Enhancing the Explainability of Gradient Boosting Classification through Comparable Samples Selection

Emilien Boizard
Université Paris-Saclay, CNRS, CentraleSupélec
Laboratoire des signaux et systèmes
91190, Gif-sur-Yvette, France
ORCID: 0000-0002-7627-5540

Gilles Chardon
Université Paris-Saclay, CNRS, CentraleSupélec
Laboratoire des signaux et systèmes
91190, Gif-sur-Yvette, France
ORCID: 0000-0003-0483-9957

Frederic Pascal
Université Paris-Saclay, CNRS, CentraleSupélec
Laboratoire des signaux et systèmes
91190, Gif-sur-Yvette, France
ORCID: 0000-0003-0196-6395

***Abstract*—Gradient-Boosted Decision Trees (GBDT) stand out as a powerful Machine Learning tool in tackling classification and regression tasks. Despite its effectiveness, GBDT, like other ensemble methods, suffers from a lack of explainability. Understanding these models' inner workings is crucial for comprehensively grasping their decision-making processes. In this study, we propose a method to enhance the explainability of GBDT, focusing on identifying specific training data points termed "comparable samples," which play a pivotal role in the model's predictions. Inspired by the Frank-Wolfe algorithm, we introduce Explainable Gradient Boosting (ExpGB), which aims to shed light on the relationships between input data attributes and model predictions. ExpGB operates by ranking training samples based on their decomposition coefficients within the algorithm's output. Higher weights assigned to particular training samples indicate a closer resemblance to the sample being analyzed. To validate the efficiency of our approach, we conduct a comparative analysis across three diverse datasets, contrasting ExpGB with traditional GBDT algorithms. Through this analysis, we evaluate the quality of estimation provided by ExpGB, thereby enhancing our understanding of GBDT's workings.**

***Index Terms*—Multiclass classification, explainable machine learning, gradient boosting**

I. Introduction

Tree-based gradient boosting algorithms [1]–[3] are now widely used in real-world applications due to their accuracy performances and their relatively low computation cost. Despite their impressive performance, these algorithms share a common drawback with other ensemble methods: a compromise between performance and model explainability. Explainability is of crucial importance as it builds user confidence in the algorithms and helps understand model behaviour, which is crucial when deploying models in real-world scenarios where operational data might deviate from training and validation sets. Several methods and tools have thus been developed in the literature to enhance the explainability of machine learning models. Of these, the most commonly used are probably SHAP values [4], [5], LIME [6], and feature importance [7], [8]. All three focus on the effect of features on the models' predictions, which is not necessarily relevant when the features do not represent a physical aspect, for instance, when a principal component analysis (PCA) is used on the data.

In this study, our main focus is to address classification tasks through the use of gradient-boosting algorithms. Instead of summarizing features, our method identifies and leverages training data similar in features and response to a test instance. This approach is particularly beneficial for non-expert users and in scenarios where data instances, like images, can be directly interpreted by humans, even though the underlying features (in this context, pixels) may not be directly interpretable.

The problem considered here is similar to the prototype selection problem, which was extensively studied for classification problems [9]–[13]. These global methods allow for determining representative data in each class to reduce the size of the dataset. On the other hand, our method is local because the chosen representative training samples will depend on the sample of interest, and it provides a better representation of the data set. Indeed, although several elements in the dataset belong to the same class, they may have very different features, which is why it is helpful to have a local method.

The main contribution of the paper is the introduction of a variant of GBDT for classification problems using the Frank-Wolfe algorithm, aiming at improving the explainability of the results. To this end, the method extracts training samples similar to a given test sample, both in terms of features and response. The proposed classification algorithm aims to express its prediction as a convex combination of the target values of the training dataset, which corresponds to the probability of belonging to each class. This selection of similar samples facilitates a more intuitive explanation of predictions.

Empirical evaluations on real datasets show that the proposed method achieves predictive performance on par with the best state-of-the-art GBDT algorithms: CatBoost [14] and XGBoost [15]. In addition, similar samples extracted using this method are statistically evaluated on the same datasets to show that they are very close to the considered sample both in terms of features and response.

