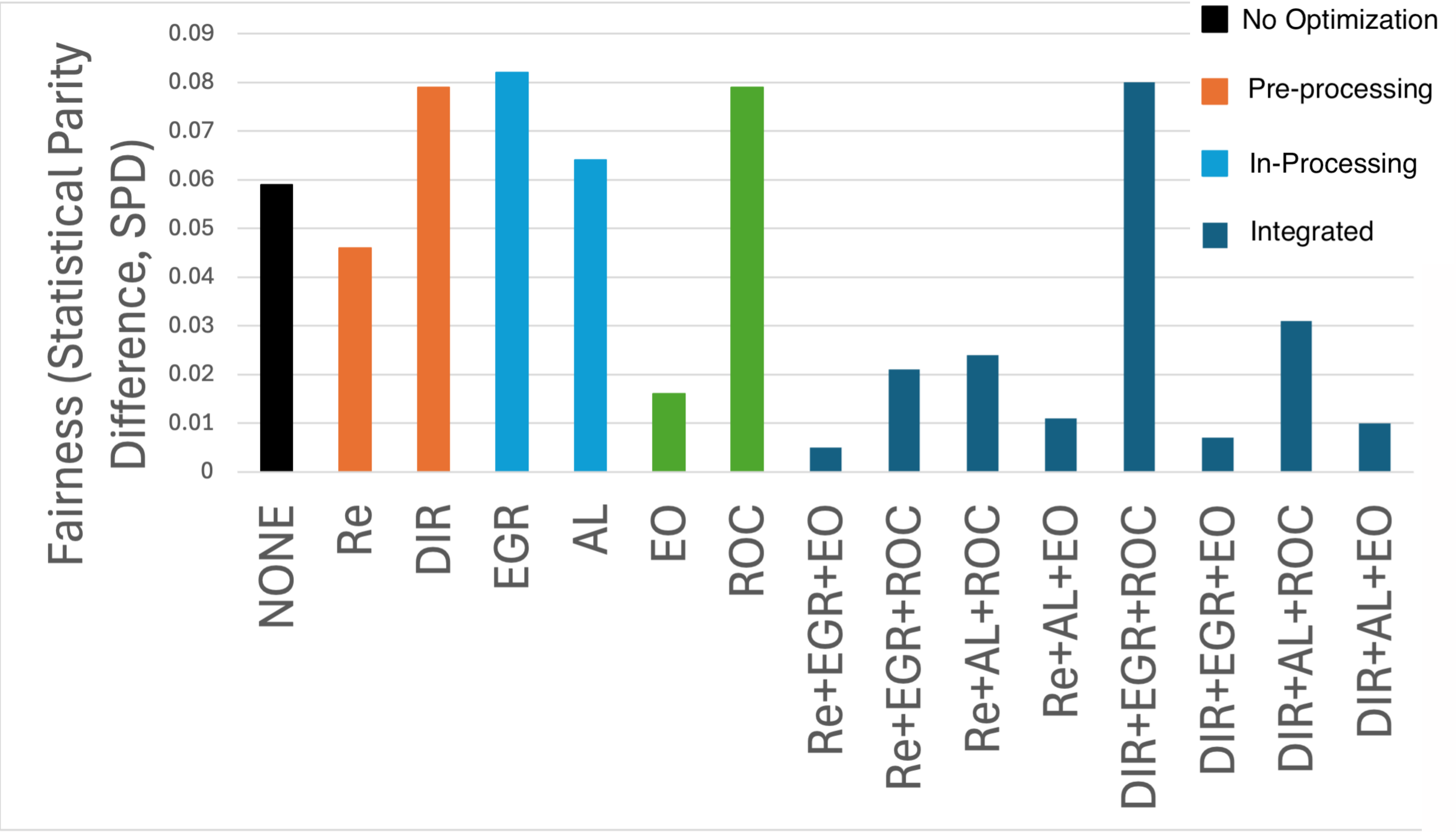
All maturity levels above inherit from these challenges and problems (at a larger scale)

Phase 3 (★★★☆☆): Pipeline-level optimization

AutoML pipeline construction (Covered)
DAG scheduling
Distributed training

Optimizing ML Components Separately Often Fails

- Optimizing each component individually ignores interactions between stages.
- Locally optimal choices may lead to globally suboptimal pipelines.
    - E.g., Selecting the best data processing approaches for Fairness Optimisation [1].



The whole pipeline is more than the sum of its parts!

[1] Farayola, M.M., Tal, I., Saber, T., Connolly, R. and Bendechache, M., 2025. A fairness-focused approach to recidivism prediction: implications for accuracy, trust, and equity. *AI & SOCIETY*, pp.1-19.

AutoML Pipeline Construction

Approach 1: Combined Algorithm Selection and Hyperparameter optimization (CASH)
Focuses on, searching a vast space to find the optimal "single model" and "its hyperparameters."

E.g., Generalizing the Bayesian Algorithm.

The search space is defined in two levels as fellows:
- Choices of classifier / regressor and preprocessing methods are top level categorical hyperparameters
- Hyperparameters of selected methods become active when they are chosen.

The combined space can then be searched with Bayesian Optimization methods.
- But need to be able to handle high-dimensional hierarchical spaces

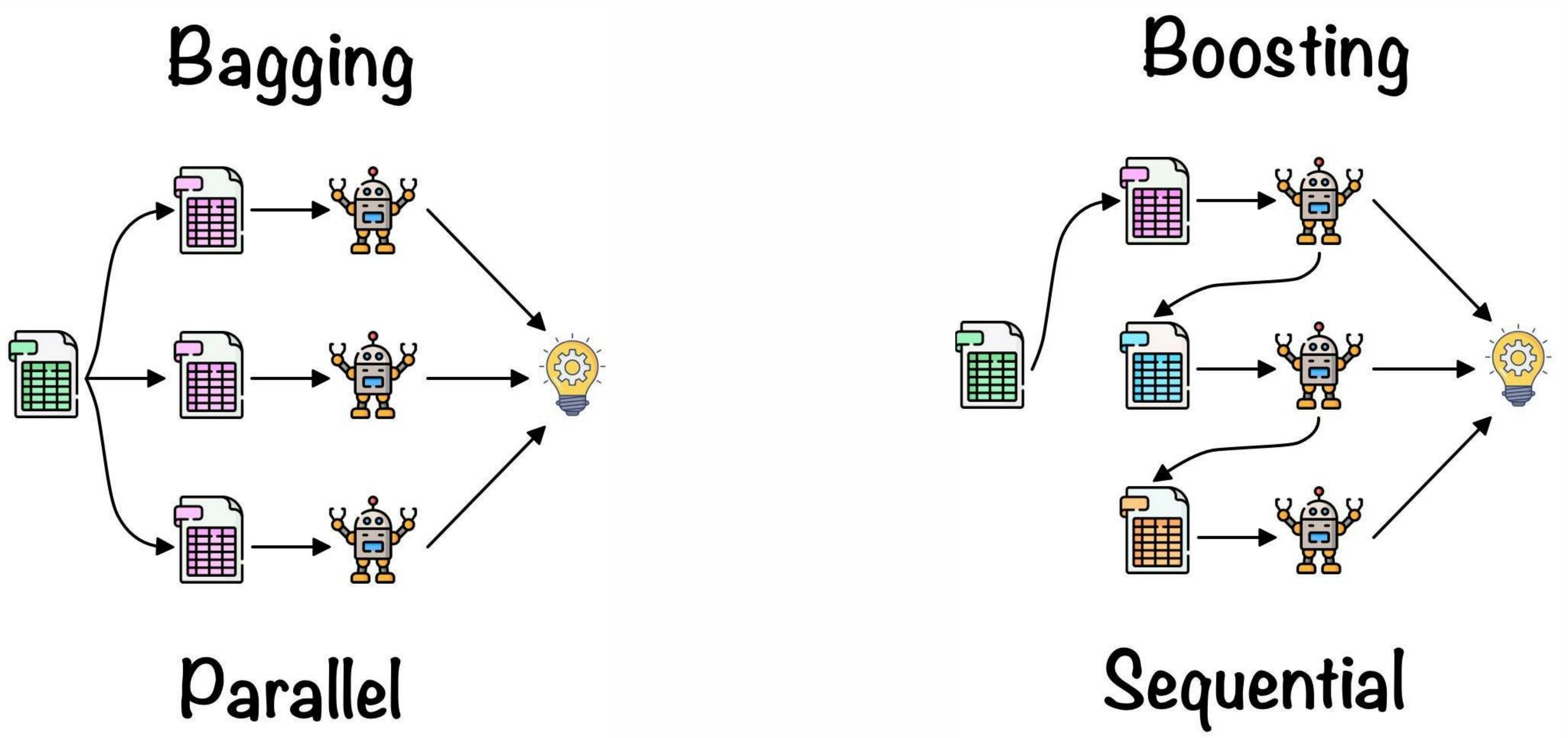
Auto-Sklearn

⚠ Issues various **common issues**, especially related to dependencies, system setup, and resource limits.

AutoML Pipeline Construction (Cont'd)

Approach 2: Ensembling
Achieves success by **ensembling** multiple models and stacking them in **multiple layers**.

The belief is that wisely combining many "good models" yields better results within a given time limit than searching for a "single perfect model."



AutoML Pipeline Construction (Cont'd)

Ollscoil na Gaillimhe
University of Galway

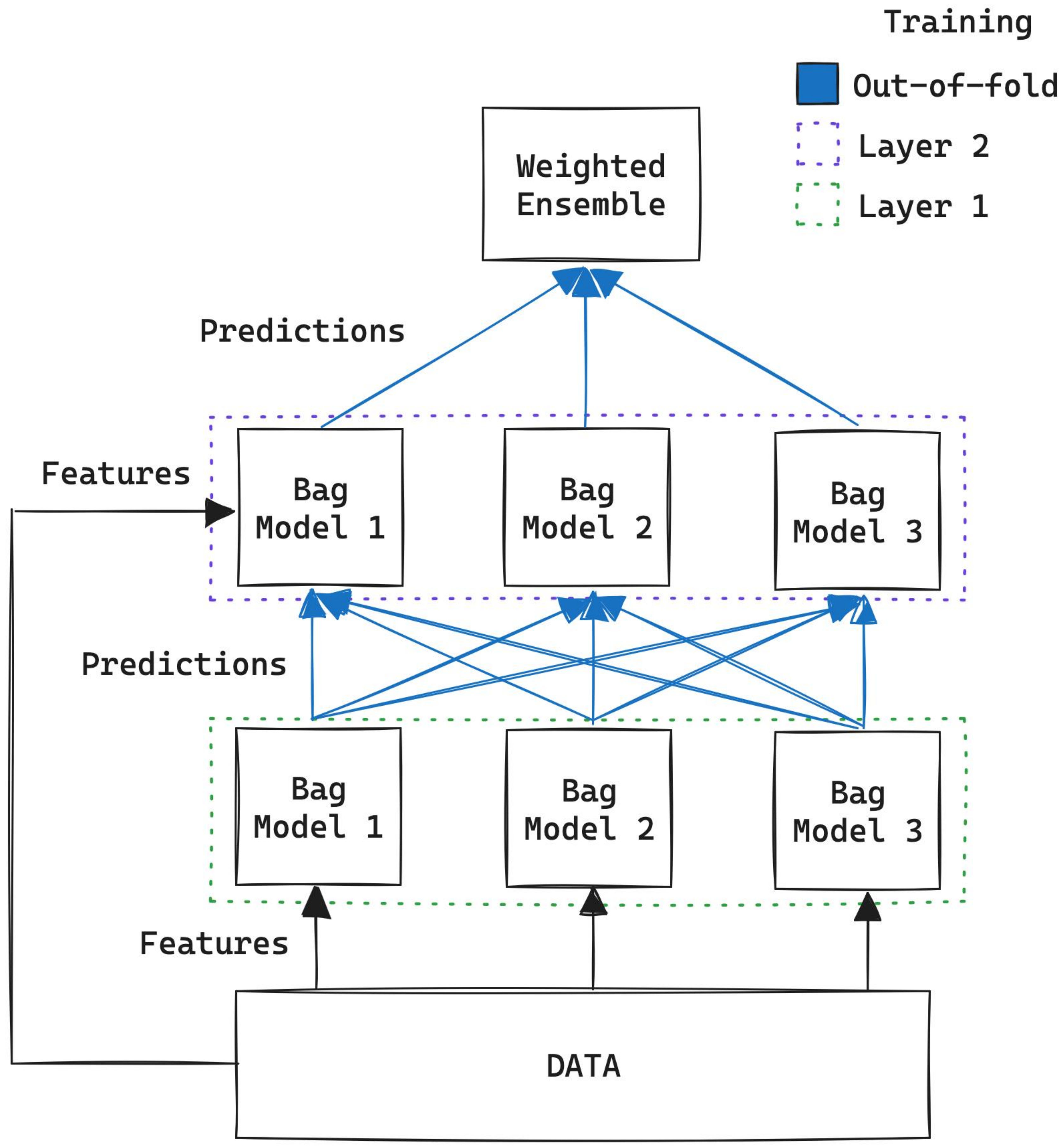
Approach 2: Ensembling

AutoGluon

- Layer(s) of Models
- Weighted Ensemble
- Residual Connections

What Models To Use: pre-defined selection.

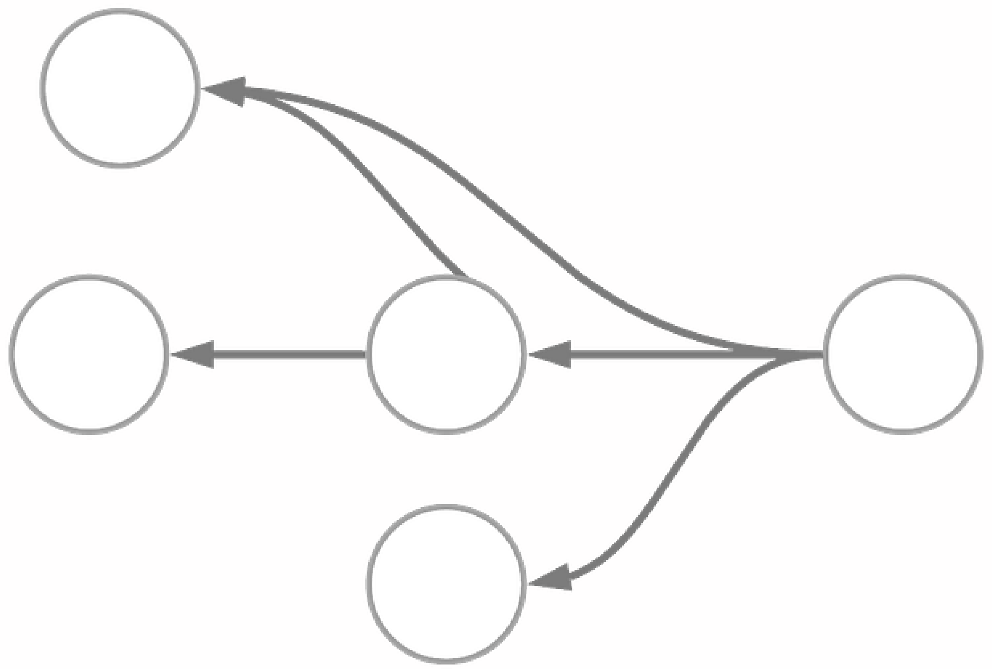
- Based on the comparison of 1,310 models on 200 distinct datasets with various configs/hyperparameters https://huggingface.co/spaces/TabArena/leaderboard



AutoML Pipeline Construction (Cont’d)

