Tree Space Prototypes: Another Look at Making Tree Ensembles Interpretable

Sarah Tan
ht395@cornell.edu
Cornell University

Matvey Soloviev
ms2837@cornell.edu
Cornell University

Giles Hooker
gjh27@cornell.edu
Cornell University

Martin T. Wells
mtw1@cornell.edu
Cornell University

ABSTRACT

Ensembles of decision trees perform well on many problems, but are not interpretable. In contrast to existing approaches in interpretability that focus on explaining relationships between features and predictions, we propose an alternative approach to interpret tree ensemble classifiers by surfacing representative points for each class – prototypes. We introduce a new distance for Gradient Boosted Tree models, and propose new, adaptive prototype selection methods with theoretical guarantees, with the flexibility to choose a different number of prototypes in each class. We demonstrate our methods on random forests and gradient boosted trees, showing that the prototypes can perform as well as or even better than the original tree ensemble when used as a nearest-prototype classifier. In a user study, humans were better at predicting the output of a tree ensemble classifier when using prototypes than when using Shapley values, a popular feature attribution method. Hence, prototypes present a viable alternative to feature-based explanations for tree ensembles.

CCS CONCEPTS

• **Computing methodologies → Instance-based learning**; *Classification and regression trees*; • **Human-centered computing**;

KEYWORDS

Interpretability; Tree Ensemble Classifiers; Prototypes

ACM Reference Format:
Sarah Tan, Matvey Soloviev, Giles Hooker, and Martin T. Wells. 2020. Tree Space Prototypes: Another Look at Making Tree Ensembles Interpretable. In *Proceedings of the 2020 ACM-IMS Foundations of Data Science Conference (FODS '20), October 19–20, 2020, Virtual Event, USA.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3412815.3416893

1 INTRODUCTION

As machine learning is increasingly employed alongside human reasoning in a wide range of tasks, it has been recognized that it is desirable for these systems to be made interpretable: a human user working alongside an ML system should be able to maintain a mental model of why and how the system arrives at its outputs, so she may either obtain confidence in the outputs or conversely recognize when they are wrong [8].

Ensembles of decision trees such as random forests [3] and boosted trees [11] perform well across a variety of problems [7]. However, while their decision tree components may be interpretable [10], this is no longer true for ensembles with hundreds or thousands of trees. Current attempts to interpret tree ensembles include seeking one tree that best represents the ensemble [18, 54], model-agnostic explanations not exclusive to tree ensembles [43], feature importance [20], partial dependence plots [11], etc. However, many of these describe how features affect predictions, and their complexity increases with the number of features.

Prototypes are representative points that provide a condensed view of a dataset [2, 19]. The value of prototypes for case-based reasoning [44] has been discussed in studies of human decision making [24]. Prototypes have also been used to summarize large datasets [39] when not all points can be inspected. In this paper, we propose an alternative to feature-based explanations for tree ensemble classifiers: rather than explaining which features led to a certain class being predicted, we propose to explain a prediction by presenting similar points that "represent" that class (Figure 1). Since these prototypes will be identified using distance functions derived from the tree ensemble, we call them tree space prototypes.

A key question is how to define similarity. Unsupervised distances such as Euclidean distance in feature space do not capture relationships between features and labels, whether actual or predicted. Instead, we need a distance that takes into account: (1) the predictions made by the tree ensemble; (2) how the predictions came about (i.e. how the tree ensemble used the features to arrive at the predictions). Such a distance has been defined for random forest models (RF) where each tree contributes equally to the overall prediction. We generalize this to gradient boosted trees (GBTs), where individual trees that make up a GBT model can have different contributions to the overall prediction.

By adapting a known approximation algorithm for the k-medoids problem, we can efficiently search for prototypes that are "central" for a class according to these proximity functions. Nearest-prototype classifiers using these prototypes sometimes even exceed the accuracy of the original tree ensemble. However, this algorithm has no notion of when one class may benefit from more prototypes than another class (e.g. if one class is more complex: consider for instance a disease which affects many different types of individuals, but does not affect one type of individual, so the class of sick individuals is more complex to characterize than the class of healthy individuals). Hence, we introduce new *class-aware* prototype selection methods with the flexibility to choose a variable number of prototypes per class.

FODS '20, October 19–20, 2020, Virtual Event, USA

ACM ISBN 978-1-4503-8103-1/20/10…$15.00
https://doi.org/10.1145/3412815.3416893

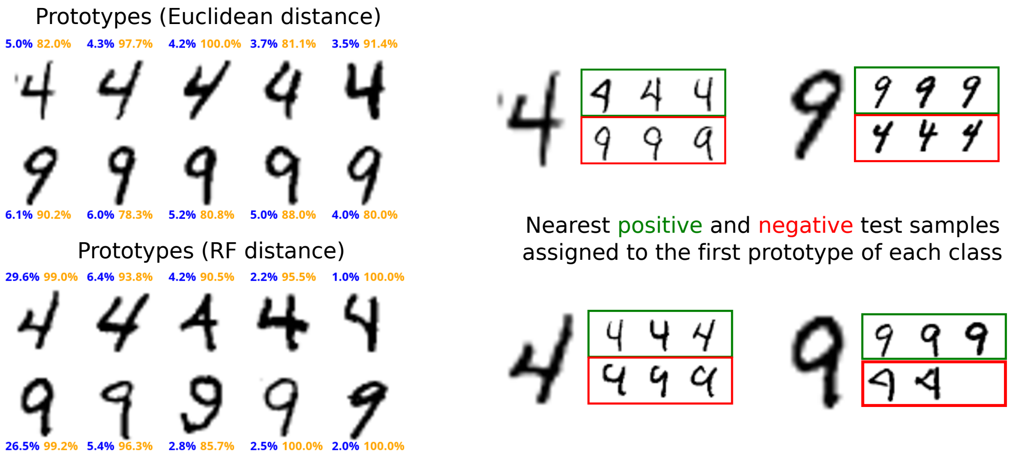


Figure 1: *Left:* Prototypes with largest coverage for the classes 4 and 9 on the MNIST dataset when using Euclidean distance and random forest distance to find the prototypes. The number in blue denotes the coverage of the prototypes (percentage of test points assigned to that prototype) while the number in orange denotes the accuracy of the prototype (percentage of points assigned to that prototype that have the same label as the prototype). Note the hooked 9 and closed 4, which are only captured by RF distance but not Euclidean distance. *Right:* Nearest correct and incorrect test points assigned to the top prototypes. The first row denotes points with the same label as the prototype, second row are points incorrectly classified by the prototype.

