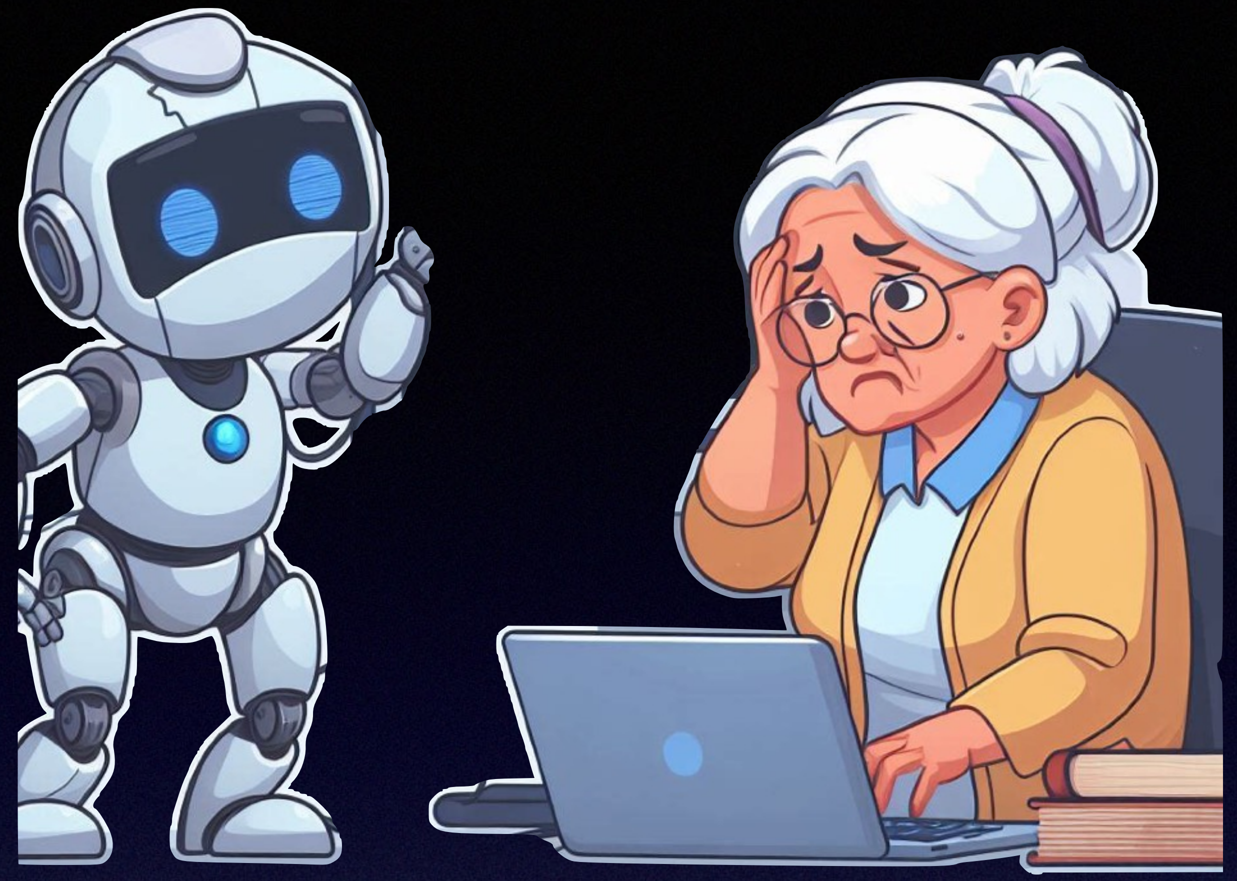
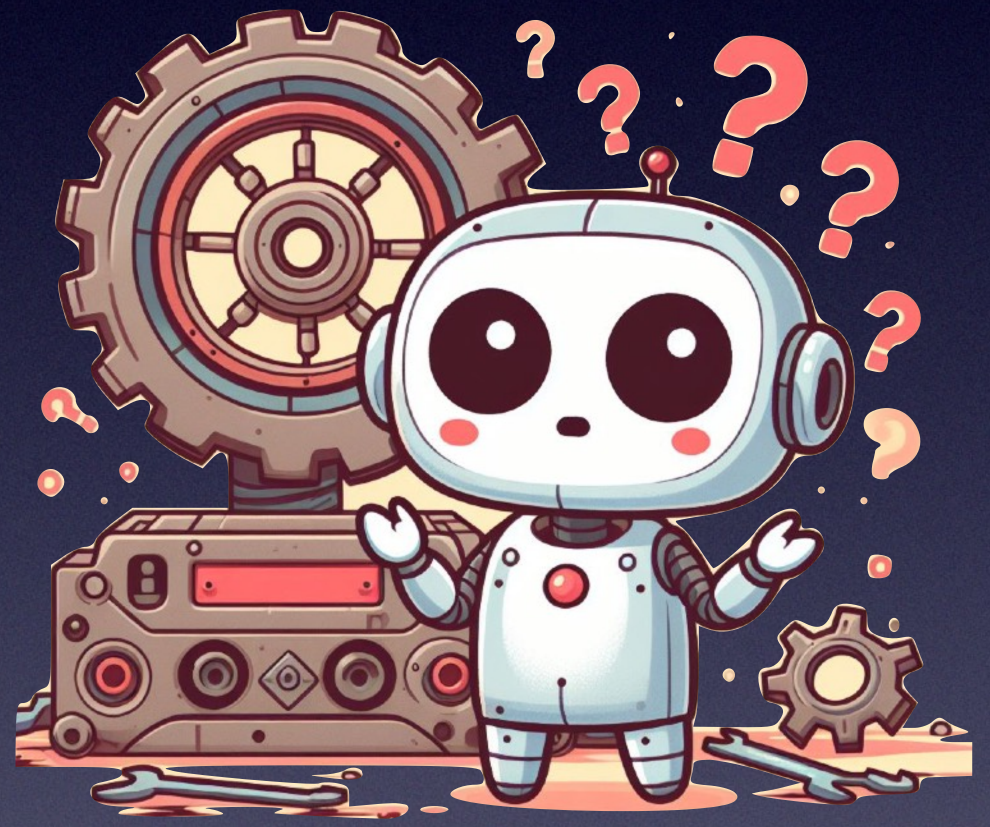


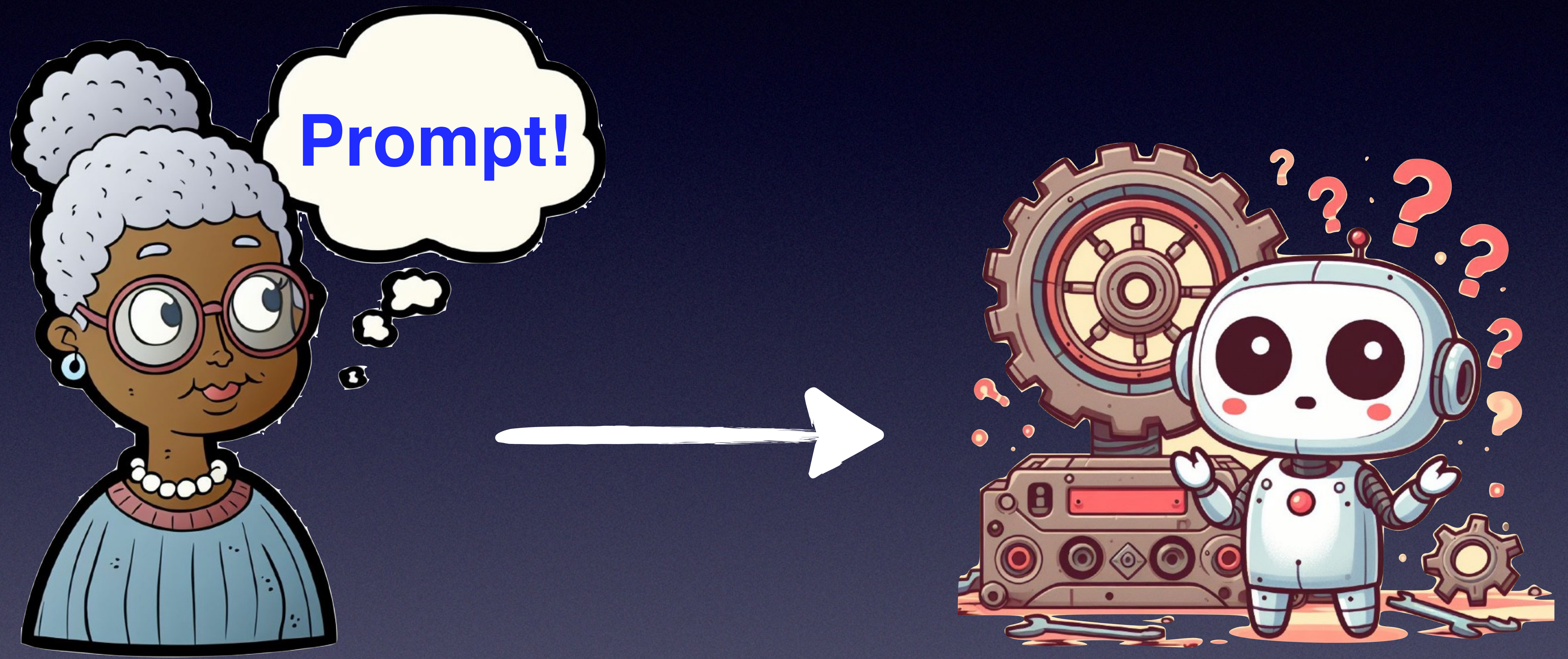
Last Mile Headaches
(Using LLMs for Code)

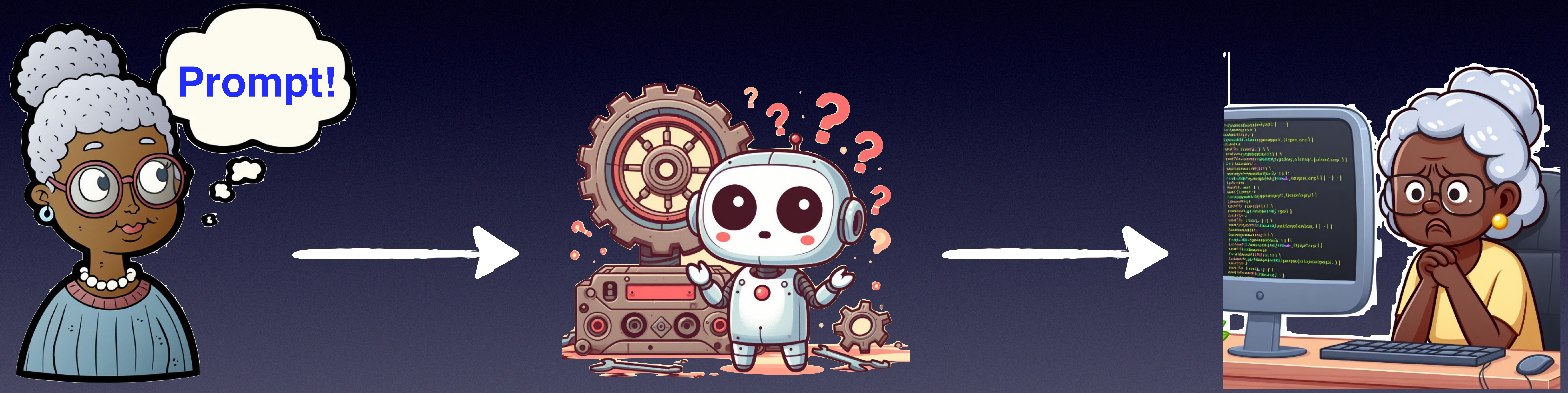
(With) **Claudio Spiess, David Gros**, Toufique Ahmed
Yuvraj Virk, Somesh Jha, Amin Alipour, Michael Pradel, Prem Devanbu

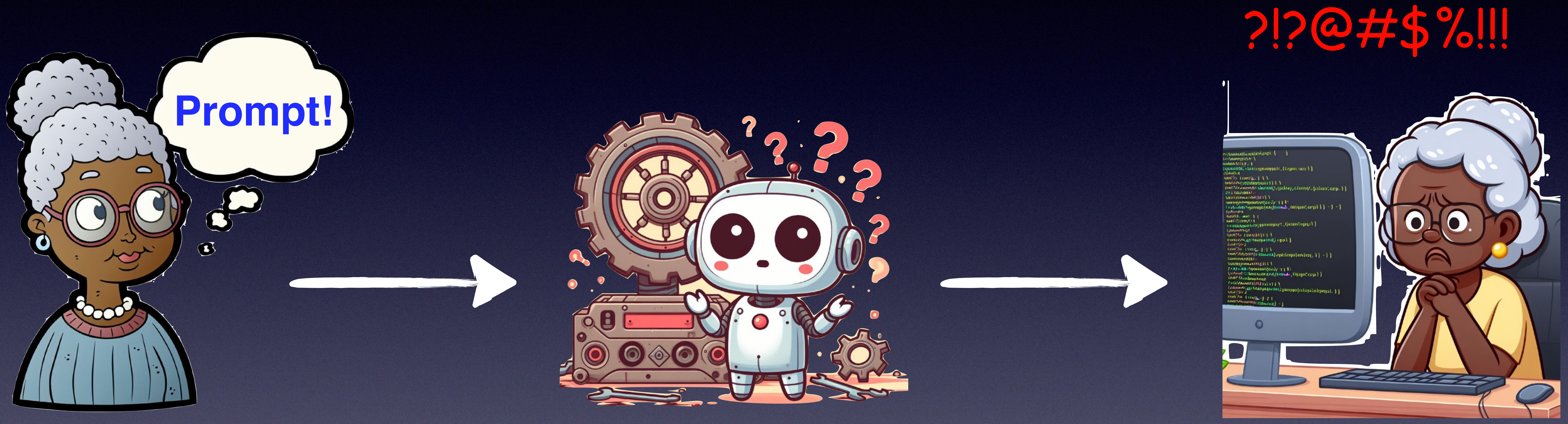
UC Davis *(ICSE 2025, to appear)*

Unterstützt von / Supported by
Alexander von Humboldt Stiftung / Foundation

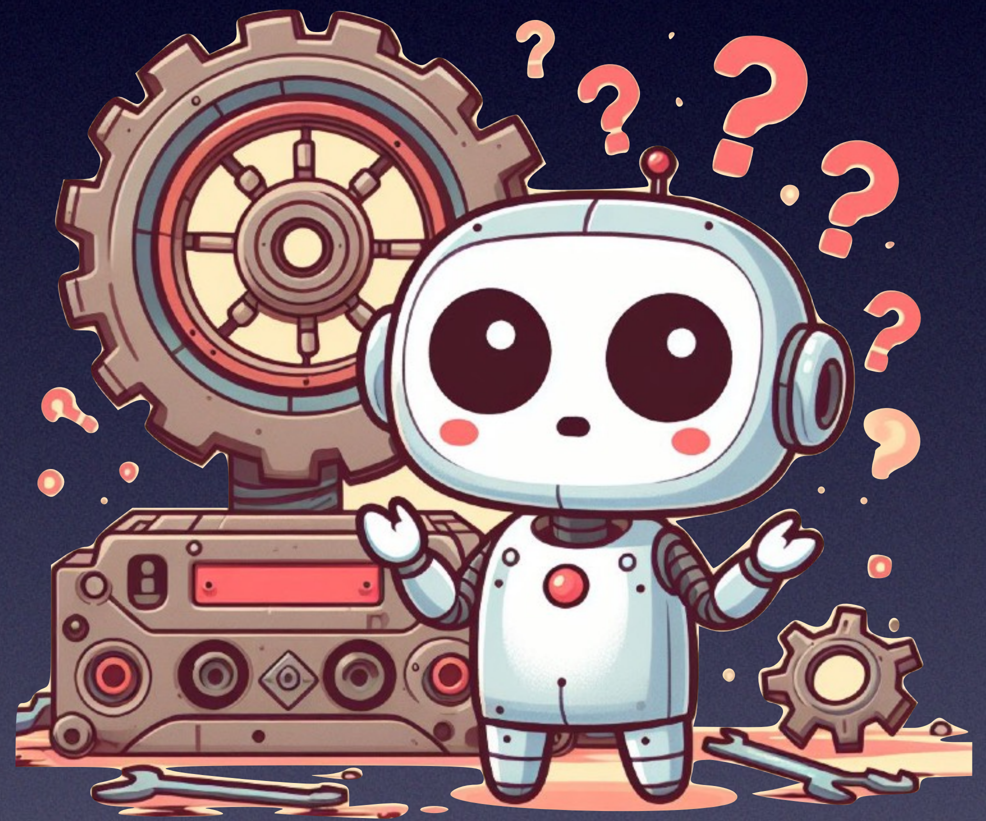



Prompt!


Prompt!

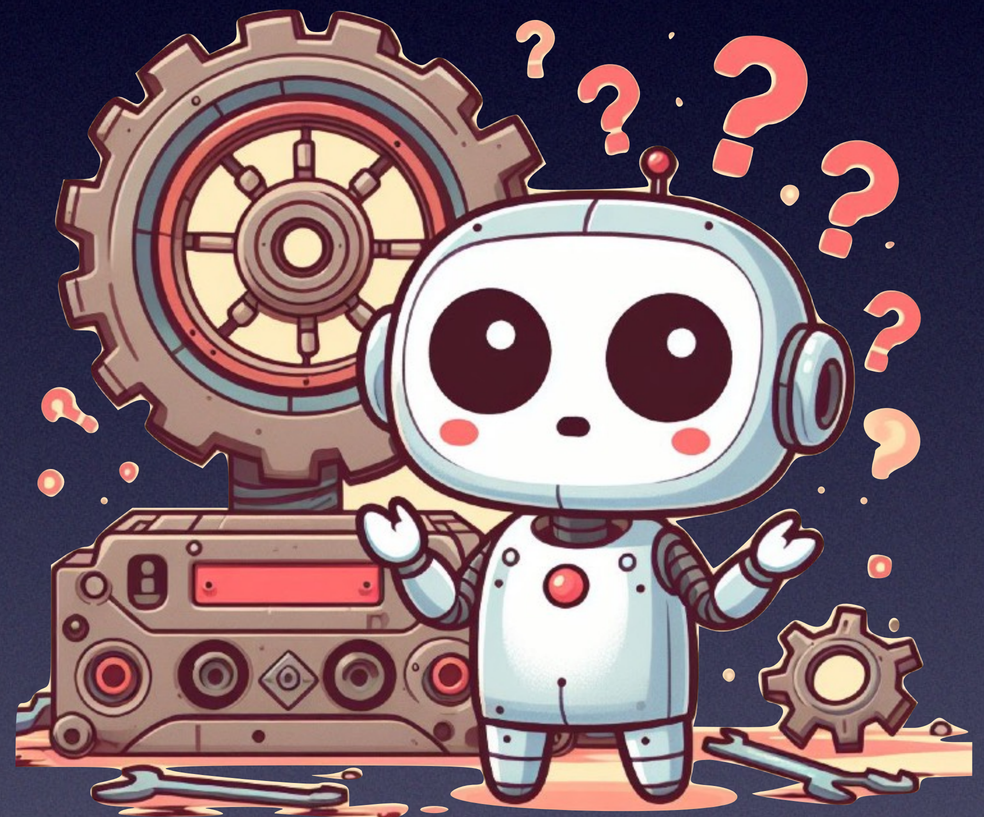

Prompt!
?!?@#$%!!!

First
Mile



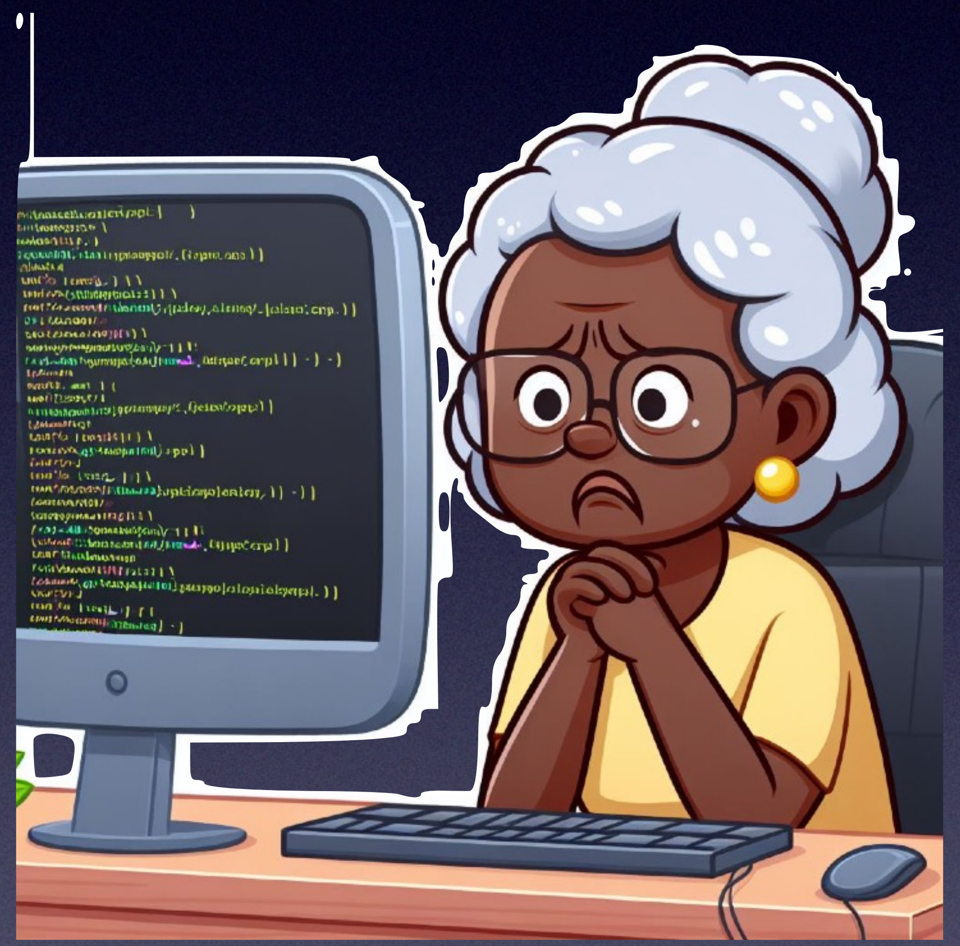
?!?@#$%!!!

First Mile

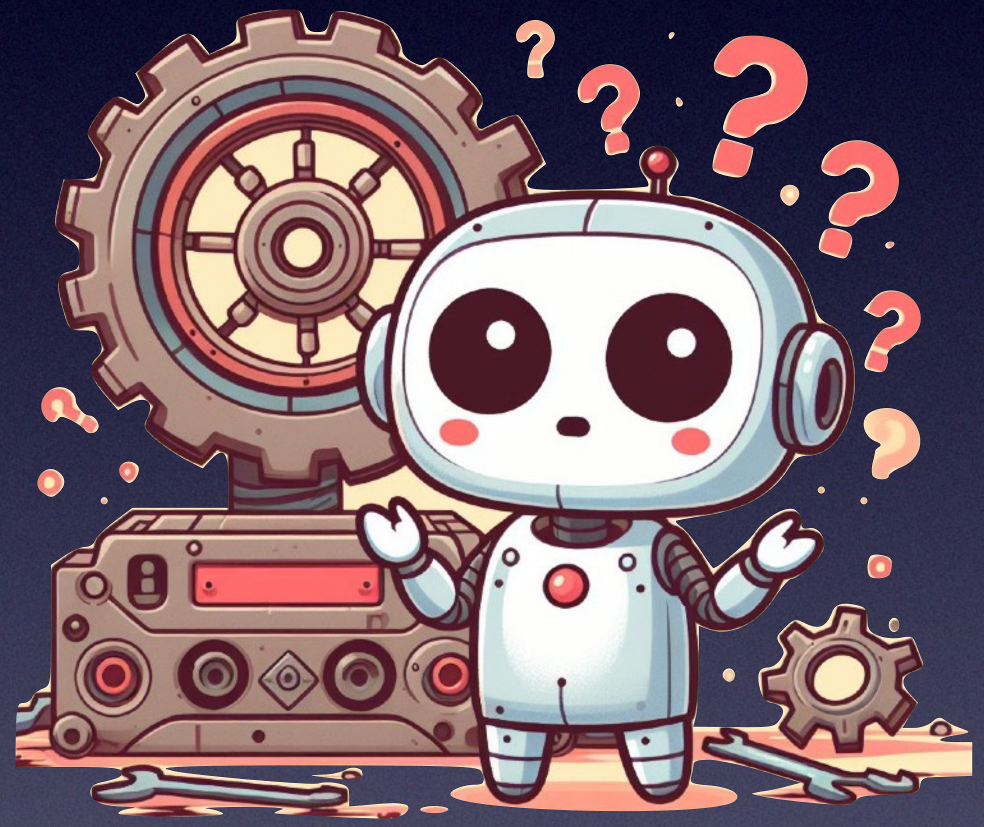


“Last” Mile

?!?@#$%!!!



First
Mile



“Last”
Mile

?!?@#$%!!!

.. and the rest of
the way?

.. and the rest of
the way?

.. and the rest of
the way?

Maintenance costs are heavy!

.. and the rest of
the way?

Maintenance costs are heavy!

..and we don’t know what they are…

GOOGLE / TECH / ARTIFICIAL INTELLIGENCE

More than a quarter of new code at Google is generated by AI / AI is hugely important to Google's products, and it sounds like the company relies on it internally, too.

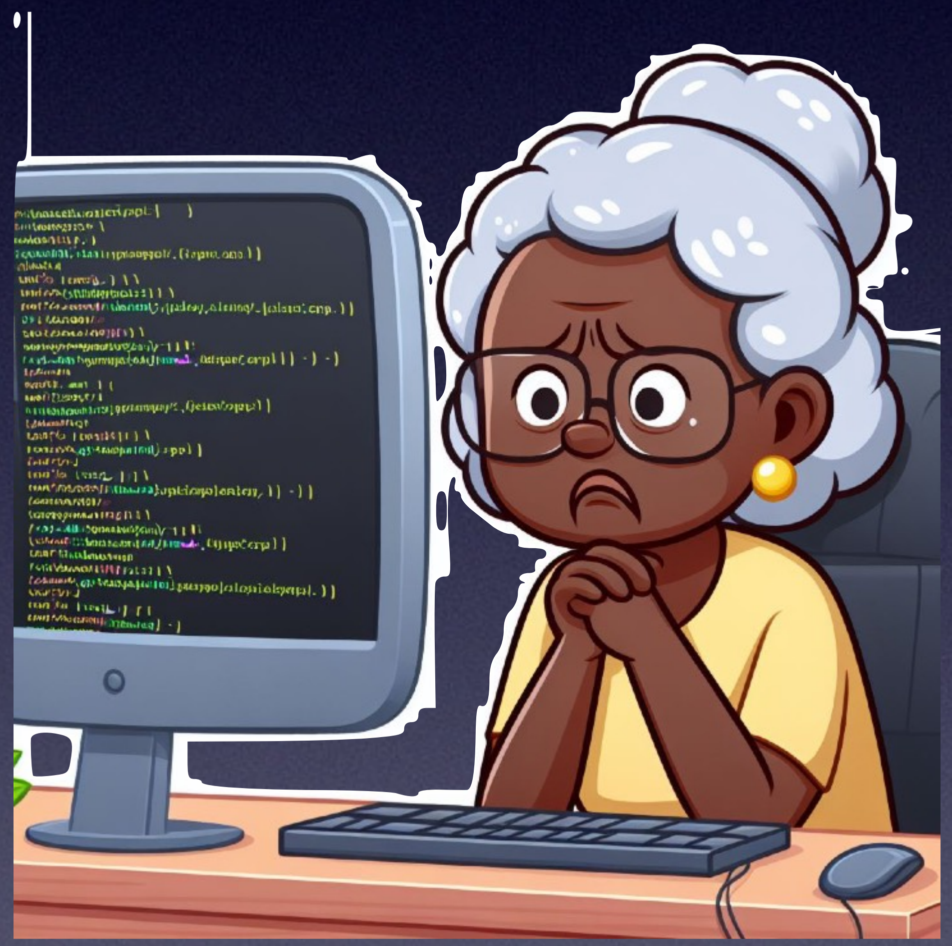
By Jay Peters, a news editor who writes about technology, video games, and virtual worlds. He's submitted several accepted emoji proposals to the Unicode Consortium.

Oct 29, 2024, 5:05 PM EDT

Last Mile can be problematic

(L2R Completion: CodeGen2-16B, Codex, GPT-3.5)

?!?@#$%!!!



Last Mile can be problematic

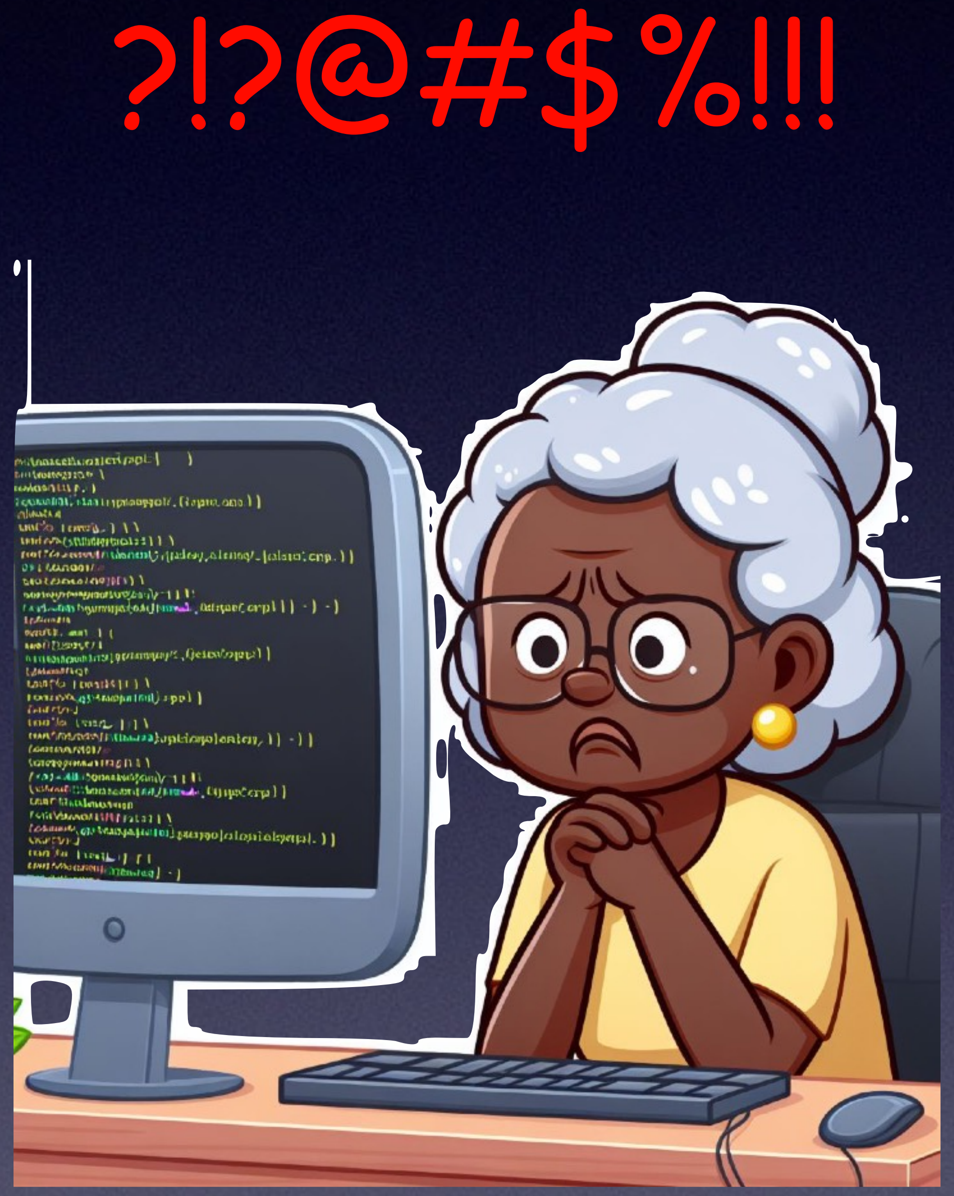
(L2R Completion: CodeGen2-16B, Codex, GPT-3.5)

?!?@#$%!!!

- Code Completion (DyPyBench)

Last Mile can be problematic

(L2R Completion: CodeGen2-16B, Codex, GPT-3.5)

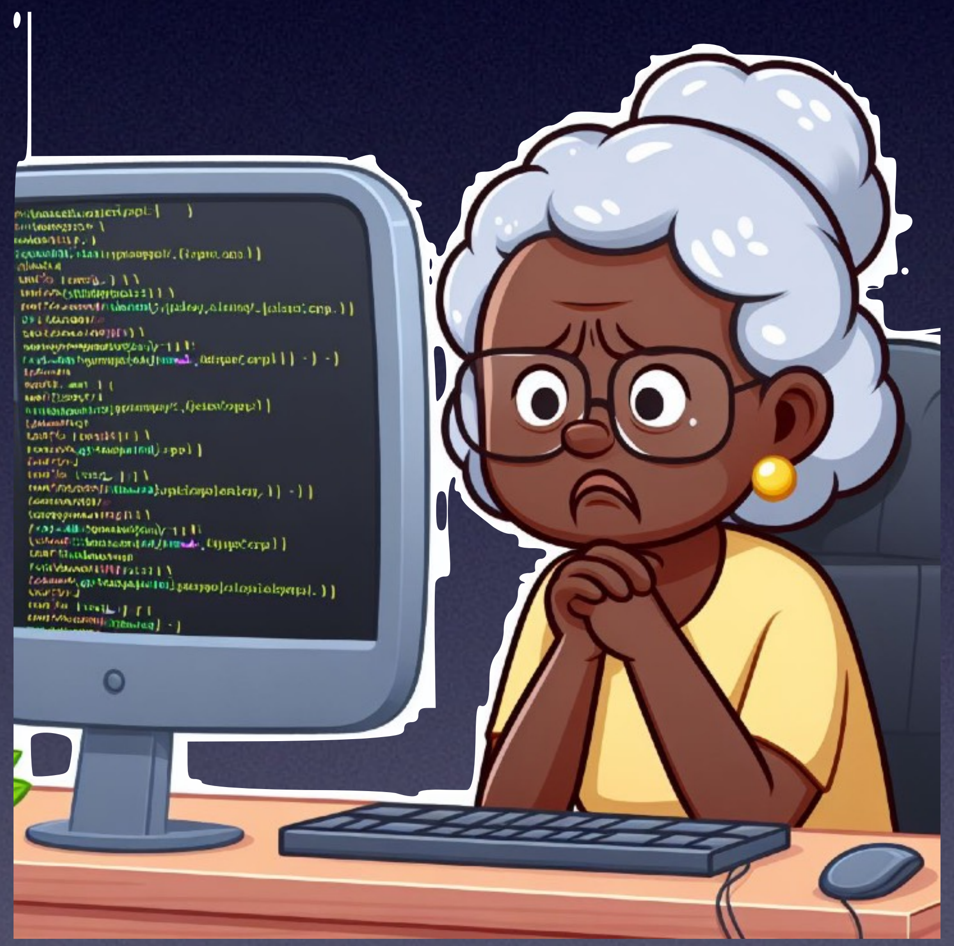


- Code Completion (DyPyBench)
  - ✓ *Correctness around 30%*

Last Mile can be problematic

(L2R Completion: CodeGen2-16B, Codex, GPT-3.5)

?!?@#$%!!!

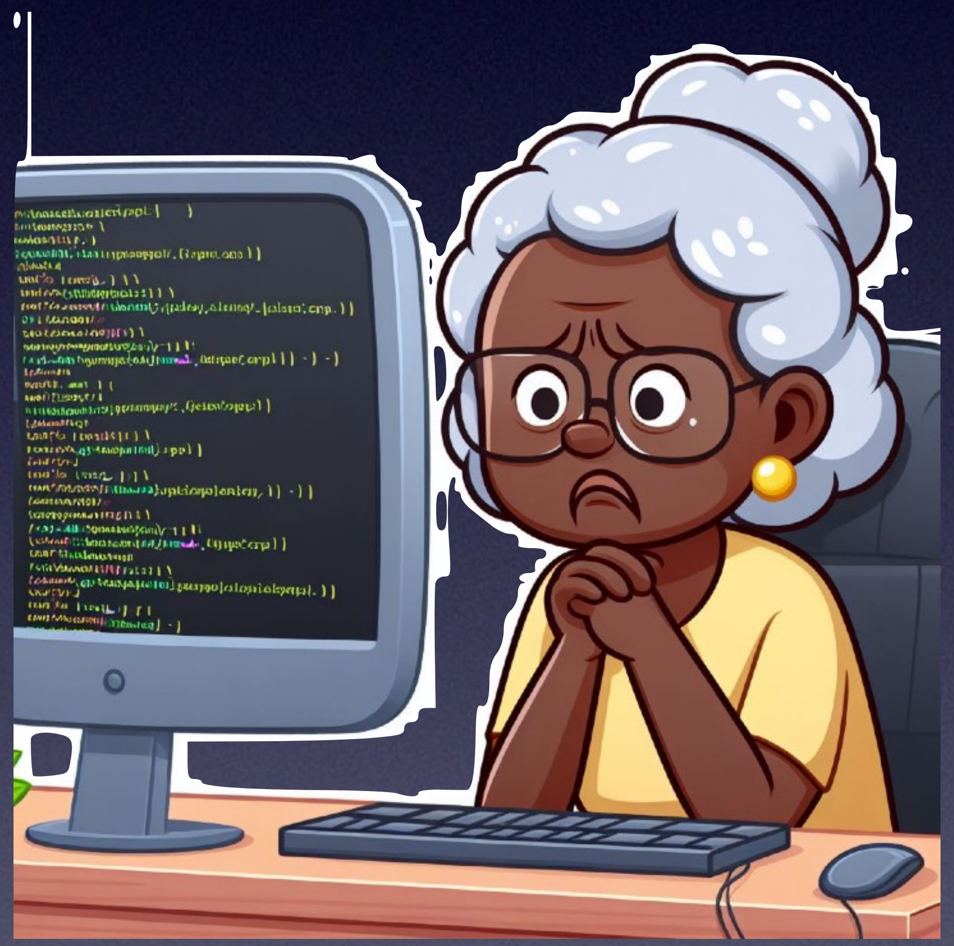


- Code Completion (DyPyBench)
  - ✓ *Correctness around 30%*
- Buggy/Fixed Code (SStubs4J)

Last Mile can be problematic

(L2R Completion: CodeGen2-16B, Codex, GPT-3.5)

?!?@#$%!!!

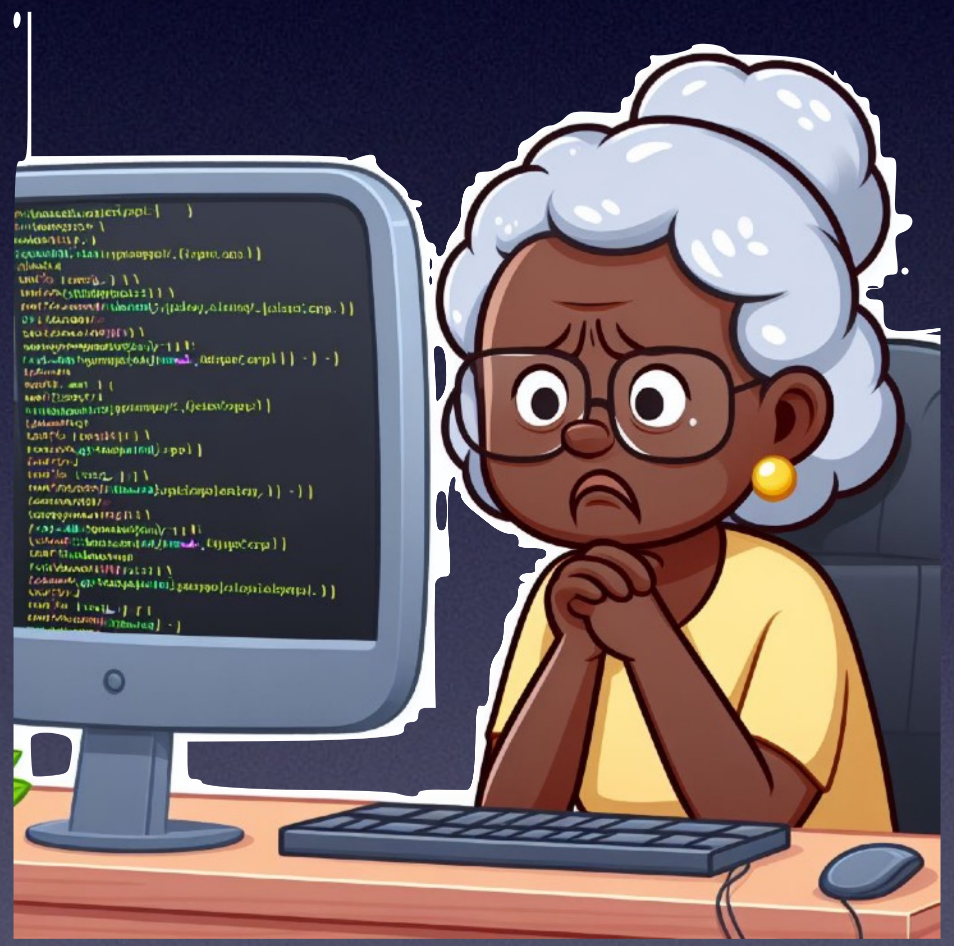


- Code Completion (DyPyBench)
  - ✓ *Correctness around 30%*
- Buggy/Fixed Code (SStubs4J)
  - ✓ *Correctness 1-27%*

Last Mile can be problematic

(L2R Completion: CodeGen2-16B, Codex, GPT-3.5)

?!?@#$%!!!