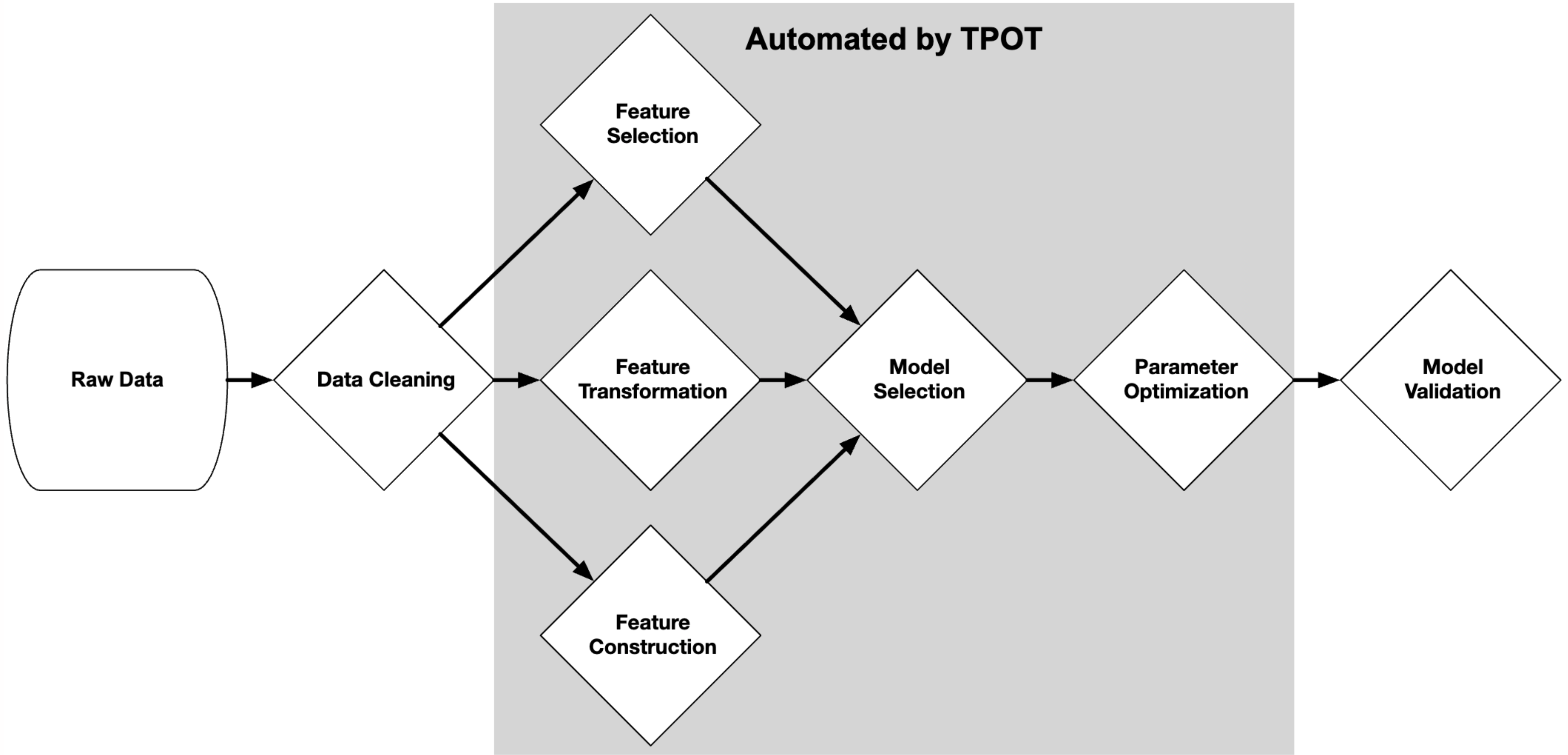
Approach 3: Directed Acyclic Graph (DAG)-Based Optimization

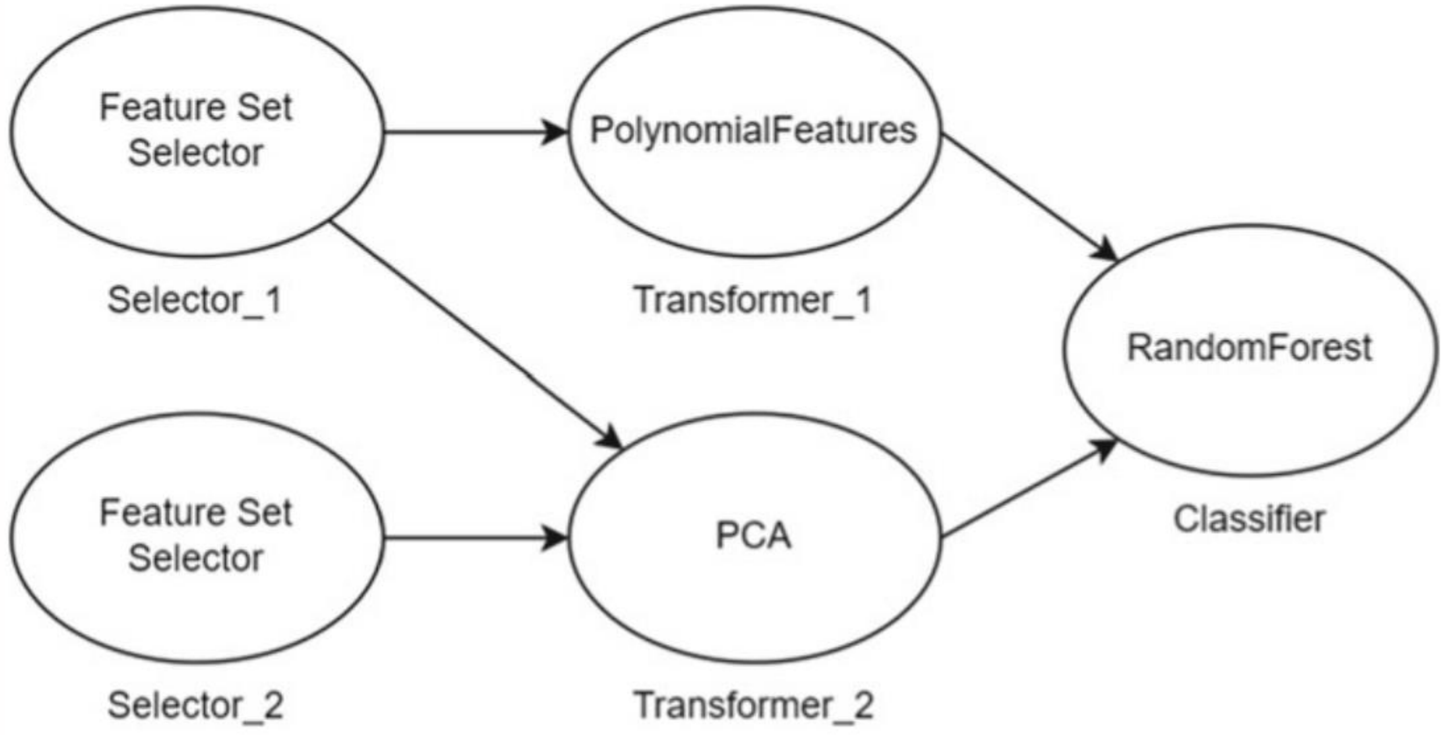
- Define the ML pipeline as a Directed Acyclic Graph (DAG) search space
- Search for the best DAG



AutoML Pipeline Construction (Cont'd)

Approach 3: Directed Acyclic Graph (DAG)-Based Optimization





Example of TPOT Pipeline

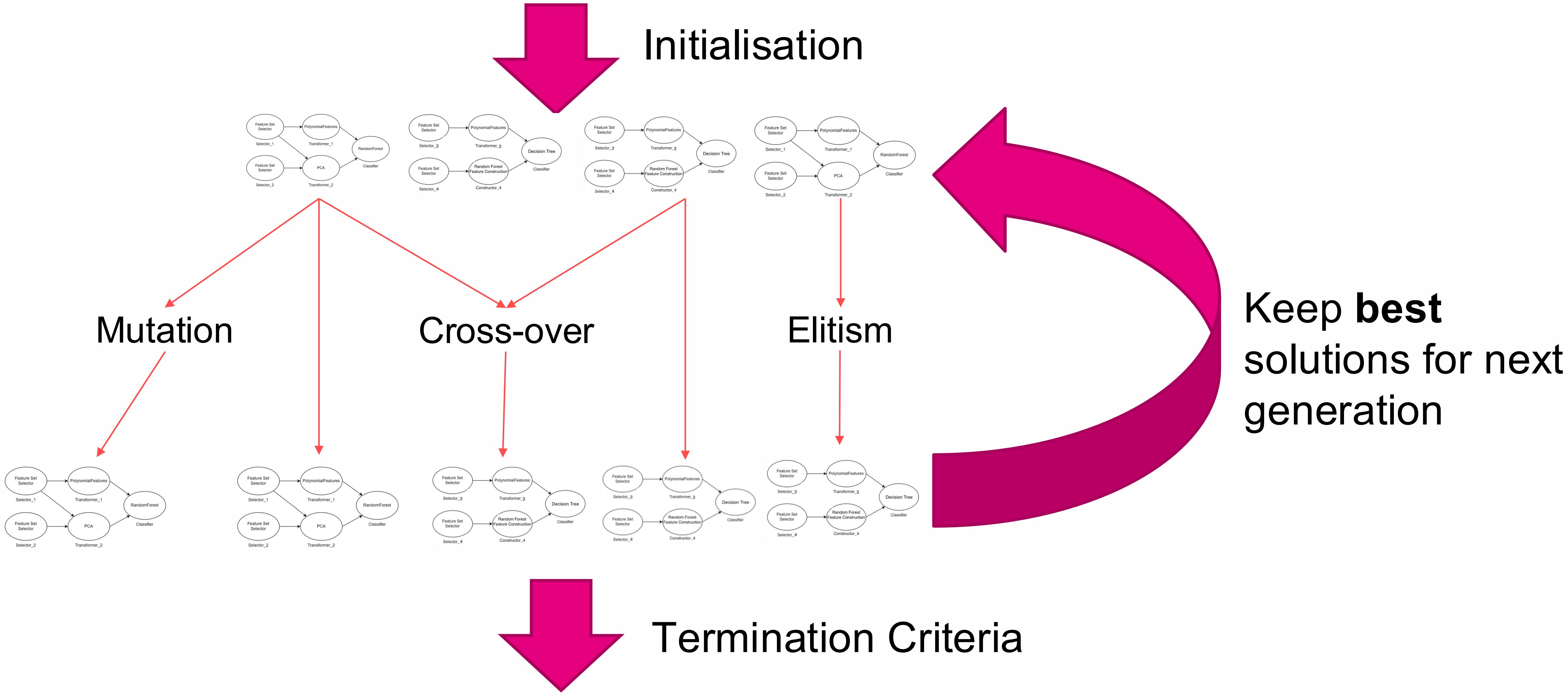
[1] Ribeiro, P., Saini, A., Moran, J., Matsumoto, N., Choi, H., Hernandez, M. and Moore, J.H., 2024. TPOT2: A New Graph-Based Implementation of the Tree-Based Pipeline Optimization Tool for Automated Machine Learning. In *Genetic programming theory and practice XX* (pp. 1-17). Singapore: Springer Nature Singapore.

AutoML Pipeline Construction (Cont'd)

Approach 3: Directed Acyclic Graph (DAG)-Based Optimization

Using Genetic Programming

Returns the code of the pipeline



Initialisation

Mutation

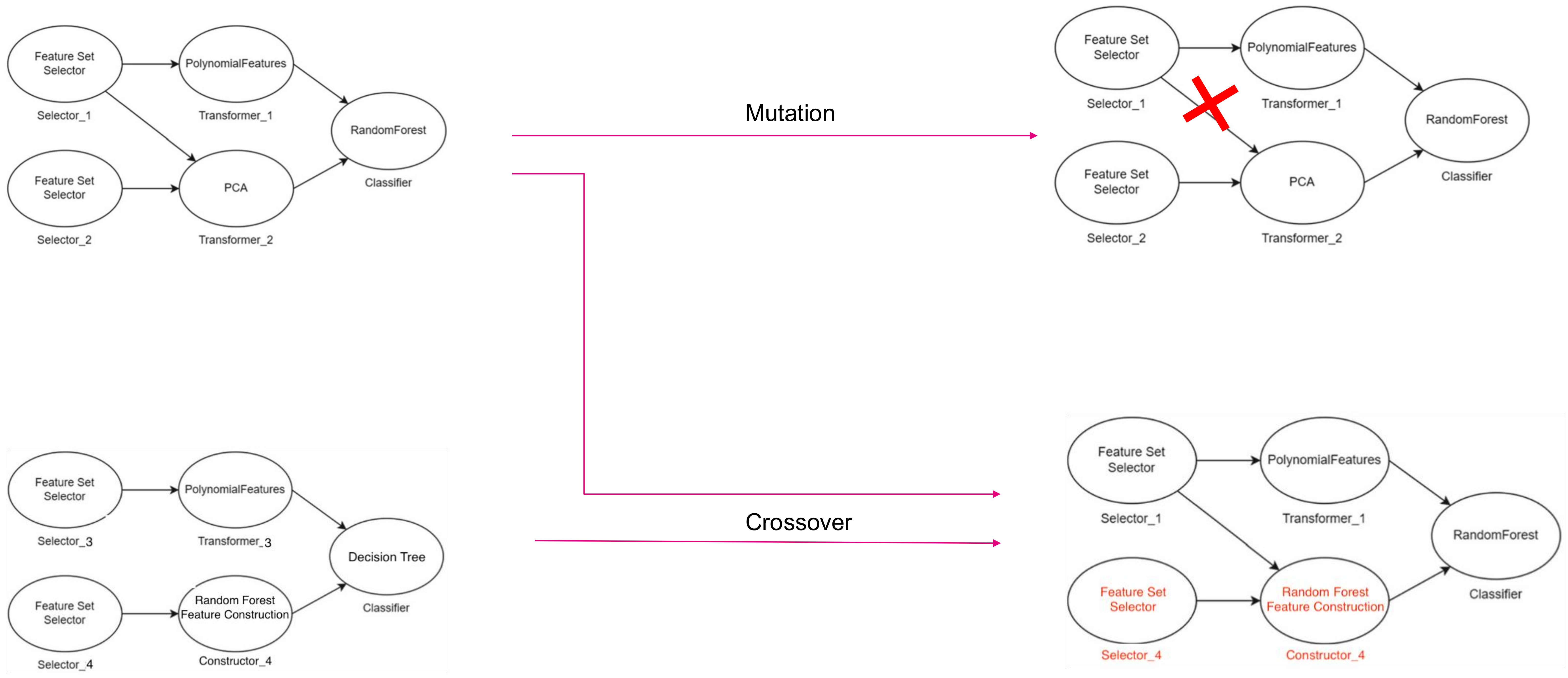
Cross-over

Elitism

Keep **best** solutions for next generation

Termination Criteria

TPOT2 Evolutionary Operators



Comparison of AutoML frameworks

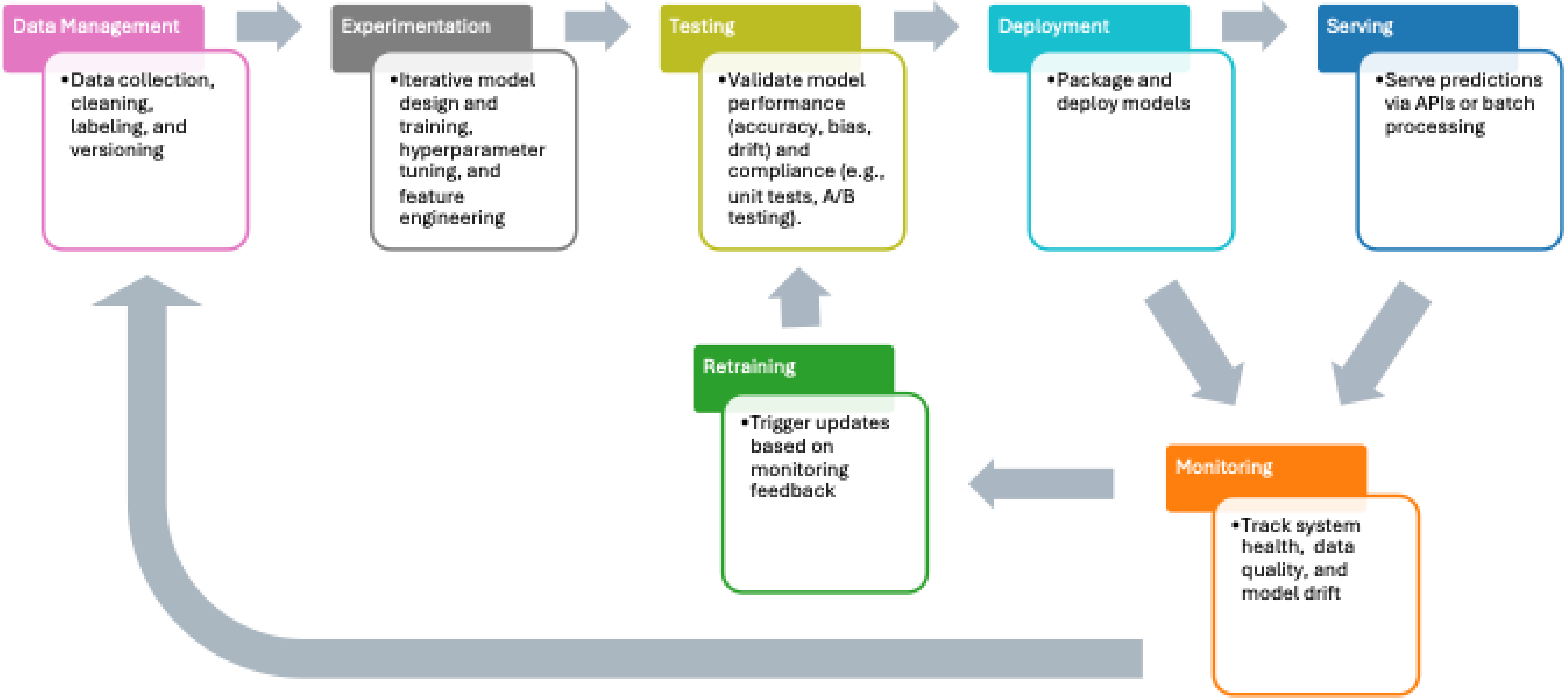
Library	Strengths	Weaknesses
AutoGluon	High performance, multi-data support, ensembling	Resource-heavy, MXNet backend
H2O AutoML	Robust, scalable, many model types	Steep learning curve, high memory use
TPOT	Customizable pipelines, exportable code	Slow, not ideal for large datasets
Auto-sklearn	Strong ML foundation, reproducible	Less deep learning support, tricky setup
MLJAR	User-friendly, explainable reports	Slightly lower performance, smaller community
PyCaret	Quick prototyping, low-code	Less control, may not reach top performance

Ollscoil na Gaillimhe
University of Galway

Phase 4 (★★★★☆): System-wide multi-objective optimization

Balancing competing objectives across the entire MLOps lifecycle

MLOps



Data Management

- Data collection, cleaning, labeling, and versioning

Experimentation

- Iterative model design and training, hyperparameter tuning, and feature engineering

Testing

- Validate model performance (accuracy, bias, drift) and compliance (e.g., unit tests, A/B testing).