II. Gradient boosting framework

In multiclass classification problems, one aims at learning a function $\mathbf{p}(\mathbf{x}) = (p_1(\mathbf{x}), \dots, p_K(\mathbf{x}))$ mapping features $\mathbf{x} \in \mathbf{R}^d$ to the vector of the probability of belonging to each class k - so that it minimizes the expected loss $\mathcal{L}(F) = E(L(\mathbf{y}, \mathbf{p}(\mathbf{x})))$ where L denotes a loss function, E is the mathematical expectation, and K is the number of classes. This function $\mathbf{p}$ is trained using N training samples $\{(\mathbf{x}_n, \kappa_n)\}$, where κ_n is the class of $\mathbf{x}_n$, or equivalently $\{(\mathbf{x}_n, \mathbf{y}_n)\}$ where the class κ_n of $\mathbf{x}_n$ is one-hot encoded in $\mathbf{y}_n$, that is $y_{n\kappa_n} = 1$, and zero for the other coordinates of $\mathbf{y}_n$. The empirical expectation of the loss

$$\mathcal{E} = \frac{1}{N} \sum_{n=1}^{N} L(\mathbf{y}_n, \mathbf{p}(\mathbf{x}_n)) \tag{1}$$

is minimized, where a usual choice for the loss L is the cross-entropy loss, or logarithmic loss

$$L(\mathbf{p}) = -\sum_{n=1}^{N} \sum_{k=1}^{K} y_{nk} \log p_k(\mathbf{x}_n). \tag{2}$$

Usually, the problem is reparameterized using the symmetric multiple logistic transform, with

$$F_k(\mathbf{x}) = \log p_k(\mathbf{x}) - \frac{1}{K} \sum_{l=1}^{K} \log p_l(\mathbf{x}). \tag{3}$$

Using this transformation has the significant consequence that the optimization problem is now unconstrained.

As the amount of available data is limited, the minimizers of Eq. (1) are not guaranteed to yield accurate estimations of $\mathbf{p}(\mathbf{x})$ for $\mathbf{x}$ not in the training dataset. Regularity assumptions are needed in order to generalize the training data and avoid overfitting.

Gradient tree boosting follows an approach where a limited set of parameters describes the functions F_k. More specifically F_k is restricted to belong to the set : $\left\{F, F(\mathbf{x}) = \sum_{t=1}^{T} \gamma_k^t h_{\theta_k^t}(\mathbf{x})\right\}$, where the $h_{\theta_k^t}$ are simple functions parameterized by low-dimensional parameters $\theta_k^t \in \Theta$, where Θ is a low-dimensional parameter space, and the γ_k^t are real weights. Here, the exponent $\cdot^t$ denotes the value at the t-th iteration.

Gradient boosting is an iterative algorithm, which generates a sequence of functions F_k^t, inspired by gradient descent: at each iteration t, an update $h_{\theta_k^t}$ is added in order to decrease the empirical loss $\mathcal{E}$. This update is selected so that its values $(h_{\theta_k^t}(\mathbf{x}_n)) \in \mathbf{R}^N$ on the training points $\mathbf{x}_n$ is the most parallel with the opposite of the gradient $-\mathbf{g}_k^{t-1}$ of $\mathcal{E}$ with respect to the values of F_k at the training samples, which writes

$$-g_{kn}^{t-1} = \delta_{k,\kappa_n} - p_k^{t-1}(\mathbf{x}_n) \tag{4}$$

where δ is the Kronecker symbol. The update $h_{\theta_k^t}$ is the solution to the regression problem

$$\theta_k^t = \arg\min_{\theta \in \Theta} \| - \mathbf{g}_k^{t-1} - \mathbf{h}_\theta \|_2^2 \tag{5}$$

A popular choice for $h_{\theta_k^t}$ is to use shallow regression trees [16], θ_k^t corresponding in this case to the splitting features, splitting location, and terminal node values of the tree.

