frontiers in Applied Mathematics and Statistics

METHODS
published: 22 March 2021
doi: 10.3389/fams.2021.598833

An Explainable Bayesian Decision Tree Algorithm

Giuseppe Nuti[1], *Lluís Antoni Jiménez Rugama*[1]* *and Andreea-Ingrid Cross*[2]

[1]UBS, New York, NY, United States, [2]UBS, London, United Kingdom

Bayesian Decision Trees provide a probabilistic framework that reduces the instability of Decision Trees while maintaining their explainability. While Markov Chain Monte Carlo methods are typically used to construct Bayesian Decision Trees, here we provide a deterministic Bayesian Decision Tree algorithm that eliminates the sampling and does not require a pruning step. This algorithm generates the greedy-modal tree (GMT) which is applicable to both regression and classification problems. We tested the algorithm on various benchmark classification data sets and obtained similar accuracies to other known techniques. Furthermore, we show that we can statistically analyze how was the GMT derived from the data and demonstrate this analysis with a financial example. Notably, the GMT allows for a technique that provides explainable simpler models which is often a prerequisite for applications in finance or the medical industry.

Keywords: explainable machine learning, Bayesian statistics, greedy algorithms, Bayesian decision trees, white box

OPEN ACCESS

Edited by:
Victor Wu,
Tilt Dev, United States

Reviewed by:
Xiangyu Chang,
Xi'an Jiaotong University, China
Chandan Gautam,
Indian Institute of Science, India
Marcela Svarc,
Universidad de San Andrés, Argentina

***Correspondence:**
Lluís Antoni Jiménez Rugama
lluis.jimenez-rugama@ubs.com

Specialty section:
This article was submitted to Mathematics of Computation and Data Science, a section of the journal Frontiers in Applied Mathematics and Statistics

Received: *25 August 2020*
Accepted: *12 January 2021*
Published: *22 March 2021*

Citation:
Nuti G, Jiménez Rugama LIA and Cross A-I (2021) An Explainable Bayesian Decision Tree Algorithm. Front. Appl. Math. Stat. 7:598833. doi: 10.3389/fams.2021.598833

1 INTRODUCTION

The success of machine learning techniques applied to financial and medical problems can be encumbered by the inherent noise in the data. When the noise is not properly considered, there is a risk to overfit the data generating unnecessarily complex models that may lead to incorrect interpretations. Thus, there has been lot of efforts aimed at increasing model interpretability in machine learning applications [1–5].

Decision Trees (DT) are popular machine learning models applied to both classification and regression tasks with known training algorithms such as CART [6], C4.5 [7], and boosted trees [8]. With fewer nodes than other node-based models, DT are considered an explainable model. In addition, the tree structure can return the output with considerably fewer computations than other more complex models. However, as discussed by Linero in [9], greedely constructed trees are unstable. To improve the stability, new algorithms utilize tree ensembles such as bagging trees [10], Random Forests (RF) [11], and XGBoost (XG) [12]. But increasing the number of trees also increases the number of nodes and therefore the complexity of the model.

The Bayesian approach was introduced to solve the DT instability issue while producing a single tree model that accounts for the noise in the data. The first techniques, also known as Bayesian Decision Trees, were introduced in [13], BCART [14, 15], and BART [16]. The former article proposed a deterministic algorithm while the other three are based on Markov Chain Monte Carlo convergence. Some recent studies have improved upon these algorithms, for review see [9], and include a detailed interpretability analysis of the model, [17]. While most of the Bayesian work is based on Markov Chain convergence, here we take a deterministic approach that: 1) considers the noise in the data, 2) generates less complex models measured in terms of the number of nodes, and 3) provides a statistical framework to understand how the model is constructed.

The proposed algorithm departs from [13], introduces the *trivial partition* to avoid the pruning step, and generalizes the approach to employ any conjugate prior. Although this approach is

Bayesian, given the input data and model parameters the resulting tree is deterministic. Since it is deterministic, one can easily analyze the statistical reasons behind the choice of each node. We start with an overview of the Bayesian Decision Trees in **Section 2**. **Section 3** describes the building block of our algorithm, namely the partition probability space, and provides the algorithms to construct the greedy-modal tree (GMT). **Section 4** benchmarks the GMT vs. common techniques showing that the GMT works well for various publicly available data sets. Finally, a trading example is discussed in **Section 5** followed by some conclusive remarks in **Section 6**.

2 BAYESIAN DECISION TREES OVERVIEW

A Decision Tree is a directed acyclic graph. All its nodes have a parent node except the root node, the only one that has no parent. The level $\ell \in \mathbb{N}_0$ of a node is the number of ancestors of the node, starting from 0 at the root node. We classify the nodes as either *sprouts* or *leaves*. While sprouts point to two other child nodes in the case of binary trees, leaves are terminal nodes containing the model information. Each sprout contains a rule used to choose one of its children. To query the tree, we start at the root node and apply the rules to an input to select the child nodes until we reach a leaf.

We can use Decision Trees to partition $\mathbb{R}^d$ and assign an output to each subset of the partition. In this work, we restrict ourselves to finite partitions of $\mathbb{R}^d$. Each leaf of the tree will correspond to one of the subsets of the partition, one-to-one and onto. Our approach of Decision Trees departs from the Bayesian Decision Tree framework which provides a marginal likelihood to a Decision Tree based on some input data. Let's define the input data as $\mathcal{D} = [(x_i, y_i)]_{i=1}^n$ with n independent observations. A point $x = (x^1, \ldots, x^d)$ in $\mathbb{R}^d$ contains the features of each observation whose outcome y is randomly sampled from a random field Y_x. The Bayesian Decision Tree assumes the distribution of Y_x is constant at each leaf. Given x, the tree will return the posterior distribution of the parameters θ generating Y within the leaf x belongs to. In practice, the distribution of Y will determine the type of problem we are solving: a discrete random variable translates into a classification problem whereas a continuous random variable translates into a regression problem.

