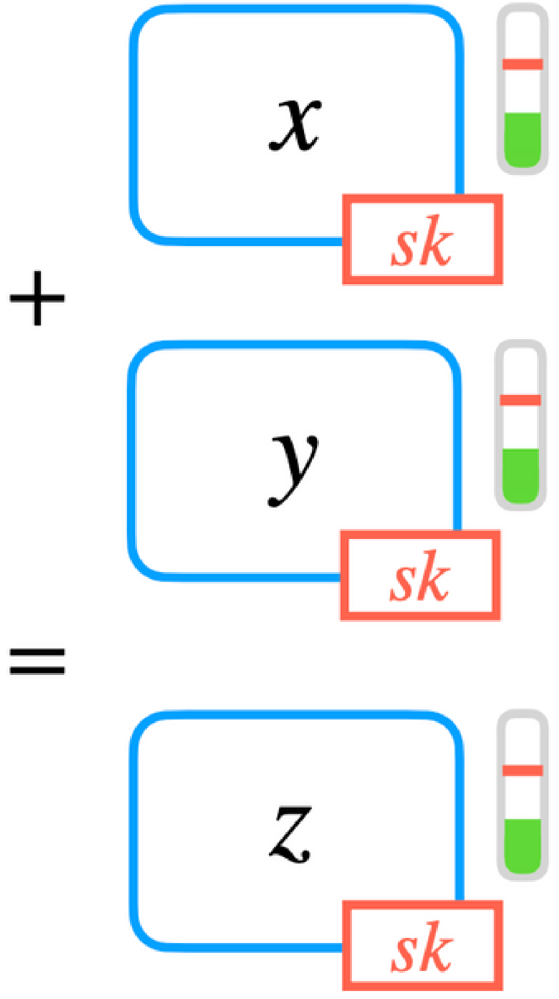
benoit.chevalliermames@zama.ai | August 2023