To evaluate the interpretability of tree space prototypes, we conducted a user study in which human subjects anticipated the output of a tree ensemble classifier on a dataset of car fuel efficiency, using either prototypes or Shapley values, a popular feature attribution method [37]. Our results suggest ($p \approx 0.035$) that prototypes convey better understanding of the tree ensemble classifier's behavior.

To summarize, the contributions of this paper are: (1) An alternative approach to interpreting tree ensemble classifiers by selecting representative points – prototypes – for each class; (2) a new distance function for GBT models; (3) new prototype selection methods with theoretical guarantees, that have the flexibility to choose a different number of prototypes in each class.

2 BACKGROUND AND NOTATION

2.1 RF Distance

Let t be the number of trees in the RF model. The ith tree ($i \in [t]$) has τ_i leaves, each of which represents a region $R_{j,i}$ ($j \in [\tau_i]$) of feature space. Each individual tree induces a classifier $c_i^{\text{Tree}}(s) = \sum_{j=1}^{\tau_i} \alpha_{j,i} \mathbb{I}(s \in R_{j,i})$, where $\alpha_{j,i}$ is the predicted value in the jth leaf of the ith tree (for binary classification, this is just the proportion of points in that leaf with label 1) and $\mathbb{I}$ is the indicator function. The RF classifier is the average of this, taken over all trees:

$$c^{\text{RF}}(s) = \frac{1}{t} \sum_{i=1}^{t} c_i^{\text{Tree}}.$$

Using the above notation, we can re-write the random forest classifier as

$$c^{RF}(s) = \frac{1}{t} \sum_{i=1}^{t} \sum_{j=1}^{\tau_i} \frac{1}{N} \sum_{k=1}^{N} \mathbb{I}(s \in R_{j,i}) \mathbb{I}(s_k \in R_{j,i}) y_k = \frac{1}{N} \sum_{k=1}^{N} K(s, s_k) y_k$$

for the kernel

$$K(s, s') = \frac{1}{t} \sum_{i=1}^{t} \sum_{j=1}^{\tau_i} \mathbb{I}(s \in R_{j,i}) \mathbb{I}(s' \in R_{j,i}).$$

This connection has allowed the study of random forests in which tree structure is generated independently of the data; in particular [45] provides explicit formulae for the corresponding $K(s, s')$, although this is much more challenging for supervised trees which adapt to the contours of the underlying response. This same representation results in the proximity function between two points:

Definition 1. *[4] The RF proximity of a pair of points is an unweighted average of the number of trees in the RF model in which the points end up in the same leaf:x*

$$\begin{aligned} &\text{proximity}^{\text{RF}}(s, s') \\ &= \frac{1}{t} \sum_{i=1}^{t} \sum_{j=1}^{\tau_i} \mathbb{I}(s \in R_{j,i}) \mathbb{I}(s' \in R_{j,i}). \end{aligned} \tag{1}$$

The RF distance between a pair of points is then:

$$d^{\text{RF}}(s, s') = 1 - \text{proximity}^{\text{RF}}(s, s').$$

Since the regions $\{R_{j,i}\}_{j=1}^{\tau_i}$ partition the feature space, each point $s \in S$ can be in at most one region, and so the inner sum in Equation (1) takes on value 0 or 1 for each tree. Thus the proximity, as a convex combination of these, lies between 0 and 1, and so does the distance function. It is easily confirmed that the proximity of a point to itself is 1, and hence $d(s, s) = 0$, but it should be noted that d is not in general a metric, but a pseudosemimetric as it does not satisfy the triangle inequality – as noted by [50]. This it not uncommon in the metric learning literature, and in fact, no locally adaptive distance – distance that varies across feature space [33]

– can satisfy the triangle inequality [50]. Later, we will adapt RF distance to construct a distance function for GBTs.

2.2 The k-Medoids Problem

Given a proximity function, it is natural to construct a classifier by taking those points that are particularly close to some point or region considered representative (prototypical) of a class to belong to that class. How should these prototypes be selected?

For accuracy, we would want every point in a class to be closer to a prototype of that class than any prototype of another class. This is generally hard; a more tractable related approach is to instead seek to simply make the points in each class as close to a prototype as possible. If we further take the tradeoff between different points' distance from a prototype to be linear, this is known as the k-*medoids clustering problem* (see e.g. [2] for another application to prototypes). The objective of this problem is to find a subset $M \subseteq S$ of medoids, $|M| = k$, such that the sum distance from each object to the nearest medoid is minimized. Formally, we seek to minimize the objective function

$$f(M) = \sum_{s \in S} \min_{m \in M} d(s, m). \tag{2}$$

This problem is known to be NP-hard [41]. However, [13] present a greedy algorithm that starts with an empty set and repeatedly adds the single point $s \in S \setminus M$ that increases the value of a related function by the most, which they show produces a reasonable approximation in polynomial time.

If the points are labelled by a classifier, it is natural to only consider, for each point, medoids that belong to the same class. Thus, we define the q-*classwise* k-*medoids problem* as finding the subset $M \subseteq S$ of k medoids such that the sum distance from each point to the nearest medoid *of the same class* is minimized, i.e. that minimizes

$$f(M) = \sum_{s \in S} \min_{m \in M : c(m) = c(s)} d(s, m). \tag{3}$$

Even in the presence of multiple classes, it is possible to use the single-class algorithm of [13] by applying it separately to every class in turn to generate $k_1, \ldots, k_q$ prototypes for each class ($\sum_i k_i = k$). However, it is not clear what the right choice of k_i for each class is, and one could easily lose accuracy by overprovisioning one compact class that would be adequately covered by a small number of prototypes while not having sufficiently many prototypes for another class whose points are spread into many clusters. With the naive choice that $k_1 = \ldots = k_q = k/q$, we call this the uniform greedy submodular (SM-U) prototype selection method, and use it as one of our baselines.

