## Question 1

The question that we wanted to answer was:

* Theos DPGBDT algorithm uses a slightly different approach when splitting the training data amongst the decision trees.
* Does sticking to the “textbook algorithm” affect the accuracy of the predictions?

Unfortunately I misunderstood the code and confused things in our last meeting.

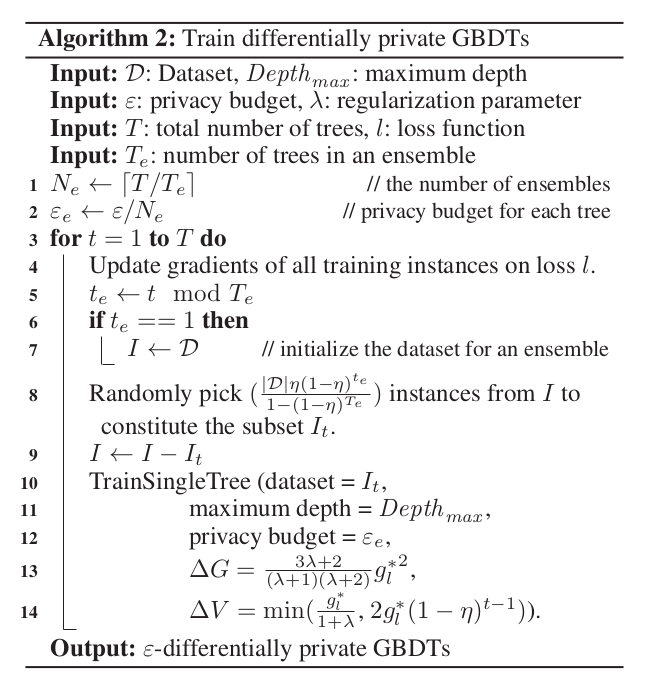
Let me clarify:

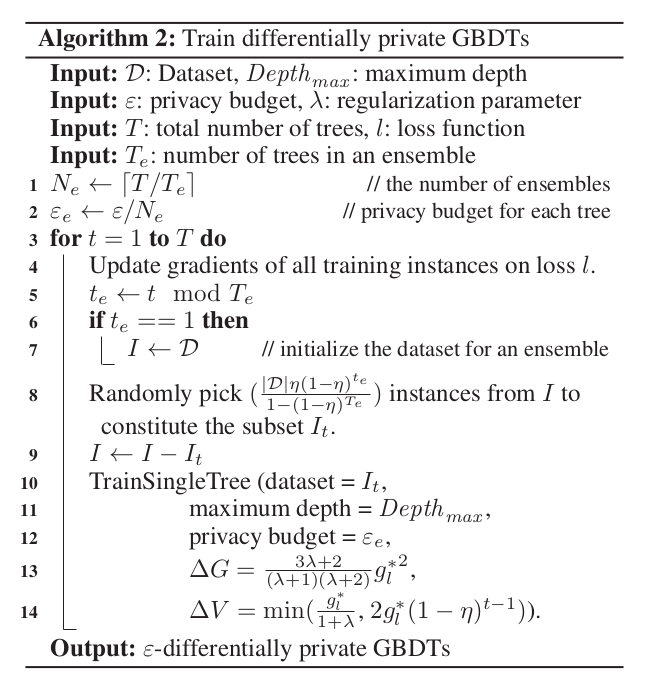
* In Theos code there exists an option to make the code exactly follow the textbook algorithm.
* The textbook algorithm gives each tree a different amount of samples according to some formula.
* Theo noticed that sometimes the last few trees don’t get enough samples, which can make predictions worse.
* Therefore he used his own method of splitting the training data as default.
* His own method just divides num\_samples by num\_trees. So each tree gets the same amount of samples. Whether the trees fetch their samples like in Theos code (randomly, on the fly, in the loop), or whether we distribute the samples beforehand, should not really matter in terms of correctness. It might however matter from a side-channel perspective.

All in all, what he does seems reasonable to me and should not be a privacy concern if implemented correctly. It’s just not what is formally proven in the paper.