- Code Completion (DyPyBench)
  - ✓ *Correctness around 30%*
- Buggy/Fixed Code (SStubs4J)
  - ✓ *Correctness 1-27%*

Yes, *L2R* is a difficult setting

(FIM, SAFIM easier)

Another Evaluation: Simple Stupid bugs
(Sutton & Karampatsis, 2020)

- Single-statement bug fixes from project version history ~ 17K examples after cleaning.
- Collected from about 1000 projects
- Median fix-time, about 4 days..but sometimes much longer.
- Widely used dataset, entire conference track devoted to it *(MSR 2021)*

*All samples in dataset used
were fixed before
LLM training data was gathered.*

Methodology

Methodology

RQ: Does Codex repeat human mistakes?

Methodology

RQ: Does Codex repeat human mistakes?

Using ***17K "Simple, Stupid Bugs" (SStuB)***

Methodology

RQ: Does Codex repeat human mistakes?

Using ***17K "Simple, Stupid Bugs" (SStuB)***

1. Find the SStuB introduction in version history.

Methodology

RQ: Does Codex repeat human mistakes?

Using ***17K "Simple, Stupid Bugs" (SStuB)***

1. Find the SStuB introduction in version history.

2. Use the prefix to the SStuB as prompt, and..

Methodology

RQ: Does Codex repeat human mistakes?

Using ***17K "Simple, Stupid Bugs" (SStuB)***

1. Find the SStuB introduction in version history.

2. Use the prefix to the SStuB as prompt, and..

3. Ask LLM to prompt.

Methodology

RQ: Does Codex repeat human mistakes?

Using ***17K "Simple, Stupid Bugs" (SStuB)***

1. Find the SStuB introduction in version history.

2. Use the prefix to the SStuB as prompt, and..

3. Ask LLM to prompt.

4. Classify resulting completion:

Bug? Patch? Other?

Example

```
            g2d.setColor(tabFillColor);
            g2d.fill(shaper.reset().doRect(boundsX, topY + shape.path.deltaY(1),
                    boundsWidth, paintBorder.top).getShape());

            //If the top of the border is a non-paint border, then the border is painted.
            if (
      Bug           paintBorder.top >= 1
                    paintBorder.top > 1      Fix
            ) {
                g2d.setColor(borderColor);
                final int topLine = topY + shape.path.deltaY(paintBorder.top - 1);
                g2d.draw(shaper.reset().doRect(boundsX, topLine, boundsWidth - 1,
                        1).getShape());
            }
        }
    }
```

Example

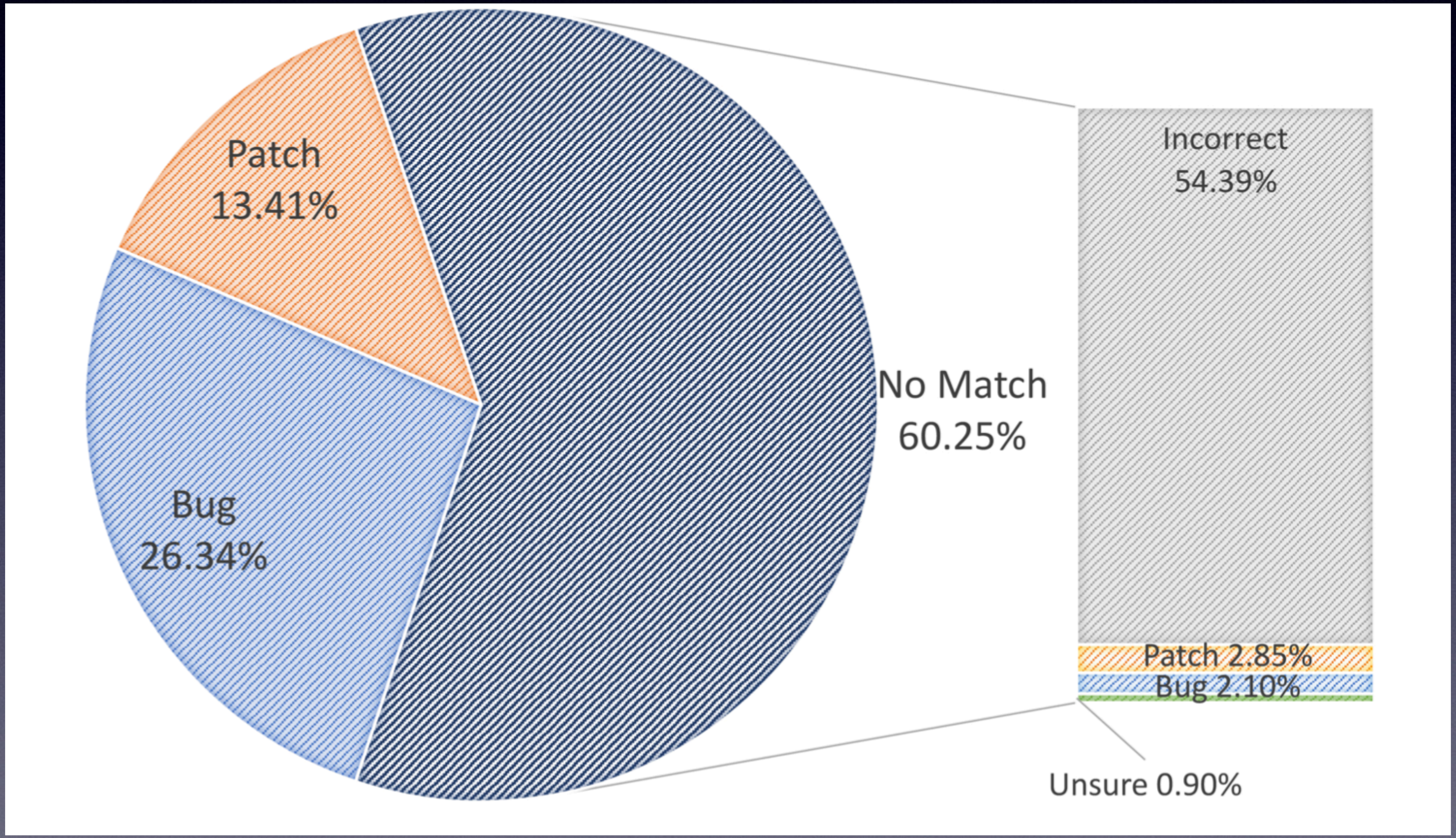
```
//If the top of the border is a non-paint border, then t
if (
    paintBorder.top >= 1
    paintBorder.top > 1
) {
    g2d.setColor(borderColor);
```

Bug

Fix

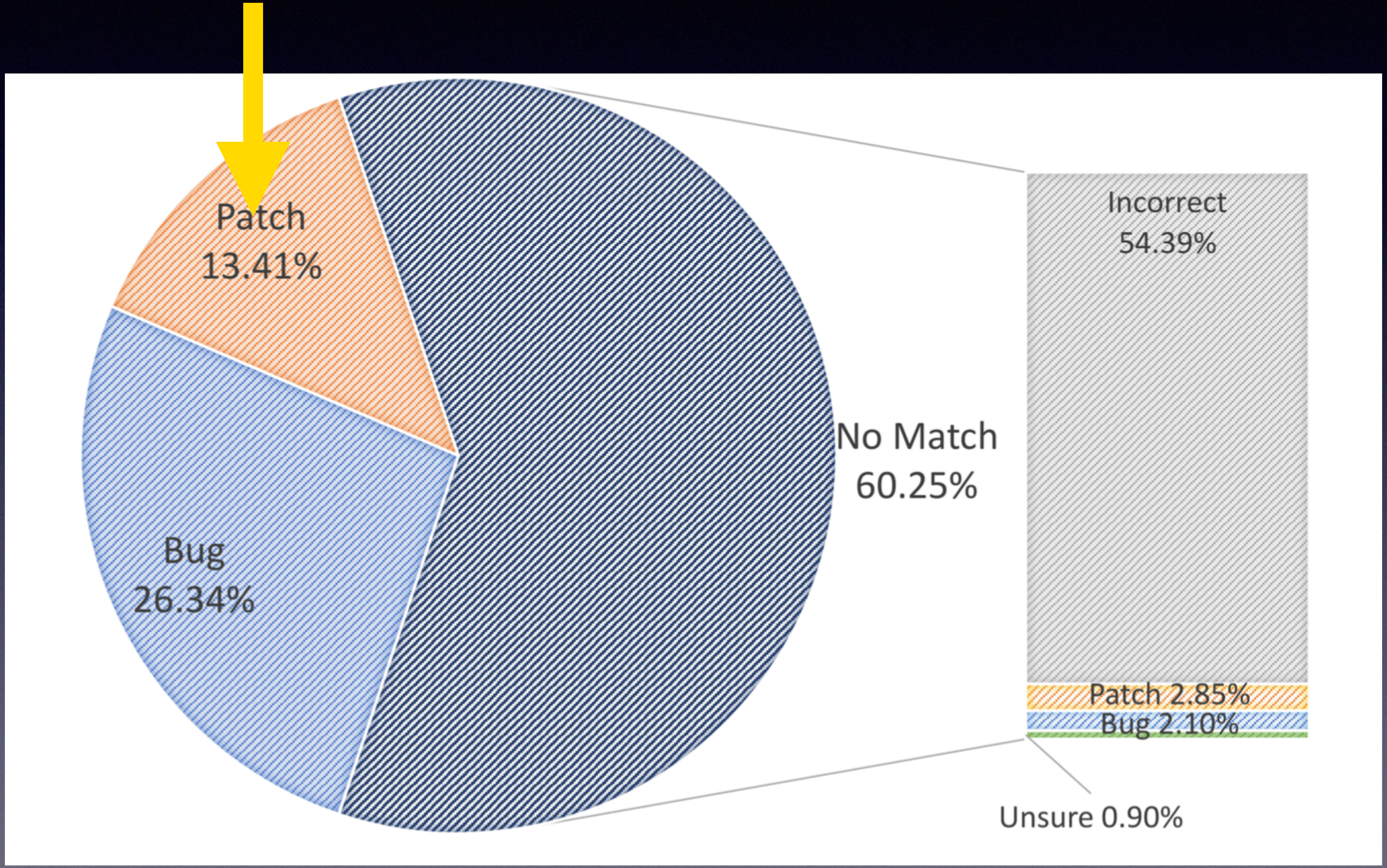
Result

Result


Patch
13.41%
Bug
26.34%
No Match
60.25%
Incorrect
54.39%
Patch 2.85%
Bug 2.10%
Unsure 0.90%

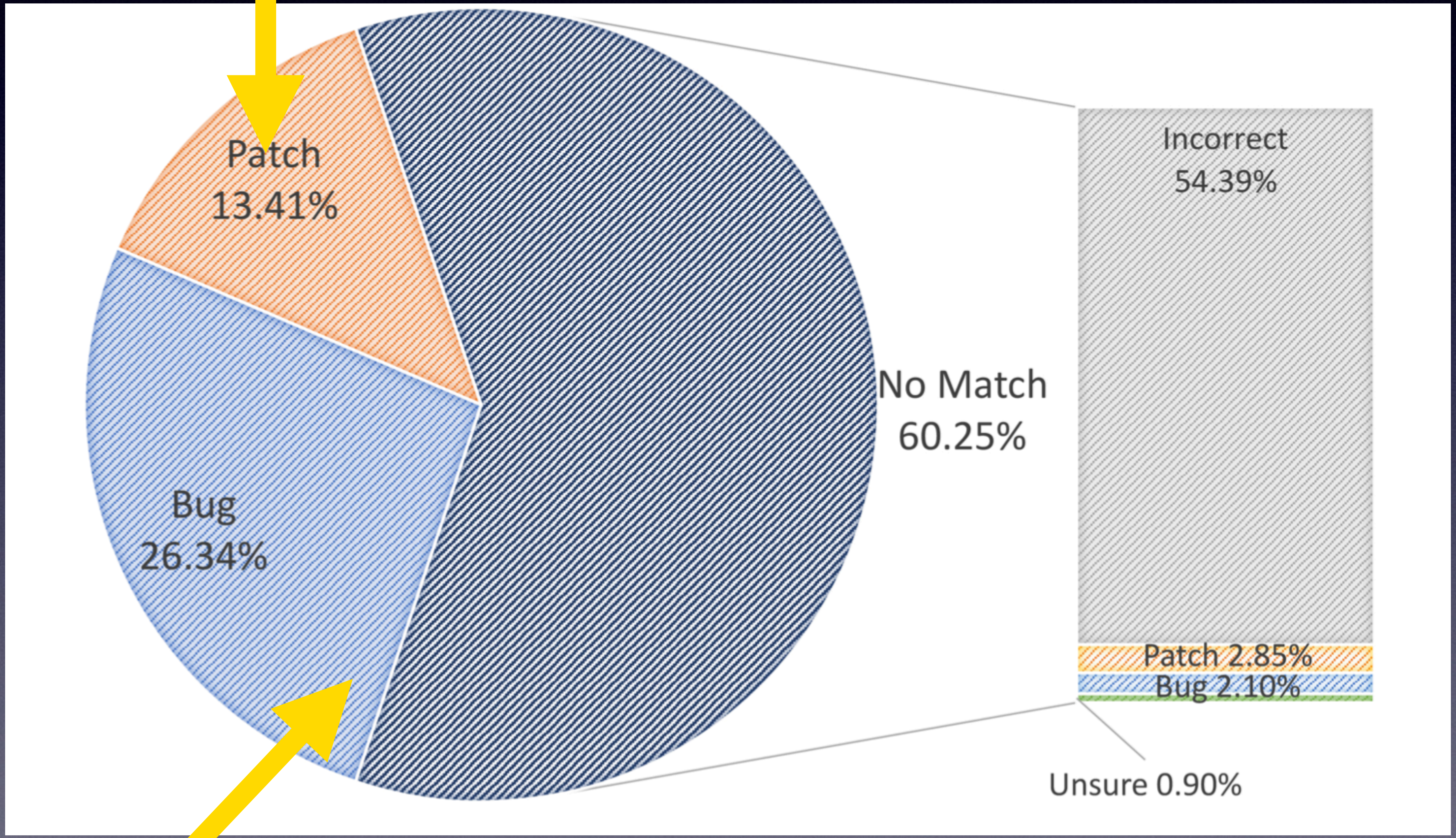
Result

Codex produces fixed code



Result

Codex produces fixed code

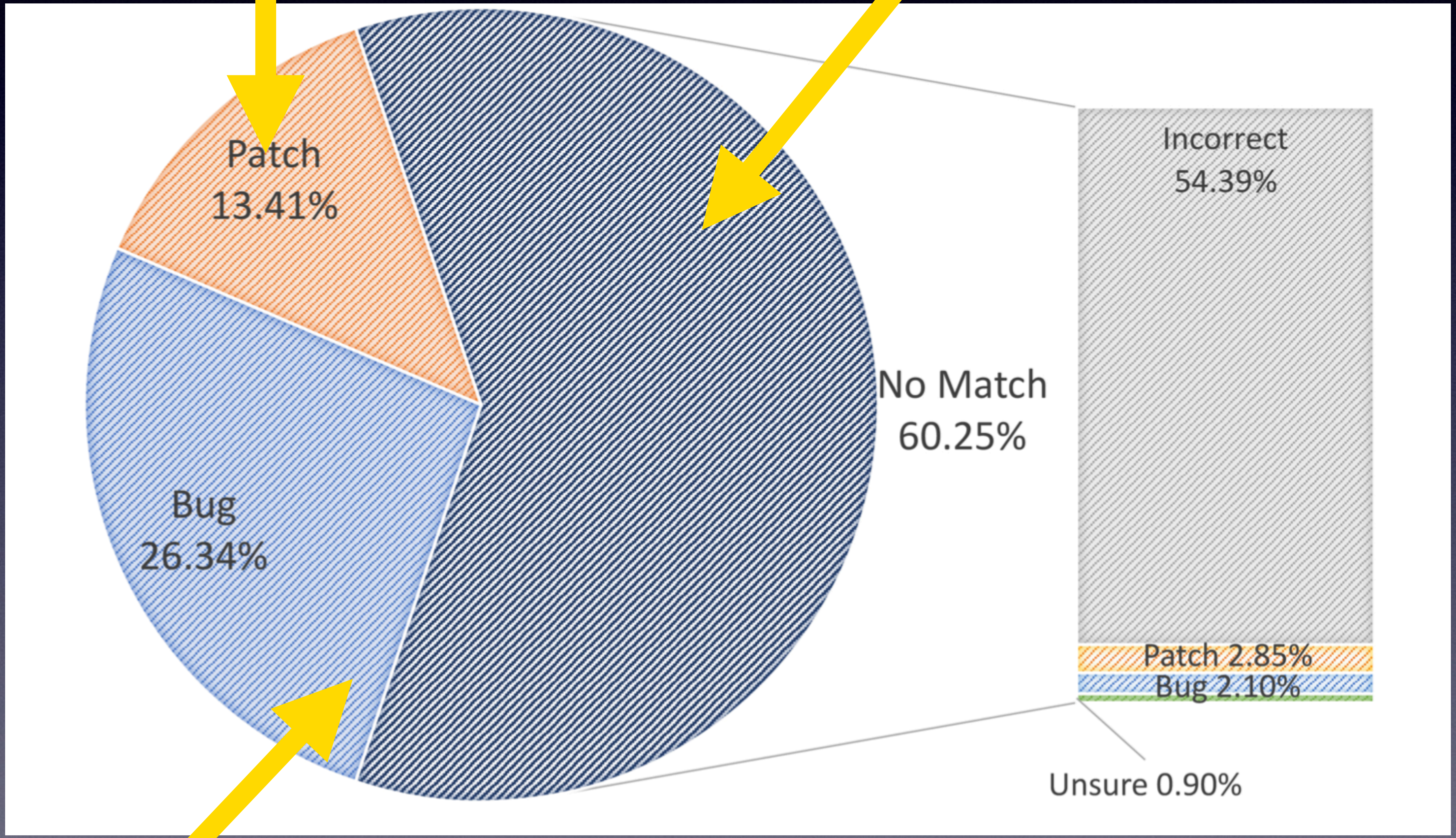


Codex produces buggy code TWICE as often

Result

Codex produces fixed code

Something else



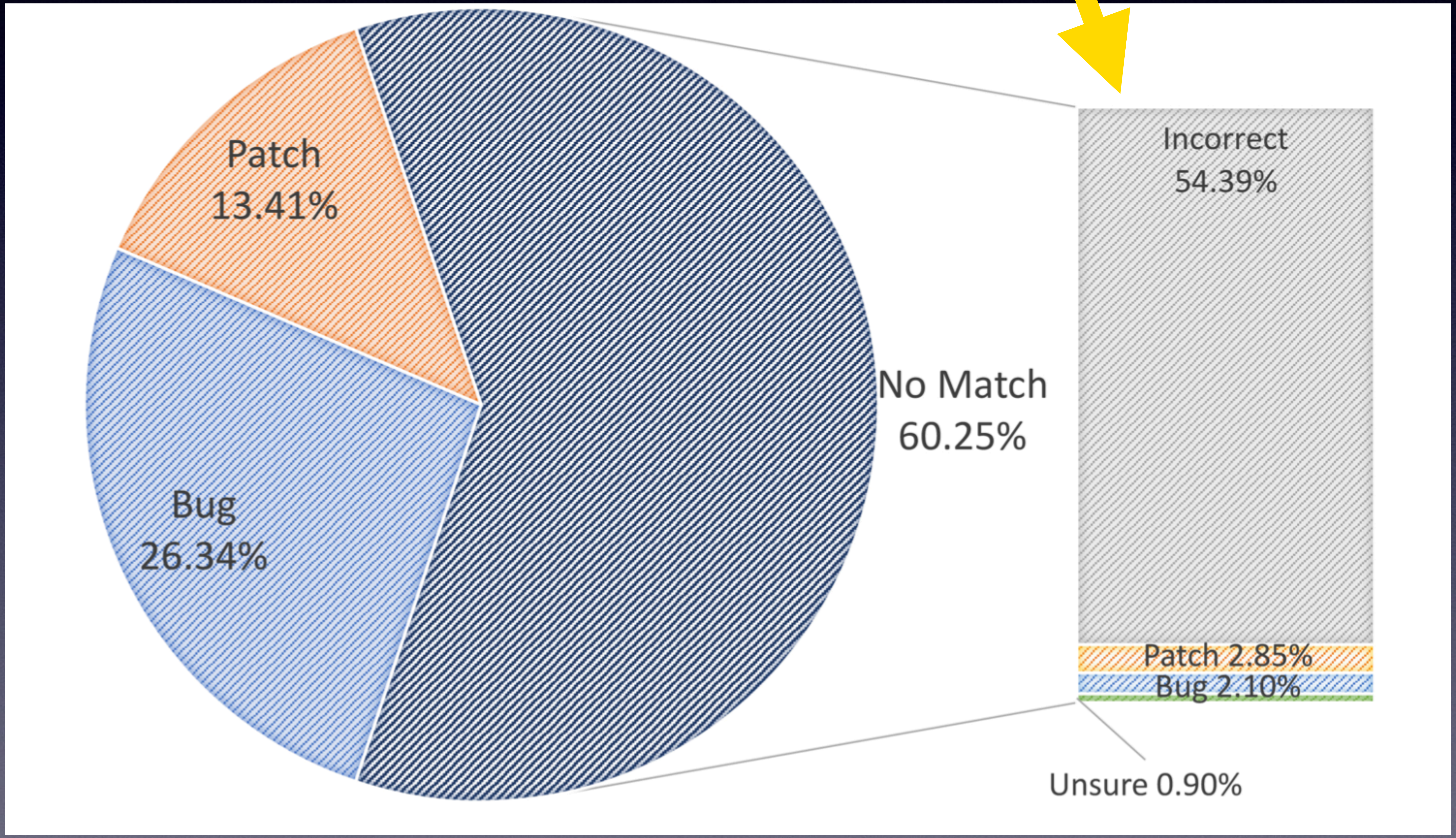
Codex produces buggy code TWICE as often

Result

Manual Review,
401 samples

Result

Manual Review,
401 samples



How “sticky” are Codex-generated bugs?

How “sticky” are Codex-generated bugs?

“Sticky” $\Longrightarrow$ Takes Longer to Fix. 🤔🤔

How “sticky” are Codex-generated bugs?

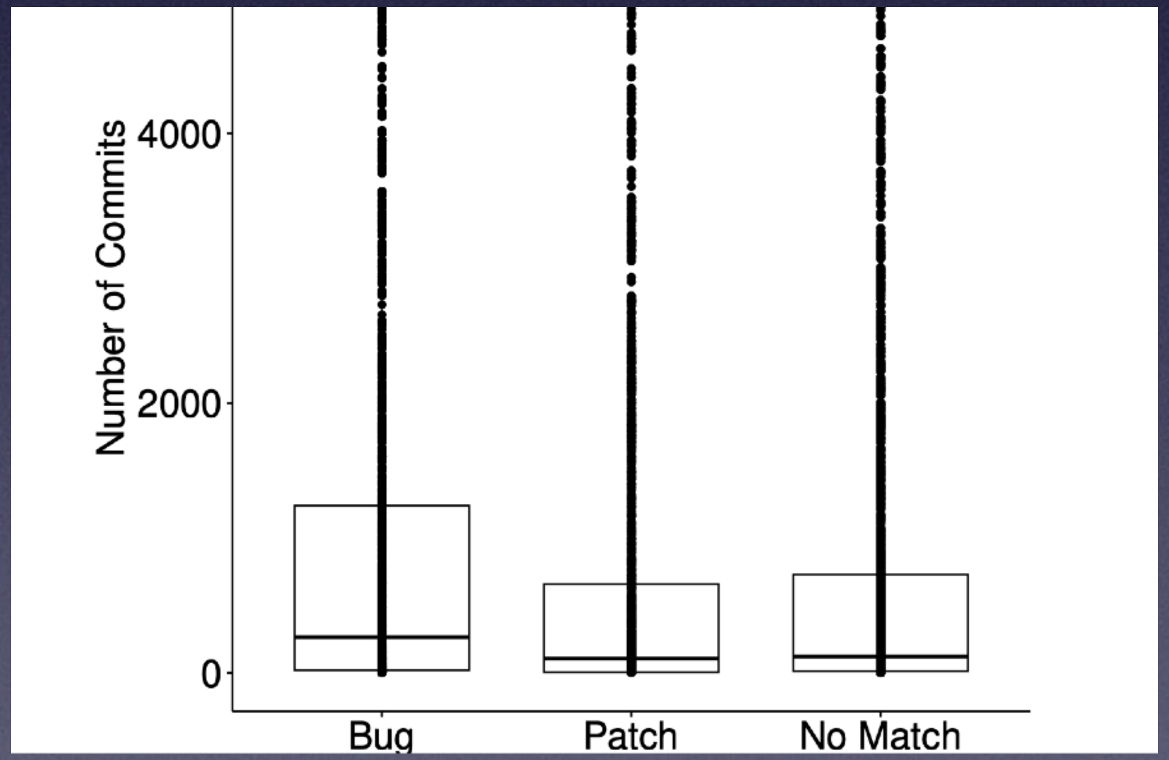
“Sticky” $\Longrightarrow$ Takes Longer to Fix. 🤔🤔

When Codex makes a mistake, did that bug stick around longer?

How “sticky” are Codex-generated bugs?

“Sticky” $\Longrightarrow$ Takes Longer to Fix. 🤔🤔

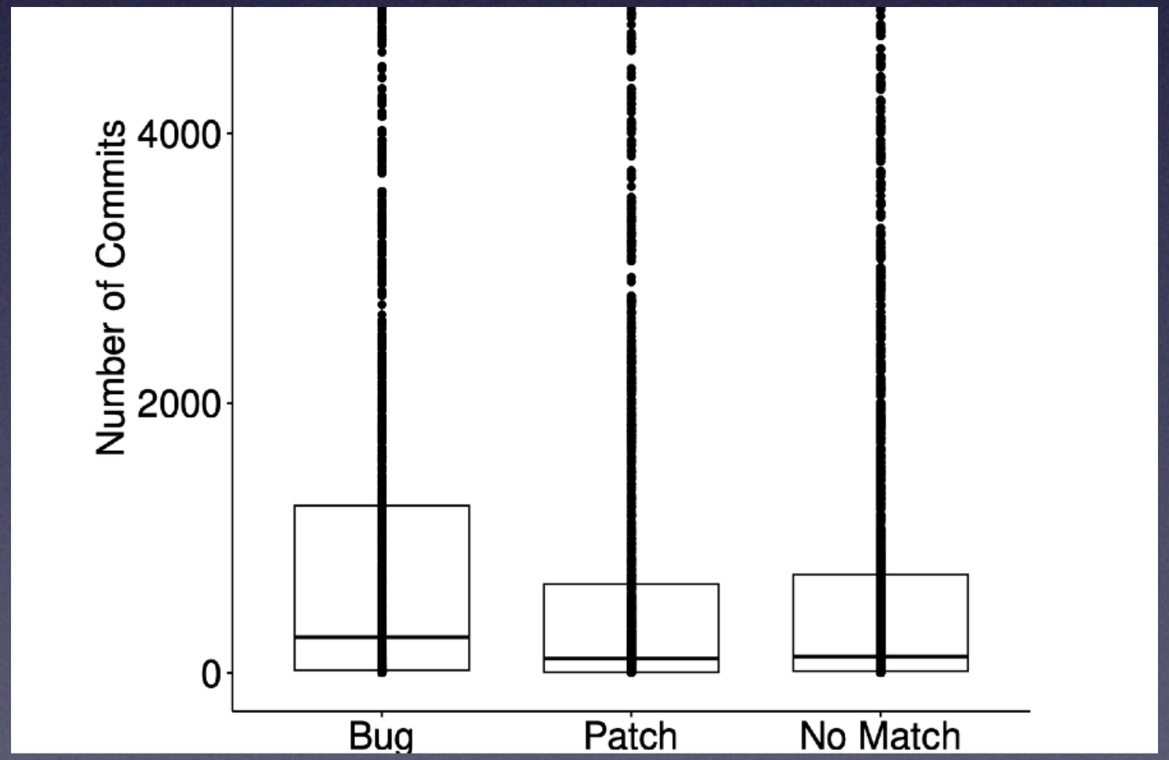
When Codex makes a mistake, did that bug stick around longer?



How “sticky” are Codex-generated bugs?

“Sticky” $\implies$ Takes Longer to Fix. 🤔🤔

When Codex makes a mistake, did that bug stick around longer?



More “Natural” Bugs $\implies$ Longer to Fix ???

Overall Theme

Overall Theme

- LLMs: Codex, GPT-x, etc are now widely used to generate code and related artifacts.

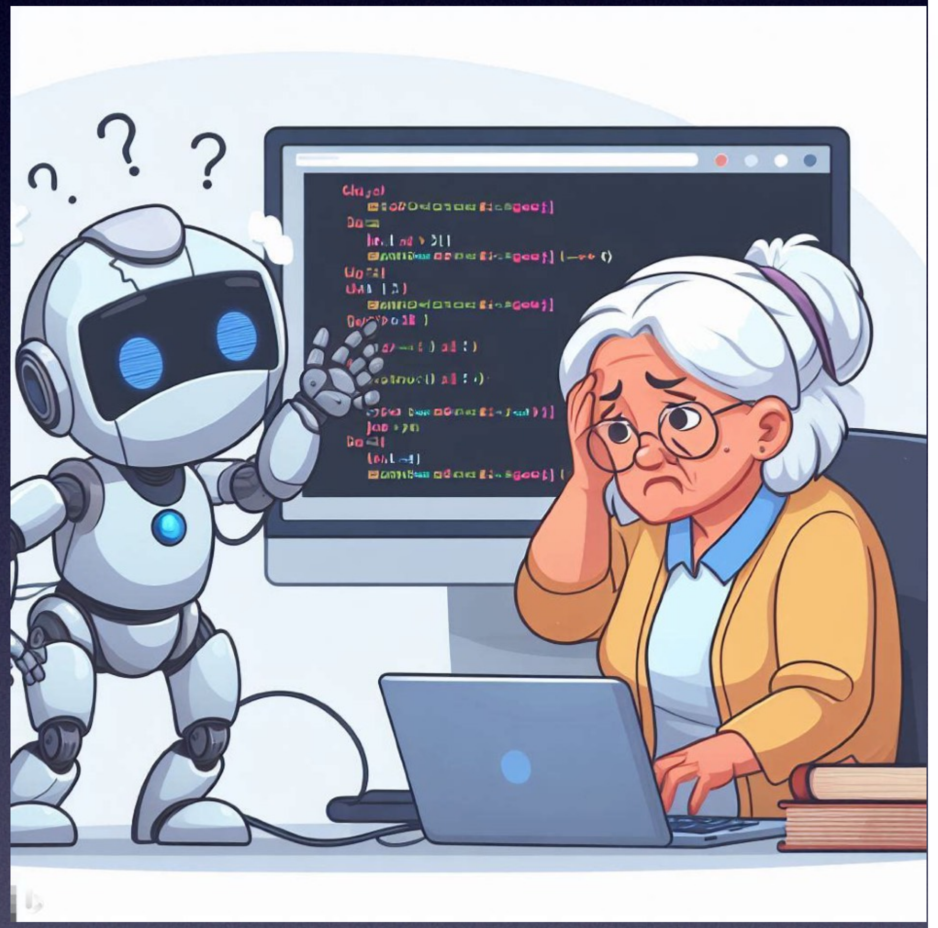
Overall Theme

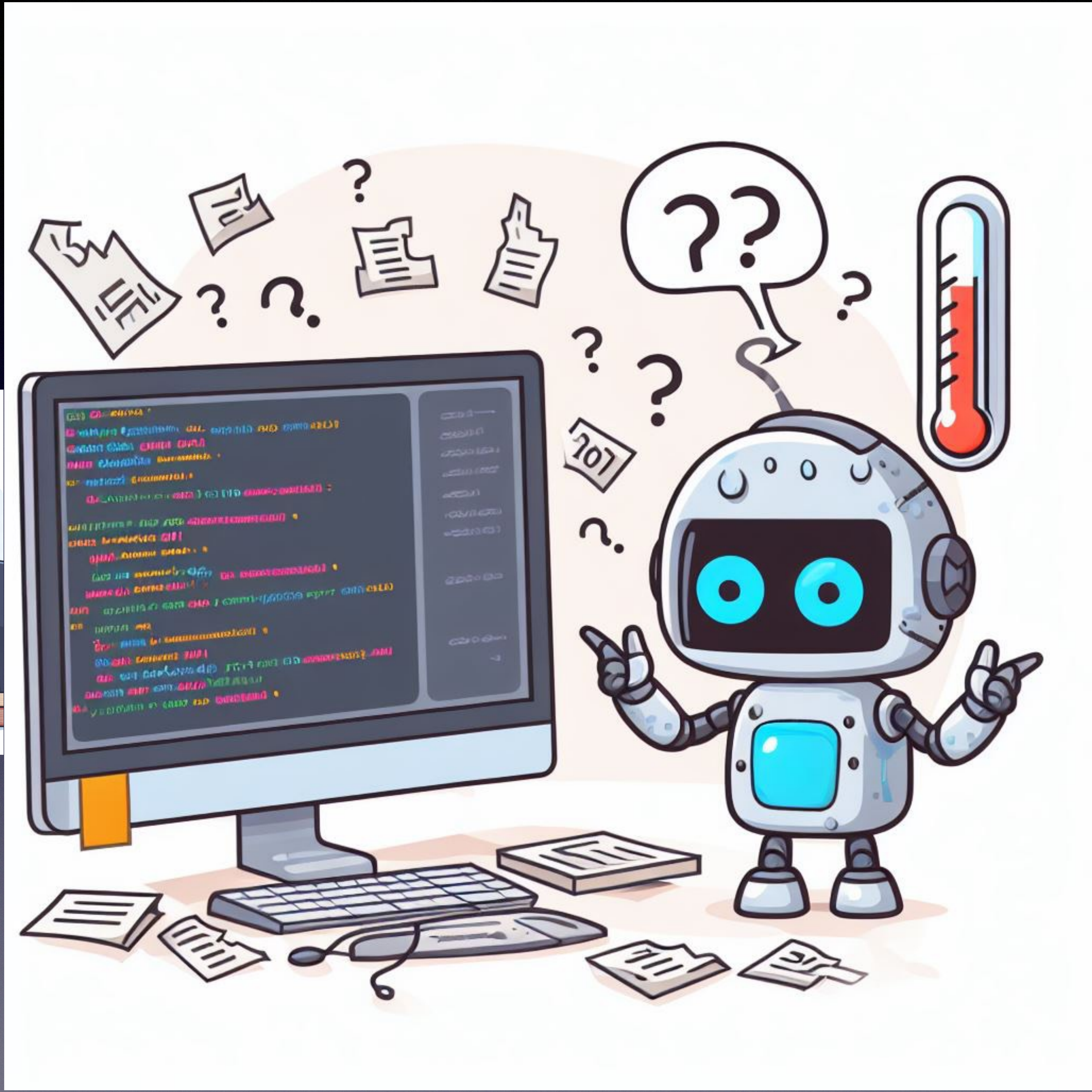
- LLMs: Codex, GPT-x, etc are now widely used to generate code and related artifacts.