However, it turns out that an analysis similar to that for the single-class case can also be applied directly to the q-classwise objective function. Based on this, we will introduce a greedy algorithm that operates on all classes in the q-classwise k-medoids problem simultaneously. Since this algorithm in effect chooses the class where adding another prototype yields the largest improvement, we will call it *adaptive* in contrast with the uniform algorithm.

3 METHOD

Our goal is to find prototypes for tree ensemble classifiers. In this section, we describe two methodological contributions of this paper: defining a distance function for GBT models, and new, adaptive prototype selection methods that choose a variable number of prototypes based on which class could benefit the most from another prototype.

3.1 Constructing a Distance Function for GBT

We start by considering the prediction function of the GBT classifier, which is learned iteratively:

$$c_i^{\text{GBT}}(s) = c_{i-1}^{\text{GBT}}(s) + \gamma_i c_i^{\text{Tree}}(s)$$

where the initial value c_0^{GBT} is initialized, depending on implementation, as zero, or the fraction of elements of S with label 1 in the case of binary classification, etc. γ_i is a step size, typically found using line-search, that provides a correction to account for the quadratic approximation to the loss that is used by gradient boosting. The GBT classifier then is the one that incorporates all t trees:

$$c^{\text{GBT}}(s) = c_t^{\text{GBT}}(s).$$

Unlike RF, GBTs cannot be expressed directly as kernel methods: the values in each leaf are not given by averages of the corresponding responses. Further, each tree is no longer generated by an identical process or contributes equally to the prediction. Hence, each tree can no longer be weighted equally, unlike in RF models. Instead, we propose that a natural way is to weigh the contribution of each tree to the proximity function by the size of its contribution to the overall prediction. By using the L_2 norm to measure size, we arrive at the following definition:

Definition 2. *The GBT proximity of a pair of points is a weighted average of the number of trees in the GBT model in which the points end up in the same leaf:*

$$\begin{aligned} &\text{proximity}^{\text{GBT}}(s, s') \\ =\ &\sum_{i=1}^{t} \sum_{j=1}^{\tau_i} \frac{w_i}{\sum_{i=1}^{t} w_i} \mathbb{I}(s \in R_{j,i}) \mathbb{I}(s' \in R_{j,i}), \end{aligned}$$

where the i*th tree's weight* w_i *is*

$$w_i = \gamma_i^2 \cdot \text{Var}\{c_i^{\text{Tree}}(s) : s \in S\}.$$

The GBT distance between a pair of points is then:

$$d^{\text{GBT}}(s, s') = 1 - \text{proximity}^{\text{GBT}}(s, s').$$

The choice of the L_2 norm to measure the size of a tree's contribution to the overall prediction has a natural equivalence to measuring the variance among the predictions made by $c_i^{\text{Tree}}(s)$. As an alternative to the L^2 norm used here, one may instead consider the L^1 norm, which we leave for future work. In Section 6.3, we study the implications of selecting this weight on the constructed distance.

3.2 Adaptive Prototype Selection Methods

We now introduce two new prototype selection methods that exploit approximation guarantees for submodular objective functions, and one that tries to directly optimize for accuracy.

Our goal is to find a good approximately optimal solution for the q-classwise k-medoids problem (3). We will achieve this by using a greedy algorithm on an appropriate non-negative, monotone, submodular function (see Prop. 1). However, the function (3) itself is not monotone submodular: in fact, adding more prototypes to

M *decreases* the value of $f(M)$. This can be averted by negating f, but then the function will take non-positive values. Therefore, adapting an idea of [13], we will define a related function g as

$$g(M) = f(P) - f(P \cup M), \tag{4}$$

where P is a set of *phantom exemplars*, one from each class. In order to get the best possible theoretical guarantee on the approximation (Section 4), this set needs to be chosen in a particular fashion. The resulting algorithm is Algorithm 1.

Algorithm 1: Adaptive greedy submodular prototype selection (SM-A)

```
  Input: Set of points $S$, distance function $d : S^2 \longrightarrow [0, 1]$,
         class assignment $c : S \longrightarrow [q]$
  Output: Set of prototypes $M$, $|M| = k$
1 Create set of phantom exemplars $P = \{p_1, \dots, p_q\}$ and set
    $d(p_i, s) = d(s, p_i) = 1$ for all $s$
2 $M \leftarrow \varnothing$
3 for $i$=1 to $k$ do
4     $s^* \leftarrow \arg\max_{s \in S} [f(P) - f(P \cup M \cup \{s\})]$
5     $M \leftarrow M \cup \{s^*\}$
```

We also consider a variant of this algorithm that we call weighted adaptive greedy submodular (SM-WA), in which each class is weighed differently: line 4 is replaced by

$$s^* \leftarrow \arg\max_{s \in S} \frac{1}{|C(s)|} \left[f(P) - f(P \cup M \cup \{s\}) \right],$$

where $C(S)$ denotes all points in S that are in the same class as s. It is easily verified that this objective function is also submodular.

3.3 Supervised Greedy Prototype Selection

Instead of optimizing the k-medoids value function f of equation (3), we can instead directly pick prototypes, in a greedy fashion, that yield the best (training or validation set) improvement in classification performance. The resulting method, which we call supervised greedy (SG), beats the unsupervised k-medoids approaches in terms of accuracy in several cases (Table 1), but we do not know of any theoretical guarantees that it satisfies, as these accuracy metrics are not submodular. This is nearly identical to Algorithm 1, except that line 1 is unnecessary and we replace line 4 with

$$s^* \leftarrow \arg\max_{s \in S} \left[\text{accuracy}(S, M \cup \{s\}) \right],$$

where accuracy denotes the accuracy metric used for evaluation.