Privacy-Preserving Tree-Based Inference with Fully Homomorphic Encryption

J. Frery, A. Stoian, R. Bredehoft, L. Montero, C. Kherfallah, B. Chevallier-Mames* and A. Meyre

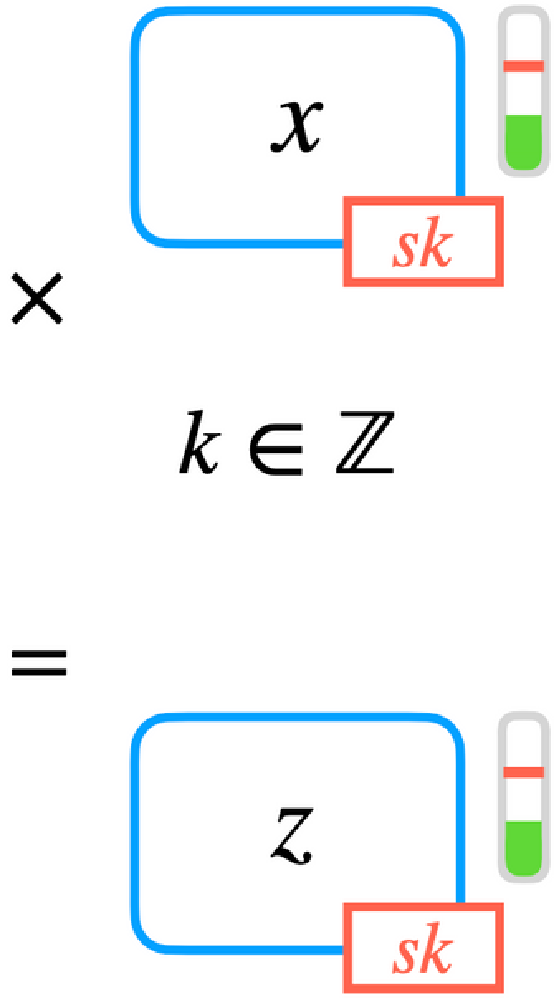
Torus Fully Homomorphic Encryption

addition



$$z \approx x + y$$

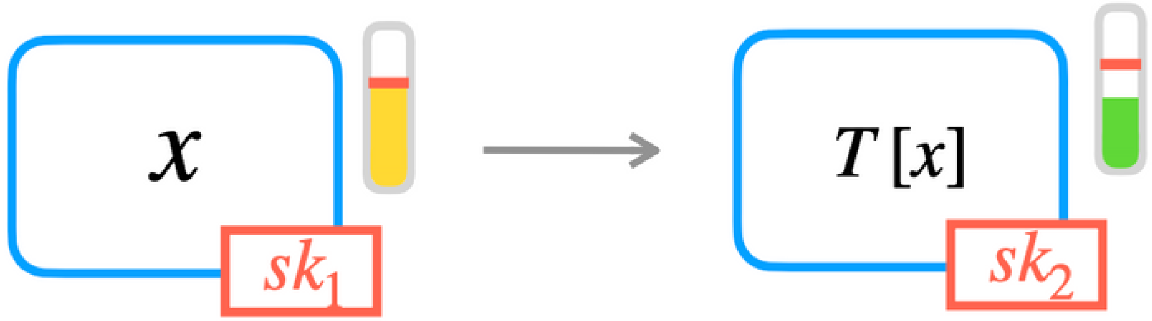
multiplication by constant



$$z \approx k \cdot x$$

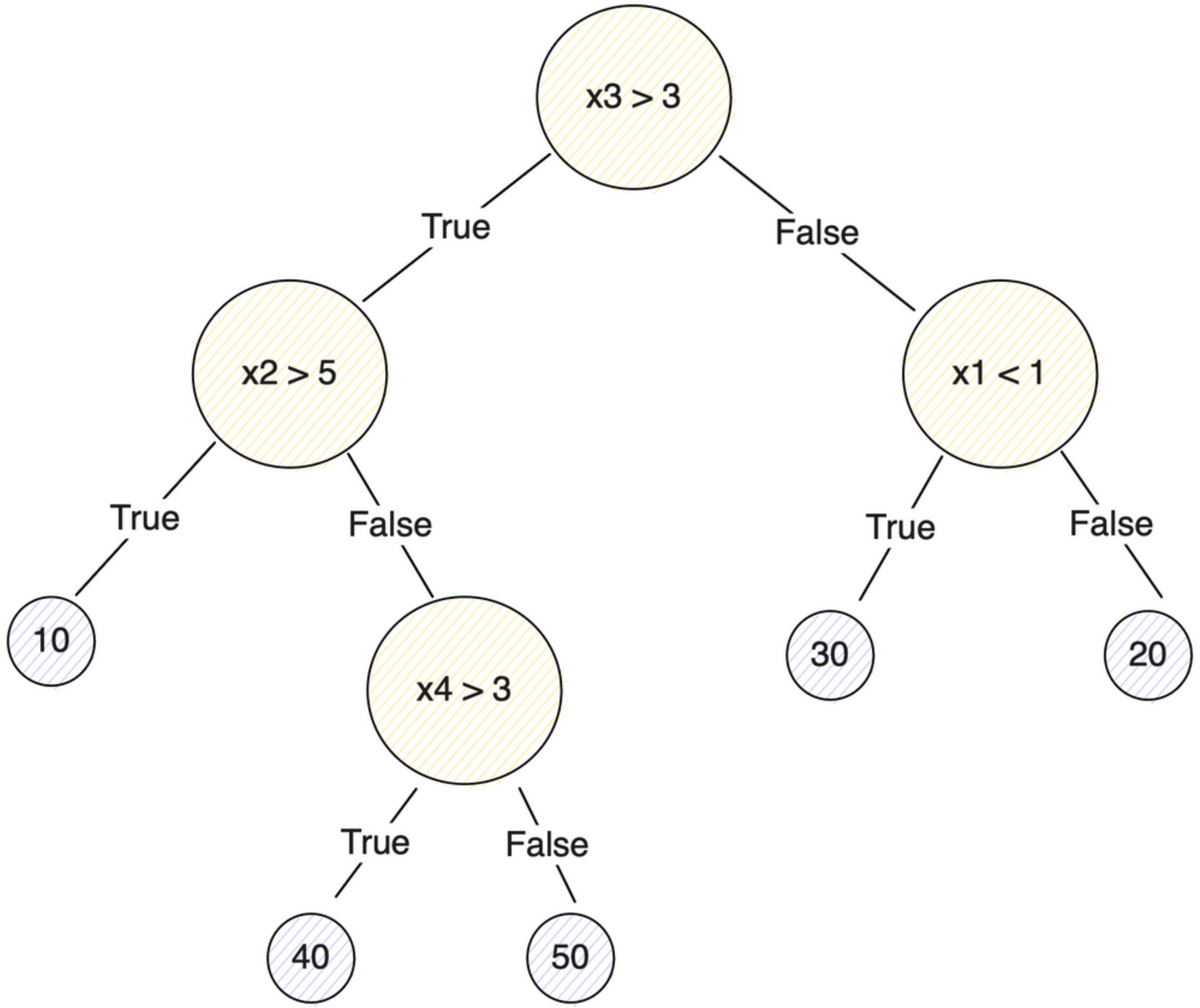
table look-up

PBS



Doing Trees in FHE?

- Powerful and highly used ML models (eg, DecisionTree, RandomForest, XGBoost)
- Depends on conditions and branches, which are not doable in FHE



What We Describe in the Paper

A METHOD FOR TREES IN FHE

- Only matrix operations and two layer of PBS
- Adapt to any tree depths or number of trees
- Pretty efficient and versatile

