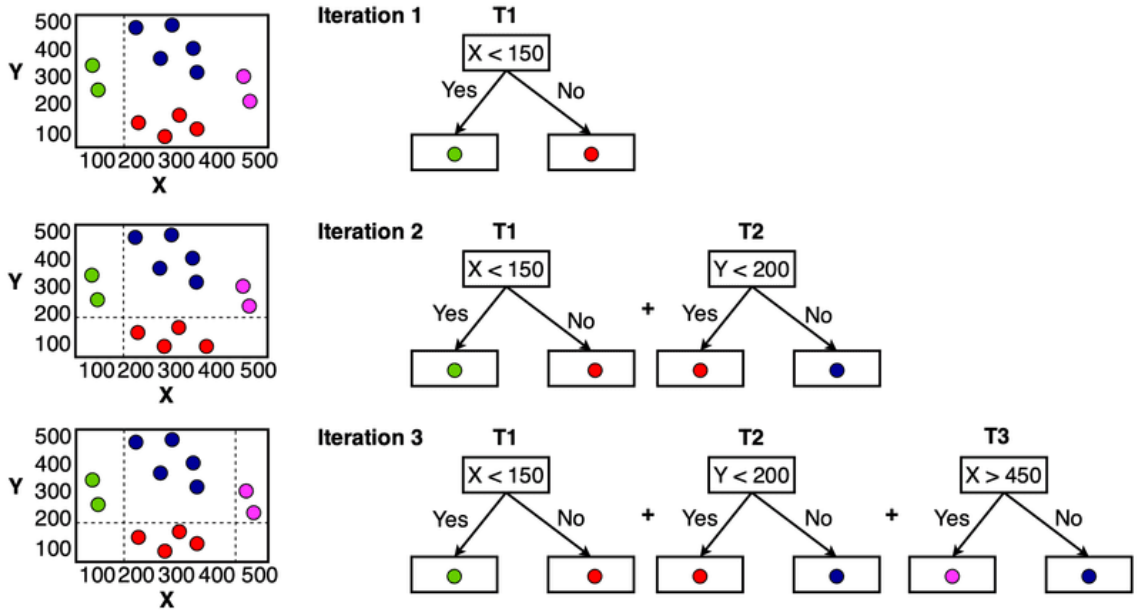
1. WTF is the Difference Between GBM and XGBoost?

For those who don't know about GBM and XGBoost, GBM (Gradient Boosting Machine) and XGBoost (eXtreme Gradient Boosting) are ensemble learning methods. Ensemble learning is a machine learning technique where multiple "weak" models (often decision trees) are trained and combined for further purposes.

The algorithm was based on the ensemble learning boosting technique shown in their name. Boosting techniques is a method that tries to combine multiple weak learners sequentially, with each one correcting its predecessor. Each learner would learn from their previous mistakes and correct the errors of the previous models.
That's the fundamental similarity between GBM and XGB, but how about the differences? We will discuss that in this article, so let's get into it.

1. GBM (Gradient Boosting Machine)

As mentioned above, GBM is based on boosting, which tries sequentially iterating the weak learner to learn from the error and develop a robust model. GBM developed a better model for each iteration by minimizing the loss function using gradient descent. Gradient descent is a concept to find the minimum function with each iteration, such as the loss function. The iteration would keep going until it achieves the stopping criterion.
For the GBM concepts, you can see it in the image below.



You can see in the image above that for each iteration, the model tries to minimize the loss function and learn from the previous mistake. The final model would be the whole weak learner that sums up all the predictions from the model.

2. XGB (eXtreme Gradient Boosting) and How It Is Different From GBM

XGBoost or eXtreme Gradient Boosting is a machine-learning algorithm based on the gradient boosting algorithm developed by Tianqi Chen and Carlos Guestrin in 2016. At a basic level, the algorithm still follows a sequential strategy to improve the next model based on gradient descent. However, a few differences of XGBoost push this model as one of the best in terms of performance and speed.

3. 1. Regularization

Regularization is a technique in machine learning to avoid overfitting. It's a collection of methods to constrain the model to become overcomplicated and have bad generalization power. It's become an important technique as many models fit the training data too well.

GBM doesn't implement Regularization in their algorithm, which makes the algorithm only focus on achieving minimum loss functions. Compared to the GBM, XGBoost implements the regularization methods to penalize the overfitting model.
There are two kinds of regularization that XGBoost could apply: L1 Regularization (Lasso) and L2 Regularization (Ridge). L1 Regularization tries to minimize the feature weights or coefficients to zero (effectively becoming a feature selection), while L2 Regularization tries to shrink the coefficient evenly (help to deal with multicollinearity). By implementing both regularizations, XGBoost could avoid overfitting better than the GBM.

4. 2. Parallelization

GBM tends to have a slower training time than the XGBoost because the latter algorithm implements parallelization during the training process. The boosting technique might be sequential, but parallelization could still be done within the XGBoost process.
The parallelization aims to speed up the tree-building process, mainly during the splitting event. By utilizing all the available processing cores, the XGBoost training time can be shortened.
Speaking of speeding up the XGBoost process, the developer also preprocessed the data into their developed data format, DMatrix, for memory efficiency and improved training speed.

5. 3. Missing Data Handling

Our training dataset could contain missing data, which we must explicitly handle before passing them into the algorithm. However, XGBoost has its own in-built missing data handler, whereas GBM doesn't.
XGBoost implemented their technique to handle missing data, called Sparsity-aware Split Finding. For any sparsities data that XGBoost encounters (Missing Data, Dense Zero, OHE), the model would learn from these data and find the most optimum split. The model would assign where the missing data should be placed during splitting and see which direction minimizes the loss.

6. 4. Tree Pruning

The growth strategy for the GBM is to stop splitting after the algorithm arrives at the negative loss in the split. The strategy could lead to suboptimal results because it's only based on local optimization and might neglect the overall picture.

XGBoost tries to avoid the GBM strategy and grows the tree until the set parameter max depth starts pruning backward. The split with negative loss is pruned, but there is a case when the negative loss split was not removed. When the split arrives at a negative loss, but the further split is positive, it would still be retained if the overall split is positive.

7. 5. In-Built Cross-Validation

Cross-validation is a technique to assess our model generalization and robustness ability by splitting the data systematically during several iterations. Collectively, their result would show if the model is overfitting or not.

Normally, the machine algorithm would require external help to implement the Cross-Validation, but XGBoost has an in-built Cross-Validation that could be used during the training session. The Cross-Validation would be performed at each boosting iteration and ensure the produce tree is robust.