The probability of such a Bayesian Decision Tree, namely $\mathcal{T}$, can be computed with the usual Bayes approach,

$$p(\mathcal{T}|\mathcal{D}) = \frac{p(\mathcal{D}|\mathcal{T})}{p(\mathcal{D})} p(\mathcal{T}), \tag{1}$$

where $p(\mathcal{T})$ is the prior distribution over the tree space. To compute the marginal likelihood $p(\mathcal{D}|\mathcal{T})$, we consider the partition $\mathcal{D} = \{\mathcal{D}_1, \ldots, \mathcal{D}_k\}$ induced by $\mathcal{T}$ and take the product of the marginal likelihoods at each leaf,

$$p(\mathcal{D}|\mathcal{T}) = \prod_{j=1}^{k} L(\mathcal{D}_j) = \prod_{j=1}^{k} \int_{\Theta} p(\mathcal{D}_j|\theta) p(\theta) d\theta \tag{2}$$

The probability $p(\theta)$ from **Eq. 2** is the prior distribution of the parameters θ. In this article, we will assume for simplicity that

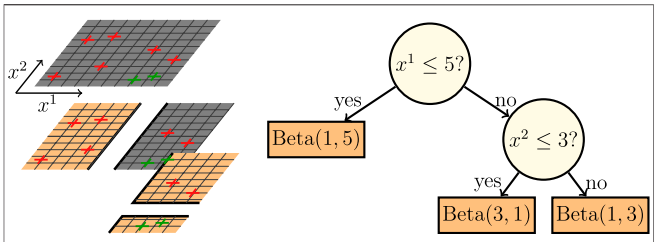

FIGURE 1 | Example of a Bayesian Decision Tree for a 2-categories example in $\mathbb{R}^2$. On the left: the data set is displayed three times. The first layer corresponds to the data set before any split. The second layer displays the two sets resulting from splitting along dimension 1. The third layer is an additional split of the right subset along dimension 2. The final leaves are displayed in orange. On the right: equivalent tree with posterior distributions for probability of being red assuming a Beta $(1, 1)$ prior distribution. The marginal likelihood of the tree is $p(\mathcal{D}|\mathcal{T}) = B(1,5)B(3,1)B(1,3)/B(1,1)^3$, where B is the Beta function.

$p(\theta)$ is independent of $\mathcal{T}$ although this is not a requirement. The purpose of $p(\theta)$ is therefore two-fold:

- To obtain the tree probability from **Eq. 1**,
- To compute the posterior distribution of the parameters generating Y at each leaf.

Figure 1 shows a Bayesian Decision Tree that partitions $\mathbb{R}^2$ into $[(-\infty, 5) \times (-\infty, \infty), (5, \infty) \times (-\infty, 2.5), (5, \infty) \times (2.5, \infty)]$ and the corresponding posterior distributions Beta $(1, 5)$, Beta $(3, 1)$, and Beta $(1, 3)$. More information about conjugate priors and marginal likelihoods can be found in [18].

In an attempt to build explainable Bayesian Decision Trees, we define a greedy construction that does not apply Markov Chain Monte Carlo. This construction balances the greedy approach from [6] with the Bayesian approach discussed in [9, 14–17]. For this, we compute the probability of each split at every node and choose the modal split. This results in a model that performs well with different data sets as shown in **Section 4**.

3 FROM THE PARTITION PROBABILITY SPACE TO BAYESIAN DECISION TREES

The building block of the GMT algorithm is the *partition probability space*. For this space, we only consider binary partitions of the form $S_{r,h} = \{\{x \in \mathbb{R}^d \text{ such that } x^r \leq h\}, \{x \in \mathbb{R}^d \text{ such that } x^r > h\}\}$ where $r \in \{1, \ldots, d\}$, $h \in \mathbb{R} \backslash \{x_1^r, \ldots, x_n^r\}$. Any partition of this form will induce a partition $\{\mathcal{D}_1, \mathcal{D}_2\}$ of $\mathcal{D}$. Note that any of these two subsets are allowed to be the empty set. Finally, for each dimension we identify all partitions that result in the same non-empty $\mathcal{D}_1$ and $\mathcal{D}_2$. All partitions that leave $\mathcal{D}_1$ or $\mathcal{D}_2$ empty are also identified as the *trivial partition* S_0. After identification, we will have $1 + \sum_{r=1}^{d} (n_r - 1)$ different partitions S, n_r tbeing the number of different features along dimension r. Following the minimum margin classifier idea, the partition representative location h will be placed at the mid-point between contiguous different features in a dimension.

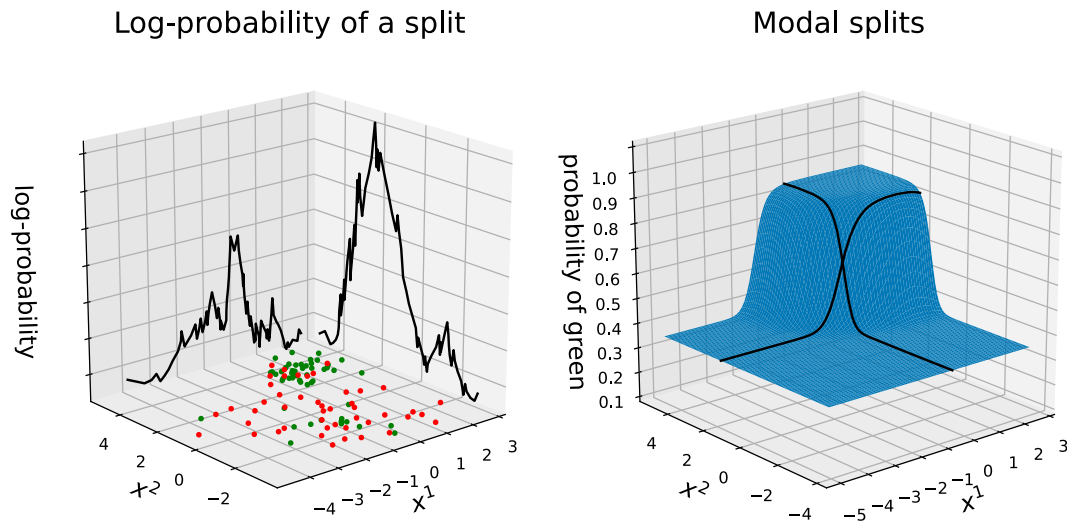

FIGURE 2 | Example of a data set whose outcomes are either green or red. The location of the points is sampled from a mixture of two Gaussian distributions with equal probability. One distribution draws outcomes from a Bernoulli distribution with probability 0.25, while the other from a Bernoulli with probability 0.75. On the left: log-probabilities of all possible non-trivial partitions given the data set. On the right: actual probability of a point being green and modal splits along each dimension.

