IJNRD.ORG ISSN : 2456-4184
INTERNATIONAL JOURNAL OF NOVEL RESEARCH AND DEVELOPMENT (IJNRD) | IJNRD.ORG
An International Open Access, Peer-reviewed, Refereed Journal

XGBoost Algorithm and Its Comparative Analysis

Prof.Ankush Patil | Divya Pawar | Sanskar Shete | Sejal Tote | Harshal Rathod

Computer Department
Government College of Engineering, Yavatmal

Abstract: The study contains fundamental details about the XGBoost Machine Learning Approach of Classification. The discussion of the XGBoost algorithm and its comparison with other Classification algorithms, such as Gradient Boosting and Random Forest algorithm, which are used to address a variety of prediction and recommendation-related real-world problems, was more broadly covered. The XGBoost algorithm, its features, and the workings of its techniques, which are used in recommendation systems, were the paper's primary topics. These algorithms' goals, features, and comparison to other classification algorithms are all described in detail.

Keywords: Machine Learning, Tree Algorithms, Tree Boosting, Random Forest, XGBoost, LightGBM, CatBoost

Introduction: The digital world has a wealth of data including Internet of Things (IoT) data, cybersecurity data, mobile data, business data, social media data, health data, etc. Knowledge of artificial intelligence (AI), and more specifically, machine learning (ML), is essential to intelligently analyze these data and create the corresponding smart and automated applications. A potential problem of information overload that prevents timely access to items of interest on the Internet has been created by the exponential growth in the amount of digital information available and the number of Internet users. To analyze abundant of data or to classify it we need a strong and accurate algorithm to categories it.

When we want to develop any prediction or classification model. Increasing the accuracy of our model takes up the majority of our time. The XGBoost algorithm, however, is a better algorithm that can be used to achieve this. Extreme Gradient Boosting is the meaning behind the acronym XGBoost. The decision tree algorithm with gradient boost is employed. If you are not already familiar, please look at gradient boosted decision tree. Download the XGBoost library to your computer in order to use it. Results show that it outperforms the majority of other algorithms in speed. R, Python, Java, C++, and other languages all support XGBoost. It is a requirement for performing XGBoost to have vectorized data, preferably numerical data. In a nutshell, I would like to say that predictions from multiple decision trees are used rather than predictions from a single decision tree. XGBoost grows the decision trees using a leaf-wise growth strategy. This indicates that it divides the branch of the tree that minimizes the loss function the most. Though this approach can make the model vulnerable to overfitting, it is still preferable. Their widespread success can be attributed to the fact that they are simple to use and don't require a deep understanding of internal workings.

The key contributions of this paper are listed as follows:

1) To specify the subject matter of our research by taking into account the nature, properties, and capacity of the XGBoost algorithm to address various real-world classification problems.
2) To discuss how XGBoost algorithm-based solutions can be used in a variety of real-world application domains, such as systems that recommend products to users.
3) To give a thorough overview of the XGBoost algorithm and its uses in prediction systems, which can be used to increase the application's intelligence and capabilities.
4) Comparative analysis of the Various Classification Machine learning approaches like Gradient Boosting, Random Forest etc.
5) To emphasize and enumerate the potential research lines within the purview of our study for intelligent data analysis and services.

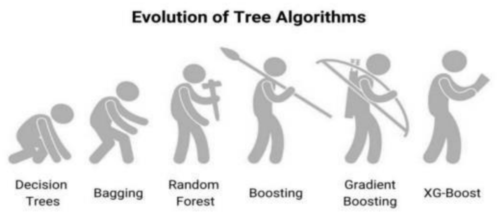
The rest of the paper is organized as follows. The next section presents the types of data and machine learning algorithms in a broader sense and defines the scope of our study. We briefly discuss and explain about XGBoost algorithm in the subsequent section followed by the Comparative analysis are discussed and summarized, and the final section concludes this paper.