Deployment

- Package and deploy models

Serving

- Serve predictions via APIs or batch processing

Retraining

- Trigger updates based on monitoring feedback

Monitoring

- Track system health, data quality, and model drift

Multi-Objective Problem Formulation

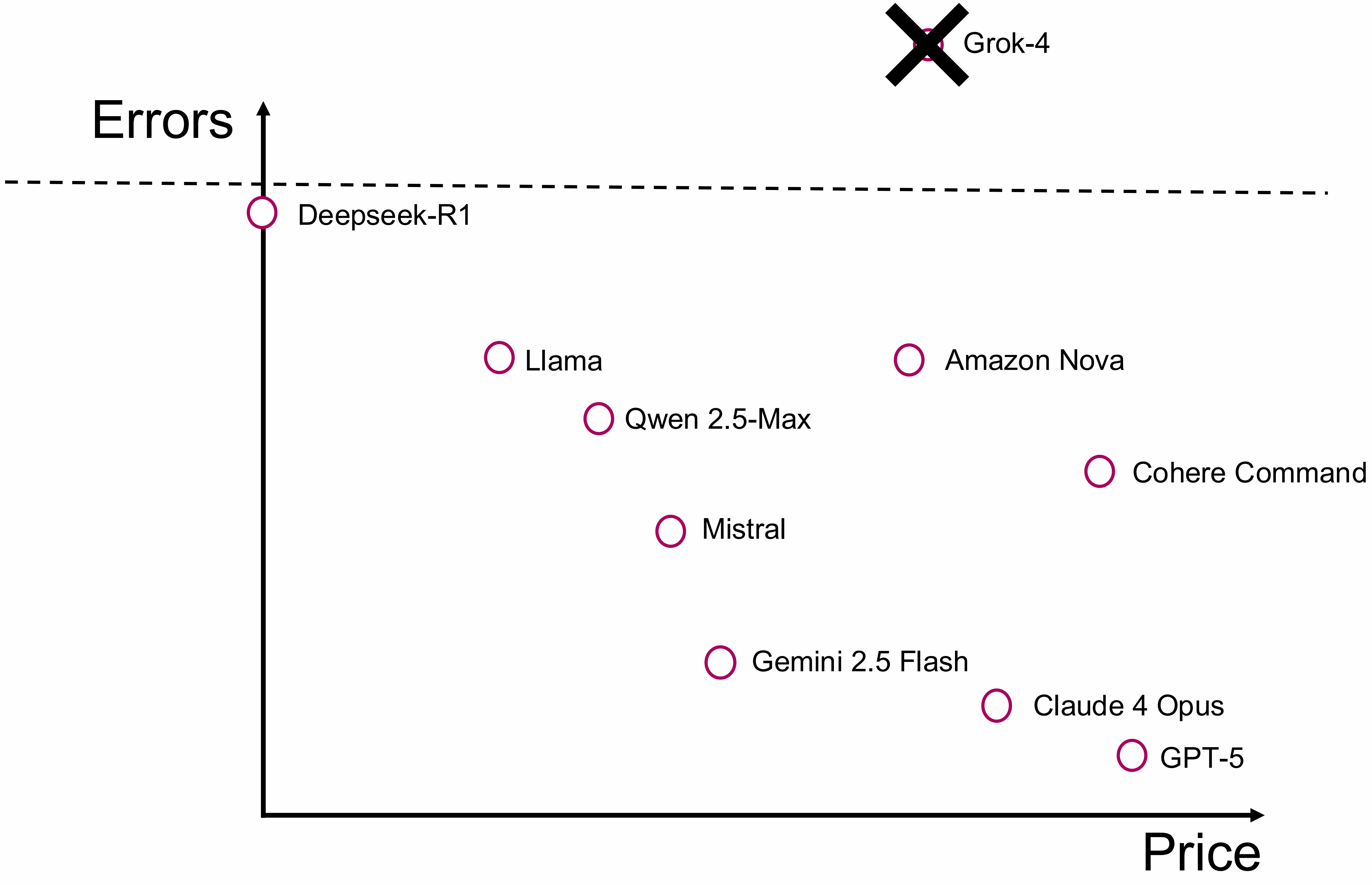
Competing Objectives:

- **Minimize**: Training cost, inference latency, model size, carbon footprint
- **Maximize**: Model accuracy, throughput, model freshness, system reliability

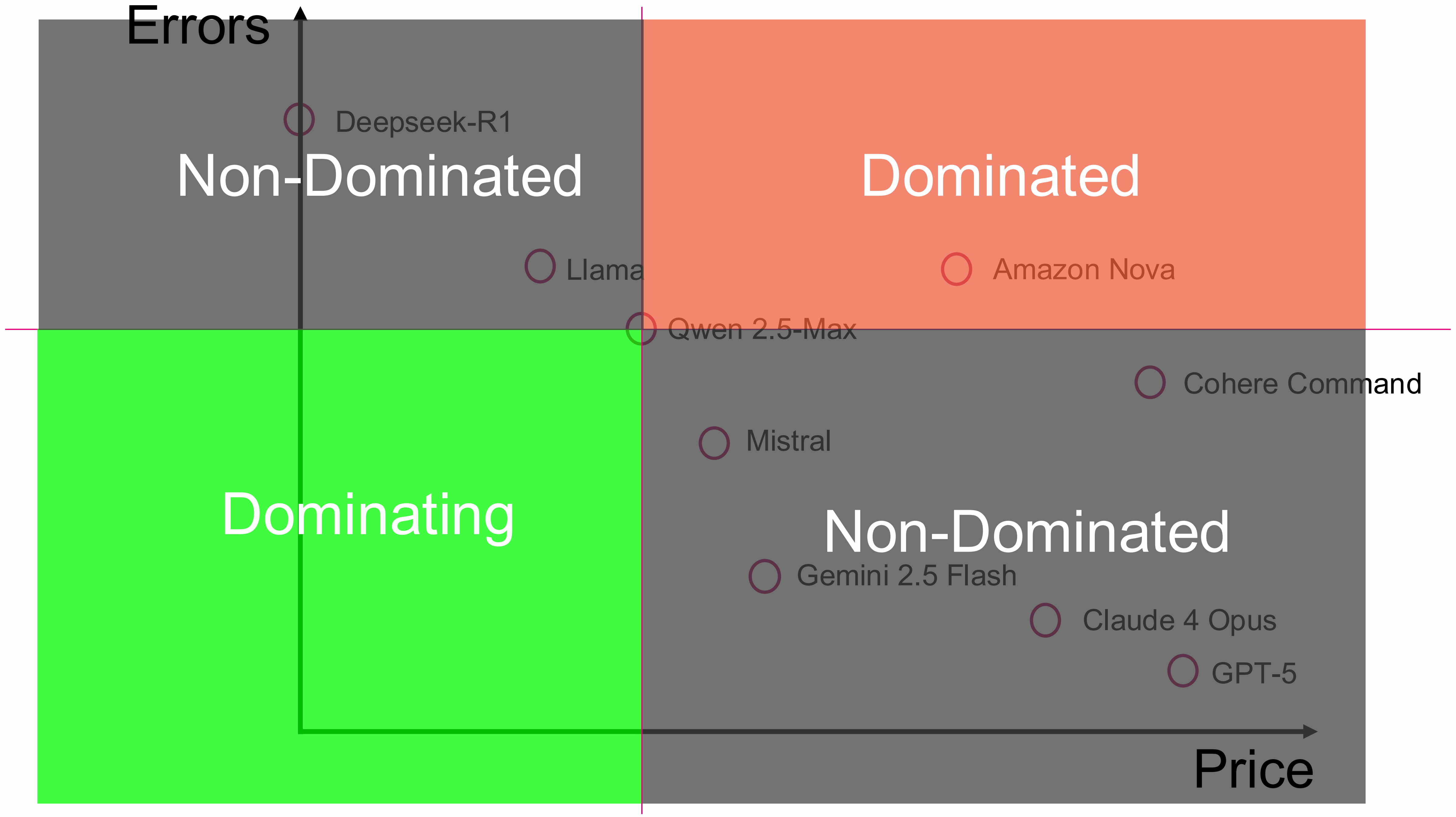
Challenge: No single optimal solution, only trade-offs

Two Objectives:

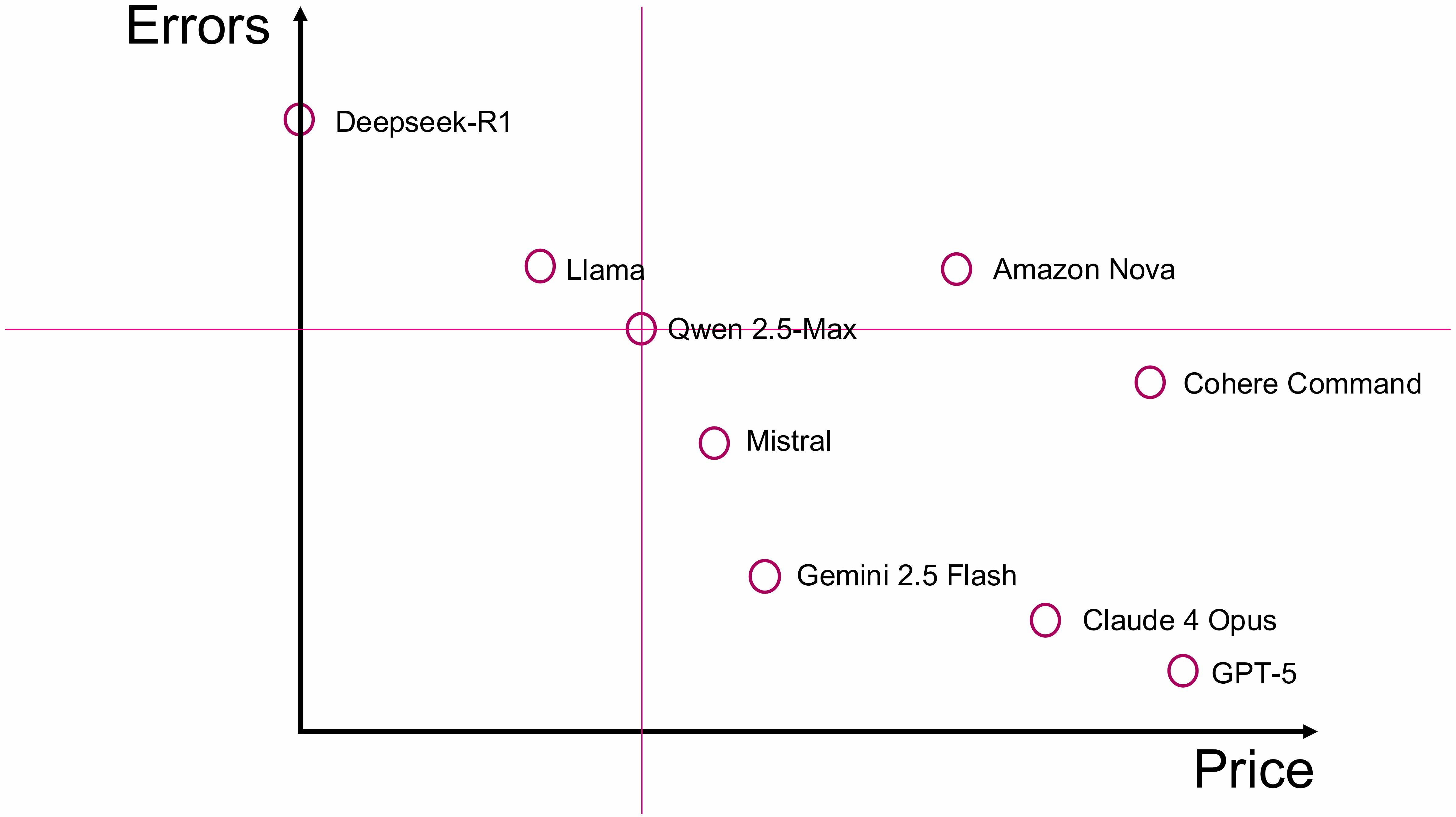
Cheap and Environmentally Friendly



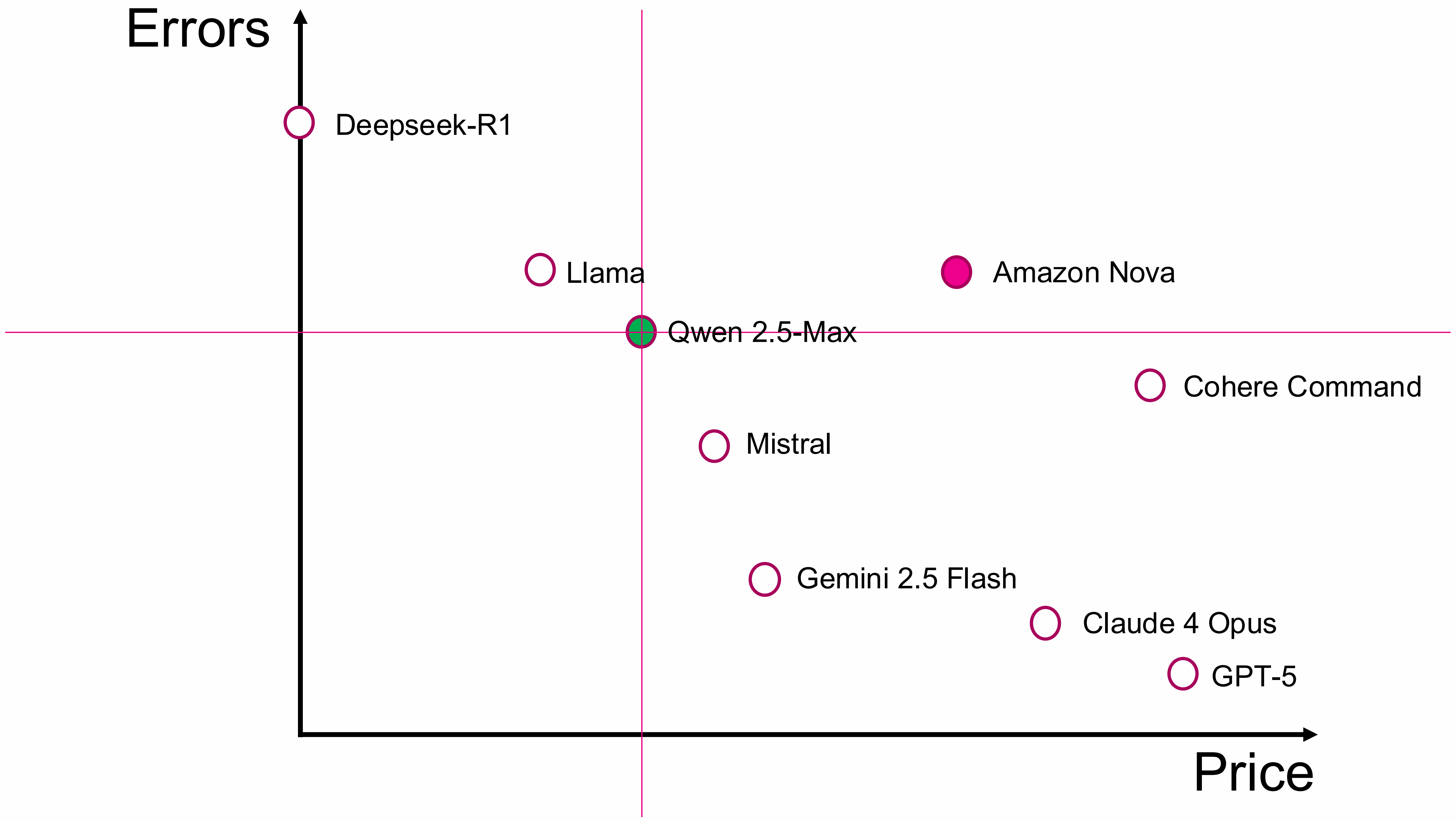
Dominance


Errors
Deepseek-R1
Non-Dominated
Dominated
Llama
Amazon Nova
Qwen 2.5-Max
Cohere Command
Mistral
Dominating
Non-Dominated
Gemini 2.5 Flash
Claude 4 Opus
GPT-5
Price

