A very simple safe-Bayesian random forest

Article (Accepted Version)

A Very Simple Safe-Bayesian Random Forest

Novi Quadrianto and Zoubin Ghahramani

Abstract—Random forests works by averaging several predictions of de-correlated trees. We show a conceptually radical approach to generate a random forest: random sampling of many trees from a prior distribution, and subsequently performing a weighted ensemble of predictive probabilities. Our approach uses priors that allow sampling of decision trees even before looking at the data, and a power likelihood that explores the space spanned by combination of decision trees. While each tree performs Bayesian inference to compute its predictions, our aggregation procedure uses the power likelihood rather than the likelihood and is therefore strictly speaking not Bayesian. Nonetheless, we refer to it as a Bayesian random forest but with a built-in safety. The safeness comes as it has good predictive performance even if the underlying probabilistic model is wrong. We demonstrate empirically that our Safe-Bayesian random forest outperforms MCMC or SMC based Bayesian decision trees in term of speed and accuracy, and achieves competitive performance to entropy or Gini optimised random forest, yet is very simple to construct.

Index Terms—Bayesian methods, random forest, decision trees

1 Introduction

Decision trees [1] represent a classical induction method that is widely used. This is because decision trees exhibit many appealing properties: they are simple to understand and interpret, applicable for both classification and regression tasks, and offer good predictive performance. Although defining a global objective to optimally learn decision trees is fundamentally hard, there are several approaches to learn trees based on Gini impurity [1] and information-theoretic considerations [2], [3].

Breiman [4] used an ensemble of *unpruned Gini-optimised* decision trees to improve the performance of learning. This random forest framework has become a very popular and powerful tool with applications ranging from machine learning, computer vision, computer graphics, to medical image analysis, among others [5], [6]. For a recent comprehensive survey about random forests, refer to [7]. The randomness is introduced during training phase of the trees via: a) random sampling of the training dataset [4], and b) partitioning the data space with only randomised subsets of data features [8]. We will call trees trained with this procedure as randomly trained trees. An important ingredient of random forest is the composition of trees that are randomly different from one another. This results in de-correlated individual tree predictions and, therefore, in an improved generalisation. The notion of randomness helps the model to be robust with respect to noisy data.

Reflecting on how a random forest is built from randomly trained trees, we ask the following natural question: can we simply *sample many trees* from some prior distributions, and perform an *ensemble* of those randomly sampled trees? The Bayesian framework is a principled way to achieve this. We note that the usage of Bayesian statistics for learning decision trees has a long history of successful methods, among others, [9], [10], [11], [12], [13], [14].

Bayesian approaches start by defining a prior distribution on the space of decision trees. Subsequently, a likelihood function, for example Gaussian (for regression tasks) or Bernoulli (for classification tasks), is evaluated within each data block induced by the decision tree. The likelihood describes the conditional distribution of the output data given input data falling into the corresponding block. In *all* previous models using Bayesian statistics, the prior distribution of the decision tree is defined conditionally on the given input data. Sophisticated Markov Chain Monte Carlo (MCMC) methods [11] or sequential Monte Carlo [14] is then used to approximate intractable posterior computations. In this paper, we will show a novel usage of a prior that allows us to sample decision trees even before looking at the data.

As more data arrives, Bayesian decision trees will put a mass to a single tree. This is a desired behaviour of Bayesian model averaging [15]. We are instead interested to explore ensemble of trees in the sense of model combination [16]. We achieve this by borrowing the concept of power likelihood [17], [18]. By utilising a data independent prior coupled with the power likelihood, we deliver fast yet *safe* Bayesian random forests that exceed the performance of Bayesian decision trees and are competitive with random forests and SVMs. We use the notion of Safe-Bayesian from [19] in a reference to procedures based on a so-called β-Bayesian posterior (power likelihood

• *N. Quadrianto is with SMiLe CLiNiC, Department of Informatics, University of Sussex, UK. E-mail: n.quadrianto@sussex.ac.uk*
• *Z. Ghahramani is with Department of Engineering, University of Cambridge, Cambridge, UK. Email: zoubin@eng.cam.ac.uk*

Manuscript received May 2013.

with the exponent β) [20] that still perform well even if the underlying probability model is wrong. The paper is organised as follows: Section 2 describes tree priors, likelihoods, and posteriors which form the basis of our Bayesian random forest formulation. Section 3 describes the predictive distribution of our model. We show how to construct an ensemble of decision trees in Section 4. Section 5 provides experiments on several binary and multi-class classification problems. Finally, Section 6 concludes the paper.

2 The Model

Here we describe our model of Safe-Bayesian random forest. Assume that we are given input $X = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$ with $\mathbf{x}_i \in \mathcal{X} = \mathbb{R}^D$ and output $Y = \{y_1, \ldots, y_N\}$ with $y_i \in \mathcal{Y} = \{1, \ldots, C\}$. The goal is to infer a function $\mathcal{F} : \mathcal{X} \to \mathcal{Y}$ that maps inputs to outputs. In this paper, we only focus on the classification tasks. We are interested in the case where the latent function is an *ensemble* of predictors, where each predictor partitions the input data space into axis-aligned blocks. Note that the partitioning can be represented graphically as a *decision tree*.

To elaborate the form of our Safe-Bayesian random forest model, we begin by establishing notation and model for a single tree. Let T be a rooted and strictly binary tree; a tree that has a single root node, internal nodes with exactly two outgoing edges (also known as children), and terminal (leaf) nodes. Each node of the tree $v \in T$ corresponds to a block $B_v \subset \mathbb{R}^D$, and has a splitting rule applied on B_v. The rule splits the block B_v into two halves: B_v^l and B_v^r associated with the left and right child nodes, respectively. The splitting rule is characterised by the dimension of the split, $\kappa_v \in \{1, \ldots, D\}$, and the location of the split (also known as cut value), τ_v. Thus, we have $B_v^l = B_v \cap \{\mathbf{x} \in \mathbb{R}^D : x_{\kappa_v} \le \tau_v\}$, and $B_v^r = B_v \cap \{\mathbf{x} \in \mathbb{R}^D : x_{\kappa_v} > \tau_v\}$. Note that the root node has a block $B(\texttt{root}) = \mathbb{R}^D$, and terminal nodes do not have splitting rules associated to them. Each terminal node however describes conditional distribution of the labels given all input data falling into that node. We will denote a decision tree with tree structure, split dimensions, and split values as $\mathcal{T} = \{T, \kappa, \tau\}$. Our Bayesian analysis proceeds by specifying a prior probability distribution on the space of decision trees, that is $p(\mathcal{T})$.