Related Work: As device studying is turning into a crucial a part of the achievement of greater and greater applications together with credit score scoring, bioactive molecule prediction, sun and wind power prediction, oil rate prediction, type of galactic unidentified sources, sentiment analysis, it's miles crucial to find fashions which can deal efficaciously with complicated data, and with big quantities of it. With that angle in mind, ensemble strategies were a completely effective device to enhance the overall performance of more than one present fashions.

These strategies especially rely upon randomization techniques, which consist in generating many various answers to the trouble at hand, or on adaptive emphasis procedures. In fact, the abovementioned applications have in common that they all use ensemble methods and, in particular, a recent ensemble method called Extreme Gradient Boosting or XGBoost with very competitive results. This method, based on gradient boosting, has been consistently placed among the top contenders in Kaggle competitions. But XGBoost is not the only one to achieve remarkable results over a wide range of problems. Random forest is also well known as one of the most accurate and as a fast-learning method independently from the nature of the datasets, as shown by various recent comparative studies.

What is XGBoost?: Extreme Gradient Boosting is the abbreviation for XGBoost. A supervised learning method called XGBoost employs an ensemble strategy based on gradient boosting. Data scientists frequently use this scalable end-to-end tree boosting system to get cutting-edge outcomes on a variety of machine learning challenges. It can efficiently solve classification and regression issues while producing better outcomes. The Gradient Boosting machines were used to implement the first iteration of this algorithm. After making this work open-source, a sizable data science community began making contributions to the XGBoost projects, which helped to further improve the algorithm. XGBoost has developed into a software library and is now directly installable into our systems thanks to such an amazing community. It supports a number of interfaces, such as Python, R, and C++. XGBoost is a distributed gradient boosting library that has been optimized to be very effective, adaptable, and portable. It uses the Gradient Boosting framework to implement machine learning algorithms. It offers a parallel tree boosting to quickly and accurately solve a variety of data science problems. The comparison is carried out in terms of accuracy and training speed.

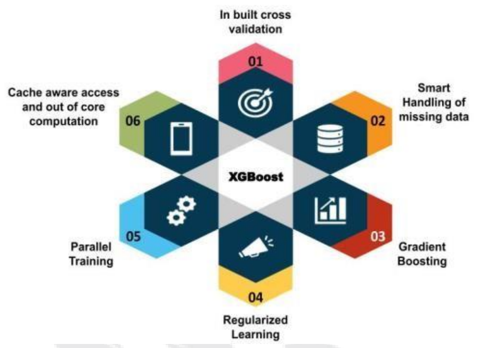
Evolution of Tree Algorithms:



Deep learning and artificial neural networks dominate the market for processing unstructured data like texts, audio, and image files. At the same time, tree-based algorithms predominate in the market when dealing with small or medium-sized structured data. And when we refer to a tree, Decision Trees serve as the fundamental unit of analysis. Although DTs could solve classification and regression problems, they quickly experienced overfitting problems. To address this, we combined several DTs with minimal changes to the data formation process. Bagging and Random Forest algorithms were produced by it. Following that, scientists concluded that randomly assembling trees was time-consuming and computationally inefficient. Why not construct trees successively and enhance those areas where earlier trees failed. Boosting was used in this situation. Later, to further enhance the performance of the Gradient boosting algorithms, researchers suggested model, algorithmic, and hardware optimizations. The result of all these improvements over Gradient boosting is called XG-Boost.

Features supported by XGBoost:

There are multiple supports that XGBoost provides us. These supports are growing over time as this framework is open-source, and people contribute continuously to enhance the features. Some of the most prominent features are:


In built cross validation
01
Smart Handling of missing data
02
Gradient Boosting
03
Regularized Learning
04
Parallel Training
05
Cache aware access and out of core computation
06
XGBoost

Model Support:

1. **Regularized learning:** Stronger algorithms are constantly concerned with overfitting; to address this, XG-Boost penalizes the overfitting tree using both L1 and L2 regularization methods.

2. **Gradient Tree Boosting:** Gradient Boosting Machines is the main gradient boosting algorithm that XG-Boost supports (GBM).