4 THEORETICAL ANALYSIS

We will now briefly review the rationale behind the design of Algorithm 1 and derive an approximation guarantee for it. Optimization problems such as (3) are often approached using approximation algorithms that are guaranteed to find solutions within some factor of the optimum. Previous work on k-medoids [13, 39] has achieved this by identifying a related positive monotone submodular function and finding a good element of its domain by greedy search. Such an element is guaranteed to be within a factor of $(1 - 1/e)$ of the optimum for that function, where e is the Euler constant. We quickly review the relevant result.

Definition 3. *A function $f : \mathcal{P}(S) \longrightarrow \mathbb{R}$ that maps subsets of S to reals is* monotone *if $f(X) \leq f(Y)$ whenever $X \subseteq Y$. It is* submodular *if whenever $X \subseteq Y$, adding a particular element $s \in S$ to Y is not more useful than adding it to X:*

$$f(Y \cup \{s\}) - f(Y) \leq f(X \cup \{s\}) - f(X).$$

Proposition 1. *[40] Suppose $f : \mathcal{P}(S) \longrightarrow \mathbb{R}^+$ is a non-negative monotone submodular function. Let $T_0 = \varnothing$ and*

$$T_i = T_{i-1} \cup \arg\max_{s \in S} f(T_{i-1} \cup \{s\})$$

be the result of greedily maximizing f for i steps. Also, let

$$T_i^* = \arg\max_{T \subset S : |T| = k} f(T)$$

be the set of size i that maximizes f. Then

$$f(T_i) \geq (1 - 1/e) f(T_i^*).$$

We want to derive a similar approximation guarantee for Algorithm 1. To that end, we first need to show that g satisfies the necessary conditions.

Lemma 1. *The objective function (4) is non-negative, monotone and submodular.*

Proof. See appendix. □

By selecting the set of phantom exemplars P in such a fashion that $d(p, s) \geq d(s', s)$ for all $p \in P$ and $s, s' \in S$, we ensure that $f(T \cup P) = f(T)$ for all nonempty sets $T \subseteq S$. Hence, the set T_i^* that maximizes g among all sets of size i also minimizes f among all such sets.

Let T_i be the result of running the greedy maximization algorithm on (4) for i steps, and f be the original objective function (3). Then by Prop. 1 and choice of P,

$$f(T_i) \leq f(P) + (1 - 1/e)(f(T_i^*) - f(P)),$$

i.e. the approximation T_i takes us $1 - 1/e$ of the way from $f(P)$ to the optimum. Crucially, this means that the approximation guarantee depends on $f(P)$, i.e. how good the phantom exemplars alone would be as a solution to the q-classwise k-medoids problem.

Complexity analysis. Evaluating $f(M)$ takes time $O(|S||M| \cdot T(d))$, where $T(d)$ is the time to compute the distance $d(s, m)$ for a single pair of points. This computation can be made efficient by prepopulating an $|S| \times |S|$ matrix with all pairwise distances, and then simply implementing d as an array lookup. The same complexity bound applies to calculating accuracy(S, M), which iterates over $|S|$ points and finds the d-closest of $|M|$ medoids to check if it belongs to the correct class. The submodular (SM-A, SM-WA) and supervised greedy (SG) variants of Algorithm 1 essentially only differ in whether they invoke f or accuracy with an $O(k)$-sized set M of medoids in the arg max (Algorithm 1, Line 4). Either way, this arg max is over $|S|$ points, and the loop runs for k iterations. Therefore, all our instantiations of Algorithm 1 have time complexity $O(|S|^2 k^2 \cdot T(d))$.

5 RELATED WORK

Tree ensemble distance. Breiman and Cutler defined RF proximity in the documentation accompanying their software [4]. It is common to set distance as 1 − proximity [46, 50, 52], as we do in this paper. RF proximity has found a variety of applications, including clustering [46], outlier detection [53], imputation [47], etc., however less is known of its theoretical properties. The connection between random forests and kernel methods has been pointed out [33, 45] and proximity itself can be expressed as a kernel [35]. While we were inspired by RF distance, to the best of our knowledge, our paper presents the first proposal for GBT distance and the first method to seek prototypes for GBT models.

Not many RF implementations provide prototypes. The exceptions are the R `randomForest` package [31] and RAFT, a random forests visualization tool [4]. The RAFT documentation describes a heuristic prototype-finding procedure that is partially implemented in the `randomForest` package. It generates a single new point not from the dataset, a distinct goal from ours, which is to select a subset of representative points from existing points.

Prototype selection. There is a long line of literature on prototype selection methods, also known as instance reduction, data summarization, exemplar extraction, etc. We point the reader to the review by [12] that suggested that prototype selection methods can be grouped into three categories: condensation [19], edition [49], or hybrid methods that remove both noisy and redundant points from the prototype selection set. We briefly mention a few methods: k-medoids clustering is a classic problem for which different algorithms have been proposed, such as PAM [21] and greedy submodular approaches [13, 32], which we compare against and extend by adding the flexibility to choose varying numbers of prototypes by class. Kim et al. used a similar greedy submodular approach with maximum mean discrepancy objective to select prototypes and criticisms [23]; we provide a comparison to their prototype (not criticisms) selection method. We do not compare against set cover methods [2] as they tend to select significantly more prototypes than k-medoids [2] to achieve their objective of maximal coverage, at the cost of interpretability.

Point-based explanations. Besides feature-based explanations such as Shapley values [37] and LIME [43], point-based explanations have been proposed to explain model predictions. Examples include counterfactual explanations that determine the changes necessary to flip a point's prediction [48], models that automatically provide prototypes [24, 30], and identifying points most "influential" for a prediction [22, 25, 51]. There is a subtle distinction between prototype selection methods and influential point methods, as points that best represent a class (prototypes) may not be the most influential. Moreover, since trees are not differentiable except trivially within each node, influential point methods that typically take gradients of loss functions are not easily applicable. Hence we do not compare against them, but mention them for completeness.