2.1 Tree Prior $p(\mathcal{T})$

In defining a prior on decision trees, we note that the dimension of the split κ, and the location of the split τ serve as an index for the splitting rule of each T. It is then more convenient to exploit the relationship that $p(T, \kappa, \tau) = p(\kappa, \tau|T)p(T)$, and specify $p(\kappa, \tau|T)$ and $p(T)$ separately. We discuss the specifications of $p(T)$ and $p(\kappa, \tau|T)$ in the subsequent sections.

A generative process of trees structure $p(T)$

The following process describes the structure of trees T. Let 0 be the code for a terminal or *leaf* node, and 1 be the code for an *internal* node. The latter will be followed by the 2 code strings for the 2 children. We assume that 1s occur independently with some fixed probability λ. Further, we consider that the tree structure is formed when number of terminal nodes is equal to number of internal nodes + 1. The tree codes are generated in depth-first order starting from 1 at the root node. Refer to Figure 1 for examples of tree coding, and to Figure 2 for samples of different trees structures.

A prior on cut dimensions and values $p(\kappa, \tau|T)$

Given the structure of the tree, we will then need to specify the splitting rules for each of the internal nodes of the tree. For internal node v, the dimension κ_v and location τ_v of the cut are chosen uniformly from $\{1, \ldots, D\}$ and $[0, 1]$, respectively. That is

$$\kappa_v \sim \mathcal{U}(\{1, \ldots, D\}) \tag{1}$$

$$\tau_v \sim \mathcal{U}([0, 1]). \tag{2}$$

We have assumed that the input features at each dimension lie in $[0, 1]$. We will describe in Section 5 on how to enforce this.

Note that our prior on the space of decision trees defined above is *independent* of the data X. Once we generate the structure of the tree, and fill up the internal nodes with splitting rules, we have completely specified the decision tree. This is in contrast to existing work on Bayesian decision trees, for example [11], [13], [14]. Under the Chipman et al. model that is the model used in these existing works, the probability that a node v is split into two children is $\alpha_s/(1+|v|)^{\beta_s}$ with the parameters $\alpha_s \in (0, 1)$ and $\beta_s \in [0, \infty)$ governing the shape of the resulting tree, and $|v|$ is the depth of the node. For larger α_s and β_s the typical trees are larger, while deeper v is in the tree the less likely it will be split. This is in fact a data-independent tree structure prior. However, as noted in [14], given the data at a particular internal node, the split dimension and location are sampled from the uniform distribution over the available data features and data values. Therefore, for all Bayesian decision trees, the probability of tree growing, and the distribution of the split dimension and location, depend on X so that every node in the tree will contain at least one data point. [14] put forward the theoretical need for data independent priors over the space of decision trees that include tree structures, cut dimensions and values to make the model coherent with respect to changing dataset sizes. On the practical side, the effect of data independent prior enables us to use exactly the same trees structures in all our experiments. We have just defined a way to partition the data, the next required thing is to define a conditional distribution of the labels given input data falling into corresponding blocks. This is described in the next section.

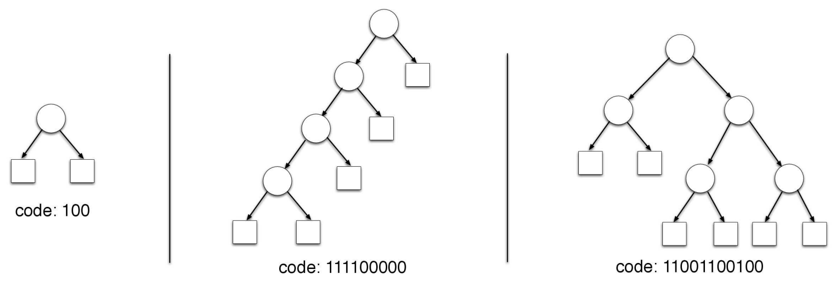


Fig. 1: Coding of a rooted and strictly binary tree. Internal nodes are denoted with circles and terminal nodes with squares. Each internal node has exactly two outgoing edges. The code for an internal node is 1 and 0 is for a terminal node. Codes are generated in depth-first order starting from 1 at the root node.