## 

Formula (line 8):





Regarding my question if “shuffle & pick consecutive chunks” is ok for privacy (instead of the formula on line 8).

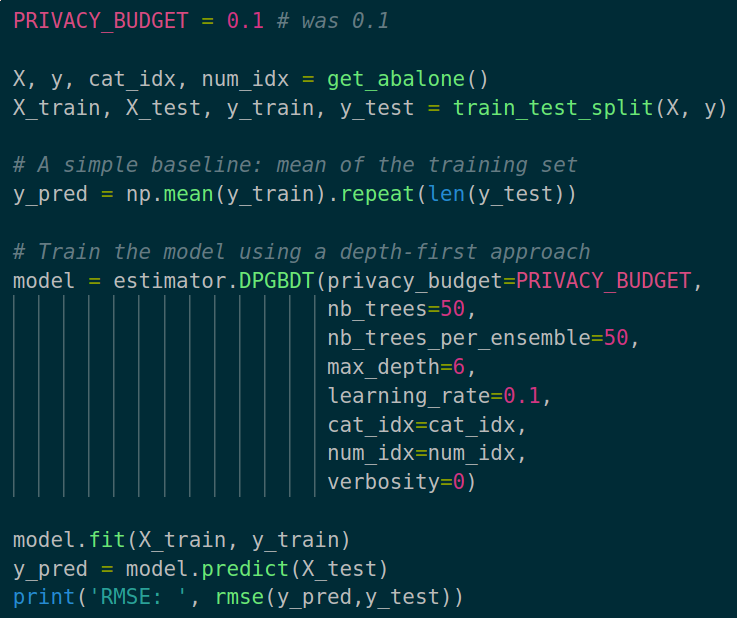
Esfandiar: The proof shuffles and then takes “connected” chunks in order.

so there should be no problem there.

## 

## Question 2

A minimal code example to use Theos DPGBDT code looks as follows:



For the abalone dataset this gave me the following results (10 runs each):

|  |  |  |  |
| --- | --- | --- | --- |
| Privacy budget 0.1: | Training method | RMSE | Number of trees used |
|  | Mean | 3.26 |  |
|  | DFS growth | 3.27 | 24/50 |
|  | BFS growth | 3.26 | 6/50 |
|  | 3-nodes growth | 3.28 | 22/50 |

|  |  |  |  |
| --- | --- | --- | --- |
| Privacy budget 1: | Training method | RMSE | Number of trees used |
|  | Mean | 3.2 |  |
|  | DFS growth | 2.4 | 37/50 |
|  | BFS growth | 3.2 | 4/50 |
|  | 3-nodes growth | 2.4 | 37/50 |

|  |  |  |  |
| --- | --- | --- | --- |
| Privacy budget 3: | Training method | RMSE | Number of trees used |
|  | Mean | 3.21 |  |
|  | DFS growth | 2.31 | 38/50 |
|  | BFS growth | 3.21 | 4/50 |
|  | 3-nodes growth | 2.35 | 38/50 |

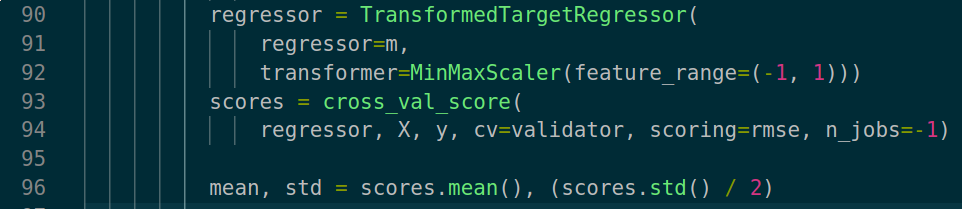
After that the RMSE does not get significantly smaller anymore, and the amount of “used/good” trees does not really increase. Even with privacy budget 8.

(There are more outputs and details here: <https://gitlab.inf.ethz.ch/kkari/enclave-hardening-ML/-/blob/master/ppml-insurance/my_results/>)

First of all our suspicion from the last meeting was correct. More “good” trees are created when increasing the privacy budget.

