UNIVERSITY OF CAMBRIDGE

Session 5

Learning the capabilities of large language models

Marko Tešić

CFI LEVERHULME CENTRE FOR THE FUTURE OF INTELLIGENCE

Vancouver, 20th February, 2024

Goals

- Understand the key challenges in building measurement layouts for learning the capabilities of large language models from benchmark datasets

- Run through a worked example of building measurement layouts for understanding the capabilities of 10 LLMs to add two numbers

Goals

- The session will provide you with an understanding of:
  - Benchmark characteristics needed to build measurement layouts
  - The relationship between abilities tested in the benchmarks and meta-features/demands that describe the benchmark instances
  - Assessing the performance of measurement layouts against common predictive methods and baselines
  - How is building measurement layouts and predicting performance on tasks different from predicting performance from common predictive models like logistic regression and XGBoost
  - Modeling hierarchical abilities with measurement layouts

Key challenges in building measurement layouts for LLMs: Performance data

- We need instance-level performance data
  - For each prompt need to have how the language model performed
    - Success/failure or if multiple choise which are the choises, which one did LLM pick and if correct or not
    - O-PIAAGETS: for each run we know whether the accent succeded or not (and more, e.g. did it guess the correct side?)
    - Addition example -- prompt: '`Sum of 112359321 and 23456543 is`'; success: `yes`

- Some good examples include: HELM and BIG-bench

Key challenges in building measurement layouts for LLMs: Meta-features/demands

- Many benchmarks don't include and/or don't have meta-features that describe each instance in the benchmark
  - How is the benchmark built?
  - What was considered when building the benchmark beyond just this a benchmark that tests X?
  - Are there any features that were explicitly manipulated?
- If no meta-features are available, we need to build some
- O-PIAAGETS: for each instance we know whether there is **lava, the size of the reward, reward distance, occluder present** etc.
- Addition: **number of digits in each number**; create new features: **average number of digits**, **number of carrying operations**, etc.

Key challenges in building measurement layouts for LLMs: Capabilities

- What capabilities are tested by those benchmarks?
  - O-PIAAGETS: number of meta-features/demands impact *OP ability*: **occluder presence, time under occlusion** etc.
  - Addition: what are the abilities that are being tested/informed or that need to match demands such as **average number of digits**, **number of carry operations** etc.
- Are those capabilities characterizable and have some backing in cognitive science (or some other sciences)?
- How do these capabilities interact? Does having one capability compensate for not having some other capability test on the benchmark?

Summary

- Need **instance-level** performance data
- Need **meta-features/demands** that describe each instance
- Need **capabilities** that are tested/informed by those demands

Addition dataset

- A simple benchmark testing LLMs abilities to add two numbers
  - E.g. prompt: `The sum of 656050998910983832047 and 4871137 is`
- 10 LLMs tested: GPT-3 Ada, GPT-3 Babbage, GPT-3 Curie, GPT-3 Davinci, text-davinci-001, text-davinci-002, text-davinci-003, GPT-4-0314, GPT-3.5-turbo, GPT-4-0613
- We will see how to create some of the **meta-features** for this benchmark and how to relate them to **capabilities** learnt by the measurement layout
- We will also train some common predictive models and baselines to compare to the performance of measurement layouts

Addition dataset

- Addition dataset is not yet publicly available

- For mode details, follow the notebook: https://github.com/Kinds-of-Intelligence-CFI/measurement-layout-tutorial/blob/main/tutorial-notebooks/5_1_Learning_the_capabilities_of_LLMs_Addition_data.ipynb

Addition dataset: overview

- `instance_id`: ID of an instance.
- `template_id`: ID of a prompt template. Each addition prompt was phrased in 8 different ways.
- `llm`: The large language model tested.
- `prompt`: The actual prompt.
- `response`: The LLM's response.
- `summand1`: The first number to be added.
- `summand2`: The second number to be added.
- `target`: The correct sum of the two summands.
- `digits1`: The number of digits in the first summand.
- `digits2`: The number of digits in the second summand.
- `min_digits`: $min(digits_1, digits_2)$, i.e., the number of digits in the smaller summand.
- `harm_mean`: $2/(1/digits_1 + 1/digits_2)$, i.e., the harmonic mean of the number of digits in the two summands.
- `art_mean`: $(digits_1 + digits_2)/2$, i.e., the arithmetic mean of the number of digits in the two summands.
- `max_digits`: $max(digits_1, digits_2)$, i.e., the number of digits in the larger summand.
- `carry`: The number of carrying operations required to add the two numbers.
- `success`: Indicates whether the response is correct (1) or incorrect (0).

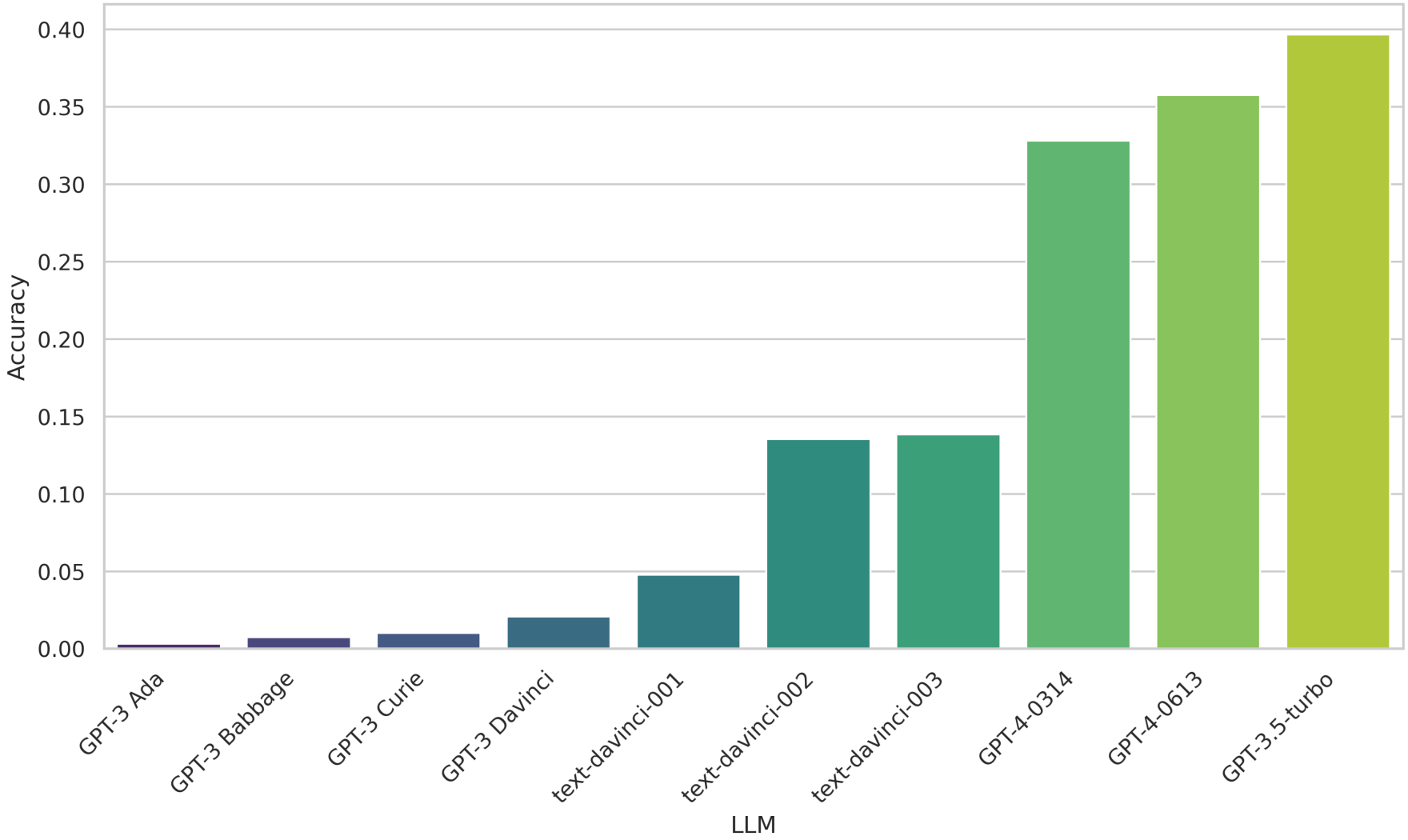
```
Template 1: The sum of 656050998910983832047 and 4871137 is
Template 2: Add: 656050998910983832047 + 4871137 =
Template 3: Make the addition of 656050998910983832047 and 4871137.
Template 4: By adding 656050998910983832047 and 4871137, the result is
Template 5: If you add 656050998910983832047 and 4871137, you get
Template 6: If you have 656050998910983832047 and 4871137, and you add them up, you get
Template 7: Find the value of x + y when x=656050998910983832047 and y=4871137.
Template 8: Imagine you have two numbers, 656050998910983832047 and 4871137, and you added them together. What number would you get?
```

Between 1 and 30 digits

Addition dataset

success	0	1
llm		
GPT-3 Ada	3286	10
GPT-3 Babbage	3272	24
GPT-3 Curie	3263	33
GPT-3 Davinci	3228	68
GPT-3.5-turbo	1989	1307
GPT-4-0314	2214	1082
GPT-4-0613	2117	1179
text-davinci-001	3139	157
text-davinci-002	2850	446
text-davinci-003	2840	456

Addition dataset: aggregate accuracy



Meta-features/demands

- `digits1`: The number of digits in the first summand.
- `digits2`: The number of digits in the second summand.
- `min_digits`: $min(digits_1, digits_2)$, i.e., the number of digits in the smaller summand.
- `harm_mean`: $2/(1/digits_1 + 1/digits_2)$, i.e., the harmonic mean of the number of digits in the two summands.
- `art_mean`: $(digits_1 + digits_2)/2$, i.e., the arithmetic mean of the number of digits in the two summands.
- `max_digits`: $max(digits_1, digits_2)$, i.e., the number of digits in the larger summand.
- `carry`: The number of carrying operations required to add the two numbers.

- Do these meta-features capture everything that could we could potentially think of when it comes to addition of two number? What are some of the things that make the addition of two number ‘difficult’?
- Size of the two numbers
- Number of carrying operations
- Can we have lots of carrying operations but the additions is still ‘easy’?

Meta-features/demands

- Standard deviation of digits in the two numbers (digit variety):

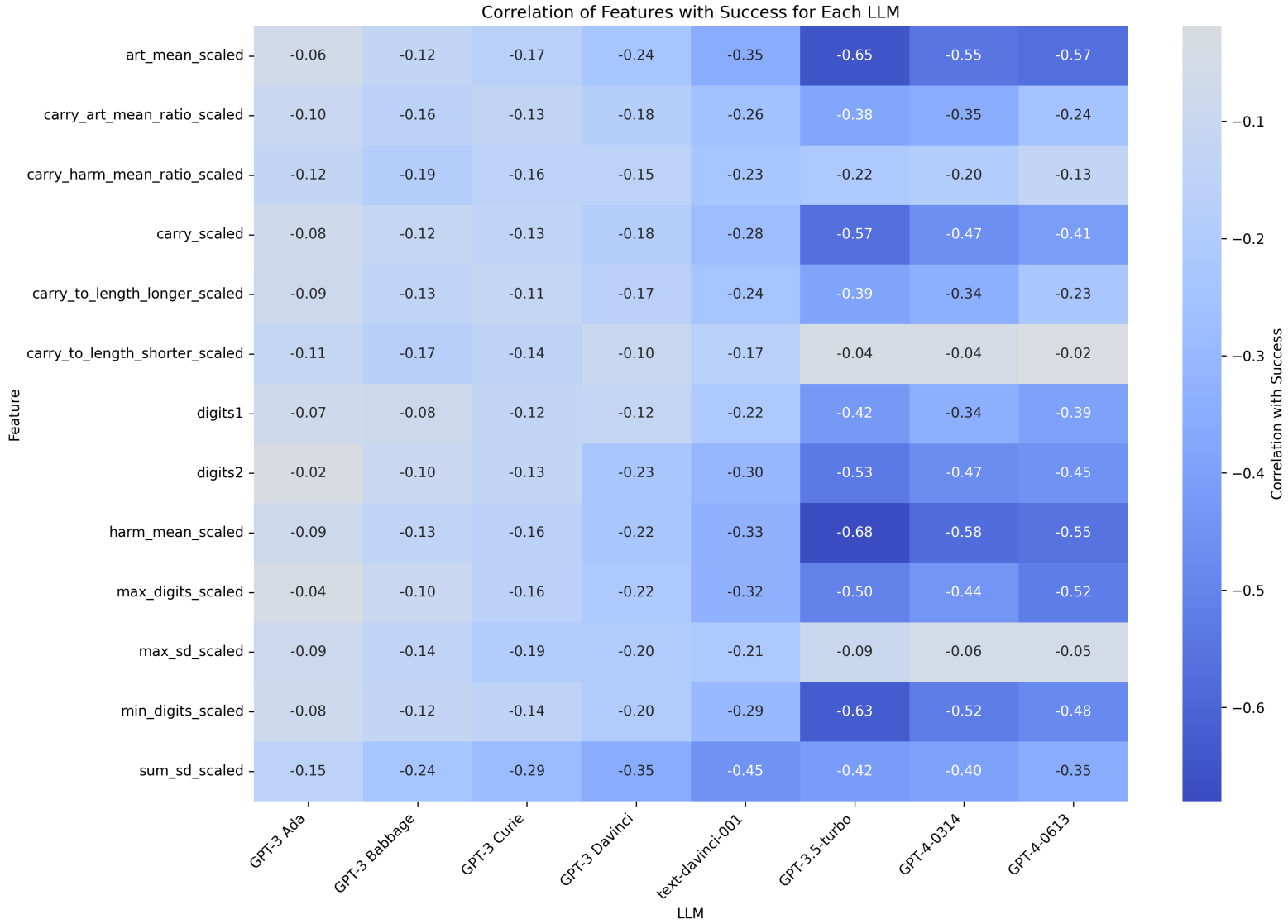
```
# Calculate max and sum of digit standard deviations
addition_data['max_sd'] = addition_data.apply(lambda x: max(num_sd(x['summand1']), num_sd(x['summand2'])), axis=1)
addition_data['sum_sd'] = addition_data.apply(lambda x: num_sd(x['summand1']) + num_sd(x['summand2']), axis=1)
```