AN IMPLEMENTATION

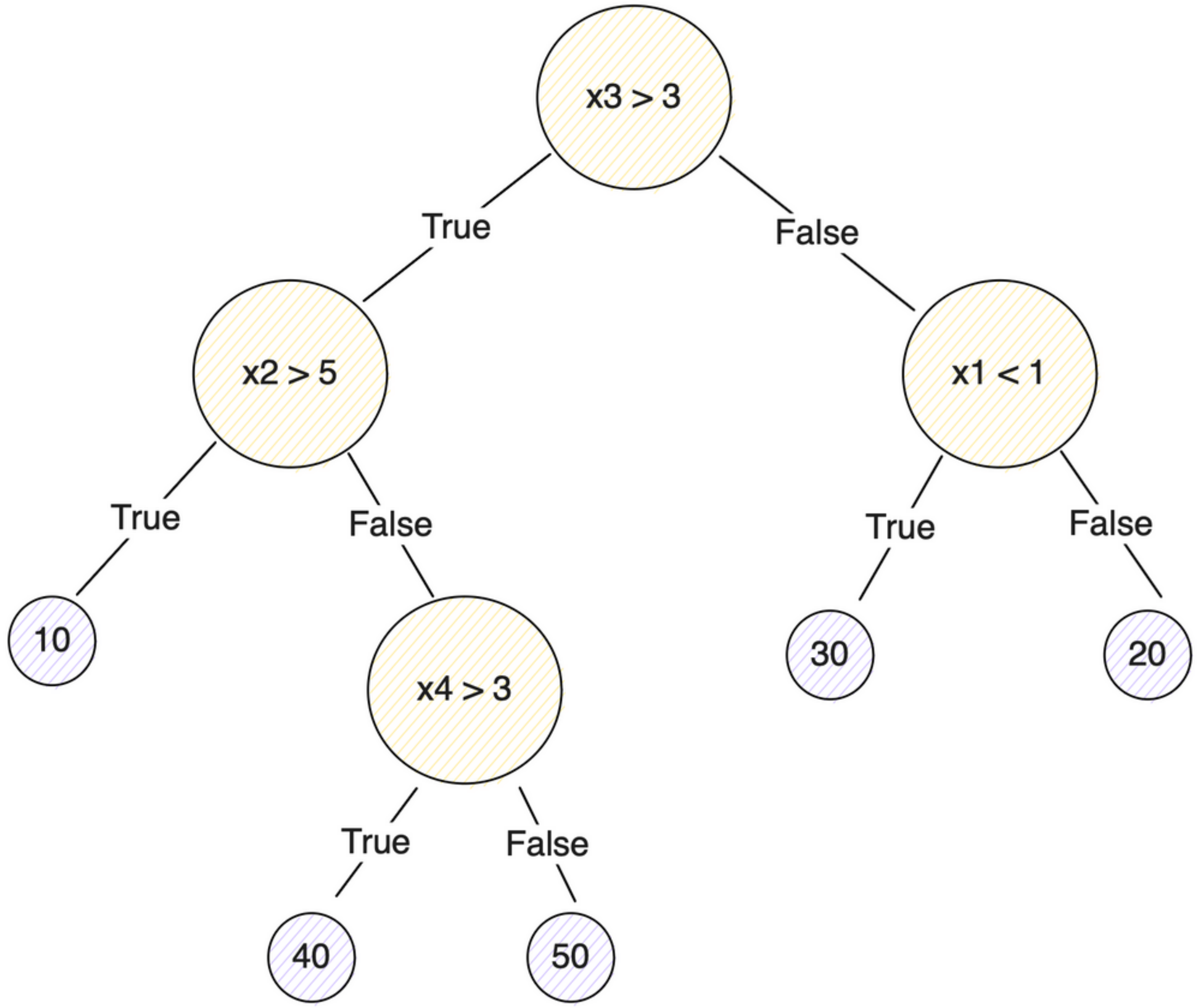
- Completely integrated in Concrete ML
- Open source
- Easy to use without crypto knowledge

EXPERIMENTAL RESULTS

- Experimental results over several datasets
- Available live demos on Hugging Face