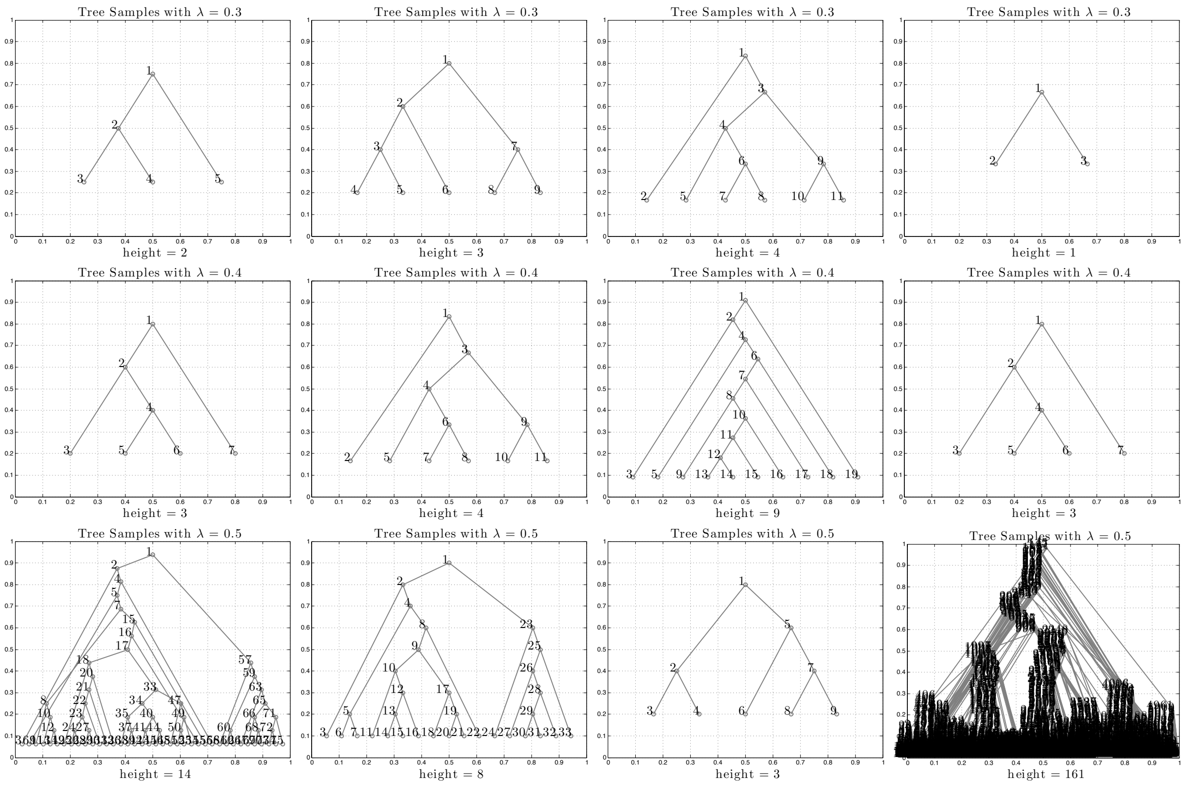


Fig. 2: Sampling trees structures from the prior at $\lambda \in \{0.3, 0.4, 0.5\}$. Once we generate the structures, and fill up the internal nodes with splitting rules, we will completely specify decision trees.

2.2 Dirichlet-Multinomial Tree Likelihood $p(Y|X, \mathcal{T})$

To specify a predictive model within each block, we follow the Bayesian model of [9], [11]. We will repeat them here for the self consistency of the paper. For an input vector $\mathbf{x}$ that falls into the ν-th terminal node, we assign a parametric family f indexed by $\boldsymbol{\theta}_\nu$, $f(y|\boldsymbol{\theta}_\nu)$, that models the conditional distribution $y|\mathbf{x}$. For a classification problem where the output y belongs to one of C classes $\{1, \ldots, C\}$, the natural choice for $f(y|\boldsymbol{\theta}_\nu)$ is a multinomial distribution with conjugate Dirichlet prior on terminal node parameters. Given a block partitioning and a predictive model within each partition, we assume that y values within a terminal node are *i.i.d.* given $\boldsymbol{\theta}_\nu$ and y values across terminal nodes are independent. Thus the form of class conditional distribution at a particular node v for a specific decision tree $\mathcal{T}$ is $f(Y_{N_v}|X_{N_v}, \boldsymbol{\theta}, \mathcal{T}) = \prod_{i \in N_v} \prod_{c=1}^{C} \theta_c^{\mathbb{I}[\ y_i = c\ ]}$, where $\boldsymbol{\theta} = (\theta_1, \ldots, \theta_C)$, $\boldsymbol{\theta}^\top \mathbf{1} = 1$, and $\theta_c \geq 0$ for all $c \in \{1, \ldots, C\}$. As well, for simplicity of presentation, we have denoted as N_v the set of indices of the input vectors that fall into partition block B_v, that is $N_v = \{i : \mathbf{x}_i \in B_v\}$. Thus, X_{N_v} and Y_{N_v} are input vectors and their labels in block B_v, respectively. In the above, we make use of Iverson's bracket notation: $\mathbb{I}[P] = 1$ for the condition P is true and it is 0 otherwise. We put a conjugate Dirichlet prior on $\boldsymbol{\theta}$ with parameter $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_C)$, $\alpha_c > 0$, and assume conditional independence of parameters across terminal nodes. Thus, we have the following likelihood: $p(Y|X, \mathcal{T}) = \int \prod_{v \in \Omega_T} f(Y_{N_v}|X_{N_v}, \boldsymbol{\theta}, \mathcal{T}) p(\boldsymbol{\theta}|\mathcal{T}) d\boldsymbol{\theta}$. We

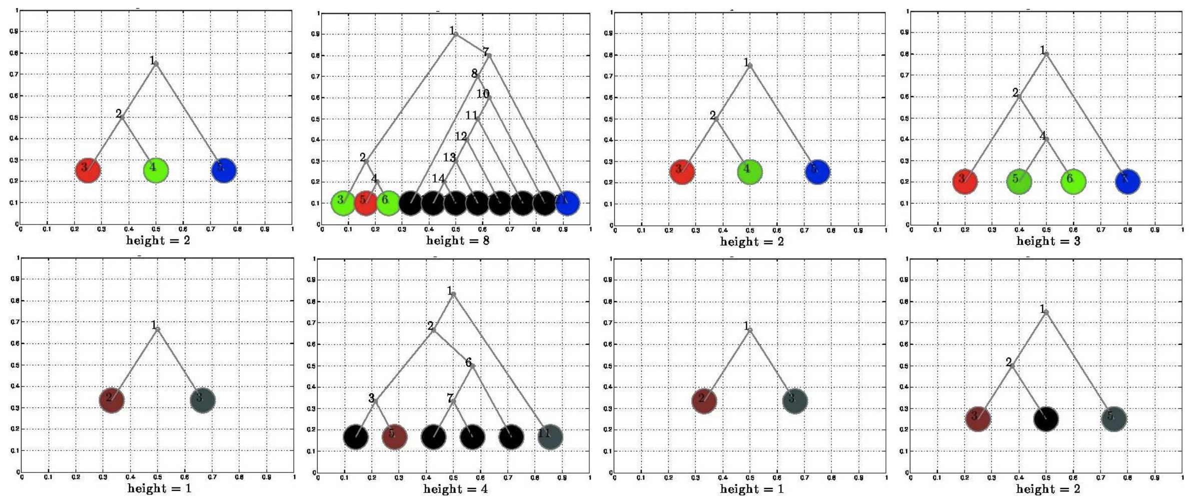


Fig. 3: A visualisation of 4 trees with the highest (top row) and 4 trees with the lowest (bottom row) importance weights for a 3-class problem 'iris' dataset (best viewed in colour). The *terminal* nodes are colour encoded according to the label proportions of data points falling into the terminal nodes. The three colours, red, green, and blue, correspond to the three class labels. More homogeneous labels in the terminal node correspond to purer colour encoding. The terminal nodes with mixed class labels produce colour in RGB space. Some of the terminal nodes are coloured black as no data point falls into those particular nodes. The black internal nodes will not contribute to the importance weight of the tree as the associated terms in Equation (3) are 1. Essentially, this allows us to have an in-built pruning of the trees structures.

have used Ω_T to denote terminal nodes of decision tree $\mathcal{T}$. The above integration can be done analytically using standard properties of the Dirichlet distributions (see for example [21]), we have

$$p(Y|X,\mathcal{T}) = \prod_{v\in\Omega_T} \frac{\Gamma(\sum_{c=1}^C \alpha_c)\prod_{c=1}^C \Gamma(m_{v,c}+\alpha_c)}{\prod_{c=1}^C \Gamma(\alpha_c)\Gamma(\sum_{c=1}^C (m_{v,c}+\alpha_c))}. \tag{3}$$

In the above, the symbol $m_{v,c}$ denotes the number of labels $y_i = c$ among those $i \in N_v$. As noted by [11], for a given tree, the value $p(Y|X,\mathcal{T})$ will be larger whenever more homogeneous values of y are assigned to the terminal nodes. Further, a simple choice of $\boldsymbol{\alpha}$ is the vector $(1,\ldots,1)$ for which the Dirichlet prior is the uniform, unless further prior information is available.