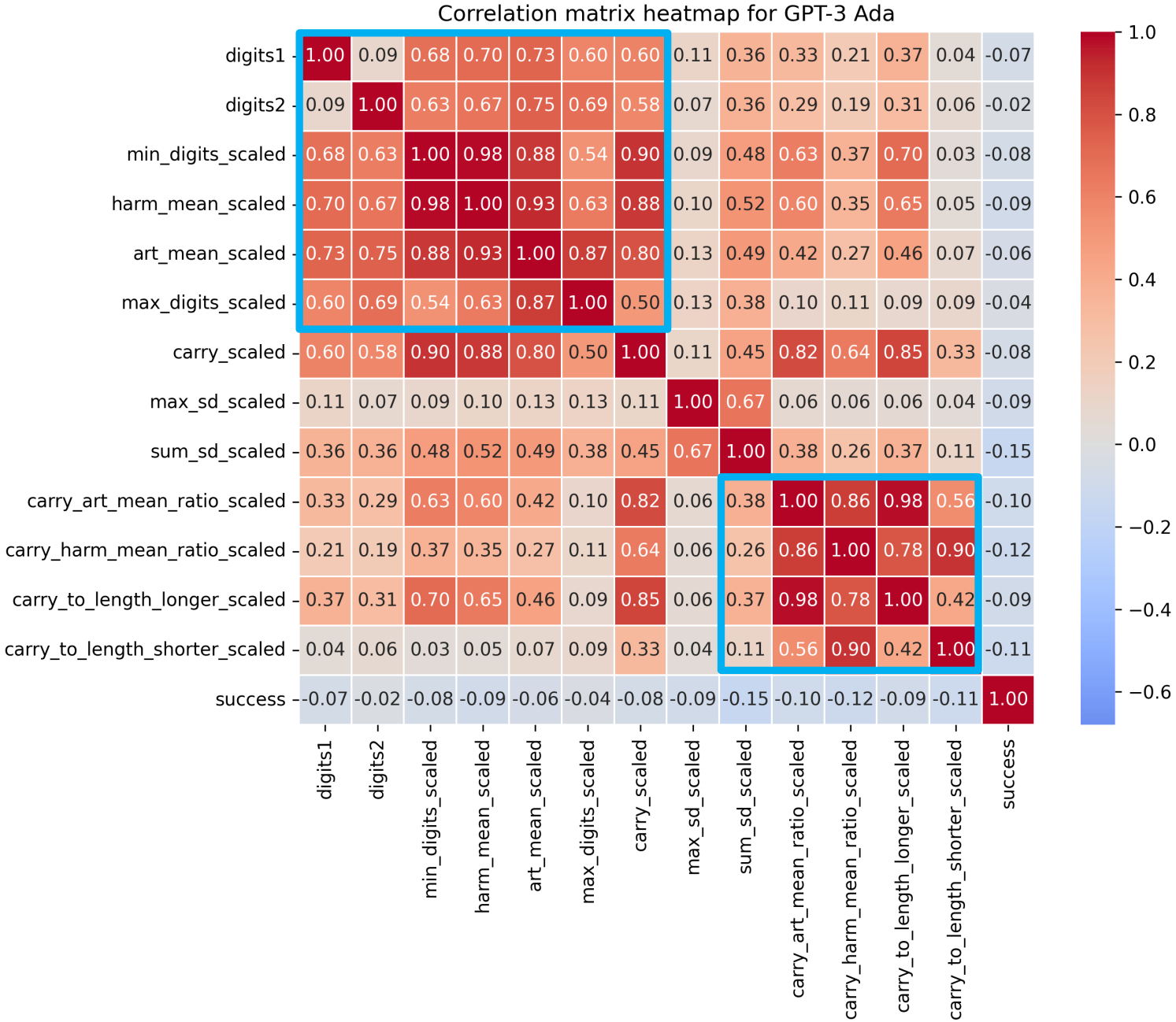
- Calculate the ratio of carry to size:

```
# Calculate ratio of carry and harmonic mean columns
addition_data['carry_harm_mean_ratio'] = addition_data['carry'] / addition_data['harm_mean']
addition_data['carry_art_mean_ratio'] = addition_data['carry'] / addition_data['art_mean']
addition_data['carry_to_length_longer'] = addition_data['carry'] / addition_data['max_digits']
addition_data['carry_to_length_shorter'] = addition_data['carry'] / addition_data['min_digits']
```

- Scale data: min-max normalization

Meta-features/demands: correlations with success



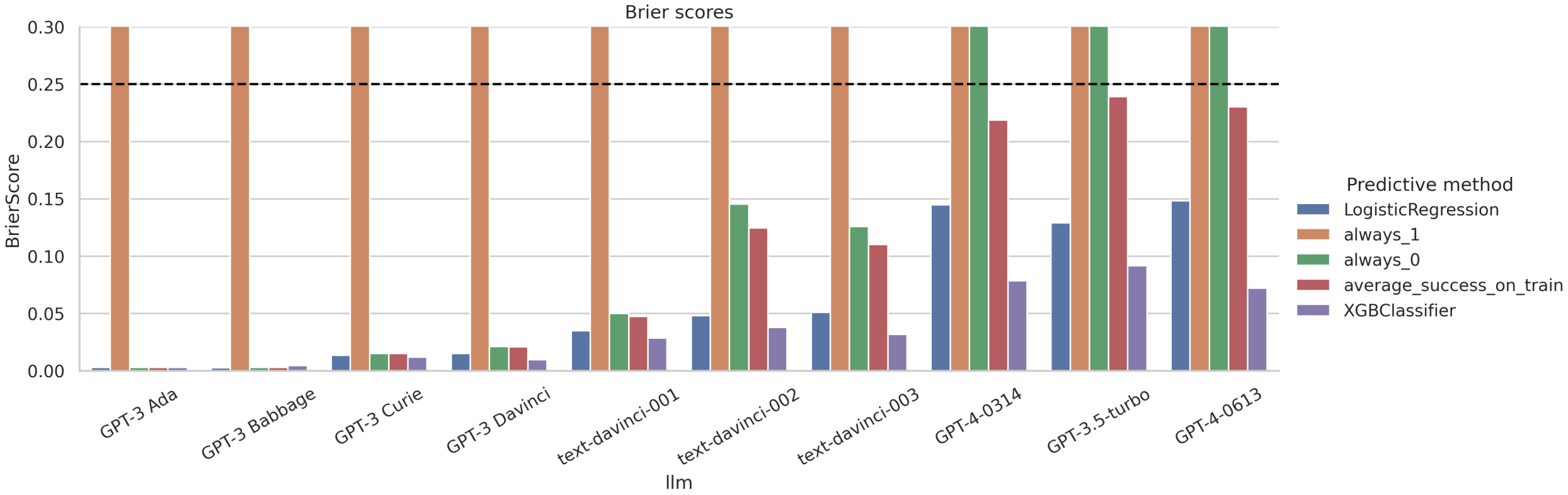

Correlation matrix heatmap for GPT-3 Ada
digits1 1.00 0.09 0.68 0.70 0.73 0.60 0.60 0.11 0.36 0.33 0.21 0.37 0.04 -0.07
digits2 0.09 1.00 0.63 0.67 0.75 0.69 0.58 0.07 0.36 0.29 0.19 0.31 0.06 -0.02
min_digits_scaled 0.68 0.63 1.00 0.98 0.88 0.54 0.90 0.09 0.48 0.63 0.37 0.70 0.03 -0.08
harm_mean_scaled 0.70 0.67 0.98 1.00 0.93 0.63 0.88 0.10 0.52 0.60 0.35 0.65 0.05 -0.09
art_mean_scaled 0.73 0.75 0.88 0.93 1.00 0.87 0.80 0.13 0.49 0.42 0.27 0.46 0.07 -0.06
max_digits_scaled 0.60 0.69 0.54 0.63 0.87 1.00 0.50 0.13 0.38 0.10 0.11 0.09 0.09 -0.04
carry_scaled 0.60 0.58 0.90 0.88 0.80 0.50 1.00 0.11 0.45 0.82 0.64 0.85 0.33 -0.08
max_sd_scaled 0.11 0.07 0.09 0.10 0.13 0.13 0.11 1.00 0.67 0.06 0.06 0.06 0.04 -0.09
sum_sd_scaled 0.36 0.36 0.48 0.52 0.49 0.38 0.45 0.67 1.00 0.38 0.26 0.37 0.11 -0.15
carry_art_mean_ratio_scaled 0.33 0.29 0.63 0.60 0.42 0.10 0.82 0.06 0.38 1.00 0.86 0.98 0.56 -0.10
carry_harm_mean_ratio_scaled 0.21 0.19 0.37 0.35 0.27 0.11 0.64 0.06 0.26 0.86 1.00 0.78 0.90 -0.12
carry_to_length_longer_scaled 0.37 0.31 0.70 0.65 0.46 0.09 0.85 0.06 0.37 0.98 0.78 1.00 0.42 -0.09
carry_to_length_shorter_scaled 0.04 0.06 0.03 0.05 0.07 0.09 0.33 0.04 0.11 0.56 0.90 0.42 1.00 -0.11
success -0.07 -0.02 -0.08 -0.09 -0.06 -0.04 -0.08 -0.09 -0.15 -0.10 -0.12 -0.09 -0.11 1.00
digits1
digits2
min_digits_scaled
harm_mean_scaled
art_mean_scaled
max_digits_scaled
carry_scaled
max_sd_scaled
sum_sd_scaled
carry_art_mean_ratio_scaled
carry_harm_mean_ratio_scaled
carry_to_length_longer_scaled
carry_to_length_shorter_scaled
success
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4
−0.6


harm_mean_scaled
sum_sd_scaled
carry_art_mean_ratio_scaled
GPT-3 Ada
GPT-3 Babbage
GPT-3 Curie
GPT-3 Davinci
text-davinci-001
text-davinci-002
text-davinci-003
GPT-4-0314
GPT-3.5-turbo
GPT-4-0613
0.00
0.25
0.50
0.75
1.00

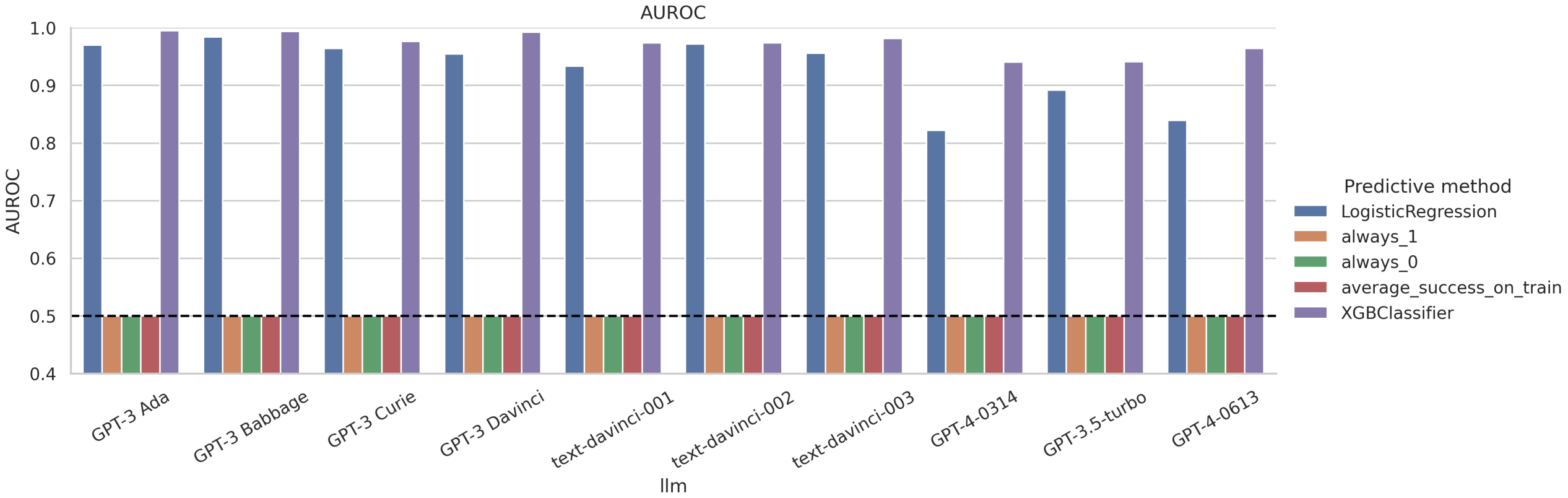
Common predictive models

- Build Logistic regression and XGBoost classifier for each LLM to predict success using the 3 meta-features (harm_mean, carry_art_mean_ratio, sum_sd)
- Add baselines:
  - Predict success with probability 1 for each instance
  - Predict success with probability 0 for each instance
  - Predict success with probability equal to aggregate probability of success on the training set (i.e. the proportion of successes on the training data per LLM)
- Evaluate the performance of the predictive models using AUROC and Brier score