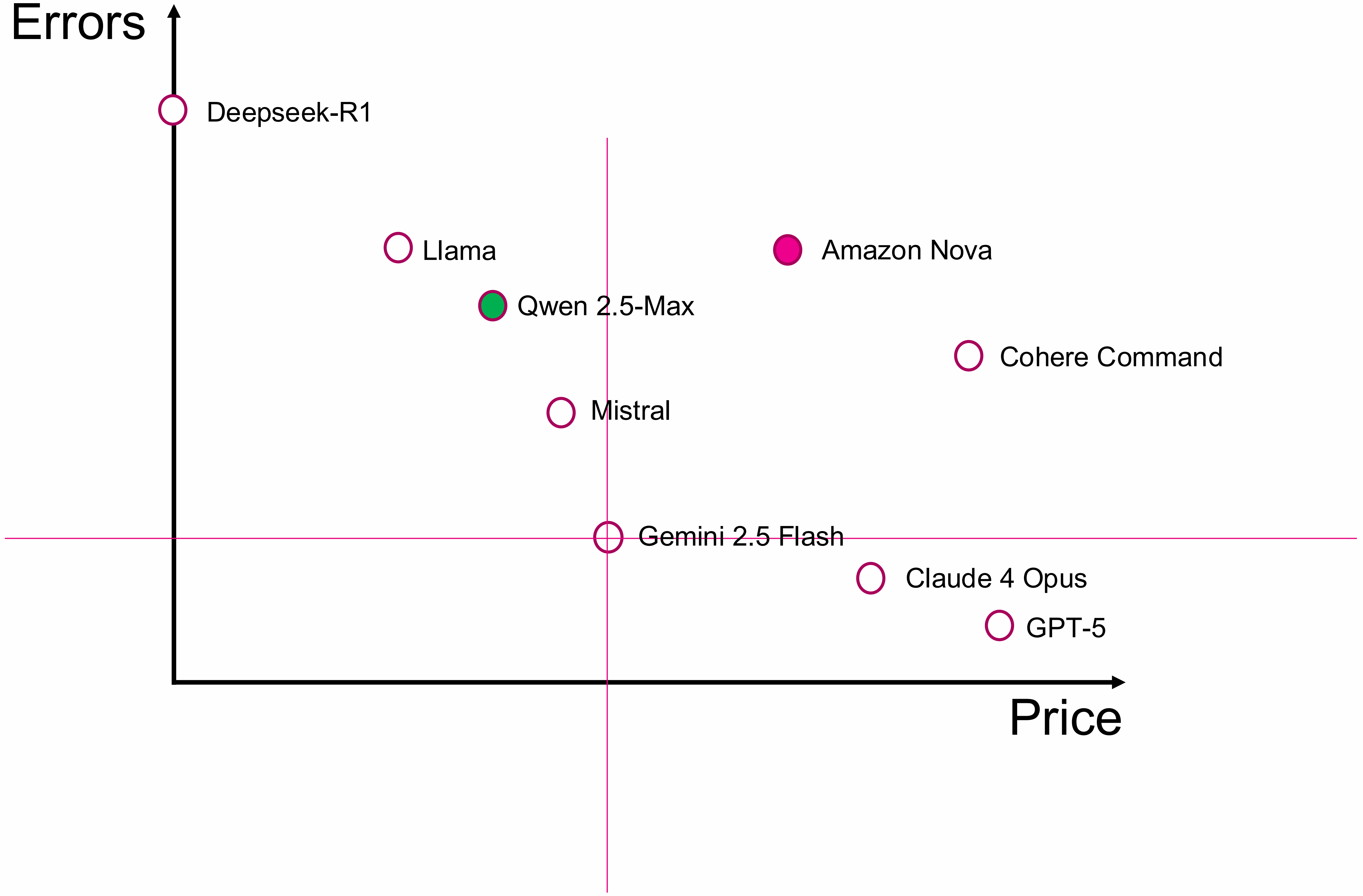
Dominance


Errors
Deepseek-R1
Llama
Amazon Nova
Qwen 2.5-Max
Cohere Command
Mistral
Gemini 2.5 Flash
Claude 4 Opus
GPT-5
Price

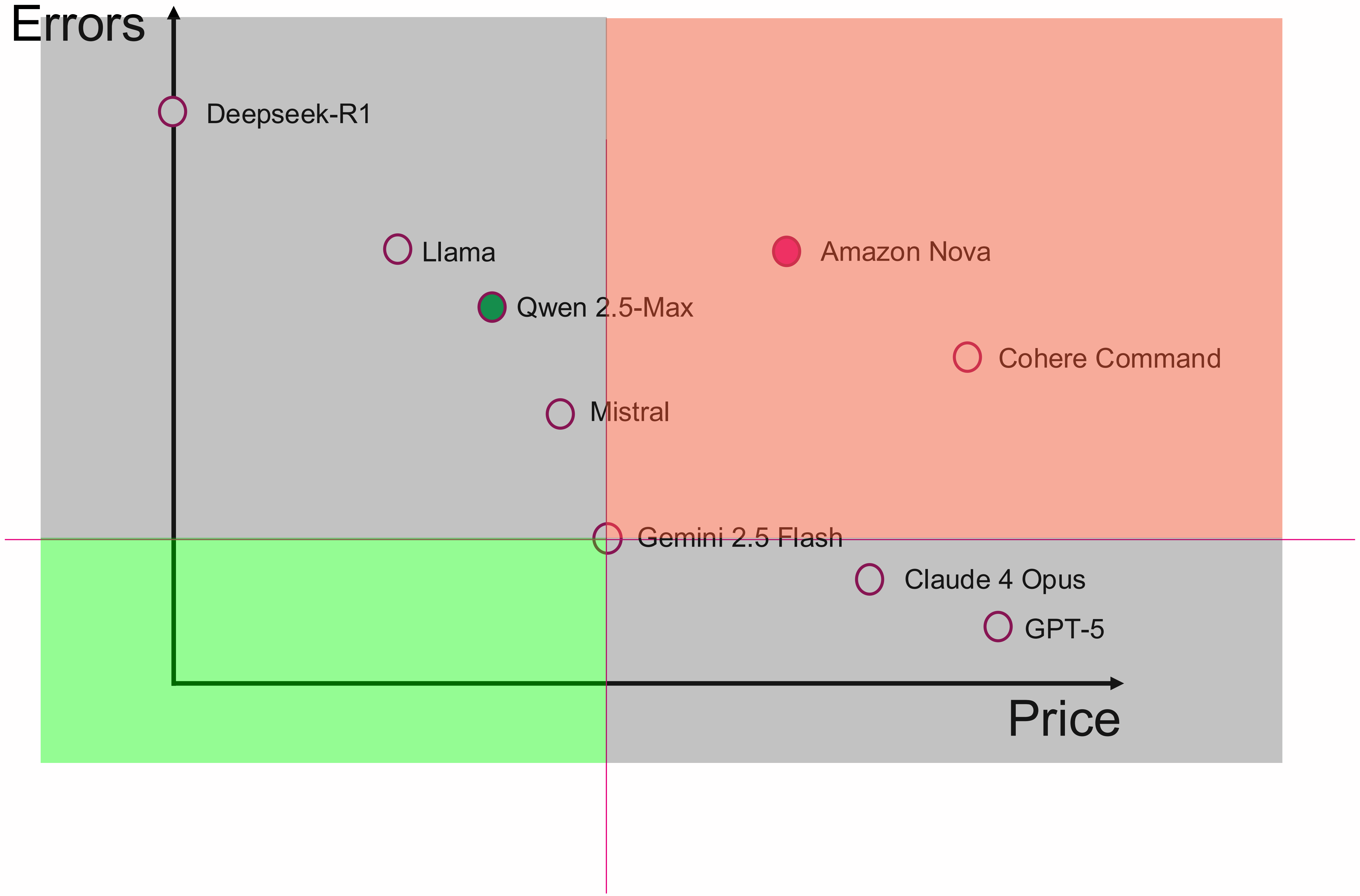
Dominance


Errors
Deepseek-R1
Llama
Amazon Nova
Qwen 2.5-Max
Cohere Command
Mistral
Gemini 2.5 Flash
Claude 4 Opus
GPT-5
Price

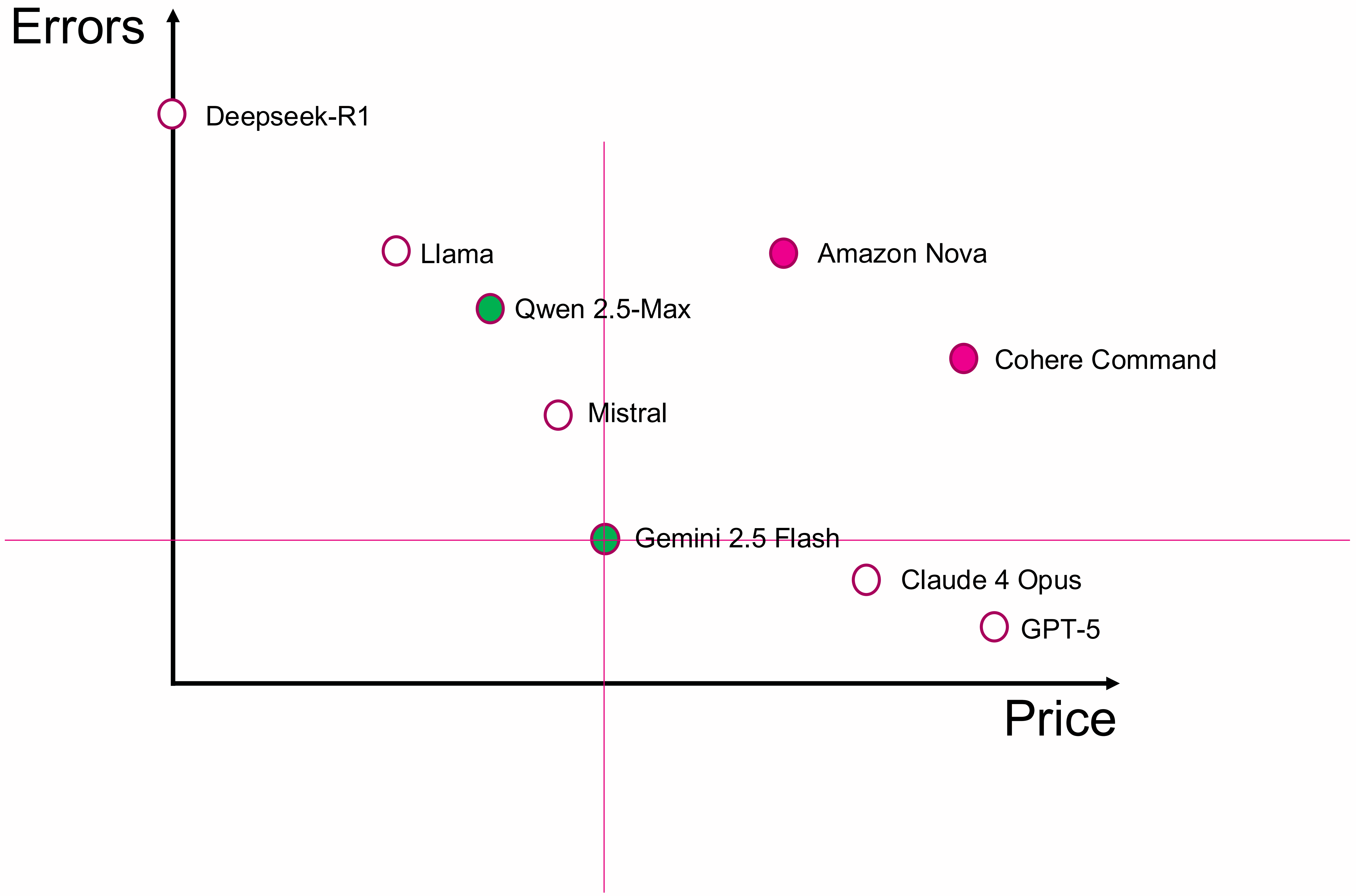
Dominance


Errors
Deepseek-R1
Llama
Amazon Nova
Qwen 2.5-Max
Cohere Command
Mistral
Gemini 2.5 Flash
Claude 4 Opus
GPT-5
Price

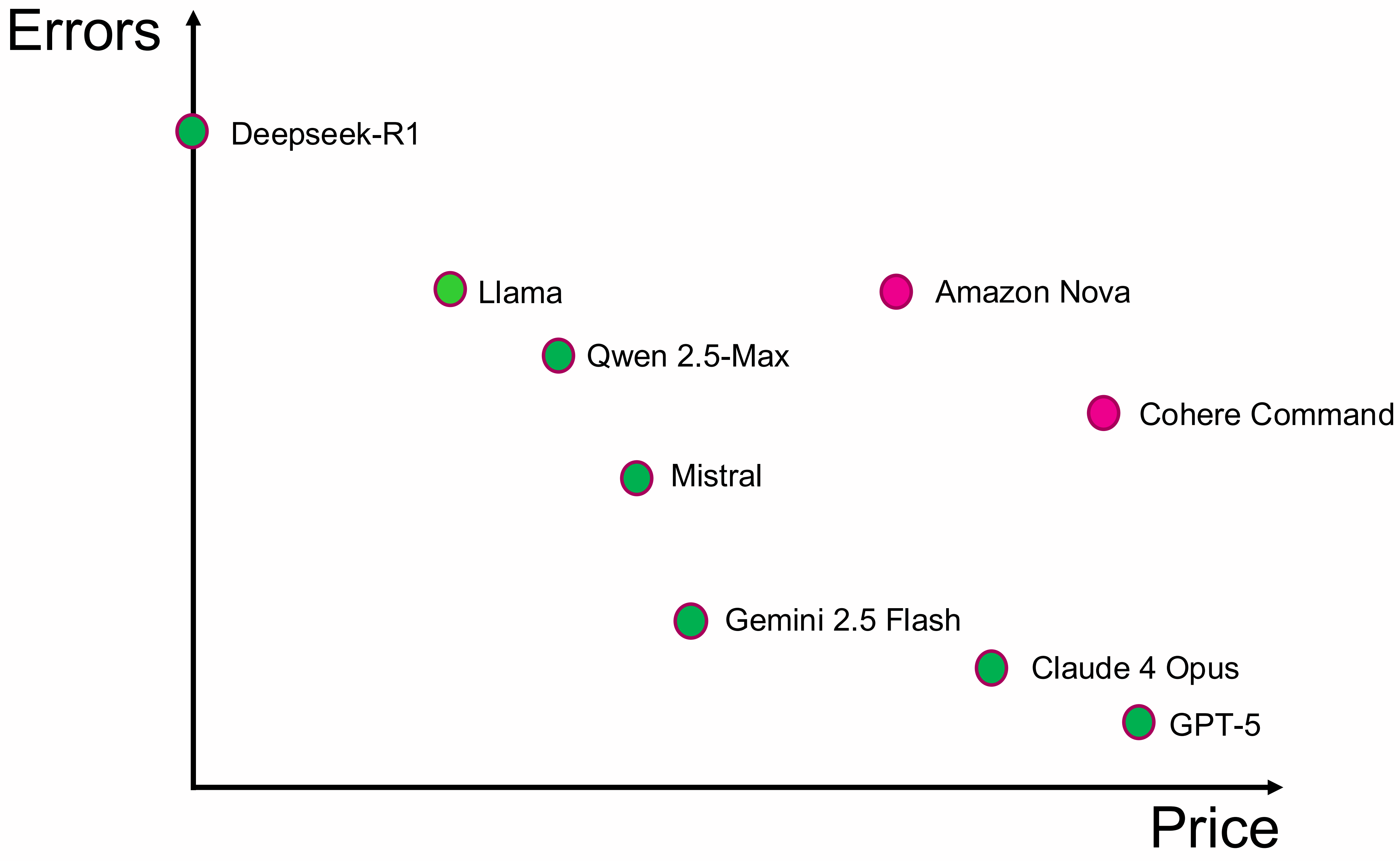
Dominance


Errors
Deepseek-R1
Llama
Qwen 2.5-Max
Mistral
Amazon Nova
Cohere Command
Gemini 2.5 Flash
Claude 4 Opus
GPT-5
Price