The *partition probability space* is the finite probability space defined by partitions S and their probabilities $p(S|\mathcal{D})$. These probabilities can be computed using **Eqs 1**, **2** when replacing $\mathcal{T}$ by S. In practice, we will work with $\ln[p(\mathcal{D}|S)p(S)]$ which are the log-probabilities from **Eq. 1** omitting the constant normalizing factor $p(\mathcal{D})$:

$$\ln\left[p(\mathcal{D}|S)p(S)\right] = \ln\left[p(S|\mathcal{D})\right] + \ln\left[p(\mathcal{D})\right] \tag{3}$$

We will also need the feature sorted indices of the input features for computation and visualization purposes, namely $i_1, \ldots, i_n$ such that $x^r_{i^r_j} \leq x^r_{i^r_{j+1}}$ for all $r = 1, \ldots, d$ and $j = 1, \ldots, n-1$. An example of the split probability space is shown in **Figure 2**. In this example, the inputs x live in $\mathbb{R}^2$ and the outcomes are drawn from a Bernoulli random variable. The points x are generated from two independent Gaussian distributions equally likely, i.e. we drew from each distribution with probability 0.5. The first distribution is a multivariate Gaussian with mean $(-1, -1)^t$ and covariance $2I$. Points sampled from this distribution have a probability of 0.25 of being green. The second distribution is another multivariate Gaussian with mean $(1, 3)^t$ and covariance $0.5I$. In this case, the probability of being green is 0.75. Because the mean of these Gaussian distributions are further apart along the x^2 axis, the most probable partitions given the data are found along this dimension.

Each partition S can be encoded into a tree node $\mathcal{N}$: if the partition is $\int_0$, the node becomes a leaf and stores the posterior hyper-parameters of θ; for any other $S_{r,h}$, the node becomes a sprout and stores the values r and h. Among all partitions, the

Algorithm 1 Returns the modal node for the general problem given $\mathcal{D}$ and prior hyper-parameters α.

procedure FIND_MODAL_NODE($\mathcal{D}, \alpha$)
  $\alpha^* \leftarrow$ posterior_hyper($\mathcal{D}, \alpha$) ▷ Obtain the posterior hyper-parameters using $\mathcal{D}$.
  $q \leftarrow \ln(L(\mathcal{D})) + \ln(p(\mathcal{S}_0))$ ▷ Compute (3) for $\mathcal{S}_0$.
  $\mathcal{N} \leftarrow$ create_leaf(α^*) ▷ Initialize with the leaf.
  for each dimension r in $1, \ldots, d$ **do**
    for each observation j in $1, \ldots, n-1$ **do**
      if $x^r_{i^r_j} \neq x^r_{i^r_{j+1}}$ **then**
        $h \leftarrow (x^r_{i^r_j} + x^r_{i^r_{j+1}})/2$ ▷ Compute the location of the split.
        $\mathcal{D}_1 \leftarrow \{\boldsymbol{x}_{i^r_a}, y_{i^r_a}\}_{a=1}^{j}$ ▷ Generate $\mathcal{D}_1$ given $\mathcal{D}$, r, and h.
        $\mathcal{D}_2 \leftarrow \{\boldsymbol{x}_{i^r_a}, y_{i^r_a}\}_{a=j+1}^{n}$ ▷ Generate $\mathcal{D}_2$ given $\mathcal{D}$, r, and h.
        $q_{\text{new}} \leftarrow \ln(L(\mathcal{D}_1)) + \ln(L(\mathcal{D}_2)) + \ln(p(\mathcal{S}_{r,h}))$ ▷ Compute (3) for $\mathcal{S}_{r,h}$.
        if $q_{\text{new}} > q$ **then** ▷ Store the new node with higher log-probability.
          $q \leftarrow q_{\text{new}}$
          $\mathcal{N} \leftarrow$ create_sprout(r, h)
        end if
      end if
    end for
  end for
  Return $\mathcal{N}$
end procedure

mode and its node are of particular interest. **Algorithm 1** returns the modal node in the general case. For the classification problem, we also provide **Algorithm 2** with $\mathcal{O}(dn)$ cost assuming $p(\theta)$ follows a Dirichlet conjugate prior. Both algorithms start by computing $\ln[p(\mathcal{D}|S_0)p(S_0)]$ and initializing $\mathcal{N}$ to be a leaf. Then, they loop through each dimension and the sorted features to verify whether there exists a new node with higher log-probability. Because the features are sorted, there is at most one observation that moves from $\mathcal{D}_2$ to $\mathcal{D}_1$ when j increases by one.

With the partition space and modal node defined, we can introduce the GMT construction. We start by finding the modal node $\mathcal{N}$ for our initial data set $\mathcal{D}$. This node is the root of the tree and will be returned by the train method in **Algorithm 3**. If $\mathcal{N}$ is a leaf, the GMT is completed. Otherwise, we split $\mathcal{D}$ into $\mathcal{D}_1$ and $\mathcal{D}_2$ according to $\mathcal{N}$. We repeat the process for the new input data sets $\mathcal{D}_1$ and $\mathcal{D}_2$, and link $\mathcal{N}_1$ and $\mathcal{N}_2$ to their parent $\mathcal{N}$. The recursion is defined in the grow_tree method from **Algorithm 3**. Note that **Algorithms 1 and 2** are just two implementations of find_modal_node, but one can replace this method by any other that returns the desired node based on the partition space. In addition, one can easily compute $\ln[p(\mathcal{D}|\mathcal{T})]$ for the GMT by adding the leaves' $\ln[L(\mathcal{D}_j)]$ calculated in **Algorithm 1** line 2, or **Algorithm 2** line 2. In practice, we realized that the GMT marginal log-likelihood $\ln[p(\mathcal{D}|\mathcal{T})]$ tends to be the highest when exploring for different possible roots.