3. **Shrinkage and Column Subsampling:** We scale the learnings from more recent trees by a certain amount. This shrinkage lessens the influence of each individual tree and gives new trees a chance to enhance the performance of the model. Along with shrinkage, the dataset can also support column and row subsampling to create the GBM. The Random Forest algorithm also employs this column subsampling.

Split Finding Algorithm's Support:

1. **Weighted Quantile Sketch:** The distributed weighted quantile sketch algorithm was suggested by the authors of XG-Boost to automatically locate the ideal splitting points in weighted datasets. The main goal was to suggest a data structure that supports merge and pruning operations, each of which was shown to maintain a specific level of accuracy. This algorithm's specifics are described in the research paper's appendix section.

2. **Sparsity-aware split finding:** These days, it is quite likely that the data you receive is sparse due to a number of reasons, including missing values in the data, frequent zero entries in the statistics, and feature engineering artifacts like one-hot encoding. The XG-Boost algorithm learns the best way to handle sparsity by treating it as a missing value. According to the testing, this addition outperformed the initial implementation by a factor of 50.

System Design Support:

1. **Column Block for Parallel Learning:** In XGBoost, the trees are sequentially grown using parallelization. Getting the data sorted is the part of the tree learning algorithm that takes the longest. XG-Boost suggested using in-memory units known as blocks to store the data in order to do this effectively. We employ nested loops to build the base learners, with the outer loop representing the enumeration of a tree's leaf nodes and the inner loop representing the computation of the features. Since these loops cannot be switched out in the base implementation, computation overhead results. With the aid of column blocks, XG-Boost created a range of parallelized learning by making this loop interchangeable.

2. **Cache-aware Access:** The creators of XGBoost created a cache-aware prefetching algorithm that shifts read/write dependency to a longer dependency and aids in lowering runtime overhead when the number of data's rows is sizable. Each thread allots an internal buffer, retrieves the gradient statistics into it, and then executes accumulation in a mini-batch fashion. 3. **Blocks for Out-of-core computation:** XGBoost uses out-of-core computation by dividing the dataset into multiple blocks and storing these blocks into discs to handle big data that does not fit into the memory of our systems.

Evolved Supports:

1. **In-built cross-validation method:** Now that XG-Boost has a built-in cross validation method, it can automatically search and determine how many hosting iterations are needed.

2. **Continued Training:** On a fresher dataset, an XG-Boost model that has already been trained can be fine-tuned.

These features and supports are growing continuously as the XG-Boost library is open source.

Working: The XGBoost has a tree learning algorithm as well as linear model learning, and because of that, it is able to do parallel computation on a single machine. This makes it 10 times faster than any of the existing gradient boosting algorithms.

1. System Optimization

1) **Tree Pruning** – The XGBoost algorithm uses the depth-first approach, unlike the stopping criterion for tree splitting used by GBMS, which is greedy in nature and it also depends upon the negative loss criterion. The XGBoost instead uses the max depth feature/parameter, and hence it prunes the tree in a backward direction.

2) **Parallelization –** The process of sequential tree building is done using the parallelized implementation in the XGBoost algorithm. This is made possible due to the outer and inner loops that are interchangeable. The outer loop lists the leaf nodes of a tree, while the inner loop will calculate the features. Also, in order for the outer loop to start, the inner loop must get completed. This process of switching improves the performance of the algorithm.

3) **Hardware Optimization –** Hardware optimization was also considered during the design of the XGBoost algorithm. Internal buffers are allocated for each of the threads to store the gradient statistics.

2. Algorithmic Enhancements

1) **Awareness of Sparsity –** XGBoost is known to handle all different types of sparsity patterns very efficiently. This algorithm learns the nest missing value by seeing the training loss.

2) **Regularization –** In order to prevent overfitting, it corrects more complex models by implementing both the LASSO (also called L1) and Ridge regularization (also called L2).

3) **Cross-Validation:** It is having built-in crossvalidation features that are being implemented at each iteration in the model creation. This prevents the need to calculate the number of boosting iterations needed.

4) **Distributed Weighted Quantile Sketch –** It uses the distributed weighted quantile sketch to get the optimal number of split points among the weighted datasets.