The work most similar in spirit to ours is by Caruana et al. [6] who proposed to generate case-based explanations for non-case-based learning models such as neural networks and decision trees. However, unlike this paper, they do not take advantage of naturally-learned distance functions from these models.

6 EXPERIMENTAL RESULTS

We evaluate the proposed prototype selection methods and tree ensemble distances quantitatively as well as qualitatively. We also describe results from a user study that demonstrates that humans are able to use prototypes effectively.

Datasets. We use multiple image and tabular datasets with binary classification labels. For image datasets, we select two standard image classification benchmarks, MNIST and CALTECH-256. The goal in MNIST is to recognize handwritten digits [29]; the goal in CALTECH-256 is to predict one of 256 object categories for an image [16]. For both datasets, we select two classes that are either easily-confused [17] or visually-similar – digits 4 and 9 in MNIST, guitar and mandolin in CALTECH-256 – to evaluate our prototypes on not just easily predicted classes, but also classes commonly confused by the model. For MNIST, we use the raw pixel values as features. For CALTECH-256, we extracted deep features using a ResNet-50 model pre-trained on ImageNet.

For tabular datasets, we selected four datasets from critical domains such as healthcare and criminal justice where the need for interpretability has been suggested. These four datasets are: `sklearn` breastcancer and UCI diabetes, where the prediction task is to predict incidence of that disease, the Right Heart Catherization (RHC) dataset (http://biostat.mc.vanderbilt.edu/wiki/pub/Main/DataSets/rhc.html), on the impact of performing a medical procedure on patients, and T-COMPAS [9], a dataset that examines if COMPAS risk scores [28] agree with Mechanical Turk workers' predictions of recidivism [9]. The first two datasets are common tabular data benchmarks, and we selected the last two datasets because their prediction tasks are known to be hard [9], again to validate our prototypes on a diversity of cases, not just easy ones.

Metrics. Since some datasets are imbalanced, we use balanced accuracy [5] as our primary performance metric. We do not use ranking metrics such as AUC because nearest-neighbor classifiers do not output scores. We also count the number of prototypes selected.

Training and tuning tree ensembles. Whenever a fixed train-test split was not provided (i.e. for all datasets besides MNIST, where the training set has 60,000 images and the test set has 10,000 images), we created 60-20-20% training-validation-test splits. For RF, we use Python's `scikit-learn` package, training random forests with 1,000 trees without restricting maximum tree depth. We cross-validated the number of features to consider when looking for the best split ($\sqrt{p}$, $0.33p$, $0.5p$, $0.7p$, where p is the number of features, and the constant 7). For GBT, we modified `scikit-learn` to train GBT models with one gamma multiplier per tree. We cross-validated the number of trees (up to 200), maximum tree depth (3 to 5), and learning rate (0.1 or 0.01).

Implementations. For the comparison to Kim et al. [23], we use the authors' code for greedy prototypes (not criticisms) selection provided at https://github.com/BeenKim/MMD-critic. To obtain Shapley values [37] and associated graphs for the user study, we use the authors' Python Shap package which can be found at https://github.com/slundberg/shap.

Model	Prototype Selection Method		Breastcancer	Diabetes	T-COMPAS	RHC	MNIST 4-9	CALTECH256 G-M
RF	None (original tree ensemble)		0.92	0.72	0.56	0.68	0.97	0.81
	Baselines	1-NN	0.91 (341)	0.67 (460)	0.57 (600)	0.65 (3441)	**0.97** (3000)	0.78 (129)
		SM-U	**0.92** (12)	0.76 (5)	0.61 (10)	0.69 (11)	**0.97** (187)	0.83 (16)
		Kim et al [23]	**0.92** (19)	0.68 (36)	0.62 (32)	0.68 (3)	0.96 (65)	0.81 (6)
	Proposed	SG	0.90 (4)	**0.77** (5)	**0.68** (5)	0.66 (12)	**0.97** (18)	0.83 (3)
		SM-A	**0.92** (11)	**0.77** (4)	0.57 (15)	0.68 (9)	**0.97** (163)	0.79 (15)
		SM-WA	**0.92** (15)	**0.77** (6)	0.60 (13)	**0.69** (13)	**0.97** (243)	**0.86** (18)
GBT	None (original tree ensemble)		0.94	0.69	0.55	0.70	0.97	0.84
	Baselines	1-NN	0.92 (341)	0.70 (460)	0.58 (600)	0.65 (3441)	**0.97** (3000)	0.84 (129)
		SM-U	0.92 (21)	0.70 (12)	0.53 (26)	0.65 (62)	0.96 (249)	0.84 (11)
		Kim et al [23]	0.94 (6)	0.66 (16)	0.63 (53)	0.61 (33)	0.94 (45)	**0.86** (26)
	Proposed	SG	**0.95** (3)	**0.78** (4)	**0.67** (5)	**0.69** (15)	0.96 (23)	0.82 (2)
		SM-A	0.92 (22)	0.69 (12)	0.57 (4)	0.65 (4)	0.96 (247)	0.84 (11)
		SM-WA	0.92 (20)	0.69 (12)	0.56 (27)	0.65 (4)	0.96 (261)	0.84 (11)
EUCL	Baselines	1-NN	**0.91** (341)	0.68 (460)	0.53 (600)	0.59 (3441)	**0.96** (3000)	0.81 (129)
		SM-U	0.89 (19)	0.71 (26)	0.51 (62)	0.61 (40)	0.93 (313)	0.82 (6)
		Kim et al [23]	0.88 (60)	0.66 (29)	0.51 (49)	0.60 (5)	0.92 (384)	0.78 (64)
	Proposed	SG	0.87 (3)	**0.75** (4)	**0.58** (22)	**0.67** (19)	0.90 (65)	0.74 (9)
		SM-A	0.88 (18)	0.68 (3)	0.52 (51)	0.60 (8)	0.93 (377)	**0.88** (6)
		SM-WA	**0.91** (15)	0.73 (5)	0.51 (61)	0.61 (33)	0.93 (380)	**0.88** (6)

Table 1: Best test-set balanced accuracy with corresponding optimal number of prototypes, k, in parentheses. Three distances are provided: Random Forest (RF), Gradient Boosted Tree (GBT), and Euclidean (EUCL). We compare the proposed supervised greedy (SG), adaptive greedy submodular (SM-A), and weighted adaptive greedy submodular (SM-WA) prototype selection methods against the uniform greedy submodular (SM-U) and 1-NN baselines. We also compare our results with the greedy prototype selection method from Kim et al [23]. Best results for each dataset and distance in bold.