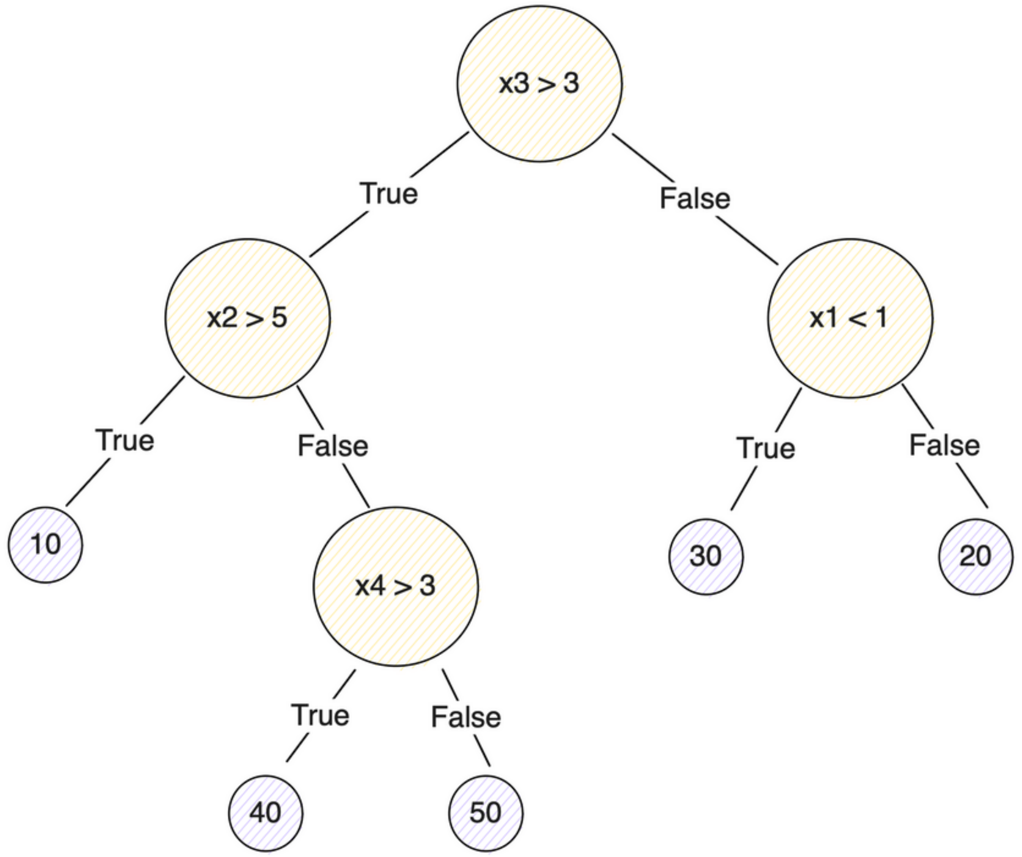
The Method

- Using **HummingBird**
- Use TLU for x_i > c: eg, for x > 2, use T[i] = [0, 0, 0, 1, 1, ..., 1]
- Have a **first layer of PBS** to compute the conditions
- Have a **second layer of PBS** to have a one-hot vector of which branch is taken. Eg, 10 will correspond to [1, 0, 0, 0, 0] while 50 will correspond to [0, 0, 1, 0, 0]
- Have a final matrix multiplication to associate the one-hot with the final tree value