Such trees can be described by subsets $\{A_{kl}^t\}$ covering the space, and values b_{kl}^t, with

$$h_{\theta_k^t} = \sum_{l=1}^{L} b_{kl}^t \mathbf{1}_{A_{kl}^t} \tag{6}$$

The number of subsets L depends on the depth of the trees.

Overfitting is avoided by limiting the complexity of each $h_{\theta_k^t}$ (e.g. by constraining the depth of the regression trees), using a finite number of iterations, and dampening the iterations by setting $F_k^t(\mathbf{x}) = F_k^{t-1}(\mathbf{x}) + \lambda \gamma_t h_{\theta_k^t}(\mathbf{x})$ with $0 < \lambda < 1$. Further regularizations can also be applied.

The cost of a prediction is the computational cost of applying the weak learners $h_{\theta_k^t}$ to $\mathbf{x}$.

III. Gradient Boosting with Frank-Wolfe

In the gradient boosting algorithm recalled in the previous section, the classifier consists of a linear combination of a large number of weak learners, and the explanatory power is lost in favour of prediction performance. We propose an alternative formulation of the gradient boosting algorithm, where the probabilities $\mathbf{p}$ are the variables to be optimized. To account for the constraints on the probabilities (positivity and summing to 1), we propose an algorithm inspired by the Frank-Wolfe algorithm, which does not necessitate any projection onto the probabilistic simplex. Additionally, we show that the probabilities can be further expanded as linear combinations of the probability values of the training data. This property will be leveraged to assess the similarity between a data point and the training samples.

A. The proposed algorithm

The standard Frank-Wolfe algorithm is given in Algorithm 1, for the minimization of a smooth function f in a compact convex feasible set Ω. At each iteration, the gradient of the objective function is computed, and the linear approximation of the objective function at the current iterate x^t is minimized in the feasible set, at s. Then, the next iterate is found as a convex combination of the current iterate x^t and s. As Ω is convex, the next iterate x^{t+1} is guaranteed to lie in the feasible set.

In the proposed algorithm, as in gradient boosting, the opposite of the gradient is replaced by a tree. Here, the opposite of the gradient of the empirical loss with respect to the values of p_k at the training samples is

$$-g_{kn}^{t-1} = \frac{\delta_{k,\kappa_n}}{p_k^{t-1}(\mathbf{x}_n)} \tag{7}$$

As in the previous section, we notice that this vector is nonnegative.

In contrast to the gradient boosting algorithm, we consider a vector-valued decision tree $\mathbf{h}$ aiming at approximating jointly the coordinates of the gradient:

$$\mathbf{h}_{\theta^t} = \sum_{l=1}^{L} \mathbf{b}_l^t \mathbf{1}_{A_l^t} \tag{8}$$

Following the principles of the Frank-Wolfe algorithm, the update $\mathbf{s}$ is found by maximizing the scalar product

$$\langle \mathbf{s}, \mathbf{h}_{\theta^t} \rangle = \int_{\mathbf{R}^d} \langle \mathbf{s}(\mathbf{x}), \mathbf{h}_{\theta^t}(\mathbf{x}) \rangle_K \, q(\mathbf{x}) d\mathbf{x} \tag{9}$$

where q is the density probability of $\mathbf{x}$ and $\langle \cdot, \cdot \rangle_K$ is the scalar product in $\mathbf{R}^K$. For a given $\mathbf{x}$, the scalar product $\langle \mathbf{s}(\mathbf{x}), \mathbf{h}_{\theta^t}(\mathbf{x}) \rangle_K$ is maximized by setting $s_{k^\star}(\mathbf{x}) = 1$ for the index $k^\star$ where $\mathbf{h}_{\theta^t}(\mathbf{x})$ is maximal, and zero elsewhere. Given the expression of $\mathbf{h}_{\theta^t}$ as a piecewise constant function, the optimal $\mathbf{s}$ is also a piecewise constant function, and

$$\mathbf{s} = \sum_{l=1}^{L} \mathbf{u}_l \mathbf{1}_{A_l^t} \tag{10}$$

where the K dimensional vector $\mathbf{u}_l$ is 1 on the maximal coordinate of $\mathbf{b}_l$ and zero elsewhere.