Common predictive models



Common predictive models



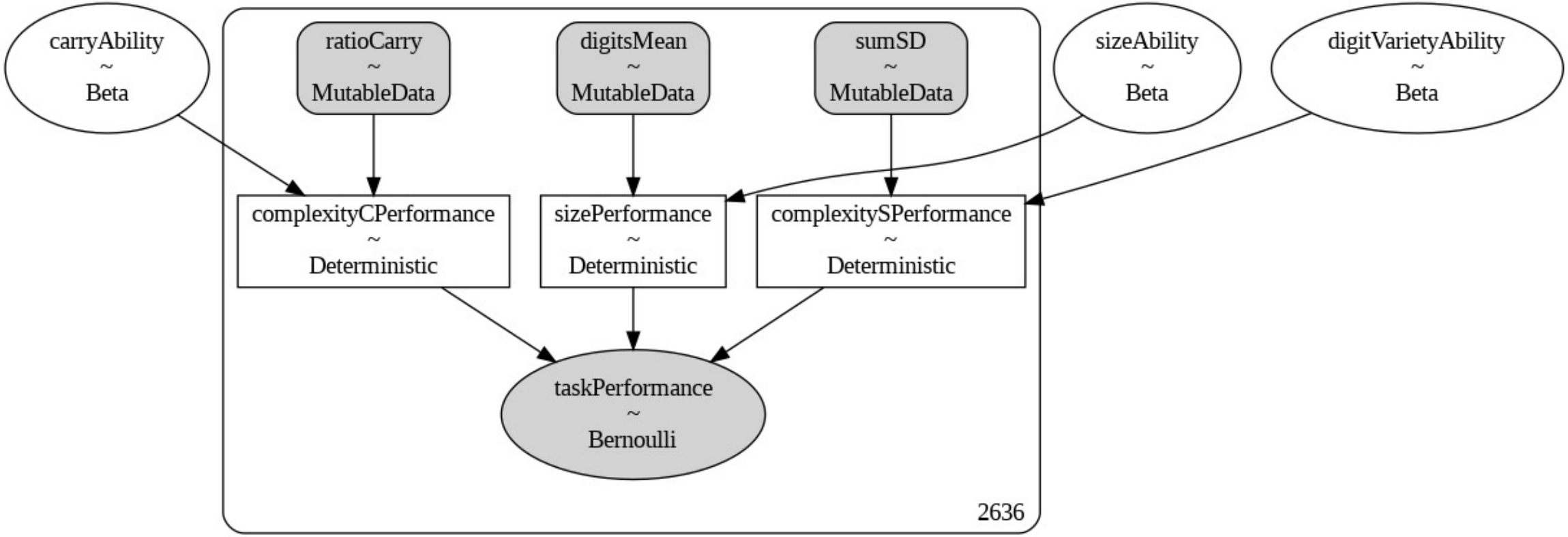
Capabilities

- 3 meta-features/demands: harm_mean, carry_art_mean_ratio, sum_sd
- Capabilities that are informed by/measured/match those demands:
  - harm_carry ↔ size ability
  - carry_art_mean_ratio ↔ carry ability (ability to perform addition that require carry to various extents)
  - sum_sd ↔ digit variety ability
- Simple case of one demand, one ability (in general doesn't have to be as we've seen in the previous session)
- Abilities non-compensatory (to an extent)

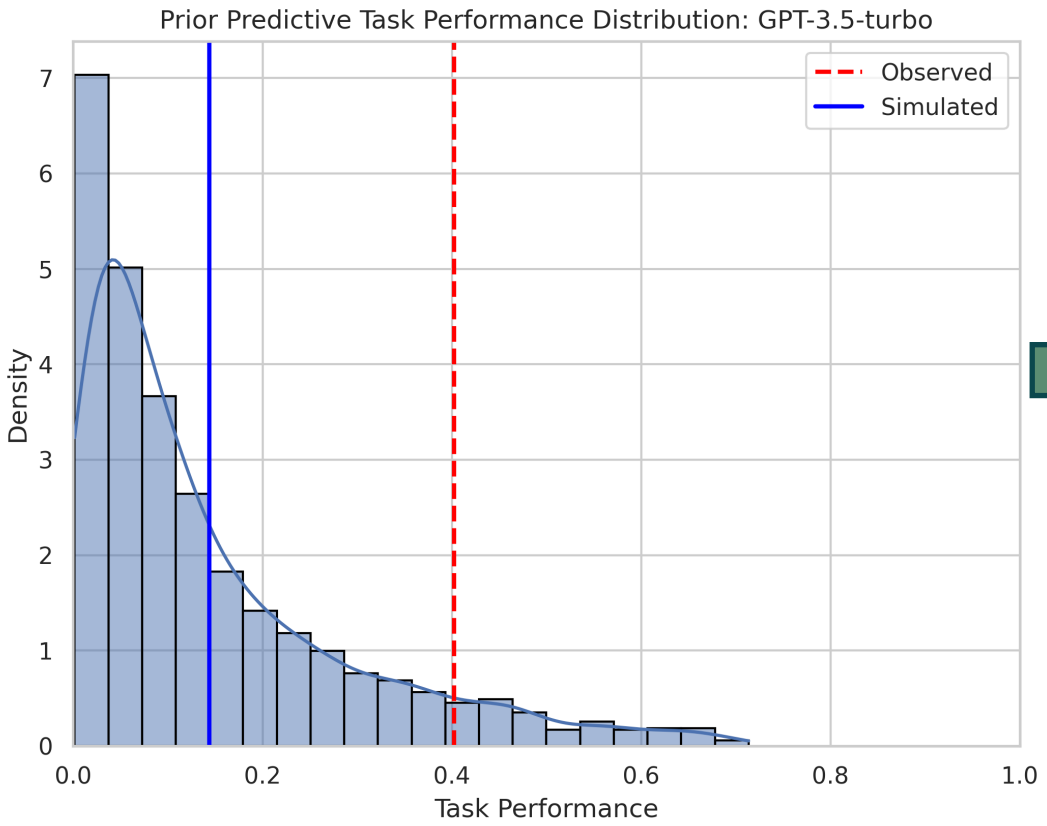
Measurement layouts

```
def setupSumModel(data):

  m = pm.Model()
  with m:

    # Define abilities and their priors
    sizeAbility = pm.Beta("sizeAbility", 1,1)
    carryAbility = pm.Beta("carryAbility", 1,1)
    digitVarietyAbility = pm.Beta("digitVarietyAbility", 1,1)

    # Define environment variables as MutableData
    digitsMean = pm.MutableData("digitsMean", data['harm_mean_scaled'].values)
    ratioCarry = pm.MutableData("ratioCarry", data['carry_art_mean_ratio_scaled'].values)
    sumSD = pm.MutableData("sumSD", data['sum_sd_scaled'].values)

    # Margins / partial performance
    sizePerformance = pm.Deterministic("sizePerformance", logistic999(margin(sizeAbility,digitsMean), min = 0, max = 1))
    carryPerformance = pm.Deterministic("carryPerformance", logistic999(margin(carryAbility,ratioCarry), min = 0, max = 1))
    digitVarietyPerformance = pm.Deterministic("digitVarietyPerformance", logistic999(margin(digitVarietyAbility,sumSD), min=0, max=1))

    #Non-compensatory interaction
    perfList = [sizePerformance,carryPerformance,digitVarietyPerformance]
    finalPerformance =  np.prod(perfList)

    taskPerformance = pm.Bernoulli("taskPerformance", finalPerformance, observed = data['success'])


  return m
```