2.3 Tree Posterior $p(\mathcal{T}|X,Y)$

We can compute the posterior distribution of the latent decision tree $\mathcal{T}$ given the observed data, up to a normalisation constant, by combining the likelihood in (3) and the tree generating prior described in Section 2.1. We have the following: $p(\mathcal{T}|X,Y) \propto p(Y|X,\mathcal{T})\times p(\mathcal{T})$. Like all other Bayesian tree models, exact computation of the posterior is computationally intractable. To overcome this, several Markov Chain Monte Carlo methods have been proposed, particularly a Metropolis-Hastings (M-H) [11], [13], and a Sequential Monte Carlo (SMC) [14]. In this paper, we choose to use a very simple method based on importance sampling [22].

3 Prediction and Computation of Probabilities in Test Data

For a *previously unseen* test point $\mathbf{x}^* \in \mathbb{R}^D$, the predictive distribution over the latent output y^* can be computed as follows:
$p(y^* = c|\mathbf{x}^*, X, Y)$

$$\begin{aligned} &= \int p(y^*=c|\mathcal{T},\mathbf{x}^*,X,Y)p(\mathcal{T}|\mathbf{x}^*,X,Y)d\mathcal{T} \\ &= \int p(y^*=c|\mathcal{T},\mathbf{x}^*,X,Y)\frac{p(\mathcal{T}|X,Y)}{p(\mathcal{T})}p(\mathcal{T})d\mathcal{T} &(4)\\ &\approx \sum_k p(y^*=c|\mathcal{T}^{(k)},\mathbf{x}^*,X,Y)\frac{p(\mathcal{T}^{(k)}|X,Y)}{p(\mathcal{T}^{(k)})} &(5)\\ &\approx \sum_k p(y^*=c|\mathcal{T}^{(k)},\mathbf{x}^*,X,Y)\underbrace{p(Y|X,\mathcal{T}^{(k)})}_{w_k}, &(6)\end{aligned}$$

where

$$\begin{aligned} p(y^*=c|\mathcal{T}^{(k)},\mathbf{x}^*,X,Y) &= \mathbb{E}[\theta_c|\mathcal{T}^{(k)},\mathbf{x}^*,X,Y] \\ &= \frac{m_c+\alpha_c}{\sum_{c=1}^C (m_c+\alpha_c)}. \end{aligned} \tag{7}$$

The importance weight w_k in Equations (6) and (3) will be larger for a decision tree $\mathcal{T}$ with more homogeneous values of y. Equation (4) of the above is an *importance sampling* method with a prior proposal distribution $p(\mathcal{T})$. We use importance sampling because of the following reasons: it is simple to implement and importantly it exploits the strength of our usage of data independent tree priors. However, we note that from a Bayesian perspective, importance sampling using the prior as the proposal is known

Algorithm 1 *Training* Phase of Simple Bayesian Random Forest

```
Input number of trees $K$, parameter value $\lambda$
Generate $K$ tree structure from the prior distribution $p(T)$
Input number of dimensions $D$
Fill up the internal nodes with splitting rules according to: $\kappa_v \sim \mathcal{U}(\{1,\ldots,D\})$, and $\tau_v \sim \mathcal{U}([0,1])$
Input parameter vector $\boldsymbol{\alpha}$ and paired input-output training data $\{(\mathbf{x}_1,y_1),\ldots,(\mathbf{x}_N,y_N)\} \subset \mathbb{R}^D \times \mathcal{Y}$
for $k = 1$ to $K$ do
  Compute the importance weight $w_k$ of the decision tree according to $p(Y|X,\kappa,\tau)$
end for
Generate operating curve based on power likelihood [18] by varying the value of $\boldsymbol{\beta}$ from $1/N, 2/N, \ldots, 1$
Return many randomly sampled decision trees $\{T^{(k)},\kappa^{(k)},\tau^{(k)}\}_{k=1}^{K}$ with associated importance weights $\{w_k\}_{k=1}^{K}$ and a forest operating curve
```

to work not so well. Nevertheless, the prior proposal offers a better predictive accuracy versus computational time tradeoff than any potentially more optimal proposal. The same observation is also made in [14]. And, Equation (7) is conditioned on the partition v that $\mathbf{x}^*$ is in. To get a point estimate $\hat{y}^*$ from the predictive distribution $p(y^* = c|\mathbf{x}^*, X, Y)$, we perform $\hat{y}^* = \arg\ \max_c p(y^* = c|\mathbf{x}^*, X, Y)$.

4 Model Combination via Power likelihood

It is important to note that in the limit of $N \to \infty$, Bayesian model averaging (BMA) in (6) will put a mass to a single decision tree [15], [16]. Instead we want to explore the space spanned by *combination* of several decision trees. We achieve this model combination by borrowing the concept of power likelihood [17], [18], that is raising the likelihood function to a power between 0 and 1. Historically a motivation of introducing the power likelihood is to guarantee posterior consistency with simply a trivial Kullback-Leibler neighbourhoods condition on the prior [17], [18]. There are other reasons for undertaking inference with power likelihood. [23] use power likelihood to *combine* historical data from similar studies in constructing the prior distributions for regression models. It has also been used to do model selection via marginal likelihood as in [24]. Recently, [25] explore related ideas of tempering the distributions by powering them up to speed up the convergence of MCMC. Empirically we will also show that model combination decision using the power likelihood greatly outperforms Bayesian model averaging of trees.