- Is this code any good? 🤨

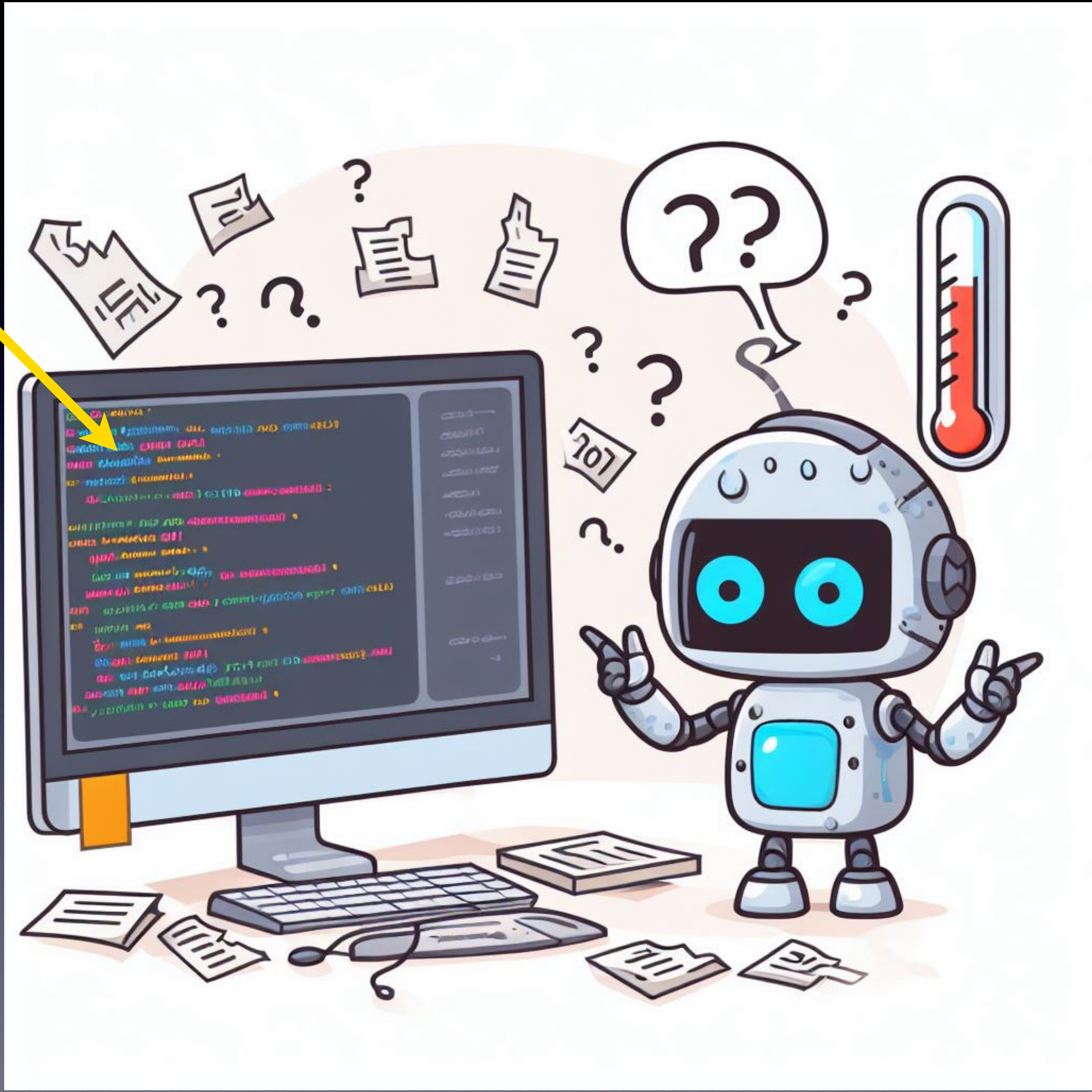
Overall Theme

- LLMs: Codex, GPT-x, etc are now widely used to generate code and related artifacts.
- Is this code any good? 🤨
- If it's not always good what happens? 😱

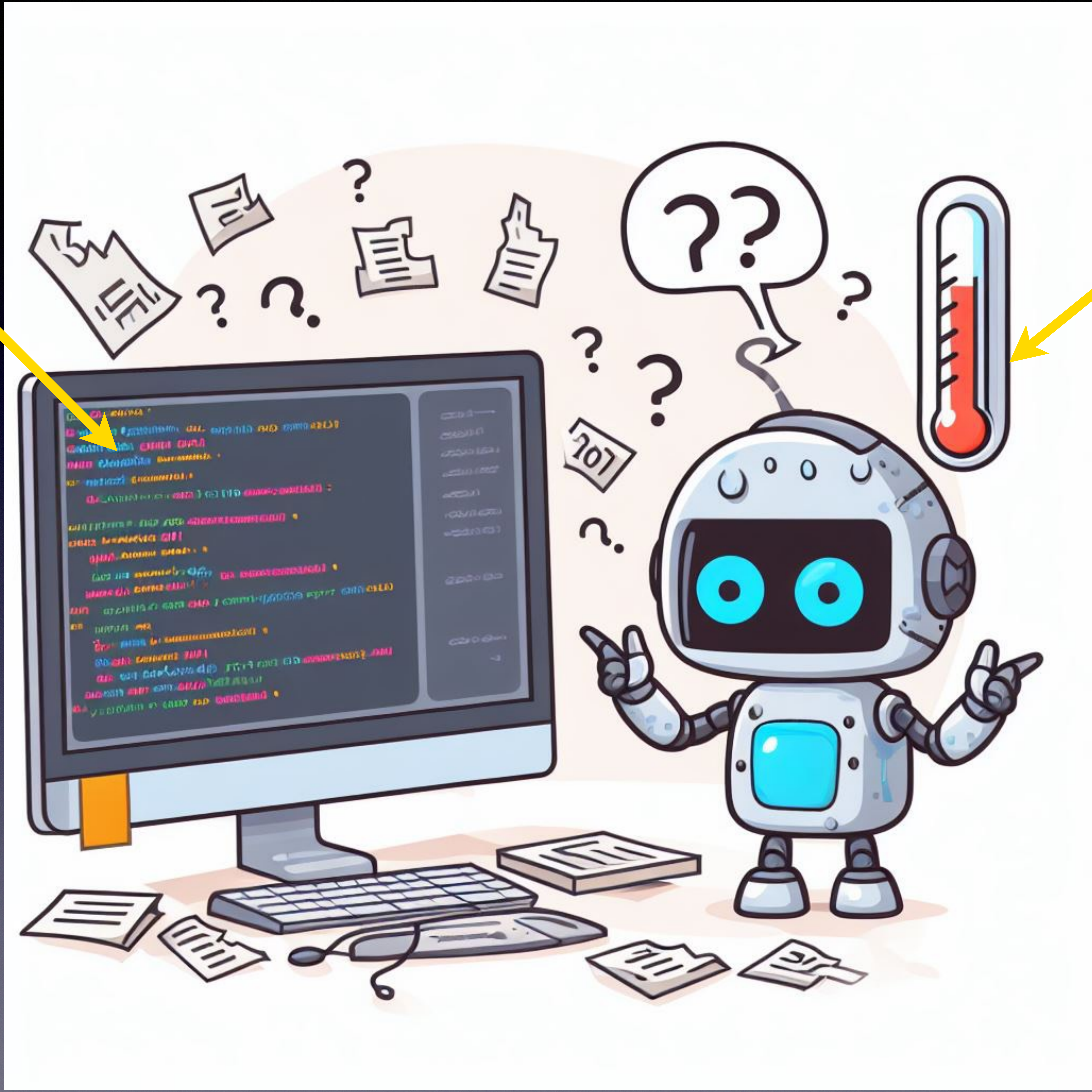




Potentially *Incorrect* Text

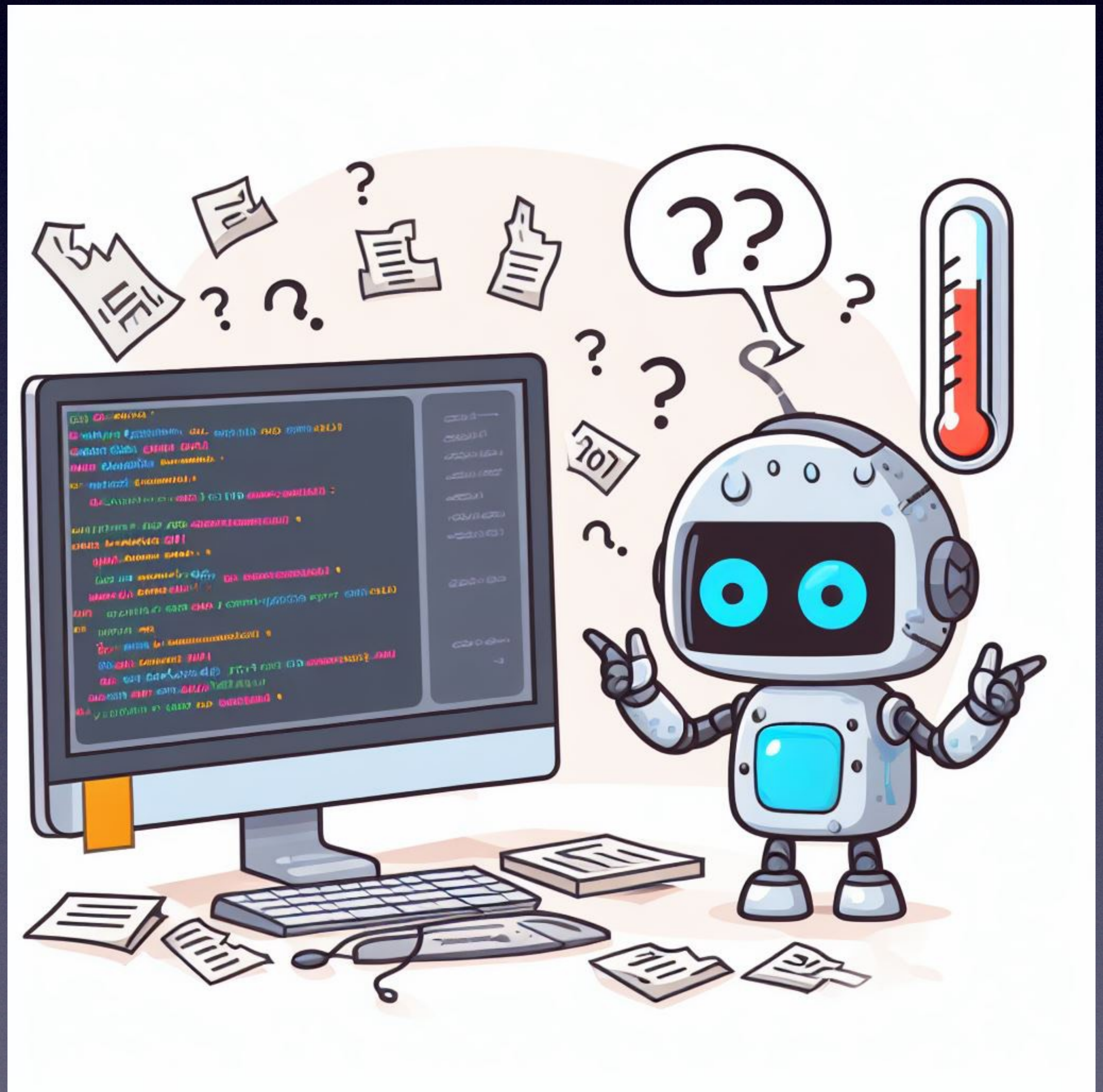


Potentially *Incorrect* Text

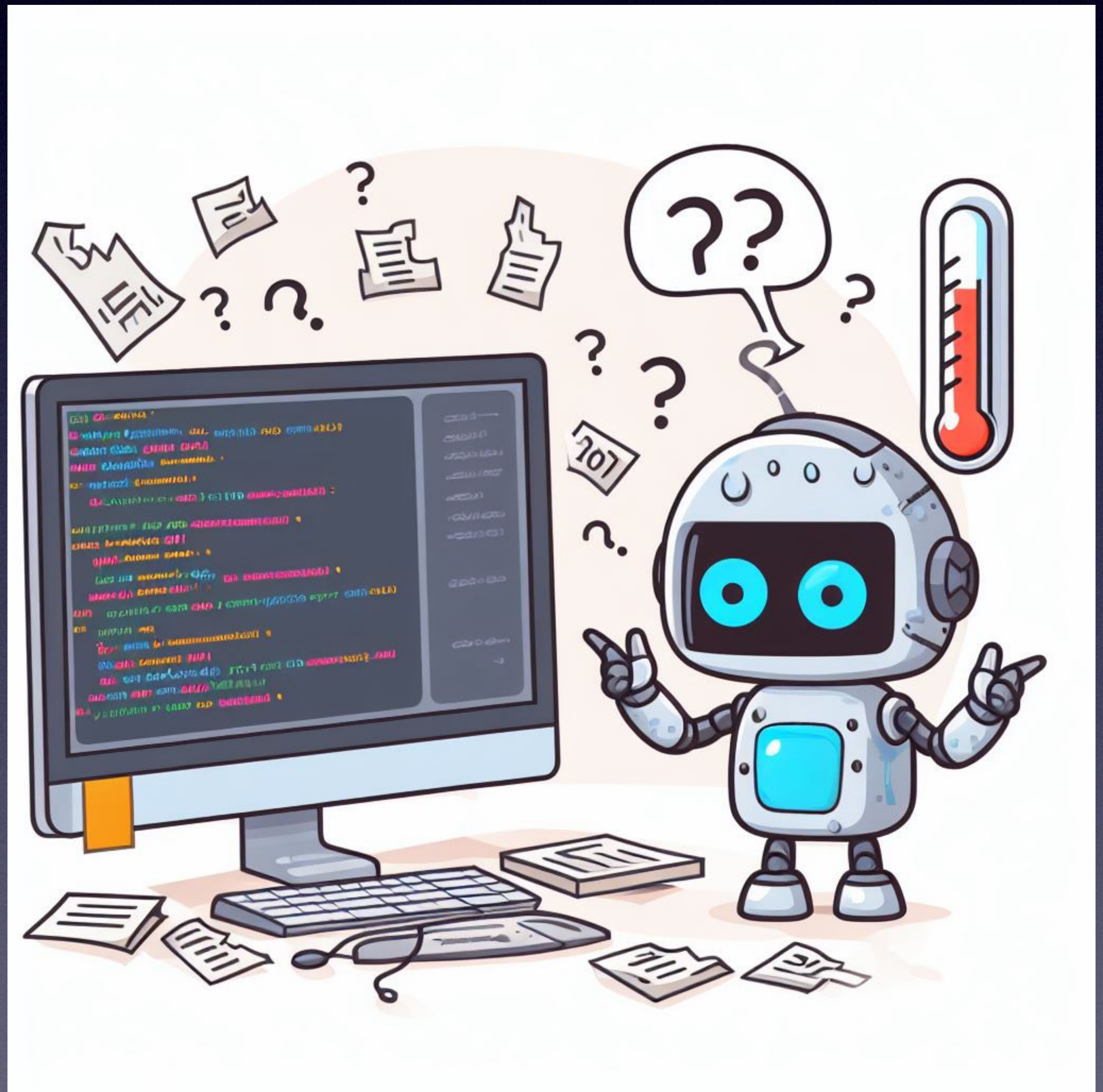


Indication Of *Confidence* In *Correctness*

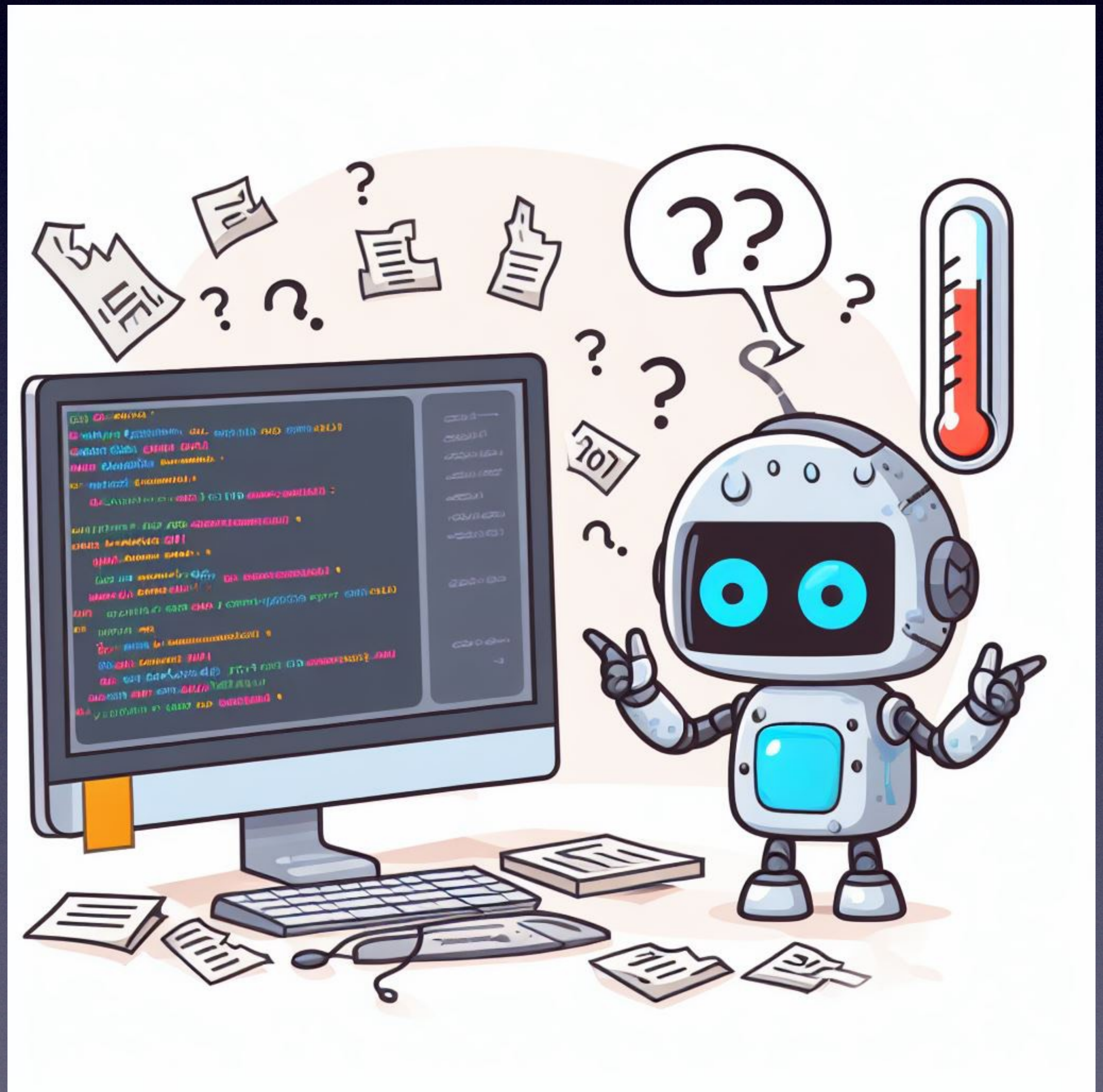
Rational Decision Making



Rational Decision Making

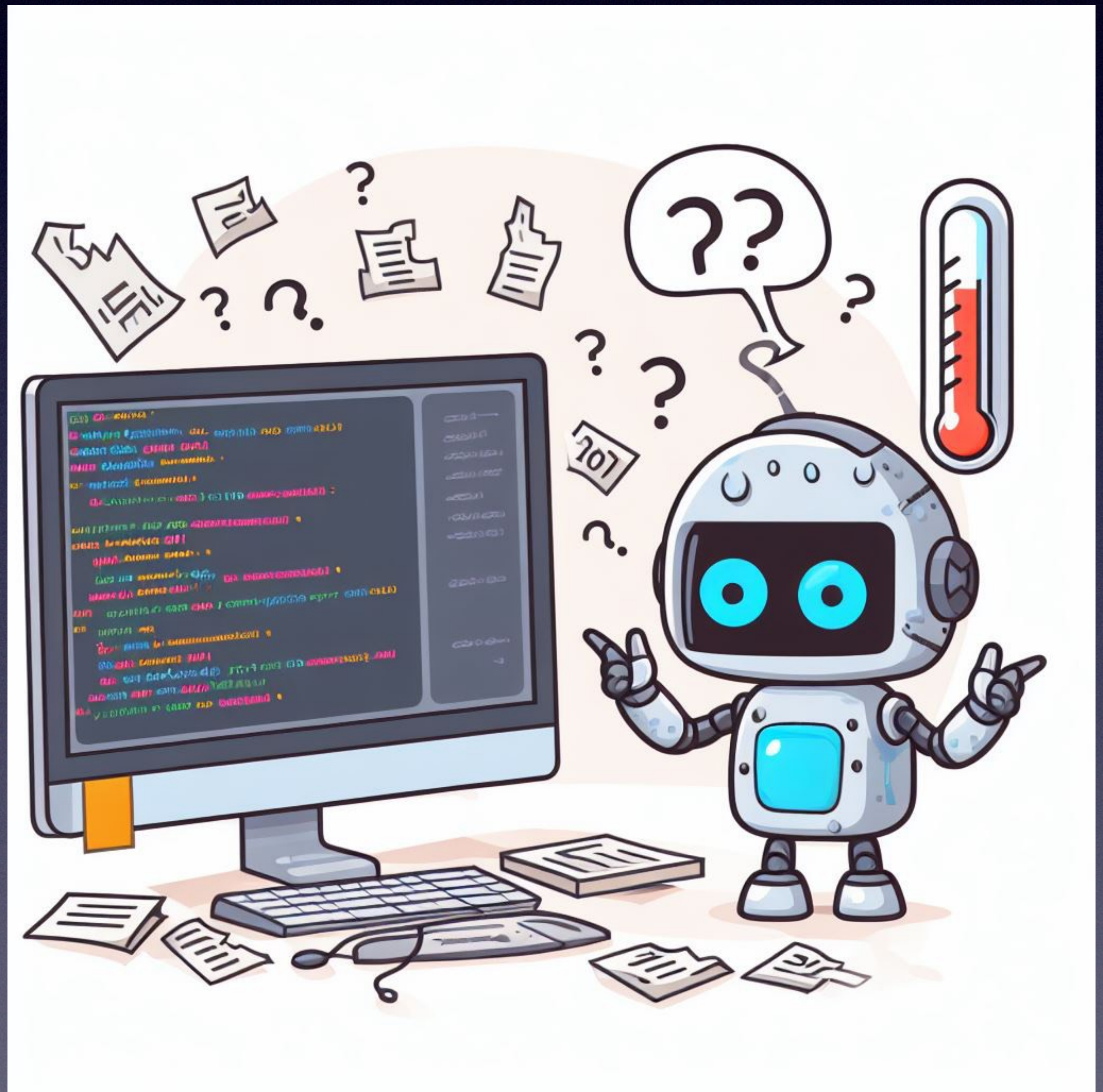


Rational Decision Making



I'll ***discard***
if Confidence
is low

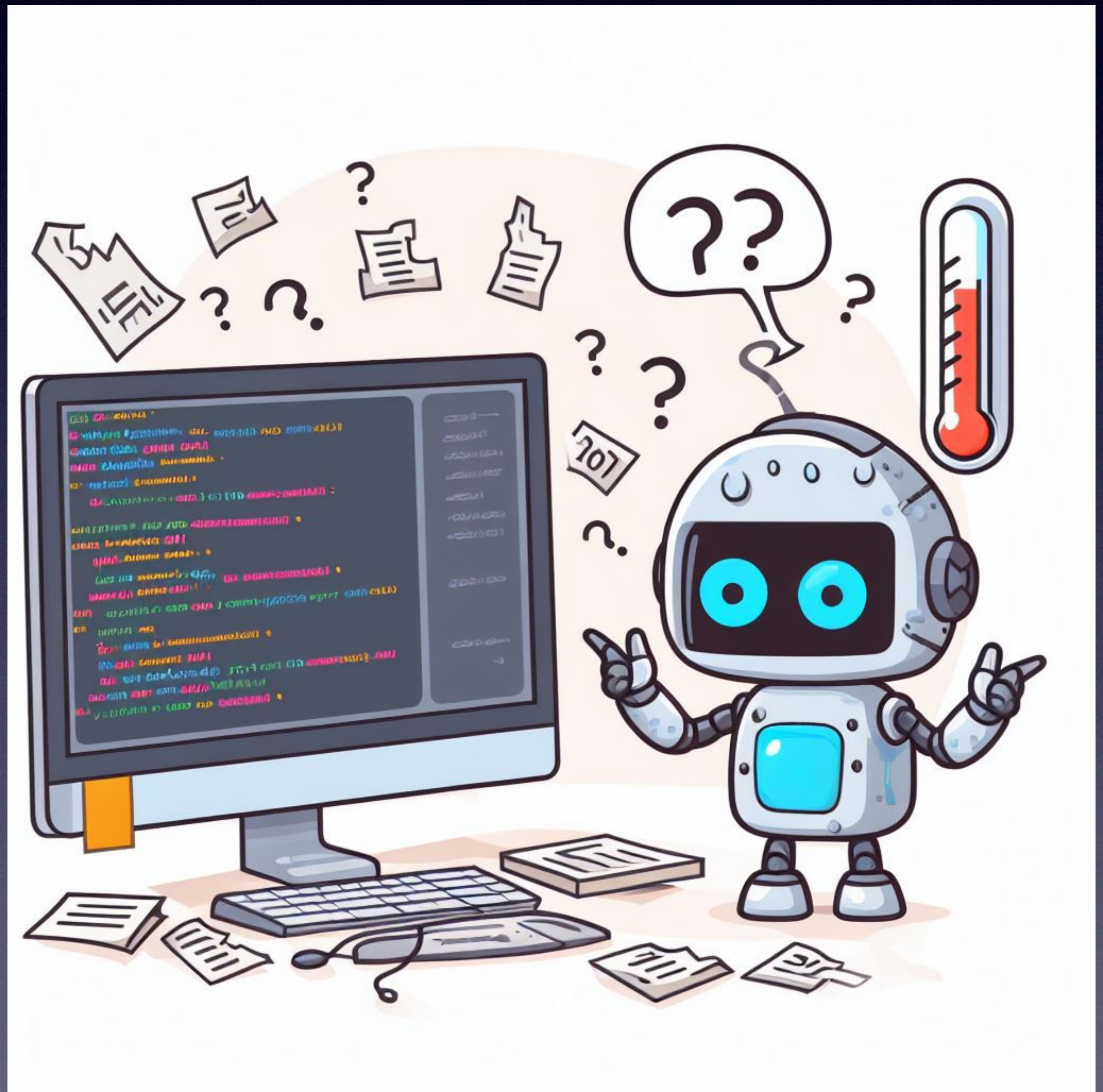
Rational Decision Making



I'll ***use Directly***
if Confidence
Is high

I'll ***discard***
if Confidence
is low

Rational Decision Making

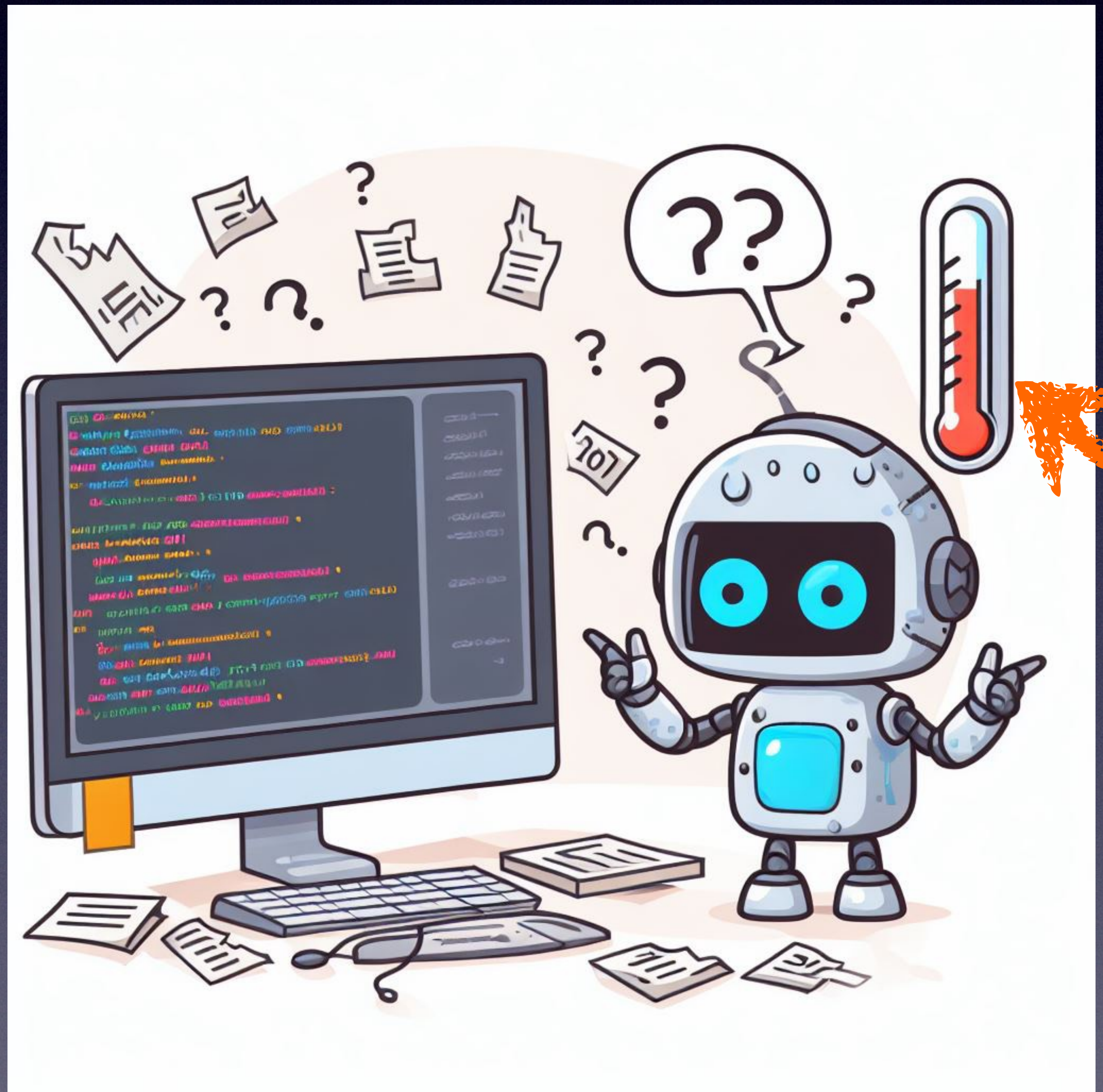


Rational Decision Making

I'll ***review &edit***
if Confidence
is medium

I'll ***use***
Directly
if Confidence
Is high

I'll ***discard***
if Confidence
is low



..but for this to work, we need well-calibrated Confidence!!

Rational Decions, *in Expectation*

Rational Decions, *in Expectation*

- Whenever output is generated at high-confidence, it should be empirically ***correct*** most often. *Otherwise…*

Rational Decions, *in Expectation*

- Whenever output is generated at high-confidence, it should be empirically ***correct*** most often. *Otherwise…*
- Whenever output is generated at low-confidence, it should be empircally ***wrong*** most often. *Otherwise…*

Rational Decions, *in Expectation*

- Whenever output is generated at high-confidence, it should be empirically ***correct*** most often. *Otherwise…*
- Whenever output is generated at low-confidence, it should be empircally ***wrong*** most often. *Otherwise…*
- Whenever output is generated at medium-confidence, it should be empirically ***right*** *and* ***wrong*** about the same. *Otherwise…*

Calibration of Predictive Models

Rain Prediction Model

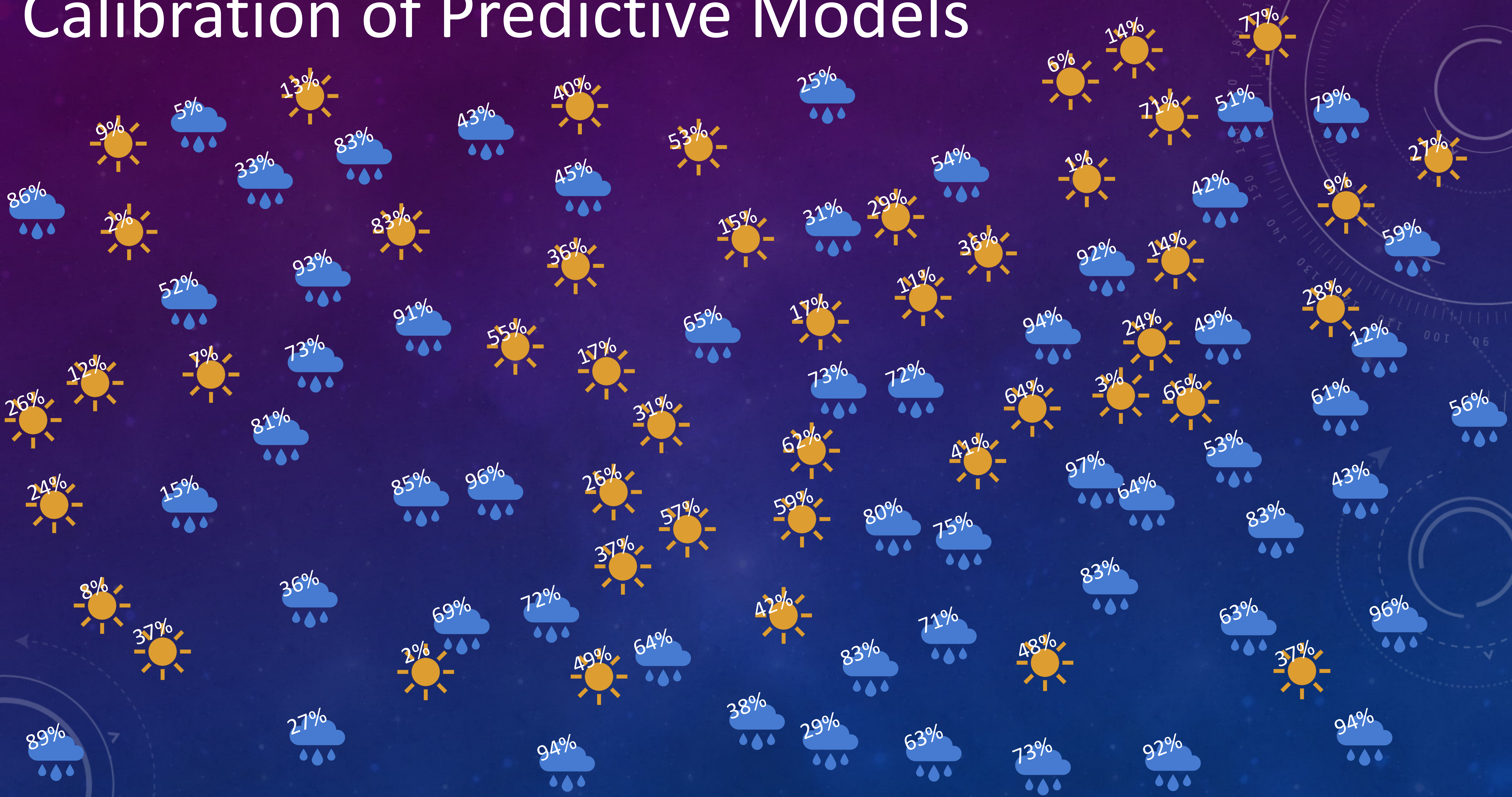
54%

Calibration of Predictive Models

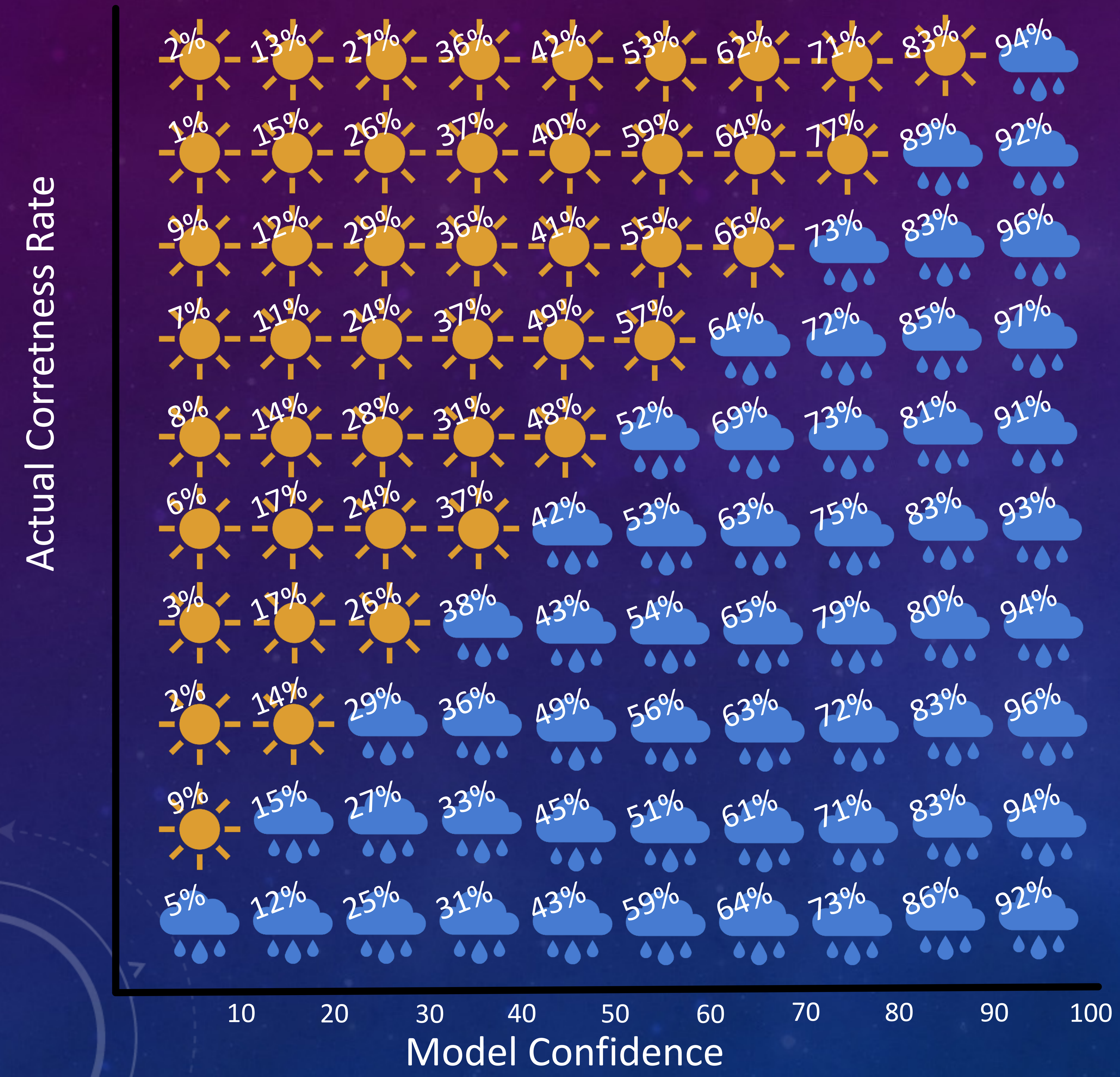
36%

54%

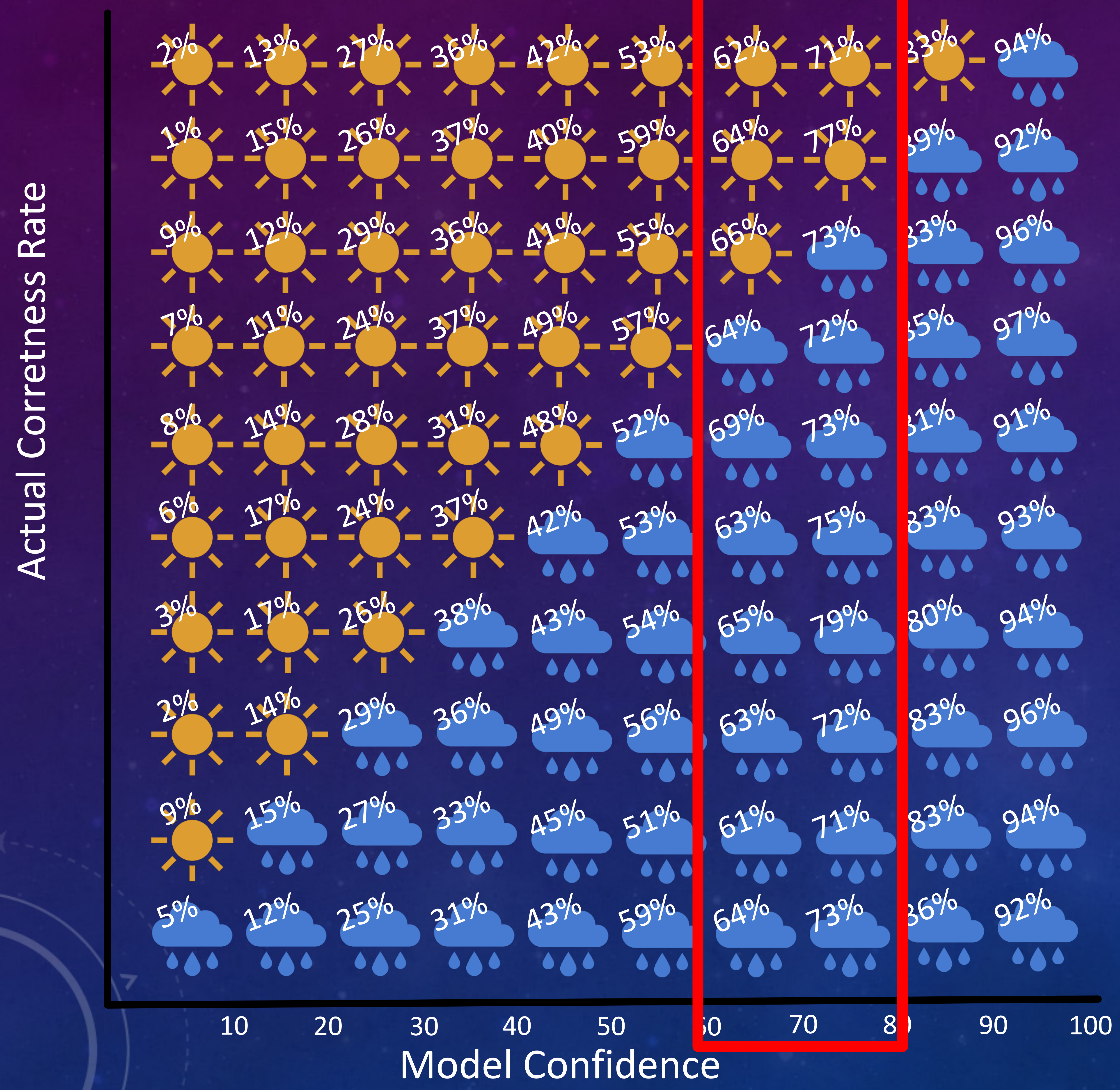
Calibration of Predictive Models



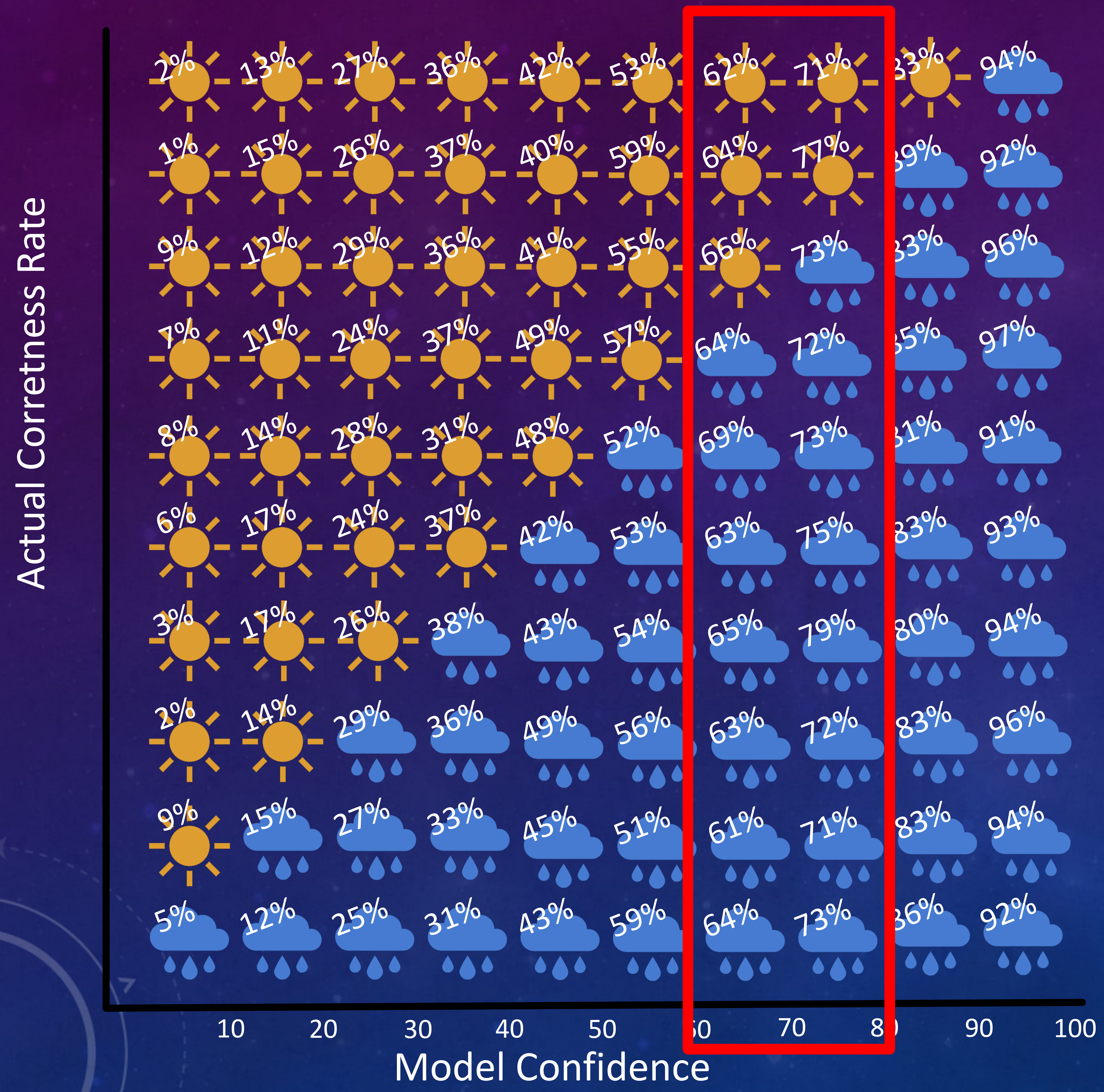
Calibration of Predictive Models



Calibration of Predictive Models



Calibration of Predictive Models



=> Rational Decision Making!