Dominance


Errors
Deepseek-R1
Llama
Amazon Nova
Qwen 2.5-Max
Cohere Command
Mistral
Gemini 2.5 Flash
Claude 4 Opus
GPT-5
Price

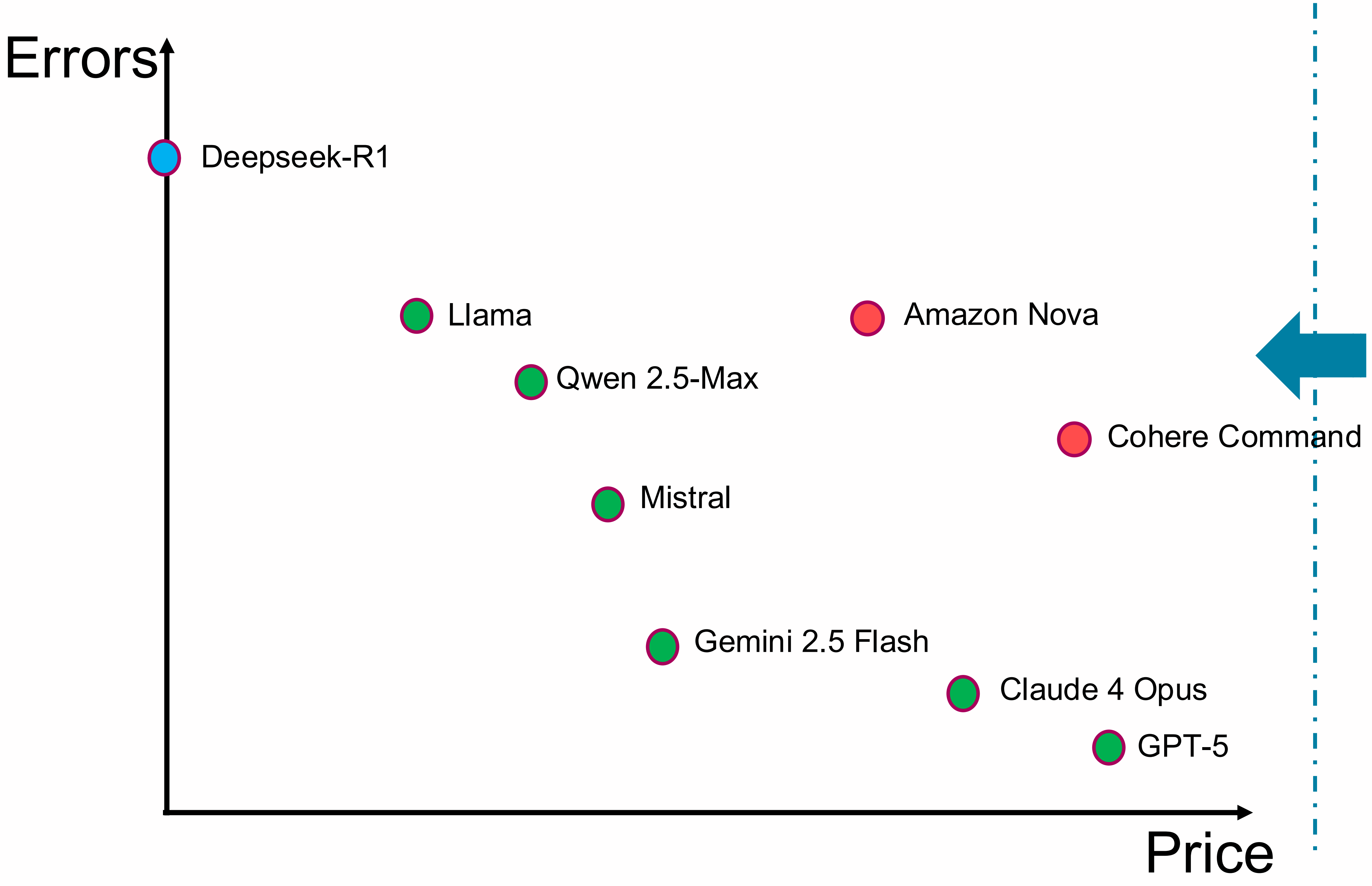
Non-Dominated Solutions (a.k.a., Pareto Front)



Pick One Objective

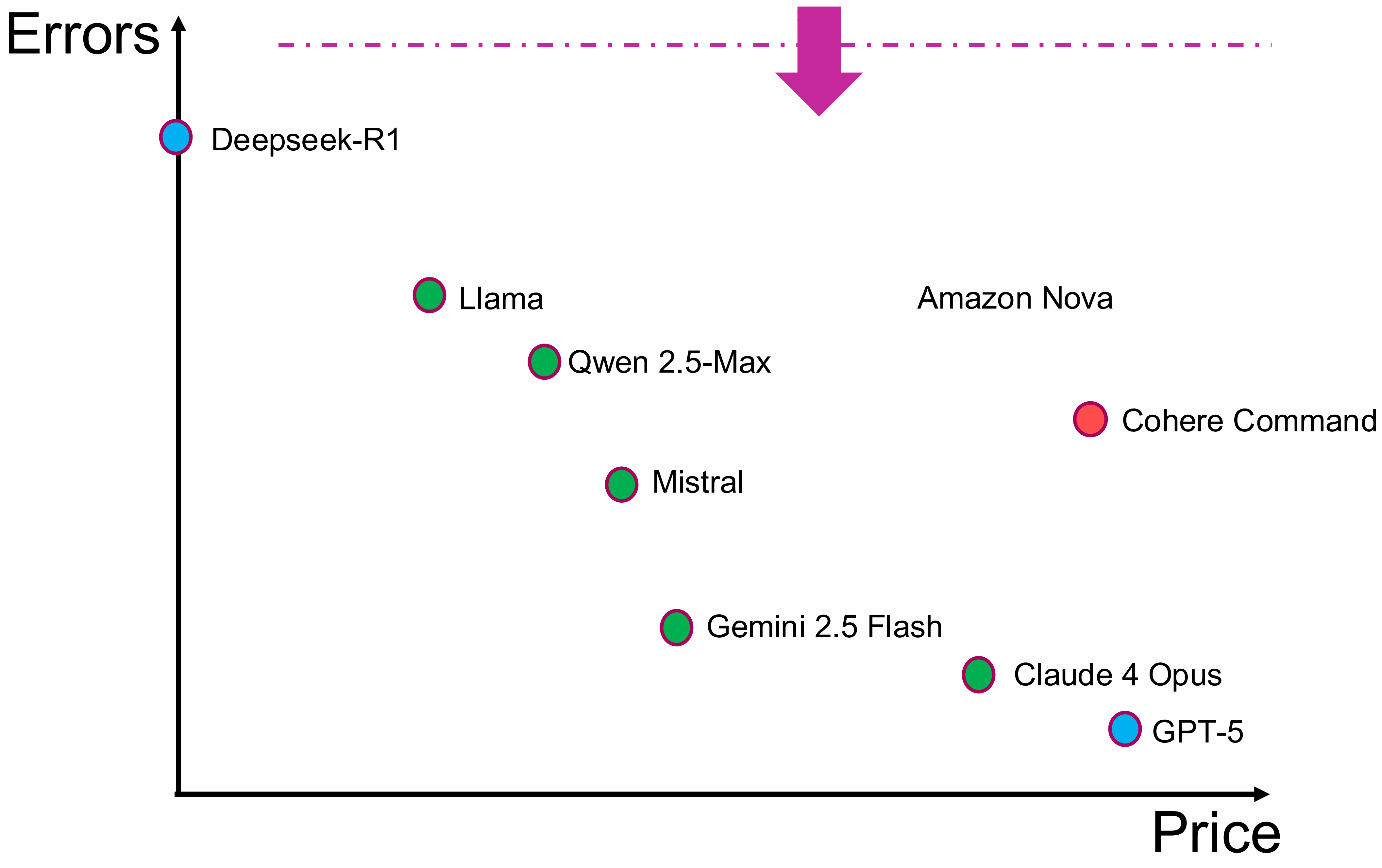
Ollscoil na Gaillimhe
University of Galway

Minimize f = Price



Pick One Objective

Minimize f = Errors



Looking for a Trade-Off

Using a Weighted-Sum!

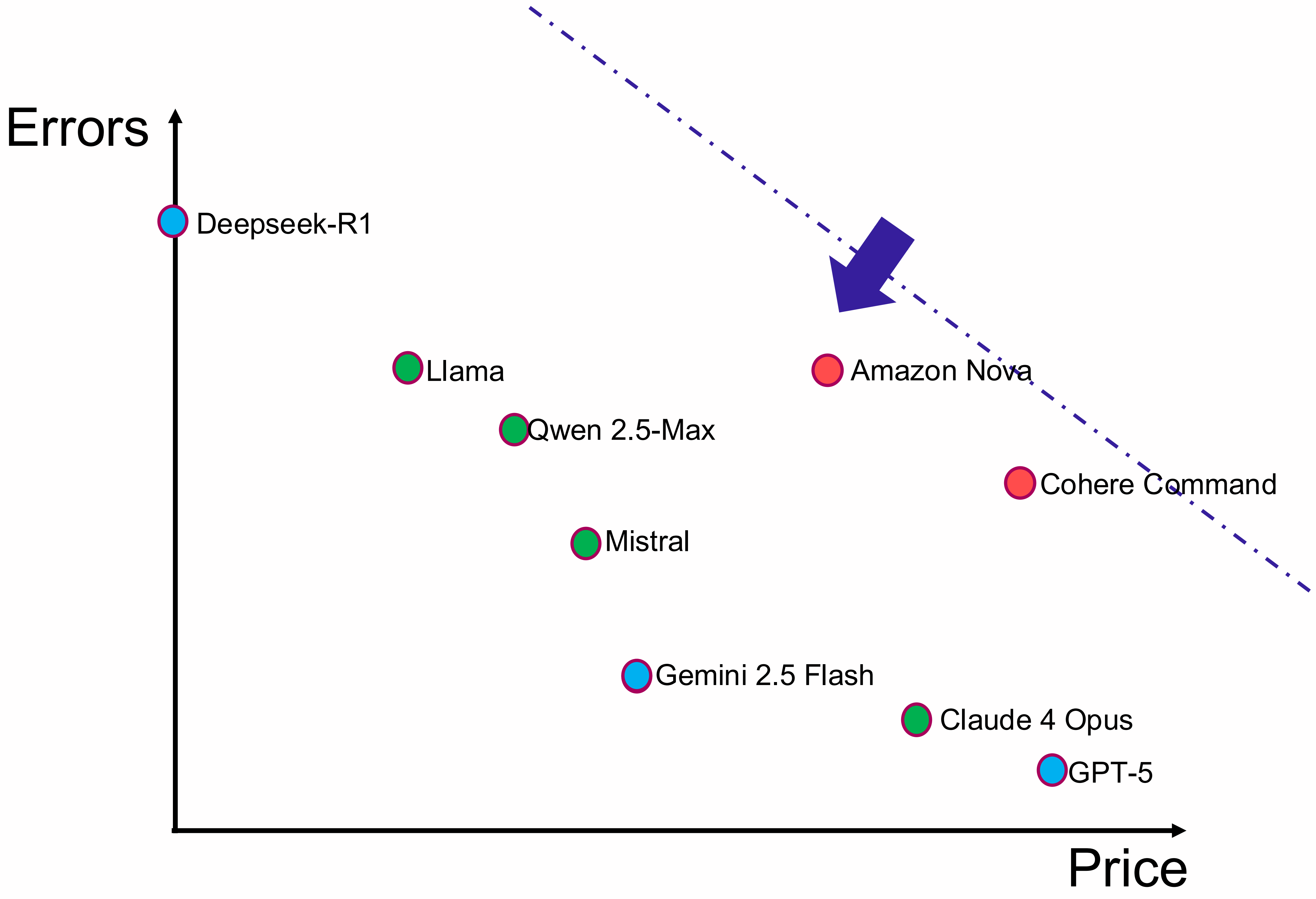
Minimize:

$$\sum_{i=0}^{\#Objectives} W_i \,.\, Objective_i$$

But, what weights to select?