The average cost of **Algorithm 3** is $\mathcal{O}[c(n)\ln(n)]$ where $c(n)$ is the cost of find_modal_node. If we choose find_modal_node to be **Algorithm 2**, the average cost of **Algorithm 3** becomes $\mathcal{O}[dn\ln(n)]$. While **Algorithms 1 and 2** only look at one successor ahead, we could improve the greedy exploration by looking at several levels ahead as suggested in [13]. Looking at m levels ahead comes at the expense of increasing the order of $c(n)$, for instance $c(n) = (dn)^m$ in the case of **Algorithm 2**. **Section 4** shows that the GMT constructed by looking at only one level ahead performs well in practice.

4 BENCHMARK

4.1 Decision Trees, Random Forests, XGBoost, and GMT

In this Section we use **Algorithm 2 and 3** to construct the GMT. We assume that the outcomes, 0 or 1, are drawn from Bernoulli random variables. The prior distribution $p(\theta)$ is chosen to be the Beta $(10, 10)$ and each tree will return the expected probability of drawing the outcome 0. The prior probabilities for each partition will be $p(S_0) = 1 - 0.9^{1+\ell}$ and $p(S_{r,h}) = 0.9^{1+\ell}/dn_r$, where ℓ is the level and n_r the number of non-trivial partitions along r. Note that the denominator d in $p(S_{r,h})$ is implicitly assuming a uniform prior distribution over the dimension space. One could also project the probabilities on each dimension to visualize which features are most informative. As an alternative to the suggested $p(S)$, one can use the partition margin weighted approach from [13].

The accuracy is measured as a percentage of the correct predictions. Each prediction will simply be the highest probability outcome. If there is a tie, we choose the category 0

Algorithm 2 Returns the modal node given $\mathcal{D}$ and Dirichlet prior hyper-parameters α.

1: **procedure** FIND_MODAL_NODE($\mathcal{D}, \alpha$)
2: $\alpha^* \leftarrow \alpha$ ▷ Initialize the posterior hyper-parameters.
3: **for** each observation j in $1, \ldots, n$ **do** ▷ Obtain the posterior hyper-parameters for $\mathcal{D}$.
4: $\alpha^*[y_j] \leftarrow \alpha^*[y_j] + 1$
5: **end for**
6: $q \leftarrow \ln(B(\alpha^*)) - \ln(B(\alpha)) + \ln(p(\mathcal{S}_0))$ ▷ Compute (3) for $\mathcal{S}_0$.
7: $\mathcal{N} \leftarrow$ create_leaf(α^*) ▷ Initialize with the leaf.
8: $s_\alpha \leftarrow \text{sum}(\alpha)$ ▷ Obtain pseudo count.
9: **for** each dimension r in $1, \ldots, d$ **do**
10: $q_1 \leftarrow 0$ ▷ Initialize $\ln(L(\mathcal{D}_1))$
11: $q_2 \leftarrow \ln(B(\alpha^*)) - \ln(B(\alpha))$ ▷ Initialize $\ln(L(\mathcal{D}_2))$
12: $c_1 \leftarrow \alpha$ ▷ Initialize the posterior hyper-parameters for $\mathcal{D}_1$.
13: $c_2 \leftarrow \alpha^*$ ▷ Initialize the posterior hyper-parameters for $\mathcal{D}_2$.
14: **for** each observation j in $1, \ldots, n-1$ **do**
15: $q_1 \leftarrow q_1 + \ln(c_1[y_{i_j}]/(j + s_\alpha - 1))$ ▷ Update q_1 adding j to $\mathcal{D}_1$
16: $c_1[y_{i_j^r}] \leftarrow c_1[y_{i_j^r}] + 1$ ▷ Update c_1 adding j to $\mathcal{D}_1$
17: $c_2[y_{i_j^r}] \leftarrow c_2[y_{i_j^r}] - 1$ ▷ Update c_2 removing j from $\mathcal{D}_2$
18: $q_2 \leftarrow q_2 - \ln(c_2[y_{i_j^r}]/(n + s_\alpha - j))$ ▷ Update q_2 removing j from $\mathcal{D}_2$
19: **if** $x^r_{i_j^r} \neq x^r_{i_{j+1}^r}$ **then**
20: $h \leftarrow (x^r_{i_j^r} + x^r_{i_{j+1}^r})/2$ ▷ Compute the location of the split.
21: $q_{\text{new}} \leftarrow q_1 + q_2 + \ln(p(\mathcal{S}_{r,h}))$ ▷ Compute (3) for $\mathcal{S}_{r,h}$.
22: **if** $q_{\text{new}} > q$ **then** ▷ Store the new node with higher log-probability.
23: $q \leftarrow q_{\text{new}}$
24: $\mathcal{N} \leftarrow$ create_sprout(r, h)
25: **end if**
26: **end if**
27: **end for**
28: **end for**
29: Return $\mathcal{N}$
30: **end procedure**

Algorithm 3 Train the GMT given $\mathcal{D}$ and prior hyper-parameters α.