Correctness? Confidence?

Correctness?

Confidence?

- Exact match with Provided Correct Code

Correctness?

Confidence?

- Exact match with Provided Correct Code
- Correctness modulo testing. (Which tests?)

Correctness?

- Exact match with Provided Correct Code
- Correctness modulo testing. (Which tests?)

Confidence?

- ***“Intrinsic probabilities”*** from the model *(average and cumulative)*

Correctness?	Confidence?

Correctness?

- Exact match with Provided Correct Code
- Correctness modulo testing. (Which tests?)

Confidence?

- ***“Intrinsic probabilities”*** from the model *(average and cumulative)*
- … Other measures later

“Intrinsic Probabilities”

“Intrinsic Probabilities”

- LLMs generate text one token at a time, each according to a probability space for choices.

“Intrinsic Probabilities”

- LLMs generate text one token at a time, each according to a probability space for choices.

```
print ( “ hello
```

“Intrinsic Probabilities”

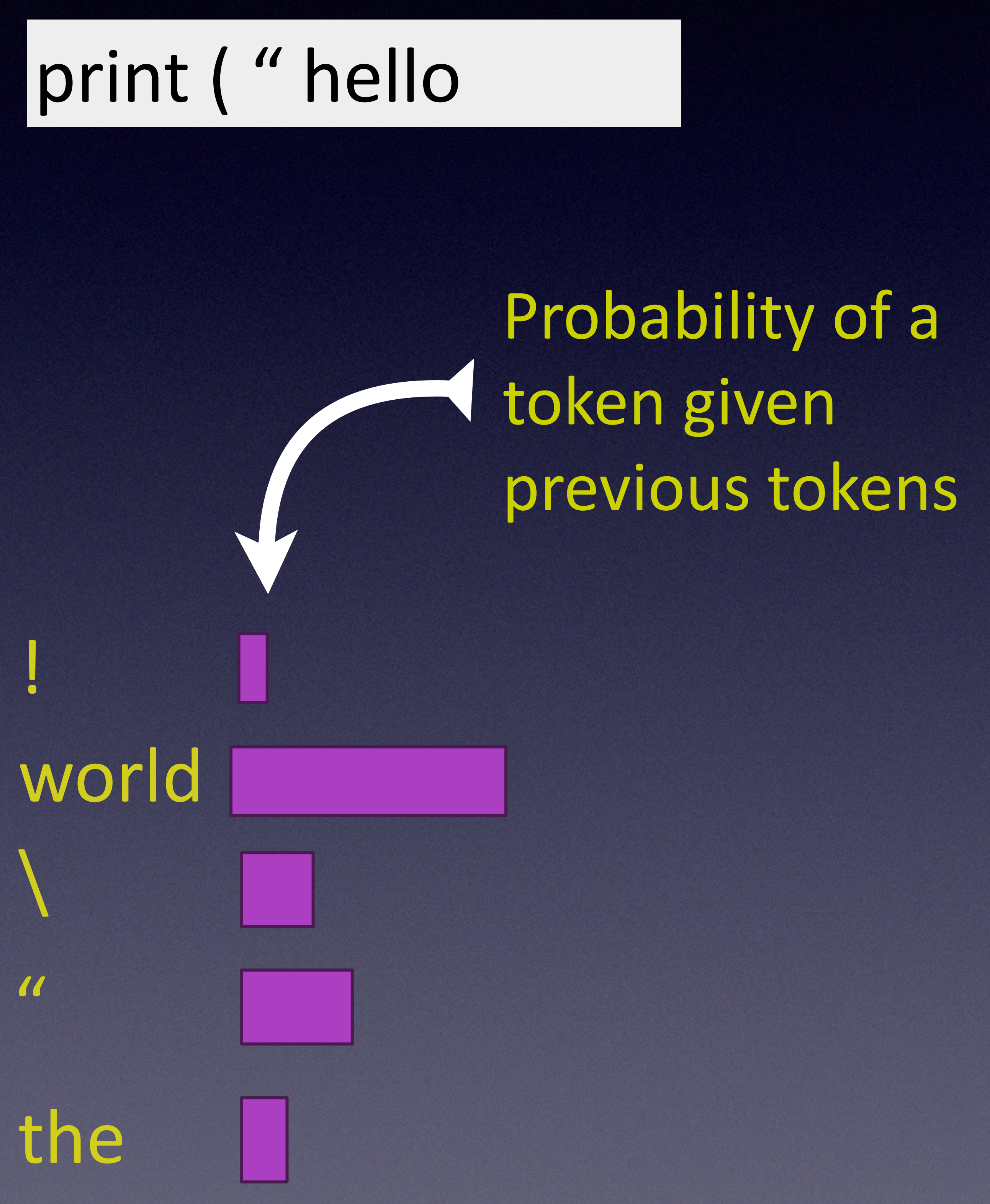
- LLMs generate text one token at a time, each according to a probability space for choices.

print (“ hello

!
world
\
“
the

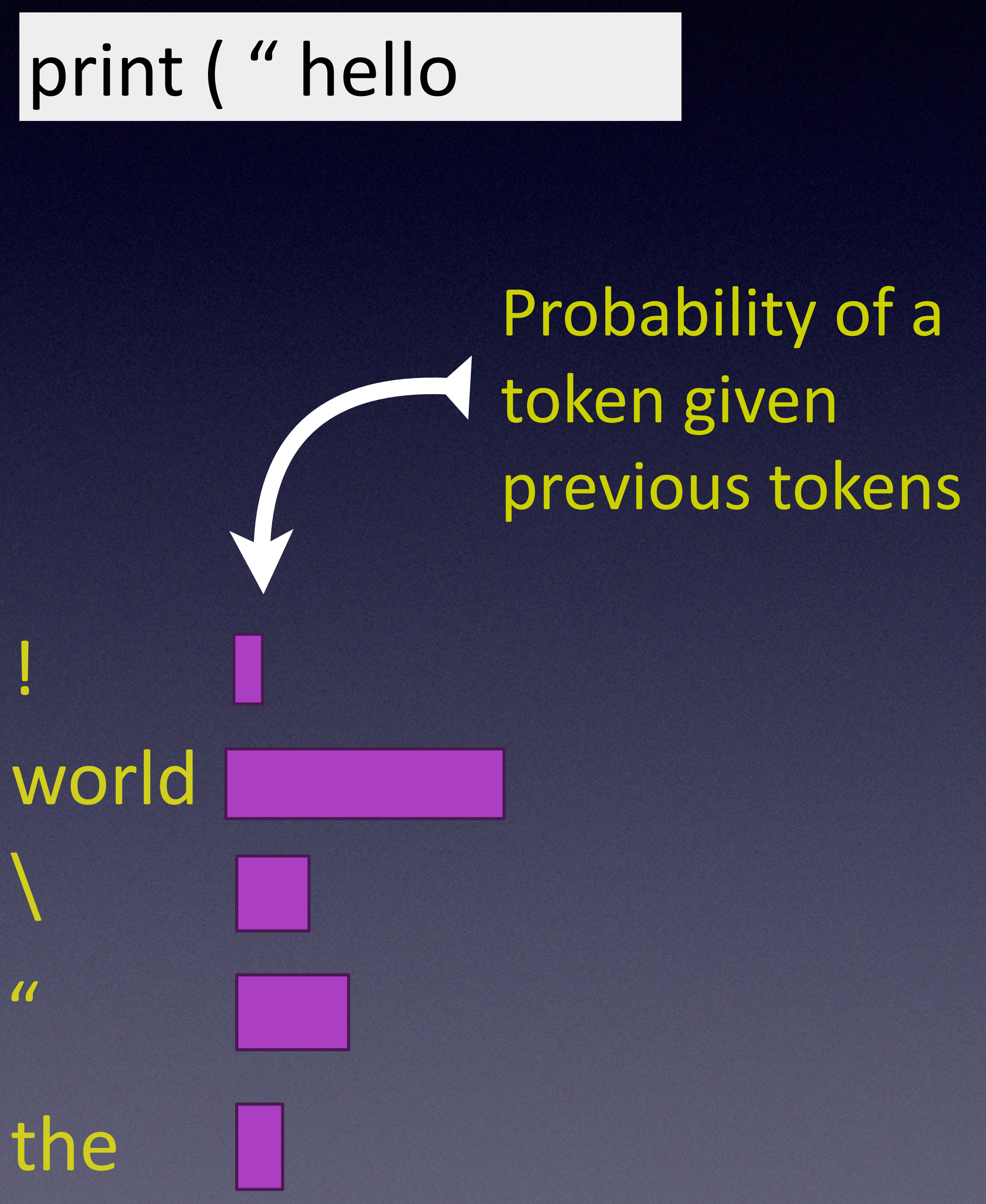
“Intrinsic Probabilities”

- LLMs generate text one token at a time, each according to a probability space for choices.



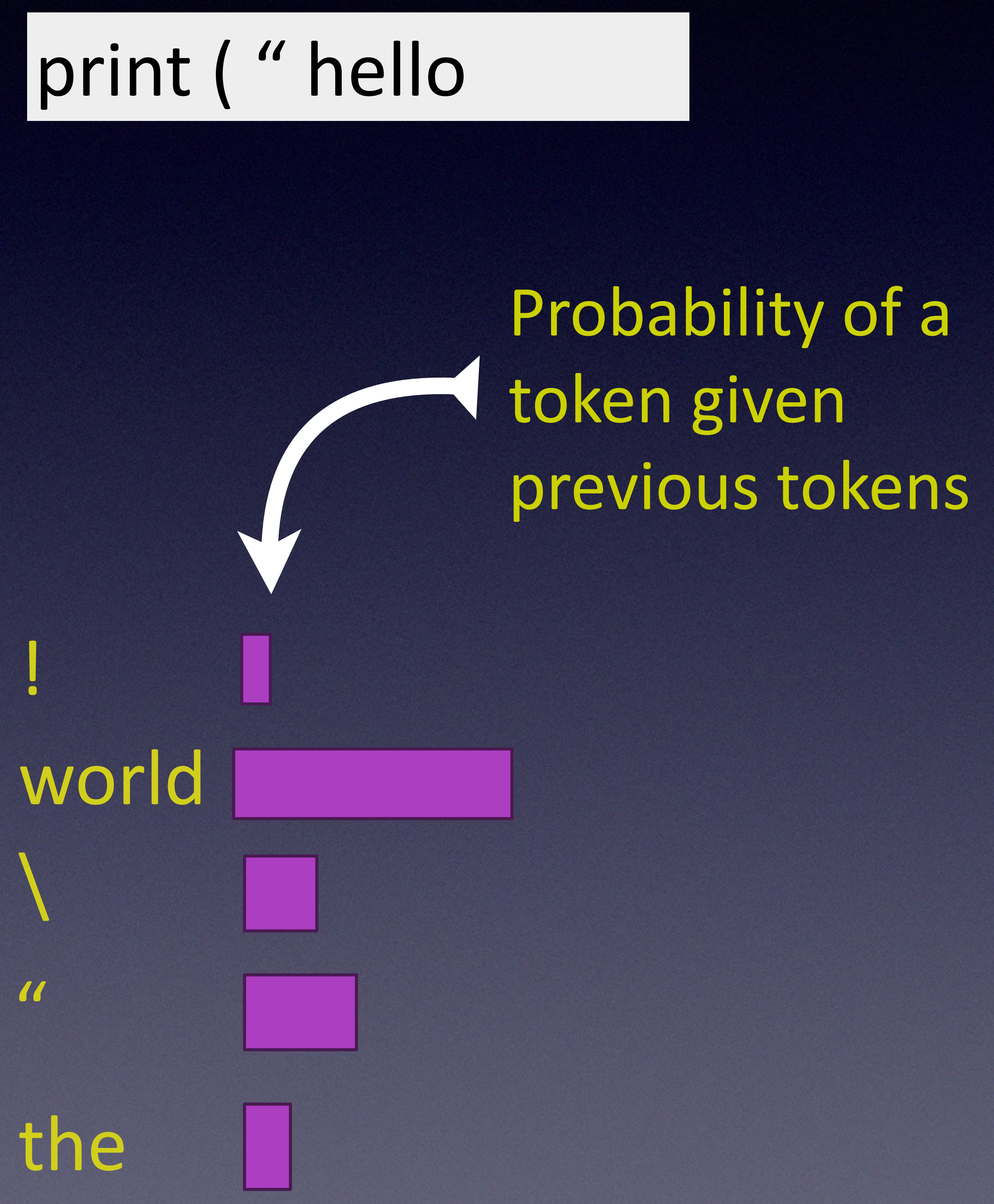
“Intrinsic Probabilities”

- LLMs generate text one token at a time, each according to a probability space for choices.
- “Intrinsic” confidence measure for us, is just this probability.



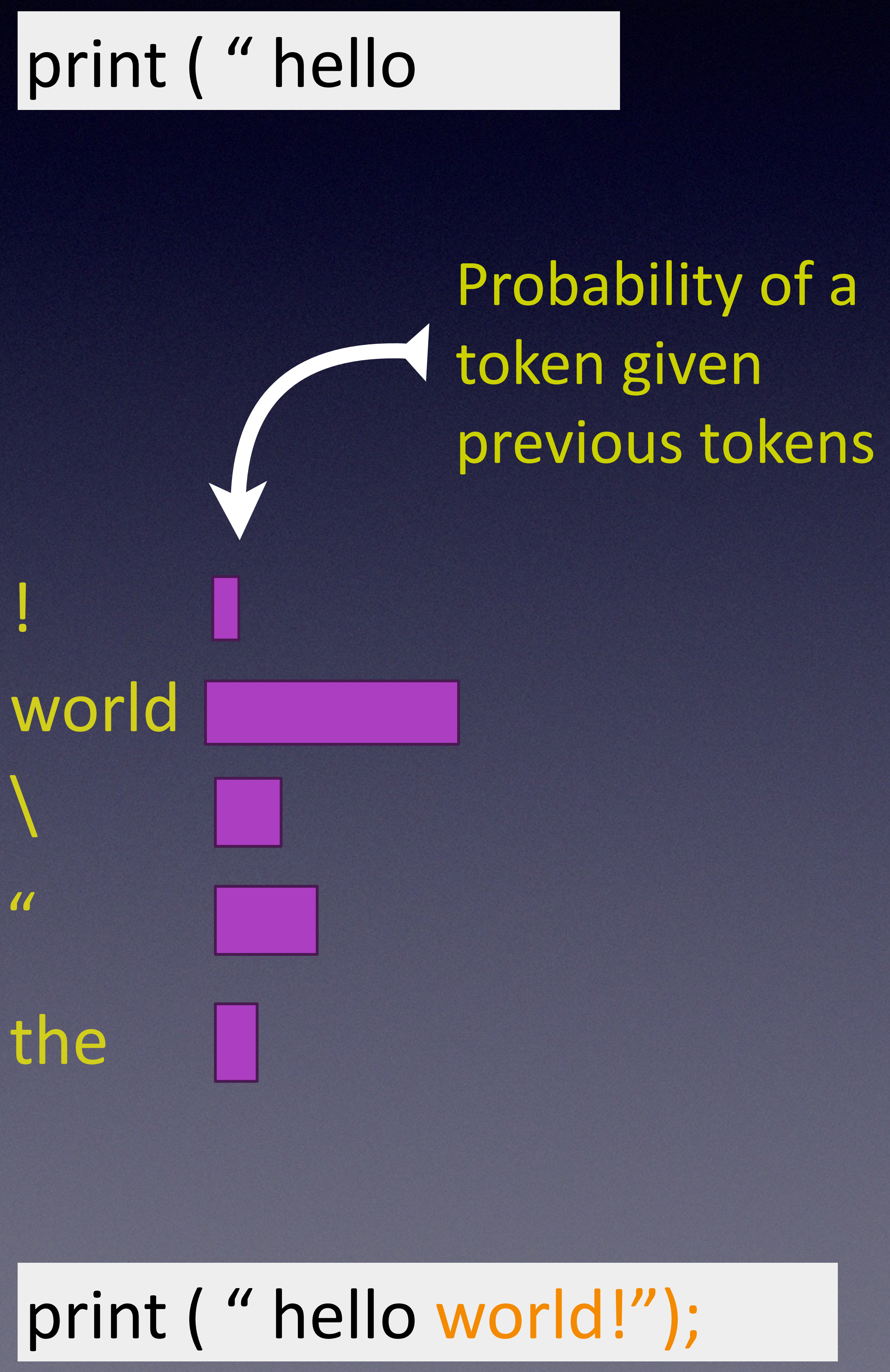
“Intrinsic Probabilities”

- LLMs generate text one token at a time, each according to a probability space for choices.
- “Intrinsic” confidence measure for us, is just this probability.
- A sequence is then produced.



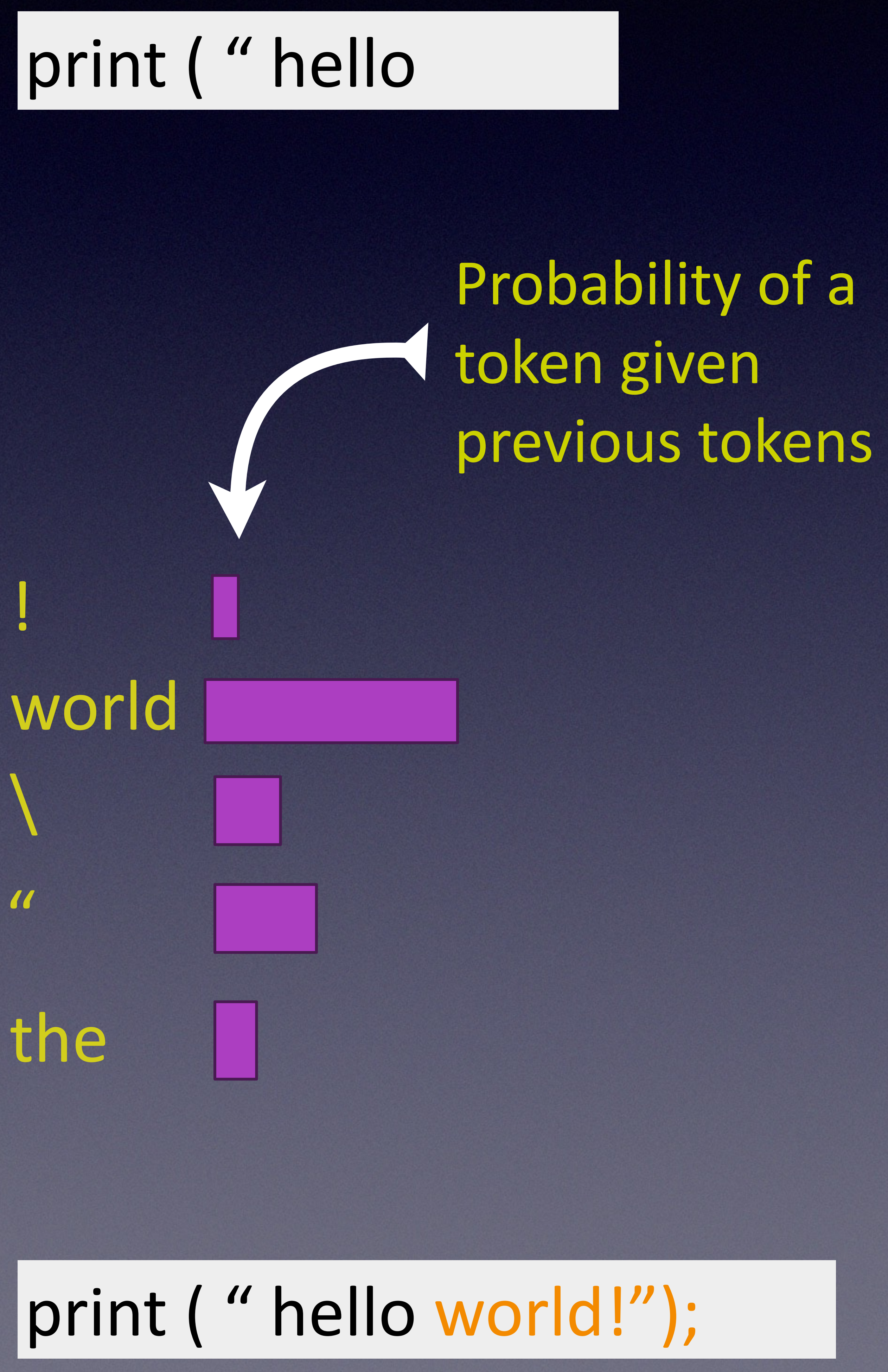
“Intrinsic Probabilities”

- LLMs generate text one token at a time, each according to a probability space for choices.
- “Intrinsic” confidence measure for us, is just this probability.
- A sequence is then produced.



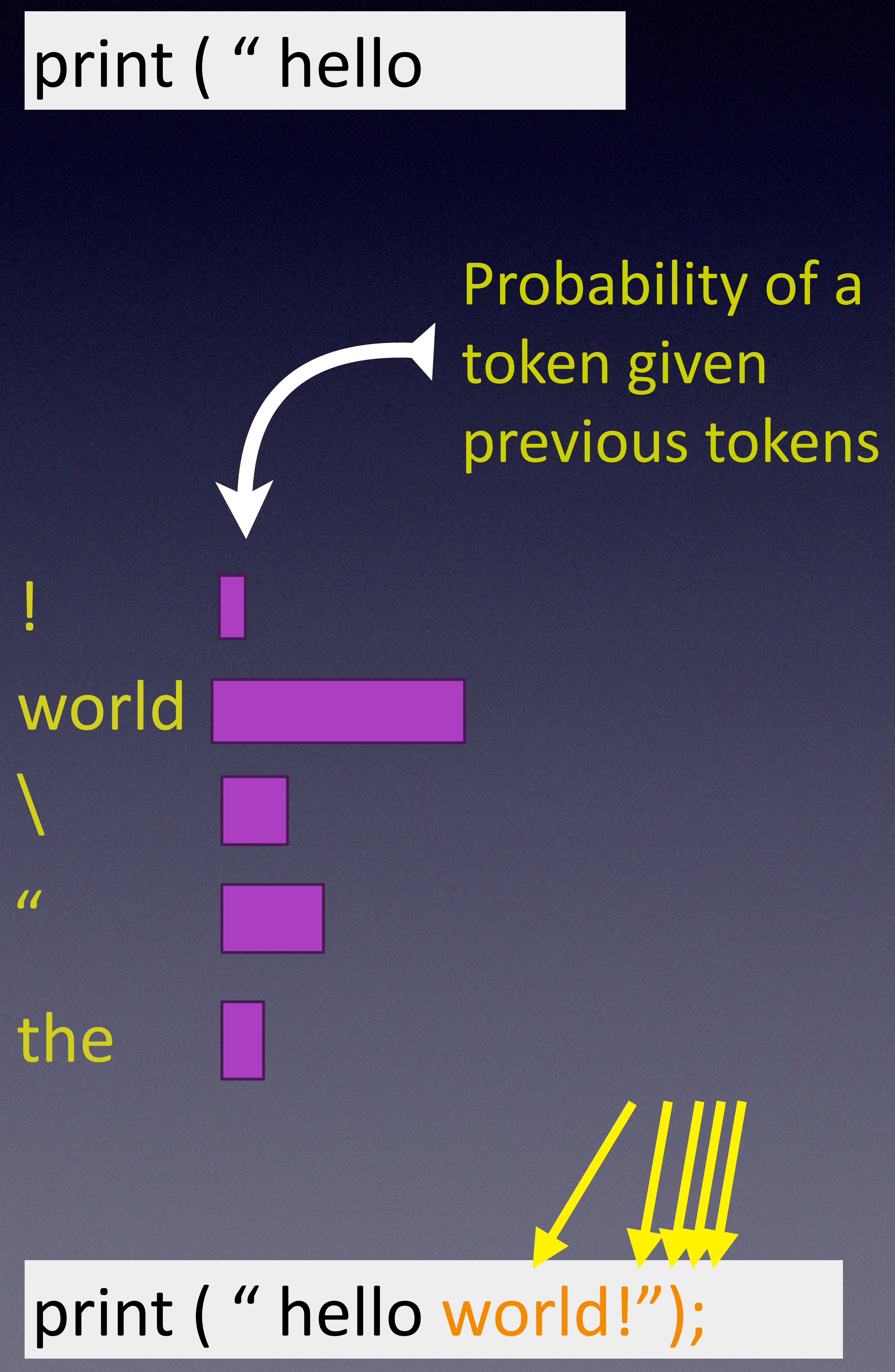
“Intrinsic Probabilities”

- LLMs generate text one token at a time, each according to a probability space for choices.
- “Intrinsic” confidence measure for us, is just this probability.
- A sequence is then produced.
- Take per-token log-probs, and summarize (*Avg & Product*)



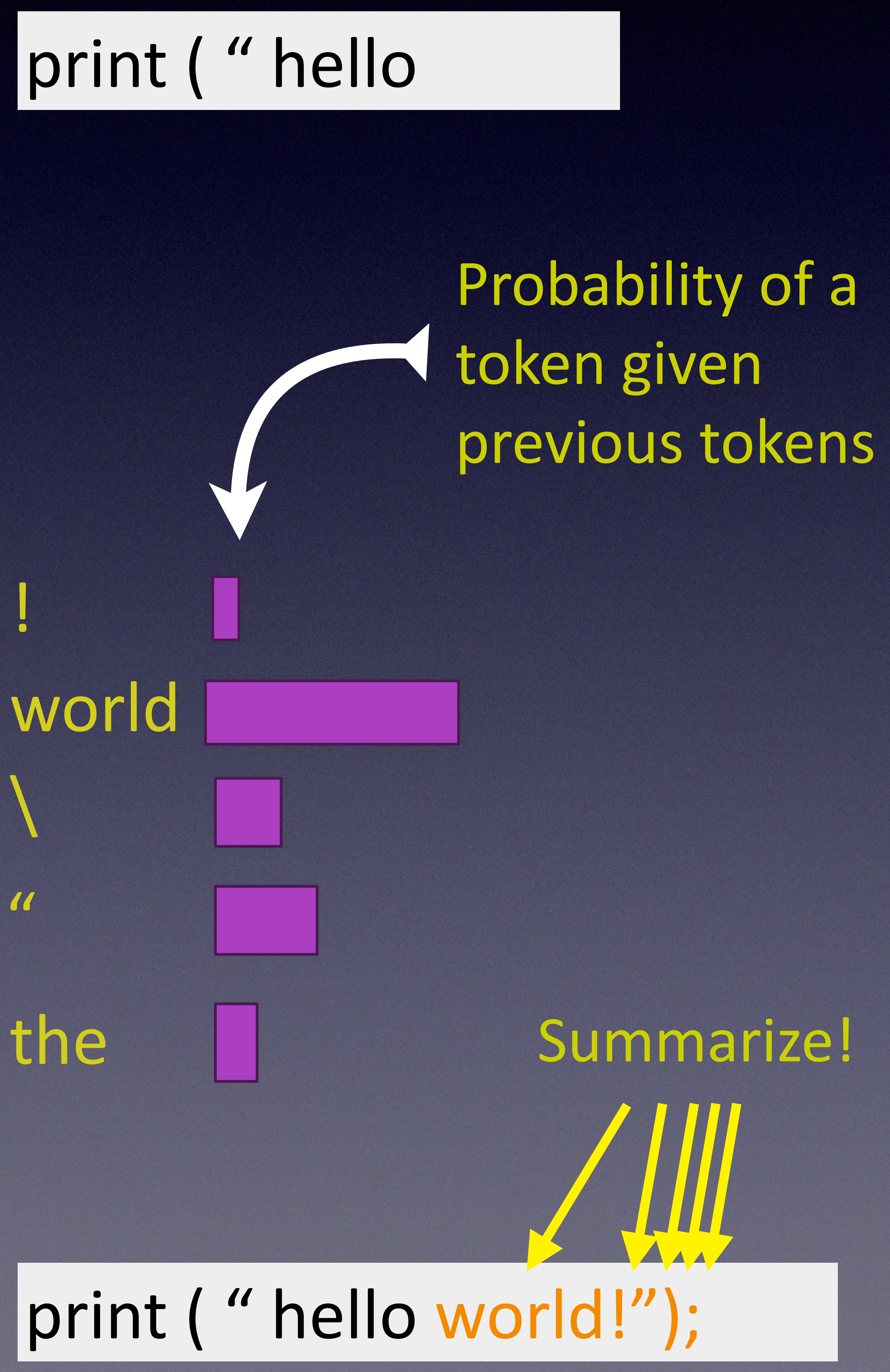
“Intrinsic Probabilities”

- LLMs generate text one token at a time, each according to a probability space for choices.
- “Intrinsic” confidence measure for us, is just this probability.
- A sequence is then produced.
- Take per-token log-probs, and summarize (*Avg & Product*)

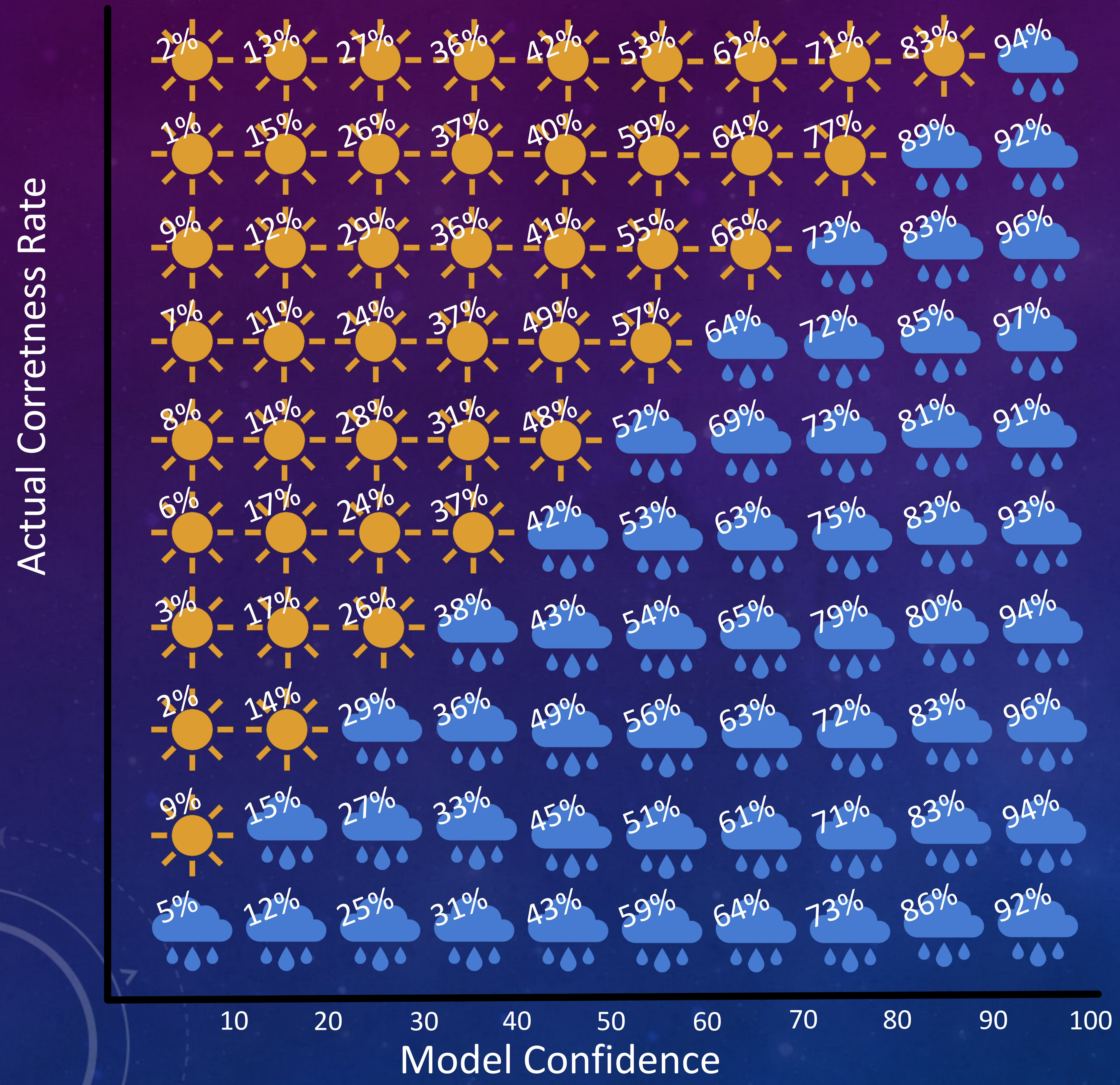


“Intrinsic Probabilities”

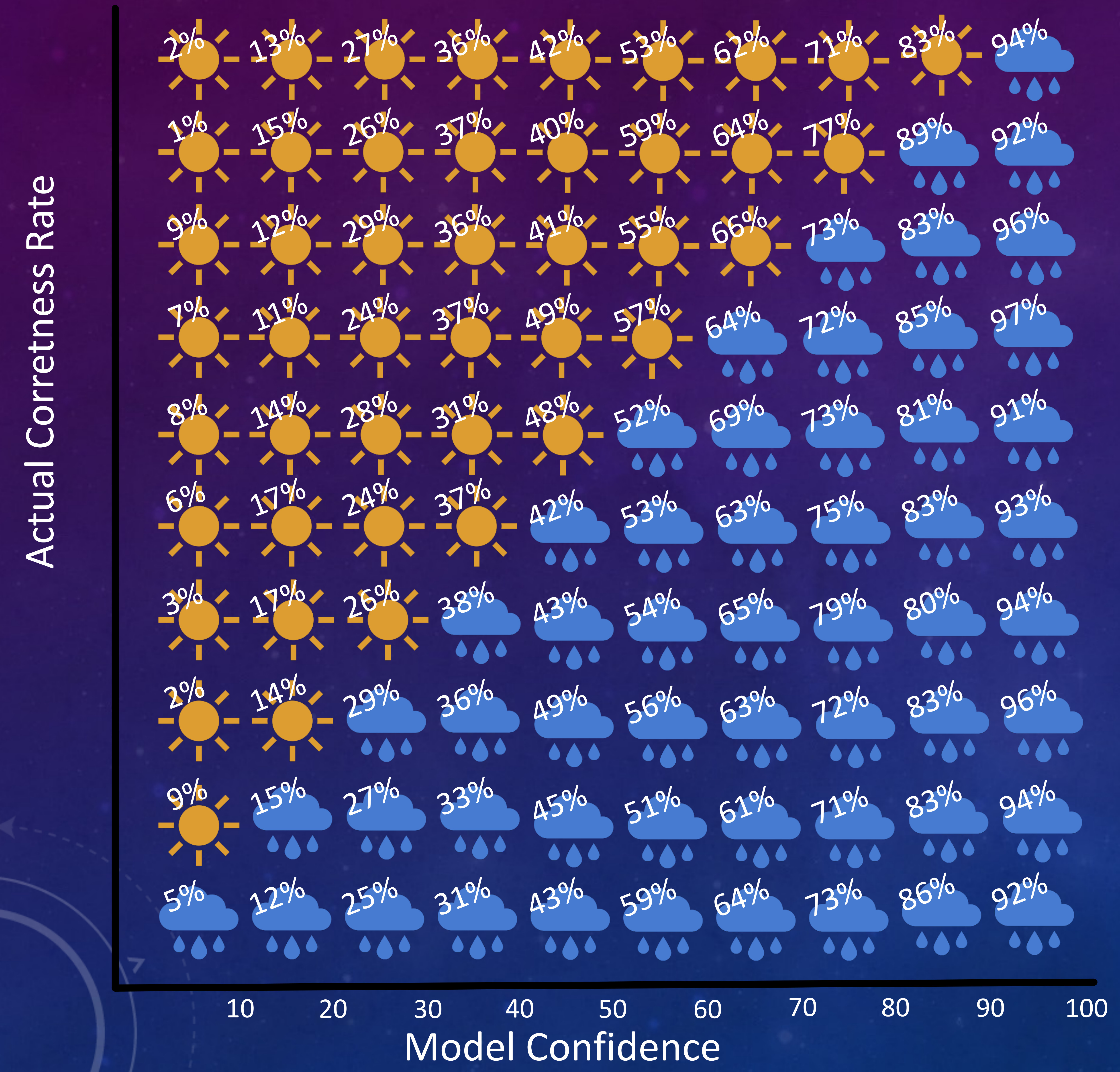
- LLMs generate text one token at a time, each according to a probability space for choices.
- “Intrinsic” confidence measure for us, is just this probability.
- A sequence is then produced.
- Take per-token log-probs, and summarize (*Avg & Product*)



Calibration of Predictive Models

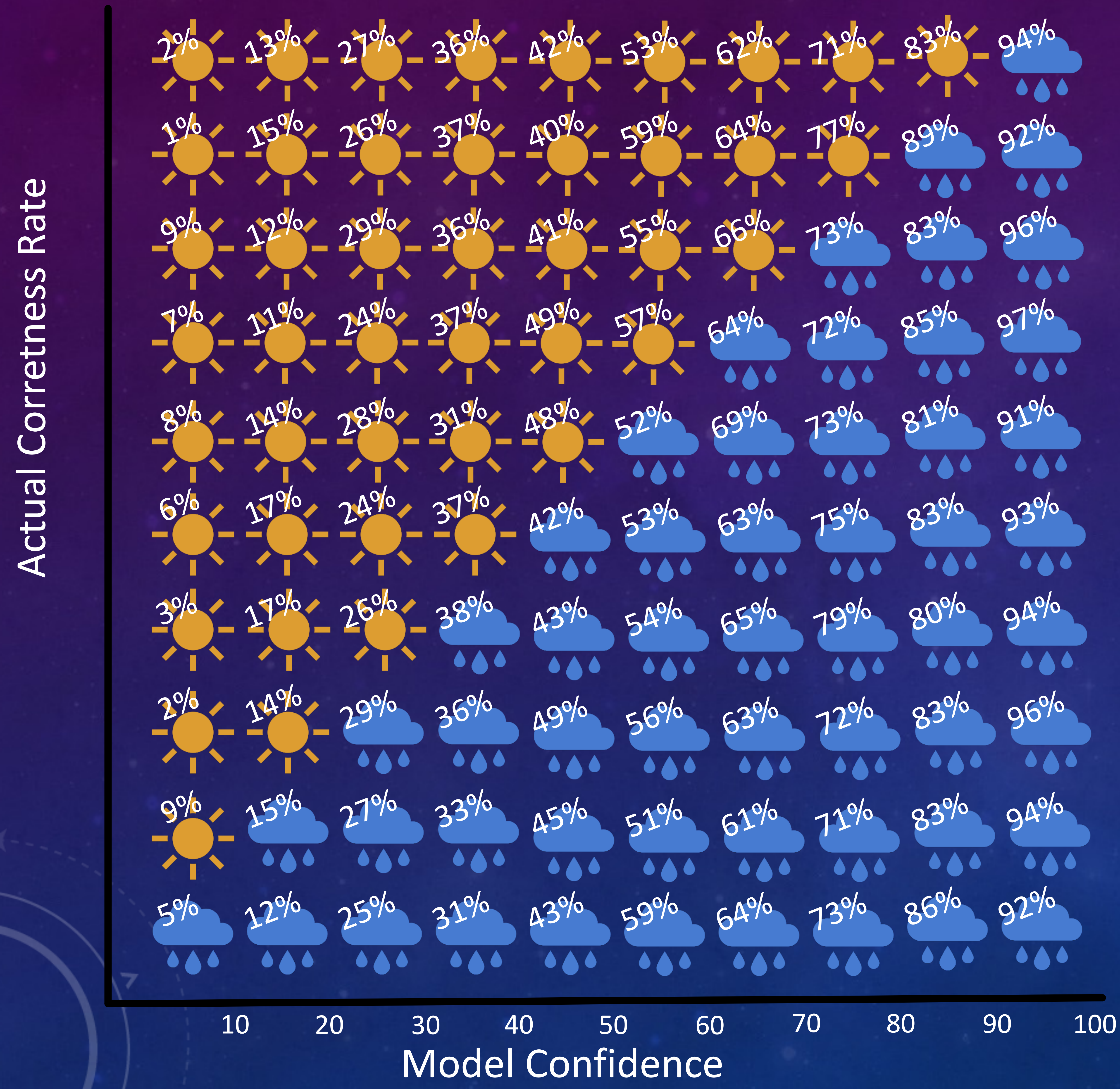


Calibration of Predictive Models



- Formalize this?