In addition to these projection-free iterations, the predictions of the proposed algorithm can be, for each testing sample, expanded as a convex combination of the training values. Indeed, a given leaf where the value of the tree is maximal at a given index $k^\star$, contains at least one training sample of the class $k^\star$. This follows from the fact that $\mathbf{b}_l^t$ being the average of the vectors $-\mathbf{g}_{kn}^{t-1}$ of the corresponding leaf, it is itself nonnegative, and its maximal value is strictly positive. Additionally, its value for a class k with no sample in the leaf is zero. Note that we have excluded the case of an empty leaf, which is avoided by the CART algorithm. The update can thus be written as

$$\mathbf{u}_l = \frac{1}{N_l^t} \sum_{\substack{n \text{ such that} \\ \mathbf{x}_n \in A_l^t \\ \kappa_n = k^\star}} \mathbf{y}_n \tag{11}$$

where N_l^t is the number of terms in the sum. By a trivial induction, all iterates can be expanded as convex combinations of the training values $\mathbf{y}_n$, with weights $\mathbf{w} = (w_n)_{n=1\ldots N}$ for the final prediction at iteration T.

Algorithm 1 Frank-Wolfe Algorithm

$x^0 \in \Omega$
for t = 1 to T **do**
  Compute $s := \arg\min_{s \in \Omega} \langle s, \nabla f(x^t) \rangle$
  $\gamma := \frac{2}{t+2}$
  Update $x^{t+1} = (1-\gamma)x^t + \gamma s$
end for

Algorithm 2 ExpGB, fit

Require: $\{(\mathbf{x}_n, y_n)\}_{n=1,\ldots,N}$, T,
  initialization : $p_k^0(\mathbf{x}) = \frac{1}{N} \sum_{n=1}^{N} y_{nk}, n = 1 \ldots N$
  for t = 1 to T **do**
    Compute $g_{kn}^{t-1}, n = 1 \ldots N$
    Fit a regression tree h_t on the data $\{(\mathbf{x}_n, -\mathbf{g}_n^t)\}, n = 1 \ldots N$ with leaves A_l^t and values b_l^t
    Define $\mathbf{s}$ using Eqs. (10) and (11)
    $\gamma := \frac{2}{t+2}$
    $\mathbf{p}^t(\mathbf{x}) = (1-\gamma)\mathbf{p}^{t-1}(\mathbf{x}) + \gamma \mathbf{s}(\mathbf{x})$
  end for
  return $h_k^t, t = 1 \ldots T$

B. Finding comparables

Once the prediction of a class for a testing sample $\mathbf{z}$ is made, we propose to explain this prediction by inspecting the training samples with the largest weights w_n in the expansion of $\mathbf{p}(\mathbf{z})$. Indeed, from Eq. (11), samples with high weights in the expansion are likely

- to frequently fall in the same leaves as $\mathbf{z}$, and thus share characteristics relevant to the prediction,
- and to belong to the same class as $\mathbf{z}$, as only the weights of training samples belonging to the class where the probability is increased at an iteration are increased.

The cost of the search of these comparable training samples is virtually free, as it only involves only the weight vector $\mathbf{w}$, and a search of its maximal coefficients. The total cost is thus the product of the number of iterations and the depth of the trees. In contrast, a comparable search based on the computation of an appropriate distance between $\mathbf{z}$ and the training samples (e.g., KNN), would necessitate a cost proportional to the product of the size of the training dataset and the dimension of the features.

IV. Results

To evaluate the performances of the proposed approach, we will compare it to other gradient boosting methods - Catboost, and XGBoost - on several public datasets with classification tasks.

The first and foremost dataset we study in this article is MNIST [17], composed of 28x28 pixel images depicting hand-written digits ranging from 0 to 9, with 60000 training samples and 10000 testing samples. The MNIST dataset is particularly relevant for testing our method due to the intuitive concept of similar samples it presents. Indeed, it is really easy to recognize visually comparable digits. Two classification tasks will be considered: standard classification of the digits, and odd/even classification. No pre-processing or feature extraction is applied before training or inference.