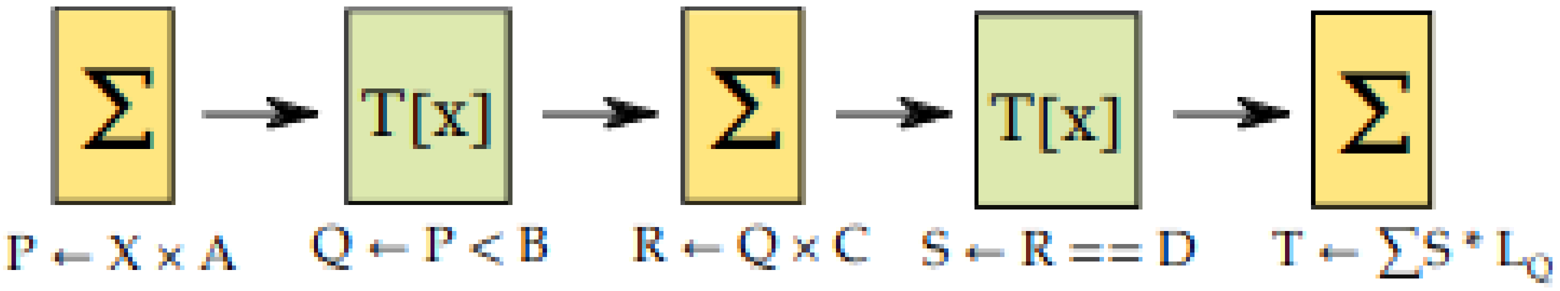


The Method - How To Have the One-Hot

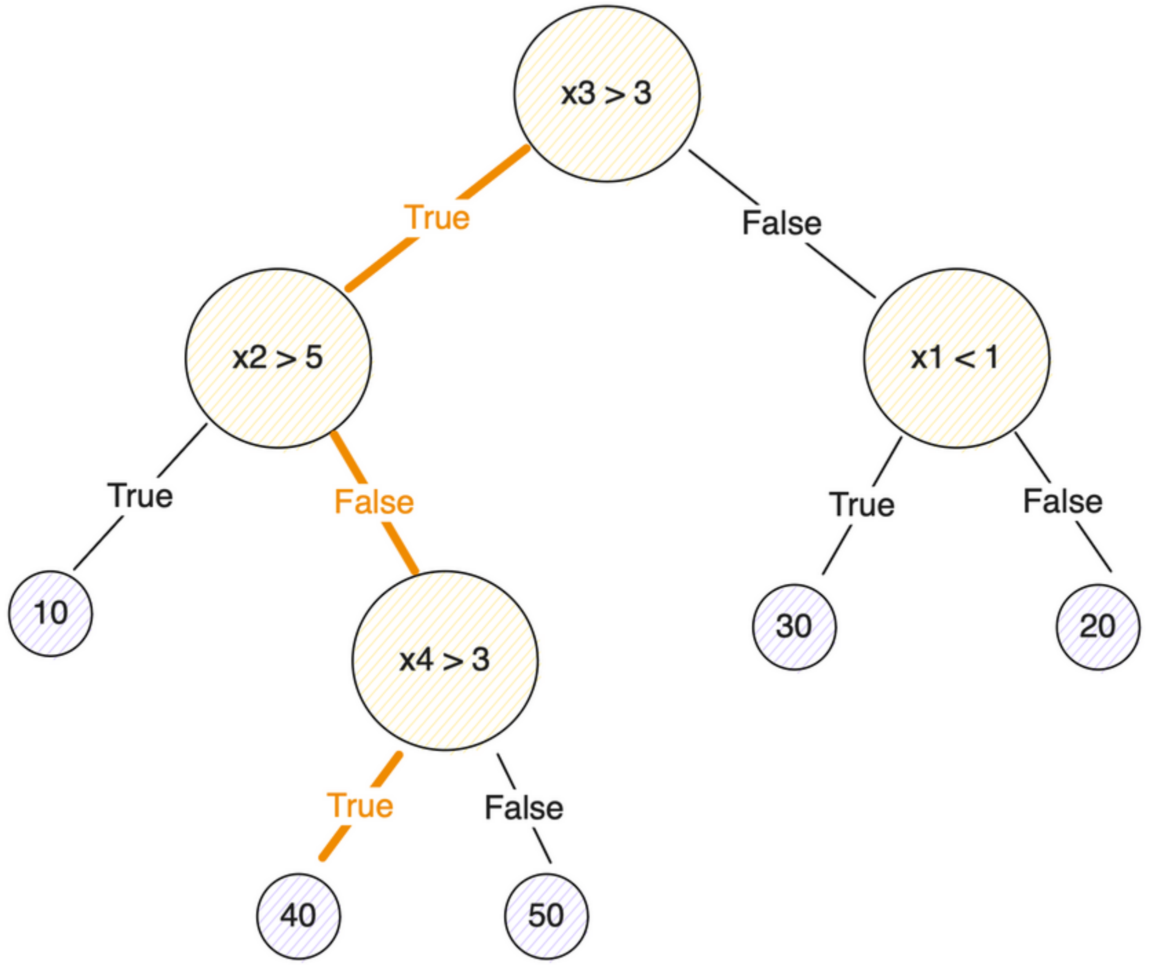
- Eg, the 30 output, which has the one-hot encoding [0, 0, 0, 1, 0] appears iff:
    - c1 := x3 > 3 is False
    - c3 := x1 < 1 is True
    - whatever c2 and c4
- So, we say that the 4-th bit of the one-hot is: **c3 -c1 == 1**
- So, one linear layer to accumulate conditions, and one PBS to find if it's the "max" value



The Method



$P \leftarrow X \times A$ $Q \leftarrow P < B$ $R \leftarrow Q \times C$ $S \leftarrow R == D$ $T \leftarrow \sum S * L_Q$