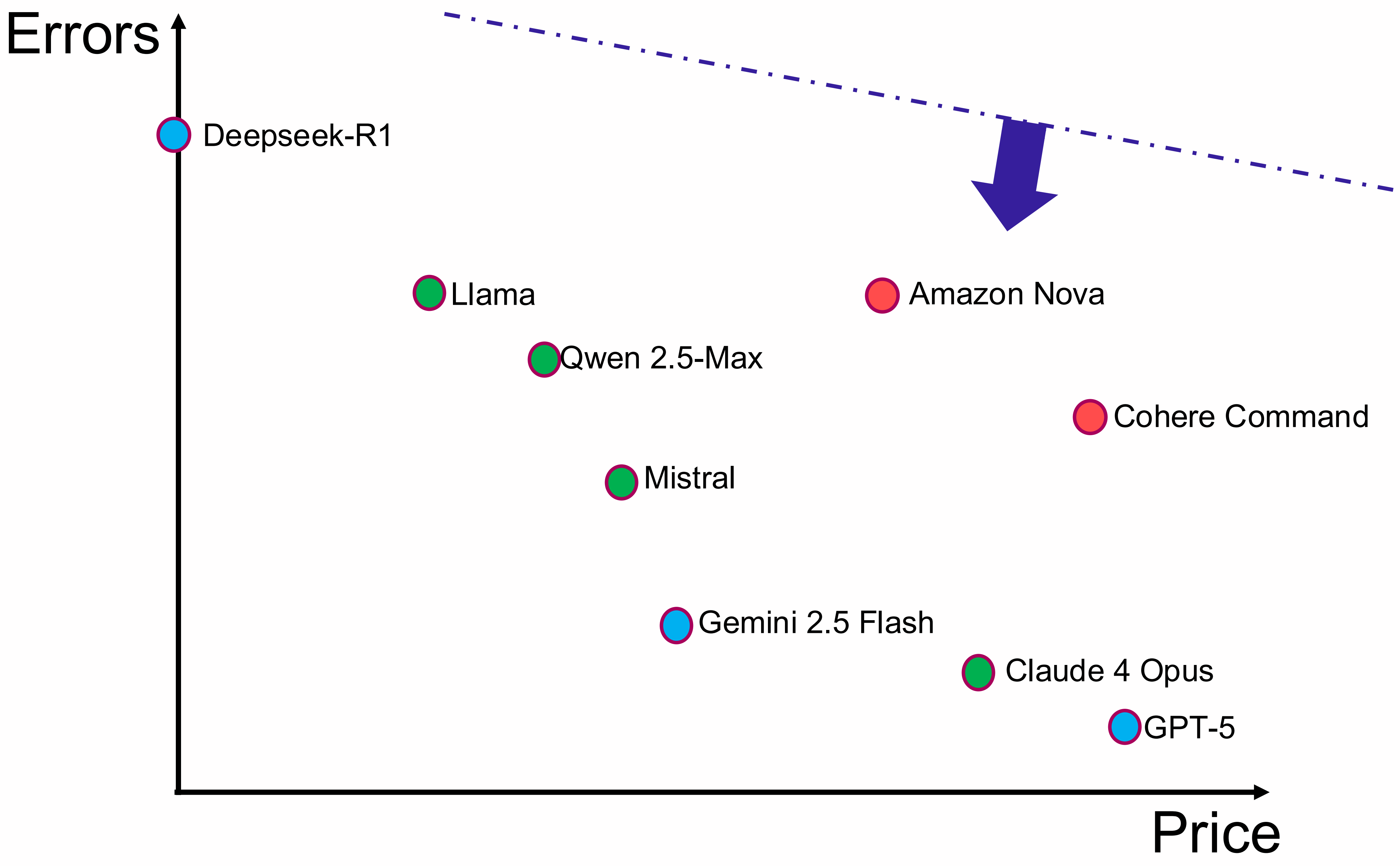
Visual Optimization of Weighted-Sum

Minimize f = **50%** Errors + **50%** Price



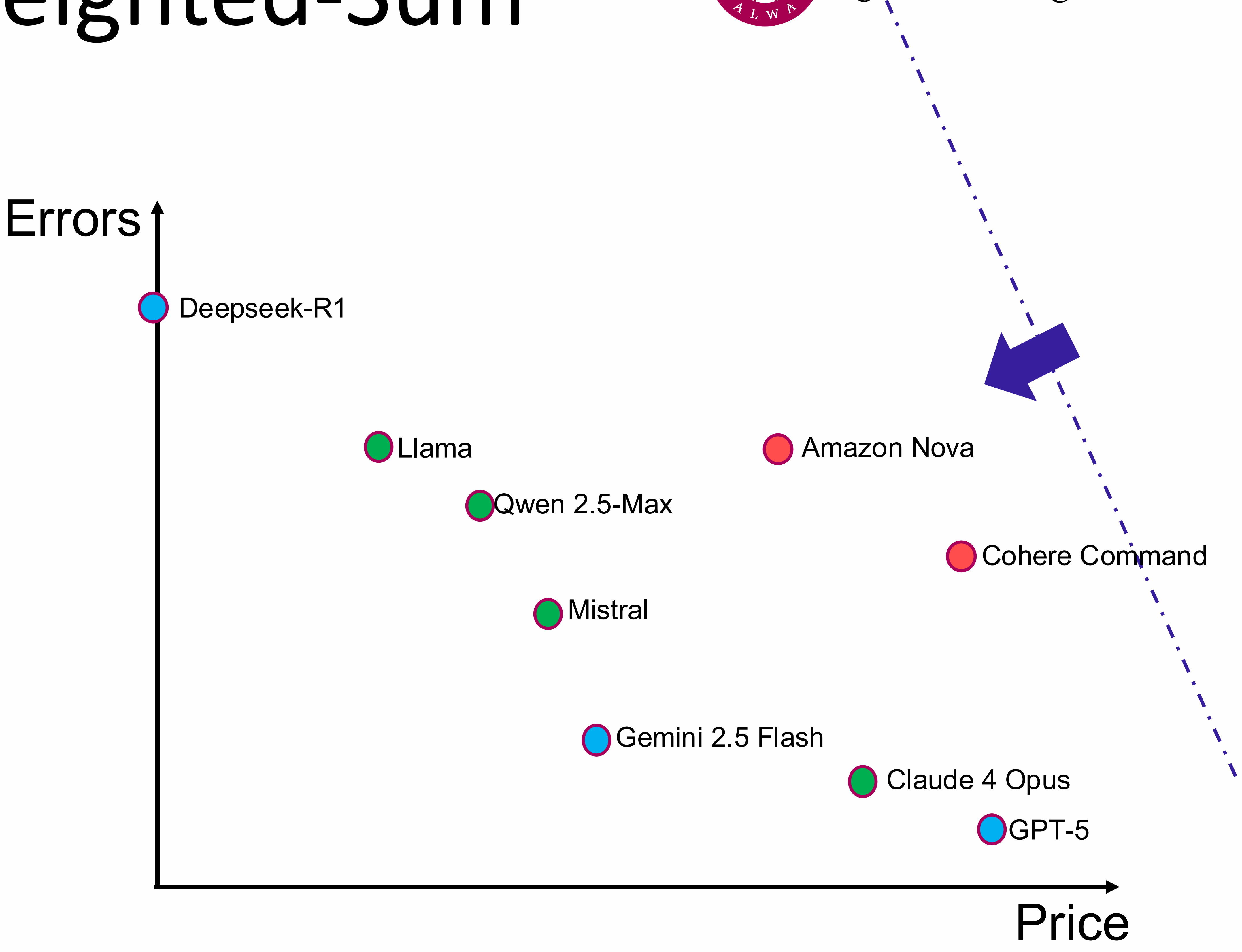
Visual Optimization of Weighted-Sum

Minimize f = **25%** Errors + **75%** Price



Visual Optimization of Weighted-Sum

Minimize f = **75%** Errors + **25%** Price



Visual Optimization of Weighted-Sum

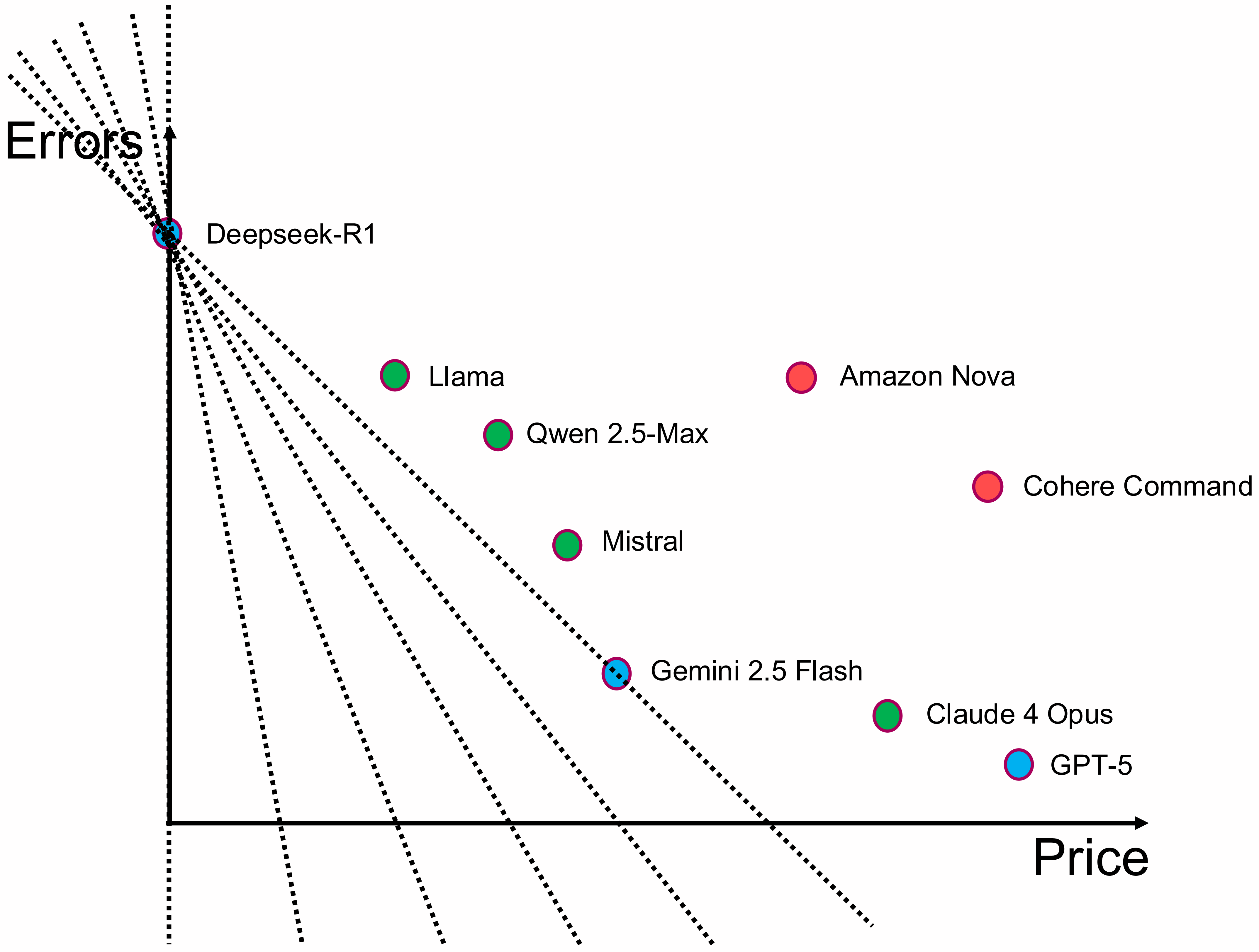
Minimize f = **0%** Errors + **100%** Price

Minimize f = **25%** Errors + **75%** Price

Minimize f = **50%** Errors + **50%** Price

…

Minimize f = **100%** Errors + **0%** Price



Visual Optimization of Weighted-Sum

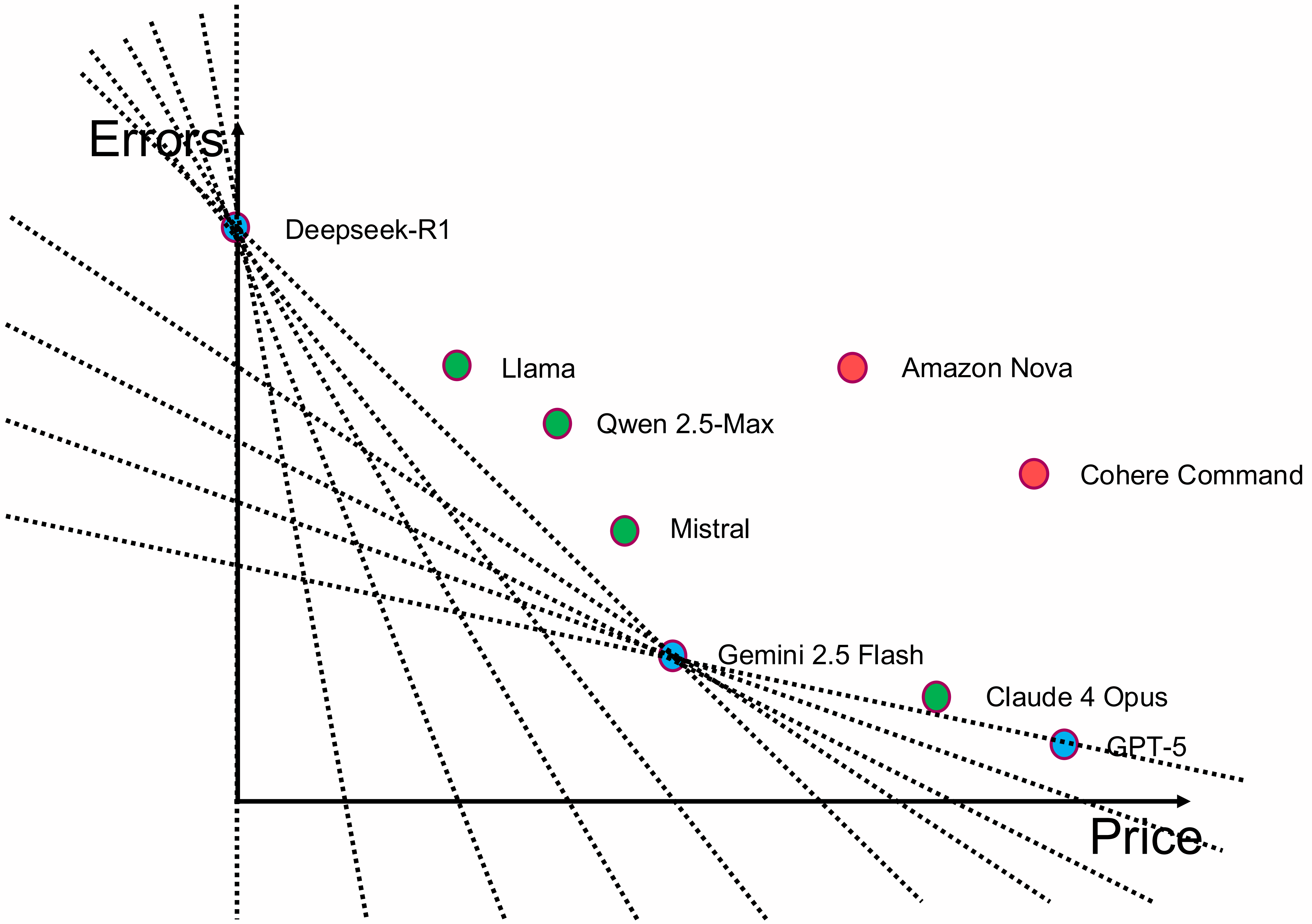
Minimize f = **0%** Errors + **100%** Price

Minimize f = **25%** Errors + **75%** Price

Minimize f = **50%** Errors + **50%** Price

. . .

Minimize f = **100%** Errors + **0%** Price



Visual Optimization of Weighted-Sum

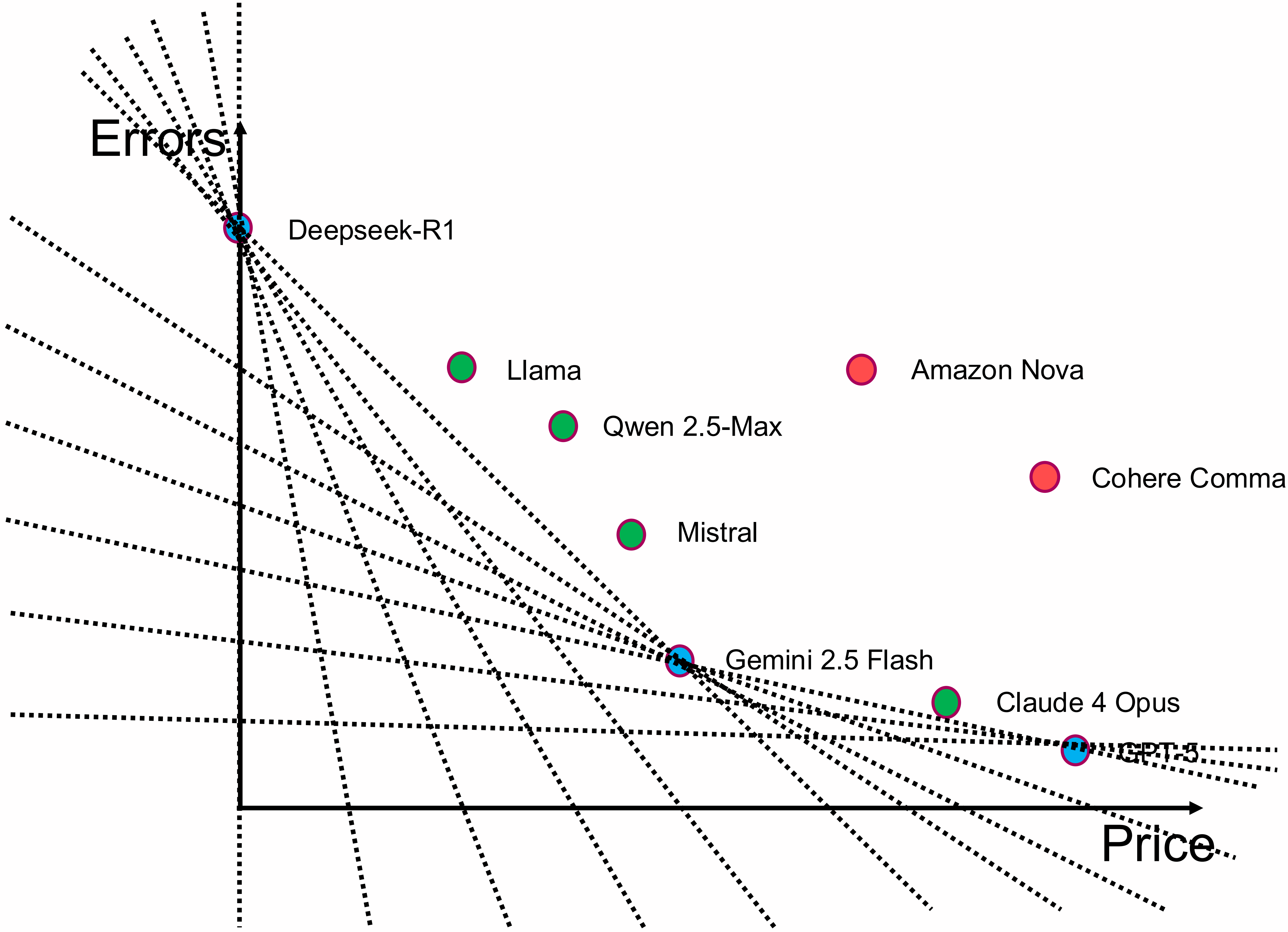
Minimize f = **0%** Errors + **100%** Price

Minimize f = **25%** Errors + **75%** Price

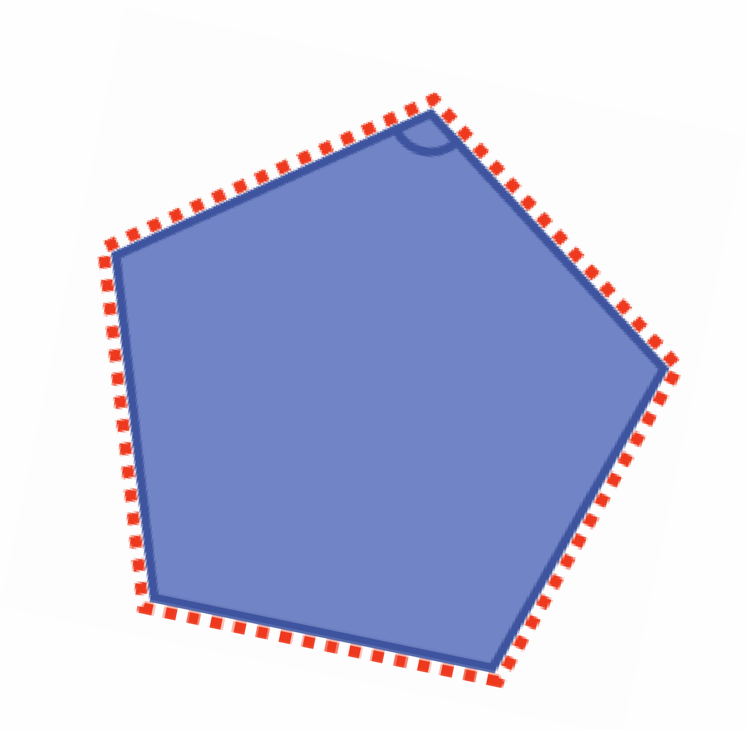
Minimize f = **50%** Errors + **50%** Price

...