procedure TRAIN($\mathcal{D}, \alpha$)
  $\mathcal{N} \leftarrow$ find_modal_node($\mathcal{D}, \alpha$) ▷ Apply Algorithm 1 or 2.
  if $\mathcal{N}$ is a sprout **then**
    grow_tree($\mathcal{D}, \mathcal{N}, \alpha$)
  end if
  Return $\mathcal{N}$
end procedure

procedure GROW_TREE($\mathcal{D}, \mathcal{N}, \alpha$)
  $\mathcal{D}_1, \mathcal{D}_2 \leftarrow$ split($\mathcal{D}, \mathcal{N}$) ▷ Split $\mathcal{D}$ into $\mathcal{D}_1$ and $\mathcal{D}_2$ using $\mathcal{N}$.
  $\mathcal{N}_1 \leftarrow$ find_modal_node($\mathcal{D}_1, \alpha$) ▷ Apply Algorithm 1 or 2.
  set_left_child($\mathcal{N}, \mathcal{N}_1$) ▷ Set $\mathcal{N}_1$ as the left child of $\mathcal{N}$.
  if $\mathcal{N}_1$ is a sprout **then** ▷ We apply recursion.
    fill_tree($\mathcal{D}_1, \mathcal{N}_1, \alpha$)
  end if
  $\mathcal{N}_2 \leftarrow$ find_modal_node($\mathcal{D}_2, \alpha$) ▷ Apply Algorithm 1 or 2.
  set_right_child($\mathcal{N}, \mathcal{N}_2$) ▷ Set $\mathcal{N}_2$ as the right child of $\mathcal{N}$.
  if $\mathcal{N}_2$ is a sprout **then** ▷ We apply recursion.
    fill_tree($\mathcal{D}_2, \mathcal{N}_2, \alpha$)
  end if
end procedure

by default. We compare the GMT results to DT [6, 19], RF [11, 19], and XG [12]. For reproducibility purposes, we set all random seeds to 0. In the case of RF, we enable bootstrapping to improve its performance. We also fix the number of trees to five for RF and XG. We provide the GMT *Python* module with integration into scikit-learn in [20].

We test the GMT on a selection of data sets from the University of California, Irvine (UCI) database [21]. We compute the accuracy of the DT, RF, XG, and GMT with a shuffled 10-fold cross validation.We do not perform any parameter tuning and keep the same $p(\theta)$ and $p(S)$ for all examples. Accuracy is shown in **Table 1** while training time and node count in **Table 2**.

The results reveal some interesting properties of the GMT. Noticeably, the GMT seems to perform well in general. In all cases, the DT accuracy is lower than the RF accuracy. The only case in which RF considerably outperforms the GMT is with the EEG data set. One reason may be that some information is hidden at the lower levels, i.e. feature correlation information that is hard to extract by looking at only one level ahead. The accuracy difference between GMT and RF indicates that these two techniques may work well for different data sets. Interestingly, the XG and GMT yield similar accuracies. Finally, in most cases the GMT takes more time to train than the other three techniques which is caused by the feature sorting overhead computation. Notably, the node count in **Table 2** shows that we successfully managed to simplify the models while producing similar accuracy. Note that for four of the seven data sets, the average number of nodes is less than ten and produces slightly better accuracies than RF. Ten nodes implies less than five sprouts in average which can be easily analyzed by a human. This highlights the importance of the priors $p(S)$ and $p(\theta)$ to avoid a pruning step. The strength of these two priors will determine how much statistical evidence do we require from our data to produce a meaningful split. In the following **Section 5**, we take a deeper look and explain the reasons behind the GMT construction with a finance application.

4.2 Bayesian Decision Trees and GMT

In this section we analyze the GMT on the Wisconsin breast-cancer data set studied in [9, 14] which is available at the University of California, Irvine (UCI) database [21]. Although this data set contains 699 observations, we are going to use the 683 that are complete. Each observation contains nine features and the outcome classifies the tumor as benign or malignant. We test the GMT for $p(S_0) = 1 - q^{1+\ell}, q \in \{0.75, 0.8, 0.85, 0.90, 0.95, 0.97\}$ and a Dirichlet prior with parameters $(\alpha_1, \alpha_2) \in \{1, 2, 3, 4, 5, 10\}^2$. For each of the 216 parameter sets we perform a 10-fold cross validation and plot the average accuracy in **Figure 3**. The results display a lower average accuracy compared to the 98.4% for BCART [14] with nine or more leaves, and the 96.8% for BART [9]. When we run the methods and parameters from Section 4.1 we obtain 95.3% for DT, 95.8% for RF, and 95.5% for XG. We were unable to compare the BCART and BART performance with the data sets from Section 4.1 due to the lack of software.

5 TRADING EXAMPLE

We consider three stocks, A, B, and C, whose price follows a multidimensional Ornstein-Uhlenbeck process, [22]. Using the

TABLE 1 | Accuracy of DT, RF, XG, and GMT for several data sets. We apply a shuffled 10-fold cross validation to each test. Results are sorted by relative performance, starting from hightest accuracy difference between GMT and RF.

	d	*n*	Accuracy				
			DT [6, 19]	**RF [11, 19]**	**XG [12]**	**GMT**	**GMT – RF**
Credit	23	30,000	72.5%	78.6%	82.0%	82.0%	3.4%
Diabetic	19	1,151	60.4%	63.1%	65.2%	65.3%	2.2%
Heart	20	270	72.6%	78.5%	80.4%	80.4%	1.9%
Seismic	18	2,584	87.8%	91.9%	93.0%	93.2%	1.3%
Haberman	3	306	59.8%	69.6%	69.9%	69.6%	0.0%
Gamma	10	19,020	81.7%	85.9%	86.2%	84.9%	−1.0%
EEG	14	14,980	83.8%	88.1%	80.5%	79.8%	−8.3%

TABLE 2 | Training time in milliseconds and average node count per fold. The node count includes the number of leaves.

	Train time (ms)				Node count			
	DT [6, 19]	**RF [11, 19]**	**XG**	**GMT**	**DT [6, 19]**	**RF [11, 19]**	**XG**	**GMT**
Credit	569.5	334.7	133.6	1,044.2	8,505.2	3,898.9	579.6	43.4
Diabetic	8.7	10.1	15.2	26.8	399.2	182.3	329.6	7.0
Heart	1.4	4.8	20.1	13.2	83.4	44.7	179.6	9.2
Seismic	10.5	11.7	20.5	18.5	410.2	177.7	304.6	6.4
Haberman	1.0	4.1	15.6	1.4	179.8	66.3	194.0	3.2
Gamma	245.2	232.2	92.2	519.1	3,564.0	1,514.9	551.6	111.4
EEG	123.4	101.2	117.9	550.8	2,553.4	1,374.7	472.8	203.4

notation from [22], we can sample the prices by applying the Euler's discretization, $X_{t+\Delta t} = \mu + e^{-\theta \Delta t}(X_t - \mu) + G$. We assume that σ is a unitary matrix, therefore the random vector G follows a normal distribution with mean 0 and covariance $(\theta + \theta^T)^{-1}[I - e^{-(\theta+\theta^T)\Delta t}]$. For this example, we set the parameters to,

$$\mu = \begin{pmatrix} 100 \\ 110 \\ 105 \end{pmatrix}, \qquad \theta = \begin{pmatrix} 4 & -1 & 0 \\ 0.4 & 2 & 0 \\ 0 & 0 & 0.2 \end{pmatrix}, \qquad \Delta t = 0.1. \quad (4)$$