In the *magic gamma telescope dataset* [18], the goal is to use measurements of high-energy gamma particles in an atmospheric Cherenkov telescope to separate photons caused by primary gammas from the images of hadronic showers

TABLE I
ACCURACY

Dataset	Cat	XGBoost	ExpGB	ExpGBlog
MNIST odd/even	0.986	0.987	0.960	0.978
MNIST number	0.973	0.976	0.9390	0.964
MagicTelescope	0.865	0.862	0.840	0.861
UjiIndoorLoc bldg. + floor	0.848	0.886	0.774	0.780
UjiIndoorLoc bldg.	0.998	0.987	0.967	0.982

TABLE II
AVERAGE DISTANCE BETWEEN THE TESTED SAMPLE, AND MOST SIMILAR TRAINING SAMPLES (WIFI DATASET) (METERS)

# samples	ExpGB weights	ExpGB counts	CatBoost counts	XGB counts
1	18.9	18.5	17.1	16.8
5	19.0	19.4	19.1	19.9
10	19.53	19.7	20.6	22.7
20	20.35	20.7	23.2	25.7

initiated by cosmic rays in the upper atmosphere. The training dataset contains 10032 samples with 10 features, and the testing dataset contains 3344 samples. This dataset was selected to evaluate the performance of the models in a binary classification task.

Finally, we consider the *UjiIndoorLoc dataset* [19], with an indoor localization problem. The RSSI of Wifi access points (WAPs) are measured in several places in three buildings, with several mobile devices. We predict the floor and building of a measurement using the RSSI levels of 520 WAPs. The training data set contains 19937 samples, and the testing data set 1111 samples, obtained with a mobile device not used in the training set. The coordinates of the localization are not used during the training of the prediction but will be used to assess the similar samples given by the proposed method.

A. Hyperparameter selection and Data Preprocessing

To optimize our models, we focused on two key parameters: the number of iterations (set at 50, 100, 150, 200, 250, 300, 350, 400, 450, 500) and the depth of the trees (ranging from 5 to 10). We employed a grid search combined with a K-fold cross-validation on the training data to select each model's hyperparameters, typically using $K = 5$. This involved dividing the training data into five parts, training the models in four parts while evaluating their performance in the remaining part. This cycle was repeated four times, ensuring each part was used as a validation set exactly once. The choice of hyperparameters was based on those that yielded the lowest average cross-entropy across the five validation subsets.

B. Prediction performance

The prediction performance of ExpGB is compared with the two reference models for each of the three datasets ExpGB is also tested using the gradient in (4) when training the regression trees, denoted as ExpGBlog. Table I shows the accuracy on each test set for the two reference models and ExpGB. As expected, the two reference models perform similarly on all datasets. ExpGB also exhibits similar performances, proving that the modifications made to enhance explainability are not done at the expense of decreased performance. The slightly better variant ExgGBlog will be used in the remaining results.

C. Extracted prototypes at a glance

The main objective of our algorithm ExpGB was to enhance the explainability of GBDT by extracting similar training samples to a given tested sample. Here, we look at these prototypes through examples extracted from the MNIST dataset. In Figure 1, four criteria are tested to assess the similarity of

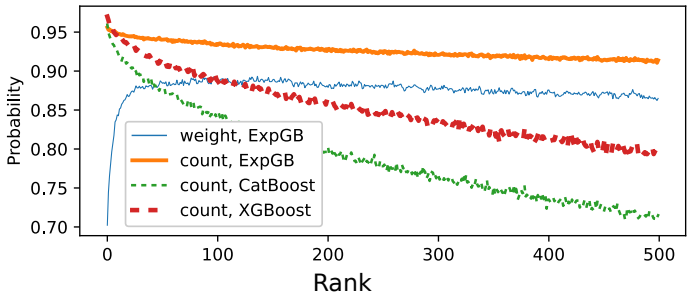

Fig. 1. Probability of a similar sample belonging to the class of the tested sample, according to decreasing weights and counts for FW, and decreasing counts for catboost

training samples to a given testing sample: by decomposition weights of the ExpGB probability decomposition, by counting the number of times a given training sample appears in the decomposition, and by counting the number of leaves containing both the testing sample and a training sample for CatBoost and XGBoost. Figure 1 shows that the counting method for ExpGB is able to extract a larger number of training samples belonging to the same class as the testing sample.