6.1 Quantitative Evaluation

We quantitatively evaluate the selected prototypes by using them in a *nearest-prototype classifier* [2, 23]. This is in line with recent ideas on evaluating explanations by checking their accuracy on independent test-data [43].

In general, the accuracy of each prototype selection method varies non-monotonically with k, the number of prototypes, suggesting that k should be tuned. Moreover, different selection methods operate at different regimes, e.g, supervised greedy (SG) obtains very good results with very few prototypes, but is sometimes outperformed by other methods when they use a larger number of prototypes. Comparing different selection methods using the same k may therefore not accurately characterize each method. Instead, we follow [2], tuning k separately for each prototype selection method and dataset, and comparing different methods at their optimal k.

It should be noted that in the limit, when k equals the size of the training set, any nearest-prototype classifier simply reduces to the 1-nearest-neighbors (1-NN) classifier. This classifier is hence one of our s, along with the original tree ensembles and the uniform greedy submodular (SM-U) approach. We also compare RF and GBT distance functions to Euclidean distance in feature space.

Table 1 summarizes the nearest-prototype classifier results. It provides test set balanced accuracy at the optimal number of prototypes, k, tuned separately for each prototype selection method. We make several observations:

(1) For all datasets besides MNIST 4-9, at least one prototype selection method outperformed 1-NN, suggesting the value of prototype selection not just for data condensation and interpretability (reducing the number of points that need to be shown to a user), but also classification accuracy.
(2) Tree ensemble distances, as supervised distances, tend to outperform Euclidean distance.
(3) SM-WA is competitive against SM-U.
(4) Despite the lack of theoretical guarantees and simplicity of the method, SG had clear advantages on a number of datasets with high achieving high accuracy.
(5) SG tends to select fewer prototypes than the other methods.
(6) At least one of our proposed prototype selection methods outperforms Kim et al. [23], with higher accuracy, lower prototype count, or both on each dataset.

6.2 Visualizing Distances and Prototypes

Figure 2 visualizes RF distance for the MNIST 4-9 dataset embedded in a two-dimensional space using t-sne [38] . On the left side are prototypes found by SM-A; on the right side are prototypes selected by SG.

We see that the prototypes selected by SM-A cover the space of points well, which is not the case for SG. We confirm this by computing the distance from each point to its nearest-prototype (SM-A: mean 0.23, sd 0.26; SG: mean 0.61, sd 0.23). Instead, the prototypes selected by SG are on the border between the two classes, and it is common to see these prototypes alternating (i.e. a 9 prototype followed by a nearby point of class 4 being selected as a prototype).

This has an intuitive explanation: as the only supervised prototype selection method, to maximize accuracy and minimize the chances of misclassification, SG selects discriminative prototypes that can separate points that are of different classes yet are close to each other, while the other proposed prototype selection methods, being non-supervised, do not select prototypes discriminatively.

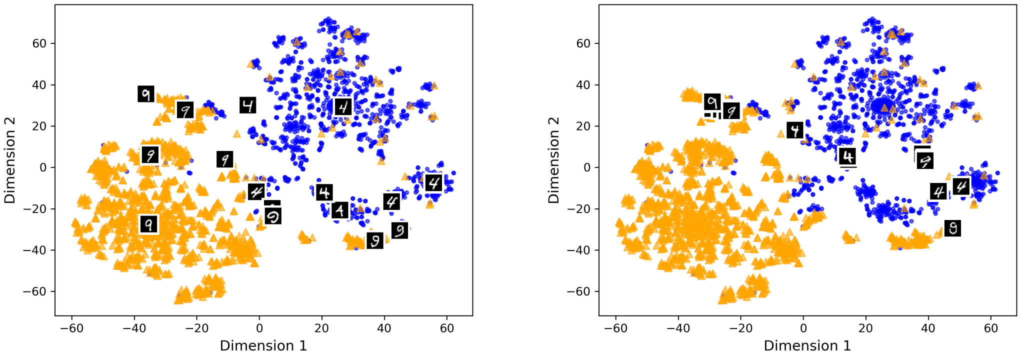


Figure 2: Visualization of distances using t-sne for optimal RF with mean depth 16 on the MNIST 4-9 dataset, using the adaptive greedy submodular (SM-A) prototype selection method (left) and supervised greedy (SG) (right) method. Orange represents the digit 9, blue represents the digit 4. and the black and white images are prototypes.

6.3 Understanding GBT Distance

Many default implementations of RF algorithms allow trees to grow to unrestricted depth [3]. As a consequence, on any given dataset, the trees in RF models tend to be deeper than those in GBT models. Table 2, which presents statistics of tree depth in RF and GBT models, confirms this. The shallower the tree, the fewer leaves, and the higher the probability of a pair of points ending up in the same leaf. With larger datasets, conversely, RF trees tend to get deeper (Table 2). However, the GBT trees we generate remain limited to depth 3 to 5. Hence, the larger the dataset, the more different we expect RF and GBT distances to be.

		GBT	RF Depth			
Dataset	n	Depth	Min	Mean	Max	Var
Breastcancer	569	3	2	3.02	4	0.40
Diabetes	768	3	5	7.36	12	1.04
T-COMPAS	1000	4	6	8.70	14	1.18
RHC	5735	3	11	14.7	21	1.41
MNIST 4-9	5000	3	8	10.95	16	1.44
CAL256 G-M	215	2	2	2.51	3	0.50

Table 2: Statistics of RF and GBT models tree depth across different datasets. n is the number of observations in the dataset. All RF models had 1000 trees. All GBT models had an optimal number of trees (based on validation set loss) less than or equal to 200.