Our goal is to train the GMT to predict the best portfolio configuration. Given that we have three stocks, we consider the following eight buy/sell configurations: +A/−B/−C (buy one stock A, sell one stock B, sell one stock C), −A/−B/−C, −A/+B/−C, +A/+B/−C, −A/−B/+C,+A/−B/+C, -A/+B/+C, +A/+B/+C. At each time step, we take the three stock prices X_{t_i} as inputs. The outcome is defined as the configuration that corresponds to the next price move, i.e. $sign(X^1_{t_{i+1}} - X^1_{t_i})A/sign(X^2_{t_{i+1}} - X^2_{t_i})B/sign(X^3_{t_{i+1}} - X^3_{t_i})C$. For example, if the prices are $(100, 105, 110)$ at t_i and $(110, 100, 120)$ at t_{i+1}, the features are $(100, 105, 110)$, the outcome is +A − B + C, and the profit between t_i and t_{i+1} for this portfolio is $+(110 - 100) - (100 - 105) + (120 - 110)$. Each portfolio configuration is identified to an integer from 0 to 7. We sample 10,000 time steps, train on the first 8,000 observations and test on the next 2,000.

We treat this problem as an eight class classification problem. The GMT is trained with the $p(S)$ from **Section 4** and a Dirichlet conjugate prior $p(\theta)$ with hyper-parameters $(1/8, 1/8, 1/8, 1/8, 1/8, 1/8, 1/8, 1/8)$. To benchmark the results, we train a 10×10. nodes neural network using the MLPClassifier from [19]. During the test phase, our predictions will be the expected modal portfolio configuration: if the input $\boldsymbol{x}$ returns the leaf posterior hyper-parameters α^*, we predict the portfolio $\arg\max_k \{\alpha^*_k / \sum \alpha^*\}$. The test results are shown in **Figure 4**.

The GMT we obtained by training on the first 8,000 observations has only four leaves: if the price of stock A is below 99.96 and the price of stock B below 109.85, we choose + A + B − C; if the price of A is below 99.96 and the price of B above 109.85, we choose + A-B-C; if the price of A is above 99.96 and the price of B below 110.14, we choose −A + B − C; and if the price of A is above 99.96 and the price of B above 110.14, we choose −A − B + C. Although the mean reversion for stock C is not captured in this model, we successfully recovered simple rules to trade the mean reversion of A and B. Since the price of C is more volatile by **Eq. 4**, the current price of C is not enough to recover the mean reversion decision logic. Some filtering of the price of C would allow to capture its mean reversion. In the neural network case, the over-parametrization makes it difficult to recover this simple model.

The deterministic nature of **Algorithm 3** provides a practical framework to explain how was the GMT constructed. We look at each of the nodes to understand how were the modal nodes chosen. The resulting GMT model contains three sprouts—node 0, node 1, node 2—and four leaves—node 3, node 4, node 5, node 6. **Figure 5** shows the log-probability 3) of splitting our data-set at a particular price by stock for the three sprouts. At the root level, node 0, we consider the whole data set. In this case, one can increase the GMT likelihood the most by choosing $S_{0,99.96}$, i.e., splitting the data according to Stock A's price at 99.96. After this node becomes a sprout, the input data is split into two subsets of sizes 3,640 (inputs with Stock A's price below 99.96), and 4,360 (inputs with Stock A's price above 99.96).

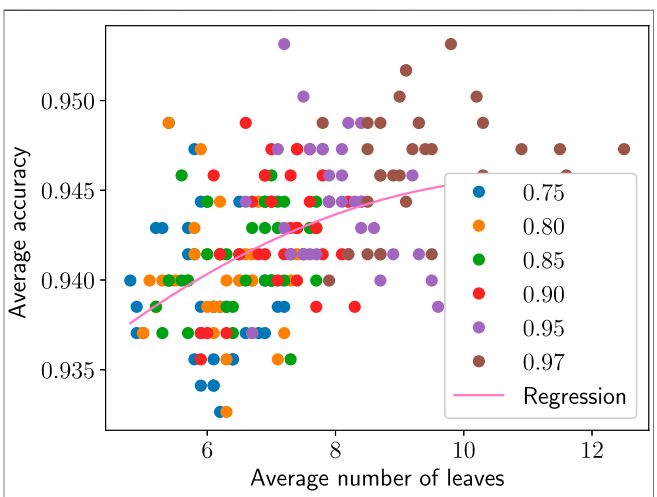

FIGURE 3 | Average accuracy vs. average number of leaves for each GMT parameter set applied to the Wisonson breast-cancer data. The color indicates which parameter $q \in \{0.75, 0.8, 0.85, 0.90, 0.95, 0.97\}$ was chosen. We include a quadratic regression of the results.