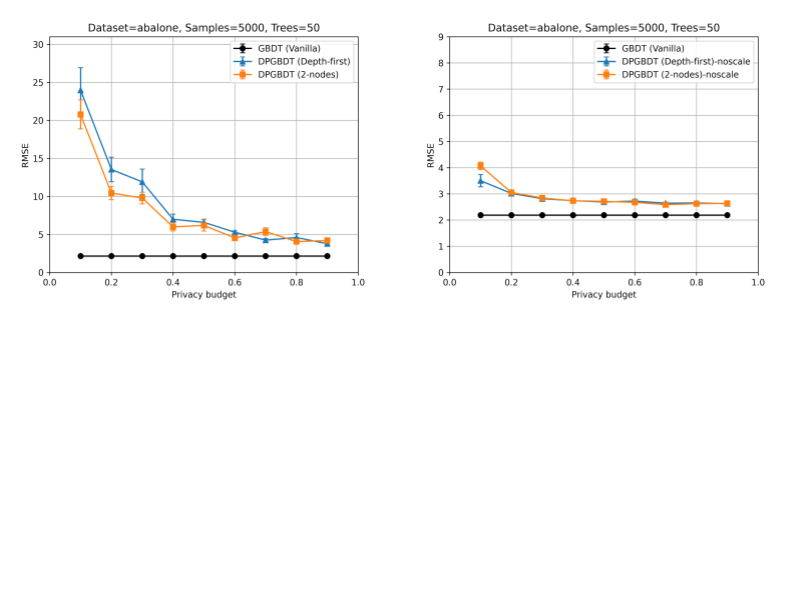
Second: it seems like the BFS (best leaf first growth) implementation is either not working, or it’s just a bad strategy. And this is probably also the reason why it does not appear in the thesis results. After some more investigation I’m quite sure the implementation is broken. but in any case I would suggest skipping it for now when translating to C++. We should be able to add it later as required.

The main thing that seemed strange to me though: For low privacy budgets (like 0.1), the computed RMSE is a lot smaller than in the graphs of the thesis (the values are around 3-4 instead of 20-25). Pinning down the reason for this, I noticed it’s due to one single line of code (line 92) which does some preprocessing of the data:



What it does is scaling all features to be between [-1,1]. In the case of abalone, X is already in that range so only y (age of the seashell) gets scaled. I read online that the scaling should normally not affect decision trees, however it somehow seems to have a pretty big effect in our DP case:

The 2 following graphs use the exact same code, except that the left uses the feature scaling (line 92), right doesn’t. Left looks exactly like in the thesis.



As of now I don’t see how “not-scaling” would have any privacy impact, so we might want to get rid of it? Or is there some more reasoning necessary @Esfandiar Mohammadi?

I noticed that some of the other datasets use the same form of scaling as well, namely (“year”, “questionnaires”, and “synthetic”), but I have not checked if the impact is similar yet.

Nope, scaling is important. If we don’t scale we would have to add more noise. The proof assumes the [-1,1].

## Question 3

Last thing I noticed, that seems like a bug in the code:

* The data is first divided into train/test splits. Let's say A and B.
* The Train() function is then called on set A, so far so good.
* However inside the Train() function there is another split into A1 and a smaller A2. A2 is then set aside to later determine if a tree (grown using his share of A1) is a “good/useful” tree.

So

* In general I am not sure if that is a thing.
* Secondly, the code uses A instead of A1 to grow the trees. So the data to judge a trees usefulness can overlap with the data used to build the tree.

@Esfandiar Mohammadi do you think we should keep the algorithm this way, and if yes is that a bug that should be corrected?

