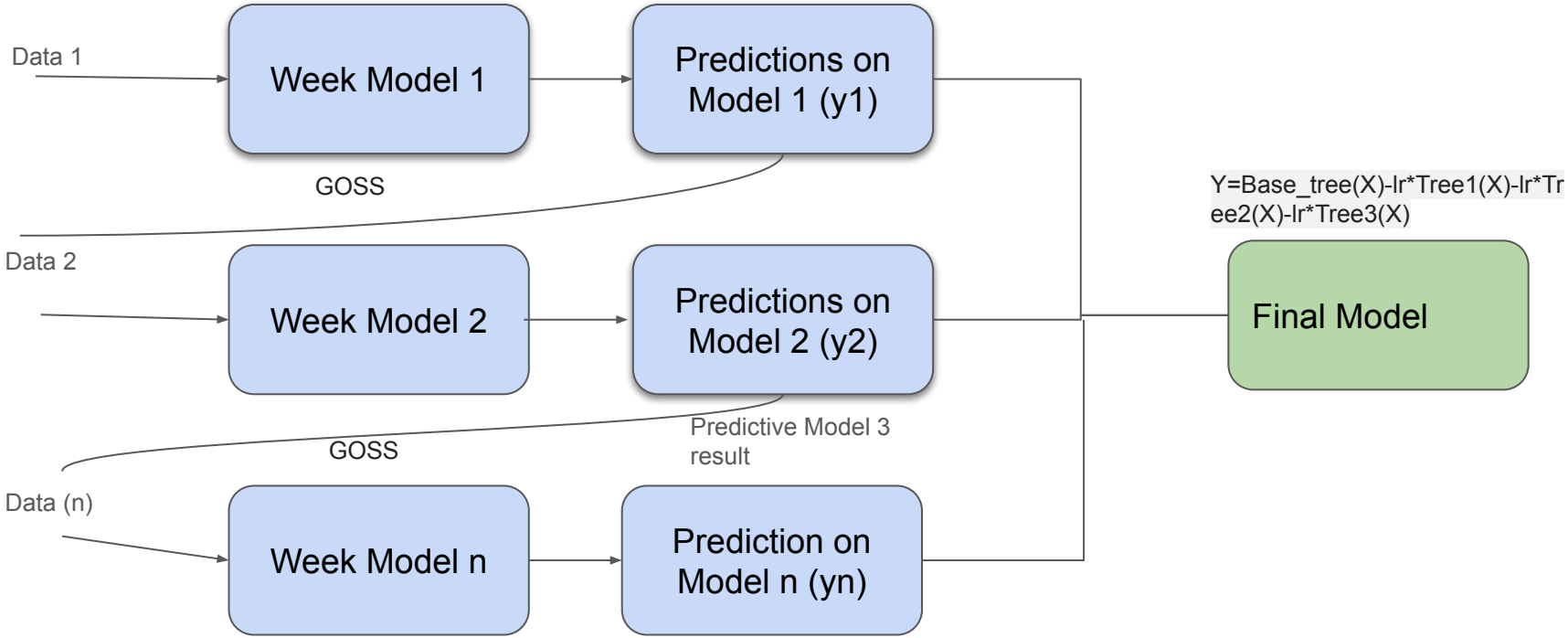
LightGBM Algorithms

Detailed presentation

What is LightGBM (Lightweight Gradient Boosting Machine) Algorithms



Week Model 1,2,..n are individual models. GOSS - Gradient based One Side Sampling

Why LGBM?

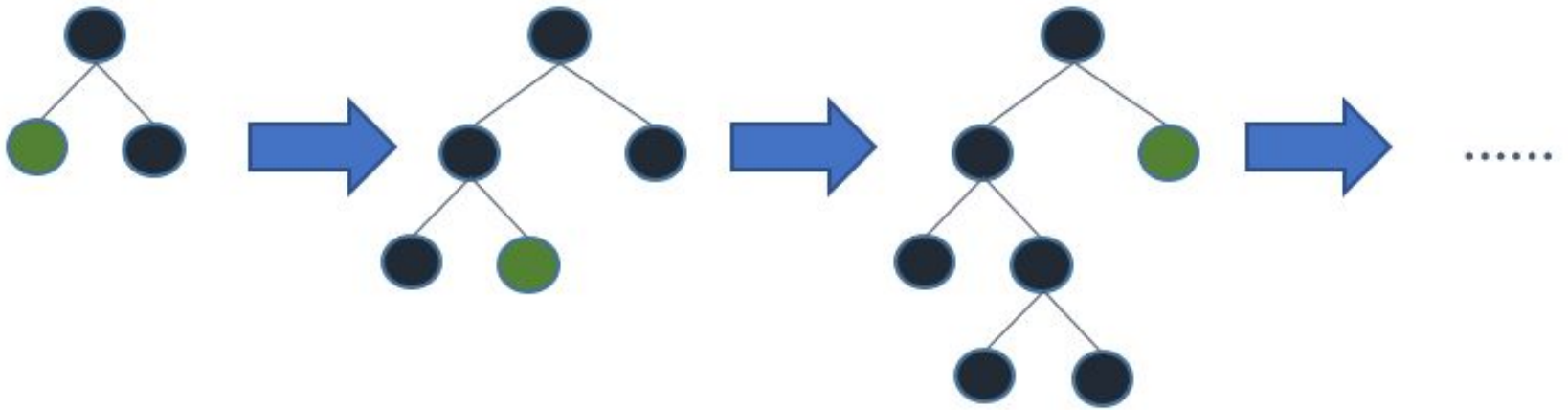
1. Faster training
2. Lower memory usage
3. It has support of parallel, distributed and GPU learning
4. Better accuracy
5. Capable to handle large scale data

How it 10X faster than XG Boost?

1. GOSS (Gradient Based One Side Sampling) - better sampling
    a. Whenever the error or gradient increases , the algorithm sort the gradients in descending order (high error …low error) and then it create next datasets with top 20 gradient record first (20%) and sample (10%) of small gradient records from the 80% of records and the sampling is happening only on the right hand side (ie 80% of the data) so it is called one side Sampling
2. EFB (Exclusive Feature Bundling) - better Feature Engineering
    a. Ex: if you have gender where M is 1 and F is 0 , we can create bundle like M is 11 and F is 10. In this way we can make the algorithm by reducing the 2D to 1D

How it split the data?

- LightGBM is a gradient boosting framework that uses tree based learning algorithms. It split the tree leaf wise i.e it will split the leaf node with the biggest split gain out of all the tree's leaves. The tree might be unbalanced.



Leaf-wise tree growth

Tuning Parameters of Light GBM

Better split:

- num_leaves : This parameter is used to set the number of leaves to be formed in a tree. Make sure num_leaves set must be smaller than 2^(max_depth) otherwise it may lead to overfitting.
- min_data_in_leaf : It is also one of the important parameters in dealing with overfitting. Setting its value smaller may cause overfitting so Its value should be hundreds to thousands of large datasets.
- max_depth: It specifies the maximum depth or level up to which tree can grow
- Try lambda_l1, lambda_l2 and min_gain_to_split for regularization

Faster speed

- bagging_fraction : Is used to perform bagging for faster results
- feature_fraction : Set fraction of the features to be used at each iteration
- max_bin : Smaller value of max_bin can save much time as it buckets the feature values in discrete bins which is computationally inexpensive.

Better accuracy

- Use bigger training data
- num_leaves : Setting it to high value produces deeper trees with increased accuracy but lead to overfitting. Hence its higher value is not preferred.
- max_bin : Setting it to high values has similar effect as caused by increasing value of num_leaves and also slower our training procedure.