Figure 3 visualizes RF and GBT distances embedded in a two-dimensional space using t-sne and several MNIST prototypes. While digits 4 and 9 are clearly separable in Figure 3, consistent with the high performance (97% accuracy in Table 1) of the RF and GBT models, the models appear to be learning different representations, with GBT grouping points together in smaller and more compact clusters than RF. Accordingly, the prototypes selected for the RF distance are also different from those selected for GBT.

A natural next question may be the following: to what degree are differences between GBT and RF distances caused by different tree depth, different weights used in constructing the distance, or that different patterns in the data are being learned by RF compared to GBT models? While the top right corner of Figure 3 visualizes distances from GBT models trained with default settings (short), and the bottom right corner of the same figures depicts distances from RF models trained with default settings (unrestricted depth), the bottom left corner shows RF models trained *to the same depth* as the corresponding default GBT model on that dataset. While the short RF model has smaller and more compact clusters than the default RF model, the RF and GBT models of same depth can still be told apart.

Finally, the top left corner of Figure 3 visualizes an unweighted distance function derived from the same GBT model as in the top right corner, which uses a weighted distance function. This unweighted GBT distance (top left corner) looks more similar to the default RF model's distance (bottom right corner).

6.4 Evaluating Interpretability: User Study

To evaluate if prototypes are interpretable to humans – the intended end users of our method – we follow the human-grounded evaluation framework outlined by Doshi-Velez and Kim [8], and design a user study where the task is to predict what the model would predict, after being presented with an explanation and inputs to the model. This task is exactly the "forward simulation/prediction" experiment described by Doshi-Velez and Kim [8], and fulfills the simulatability criterion for interpretability, one of several criteria proposed by Lipton [34].

We compare prototypes to TreeExplainer [36], a Shapley values feature attribution method [37] for tree ensembles. Here, we compare to Shapley values because it is a state-of-the-art feature attribution method that satisfies several axiomatic guarantees [37], is popular in practice [1], and, as a feature-based explanation, is presented differently to users. Hence, this comparison can inform us whether prototypes are indeed a viable alternative to feature-based explanations for tree ensembles. The hypothesis we investigate in this user study is therefore whether users are able to correctly predict a tree ensemble model's output using prototypes, and moreover, if they are able to do so with greater accuracy than when using Shapley values.

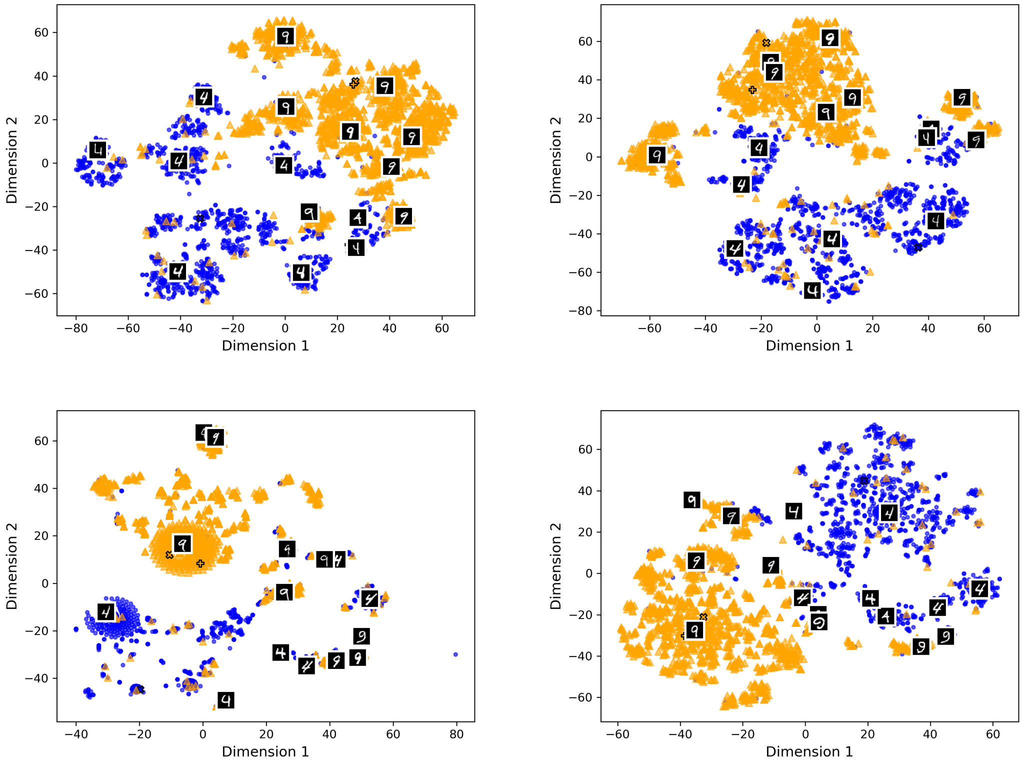


Figure 3: Visualization of distances using t-sne for optimal GBT with unweighted trees (top left), trees weighted by vg^2 (top right), RF with short trees matching GBT depth of 3 (bottom left), and optimal RF (bottom right) with mean depth 16 on the MNIST 4-9 dataset, using the adaptive greedy submodular (SM-A) prototype selection method. Orange represents the digit 9, blue represents the digit 4. The two points marked by x and + are the same across all subfigures, to indicate how points move across different distance representations. Note that the bottom right subfigure is the same as the left subfigure in Figure 2.

6.4.1 User Study Design. We recruited participants on Amazon Mechanical Turk to participate in the study. We selected a dataset on vehicle fuel efficiency, from the R ggplot2 package (https://ggplot2.tidyverse.org/reference/mpg.html). The label is whether the vehicle is fuel efficient (greater than 19 highway miles per gallon), which does not require expert domain knowledge to understand.