Specifically, when averaging, we use the marginal likelihood to the power $\beta < 1$. A sensible choice would be to use $\beta = m/N$ where m is the 'effective sample size' used in the averaging. By dividing by the size of the dataset N, we ensure that we do not get domination by a single ensemble member as the size of the training set increases. For $\beta = 1$ we will recover BMA.

To summarise, our Bayesian random forest method involves two phases: training and test phases. The *training* phase involves sampling trees structures, and their associated splitting rules from the prior distributions. This process can be performed offline. Once data arrive, we compute the importance weights via (marginal) likelihood in Equation (3). Subsequently, based on the power likelihood, we generate the operating curve of Bayesian random forest by varying the value of $\boldsymbol{\beta}$ from $1/N, 2/N, \ldots, N/N$ [18] (see Figure 4 for examples of such curves). We generate the curve based on training data. We can then choose the operating value of β based on a small validation set. For this work, we choose the value of β based on the training set. In the *test* phase, given a previously unseen data point, each decision tree hierarchically applies a number of splitting rules. Starting at the root, each internal node applies its associated split function to the new data point, and this process is repeated until the data point reaches a terminal node. At the terminal node, we compute the class predictive distribution using Equation (7). We repeat this procedure for all the trees in the forest, and the final forest prediction will be a weighted combination of individual trees prediction. Refer to Algorithm 1 for a training phase pseudo-code, and Algorithm 2 for a test phase.

5 Experiments

Datasets We use the following nine UCI[1] datasets: australian, breast-cancer, diabetes, heart, ionosphere, german.numer, iris, wine, and svmguide2, and two high dimensional image data from Israeli Image[2] and Animals with Attributes (AwA)[3] datasets. We use $4,097$ dimensional Fisher [26] representations for Israeli Image, and $2,000$ dimensional SURF [27] representations for AwA.

Algorithms We compare the performance of our Bayesian random forest with markov chain monte carlo (MCMC) and sequential monte carlo (SMC) based Bayesian decision trees[4], random forest, kNN,

1. http://archive.ics.uci.edu/ml/
2. http://people.cs.umass.edu/~ronb/image_clustering.html
3. http://attributes.kyb.tuebingen.mpg.de
4. http://www.gatsby.ucl.ac.uk/~balaji/treesmc/ [14]

Algorithm 2 *Test* Phase of Simple Bayesian Random Forest

```
Input previously unseen test data $\{(\mathbf{x}_1^*),\dots,(\mathbf{x}_{N'}^*)\} \subset \mathbb{R}^D$, a set of decision trees $\{T^{(k)},\kappa^{(k)},\tau^{(k)}\}_{k=1}^K$ with associated weights $\{w_k\}_{k=1}^K$ and an operating value $\beta$
for $i = 1$ to $N'$ do
  for $k = 1$ to $K$ do
    Compute the predictive distribution at a particular decision tree $\mathcal{T}^{(k)}$, that is $p(y_i^*|\mathcal{T}^{(k)},\mathbf{x}_i^*,X,Y)$
  end for
  Compute the predictive distribution of a forest, that is $p(y_i^*|\mathbf{x}_i^*,X,Y) = \sum_{k=1}^K w_k^\beta p(y_i^*|\mathcal{T}^{(k)},\mathbf{x}_i^*,X,Y)$
end for
Return forest predictive distribution $p(y_1^* = c|\mathbf{x}_1^*,X,Y),\dots,p(y_{N'}^* = c|\mathbf{x}_{N'}^*,X,Y)$ for all $c = \{1,\dots,C\}$
```

and SVM. For the last three methods we use scikit-learn implementations[5]. For `MCMC` and `SMC`, we fix the hyperparameters to $\alpha_s = 0.95$ and $\beta_s = 0.5$ as suggested in [14]. We set the number of iterations to be $100,000$ for MCMC. We consider two types of proposal distribution for SMC, that is prior proposal (`SMC: prior`) and optimal proposal (`SMC: post.`). For `RF`, we set minimum number of samples required to split an internal node to be 1. We investigate two widely used criteria to measure the quality of the split: Gini impurity (`RF: Gini`) and entropy (`RF: Ent.`). We set the number of features to consider when looking for the best split to be $0.5D$ where D is the dimension of the data. For `kNN`, we set the number of nearest neighbours to be 3, and for `SVM` we use a linear kernel with cross-validated regularisation parameter. For our `S-Bayes RF`, the parameter λ to be 0.475 [6], and we use a symmetric uniform Dirichlet prior. Our potential cut dimensions are chosen uniformly from $\{1,\dots,D\}$. For our S-Bayes RF, we use exactly the same trees structures for all 11 datasets[7]. We use $1,000$ trees for random forest, Bayesian random forest, and for number of particles in SMC.

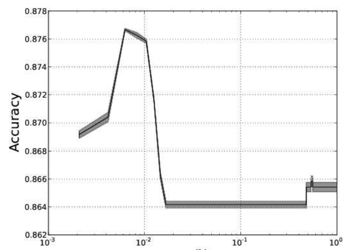
Preprocessing We use the probability integral transform (PIT) to transform the input features at each dimension $d \in \{1,\dots,D\}$ to lie in $[0,1]$. PIT allows us to generate uniform random variables out of any continuously distributed random variables by making use of the empirical estimate of the cumulative distribution function underlying our data. This transformation makes the distance between adjacent data points to be $1/(N+1)$, thus has the added potential to reduce the effect of noisy data points (outliers).