Calibration of Predictive Models



- Formalize this?
- Measuring how close we are to this ideal?

Calibration is a "constraint" On a Conditional Probability.

p(|) =

Calibration is a “constraint” On a Conditional Probability.

p(| model predicts with “π” confidence) =

Calibration is a “constraint” On a Conditional Probability.

p(model’s prediction is correct | model predicts with “π” confidence) =

Calibration is a "constraint" On a Conditional Probability.

$p(\text{model's prediction is correct} \mid \text{model predicts with "}\pi\text{" confidence}) = \pi$

Calibration is a "constraint"
On a Conditional Probability.

p(model's prediction is correct | model predicts with "π" confidence) = π

$$p(y = \hat{y} \,|\, P_{\mathscr{M}}(x, \hat{y}) = \pi) = \pi$$

Calibration is a "constraint" On a Conditional Probability.

p(model's prediction is correct | model predicts with "π" confidence) = π

$$p(y = \hat{y} \mid P_{\mathbb{M}}(x, \hat{y}) = \pi) = \pi$$

y correct output for x

$\hat{y}$ model output for x

$P_{\mathbb{M}}(x, \hat{y})$ model confidence for input x output ŷ

Reliability Diagram:

Confidence vs. Correctness

p(|) =

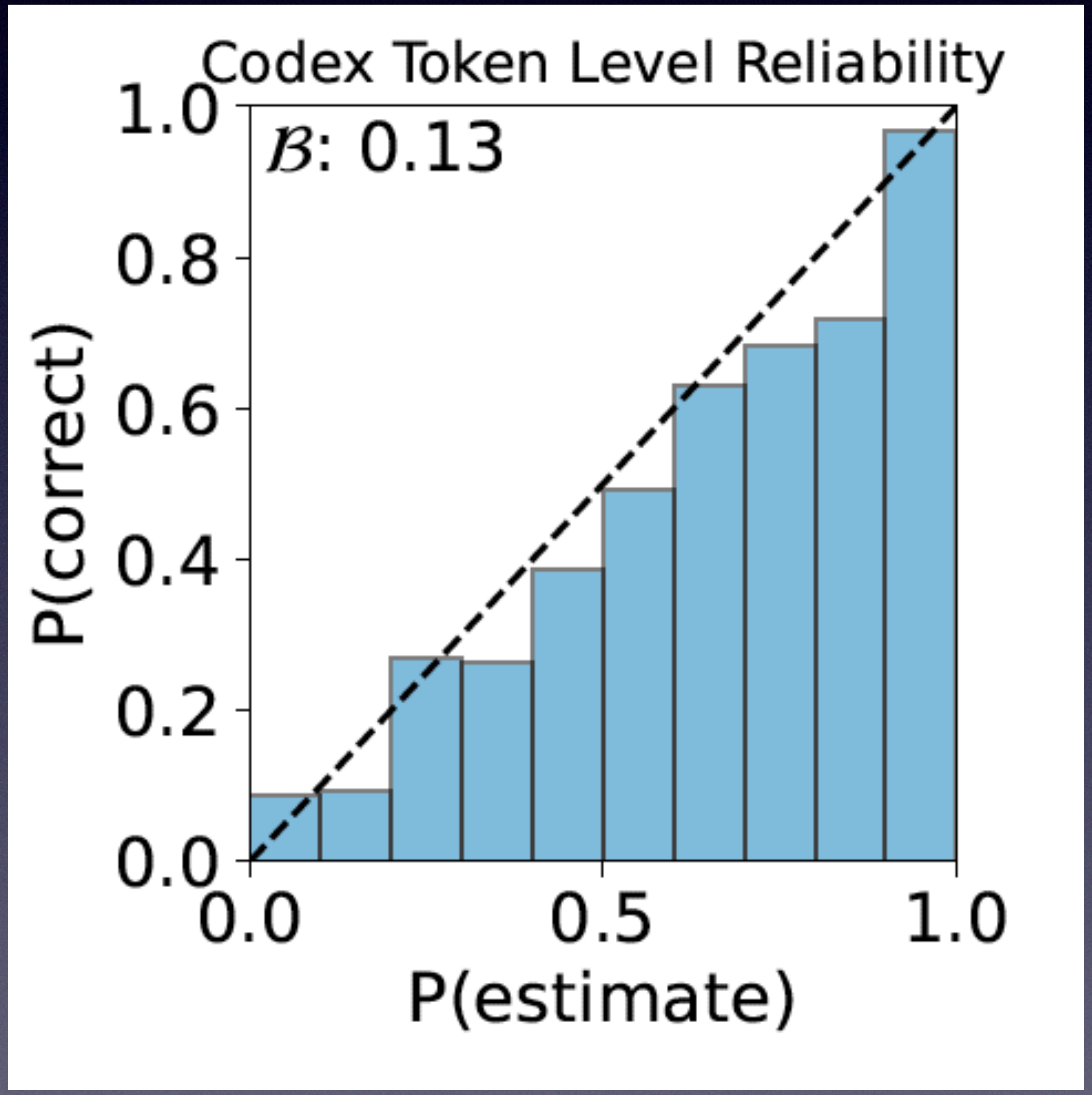
Reliability Diagram:

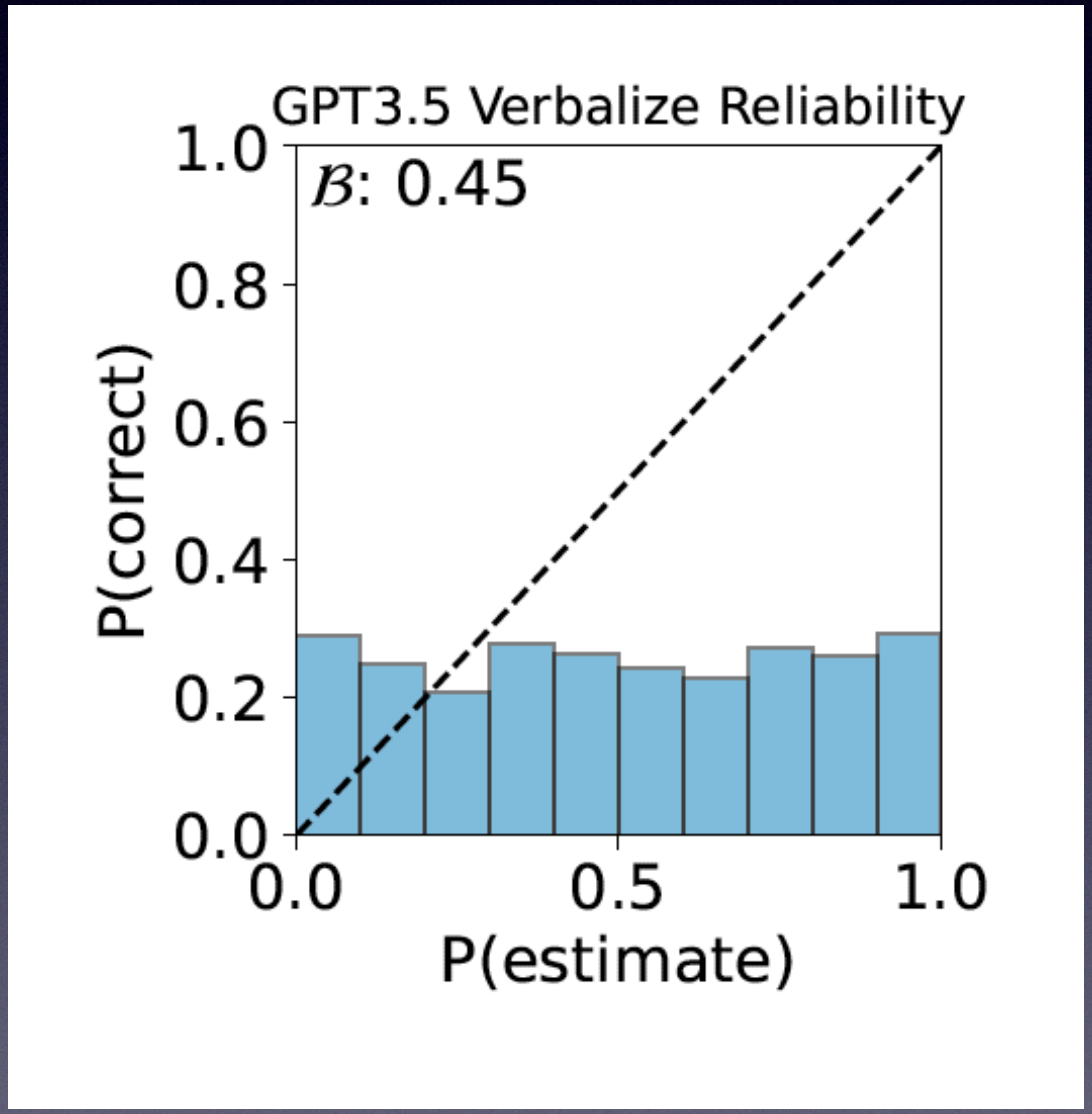
Confidence vs. Correctness

p(model's prediction is correct | model predicts with "π" confidence) = π

Reliability Diagram:

Confidence vs. Correctness

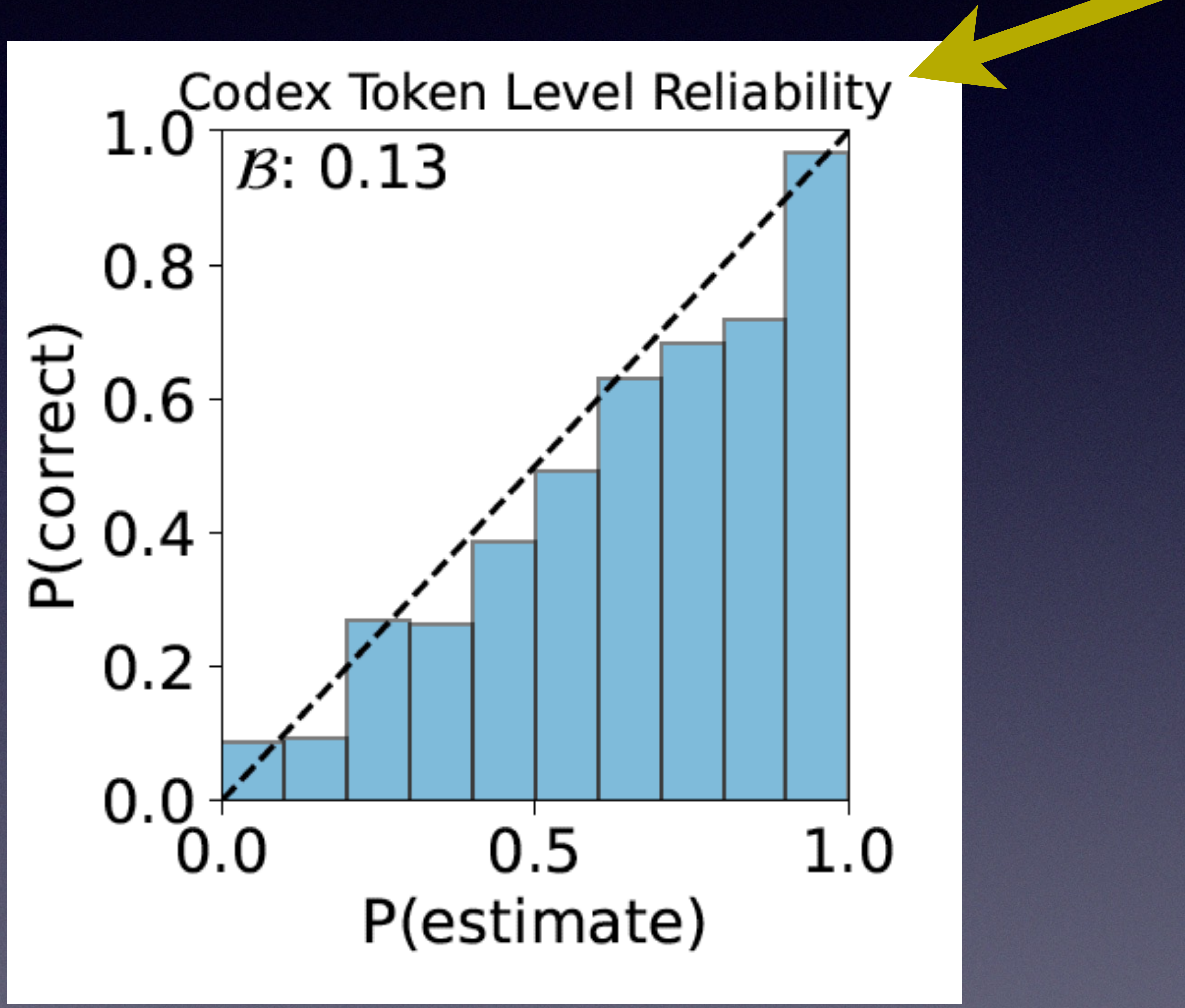


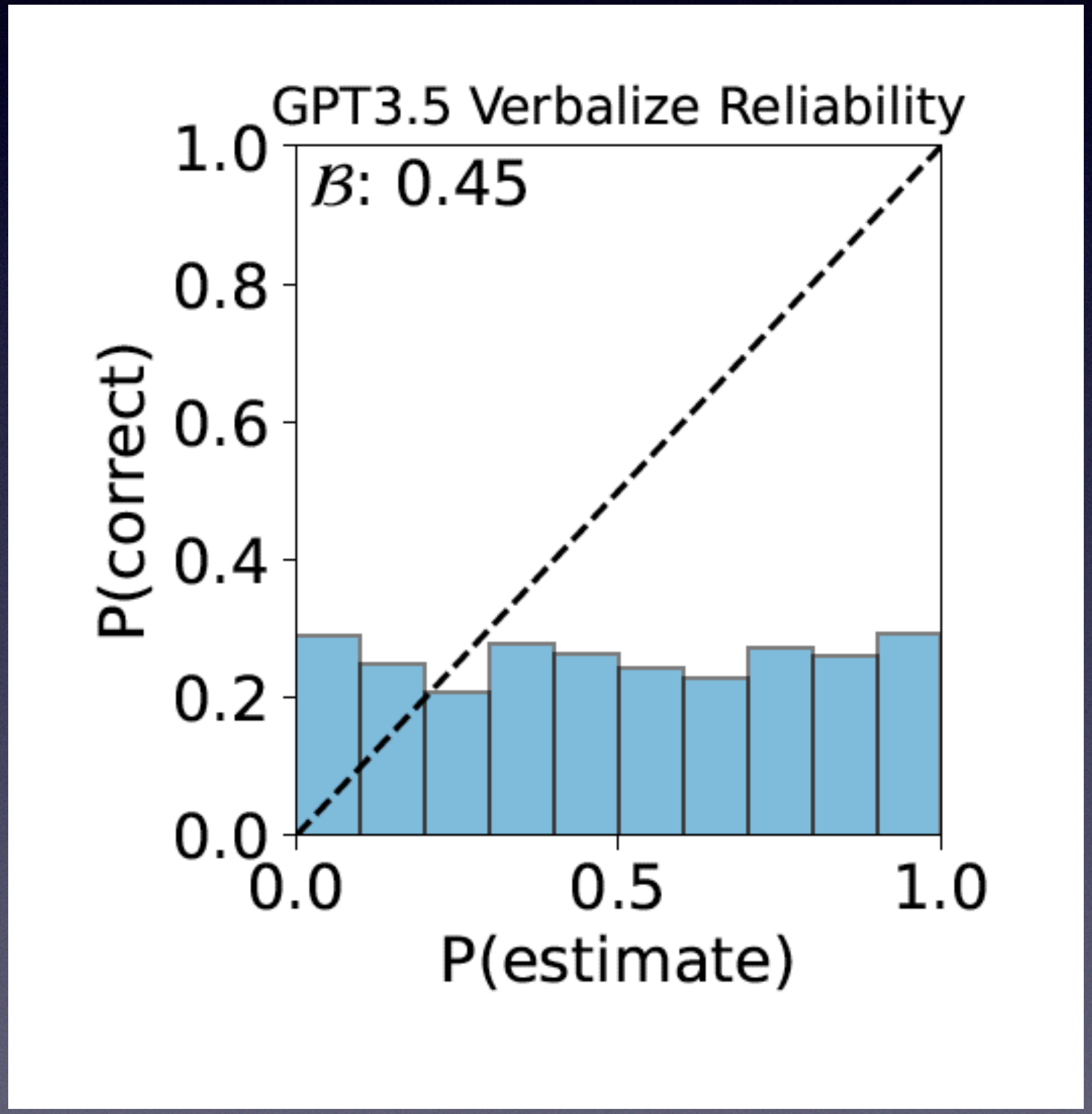


p(model's prediction is correct | model predicts with "π" confidence) = π

Reliability Diagram:

Confidence vs. Correctness

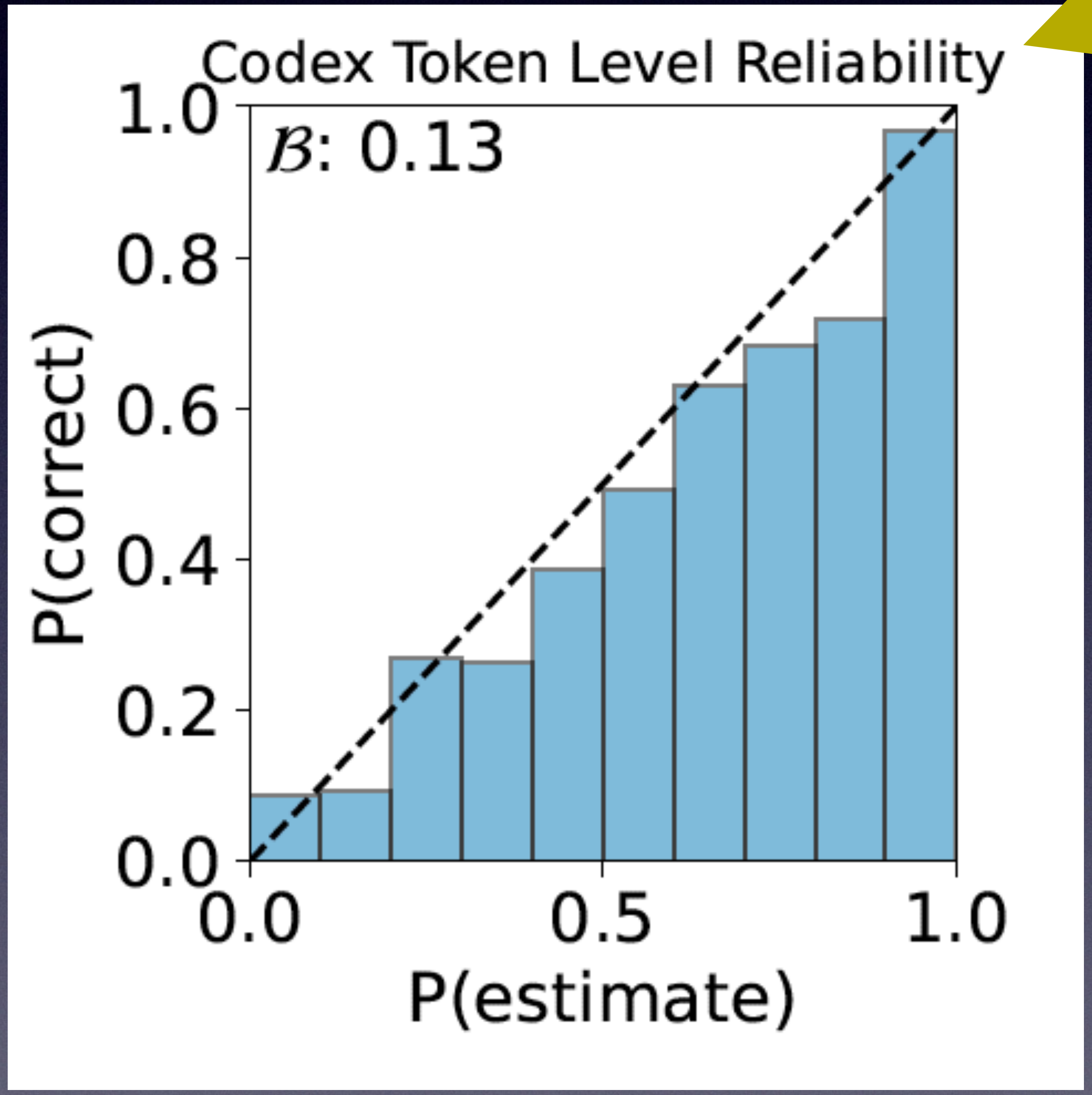


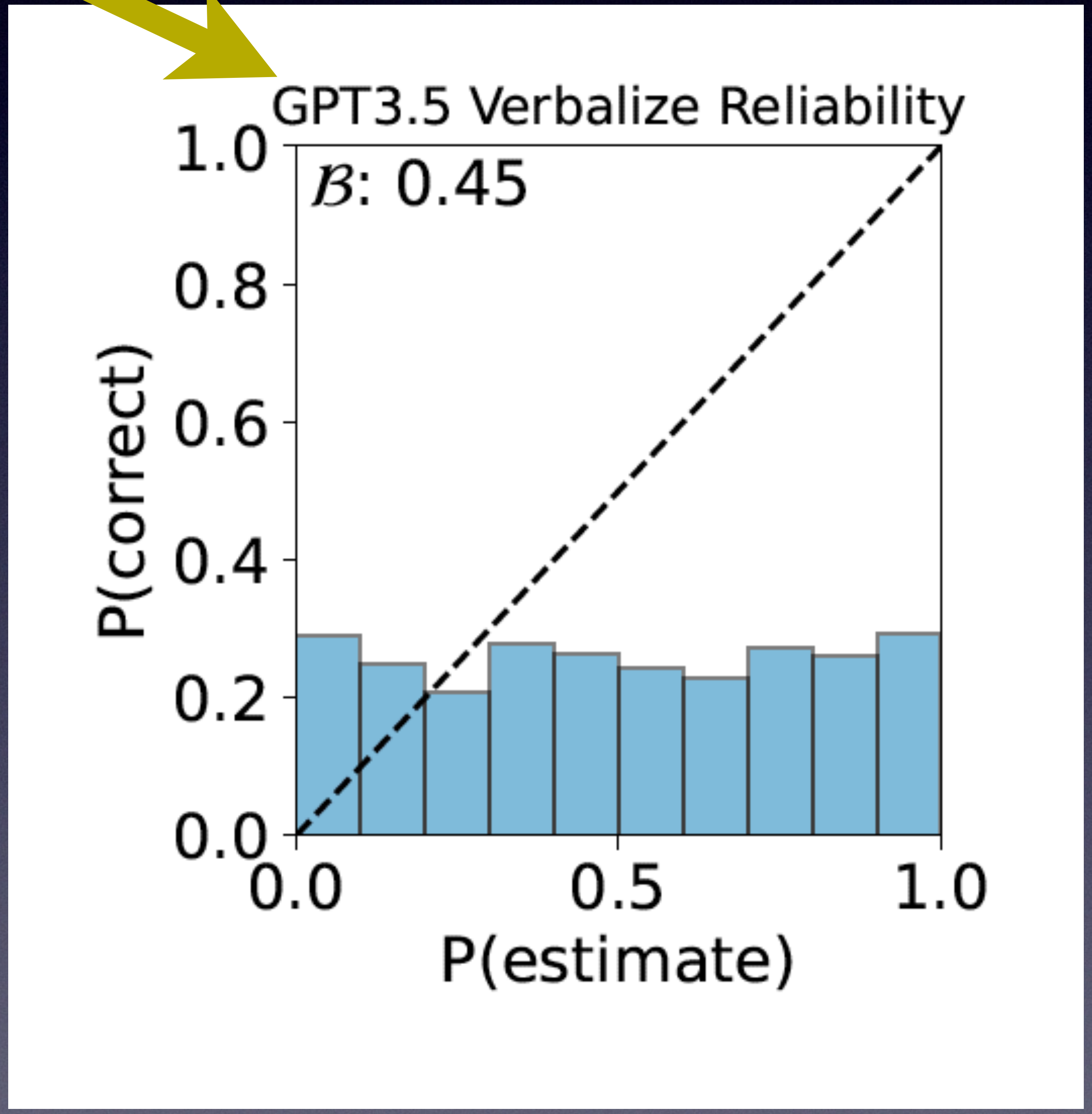


p(model's prediction is correct | model predicts with "π" confidence) = π

Reliability Diagram:

Confidence vs. Correctness

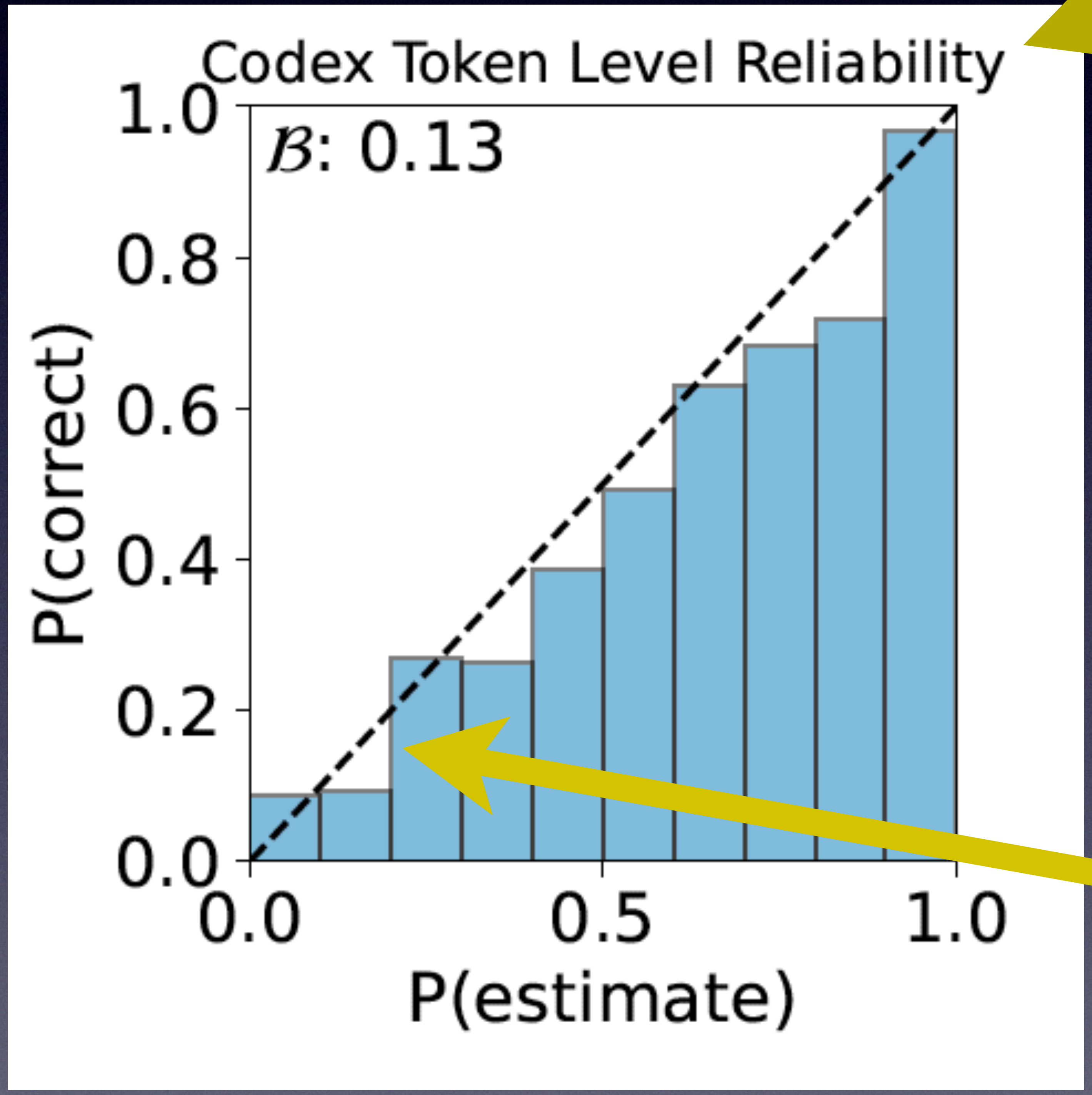




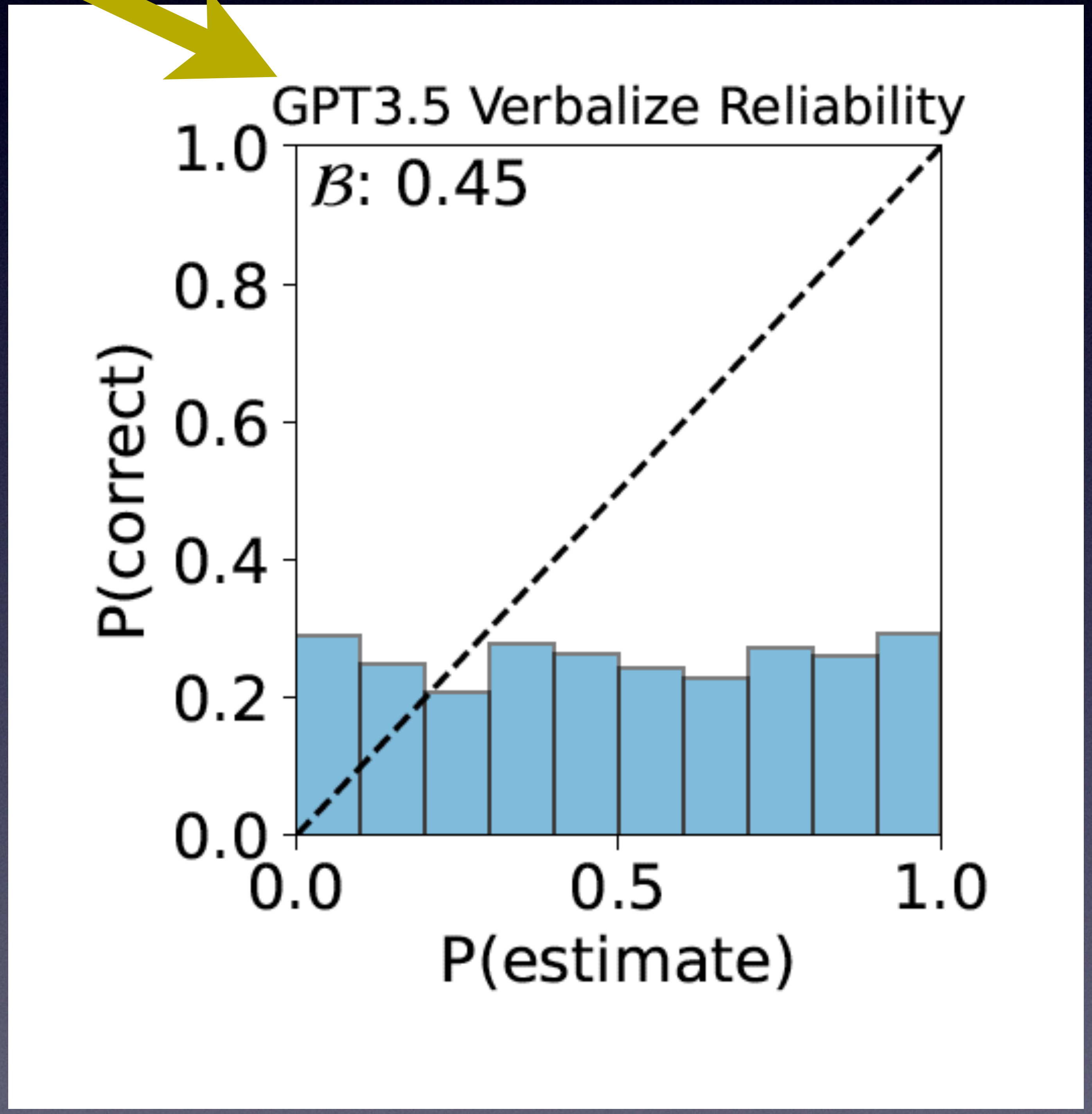
p(model's prediction is correct | model predicts with "π" confidence) = π

Reliability Diagram:

Confidence vs. Correctness



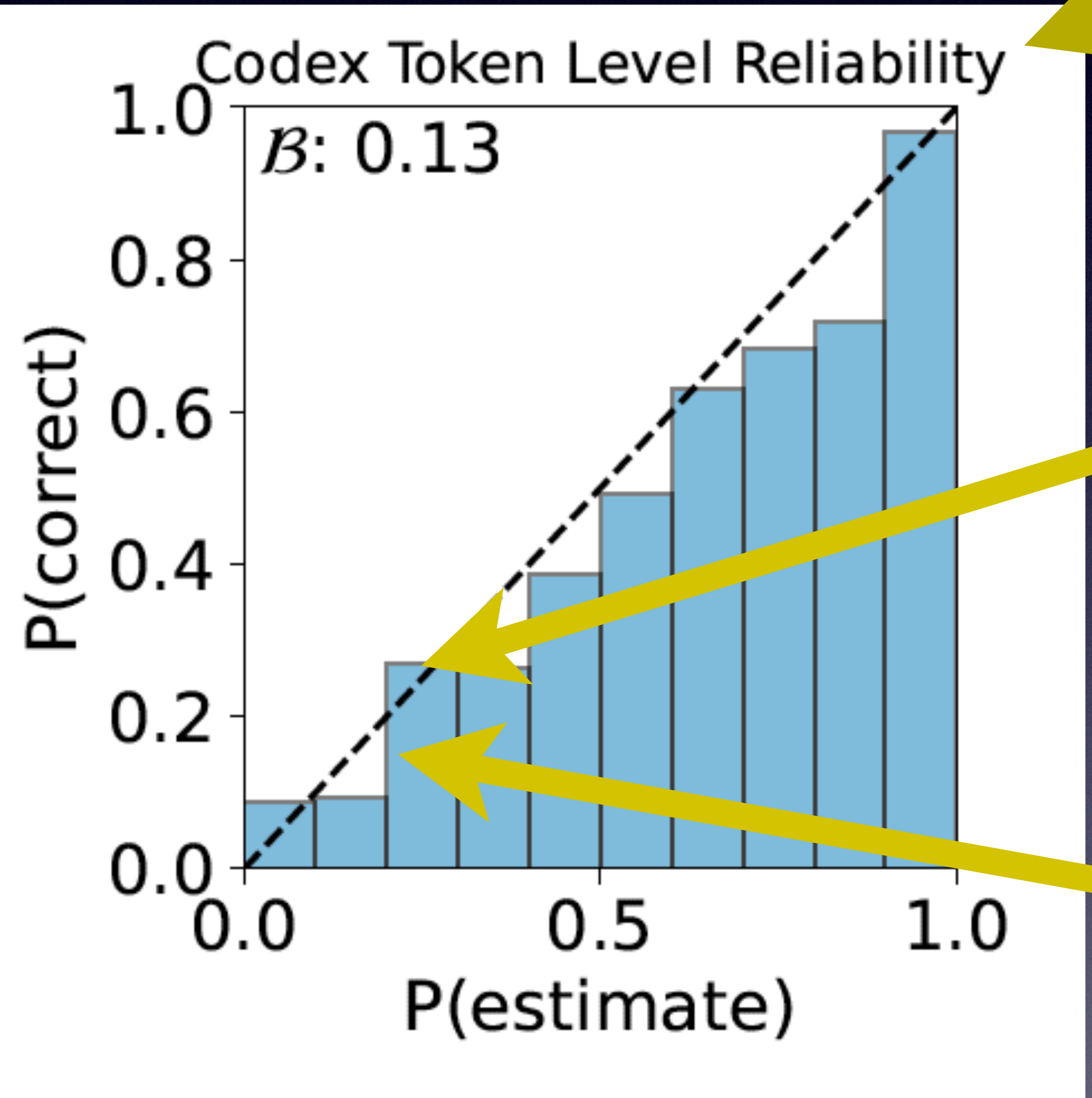
All Predictions
With confidence
Between 0.2 and 0.3



p(model's prediction is correct | model predicts with "π" confidence) = π

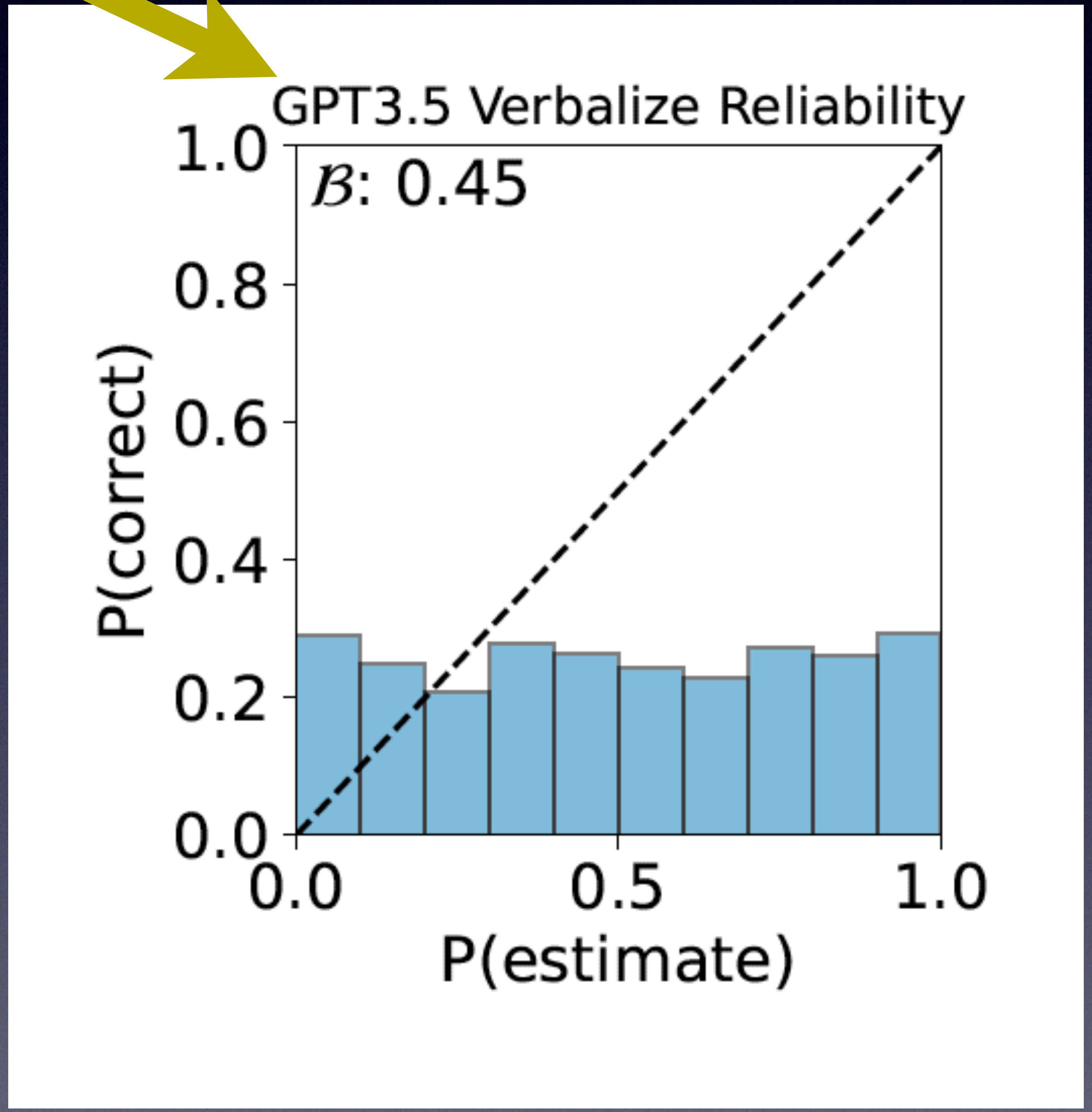
Reliability Diagram:

Confidence vs. Correctness



Predictions in this Range are about 30% Correct (good!)