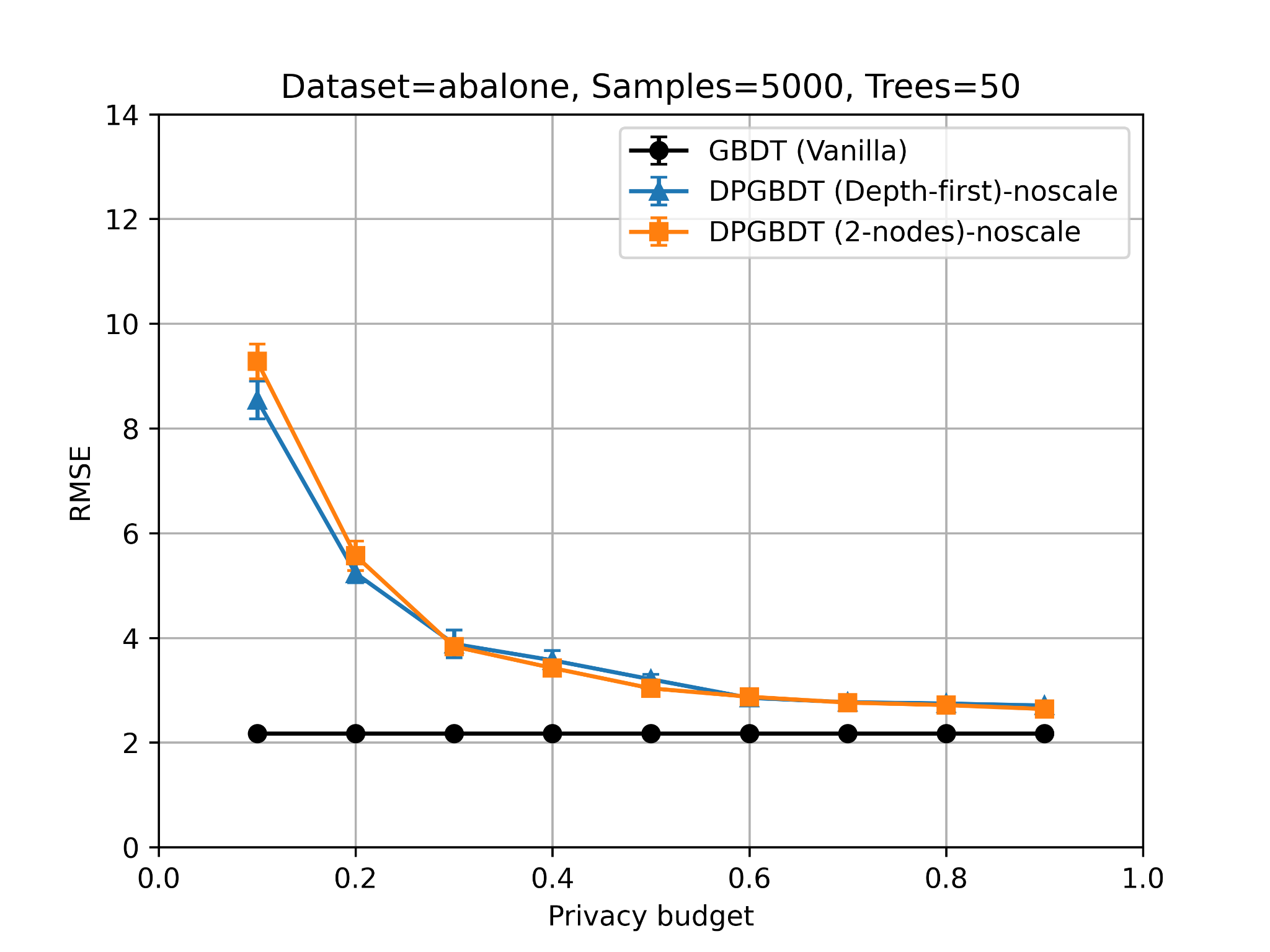
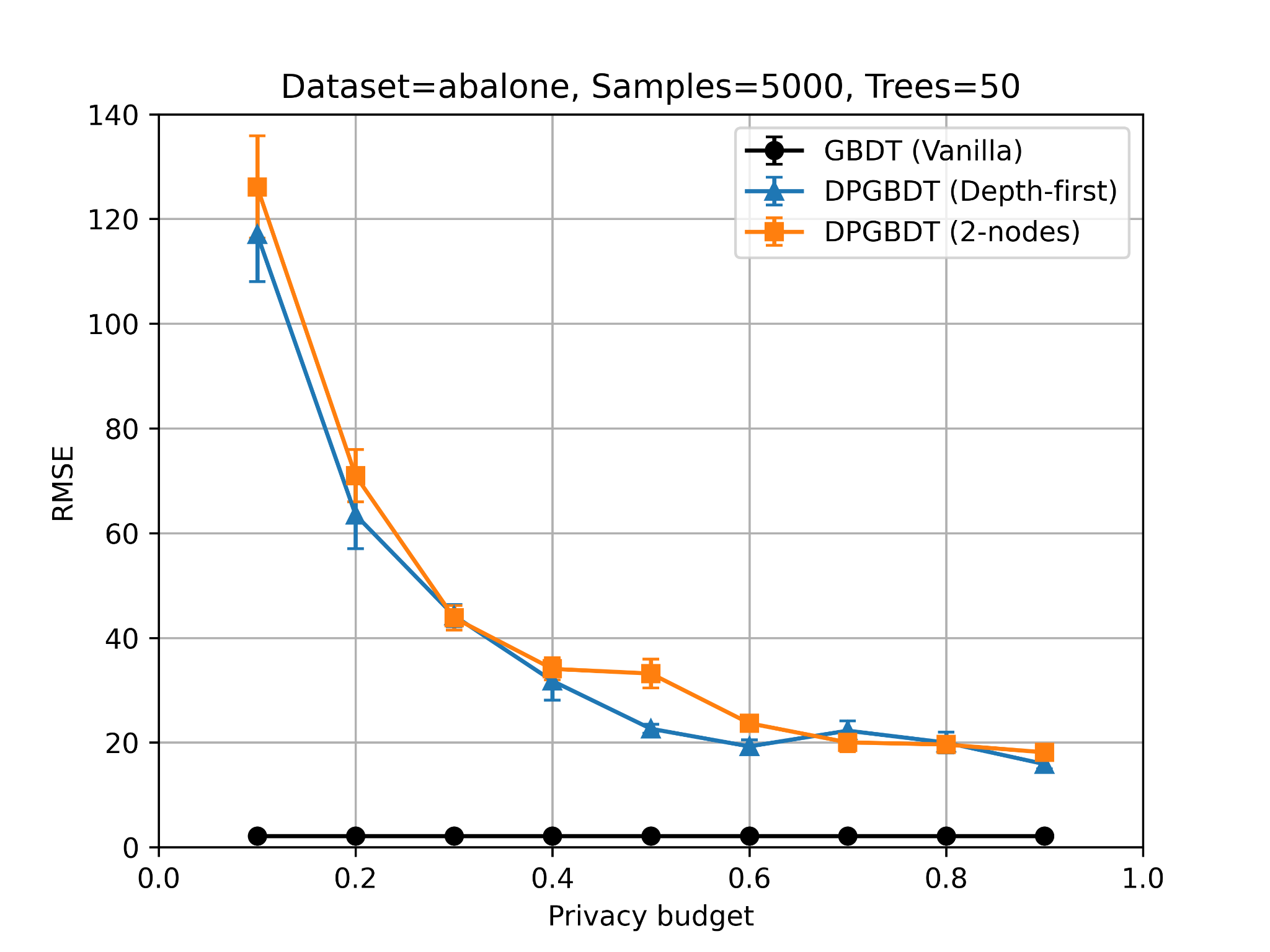
So the 2nd split is the reason why Theos code performs so well. It made sense to Esfandiar as Theos results almost seemed too good.