Results We use 80% of the available data to be a training set and the remaining 20% as a test set. We assess the performance on the test set. The whole experiment was then repeated 5 times to obtain confidence bounds. In Figure 3, we provide a visualisation of 4 trees with the highest importance weights w_k in Equation (6),(3) and 4 trees with the lowest importance weights for a 3-class problem 'iris' dataset. We encode the terminal nodes with colour according to the label proportions of data points falling into each of the terminal nodes. The more label homogeneous is the terminal node, the more red or green or blue is the colour encoding. Note also that some of the terminal nodes are coloured black due to no data point falling into those particular nodes (this is a property unique to our usage of data independent prior on decision trees). However, these black internal nodes will not contribute to the importance weight of the tree as the associated terms in Equation (3) equal to 1. We generate, in Figure 4, the operating curve of our method by varying the value of $\boldsymbol{\beta}$ from $1/N, 2/N, \dots, 1$. It is clear from the curves that Bayesian model averaging (the point where $m/N = 1$) is suboptimal, except for small 'iris' dataset. For all datasets, effective sample size $m = 5$ seems to be sufficient. The experimental results for UCI datasets are summarised in Table 1. In *all but two* datasets ('diabetes' and 'iris'), proposed S-Bayes RF method outperforms Bayesian decision trees (MCMC and SMC variants), sometimes by a large margin. We credit this to our usage of power likelihood that does not assume a single decision tree as a good hypothesis. This fact is corroborated in Table 2 where our weighted ensemble of trees always outperforms a single tree with the highest importance weight and BMA (when $\beta = 1$). Even though Bayesian random forest consists of randomly sampled trees, it is competitive to the random forest, where each tree is built with respect to some goodness criteria, in this case Gini impurity and entropy. However, Table 3 shows the drawback of our usage of power likelihood that it does not produce a well-calibrated predictive probabilities. We also assess the sensitivity of S-Bayes RF with the choice of number of trees and the splitting probability λ. The results for 4 representative datasets are visualised in Figure 5. For completeness, we also provide results of Bayesian Additive Regression Trees (BART) [13] using BayesTree package for R in Table 1–second to last column. We note that BART can be thought of as a Bayesian version of boosted decision trees. BayesTree package uses a probit link function for handling a binary classification problem. Our results are consistent with previous studies [29] that found calibrated boosted trees were the best learning algorithm on several considered binary learning

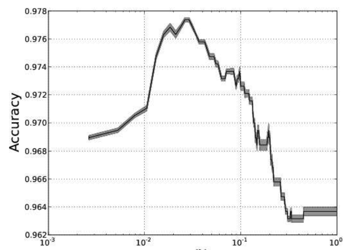
5. http://scikit-learn.org/stable/ [28]
6. We recommend the use of $\lambda \leq 0.5$ for efficiency reason.
7. We will provide the tree structures and the source code.

TABLE 1: Accuracy across 5 random repeats. D: number of dimensions and N: number of observations, equal from each class. `kNN`: k-nearest neighbours ($k = 3$), `SVM`: Support vector machine with linear kernels, `RF: Gini`: Random forest learned with Gini impurity, `RF: Ent.`: Random forest learned with information gain, `MCMC`: Markov chain monte carlo of Bayesian decision tree, `SMC: pr.`: Sequential monte carlo of Bayesian decision tree with prior proposal, `SMC: ps.`: Sequential monte carlo of Bayesian decision tree with posterior proposal, `BART`: a Bayesian version of boosted decision trees, and `S-Bayes RF (Ours)`: our Safe-Bayesian random forest.

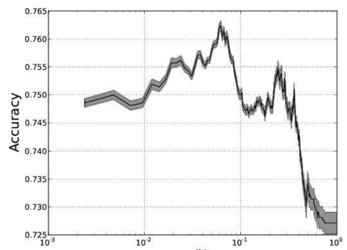
	D	N	`kNN`	`SVM`	`RF: Gini`	`RF: Ent.`	`MCMC`	`SMC: prior`	`SMC: post.`	`BART`	`S-Bayes RF`
`australian`	14	618	83.04±2.53	86.23±0.89	86.23±2.29	86.08±2.82	86.09±0.61	86.23±1.02	86.23±0.51	**87.68±0.88**	87.39±1.50
`breast-cancer`	10	517	98.39±1.25	**98.54±1.15**	98.10±1.42	98.10±1.42	96.93±1.58	96.35±2.42	96.49±1.66	98.39±1.40	97.95±1.30
`diabetes`	8	574	72.59±1.32	75.32±4.68	74.80±2.61	74.15±3.13	74.42±2.08	71.68±1.63	73.76±2.08	**76.75±5.08**	74.28±4.76
`heart`	13	270	81.11±5.54	85.18±2.62	82.96±4.01	83.33±4.34	80.37±6.23	79.63±5.86	81.48±5.71	85.56±3.56	**87.41±5.61**
`ionosphere`	34	270	81.99±3.33	81.99±6.59	90.85±3.58	**91.14±3.41**	89.43±3.13	88.85±2.93	89.71±3.41	**92.00±3.28**	91.14±2.11
`german.numer`	24	680	63.80±5.81	72.09±4.04	69.29±3.47	68.99±3.18	63.80±4.80	63.59±6.25	64.10±5.97	71.80±3.78	**72.40±2.88**
`iris`	4	60	95.99±5.33	**98.00±2.98**	96.67±3.33	96.67±3.33	96.67±3.33	97.33±2.78	95.99±3.65	NA	96.67±3.33
`wine`	13	178	97.22±1.76	97.22±3.40	98.33±2.48	98.33±2.48	95.56±1.52	86.67±9.89	94.44±1.96	NA	**99.44±1.24**
`svmguide2`	20	159	60.61±2.71	**69.70±7.72**	69.09±4.97	67.87±7.90	63.64±9.34	55.15±11.81	62.42±5.90	NA	67.27±4.97



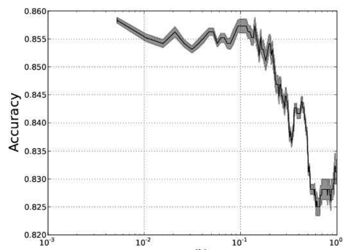
(a) `australian`



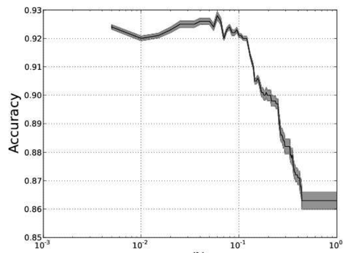
(b) `breast-cancer`



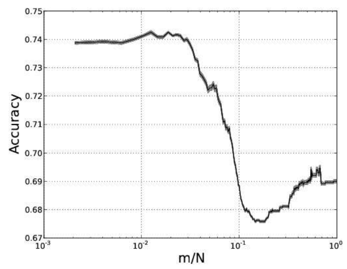
(c) `diabetes`



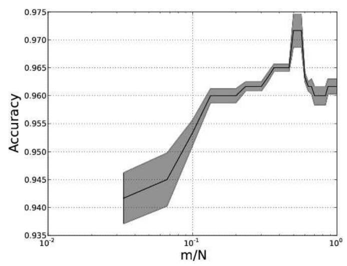
(d) `heart`



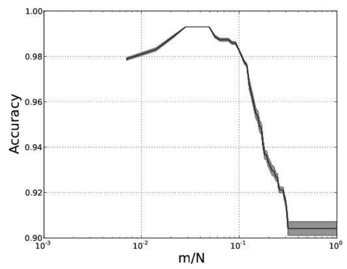
(e) `ionosphere`



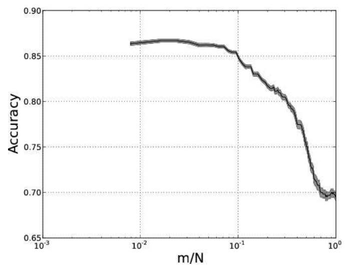
(f) `german.numer`



(g) `iris`



(h) `wine`



(i) `svmguide2`

Fig. 4: Operating curve of our Bayesian random forest based on the concept of power likelihood. Accuracy±STE.. The curves are generated based on training data. The point where $m/N = 1$ corresponds to Bayesian model averaging (BMA).