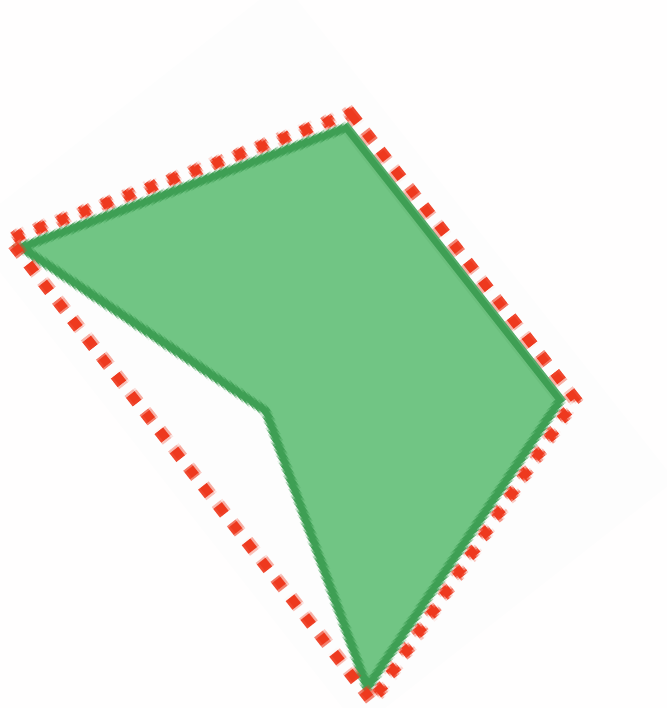
Minimize f = **100%** Errors + **0%** Price



Convex *vs.* Concave



Convex

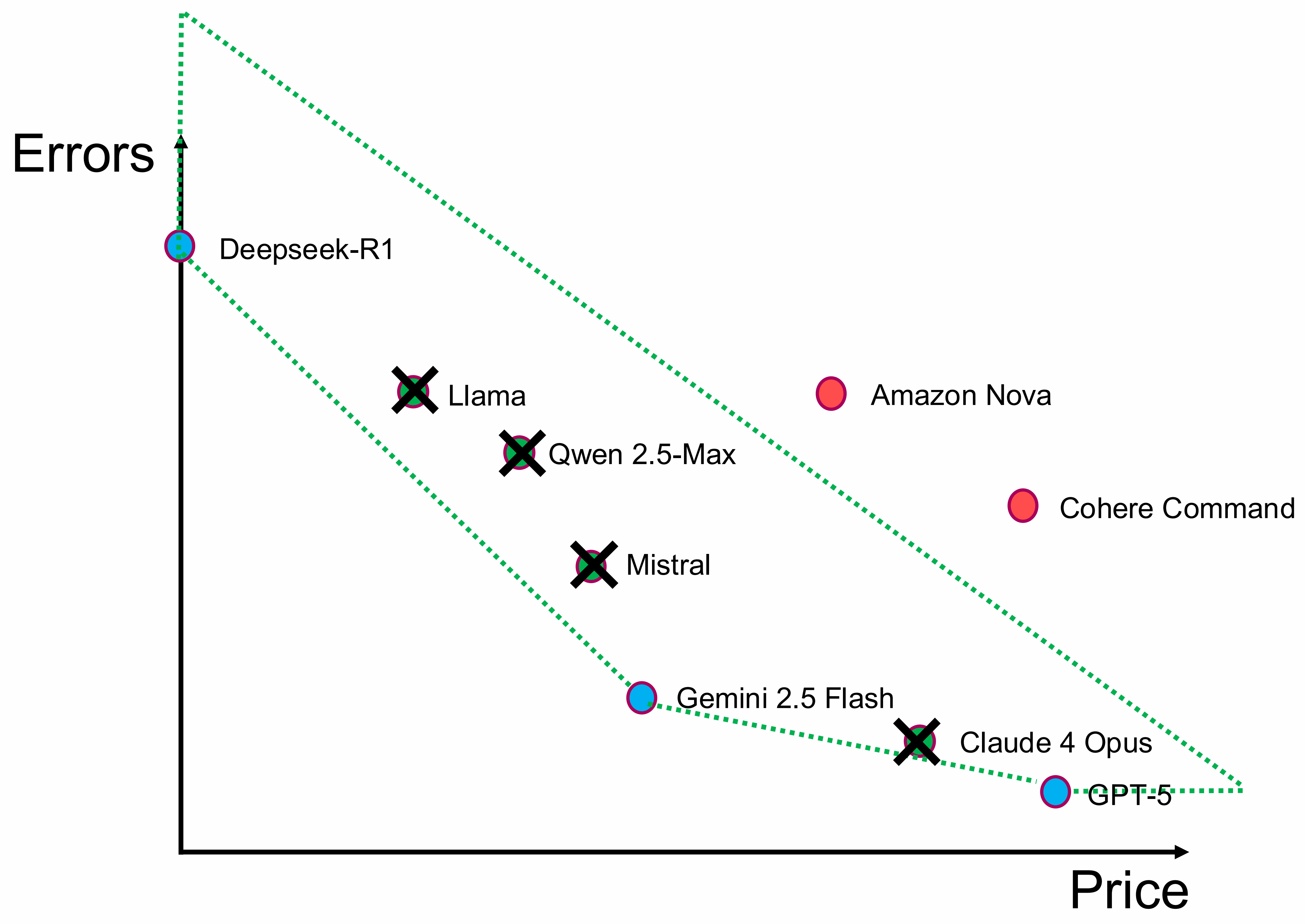


Concave

Rule:

- Weighted-Sum will only allow you to find non-dominated (best) solutions that are in the Convex Hull

Visual Optimization of Weighted-Sum



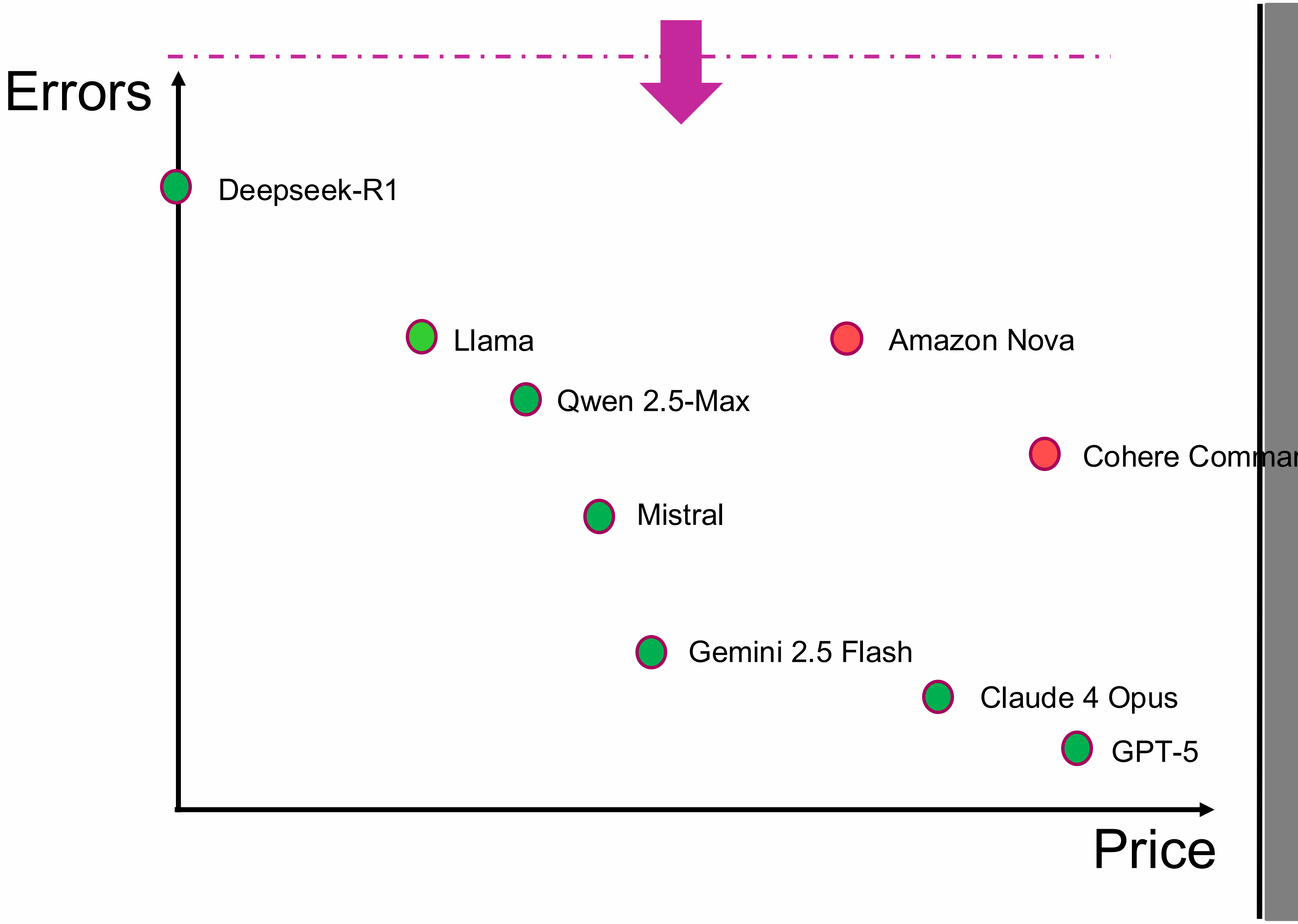
What If I still want All Non-Dominated Solutions?

What if I Still Only Want All Non-dominated Solutions?

Better use **ε**-Constraints

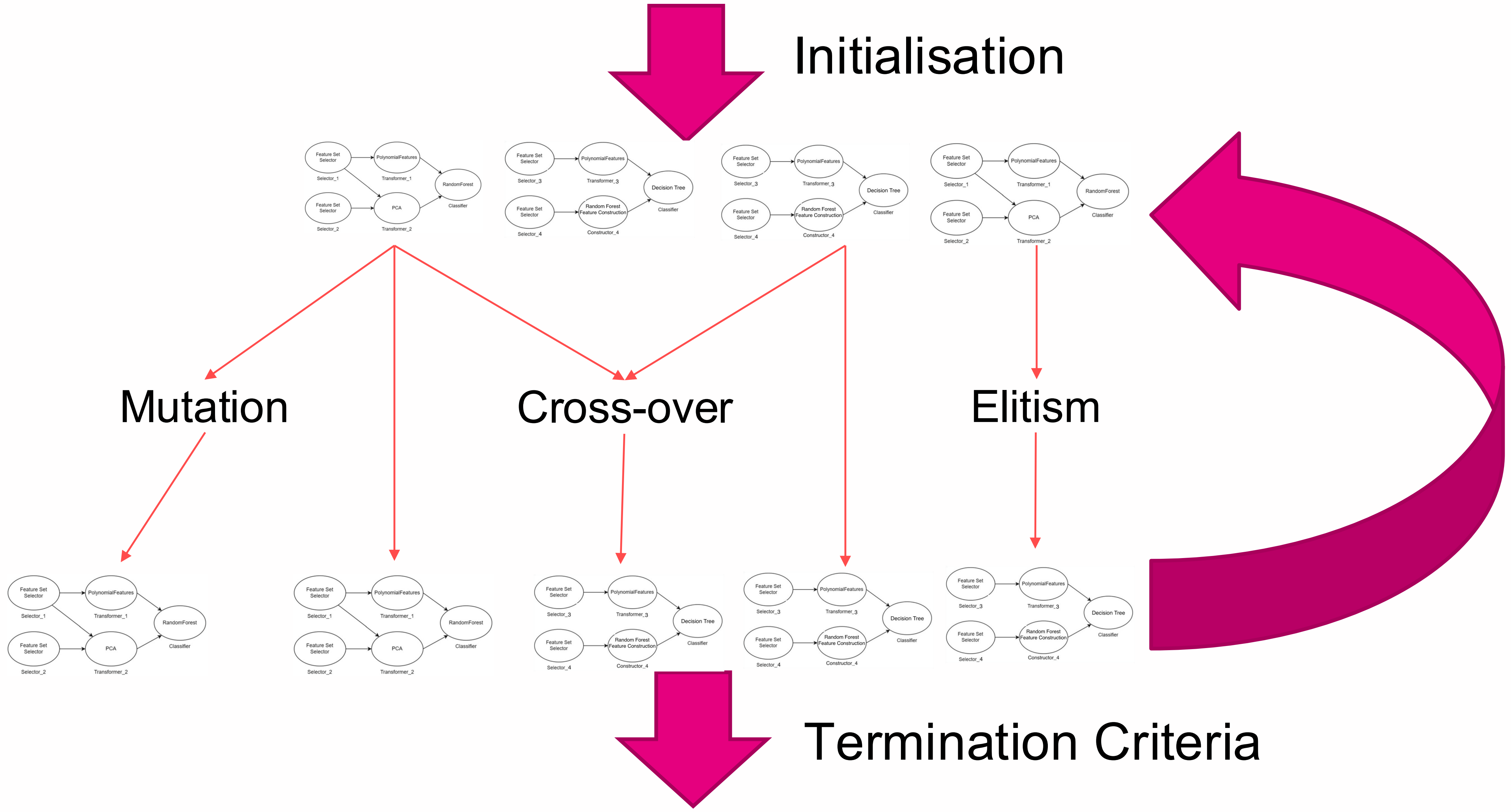
Repeat with various values for W

Minimize: Errors
Subject to:
Price < W



W=100

Multi-Objective Optimization with TPOT



Initialisation

Mutation

Cross-over

Elitism

Termination Criteria

non-dominated and most spread

Keep ~~best~~ solutions for next generation

OLLSCOIL NA GAILLIMHE
UNIVERSITY OF GALWAY

Phase 5 (★★★★★): Advanced topics

Self-optimizing systems (Covered)
Federated MLOps

Self-Optimizing MLOps Systems

Vision: **MLOps Holy Grail**

- Learns from optimization history
- System monitors itself to detect and propose optimization opportunities

Technologies:

- Meta-learning: Learning to optimize
- Neural Architecture Search for optimizers
- Automated problem formulation
- Optimization as a Service

Early Results:

- Google's AutoML Zero:
  - Automatically evolve entire ML pipelines (algorithms + architectures + optimization strategies) form scratch.
  - Minimise Human-designed bias

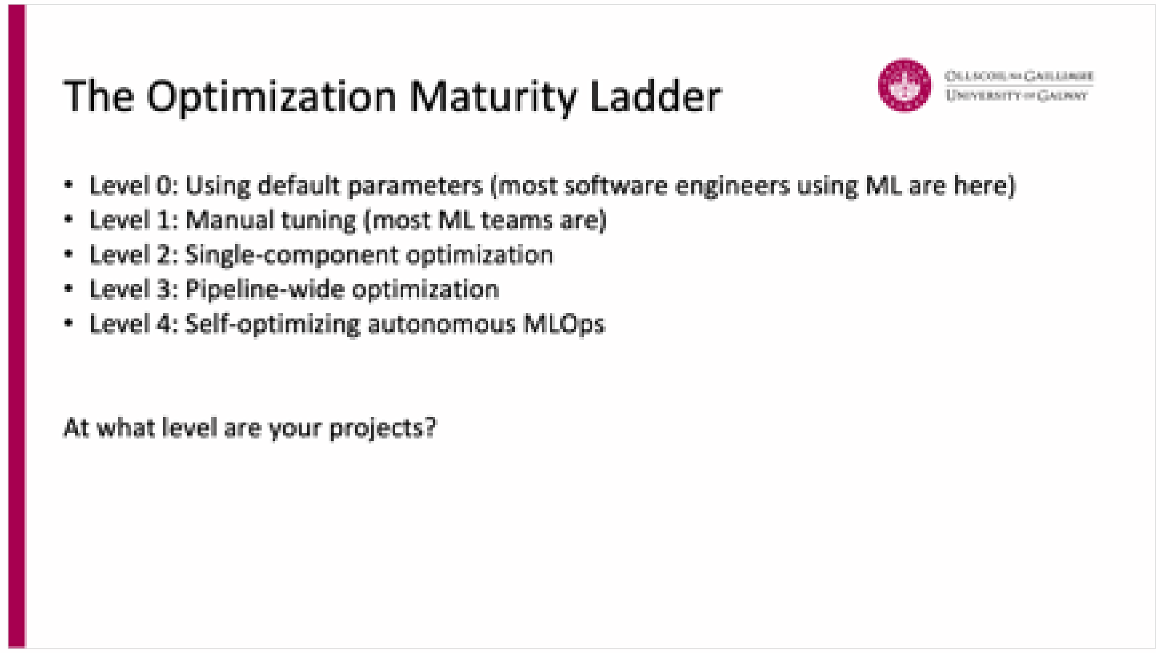
Final Takeaways

Key Messages:

- Optimization is accessible and pays off
  - Open-source tools are production-ready
  - 30-60% improvements are common
- Main advantage is not saving time, but doing things that were not possible before due to the lack of time
  - E.g., Collaboration between Data Scientist and Business Experts, Multi-Criteria Decision Analysis, etc.
- Continuous journey: Optimization is never "done"

Closing Statement: "The future of MLOps isn't just automated. It is optimized."
The question isn't whether to optimize, but how fast you can move up the maturity ladder.