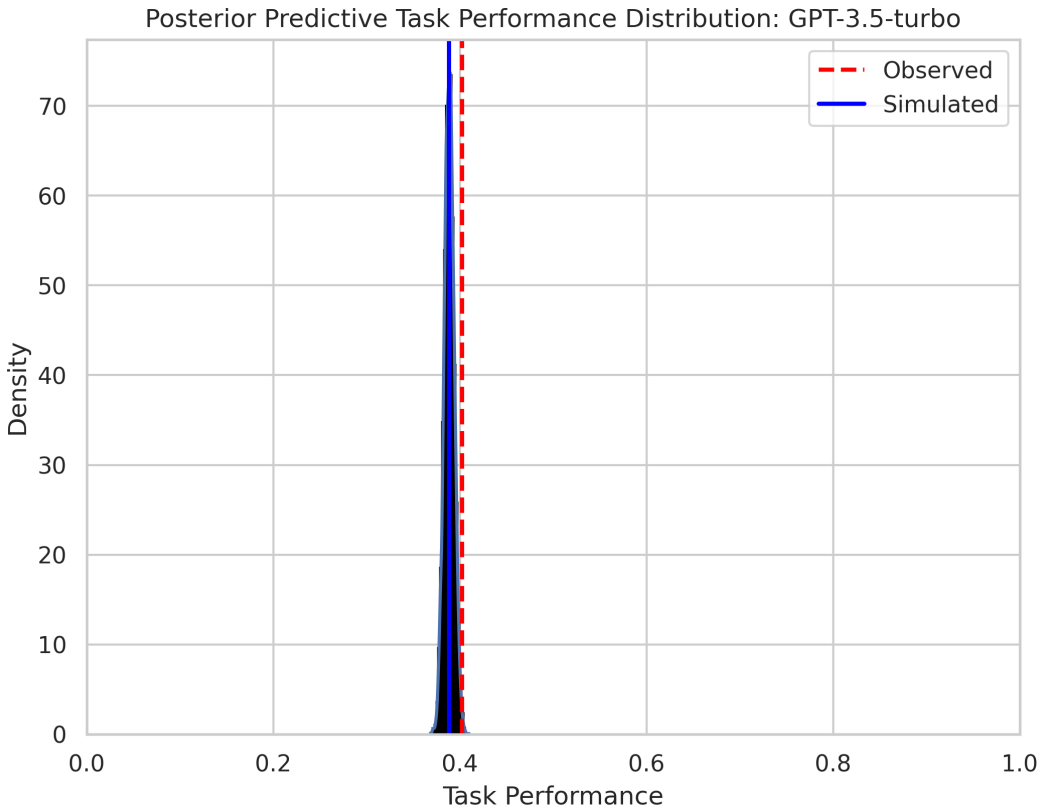
Measurement layouts



Measurement layouts: prior and posterior predictive checks

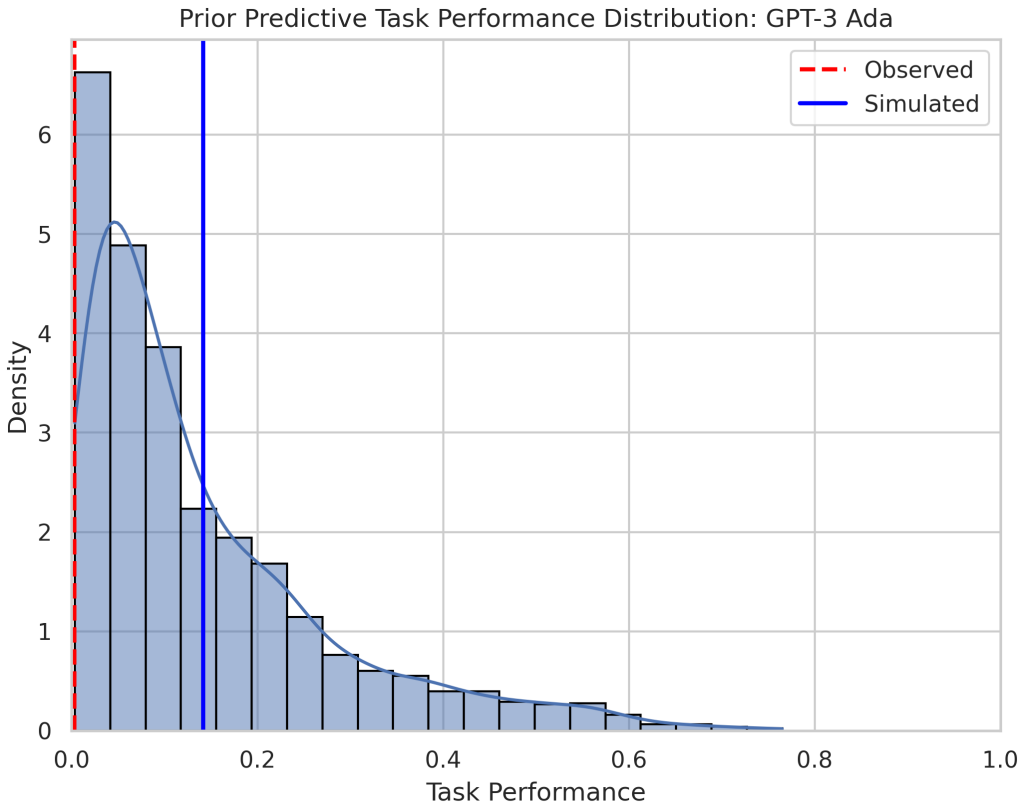


Sample from the posterior

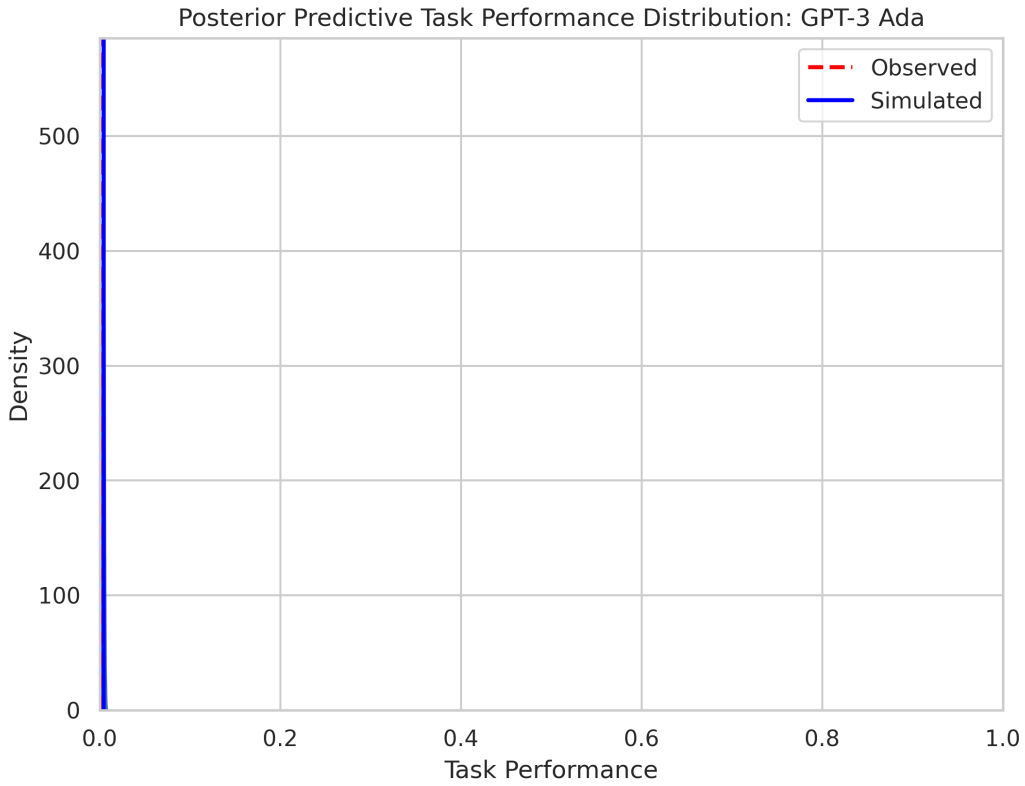


$$\theta^{sim} \sim p(\theta) \qquad y^{sim} \sim p(y|\theta^{sim})$$

Measurement layouts: prior and posterior predictive checks

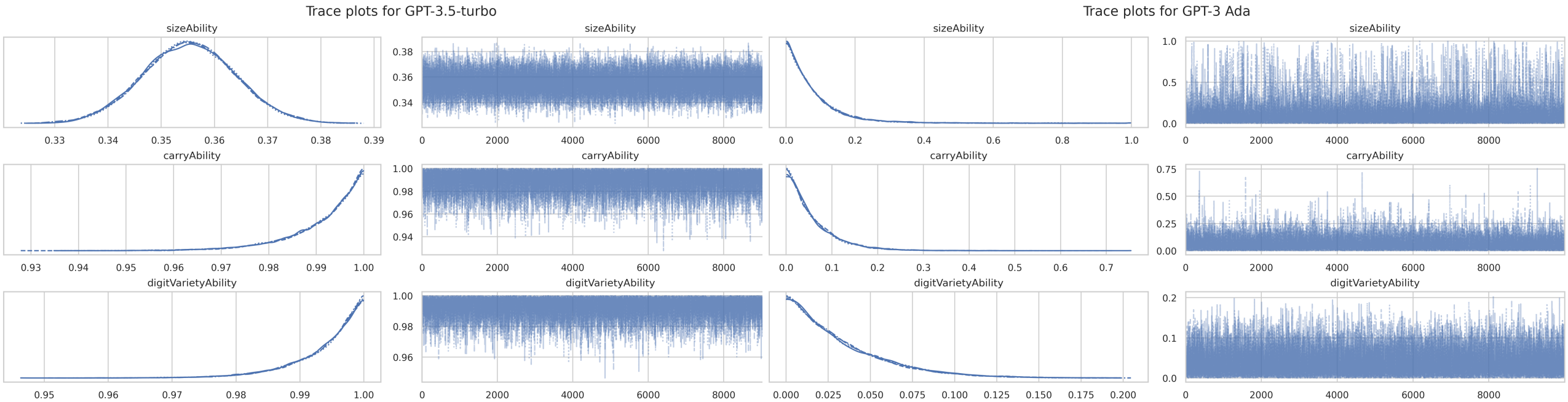


Sample from the posterior

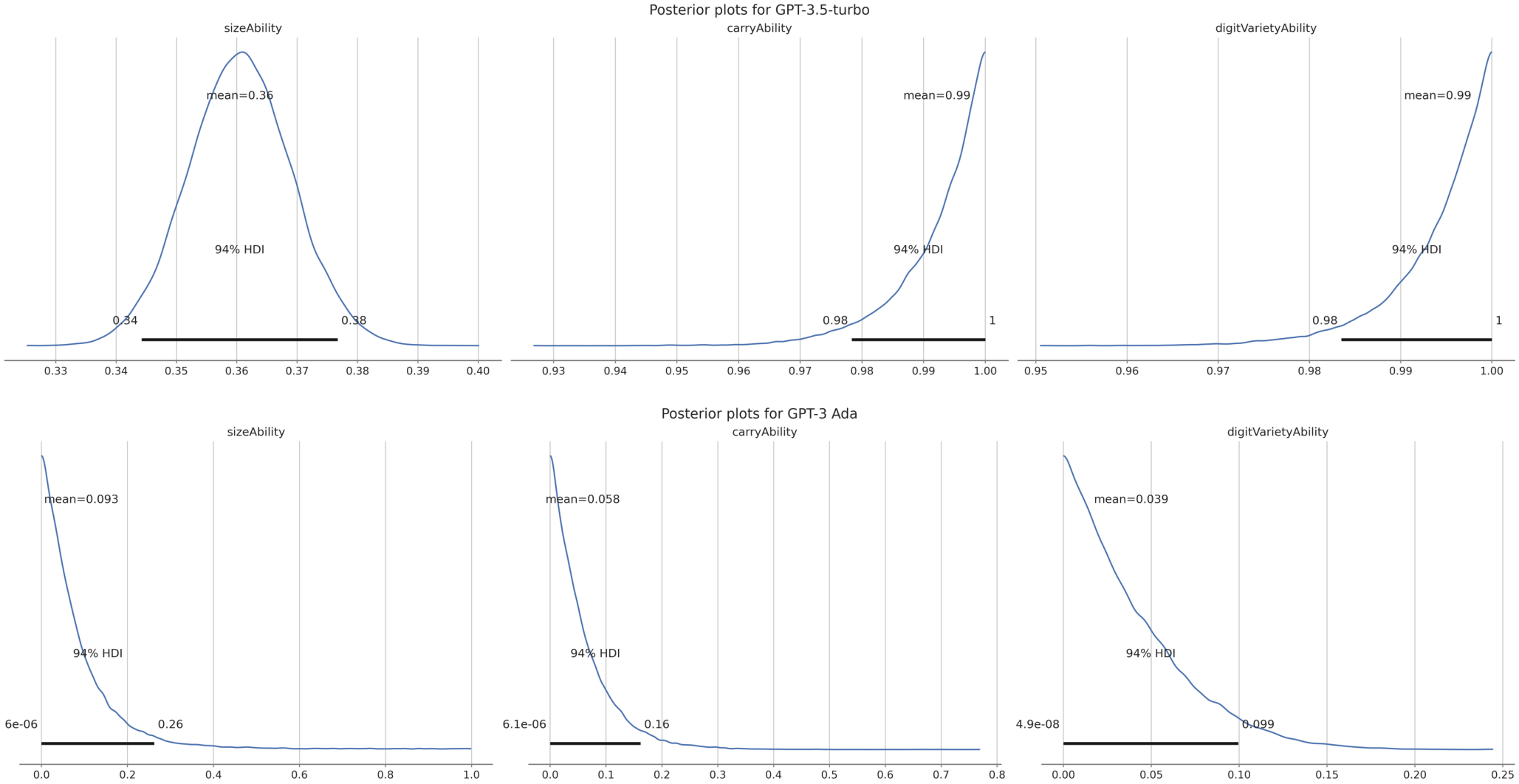


$$\theta^{sim} \sim p(\theta) \qquad y^{sim} \sim p(y|\theta^{sim})$$

Measurement layouts: MCMC chains

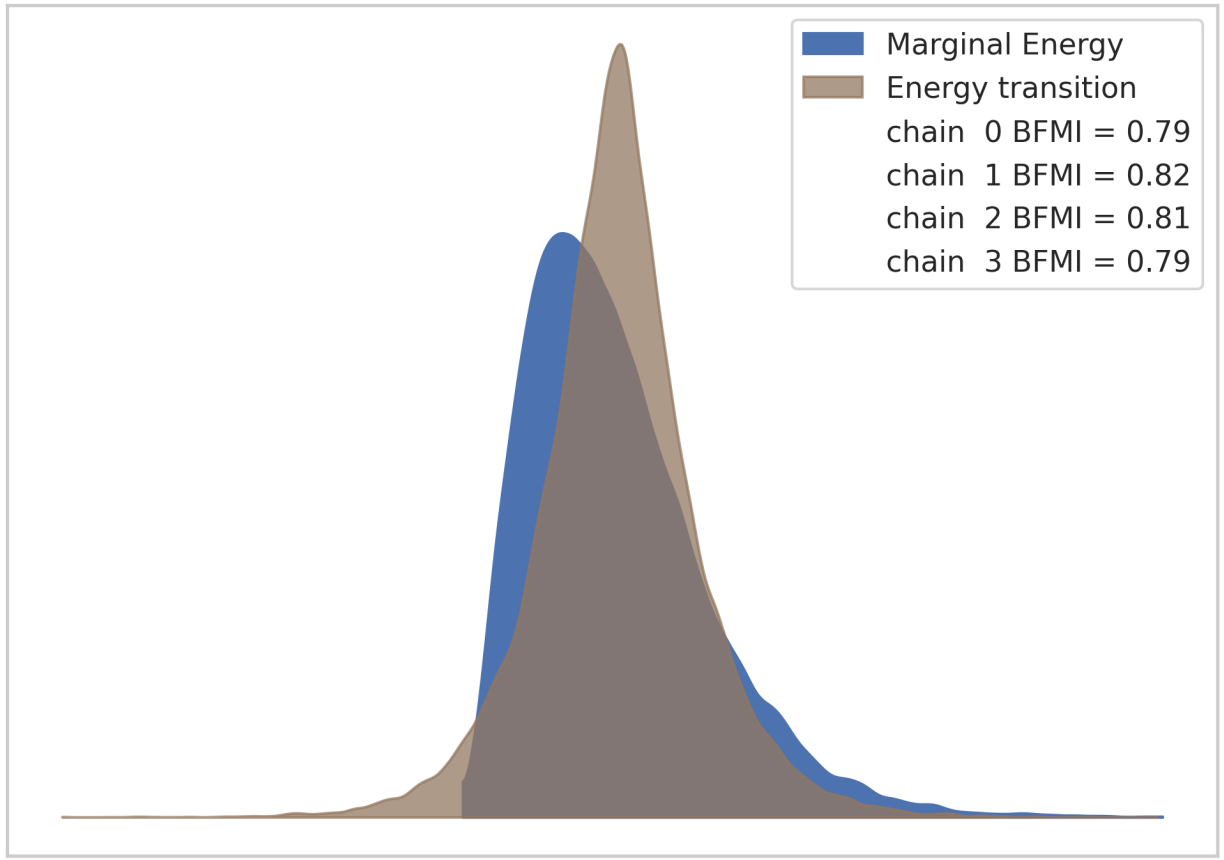


Measurement layouts: posterior plots

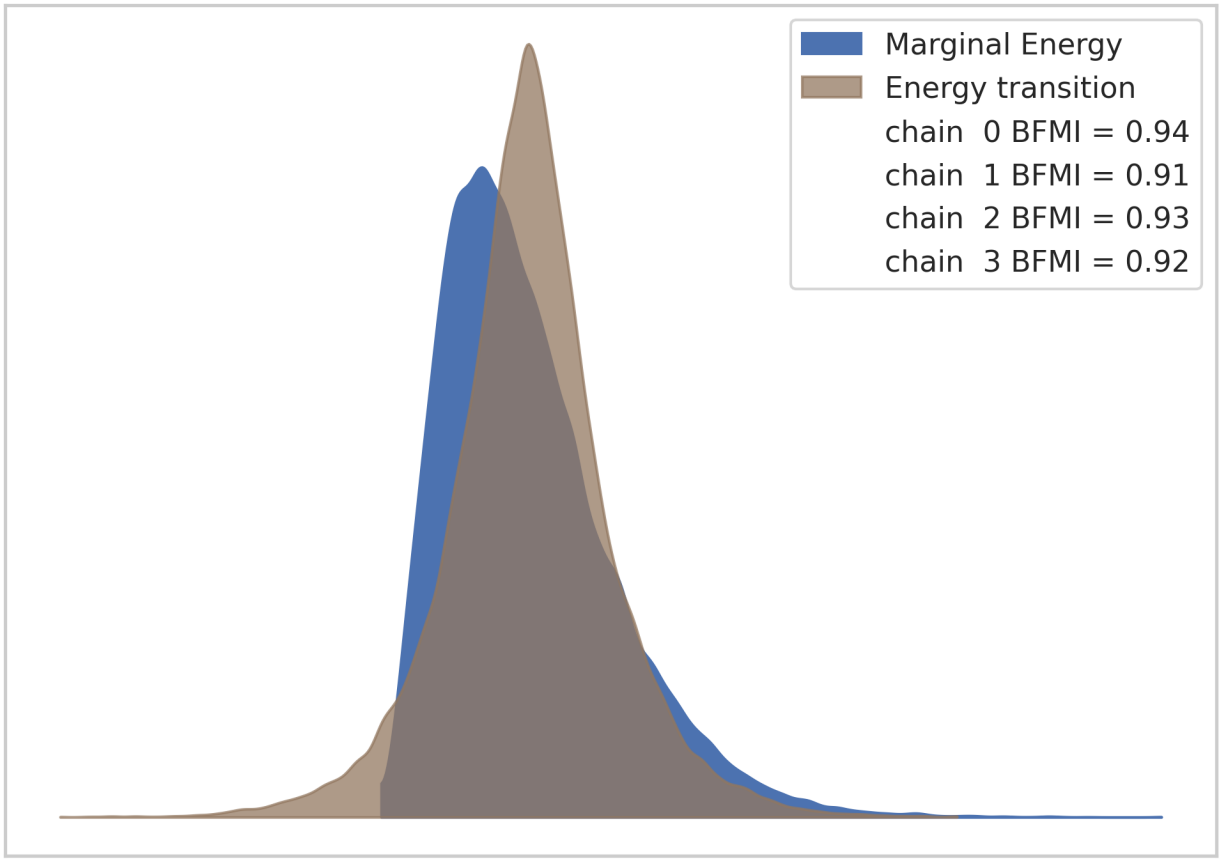


Measurement layouts: convergence metrics

Energy plot for GPT-3 Ada