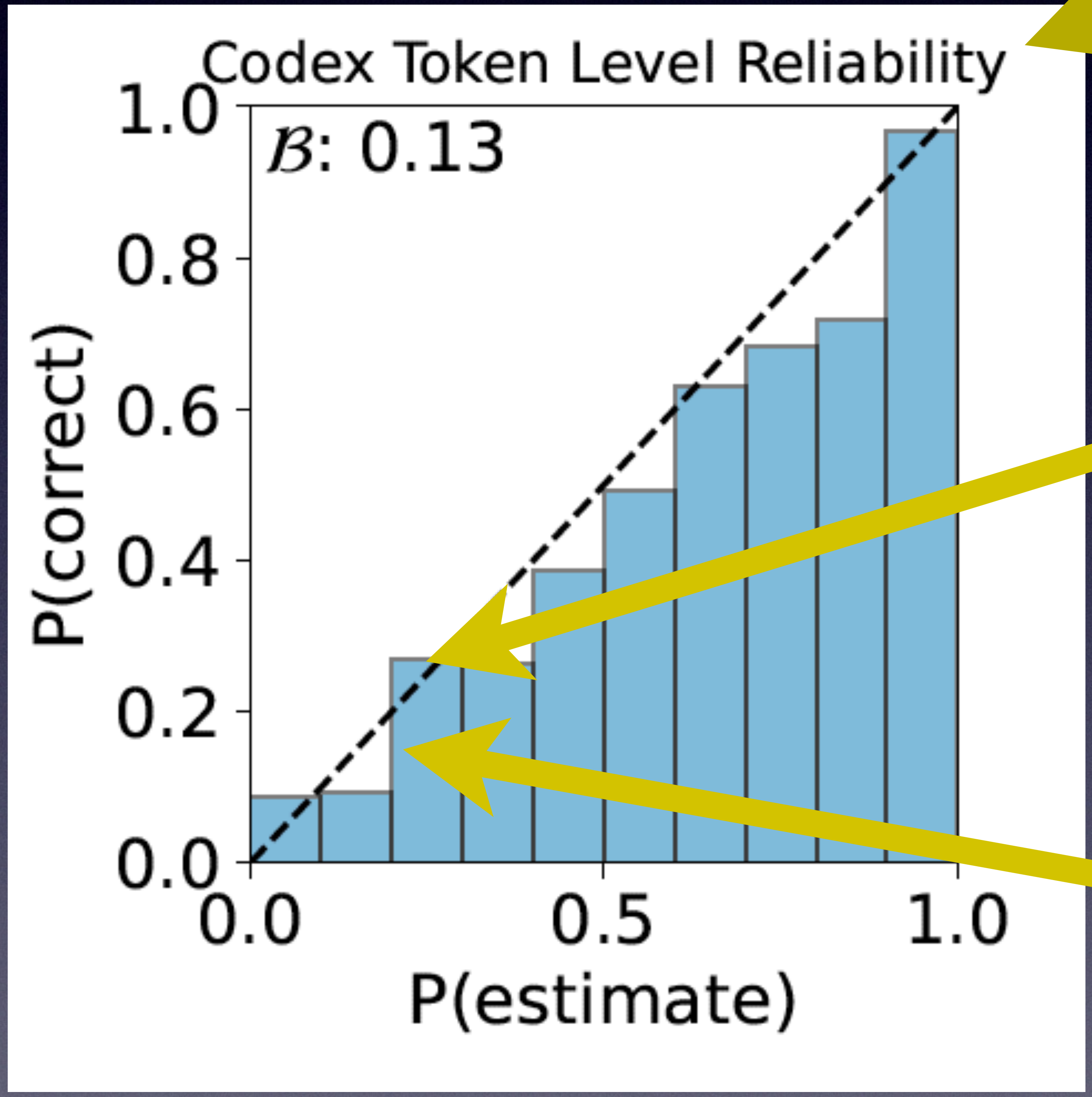
All Predictions With confidence Between 0.2 and 0.3



p(model's prediction is correct | model predicts with "π" confidence) = π

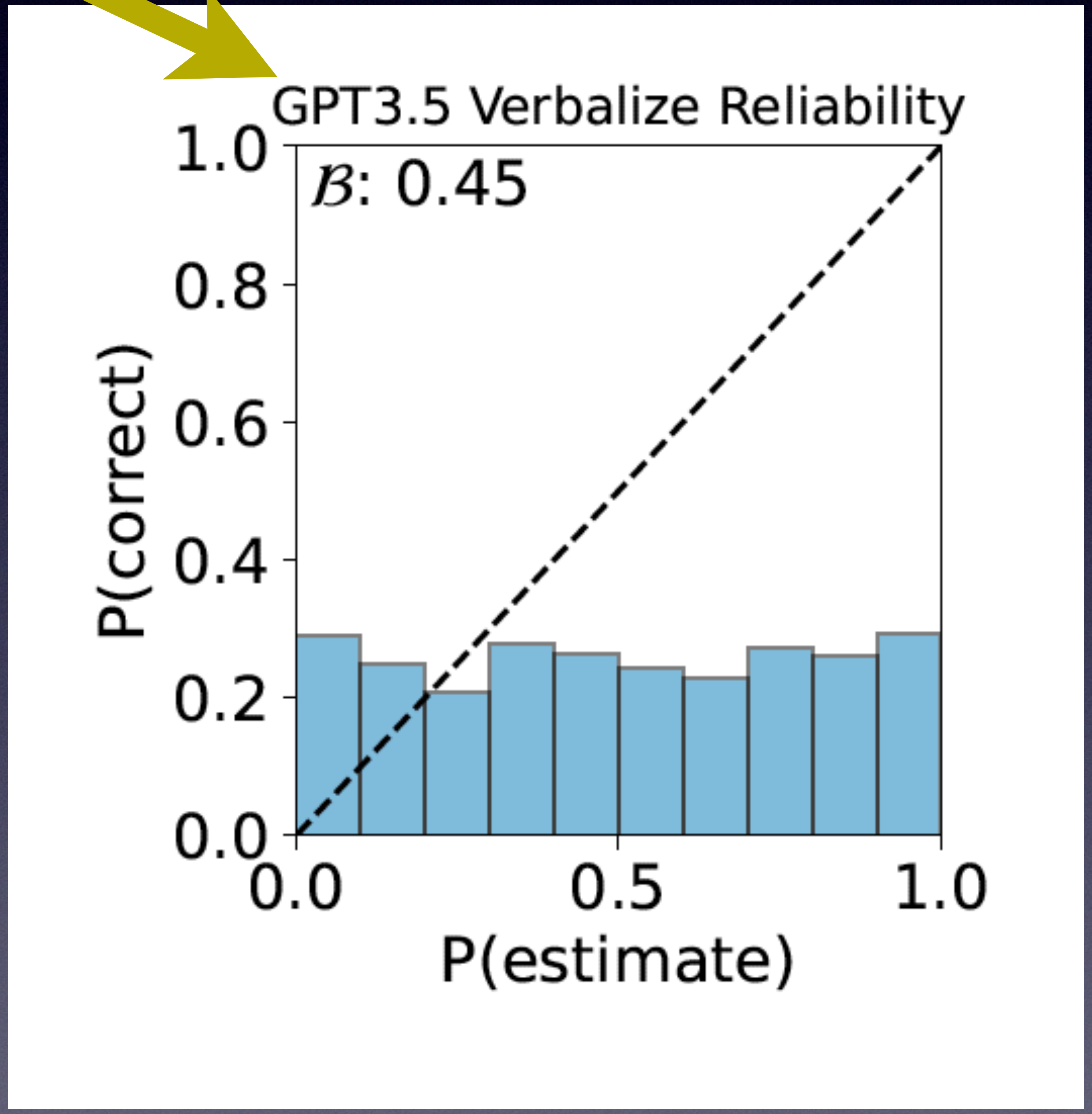
Reliability Diagram:

Confidence vs. Correctness



Predictions in this Range are about 30% Correct (good!)

All Predictions With confidence Between 0.2 and 0.3



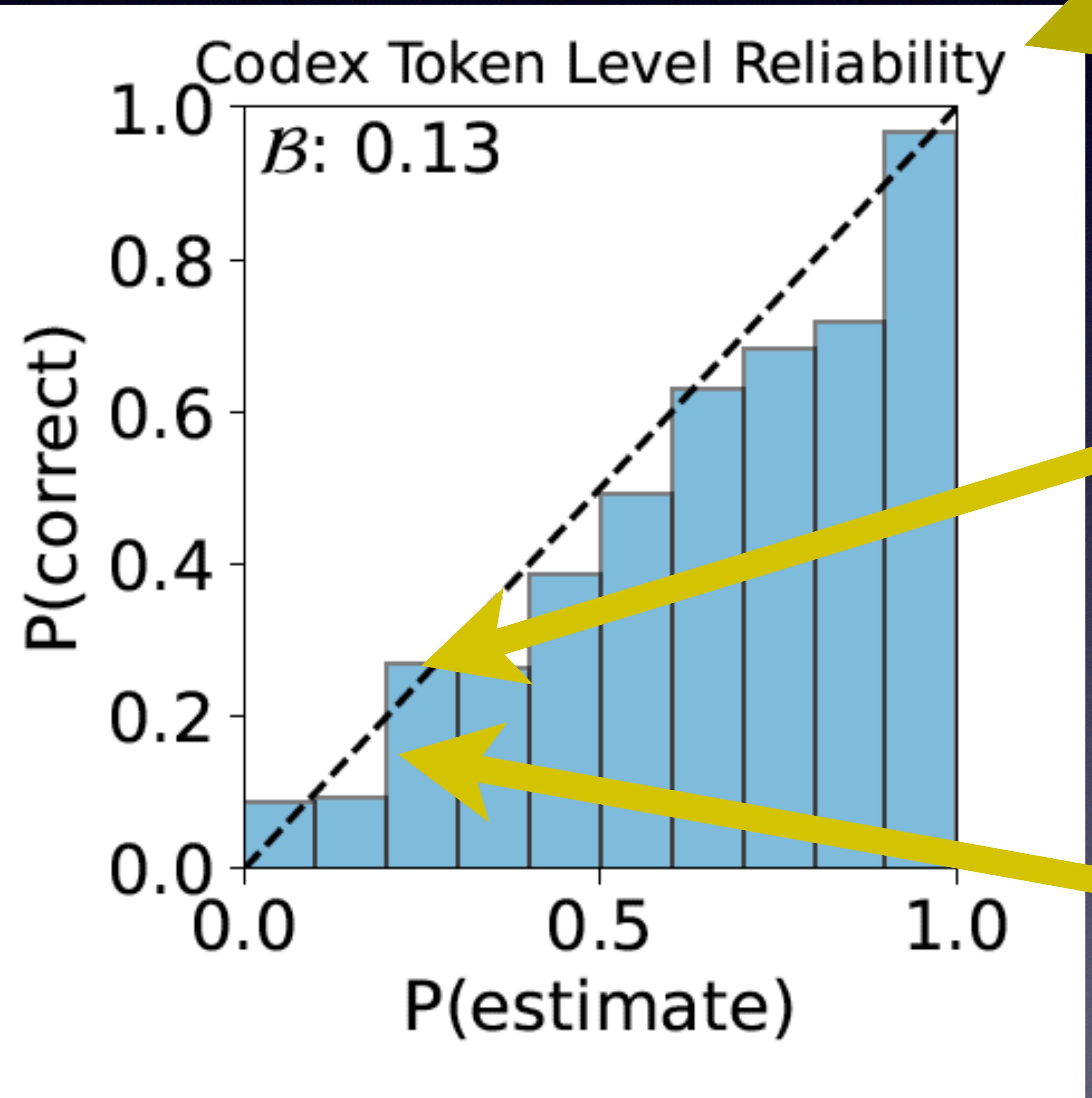
Under Confident

Over Confident

p(model's prediction is correct | model predicts with "π" confidence) = π

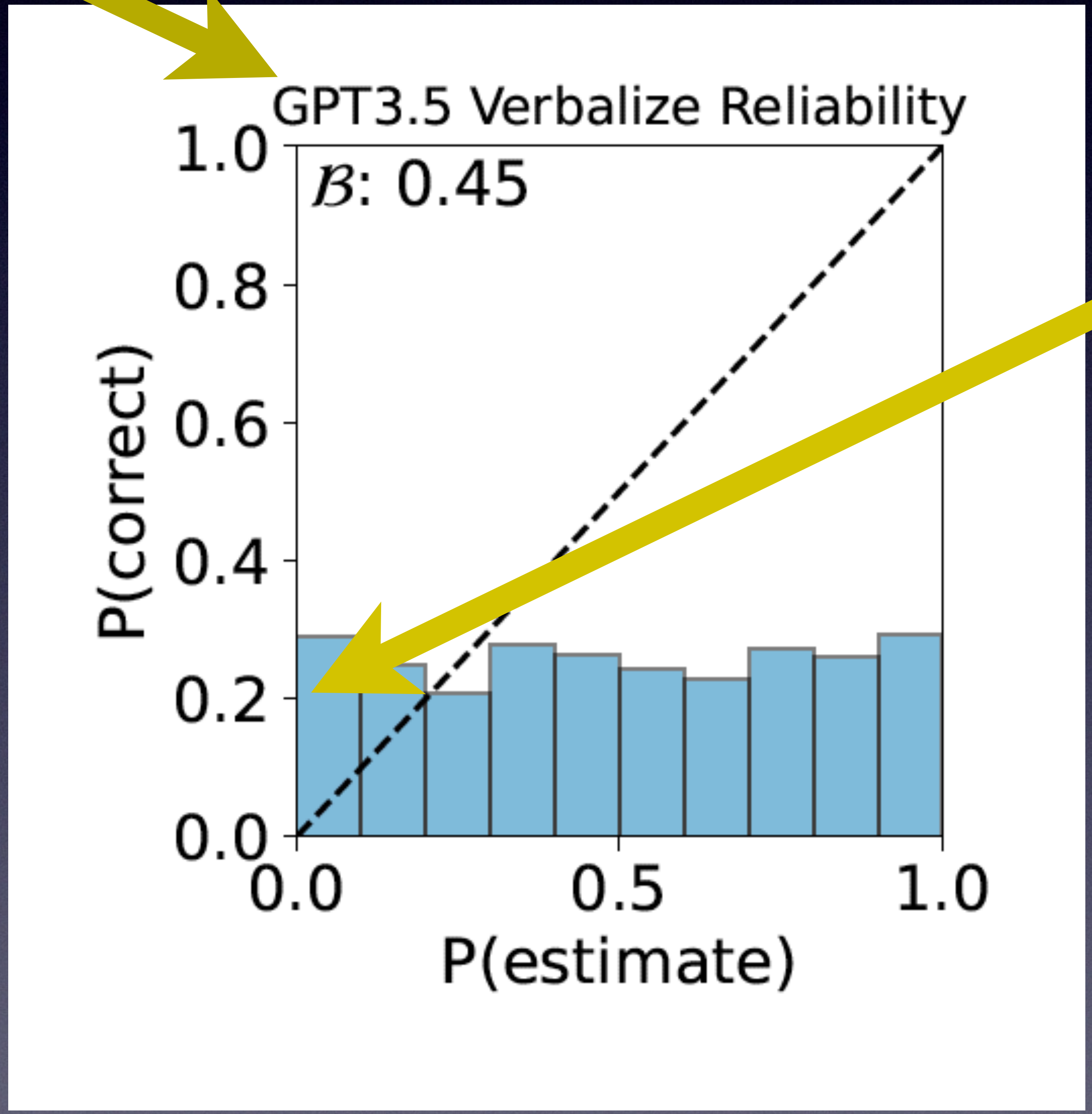
Reliability Diagram:

Confidence vs. Correctness



Predictions in this Range are about 30% Correct (good!)

All Predictions With confidence Between 0.2 and 0.3



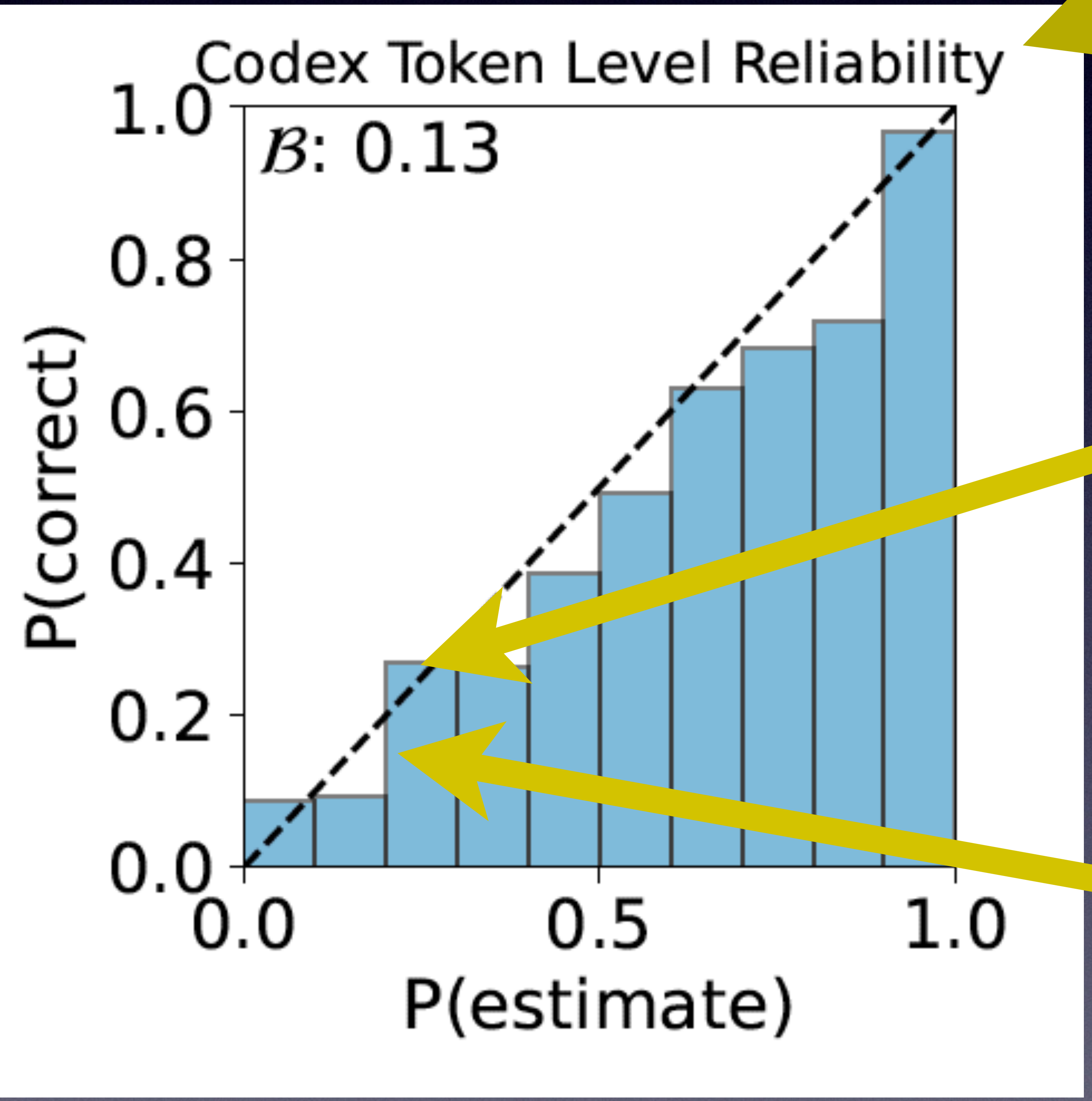
Under Confident

Over Confident

p(model's prediction is correct | model predicts with "π" confidence) = π

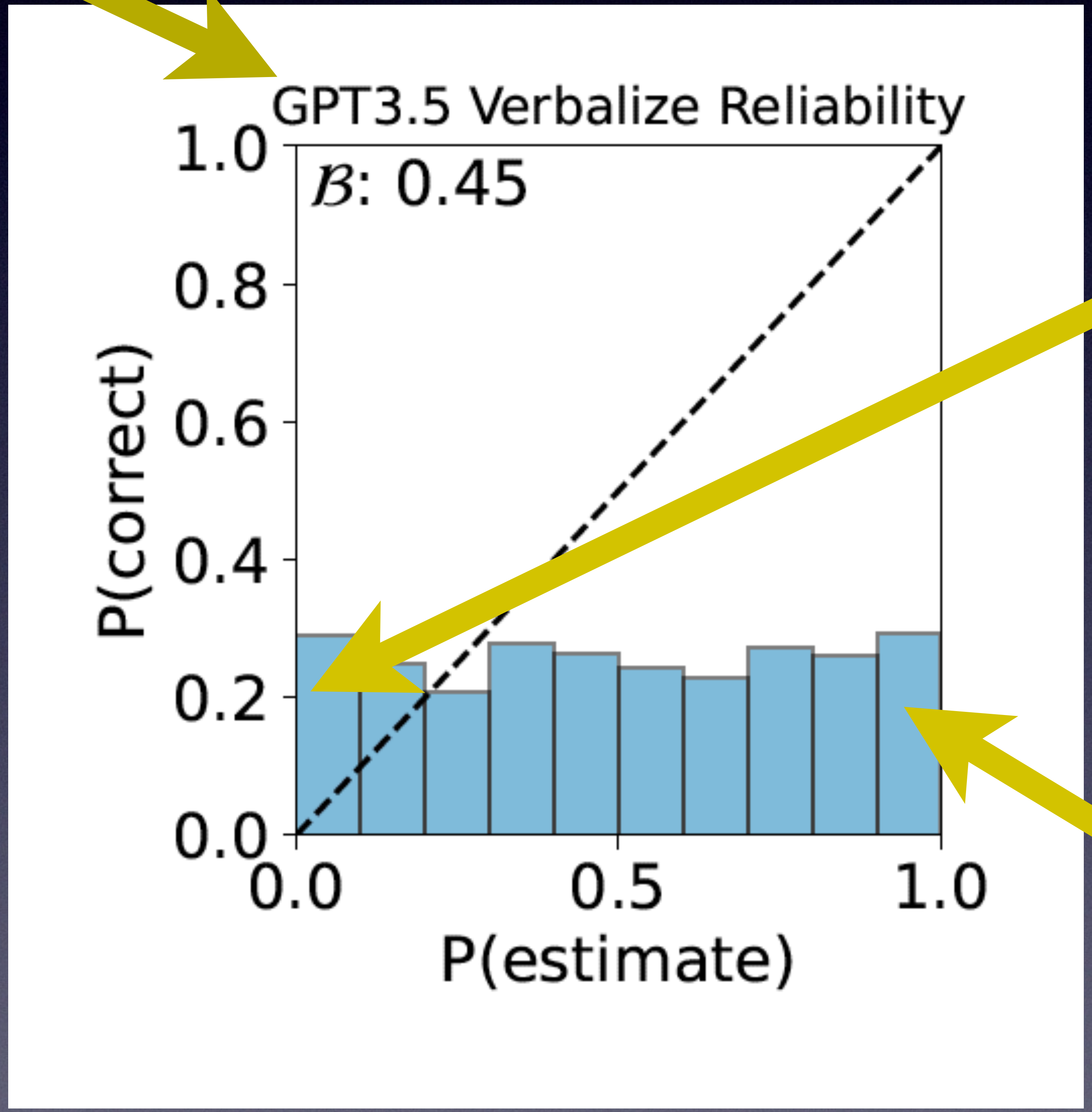
Reliability Diagram:

Confidence vs. Correctness



Predictions in this Range are about 30% Correct (good!)

All Predictions With confidence Between 0.2 and 0.3

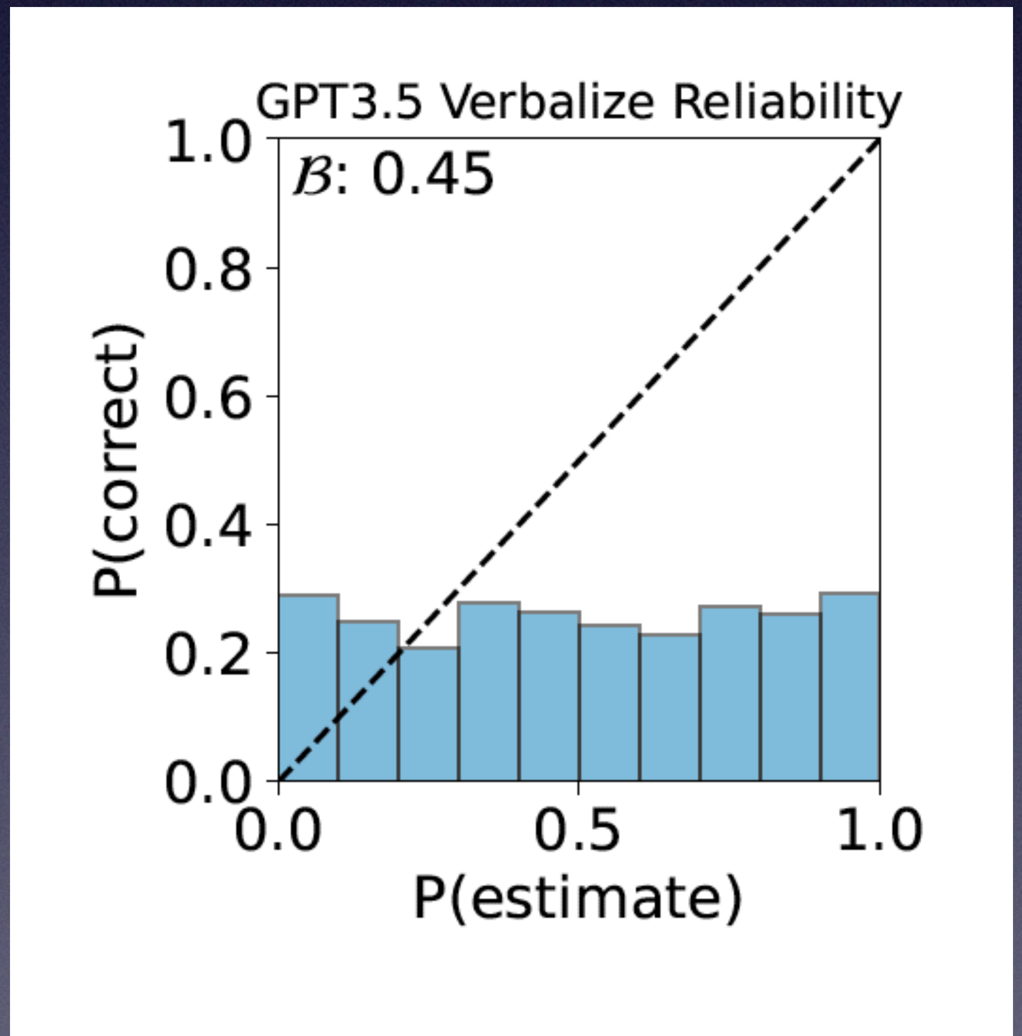


Under Confident

Over Confident

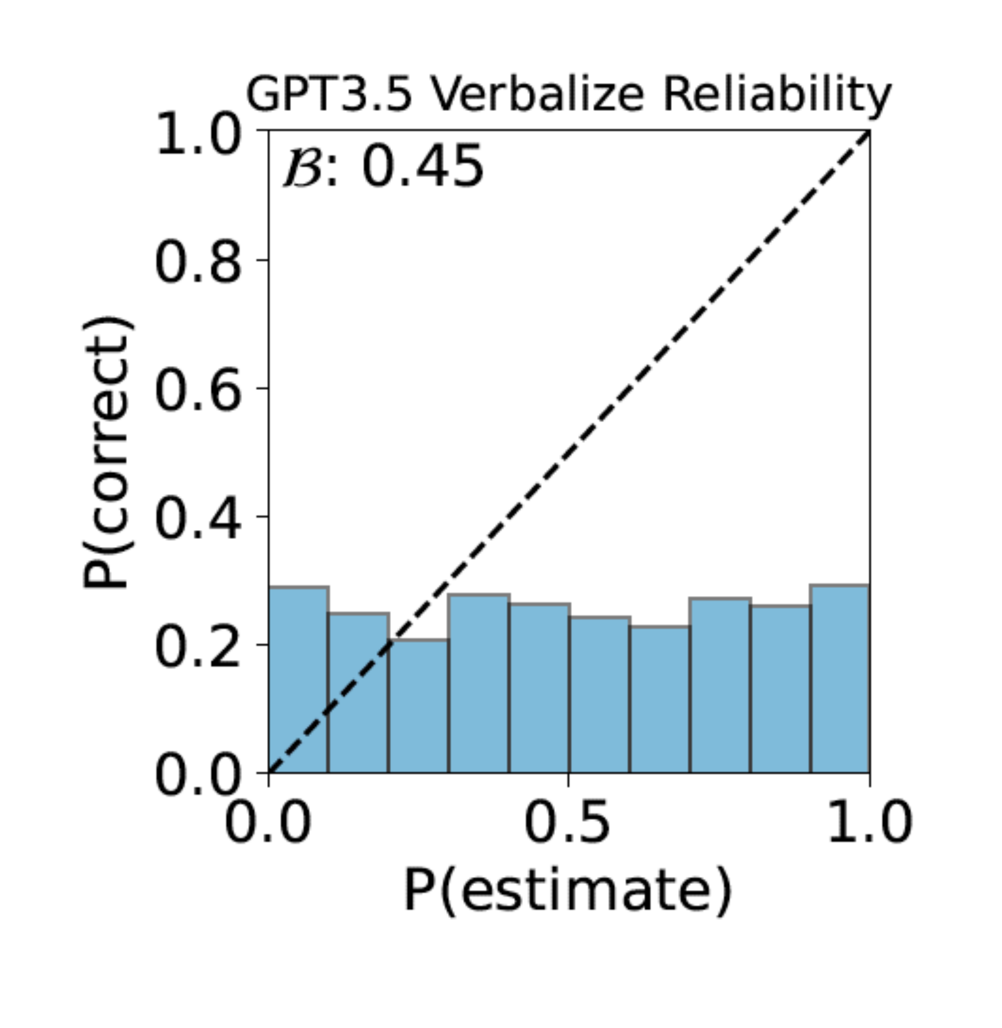
p(model's prediction is correct | model predicts with "π" confidence) = π

Calibration Measures



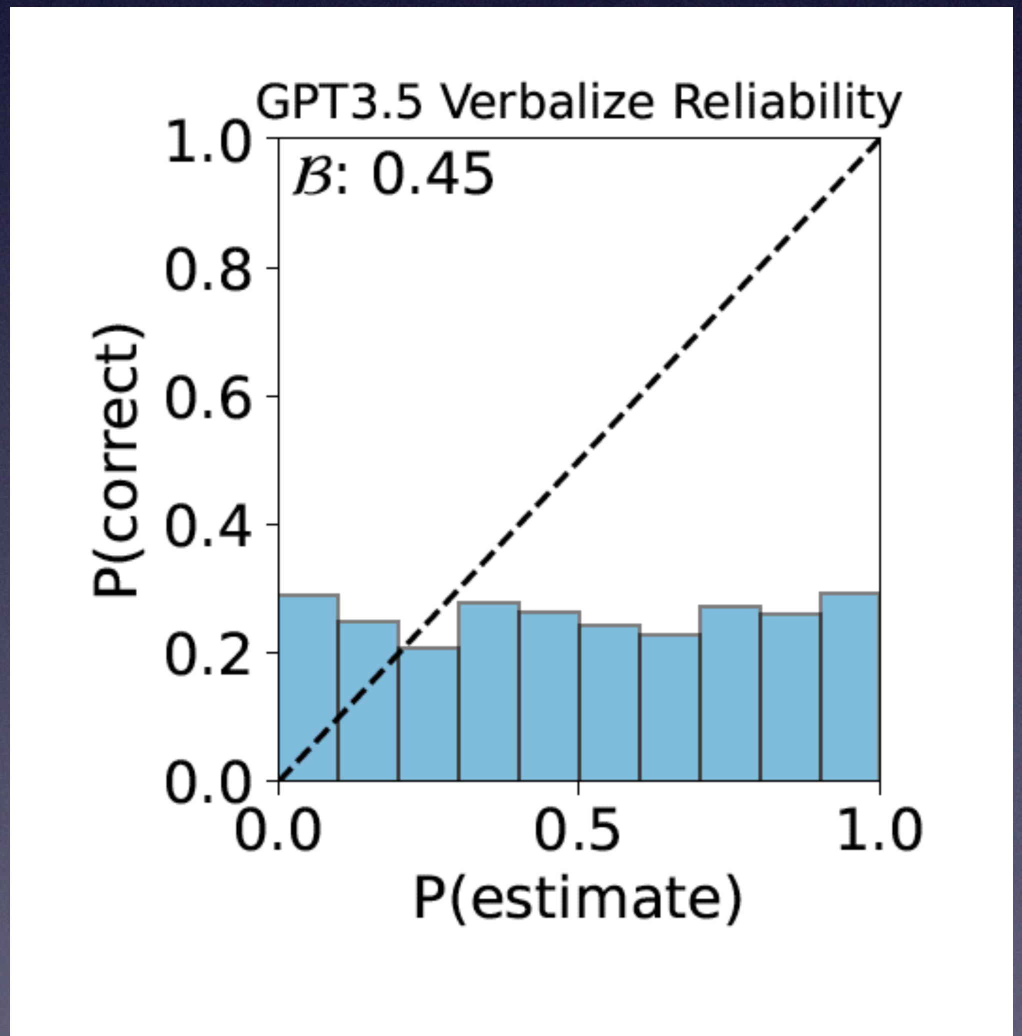
Calibration Measures

$$ECE = \sum_{all\ buckets\ b_i} \frac{|b_i|}{n} * |correct(b_i) - confidence(b_i)|$$



Calibration Measures

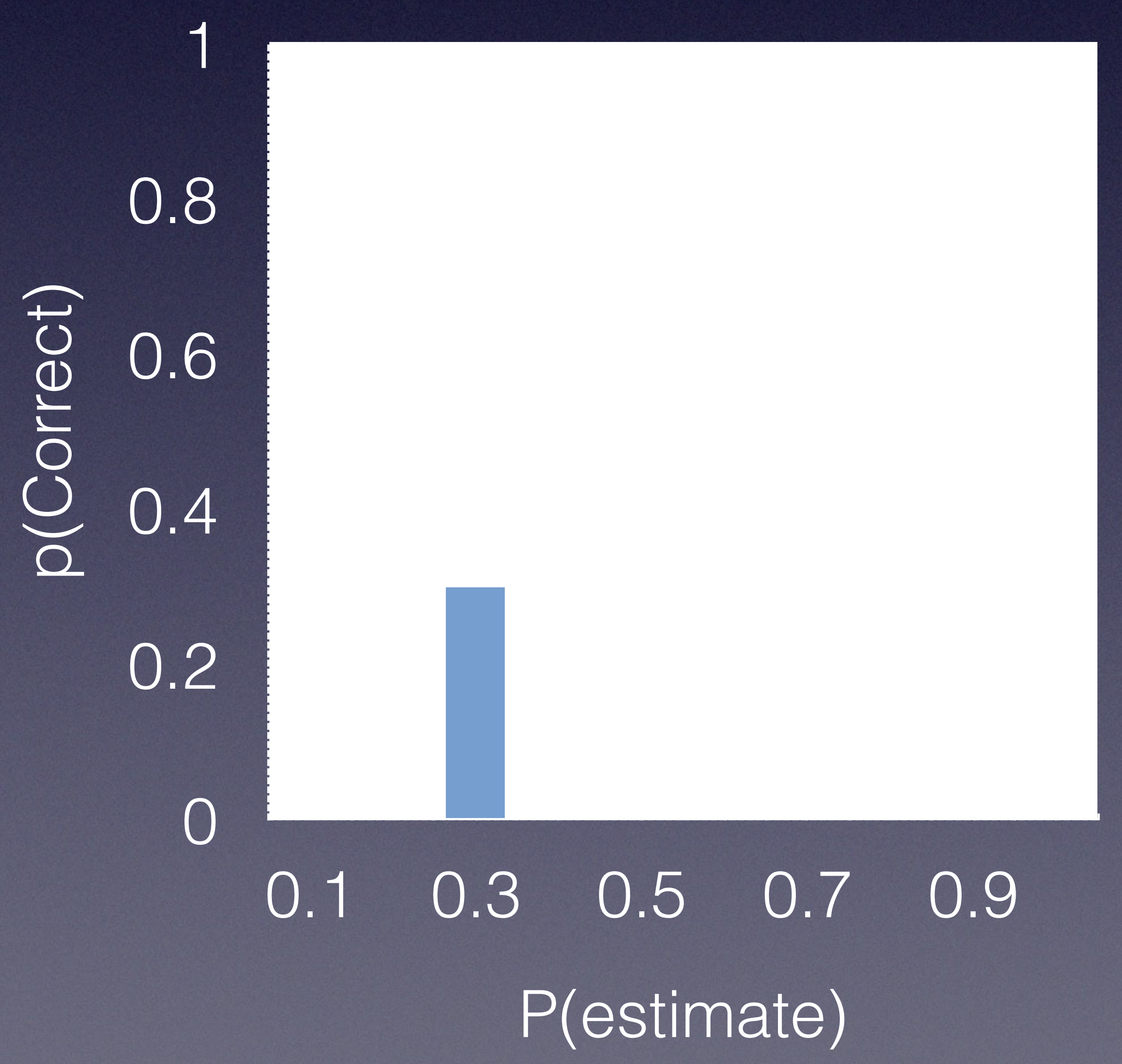
$$ECE = \sum_{all\ buckets\ b_i} \frac{|b_i|}{n} * |correct(b_i) - confidence(b_i)|$$