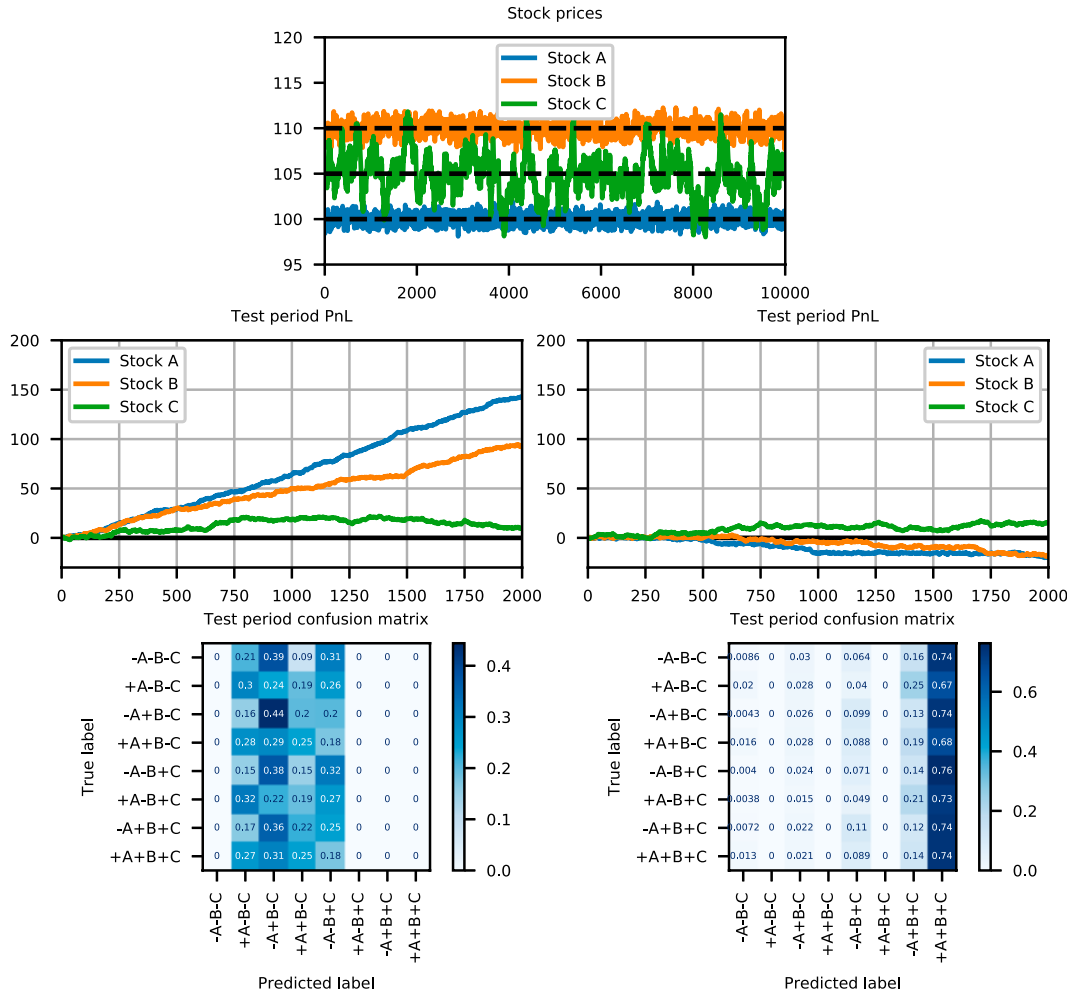

FIGURE 4 | From top to bottom, left to right: Simulated stock prices, test period PnL (Profit and Loss) for the GMT, test period PnL for the neural network, confusion matrix for the GMT, and confusion matrix for the neural network. The PnL is the cumulative profit achieved when the predicted portfolios are executed. The costs are omitted for simplicity.

These two subsets' partition log-probabilities are then shown in node one and node two plots respectively. By looking at the two figures, we conclude we can maximize the log-probability by splitting at Stock's B price 109.85 for node 1, and at Stock's B price 110.14 for node 2. The black horizontal line in each figure marks the log-probability of S_0, i.e.. the stopping condition. When any possible split log-probability is below this line, the node is chosen to be a leaf, as it happens in this example for nodes 3, 4, 5, and 6. Finally, note that by symmetry, the blue, orange, and green lines should look periodic because the extreme splits only separate one input point from the data set. In addition, the green line looks convex which indicates it is better not to split the data based on Stock C's price.

6 DISCUSSION AND FUTURE WORK

The proposed GMT is a deterministic Bayesian Decision Tree that reduces the training time by avoiding any Markov Chain Monte Carlo sampling or a pruning step. The GMT numerical example results show similar accuracies to other known techniques. This approach may be most useful where the ability to explain the model is a requirement. Hence, the advantages of the GMT are that it can be easily understood. Furthermore, the ability to specify $p(\theta)$ and $p(S)$ may be particularly suitable to noisy problems. However, it is not clear whether the hyper-parameters used in the examples are optimal for each data set. Future work will explore the sensitivity and parameter tuning for different prior distributions. It still remains to find a more efficient deterministic way to explore meaningful trees like Markov Chain Monte Carlo based Bayesian Decision Trees do.

As an extension, we would like to assess the performance of this algorithm on regression problems and experiment with larger partition spaces such as the SVM hyperplanes. Another computational advantage not explored is parallelization, which would allow for a more exhaustive exploration of the tree probability space from **Eq. 1**.