Energy plot for GPT-3.5-turbo



Measurement layouts: convergence metrics

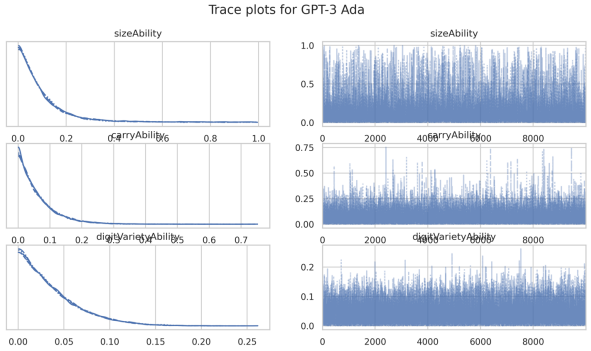
Summary statistics: GPT-3.5-turbo

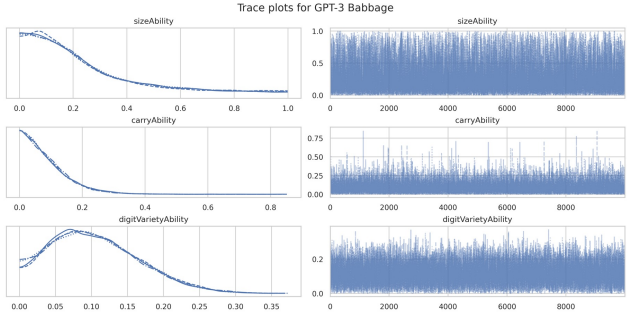
	mean	sd	hdi_3%	hdi_97%	ess_bulk	ess_tail	r_hat
sizeAbility	0.361	0.009	0.344	0.377	22956.159	22305.809	1.0
carryAbility	0.992	0.007	0.978	1.000	19785.217	13071.833	1.0
digitVarietyAbility	0.994	0.006	0.983	1.000	17494.655	11998.507	1.0

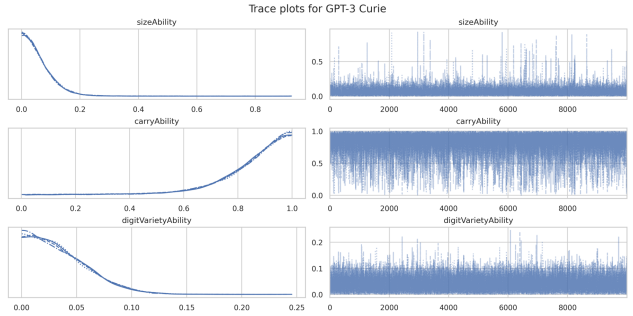
Summary statistics: GPT-3 Ada

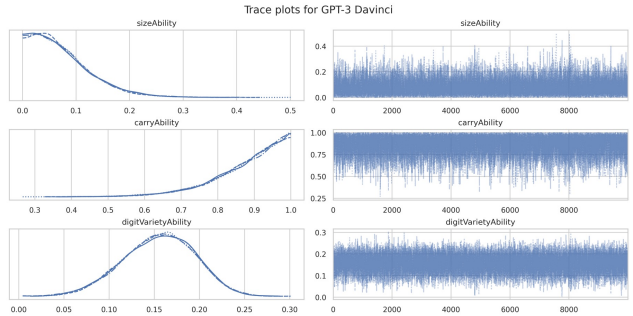
	mean	sd	hdi_3%	hdi_97%	ess_bulk	ess_tail	r_hat
sizeAbility	0.093	0.119	0.0	0.262	18410.049	14472.797	1.0
carryAbility	0.058	0.058	0.0	0.162	17052.909	14567.057	1.0
digitVarietyAbility	0.039	0.033	0.0	0.099	16271.271	12893.759	1.0