my small adaptation from “Theo's hypothesis” should be the way to go.

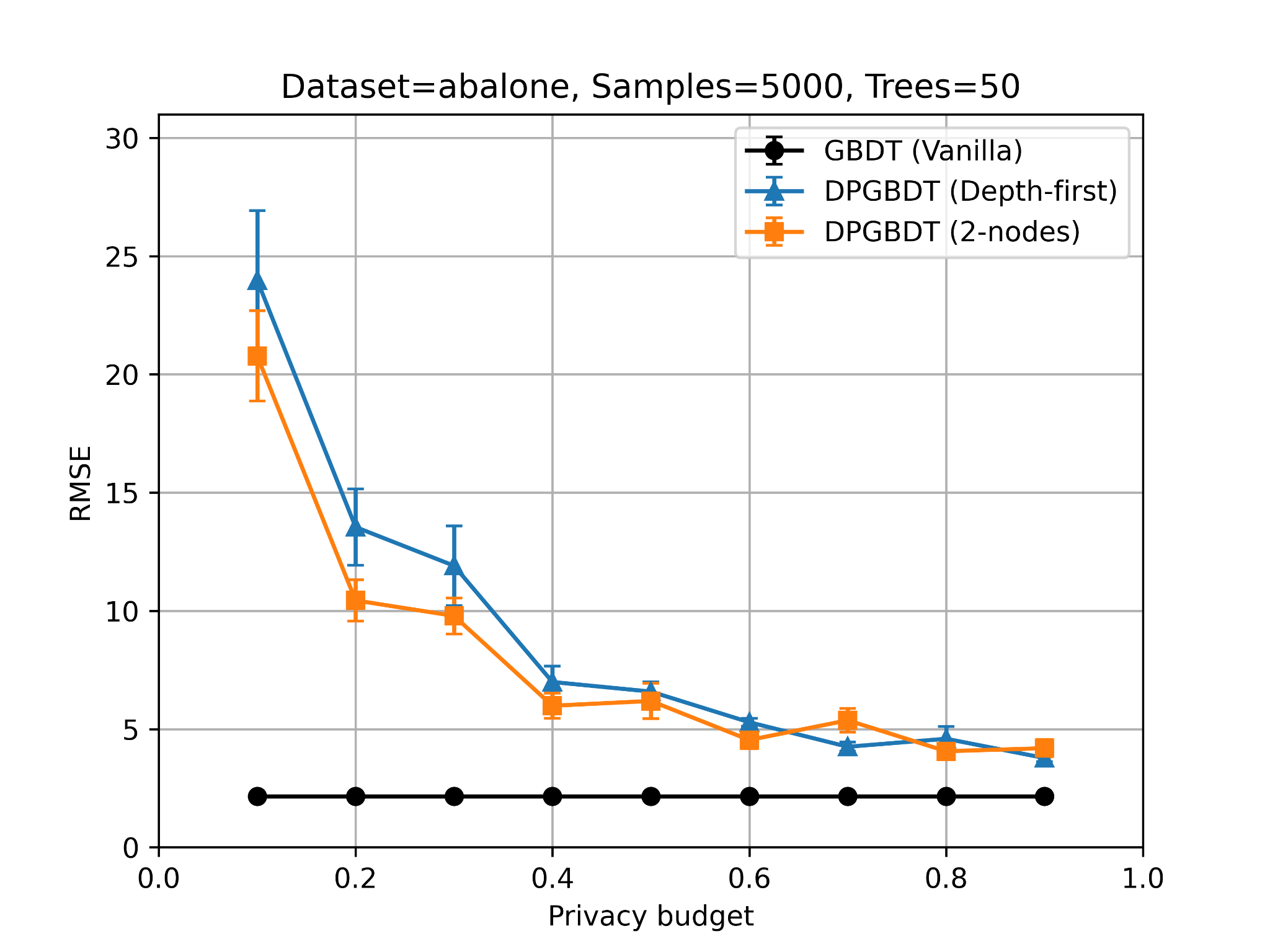
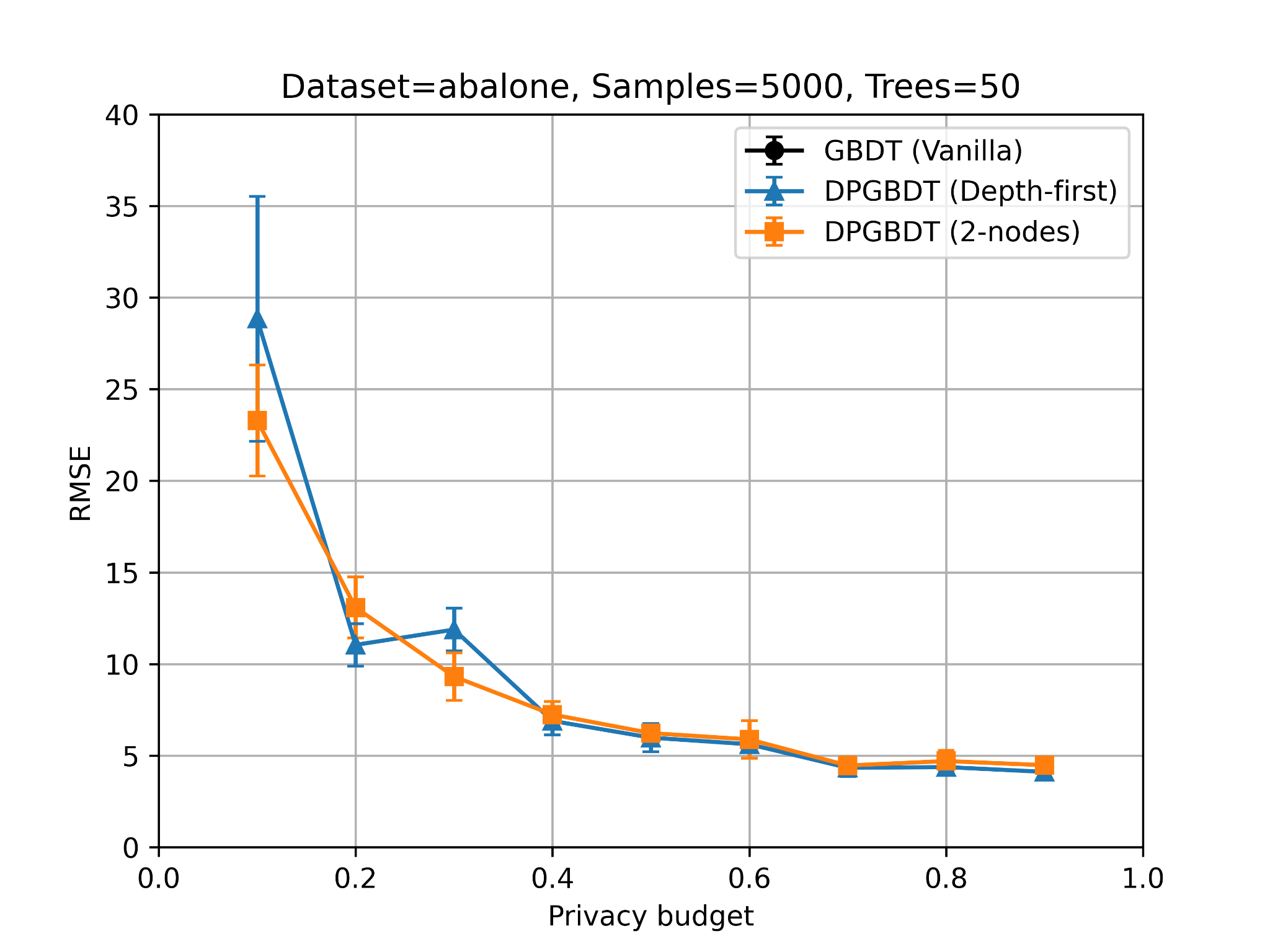
He’ll ask Moritz to try to try and prove it. He probably needs to add more noise somewhere, because A2 is basically reused. We’ll see how this goes. In any case it would be an interesting improvement to the base algorithm.

What we could always do if the dataset is large enough: Split A2 into 50 pieces. Then the data is not reused and thus it’s “easy to piece together the proof” (Esfandiar).

If we don’t do the second split into A1 and A2, and use all (in this case 50) trees for prediction, our result would look as follows (again with and w/o scaling, from question 2):



If we do the second split, and only use A1 to grow the trees and A2 for accepting/rejecting (What I think would be correct) it looks as follows:



my “fixed” version Theo/thesis version

So it gets a bit worse. Not what I expected. Probably it’s because we have less samples for training. What do you think @Esfandiar Mohammadi?

I’ll create another document for this topic