A

0	0	1	0
0	1	0	0
1	0	0	0
0	0	0	1
0	0	0	0

B

3	5	1	3

C

1	1	1	-1	-1
1	-1	-1	0	0
0	0	0	1	-1
0	1	-1	0	0

D

2	2	1	1	0

E

10
40
50
30
20

1	2	7	5	4

P = Input * A

7	2	1	5

Q = Conditions

1	0	0	1

R = Conditions * C

1	2	0	-1	-1

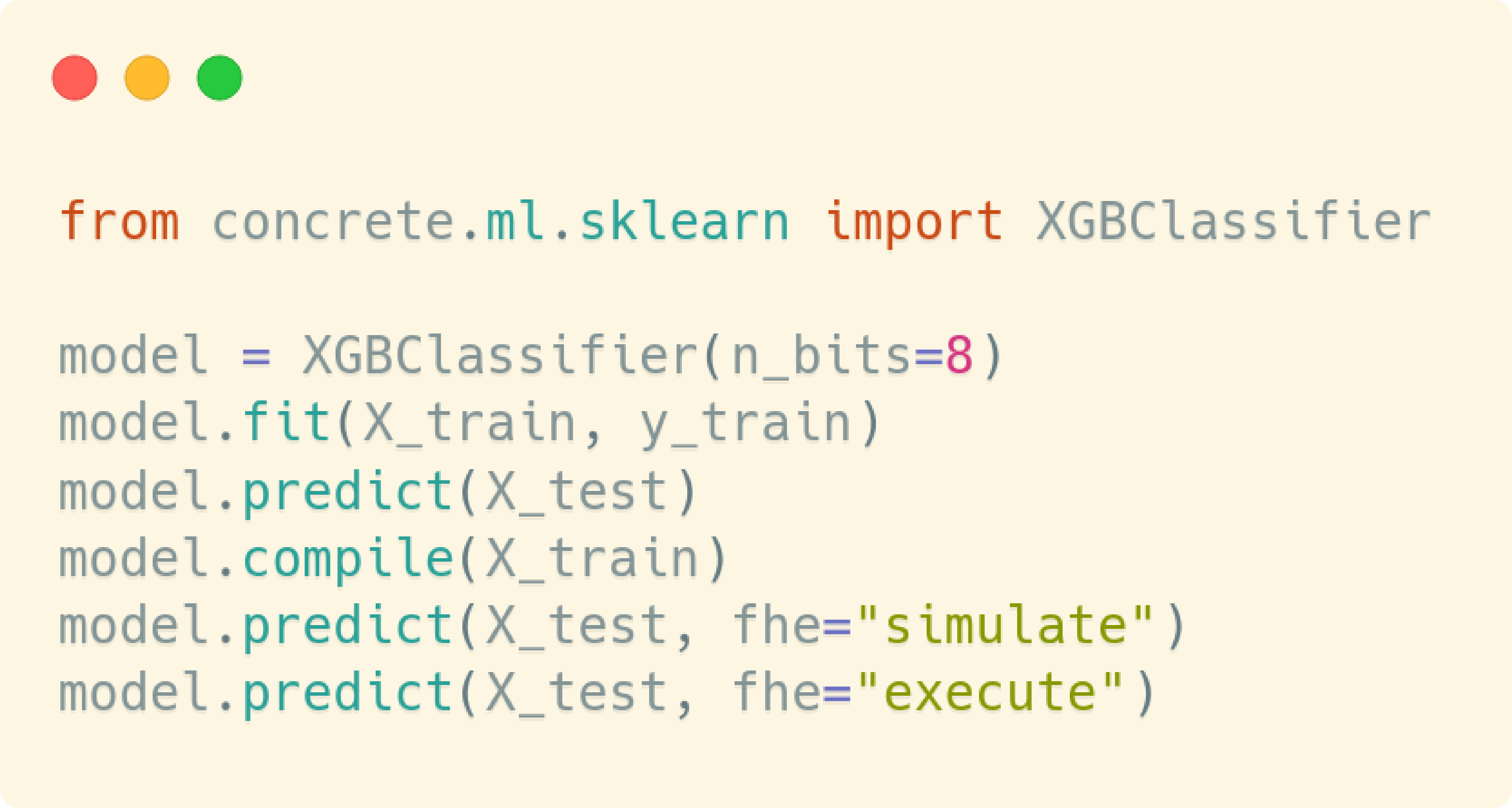
S = One-hot branch

0	1	0	0	0

T = Result

40