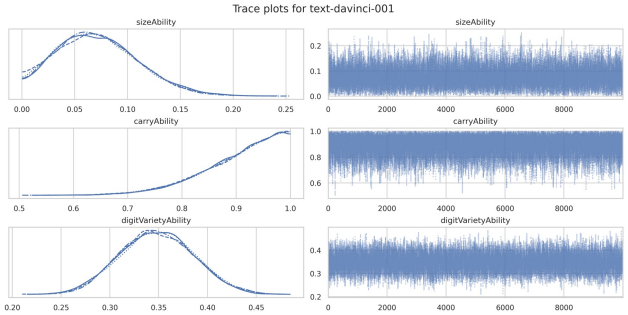
Build measurement layouts for each LLM

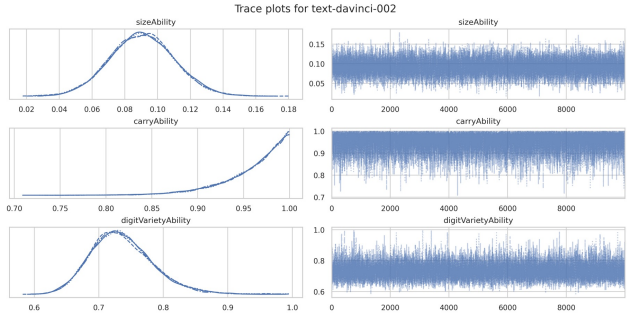


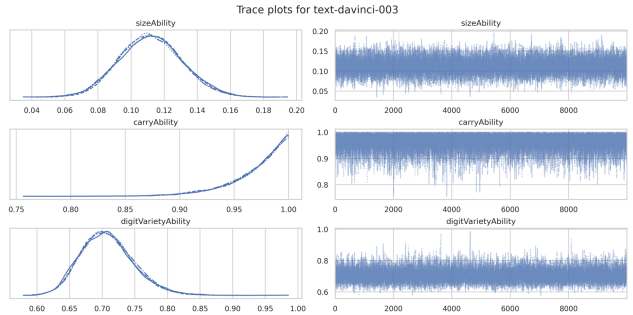


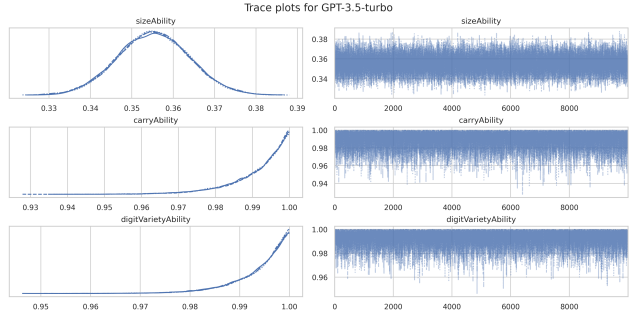


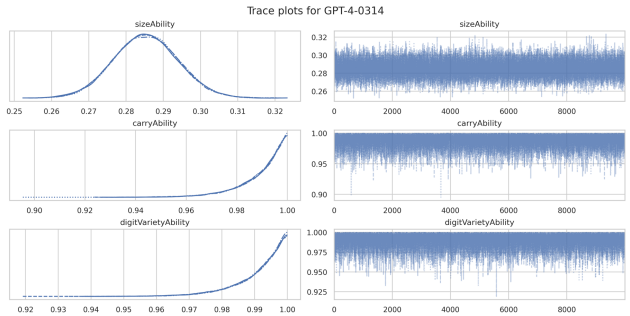


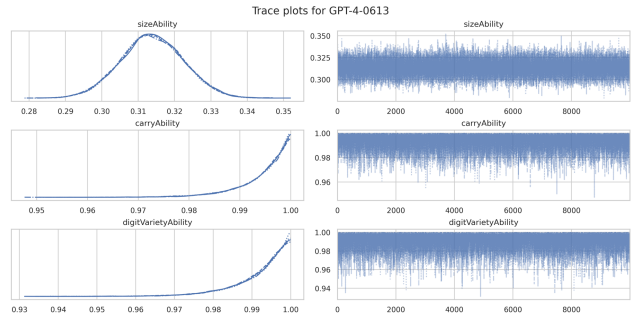




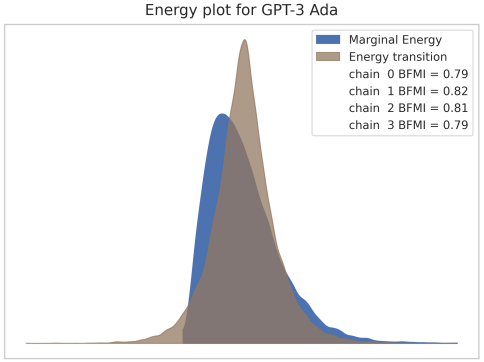


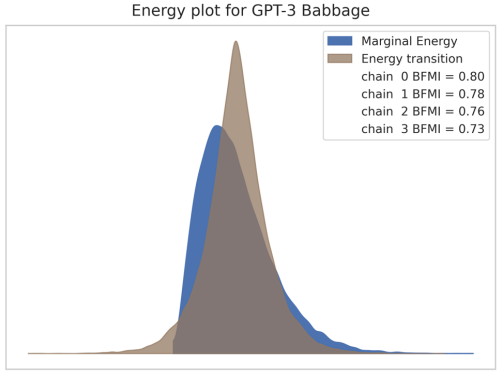


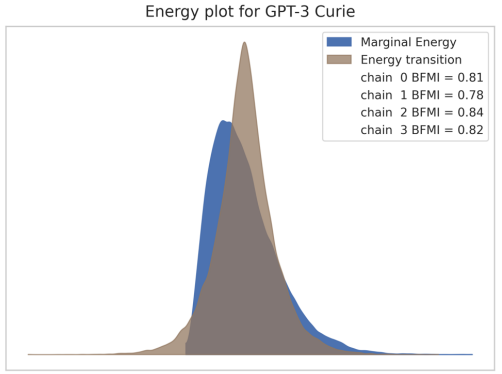


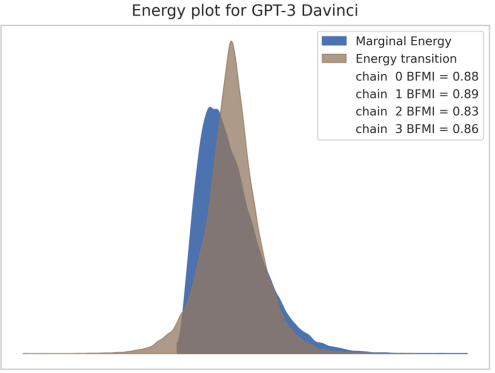


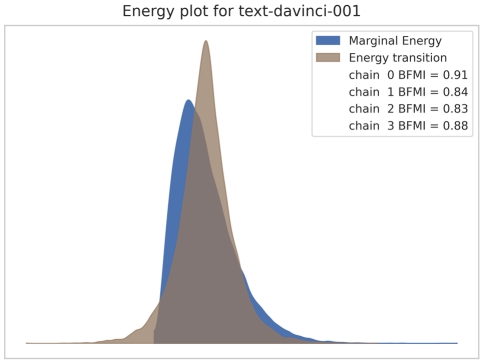
Build measurement layouts for each LLM

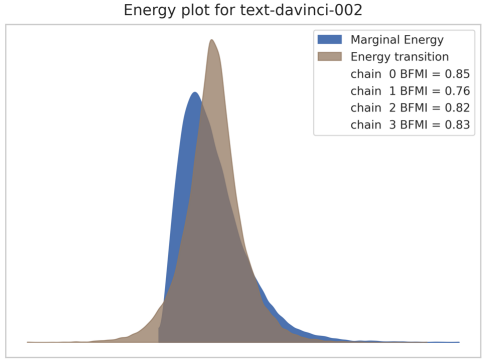


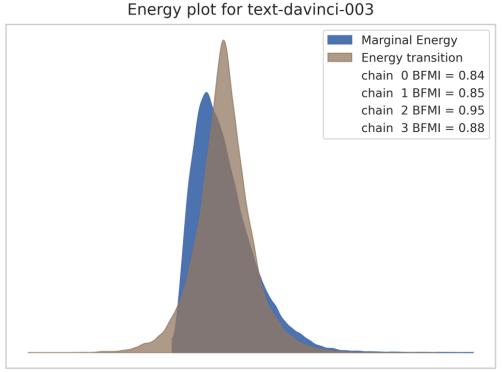


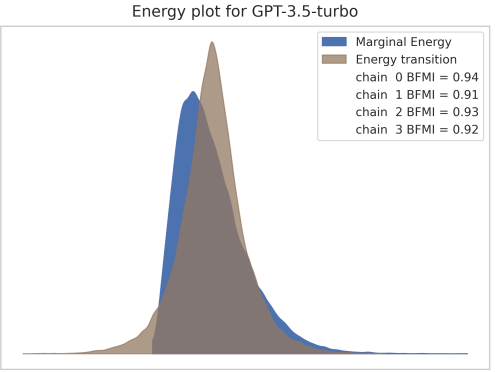


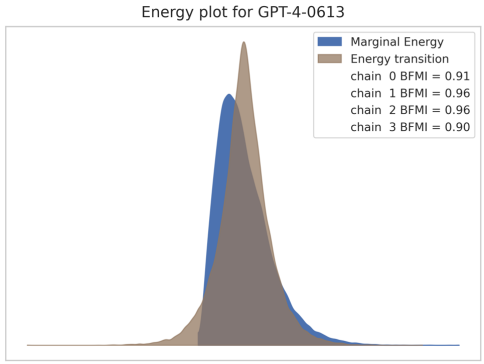


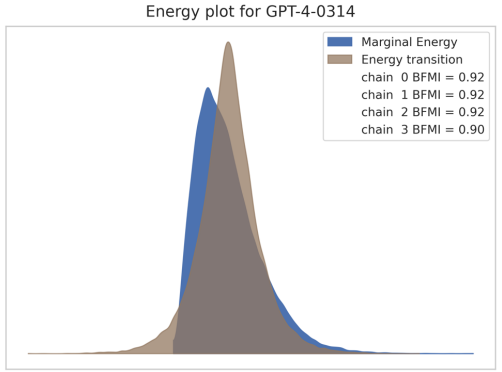






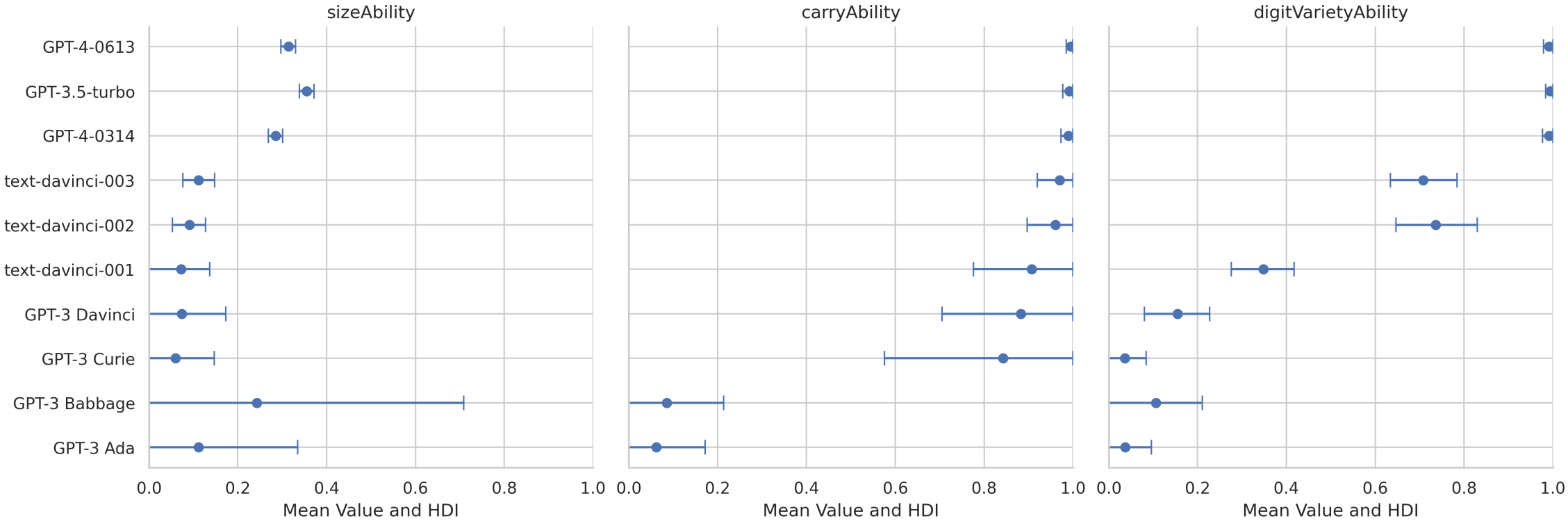




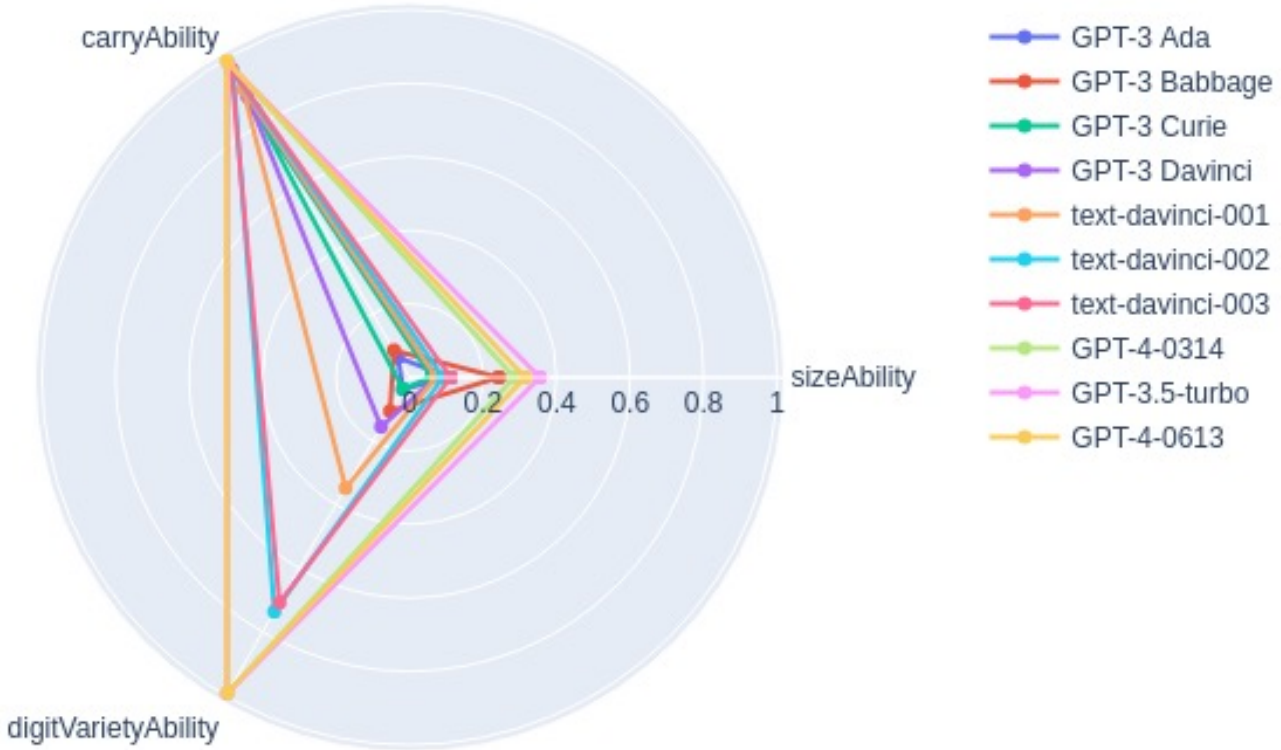


LLM	Ability	ess_bulk	ess_tail	r_hat
GPT-3 Ada	sizeAbility	15947.664	13647.912	1.000
	carryAbility	17314.838	12229.962	1.000
	digitVarietyAbility	14887.575	12581.714	1.000
GPT-3 Babbage	sizeAbility	13623.890	14121.887	1.000
	carryAbility	13543.988	10165.721	1.000
	digitVarietyAbility	12443.023	10333.832	1.000
GPT-3 Curie	sizeAbility	12067.735	10409.733	1.000
	carryAbility	16434.100	12125.648	1.000
	digitVarietyAbility	12464.858	10123.904	1.000
GPT-3 Davinci	sizeAbility	12055.600	11186.697	1.000
	carryAbility	13451.625	11293.579	1.000
	digitVarietyAbility	13493.404	12102.647	1.000
text-davinci-001	sizeAbility	9207.677	8831.978	1.000
	carryAbility	12767.635	10355.260	1.000
	digitVarietyAbility	9603.032	13422.536	1.000
text-davinci-002	sizeAbility	9819.594	9701.773	1.000
	carryAbility	12363.121	9549.371	1.000
	digitVarietyAbility	9675.911	8919.527	1.000
text-davinci-003	sizeAbility	10940.866	12531.203	1.001
	carryAbility	14302.540	10624.170	1.000
	digitVarietyAbility	11039.557	11186.508	1.001
GPT-4-0314	sizeAbility	23667.415	21057.464	1.000
	carryAbility	16964.006	12814.772	1.000
	digitVarietyAbility	17316.999	13167.265	1.000
GPT-3.5-turbo	sizeAbility	24346.102	22833.894	1.000
	carryAbility	16957.677	12185.843	1.000
	digitVarietyAbility	18844.659	12957.762	1.000
GPT-4-0613	sizeAbility	24670.705	23499.191	1.000
	carryAbility	18461.204	13153.547	1.000
	digitVarietyAbility	17580.396	13479.990	1.000