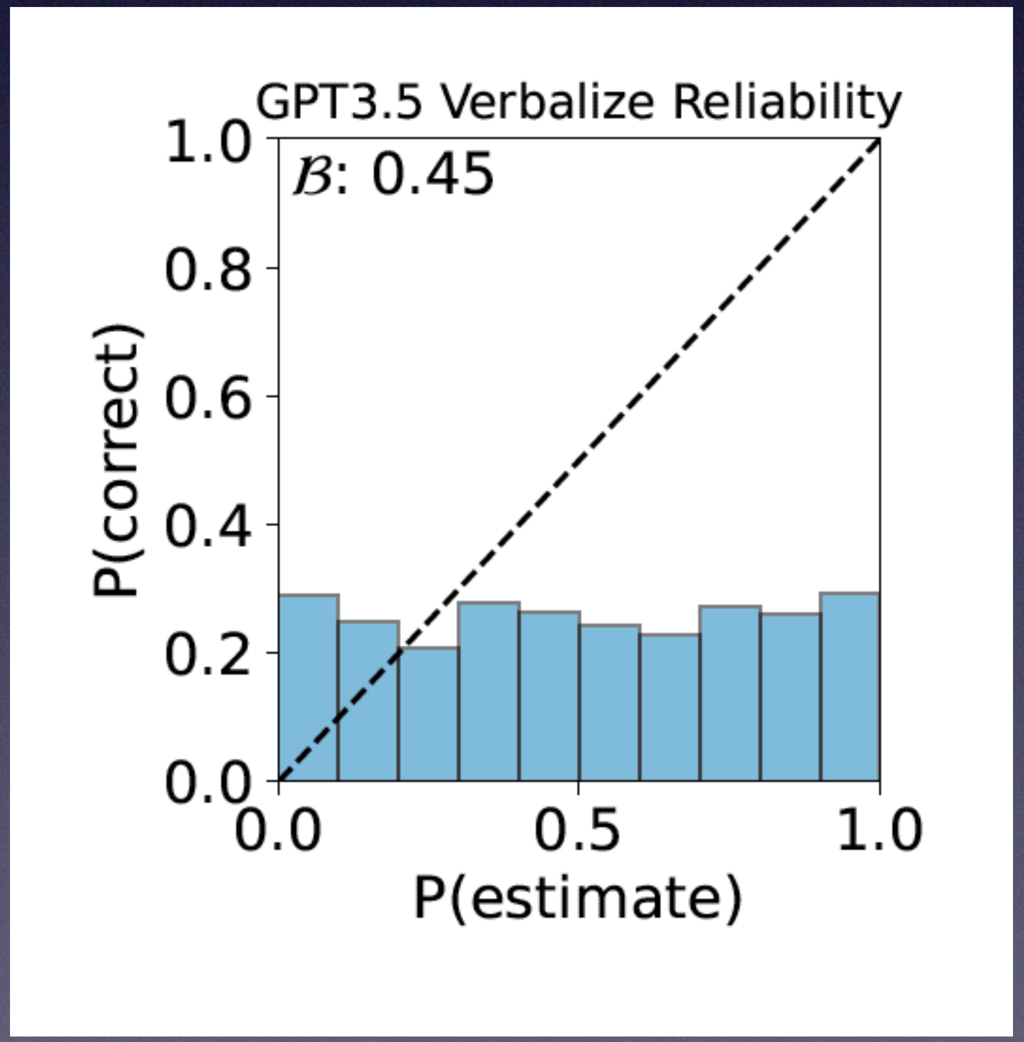


Can “CHEAT”: low ECE by always giving base-rate as Confidence

Calibration Measures

$$ECE = \sum_{all\ buckets\ b_i} \frac{|b_i|}{n} * |correct(b_i) - confidence(b_i)|$$





Can “CHEAT”: low ECE by always giving base-rate as Confidence

Calibration Measures

$$ECE = \sum_{all\ buckets\ b_i} \frac{|\ b_i\ |}{n} * |\ correct(b_i) - confidence(b_i)\ |$$

$$B_{actual} = \frac{1}{n} \sum_{all\ \ n\ samples\ x} \begin{cases} p_{\mathscr{M}}(x, \hat{y})^2 & \text{if prediction } \hat{y} \text{ wrong} \\ (1 - p_{\mathscr{M}}(x, \hat{y}))^2 & \text{otherwise} \end{cases}$$

Calibration Measures

$$ECE = \sum_{all\ buckets\ b_i} \frac{|\ b_i\ |}{n} * |\ correct(b_i) - confidence(b_i)\ |$$

$$B_{actual} = \frac{1}{n} \sum_{all\ \ n\ samples\ x} \begin{cases} p_{\mathscr{M}}(x, \hat{y})^2 & \text{if prediction } \hat{y} \text{ wrong} \\ (1 - p_{\mathscr{M}}(x, \hat{y}))^2 & \text{otherwise} \end{cases}$$

$$B_{ref} = p_{br} * (1 - p_{br})$$

Calibration Measures

$$ECE = \sum_{all\ buckets\ b_i} \frac{|b_i|}{n} * |correct(b_i) - confidence(b_i)|$$

$$B_{actual} = \frac{1}{n} \sum_{all\ \ n\ samples\ x} \begin{cases} p_{\mathscr{M}}(x, \hat{y})^2 & \text{if prediction } \hat{y} \text{ wrong} \\ (1 - p_{\mathscr{M}}(x, \hat{y}))^2 & \text{otherwise} \end{cases}$$

Can't Cheat!! $B_{ref} = p_{br} * (1 - p_{br})$

Calibration Measures

$$B_{actual} = \frac{1}{n} \sum_{all \quad n \ samples \ x} \begin{cases} p_{\mathscr{M}}(x, \hat{y})^2 & \text{if prediction } \hat{y} \text{ wrong} \\ (1 - p_{\mathscr{M}}(x, \hat{y}))^2 & \text{otherwise} \end{cases}$$

$$B_{ref} = p_{br} * (1 - p_{br})$$

Calibration Measures

$$B_{actual} = \frac{1}{n} \sum_{all \;\; n \; samples \; x} \begin{cases} p_{\mathscr{M}}(x, \hat{y})^2 & \text{if prediction } \hat{y} \text{ wrong} \\ (1 - p_{\mathscr{M}}(x, \hat{y}))^2 & \text{otherwise} \end{cases}$$

$$B_{ref} = p_{br} * (1 - p_{br})$$

$$B_{skill} = \frac{B_{ref} - B_{actual}}{B_{ref}}$$

Calibration Measures

$$B_{skill} = \frac{B_{ref} - B_{actual}}{B_{ref}}$$

Calibration Measures

Skill scores *(between -∞ and 1)*

$$B_{skill} = \frac{B_{ref} - B_{actual}}{B_{ref}}$$

Calibration Measures

Skill scores *(between -∞ and 1)*

$$B_{skill} = \frac{B_{ref} - B_{actual}}{B_{ref}}$$

- Unskilled (always base rate) = *0.0*

Calibration Measures

Skill scores *(between -∞ and 1)*

$$B_{skill} = \frac{B_{ref} - B_{actual}}{B_{ref}}$$

- Unskilled (always base rate) = *0.0*
- Deutsche Wetterdienst 🌞☔ = *0.07*

Calibration Measures

Skill scores *(between -∞ and 1)*

$$B_{skill} = \frac{B_{ref} - B_{actual}}{B_{ref}}$$

- Unskilled (always base rate) = *0.0*
- Deutsche Wetterdienst 🌞☔ = *0.07*
- 538, Nat'l B'ball Assoc = *0.13*

Calibration Measures

Skill scores *(between -∞ and 1)*

$$B_{skill} = \frac{B_{ref} - B_{actual}}{B_{ref}}$$

- Unskilled (always base rate) = *0.0*
- Deutsche Wetterdienst 🌞☔ = *0.07*
- 538, Nat’l B’ball Assoc = *0.13*
- 538, Congressional Election = *0.86*

Calibration Measures

Skill scores *(between -∞ and 1)*

$$B_{skill} = \frac{B_{ref} - B_{actual}}{B_{ref}}$$

- Unskilled (always base rate) = *0.0*
- Deutsche Wetterdienst 🌞☔ = *0.07*
- 538, Nat'l B'ball Assoc = *0.13*
- 538, Congressional Election = *0.86*

Out of the Box, LLM probabilties are negative skill.

Calibration and Correctness of Language Models for Code

Claudio Spiess*
UC Davis
USA
cvspiess@ucdavis.edu

David Gros*
UC Davis
USA
dgros@ucdavis.edu

Kunal Suresh Pai
UC Davis
USA
kunpai@ucdavis.edu

Michael Pradel
Univ. of Stuttgart
Germany
michael@binaervarianz.de

Md Rafiqul Islam Rabin
Univ. of Houston
USA
mrabin@central.uh.edu

Amin Alipour
Univ. of Houston
USA
maalipou@central.uh.edu

Susmit Jha
SRI
USA
susmit.jha@sri.com

Prem Devanbu
UC Davis
USA
ptdevanbu@ucdavis.edu

Toufique Ahmed
UC Davis
USA
tfahmed@ucdavis.edu

ICSE 2025, to appear

Datasets

Task	Dataset	Dataset Size	Correctness Measure	Confidence Measure	Calibration Metric
Function synthesis	HumanEval	164	Test-passing Correctness	Average Token Probability, Generated Sequence Probability, Verbalized Self-Evaluation, Question Answering Logit	Brier Score, ECE
	MBBP Func	880			
Line-level Completion	DyPyBench	1,988	Test-passing Correctness, EM		
Program Repair	Defects4J 1-line	120			
	ManySStubs4j	3,000	Exact-Match (EM)		

Datasets

Task	Dataset	Dataset Size	Correctness Measure	Confidence Measure	Calibration Metric
Function synthesis	HumanEval	164	Test-passing Correctness	Average Token Probability, Generated Sequence Probability, Verbalized Self-Evaluation, Question Answering Logit	Brier Score, ECE
	MBBP Func	880			
Line-level Completion	DyPyBench	1,988	Test-passing Correctness, EM		
Program Repair	Defects4J 1-line	120			
	ManySStubs4j	3,000	Exact-Match (EM)		

- Function synthesis is 🐁: small, self-contained toy problems

Datasets

Task	Dataset	Dataset Size	Correctness Measure	Confidence Measure	Calibration Metric
Function synthesis	HumanEval	164	Test-passing Correctness	Average Token Probability, Generated Sequence Probability, Verbalized Self-Evaluation, Question Answering Logit	Brier Score, ECE
	MBBP Func	880			
Line-level Completion	DyPyBench	1,988	Test-passing Correctness, EM		
Program Repair	Defects4J 1-line	120			
	ManySStubs4j	3,000	Exact-Match (EM)		

- Function synthesis is 🐁: small, self-contained toy problems
- DyPyBench dataset consists of thousands of functions **with** docstrings and running test suites in a line completion setting

Datasets

Task	Dataset	Dataset Size	Correctness Measure	Confidence Measure	Calibration Metric
Function synthesis	HumanEval	164	Test-passing Correctness	Average Token Probability, Generated Sequence Probability, Verbalized Self-Evaluation, Question Answering Logit	Brier Score, ECE
	MBBP Func	880			
Line-level Completion	DyPyBench	1,988	Test-passing Correctness, EM		
Program Repair	Defects4J 1-line	120			
	ManySStubs4j	3,000	Exact-Match (EM)		

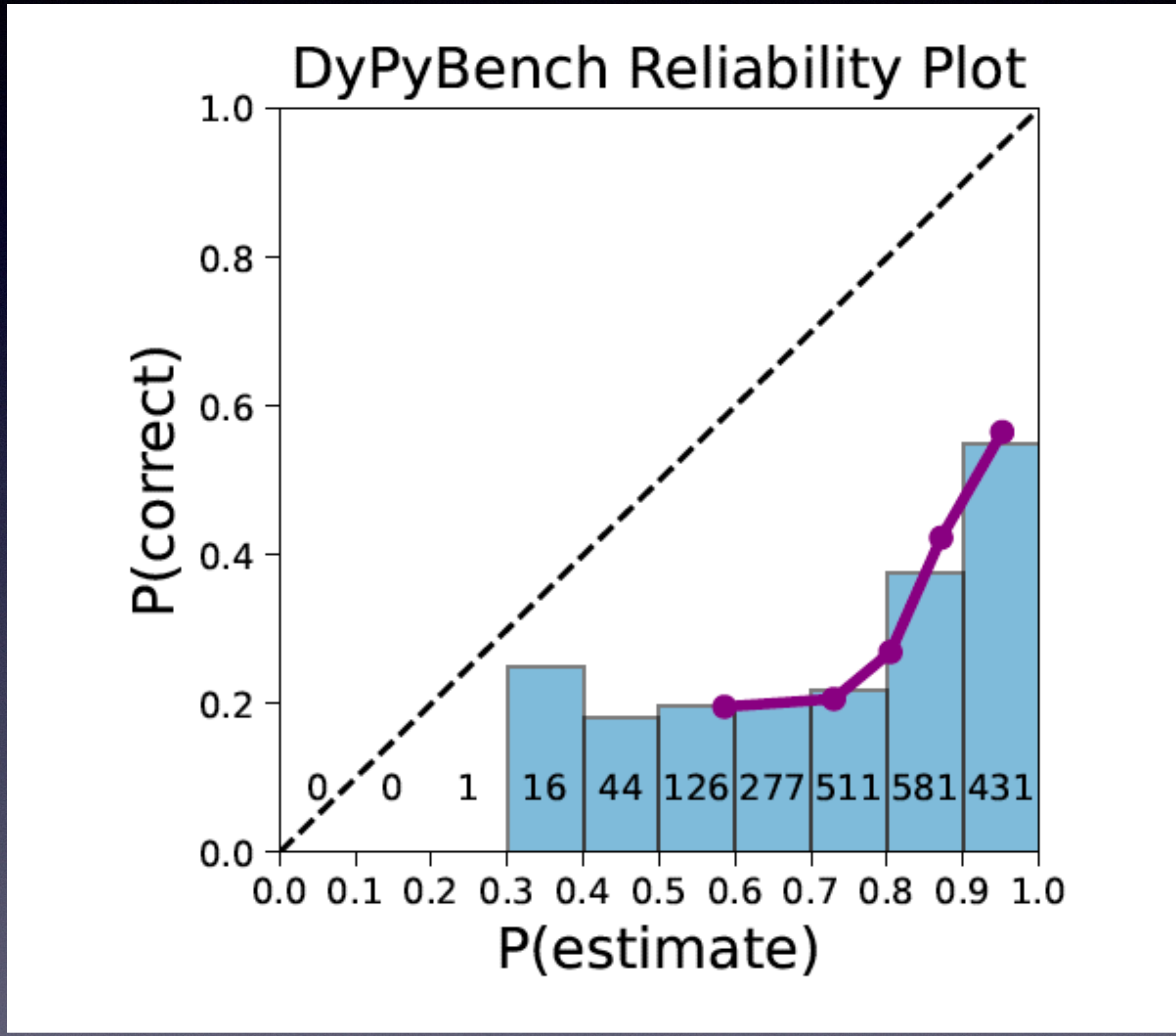
- Function synthesis is 🐁: small, self-contained toy problems
- DyPyBench dataset consists of thousands of functions **with** docstrings and running test suites in a line completion setting
- SStubs4J does not have tests, only exact-match

Models

	All Pass@1			Exact-Match		
	CodeGen2	Codex	GPT-3.5	CodeGen2	Codex	GPT-3.5
SStubs	-	-	-	0.73%	**27.77%**	20.27%
DyPyBench	28.84%	32.96%	**33.22%**	19.68%	23.60%	**23.96%**
Defects4J	0.00%	**23.33%**	19.17%	0.00%	**19.17%**	15.00%
HumanEval	23.17%	47.24%	**64.60%**	-	-	-
MBPP	29.08%	61.79%	**72.04%**	-	-	-

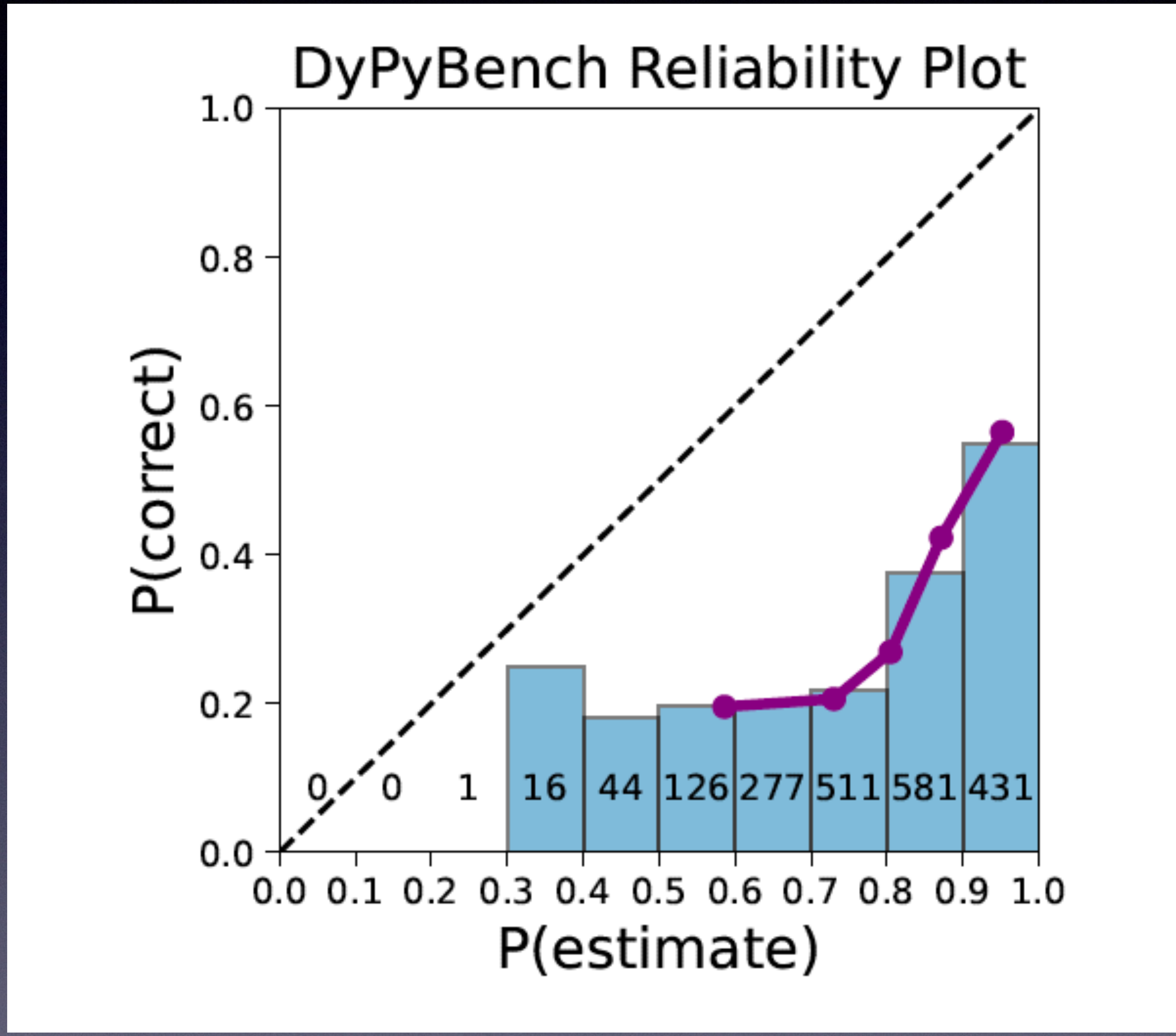
- Each cell ~ the *base rate*
- *Thus All Pass @1 Brier Ref Score for DyPyBench, Codex = 0.33*0.67 = 0.22*

GPT 3.5, Line Completion



Correctness: Test Passing, ***Confidence:*** Avg Per-token probability

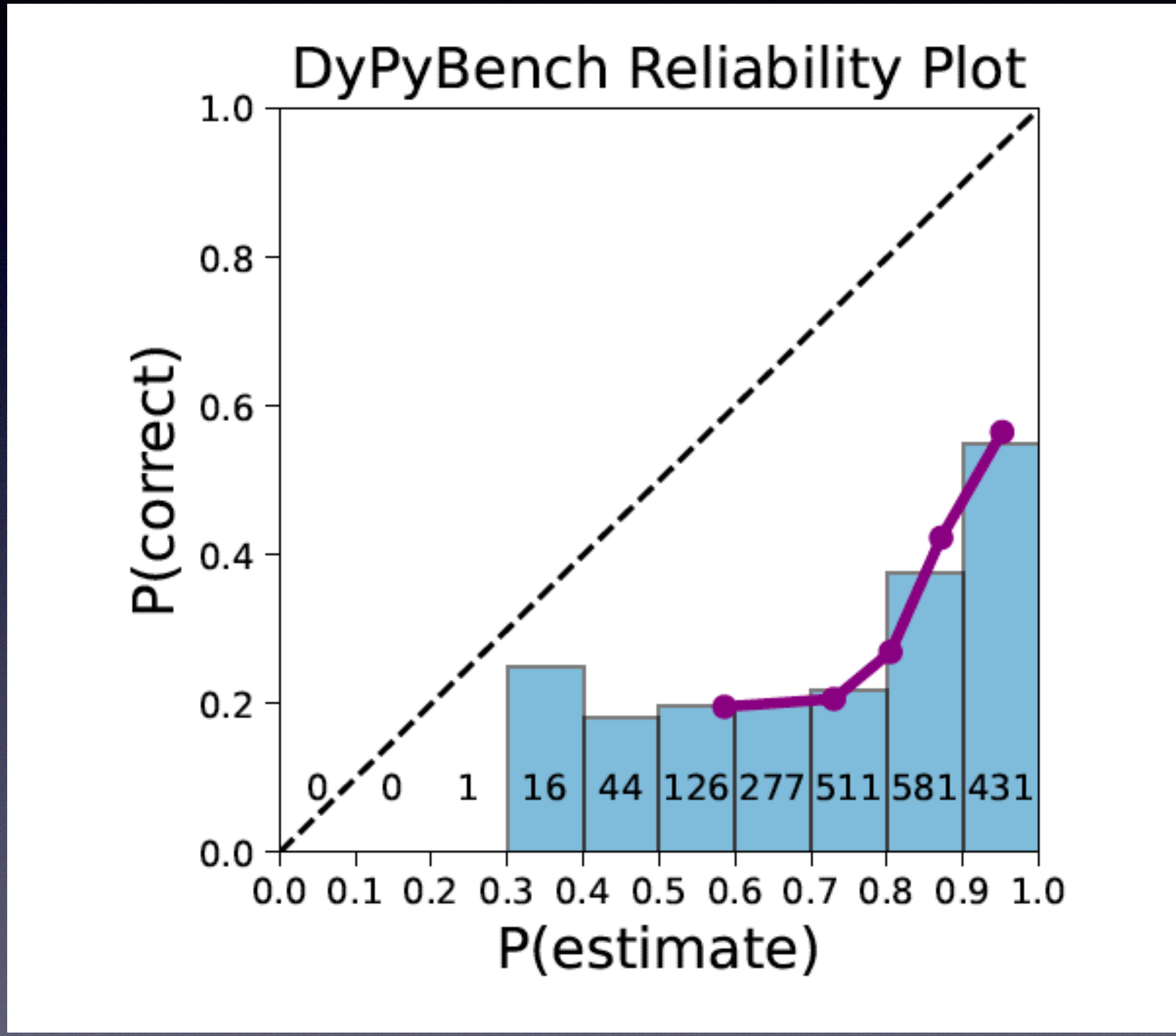
GPT 3.5, Line Completion



ECE = *0.15*

Correctness: Test Passing, ***Confidence:*** Avg Per-token probability

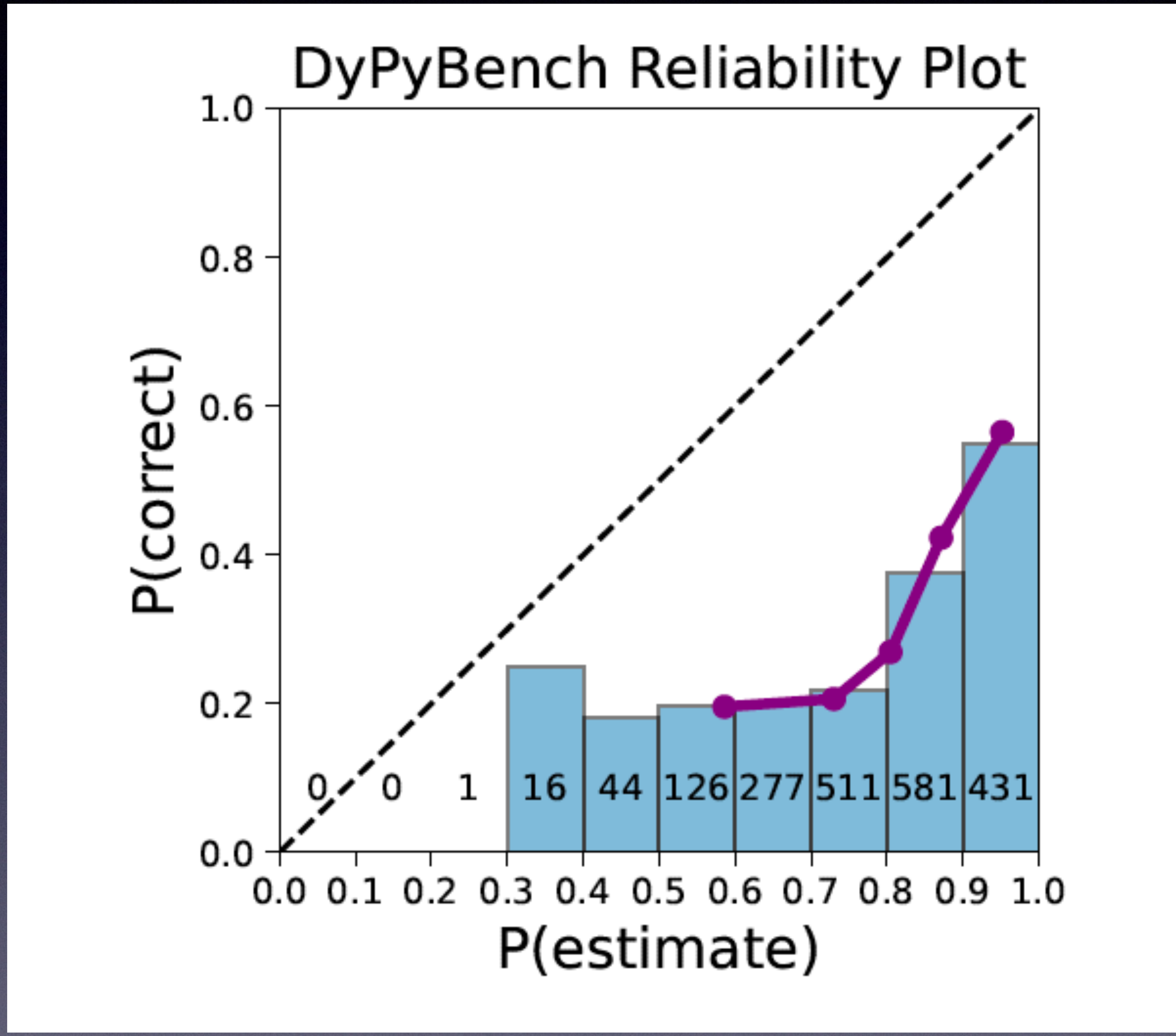
GPT 3.5, Line Completion



ECE = *0.15*
Brier (actual) = *0.41*

Correctness: Test Passing, ***Confidence:*** Avg Per-token probability

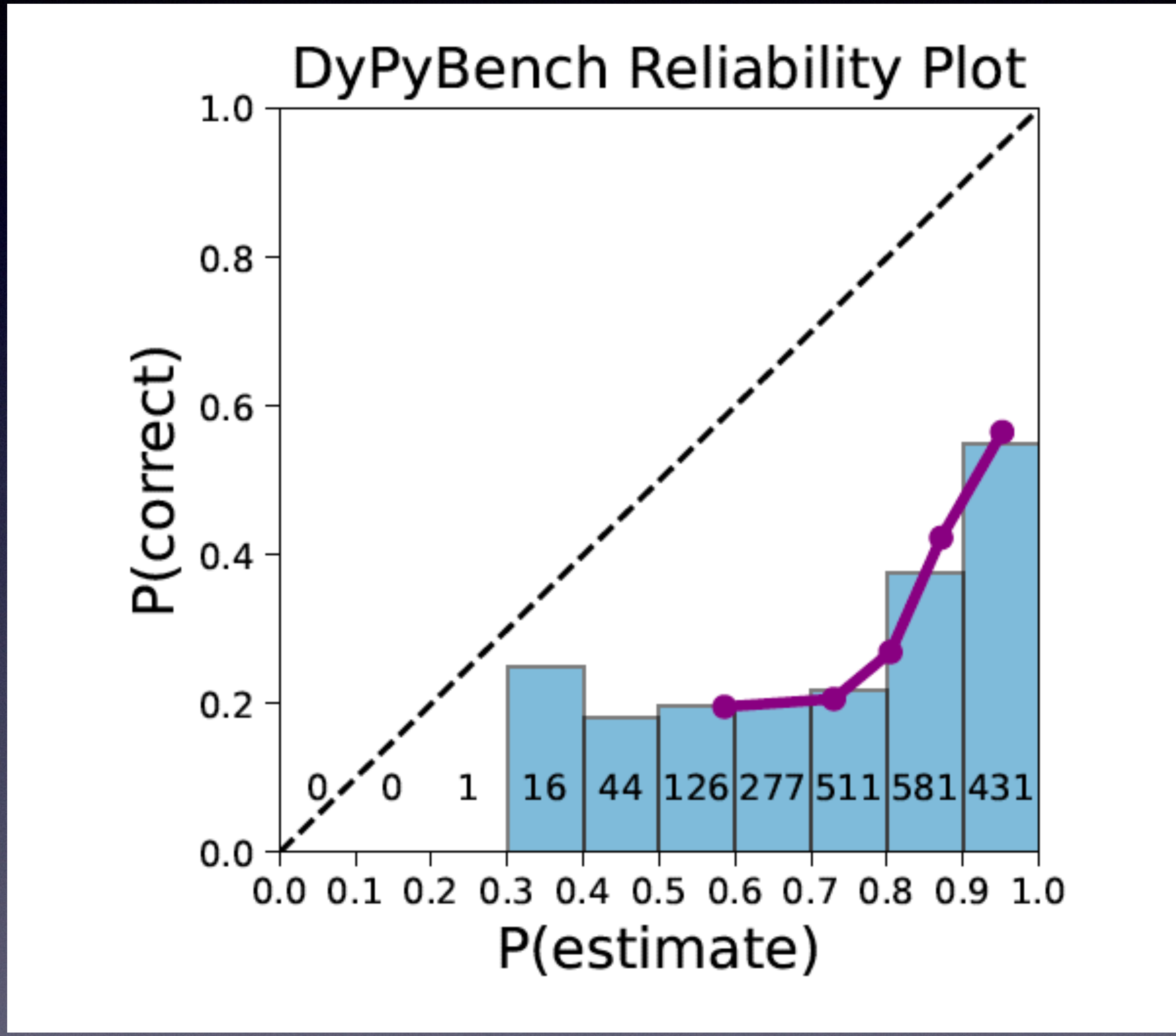
GPT 3.5, Line Completion



ECE = *0.15*
Brier (actual) = *0.41*
Brier (ref) = *0.22*

Correctness: Test Passing, ***Confidence:*** Avg Per-token probability

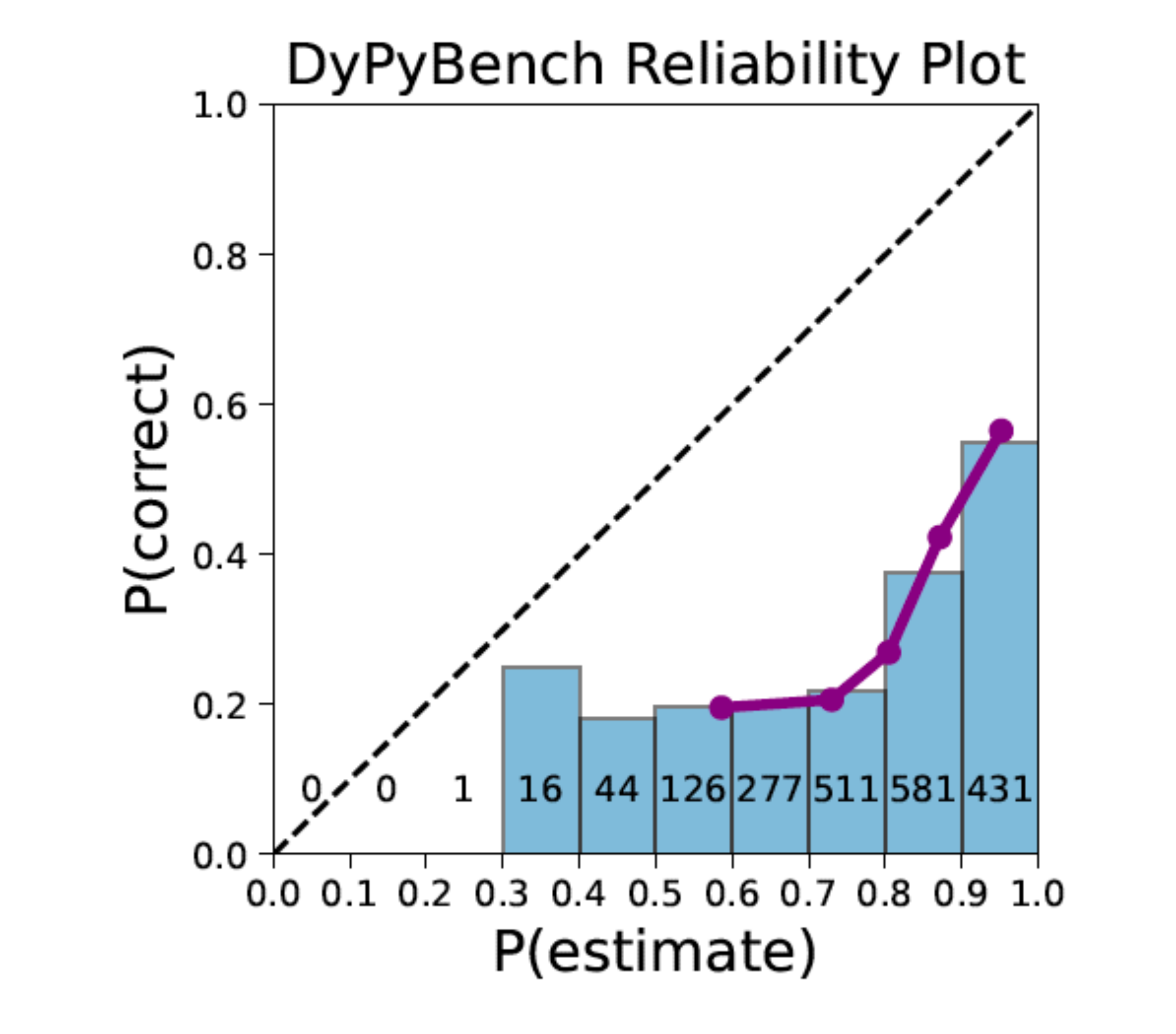
GPT 3.5, Line Completion



ECE = *0.15*
Brier (actual) = *0.41*
Brier (ref) = *0.22*
Skill Score = *-0.87*

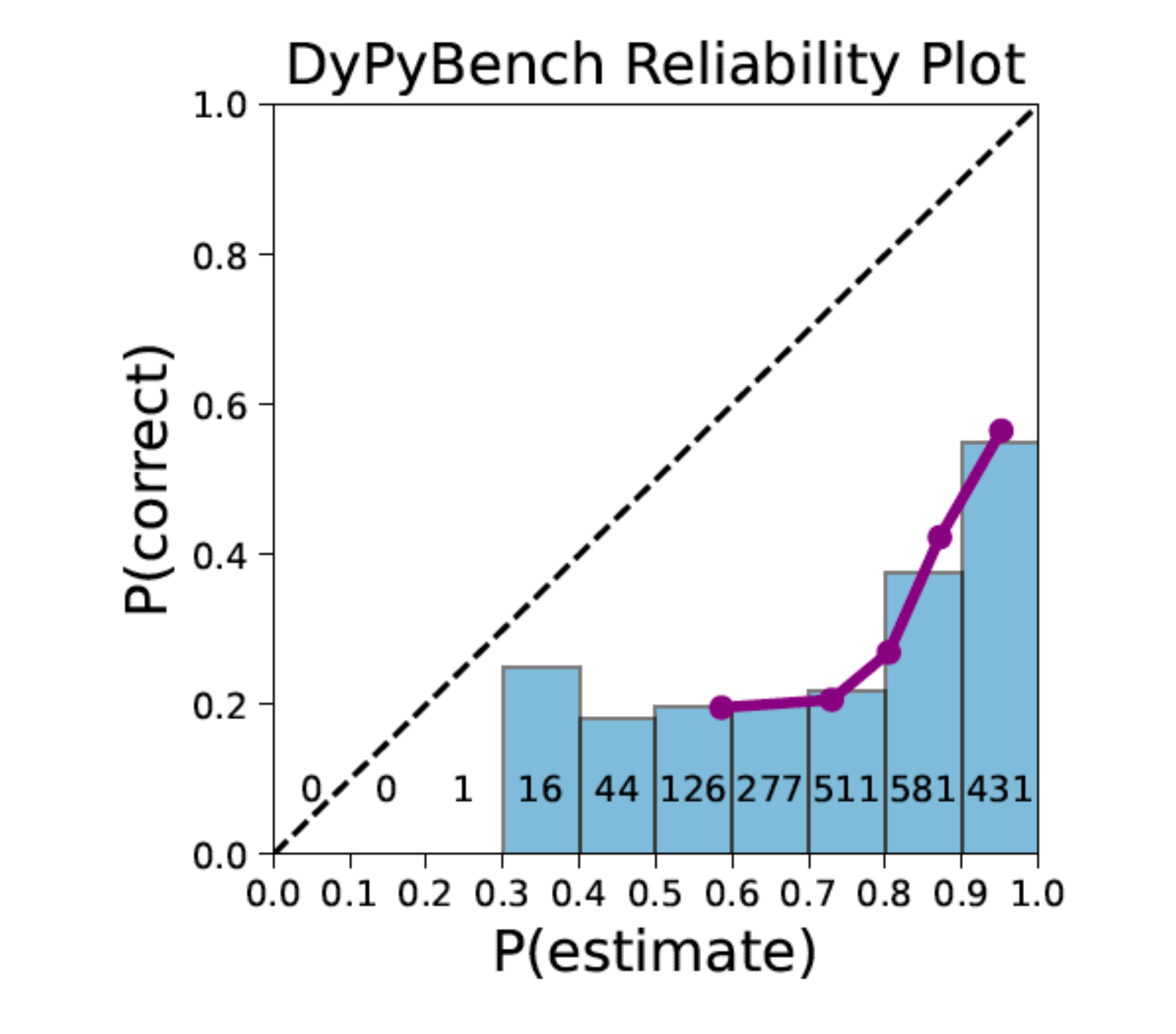
Correctness: Test Passing, ***Confidence:*** Avg Per-token probability

Recaling the Confidence



Recaling the Confidence

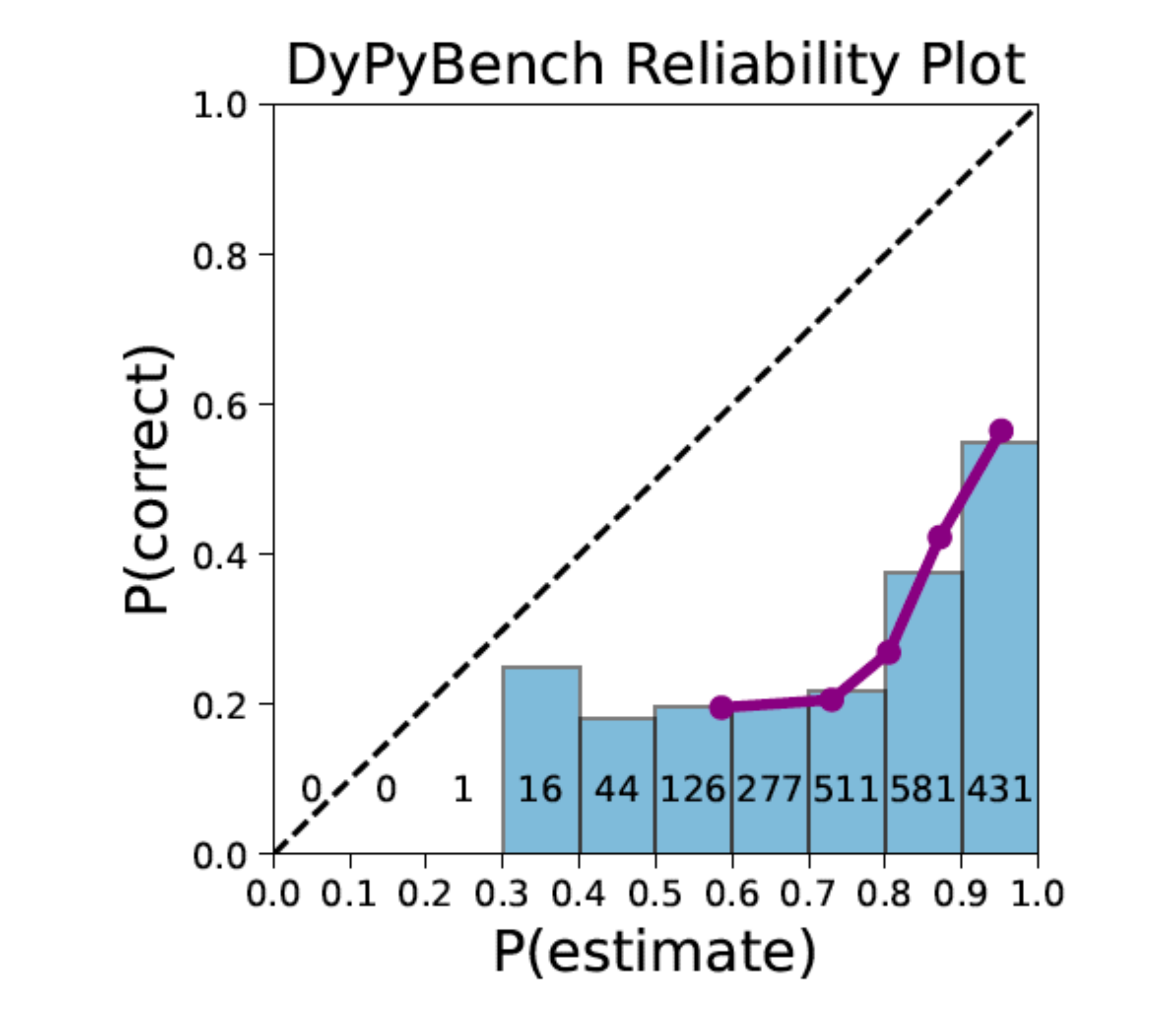
Platt Scaling: fit a logistic regression using ***some*** datapoints to better match the actual correctness response.