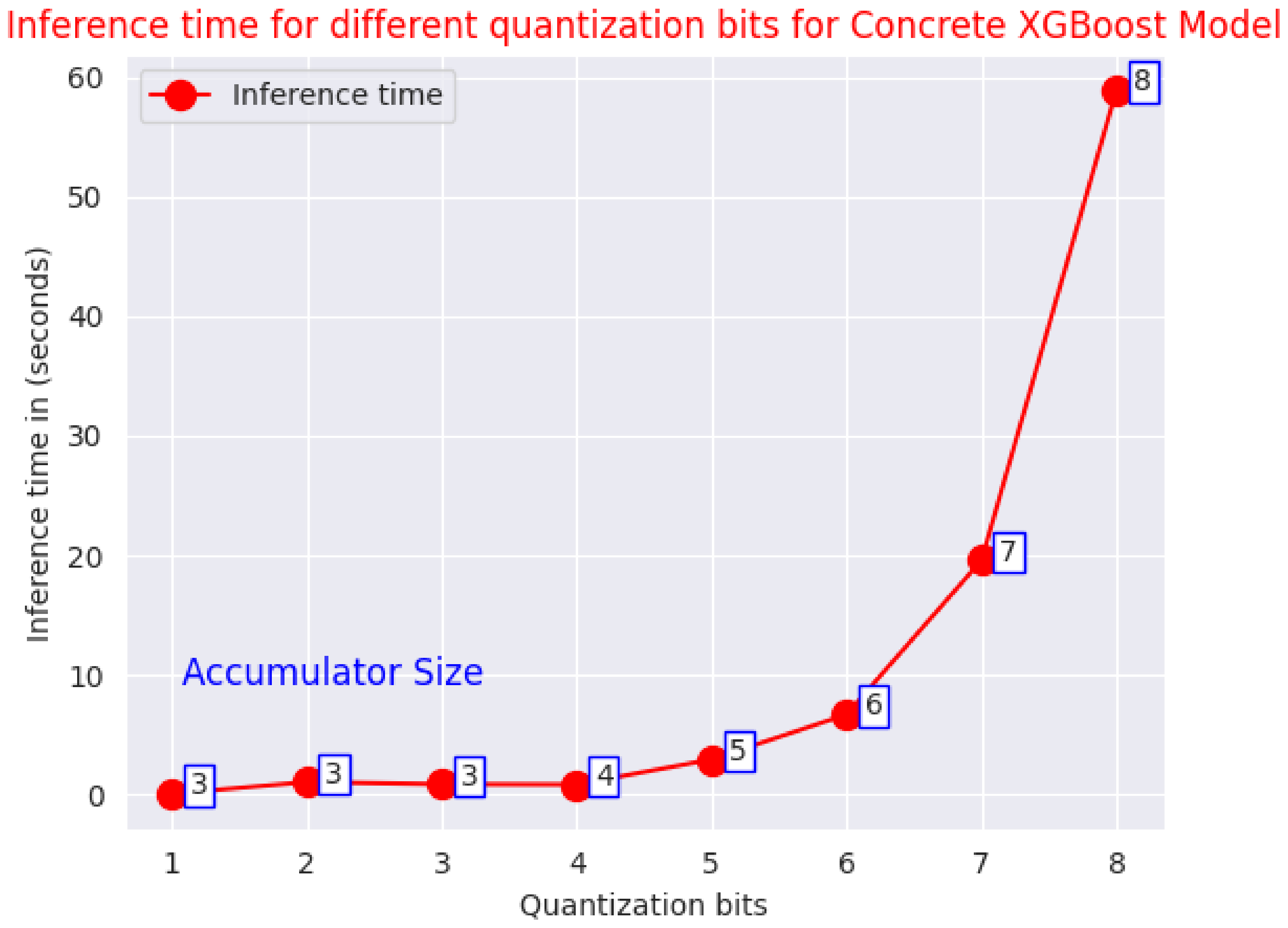
Implementation

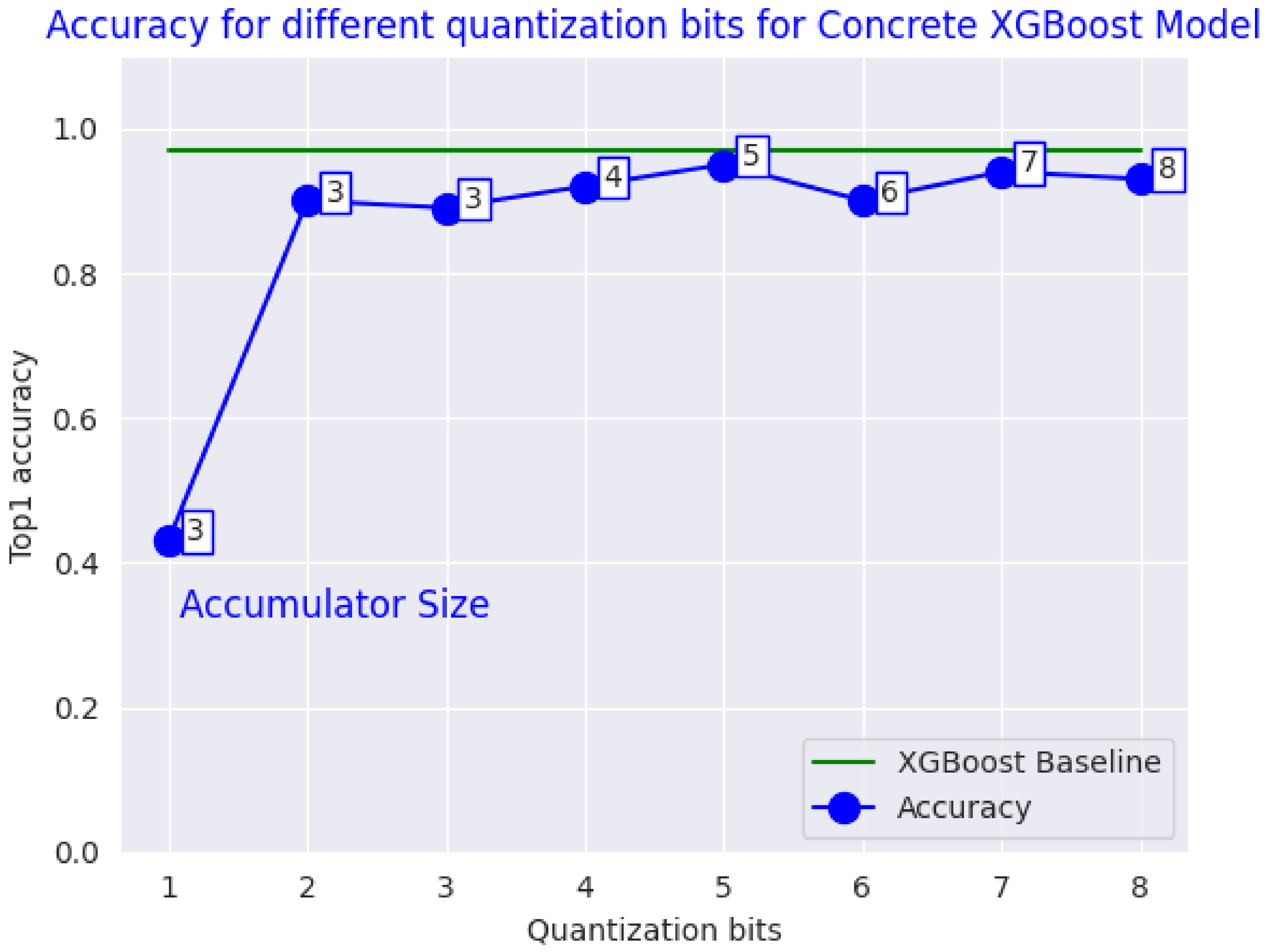


```
from concrete.ml.sklearn import XGBClassifier

model = XGBClassifier(n_bits=8)
model.fit(X_train, y_train)
model.predict(X_test)
model.compile(X_train)
model.predict(X_test, fhe="simulate")
model.predict(X_test, fhe="execute")
```