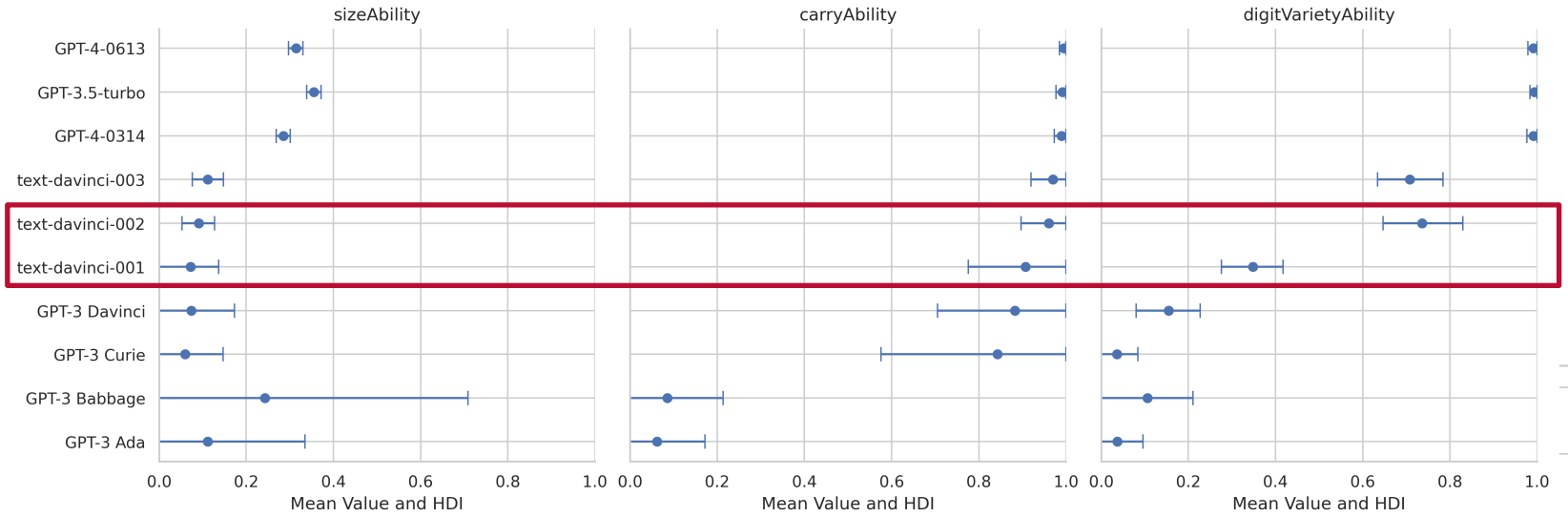
Build measurement layouts for each LLM

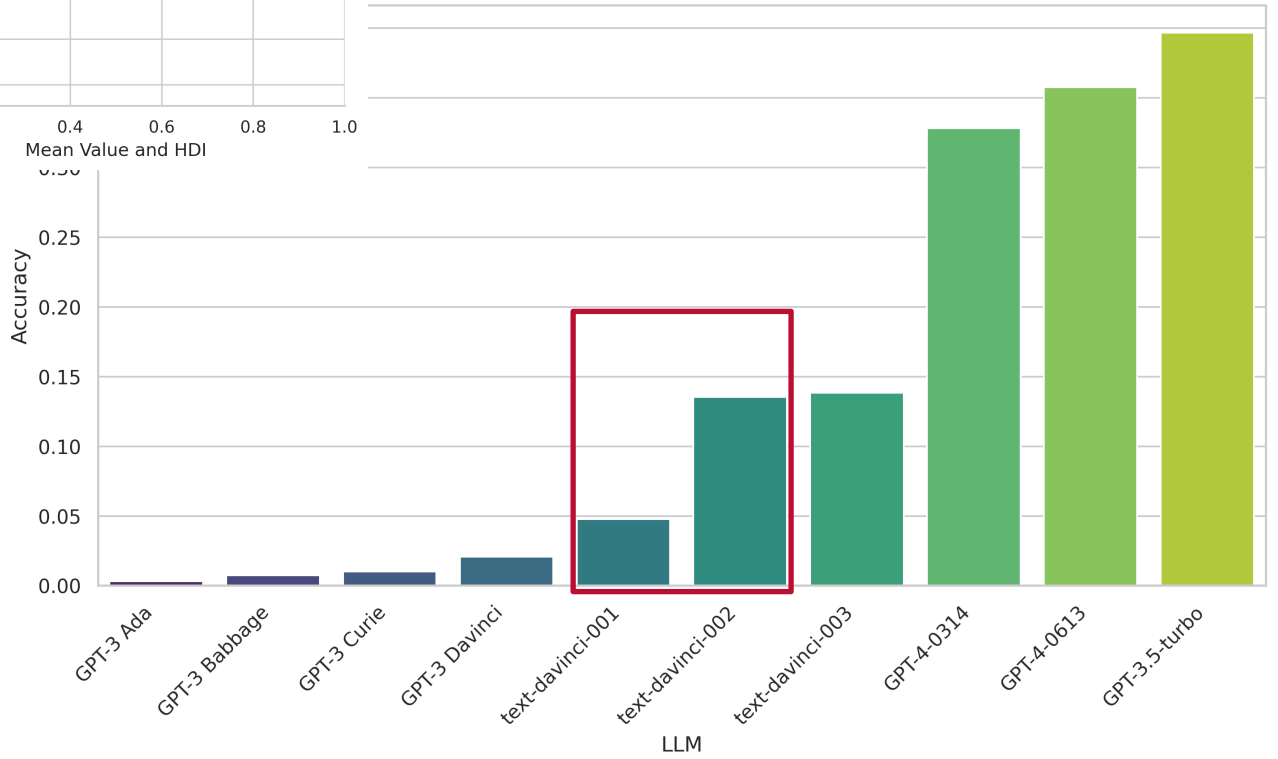


Build measurement layouts for each LLM

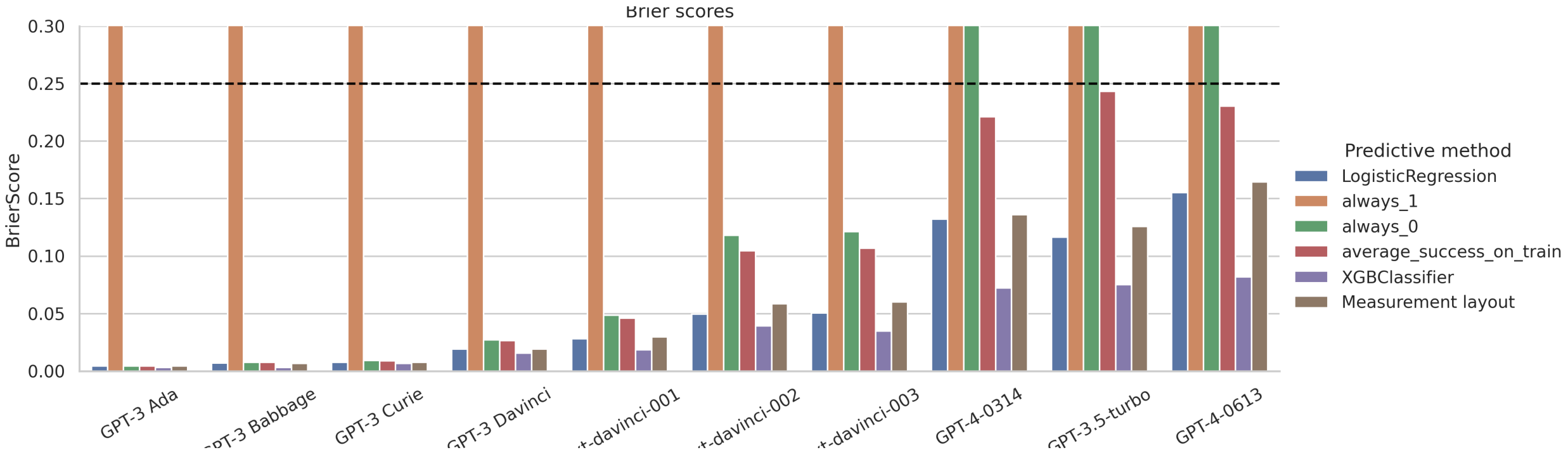


Build measurement layouts for each LLM

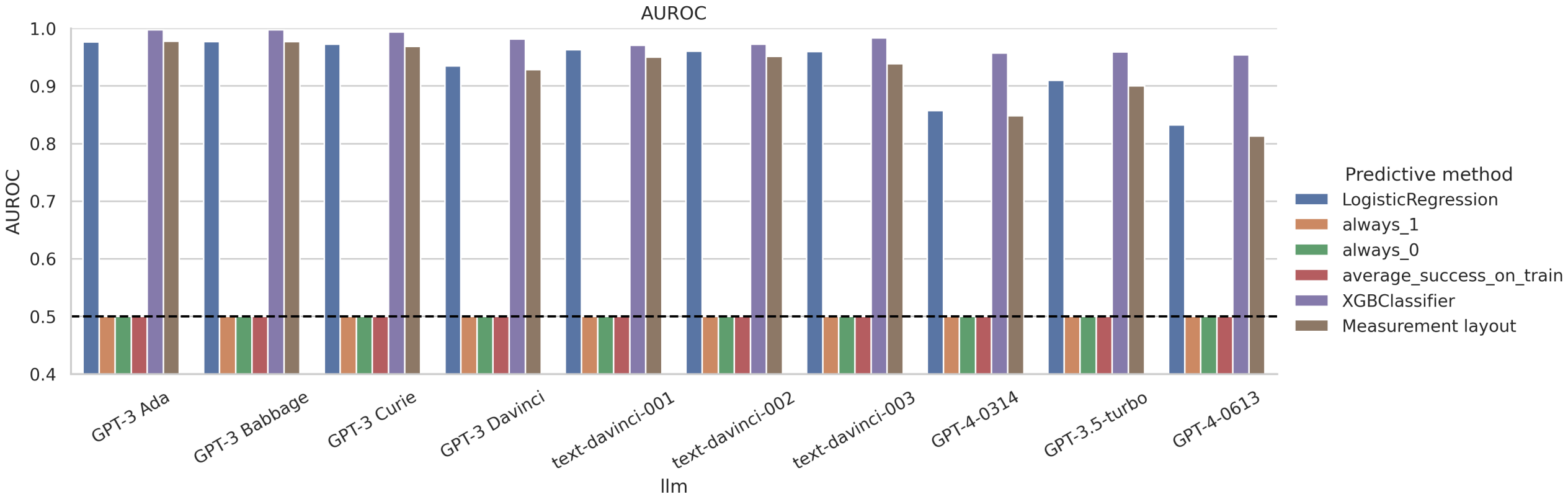




Prediction accuracy



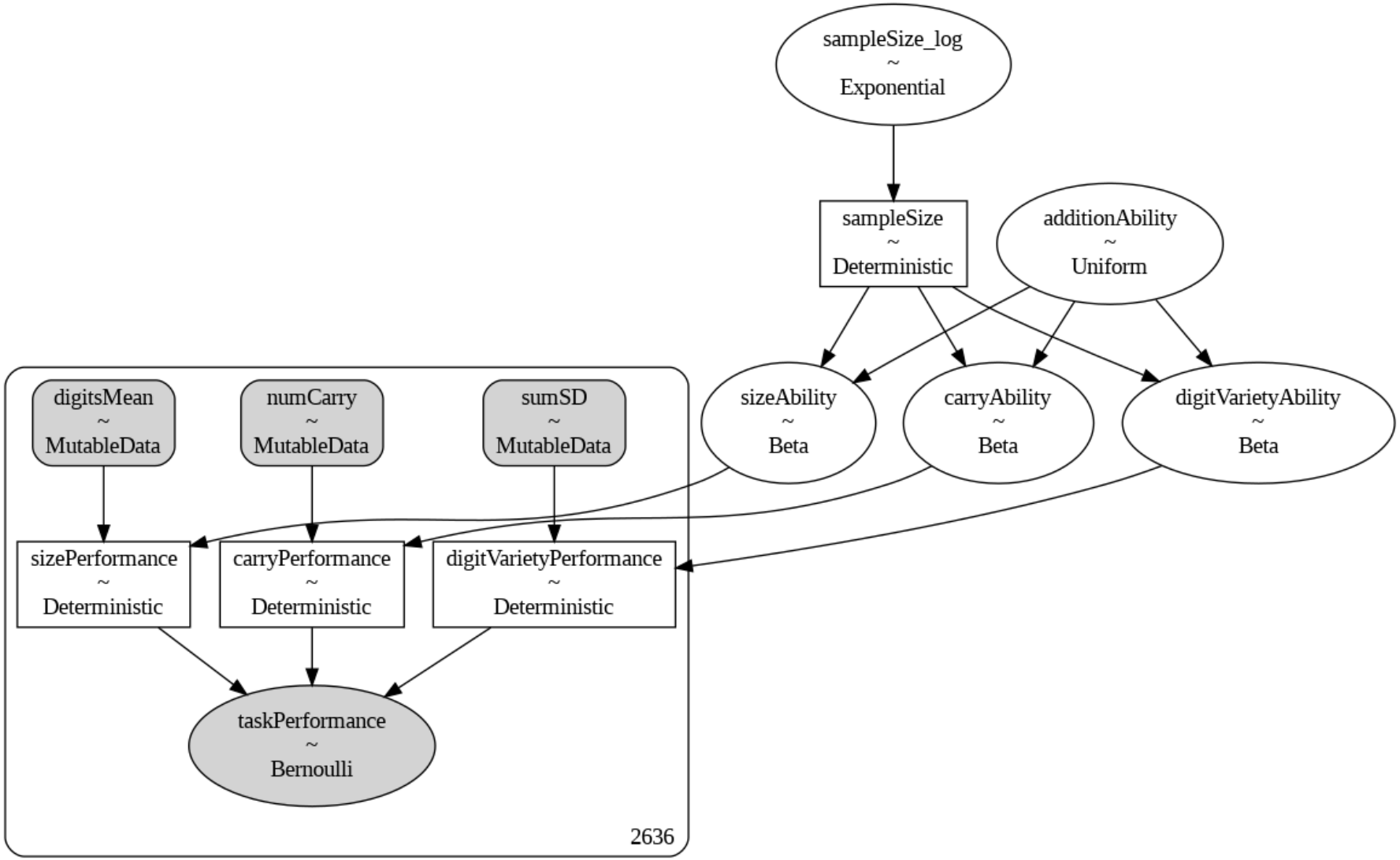
Prediction accuracy



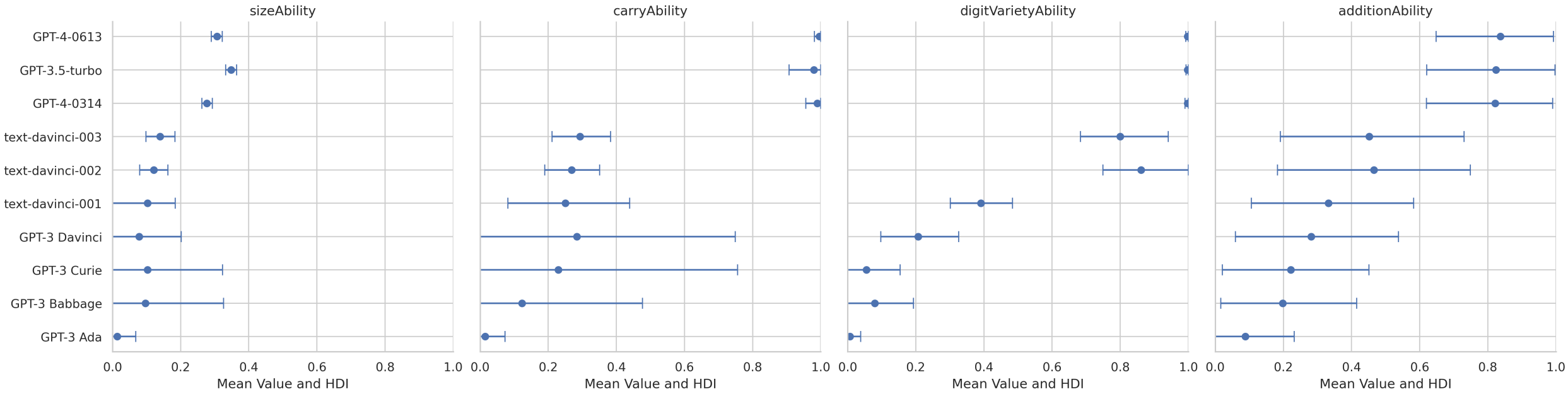
Hierarchical abilities

- We can think of size ability, carry ability and digit variety ability as indicators or markers of a more general addition ability
- Measurement layouts allows us to model this relationship