Comparative Analysis of XGBoost algorithm with Random Forest and Gradient Boosting:

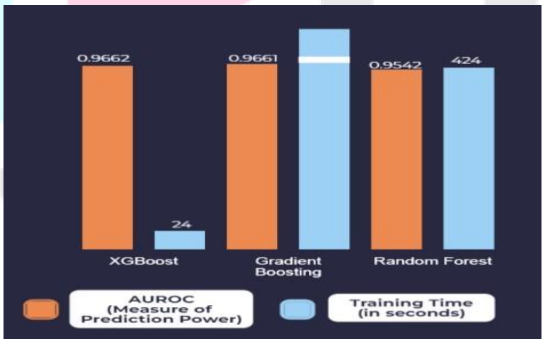


Fig: Performance comparison using "SKLearn's Make classification dataset"

Boosting is an example of iterative learning, which means that after making an initial prediction, the model will analyze its errors as a predictive worker and give more weight to the data points where it was incorrect in the subsequent iteration. After the second iteration, it analyzes its incorrect predictions once more and gives more weight to the data points that were incorrectly predicted in the subsequent iteration. This cycle of events keeps going. Therefore, theoretically, if a prediction has been made, there is a high probability that it did not occur by chance but rather as a result of careful analysis and data patterns. The most reliable model is one that avoids making predictions based solely on chance. Each tree in a random forest provides a prediction, and when all the trees have produced results, the mean, median, or mode of the collection is taken into account as the forest's prediction depending on the type of data (either continuous or categorical). On the surface, this appears to be fine, but there is a good chance that most of the trees could have predicted outcomes using some random chance, given that each tree had its own set of unique conditions, such as class imbalance, duplicate samples, overfitting, improper node splitting, etc. Recently, the family of gradient boosting algorithms has grown with a number of intriguing proposals (such as XGBoost, LightGBM, and CatBoost) that place a dual emphasis on speed and accuracy. Scalable ensemble technique XGBoost has proven to be a dependable and effective machine learning problem solver. With the use of selective sampling from high gradient instances, LightGBM is an accurate model that is dedicated to offering incredibly quick training performance. In order to increase the model's accuracy, CatBoost modifies the computation of gradients to prevent the prediction shift. In terms of training efficiency, generalization effectiveness, and hyper-parameter configuration. Additionally, both using carefully tuned models and their default settings, a thorough comparison of XGBoost, LightGBM, CatBoost, random forests, and gradient boosting has been carried out. The comparison's results show that CatBoost, despite the small differences, achieves the best generalization accuracy and AUC in the datasets under study. Although LightGBM is the fastest method, it is not the most precise. Last but not least, XGBoost comes in second place for both accuracy and training speed.

Conclusion: XGBoost is a faster algorithm when compared to other algorithms because of its parallel and distributed computing. XGBoost is developed with both deep considerations in terms of systems optimization and principles in machine learning. The goal of this algorithm is to push the extreme of the computation limits of machines to provide a scalable, portable, and accurate library.

References:

1] Towards Data Science. The Random Forest Algorithm - Towards Data Science. [online] https://towardsdatascience.com/the-randomforest-algorithmd457d499ffcd [Accessed 14 Nov. 2018]. [6]"

2] XGBoost Documentation — XGBoost 0.81 documentation. [Accessed 29 Nov. 2018]" https://xgboost.readthedocs.io/en/latest/

3] Quora. (2015). What are the advantages/disadvantages of using Gradient Boosting over Random Forests? [online] Available at:https://www.quora.com/What-arethe-advantages-disadvantages-ofusing-Gradient-Boosting-over-Random-Forests.

4] https://www.enjoyalgorithms.com/blog/xgboost-algorithm-in-ml

5] Candice Bentéjac, Anna Csörgő & Gonzalo Martínez-Muñoz A comparative analysis of gradient boosting algorithms [published:24 Aug 2022] Springer.

6] https://towardsmachinelearning.org/boostingalgorithms/

7] https://analyticsindiamag.com/top-xgboostinterview-questions-for-data-scientists/