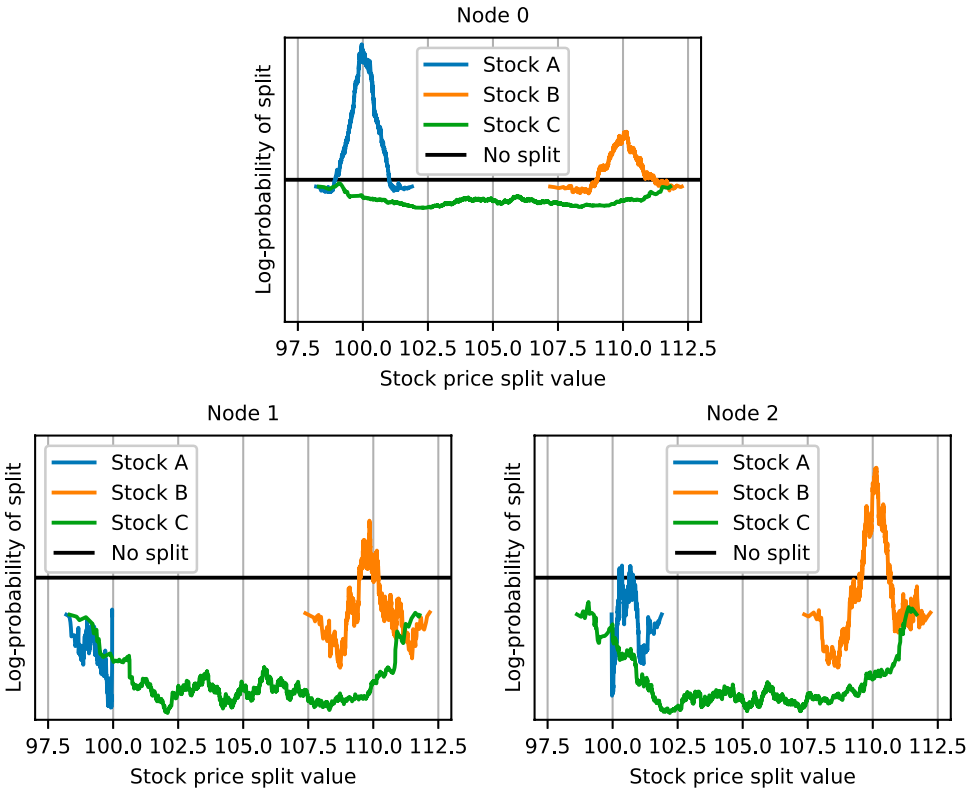

FIGURE 5 | Log-probability of $S_{r,h}$ at each sprout for the trading example. The black horizontal line indicates the S_0 log-probability.

DATA AVAILABILITY STATEMENT

The datasets presented in this study can be found at https://archive.ics.uci.edu/ml/datasets.php. The code is available at https://github.com/UBS-IB/bayesian_tree/.

AUTHOR CONTRIBUTIONS

GN was the technical advisor leading the project. LlAJR was responsible for the technical details, first implementation, running the experiments, and manuscript preparation. AIC contributed in the manuscript preparation.

FUNDING

Authors were employed by UBS during the development of the project. UBS was not involved in the study design, collection, analysis, interpretation of data, the writing of this article or the decision to submit it for publication.

ACKNOWLEDGMENTS

The authors want to thank Kaspar Thommen for the *Python* module implementation with integration to scikit-learn and his suggestions to improve the quality of this project. This manuscript has been released as a pre-print at https://arxiv.org/pdf/1901.03214, [23].

REFERENCES

1. Lipton ZC. The mythos of model interpretability. *Queue* (2018). 16:31–57. doi:10.1145/3236386.3241340
2. Herman B. The promise and peril of human evaluation for model interpretability (2017). Preprint: arXiv: abs/1711.07414.
3. Doshi-Velez F, Kim B. Towards a rigorous science of interpretable machine learning (2017). Preprint: arXiv:1702.08608.
4. Lipton ZC. The doctor just won't accept that! (2017). Preprint: arXiv:1711.08037.
5. Murdoch WJ, Singh C, Kumbier K, Abbasi-Asl R, Yu B. Definitions, methods, and applications in interpretable machine learning. *Proc Natl Acad Sci USA* (2019). 116:22071–80. doi:10.1073/pnas.1900654116
6. Breiman L, Friedman J, Stone C, Olshen R. Classification and regression trees. *The wadsworth and brooks-cole statistics-probability series*. Abingdon, UK: Taylor and Francis (1984).
7. Quinlan JR. *C4.5: programs for machine learning*. San Francisco, CA: Morgan Kaufmann Publishers Inc. (1993).
8. Friedman JH. Greedy function approximation: a gradient boosting machine. *Ann Stat* (2000). 29:1189–232. doi:10.1214/aos/1013203451

9. Linero AR. A review of tree-based Bayesian methods. *Csam* (2017). 24:543–59. doi:10.29220/csam.2017.24.6.543
10. Breiman L. Bagging predictors. *Mach Learn* (1996). 24:123–40. doi:10.1023/A:1018054314350
11. Breiman L. Random forests. *Machine Learn* (2001). 45:5–32. doi:10.1023/A:1010933404324
12. Chen T, Guestrin C. XGBoost: a scalable tree boosting system in *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*. New York, NY: ACM (2016). p. 785–94. doi:10.1145/2939672.2939785
13. Buntine W. Learning classification trees. *Stat Comput* (1992). 2:63–73. doi:10.1007/BF01889584
14. Chipman HA, George EI, McCulloch RE. Bayesian CART model search. *J Am Stat Assoc* (1998). 93:935–48. doi:10.1080/01621459.1998.10473750
15. Denison DGT, Mallick BK, Smith AFM. A Bayesian CART algorithm. *Biometrika* (1998). 85:363–77. doi:10.1093/biomet/85.2.363
16. Chipman HA, George EI, McCulloch RE. Bart: Bayesian additive regression trees. *Ann Appl Stat* (2010). 4:266–98. doi:10.1214/09-AOAS285
17. Schetinin V, Jakaite L, Jakaitis J, Krzanowski W. Bayesian decision trees for predicting survival of patients: a study on the us national trauma data bank. *Comput Methods Programs Biomed* (2013). 111:602–12. doi:10.1016/j.cmpb.2013.05.015
18. Gelman A, Carlin J, Stern H, Dunson D, Vehtari A, Rubin D. *Bayesian data analysis*. 3rd ed.. Boca Raton, FL: Chapman and Hall/CRC Texts in Statistical Science (Taylor & Francis) (2013).
19. Pedregosa F, Varoquaux G, Gramfort A, Michel V, Thirion B, Grisel O, et al.Scikit-learn: machine learning in Python. *J Machine Learn Res* (2011). 12:2825–30. doi:10.5555/1953048.2078195
20. Thommen K, Goswami B, Cross AI. Bayesian decision tree (2019). Available at: https://github.com/UBS-IB/bayesian_tree.
21. Dheeru D, Karra Taniskidou E. *UCI machine learning repository* (2017).
22. Vatiwutipong P, Phewchean N. Alternative way to derive the distribution of the multivariate Ornstein-Uhlenbeck process. *Adv Differ Equ*, 2019 (2019). 2019. doi:10.1186/s13662-019-2214-1
23. Nuti G, Jiménez Rugama LlACross A-I. A Bayesian decision tree algorithm (2019). Preprint: arXiv: abs/1901.03214

Conflict of Interest: The authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.