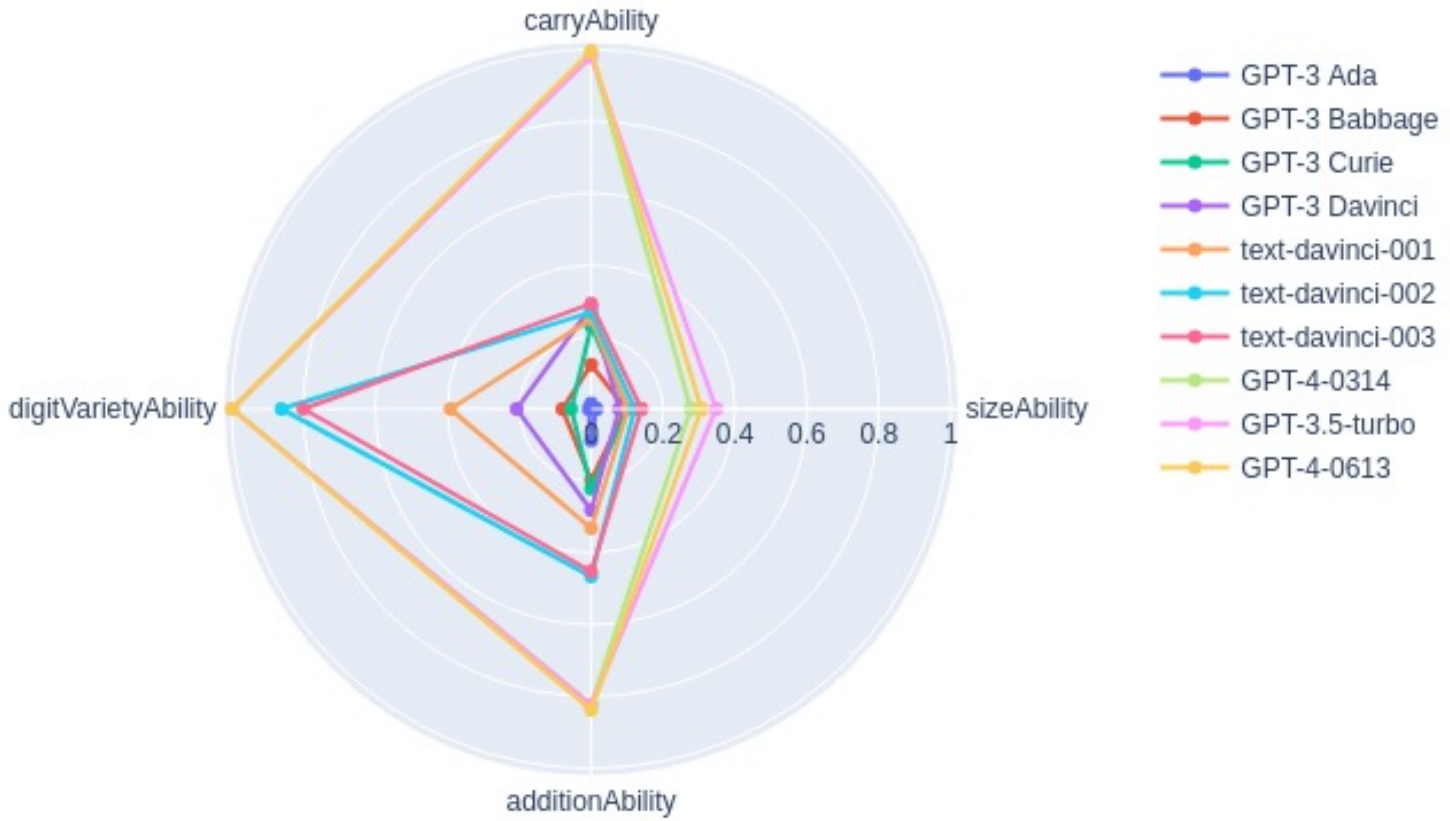
Hierarchical abilities



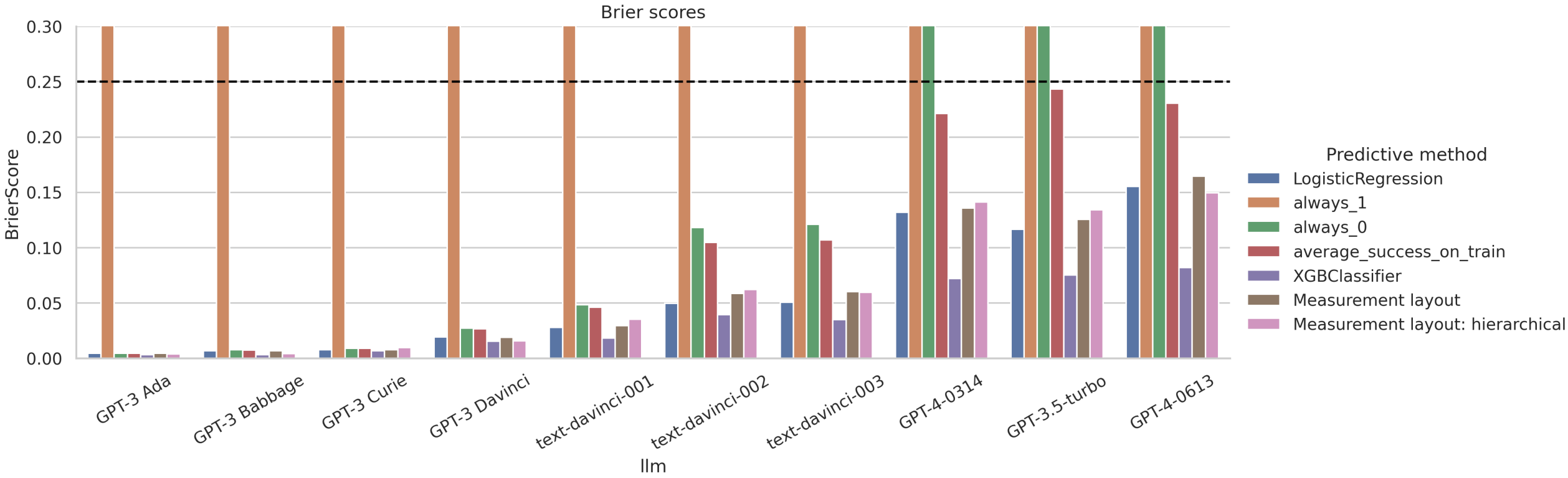
Hierarchical abilities



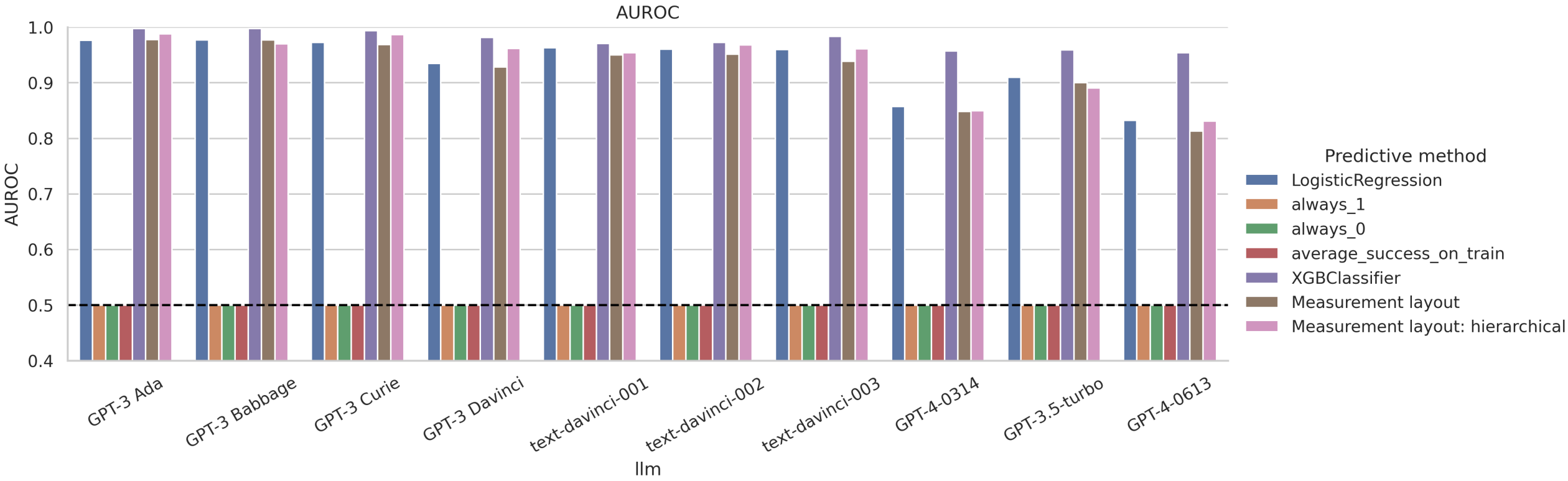
Hierarchical abilities



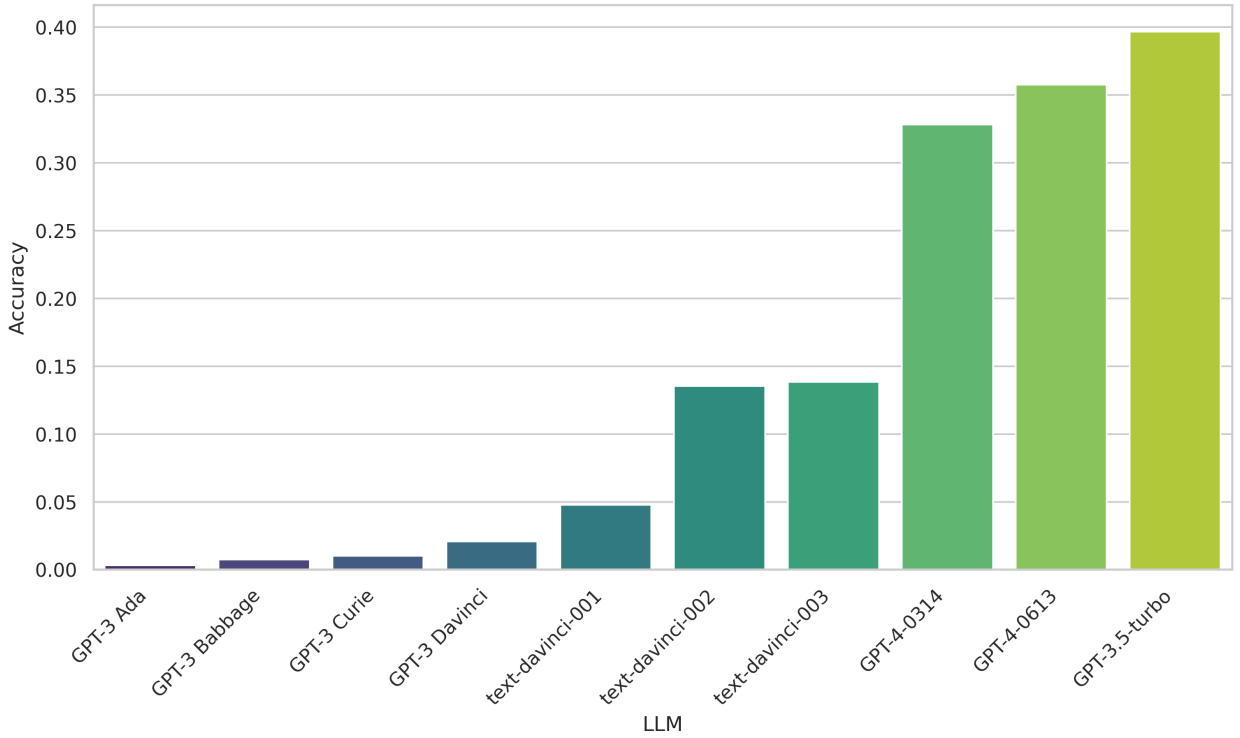
Hierarchical abilities: predictive accuracy

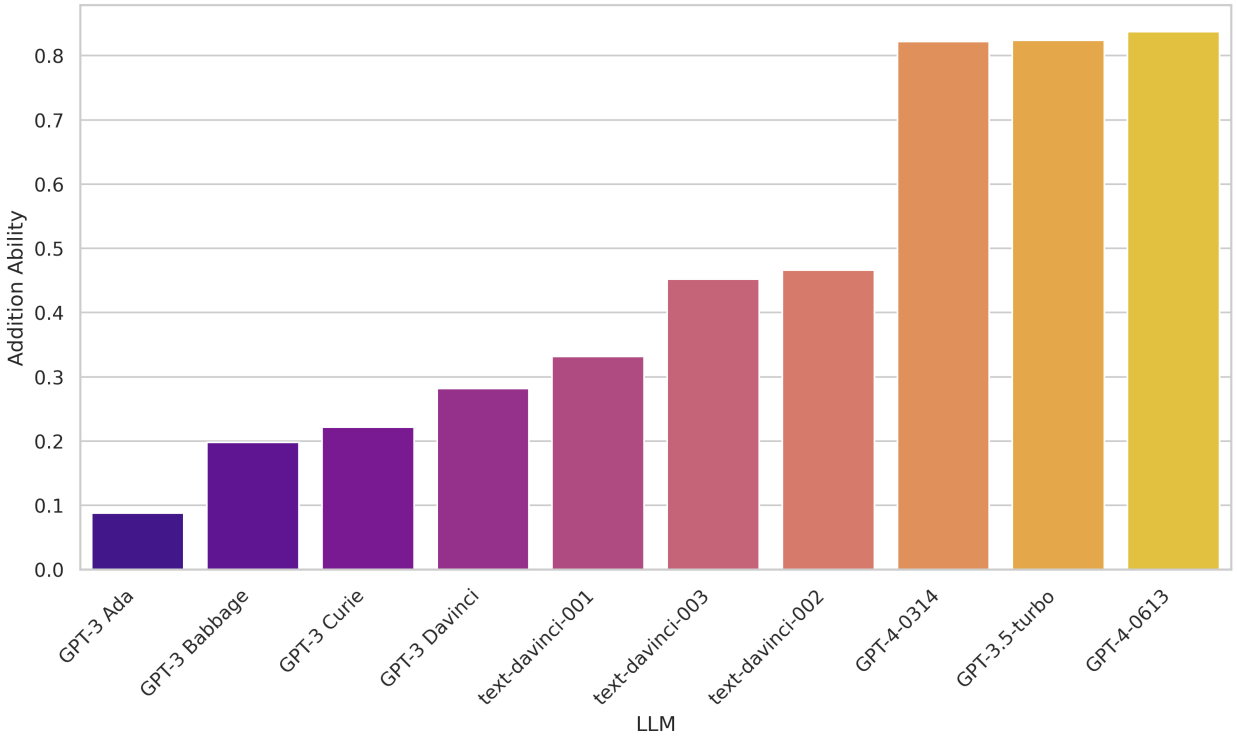


Hierarchical abilities: predictive accuracy



Aggregate accuracy and general addition ability





Session summary

- Benchmark characteristics:
  - Instance-level data
  - Meta-features/demands
  - Capabilities
- Measurement layouts for LLMs
  - Prior/posterior predictive checks
  - Convergence metrics
- Measurement layout for each LLM
  - We can explain the aggregate performance using the learnt LLM abilities
- Measurement layouts for hierarchical abilities