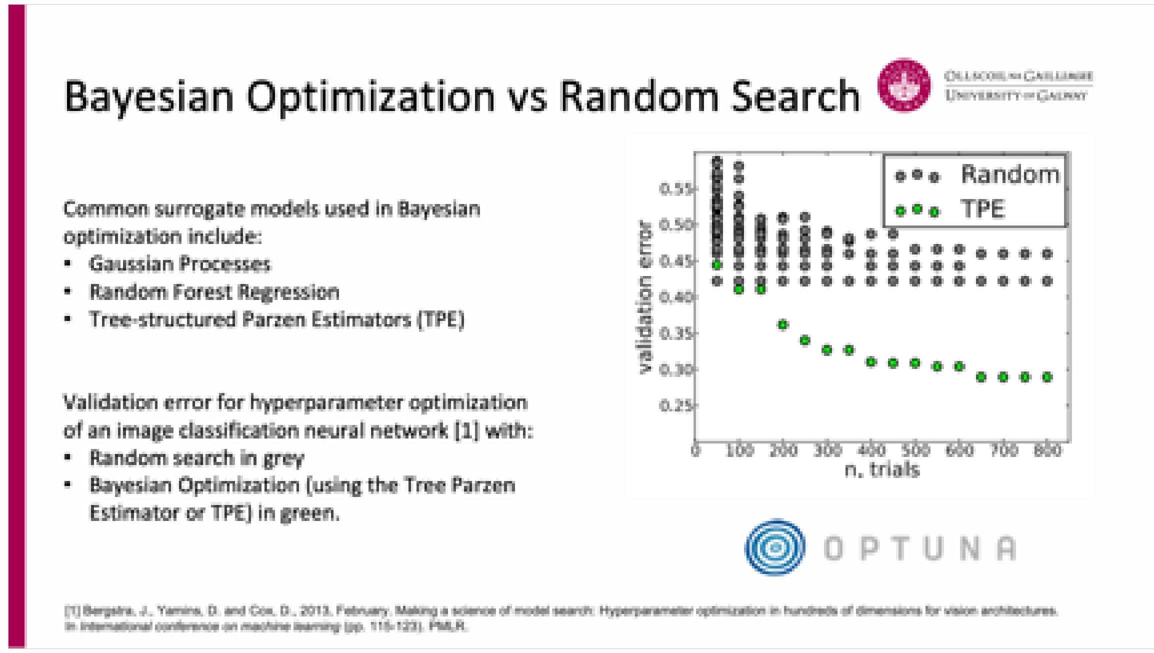
Q&A



The Optimization Maturity Ladder

- Level 0: Using default parameters (most software engineers using ML are here)
- Level 1: Manual tuning (most ML teams are)
- Level 2: Single-component optimization
- Level 3: Pipeline-wide optimization
- Level 4: Self-optimizing autonomous MLOps

At what level are your projects?



Bayesian Optimization vs Random Search

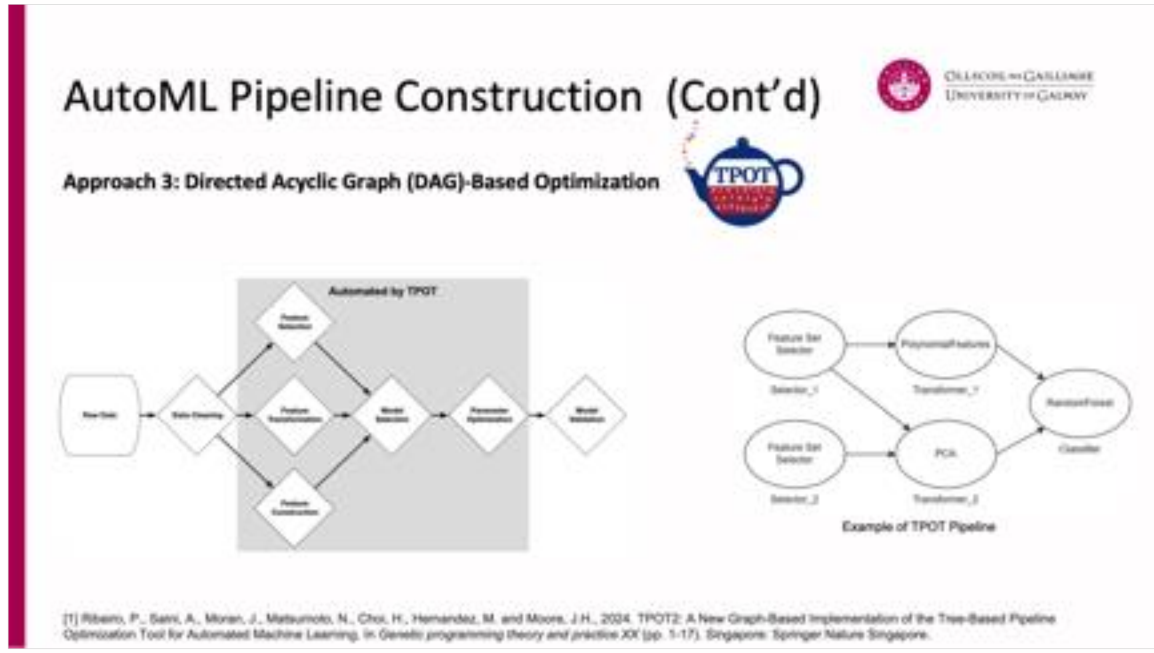
Common surrogate models used in Bayesian optimization include:

- Gaussian Processes
- Random Forest Regression
- Tree-structured Parzen Estimators (TPE)

Validation error for hyperparameter optimization of an image classification neural network [1] with:

- Random search in grey
- Bayesian Optimization (using the Tree Parzen Estimator or TPE) in green.

[1] Bergstra, J., Yamins, D. and Cox, D., 2013, February. Making a science of model search: Hyperparameter optimization in hundreds of dimensions for vision architectures. In International conference on machine learning (pp. 115-123). PMLR.

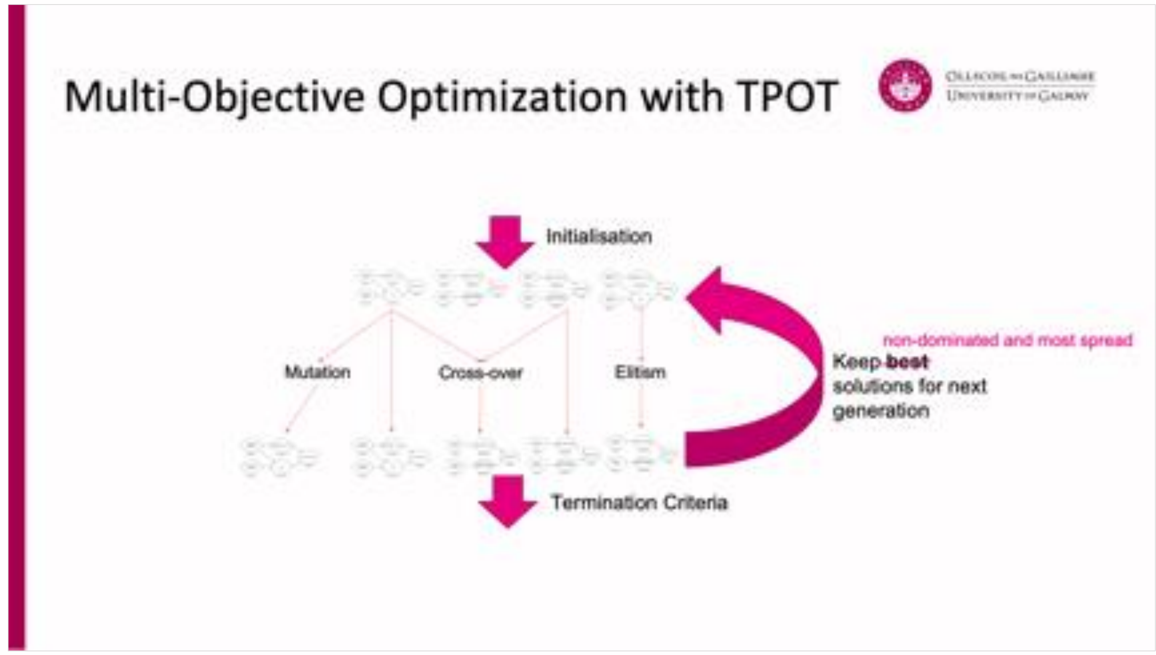


AutoML Pipeline Construction (Cont'd)

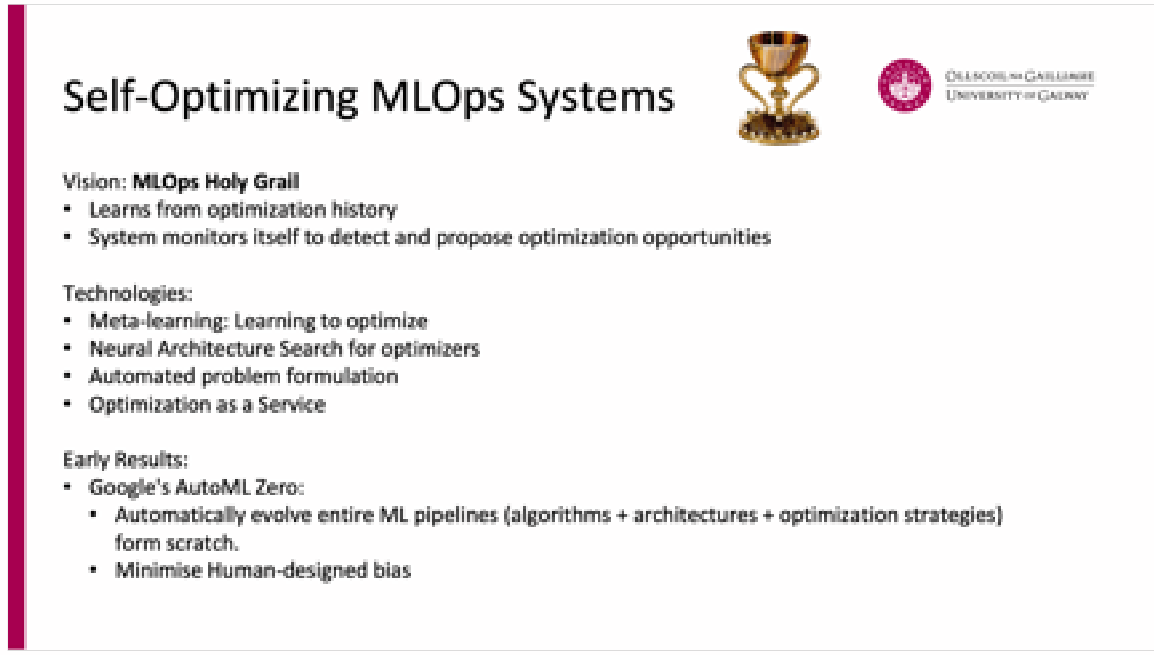
Approach 3: Directed Acyclic Graph (DAG)-Based Optimization

Example of TPOT Pipeline

[1] Ribeiro, P., Saini, A., Moran, J., Matsumoto, N., Choi, H., Hernandez, M. and Moore, J.H., 2024. TPOT2: A New Graph-Based Implementation of the Tree-Based Pipeline Optimization Tool for Automated Machine Learning. In *Genetic programming theory and practice XX* (pp. 1-17). Singapore: Springer Nature Singapore.



Multi-Objective Optimization with TPOT



Self-Optimizing MLOps Systems

Vision: **MLOps Holy Grail**

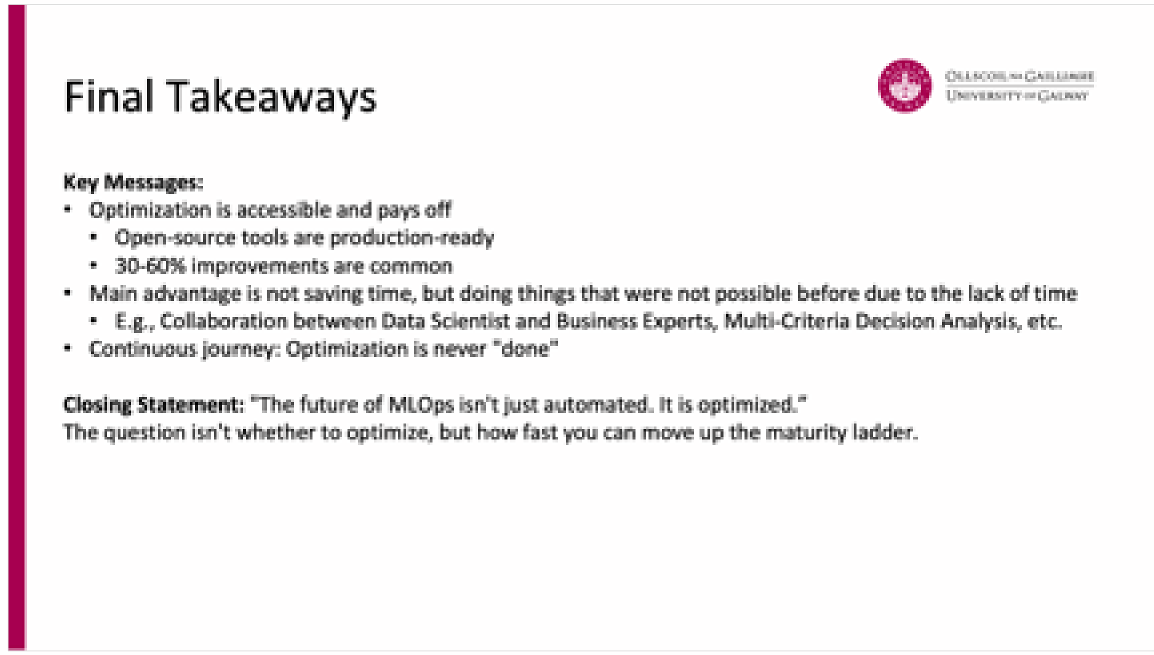
- Learns from optimization history
- System monitors itself to detect and propose optimization opportunities

Technologies:

- Meta-learning: Learning to optimize
- Neural Architecture Search for optimizers
- Automated problem formulation
- Optimization as a Service

Early Results:

- Google's AutoML Zero:
  - Automatically evolve entire ML pipelines (algorithms + architectures + optimization strategies) form scratch.
  - Minimise Human-designed bias



Final Takeaways

Key Messages:

- Optimization is accessible and pays off
  - Open-source tools are production-ready
  - 30-60% improvements are common
- Main advantage is not saving time, but doing things that were not possible before due to the lack of time
  - E.g., Collaboration between Data Scientist and Business Experts, Multi-Criteria Decision Analysis, etc.
- Continuous journey: Optimization is never "done"

Closing Statement: "The future of MLOps isn't just automated. It is optimized."
The question isn't whether to optimize, but how fast you can move up the maturity ladder.