TABLE 2: Performance comparison of Bayesian random forest: tree with the highest importance weight (`Best Tree`), weighted ensemble of trees with $\beta = 1$ (Bayesian Model Averaging–`BMA`), weighted ensemble of trees with $\beta = 5/N$ (`S-Bayes RF`). Accuracy±STD. Ensemble *always* improves performance, sometimes dramatically.

	Best Tree	BMA	S-Bayes RF
australian	86.23±0.79	86.23±0.89	87.39±1.50
breast-cancer	96.06±1.35	96.64±0.40	97.95±1.30
diabetes	71.42±2.53	70.39±3.45	74.28±4.76
heart	75.93±6.83	77.78±8.97	87.41±5.61
ionosphere	81.99±5.07	81.71±5.84	91.14±2.11
german.numer	63.20±5.71	63.70±6.02	72.40±2.88
iris	94.00±4.89	94.67±6.05	96.67±3.33
wine	86.67±5.93	88.33±4.56	99.44±1.24
svmguide2	51.51±5.42	53.94±8.41	67.27±4.97

TABLE 3: Log predictive probabilities of Markov chain Monte Carlo of Bayesian decision tree (`MCMC`) and weighted ensemble of trees with $\beta = 5/N$ (`S-Bayes RF`). Log probability±STD. As expected, in all cases MCMC provides better calibrated probabilities than S-Bayes RF with power likelihood.

	MCMC	S-Bayes RF
australian	-0.3330±0.0249	-0.5409±0.0043
breast-cancer	-0.0999±0.0336	-0.3009±0.0129
diabetes	-0.5277±0.0149	-0.6260±0.0061
heart	-0.4464±0.0560	-0.5879±0.0053
ionosphere	-0.3150±0.0605	-0.6097±0.0095
german.numer	-0.5885±0.0281	-0.6759±0.0023
iris	-0.2616±0.1284	-0.5473±0.0608
wine	-0.2475±0.0581	-0.6625±0.0275
svmguide2	-0.8702±0.0858	-1.0283±0.0133

problems but random forests are close second. In 3 datasets, BayesTree outperforms S-Bayes RF by more than 0.4% while only in 2 datasets it was the opposite.

The results on high dimensional image data are summarised in Table 4. Our results are once again consistent with [29] that found it surprising that random forest variants perform well even for high dimensional data. For this high dimensional data, we use effective sample size of one. Among trees and forest variants, our S-Bayes RF is the fastest to train, while MCMC based Bayesian decision tree is the most time consuming. We also run the random forest variant where a random subset of candidate features is used plus cut values are drawn at random for each candidate feature and the best of these randomly-generated thresholds is picked as the cut value [30], [31]. SMC with optimal proposals does not finish in three days. Our S-Bayes RF is implemented in Matlab while others are in Python. Although in general S-Bayes RF is fast, getting a precise runtime comparison is not straightforward since implementation languages differ. In terms of accuracy performance, Bayesian decision trees perform comparably to the baseline kNN (13.20 ± 2.41) for Israeli, and perform rather poorly to kNN (36.53 ± 2.69) for AwA. The random forest and our Bayes random forest *outperform* the baseline kNN for both datasets. However for this dataset, we found that SVM performs rather well in comparison to random forest achieving 42.67 ± 4.59 and 47.83 ± 1.27 in Israeli and AwA, respectively.

6 Discussion and Conclusion

The random forest framework produces state-of-the-art performance in application domains, among others, machine learning, computer vision, computer graphics, and medical image analysis. It typically works by averaging several predictions of randomly trained trees. In this paper, we show a conceptually radical approach to generate a random forest based on Bayesian statistics: random sampling of many trees from a prior distribution, and subsequently performing a weighted ensemble.

Unlike other Bayesian models of decision trees which require computationally intensive Markov Chain Monte Carlo procedures, our framework utilises a data independent tree prior that facilitates offline tree generation. This prior allows us to sample a collection of decision trees even before looking at the data. Furthermore with the use of power likelihood, our method is able to explore space spanned by combining decision trees. This is in contrast to Bayesian decision trees where in the infinite data limit will put a mass to a single tree.

Our experimental results are encouraging, our S-Bayes RF outperforms Bayesian decision trees in term of speed and predictive performance, and is competitive with state-of-the-art random forest algorithms on both speed and accuracy. In the future, we are interested in exploring the possibility of adapting the power likelihood exponent β as new data points arrive using for example the method of [19].

Acknowledgments

The authors would like to thank two anonymous reviewers for constructive comments and Sara Wade, Wray L. Buntine, Viktoriia Sharmanska, Sebastian Nowozin, and James Lloyd for discussions.

References

[1] Leo Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone. *Classification and Regression Trees*. Wadsworth, 1984.

[2] J. R. Quinlan. Induction of decision trees. *Machine Learning*, pages 81–106, 1986.

[3] Sebastian Nowozin. Improved information gain estimates for decision tree induction. In *International Conference on Machine Learning (ICML)*, 2012.

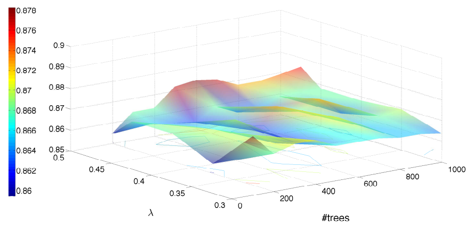
[4] L. Breiman. Random forests. Technical Report TR567, UC Berkeley, 1999.

[5] Jamie Shotton, Andrew W. Fitzgibbon, Mat Cook, Toby Sharp, Mark Finocchio, Richard Moore, Alex Kipman, and Andrew Blake. Real-time human pose recognition in parts from single depth images. In *Computer Vision and Pattern Recognition (CVPR)*, 2011.