Experimental Results


Inference time for different quantization bits for Concrete XGBoost Model

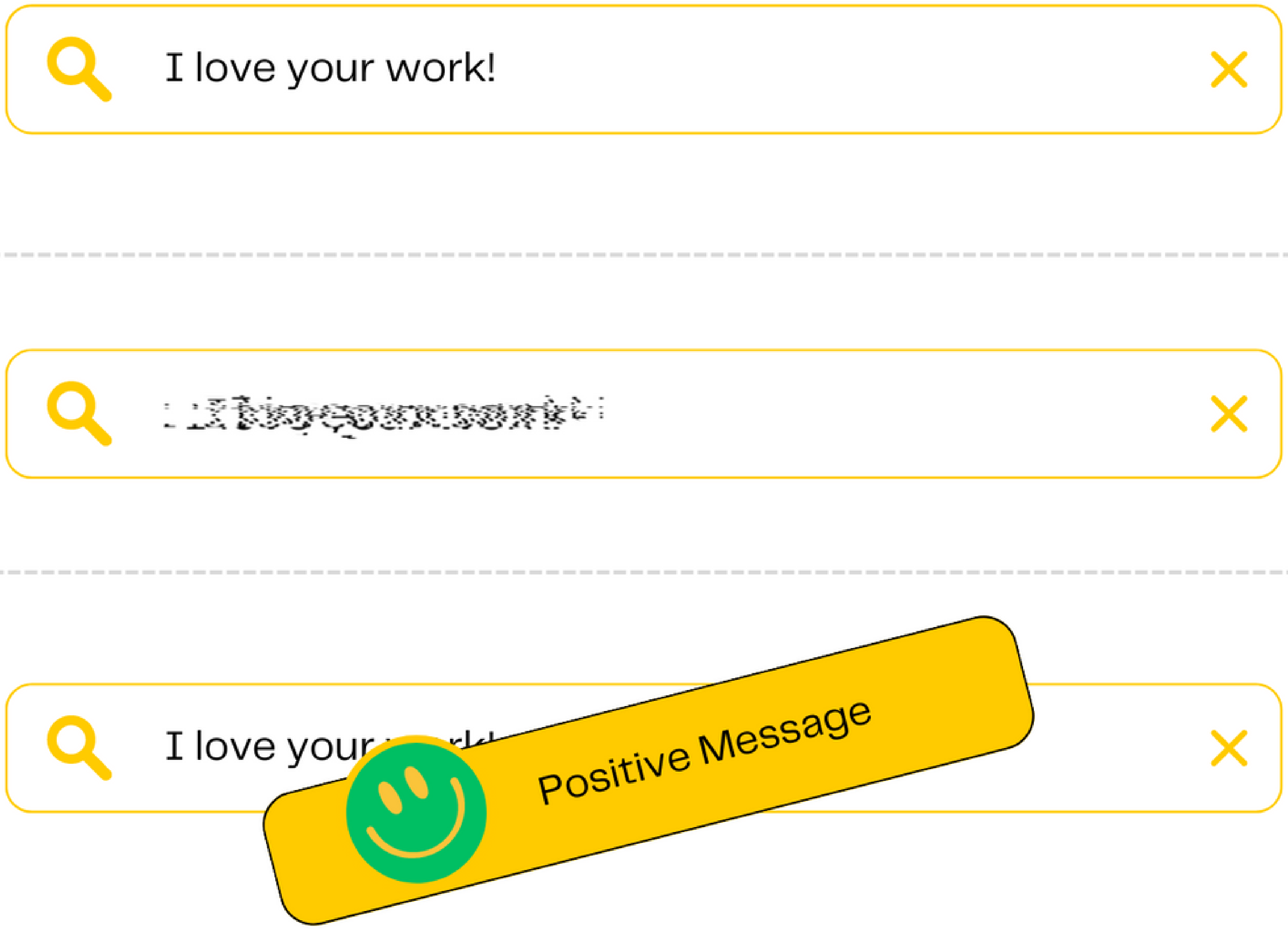

Accuracy for different quantization bits for Concrete XGBoost Model

Live Demo

Client Side
The message is quantized and then encrypted

Server side
The message is analyzed encrypted thanks to FHE

Client Side
Encrypted sentiment is sent back to client and recovered with the secret key



https://huggingface.co/spaces/zama-fhe/encrypted_sentiment_analysis

Contact and links

- https://eprint.iacr.org/2023/258.pdf
- zama.ai
- github.com/zama-ai/concrete-ml
- https://huggingface.co/zama-fhe
- discord.fhe.org

Thank you !

ZAMA