Recaling the Confidence

Platt Scaling: fit a logistic regression using ***some*** datapoints to better match the actual correctness response.

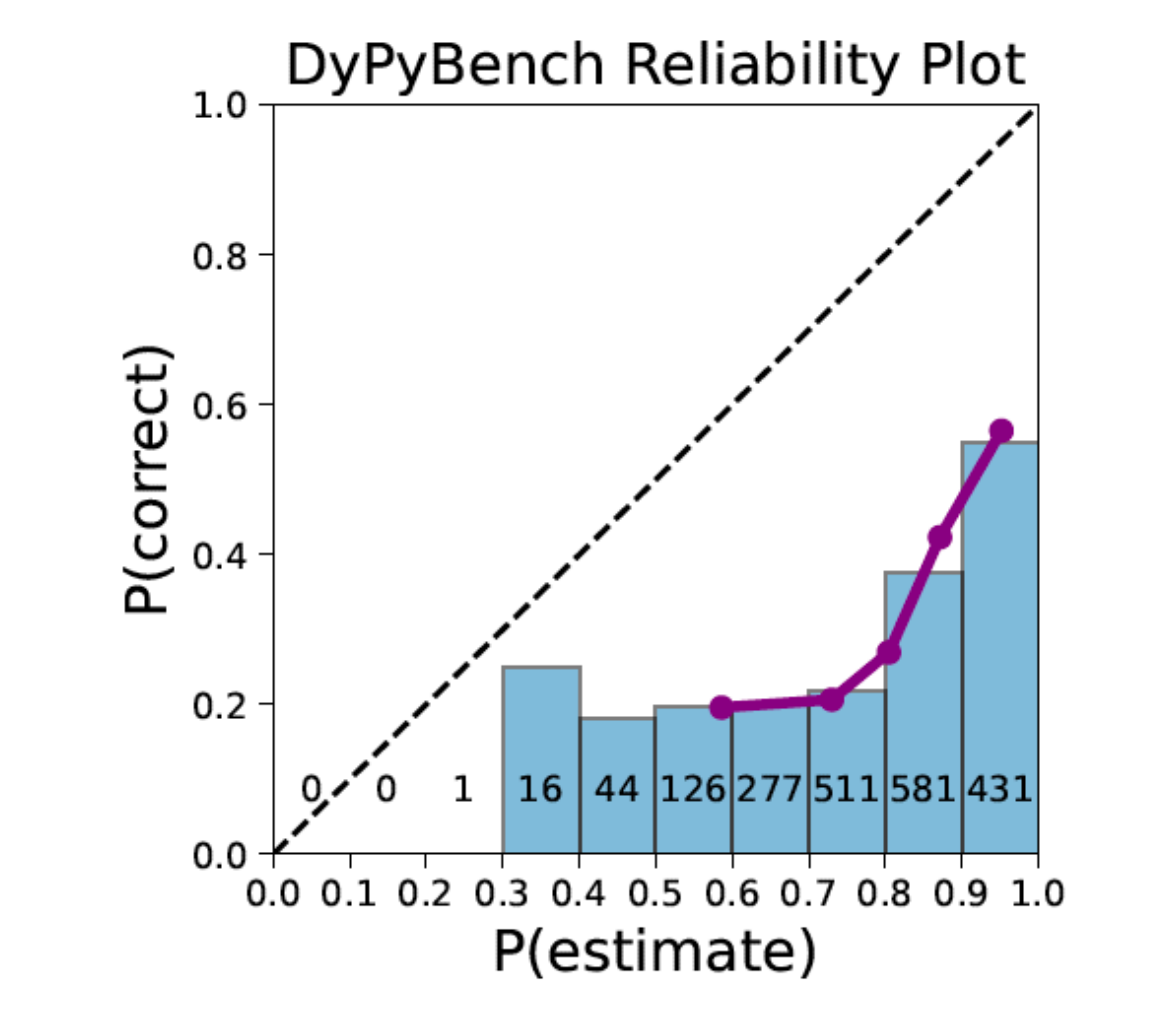
- ***RESPONSE:*** The actual correctness value (y axis)



Recaling the Confidence

Platt Scaling: fit a logistic regression using ***some*** datapoints to better match the actual correctness response.

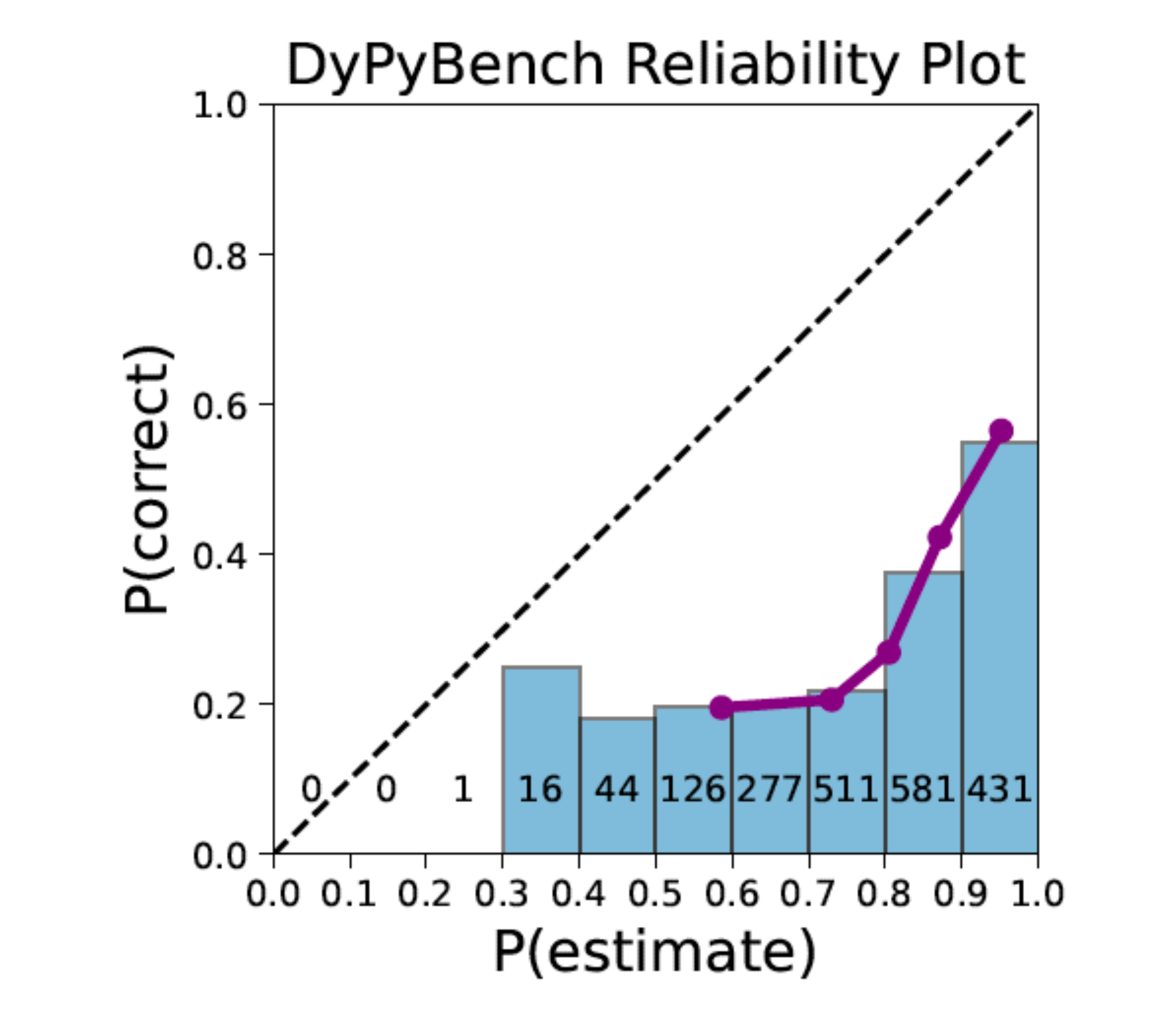
- ***RESPONSE:*** The actual correctness value (y axis)
- ***PREDICTOR:*** The model-output confidence.



Recaling the Confidence

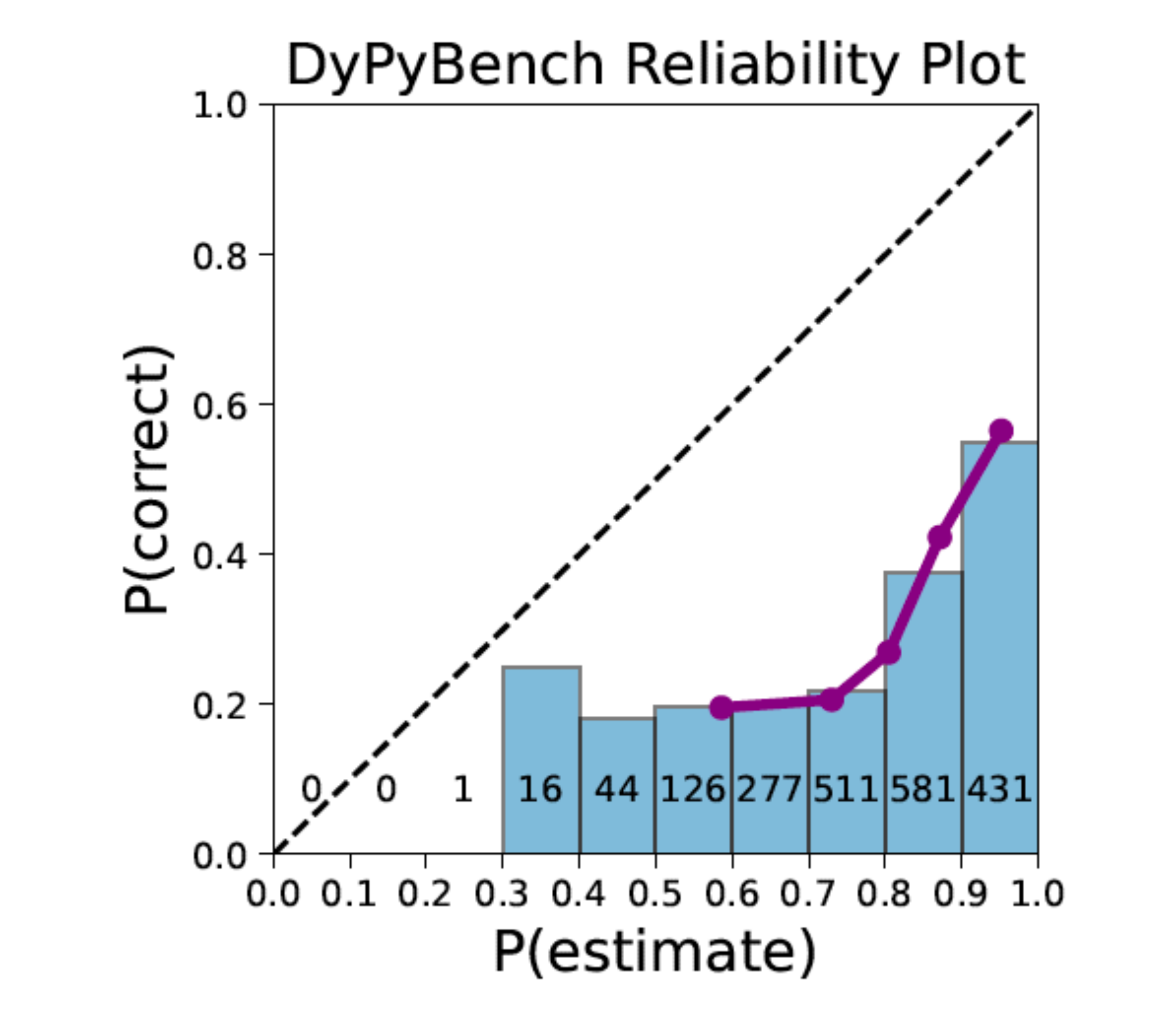
Platt Scaling: fit a logistic regression using ***some*** datapoints to better match the actual correctness response.

- ***RESPONSE:*** The actual correctness value (y axis)
- ***PREDICTOR:*** The model-output confidence.
- ***PARAMETERS:*** Two, scaling value + bias



Recaling the Confidence

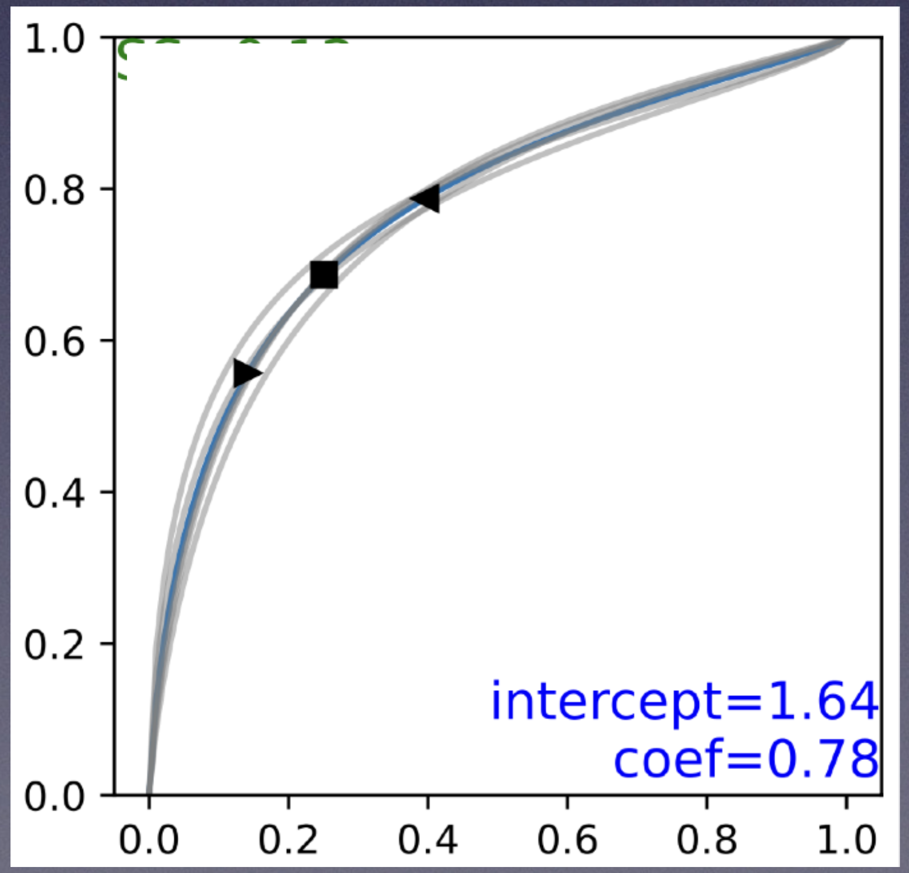
Platt Scaling: fit a logistic regression using ***some*** datapoints to better match the actual correctness response.



- ***RESPONSE:*** The actual correctness value (y axis)

- ***PREDICTOR:*** The model-output confidence.

- ***PARAMETERS:*** Two, scaling value + bias



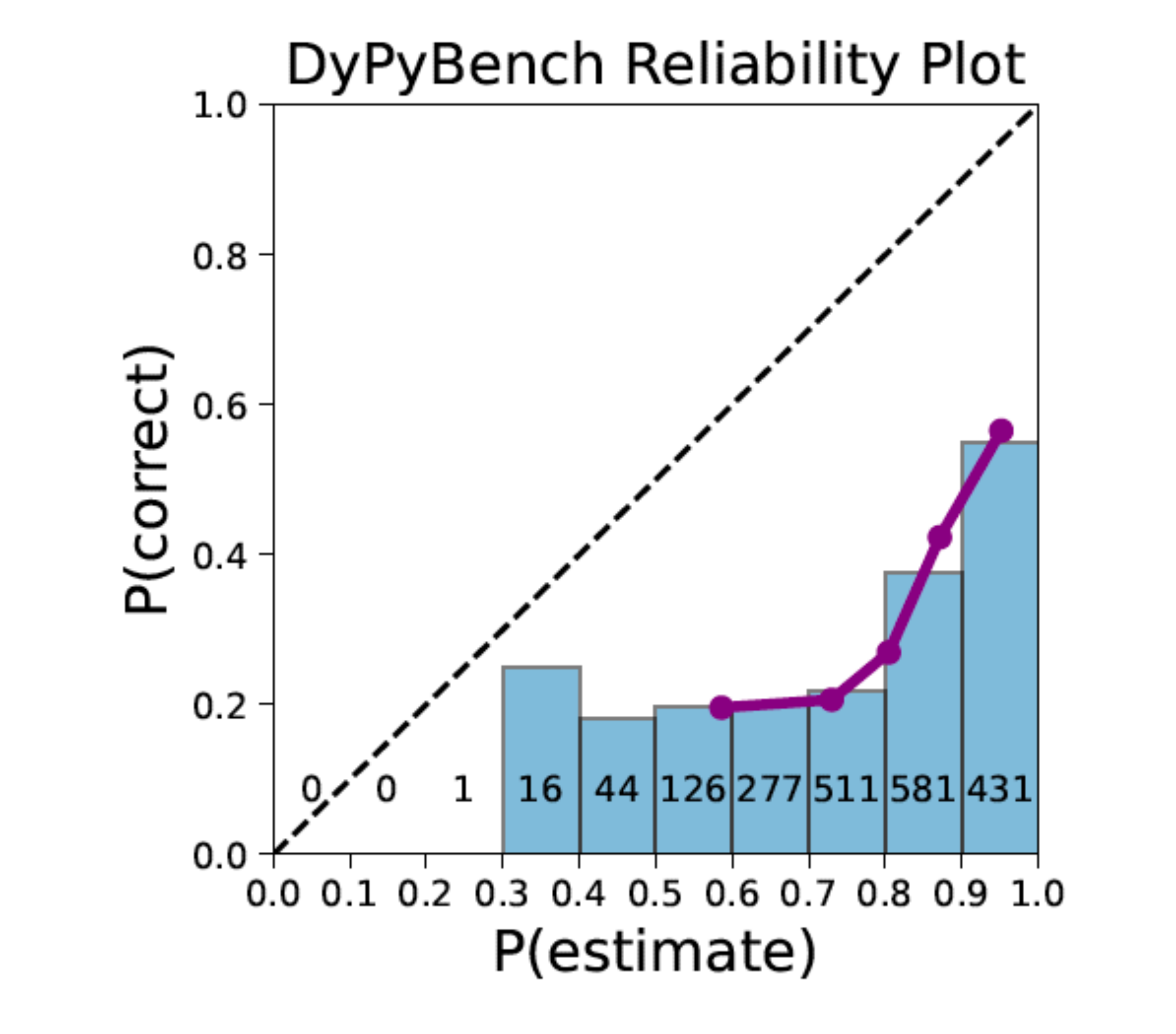
Recaling the Confidence

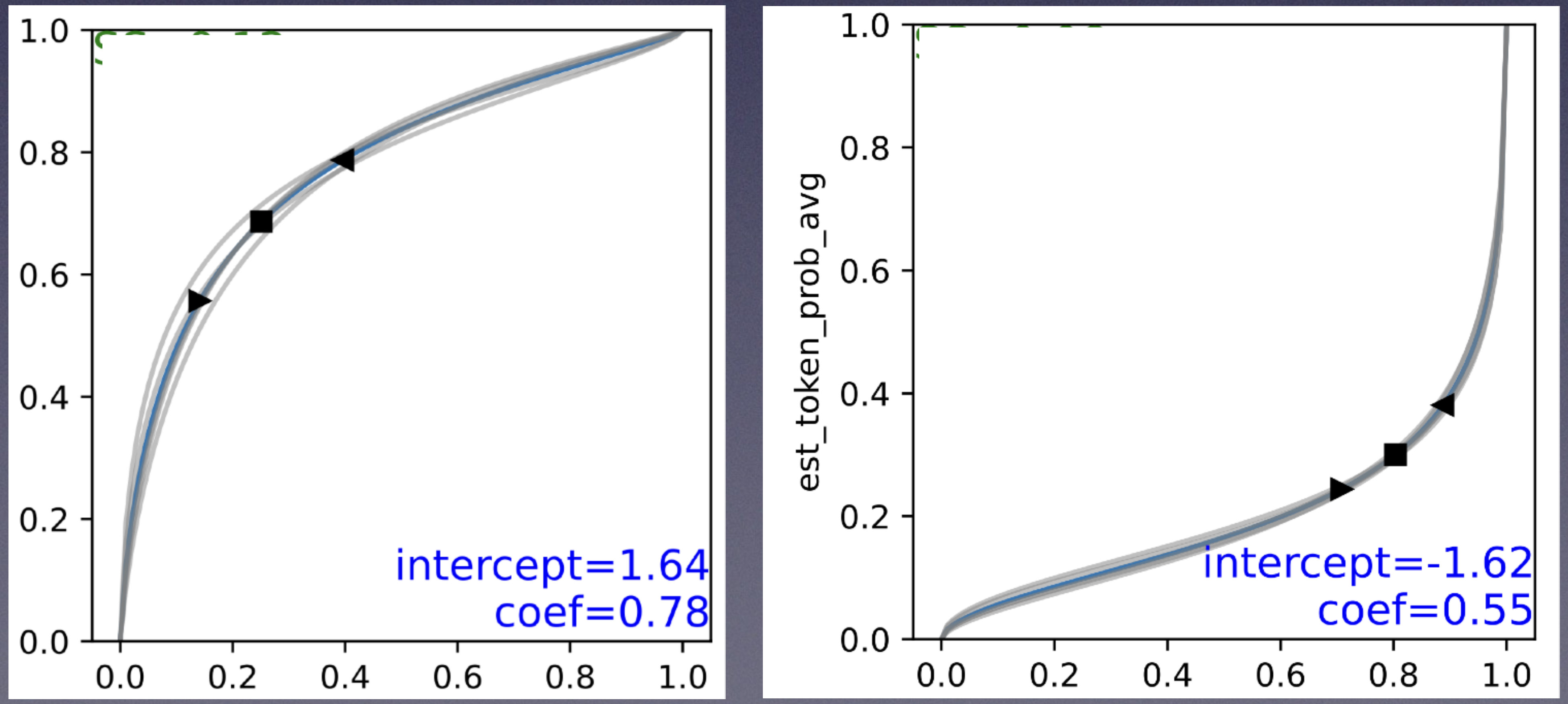
Platt Scaling: fit a logistic regression using ***some*** datapoints to better match the actual correctness response.

- ***RESPONSE:*** The actual correctness value (y axis)

- ***PREDICTOR:*** The model-output confidence.

- ***PARAMETERS:*** Two, scaling value + bias





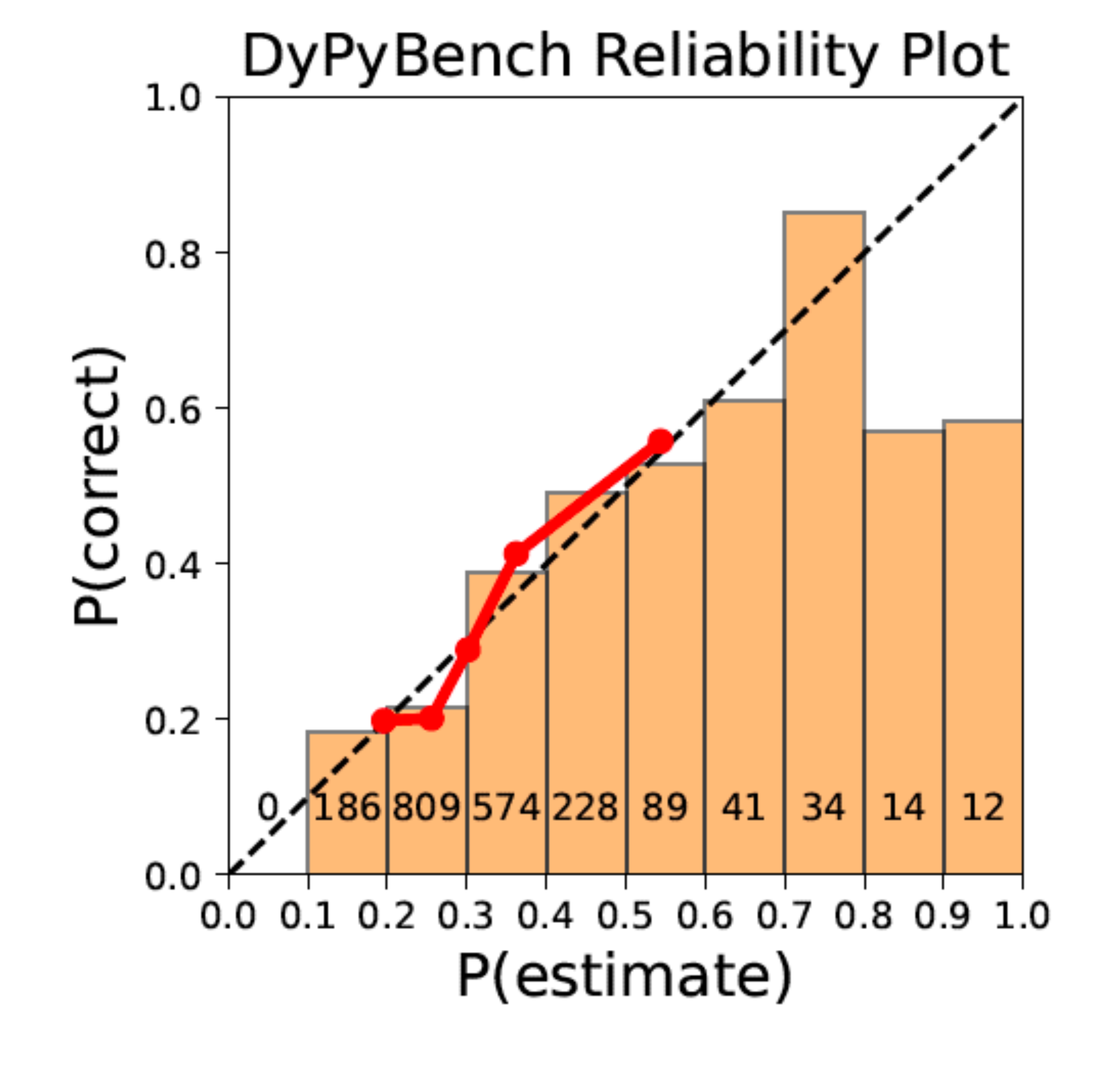
GPT 3.5, Line Completion *(Platt rescaling)*

ECE = *0.04.* *(was 0.15)*

B (actual) = *0.20.* *(was 0.41)*

B (reference) = *0.22*

Skill Score = *+0.08.* *(was -0.87)*



Beyond Intrinsic Confidence

Beyond Intrinsic Confidence

- ***Reflective Verbalized Self Ask:*** Ask the model, given it's own response, to output a probability of correctness.

Beyond Intrinsic Confidence

- ***Reflective Verbalized Self Ask:*** Ask the model, given it's own response, to output a probability of correctness.
- ***Reflective True/False Logit:*** Ask the model if it's own response is correct, output True/False and take the logistic probability of True, normalized with False.

Beyond Intrinsic Confidence

- ***Reflective Verbalized Self Ask:*** Ask the model, given it's own response, to output a probability of correctness.
- ***Reflective True/False Logit:*** Ask the model if it's own response is correct, output True/False and take the logistic probability of True, normalized with False.
- ***Few-shot:*** Same as above, except we provide few-shots (both RAG and random.

Reflective Verbalized Self-Ask

We have the following python code implementing a method.

```
from typing import List

def below_zero(operations: List[int]) -> bool:
   ””” You're given a list of deposit and withdrawal operations on a bank
account that starts with zero balance. Your task is to detect if at any point
the balance of account fallls below zero, and at that point function should
return True. Otherwise it should return False.
>>> below_zero([1, 2, 3])
False
>>> below_zero([1, 2, -4, 5])
True
   “””
   balance = 0
   for op in operations:
      balance += op
   if balance < 0:
      return True
   return False
```

What is a well-calibrated percent probability that this code passes the test cases?
Probability:

Reflective Verbalized Self-Ask

We have the following python code implementing a method.

```
from typing import List

def below_zero(operations: List[int]) -> bool:
   """ You're given a list of deposit and withdrawal operations on a bank
account that starts with zero balance. Your task is to detect if at any point
the balance of account fallls below zero, and at that point function should
return True. Otherwise it should return False.
>>> below_zero([1, 2, 3])
False
>>> below_zero([1, 2, -4, 5])
True
   """
   balance = 0
   for op in operations:
      balance += op
   if balance < 0:
      return True
   return False
```

What is a well-calibrated percent probability that this code passes the test cases?
Probability:

Reflective Logit True/False Logic

We have the following python code implementing a method.

```
from typing import List

def below_zero(operations: List[int]) -> bool:
   """ You're given a list of deposit and withdrawal operations on a bank
account that starts with zero balance. Your task is to detect if at any point
the balance of account fallls below zero, and at that point function should
return True. Otherwise it should return False.
>>> below_zero([1, 2, 3])
False
>>> below_zero([1, 2, -4, 5])
True
   """
   balance = 0
   for op in operations:
      balance += op
   if balance < 0:
      return True
   return False
```

True or False, this code matches intent and is bug-free. Answer: **True**

Reflective Logit True/False Logic

We have the following python code implementing a method.

```
from typing import List

def below_zero(operations: List[int]) -> bool:
    """ You're given a list of deposit and withdrawal operations on a bank
account that starts with zero balance. Your task is to detect if at any point
the balance of account fallls below zero, and at that point function should
return True. Otherwise it should return False.
>>> below_zero([1, 2, 3])
False
>>> below_zero([1, 2, -4, 5])
True
    """
    balance = 0
    for op in operations:
        balance += op
    if balance < 0:
        return True
```

True or False, this code matches intent and is bug-free. Answer: True

Reflective Logit True/False Logic

We have the following python code implementing a method.

```
from typing import List

def below_zero(operations: List[int]) -> bool:
    """ You're given a list of deposit and withdrawal operations on a bank
account that starts with zero balance. Your task is to detect if at any point
the balance of account fallls below zero, and at that point function should
return True. Otherwise it should return False.
>>> below_zero([1, 2, 3])
False
>>> below_zero([1, 2, -4, 5])
True
    """
    balance = 0
    for op in operations:
        balance += op
    if balance < 0:
        return True
```

True or False, this code matches intent and is bug-free. Answer: True

Logit

TL;DR Findings

TL;DR Findings

- ***<u>Intrinsic:</u>*** Somewhat decent, after rescaling.

TL;DR Findings

- ***Intrinsic:*** Somewhat decent, after rescaling.
- ***Verbalized Self Ask:*** Not much better.

TL;DR Findings

- ***Intrinsic:*** Somewhat decent, after rescaling.
- ***Verbalized Self Ask:*** Not much better.
- ***True/False Logit:*** Not much better.

TL;DR Findings

- ***Intrinsic:*** Somewhat decent, after rescaling.
- ***Verbalized Self Ask:*** Not much better.
- ***True/False Logit:*** Not much better.
- ***Few-shot:*** RAG with BM25 *(next slide)*

Few-shot Reflective, for Code Completion

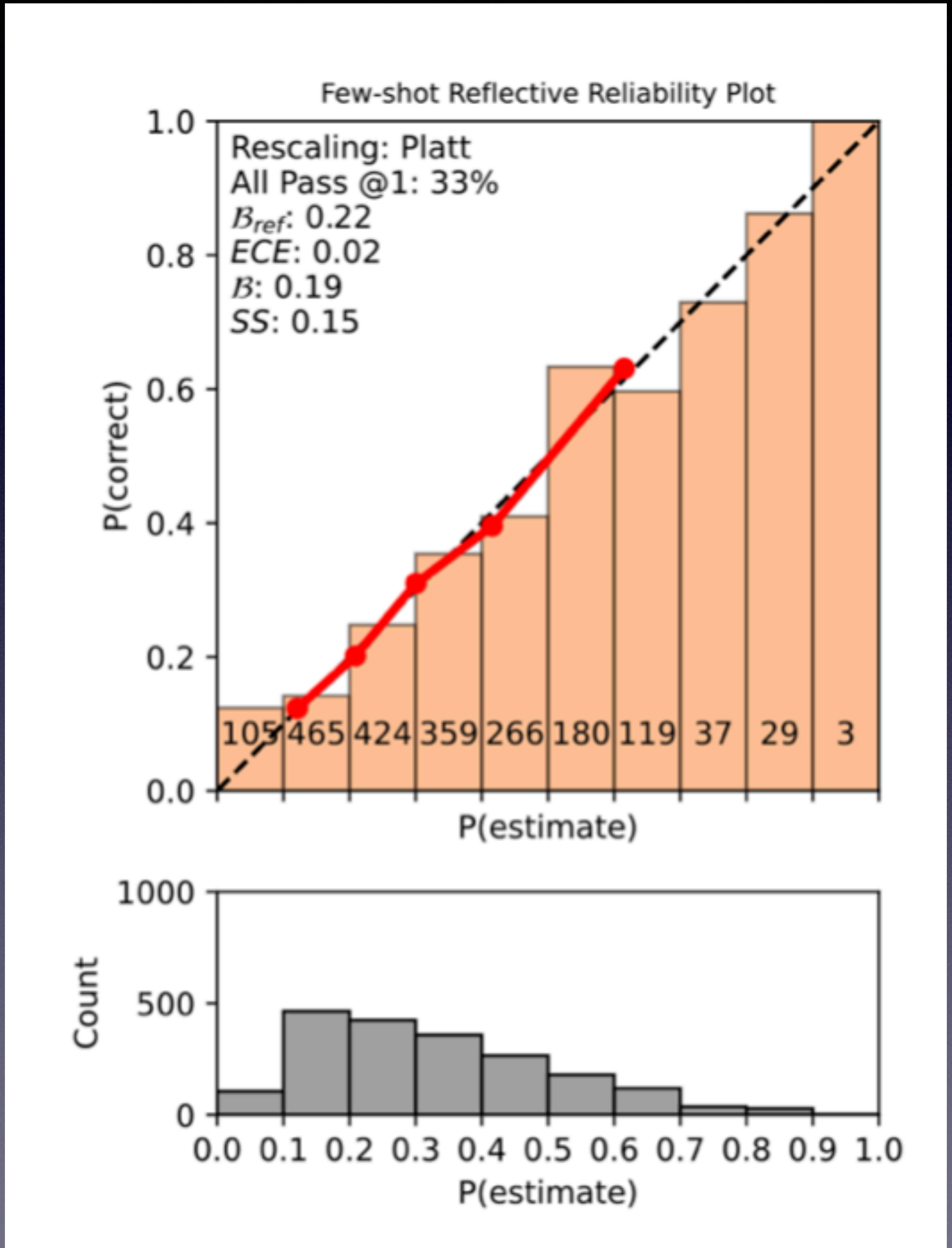
Confidence Measure	$\mathcal{B} \downarrow$	$SS \uparrow$	$ECE \downarrow$
0-Shot Reflect	0.25	-0.15	0.15
0-Shot Reflect (Rescaled)	0.22	0.00	
FS Random	0.29	-0.29	0.21
FS Random (Rescaled)	0.22	0.0	
FS BM25	0.20	0.08	0.10
FS BM25 (Rescaled)	0.19	0.15	0.02

Few-shot Reflective, for Code Completion

Confidence Measure	$\mathcal{B}\downarrow$	$SS\uparrow$	$ECE\downarrow$
0-Shot Reflect	0.25	-0.15	0.15
0-Shot Reflect (Rescaled)	0.22	0.00	
FS Random	0.29	-0.29	0.21
FS Random (Rescaled)	0.22	0.0	
FS BM25	0.20	0.08	0.10
FS BM25 (Rescaled)	0.19	0.15	0.02

Few-shot Reflective, for Code Completion

Confidence Measure	$\mathcal{B} \downarrow$	$SS \uparrow$	$ECE \downarrow$
0-Shot Reflect	0.25	-0.15	0.15
0-Shot Reflect (Rescaled)	0.22	0.00	
FS Random	0.29	-0.29	0.21
FS Random (Rescaled)	0.22	0.0	
FS BM25	0.20	0.08	0.10
FS BM25 (Rescaled)	0.19	0.15	0.02



Future: How to Present?

Future: How to Present?

```
def Convert2Base7(num):
  if num < 0:
    return '-' + Convert2Base7(-num)
  elif num < 7:
    return str(num)
  else:
    return Convert2Base7(num // 7) + str(num % 7)
```

Future: How to Present?

```
def Convert2Base7(num):
  if num < 0:
    return '-' + Convert2Base7(-num)
  elif num < 7:
    return str(num)
  else:
    return Convert2Base7(num // 7) + str(num % 7)
```

```
def Convert2Base7(num):
  if num < 0:
    return '-' + Convert2Base7(-num)
  elif num < 7:
    return str(num)
  else:
    return Convert2Base7(num // 7) + str(num % 7)
```

Fine-grained calibration?

Work underway, please stay tuned.

Fine-grained calibration?

```
def Convert2Base7(num):
  if num < 0:
    return '-' + Convert2Base7(-num)
  elif num < 7:
    return str(num)
  else:
    return Convert2Base7(num // 7) + str(num % 7)
```

Work underway, please stay tuned.

Summary

- The “Last Mile” is challenging
- Hypothesis: More information would lead to rational decision making and better outcomes.
- Well Calibrated Confidences are possible
- Need to do some User Studies. (*Collaborators ?)*