We also require that similar samples have more similarity to a testing sample rather than belonging to the same class. Figure 2 shows, for six testing samples, the five most similar samples in terms of decreasing counts, and five random training samples from the same class as the testing sample. One sees that in addition to belonging to the same class, the similar samples have similar shapes to the testing samples, the most striking example being the angles of the 1's.

Results for the odd/even classification are given in Figure 3. Although the learning tasks are only concerned with the *parity* of the digit, most of the extracted samples represent the same *digit* as the testing sample. Digits different from the testing sample, but from the same class, appear. They are still similar to the testing sample (the 6s selected as similar to the 0 in the first line is similar to its left neighbour, and the 7 in the second line is similar to a 9 with a missing stroke).

For the UjiIndoorLoc dataset, the 5 most similar training samples are given for 6 testing samples, two in each building, on figure 4. In addition, the average distance between a tested sample and its 1, 5, 10 or 20 most similar samples is given in Table II, averaged over the testing dataset. As seen in Figure 1 on the MNIST dataset, ExpGB is capable of extracting more relevant training samples than CatBoost and XGBoost.

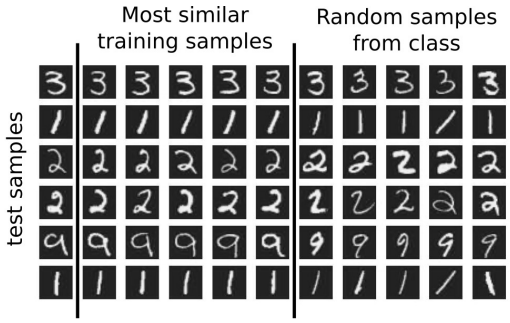

Fig. 2. Comparable samples for the MNIST dataset. From left to right: testing samples, training samples ordered by decreasing counts, random sampling from the class

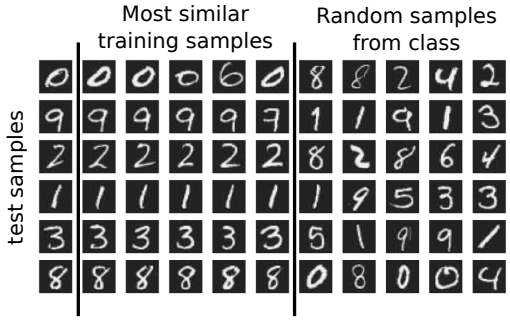

Fig. 3. Comparable samples for the MNIST dataset, for odd/even task. From left to right: testing samples, training samples ordered by decreasing counts, random sampling from the class

V. Conclusion and future works

In this article, we proposed a variant of gradient-boosting algorithms using Frank-Wolfe principles to enhance the explainability of such algorithms. More precisely, we have developed a method that allows each of its predictions to be written directly as a convex combination of the responses from the training set while having the same level of performance as the gradient boosting methods from the state-of-the-art. We then showed that the algorithm can extract samples from the training dataset similar to a testing sample, both in terms of features and response. The tradeoff between performance and explainability is not as clear-cut, suggesting that effectively balancing these two critical aspects is possible.

These outcomes pave the path for future advancements. Similar to traditional gradient boosting algorithms, performance enhancements could be achieved by employing different types of regressor trees, such as those based on histograms. Furthermore, the introduction of native handling for missing values and the management of categorical features could also be explored.