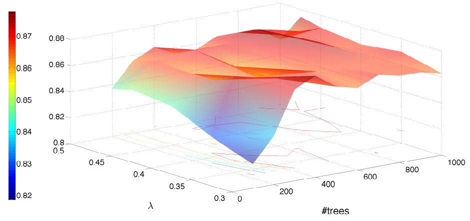
[6] Gabriele Fanelli, Matthias Dantone, Juergen Gall, Andrea Fossati, and Luc Gool. Random forests for real time 3d face analysis. *International Journal of Computer Vision*, pages 1–22, 2012.

TABLE 4: Accuracy results of trees and forest variants on high dimensional images dataset across 5 random repeats. C: number of categories. `ExtraTrees`: a random forest variant with random features and cut values.

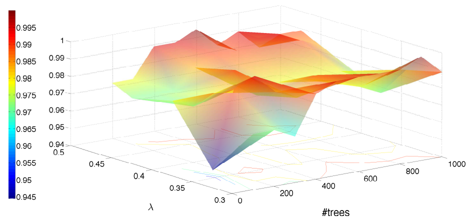
D	C	N	`MCMC`	`SMC: prior`	`RF: Gini`	`RF: Ent.`	`ExtraTrees`	`S-Bayes RF`
Israeli-Images dataset								
4,097	11	1,056	15.59±3.46	15.88±0.86	35.21±2.28	34.54±1.32	33.97±1.91	34.93±2.05
Animals with Attributes dataset								
2,000	10	3,000	27.03±1.80	27.39±2.66	41.96±0.91	41.86±1.38	41.57±0.92	38.17±0.92



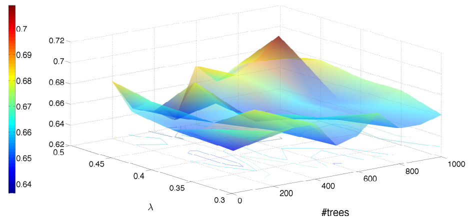
(a) `australian`



(b) `heart`



(c) `wine`



(d) `svmguide2`

Fig. 5: Sensitivity analysis of the performance of S-Bayes RF on test set with respect to the choices of splitting probability λ and number of trees #trees. We observe that in general S-Bayes RF is robust to the (λ,#trees)-parameter selection with the tendency of better accuracy performance with more trees and λ close to 0.5.

[7] Antonio Criminisi, Jamie Shotton, and Ender Konukoglu. Decision forests: A unified framework for classification, regression, density estimation, manifold learning and semi-supervised learning. *Foundations and Trends in Computer Graphics and Vision*, 7(2-3):81–227, 2012.

[8] Tin Kam Ho. The random subspace method for constructing decision forests. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(8):832–844, 1998.

[9] Wray L. Buntine. Learning classification trees. *Statistics and Computing*, 2:63–73, 1992.

[10] Jonathan J. Oliver and David J. Hand. On pruning and averaging decision trees. In *International Conference on Machine Learning (ICML)*, 1995.

[11] Hugh Chipman, Edward I. George, and Robert E. Mcculloch. Bayesian cart model search. *Journal of the American Statistical Association*, pages 935–948, 1998.

[12] David G. T. Denison, Bani K. Mallick, and Adrian F. M. Smith. A bayesian cart algorithm. *Biometrika*, 85(2):363–377, 1998.

[13] Hugh A. Chipman, Edward I. George, and Robert E. McCulloch. Bayesian ensemble learning. In *Neural Information Processing Systems (NIPS)*, 2007.

[14] Balaji Lakshminarayanan, Daniel M. Roy, and Yee Whye Teh. Top-down particle filtering for bayesian decision trees. In *International Conference on Machine Learning (ICML)*, 2013.

[15] T. P. Minka. Bayesian model averaging is not model combination. Technical report, MIT Media Lab., 2002.

[16] Hyun-Chul Kim and Zoubin Ghahramani. Bayesian classifier combination. *Journal of Machine Learning Research - Proceedings Track*, 22:619–627, 2012.

[17] Stephen Walker and Nils Lid Hjort. On bayesian consistency. *Journal of the Royal Statistical Society. Series B (Statistical Methodology)*, 63(4):811–821, 2001.

[18] Isadora AntonianoVillalobos and Stephen G. Walker. Bayesian nonparametric inference for the power likelihood. *Journal of Computational and Graphical Statistics*, 2012.

[19] Peter Grünwald. The safe bayesian: Learning the learning rate via the mixability gap. In *International Conference on Algorithmic Learning Theory (ALT)*, 2012.

[20] Tong Zhang. Learning bounds for a generalized family of bayesian posterior distributions. In *Neural Information Processing Systems (NIPS)*, 2003.

[21] Wray L. Buntine. *A Theory of Learning Classification Rules*. PhD thesis, University of Technology Sydney, 1992.

[22] J. M. Hammersley and D. C. Handscomb. *Monte Carlo methods*. Methuen London, 1964.

[23] Joseph G. Ibrahim and Ming-Hui Chen. Power prior distributions for regression models. *Statistical Science*, 15(1):pp. 46–60, 2000.

[24] N. Friel and A. N. Pettitt. Marginal likelihood estimation via power posteriors. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 70(3):589–607, 2008.

[25] Robert Gramacy, Richard Samworth, and Ruth King. Importance tempering. *Statistics and Computing*, 20(1):1–7, 2010.

[26] Florent Perronnin, Jorge Sánchez, and Thomas Mensink. Improving the fisher kernel for large-scale image classification. In *European Conference on Computer Vision (ECCV)*, 2010.

[27] Herbert Bay, Andreas Ess, Tinne Tuytelaars, and Luc Van Gool. Speeded-up robust features (surf). *Computer Vision and Image Understanding*, pages 346–359, 2008.

[28] F. Pedregosa et al. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[29] R. Caruana, N. Karampatziakis, and A. Yessenalina. An empirical evaluation of supervised learning in high dimensions. In *International Conference on Machine Learning (ICML)*, 2008.

[30] Adele Cutler and Guohua Zhao. Pert - perfect random tree ensembles. *Computing Science and Statistics*, 2001.

[31] Pierre Geurts, Damien Ernst, and Louis Wehenkel. Extremely randomized trees. *Machine Learning Journal*, 63(1), 2006.