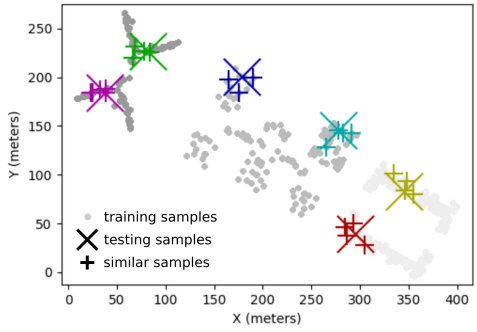

Fig. 4. Wifi localization, map of the training locations, and six testing samples with their five most similar testing samples.

References

[1] J. H. Friedman, "Greedy function approximation: A gradient boosting machine.," *The Annals of Statistics*, vol. 29, no. 5, pp. 1189 – 1232, 2001.

[2] J. H. Friedman, "Stochastic gradient boosting," *Computational statistics & data analysis*, vol. 38, no. 4, pp. 367–378, 2002.

[3] T. Hastie, R. Tibshirani, J. H. Friedman, and J. H. Friedman, *The elements of statistical learning: data mining, inference, and prediction*, vol. 2. Springer, 2009.

[4] S. M. Lundberg and S.-I. Lee, "A unified approach to interpreting model predictions," in *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, (Red Hook, NY, USA), p. 4768–4777, Curran Associates Inc., 2017.

[5] S. M. Lundberg, G. G. Erion, and S.-I. Lee, "Consistent individualized feature attribution for tree ensembles," *arXiv preprint arXiv:1802.03888*, 2018.

[6] M. T. Ribeiro, S. Singh, and C. Guestrin, ""Why should i trust you?": Explaining the predictions of any classifier," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, (New York, NY, USA), p. 1135–1144, Association for Computing Machinery, 2016.

[7] L. Breiman, "Random forests," *Machine learning*, vol. 45, pp. 5–32, 2001.

[8] H. Ishwaran, "Variable importance in binary regression trees and forests," *Electronic Journal of Statistics*, vol. 1, pp. 519–537, 2007.

[9] S. Tan, M. Soloviev, G. Hooker, and M. T. Wells, "Tree space prototypes: Another look at making tree ensembles interpretable," in *Proceedings of the 2020 ACM-IMS on Foundations of Data Science Conference*, pp. 23–34, 2020.

[10] J. Bien and R. Tibshirani, "Prototype selection for interpretable classification," *The Annals of Applied Statistics*, vol. 5, no. 4, pp. 2403–2424, 2011.

[11] B. Kim, C. Rudin, and J. A. Shah, "The bayesian case model: A generative approach for case-based reasoning and prototype classification," *Advances in neural information processing systems*, vol. 27, 2014.

[12] K. S. Gurumoorthy, A. Dhurandhar, and G. Cecchi, "Protodash: Fast interpretable prototype selection," *arXiv preprint arXiv:1707.01212*, 2017.

[13] K. S. Gurumoorthy, A. Dhurandhar, G. Cecchi, and C. Aggarwal, "Efficient data representation by selecting prototypes with importance weights," in *2019 IEEE International Conference on Data Mining (ICDM)*, pp. 260–269, IEEE, 2019.

[14] A. V. Dorogush, V. Ershov, and A. Gulin, "Catboost: gradient boosting with categorical features support," *arXiv preprint arXiv:1810.11363*, 2018.

[15] T. Chen and C. Guestrin, "Xgboost: A scalable tree boosting system," in *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, pp. 785–794, 2016.

[16] L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone, *Classification and regression trees*. New York: Chapman & Hall, 1993.

[17] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[18] D. Heck, J. Knapp, J. Capdevielle, G. Schatz, T. Thouw, *et al.*, "Corsika: A monte carlo code to simulate extensive air showers," 1998.

[19] J. Torres-Sospedra, R. Montoliu, A. Martínez-Usó, J. P. Avariento, T. J. Arnau, M. Benedito-Bordonau, and J. Huerta, "UJIIndoorLoc: A new multi-building and multi-floor database for WLAN fingerprint-based indoor localization problems," in *2014 International Conference on Indoor Positioning and Indoor Navigation (IPIN)*, pp. 261–270, Oct. 2014.