Each user was randomly assigned to either the prototypes or Shapley condition, with no users in both conditions. Users were presented with model inputs (Figure A1 in the appendix); users in the prototype condition were presented with prototype explanations (Figure A3) obtained by applying our prototype selection method to a tree ensemble; users in the Shapley condition were presented with a set of Shapley plots (Figure A4) with added guidelines on how to interpret the plots. Then, users were presented a question (Figure A2) and asked to predict what the model would predict for that vehicle. Each user was asked to evaluate 13 vehicles. These vehicles were randomly selected from the test set, while ensuring that every combination of 13 vehicles seen by a user in the prototype condition was also seen by another user in the Shapley condition, to account for certain vehicles being more difficult to predict according to either the model or human intuition.

To ensure that participants were paying attention and trying to answer the questions to the best of their ability, we designed two of these 13 questions to be ability and attention checks, known to help in identifying inattentive participants [14]. In particular, for the attention check, users were asked to select "fuel efficient" for a specific vehicle, regardless of what they believe the true answer is. Users were compensated $3.00 upon completion of the study. After removing users who failed either catch trial, 42 users remained.

6.4.2 User Study Results. We evaluate the results using a **human accuracy** metric: the fraction of vehicles where users predicted correctly the model's prediction for that vehicle. We also report user responses to a question at the end of the survey about how confident they felt about their answers.

Table 3 presents the results of two-sample t-tests comparing these metrics for the prototype and Shapley conditions. The p-values are one-sided, testing the alternative hypothesis if the metric is greater in the prototype condition than Shapley condition (i.e. higher accuracy, or greater confidence). The average self-reported confidence (on a scale of 1-4, higher is more confident) among users who used prototypes was 2.783, with users who used Shapley values self-reporting confidence of 2.736. This difference was not statistically significant. We note that self-reported user confidence has not been found to be indicative of actual performance and can sometimes even be misleading [15, 27], with studies finding that humans cannot accurately assess their own performance [26, 42]. In contrast, there is a statistically significant improvement in human accuracy when using prototypes compared to Shapley values (0.79 compared to 0.72 human accuracy; p-value 0.035), demonstrating that prototypes are a viable alternative to feature-based explanation methods.

Metrics	Prototypes	Shapley values	t-test p-value
Human accuracy	0.79	0.72	0.035*
Confidence	2.783	2.736	0.238

Table 3: Results from user study. Statistically significant differences are marked by *. The difference in human accuracy between prototype and Shapley conditions is statistically significant, with a one-sided t-test with alternative hypothesis that accuracy is higher in the prototype condition than Shapley condition returning p-value of 0.035.

We also collected qualitative feedback on the explanations and user interface. When asked "Was anything confusing? Is there anything you would have liked to know that would have helped you better answer these questions?" at the end of the survey, one user in the Shapley values condition responded "It would have helped if [Shapley values] showed the difference better", and another user reported "The way the values were weighted seemed a bit strange to me". On the other hand, in response to the same question, one user in the prototypes condition was able to articulate a simple mental model of what s/he thought the model was doing, saying "Knowing the make and model of the vehicle would have been helpful, but maybe I think that because I'm familiar with cars. (...) [The model] seems to think that vehicles with smaller engines will automatically be more fuel efficient, which is why I made the choices that I did because the instructions said to guess the predictions of the model.".

6.5 A Use Case of Fixing Mislabeled Points

We now demonstrate how the ideas presented in this paper can assist with certain tasks that machine learning model users may wish to leverage explanations for, such as debugging a dataset. We focus on the specific task of correcting mislabeled points in a dataset, where we wish to present a reasonable number of points to the user, in some meaningful ordering, for the user to correct any wrong labels. This experiment is similar to that ran by [25, 51].

We corrupt the MNIST 4-9 dataset by flipping the labels of 33% of points, and use RF distance to construct a ranking of points, which we compare to two baselines: (1) random ranking; (2) ranking based on loss of training points. The distance ranking is constructed thus: for every training point, compute $k = 10$ nearest neighbors based on the distance, then compute the proportion of neighbors that share the same label as the point. Note that the loss ranking is strong baseline, as found by [25].

We present a simulated user with a proportion (up to 30%) of training points, as ranked by the different methods, to inspect and correct. Similar to [25, 51], the simulated user is an oracle who only corrects points that are flipped, of all the points presented to her. The model is then retrained on the corrected data. We repeat the experiment 20 times, randomizing the points corrupted each time, and average the results. Figure 4 plots the mean test set performance of the model retrained on simulated user corrected data. With the same interpretability budget (amount of points the simulated user had to inspect), tree ensemble distance based ranking was better at assisting in correcting mislabeled points, generating corrected data that had higher test set accuracy than other rankings.

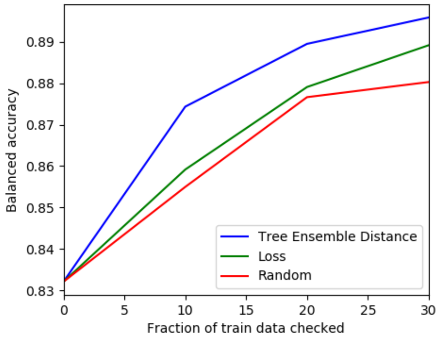


Figure 4: Performance of different algorithms, including one based on tree ensemble distance, to select points from MNIST 4-9 with corrupted labels for a simulated user to inspect. The simulated user flips the labels of points checked, and a model is retrained on the corrected data.

7 CONCLUDING REMARKS

We have proposed that tree ensemble classifiers can be made interpretable using prototypes that are central according to a distance function derived from the tree ensemble. Our user study suggests that this is indeed the case, as humans were better at predicting the output of a tree ensemble classifier using prototypes than Shapley values, a popular feature-based approach. At the same time, our quantitative evaluation (Table 1) suggests that in many cases, tree space prototypes are able to capture much of the power of the original classifier, or sometimes even constitute a superior separate classifier, since these prototypes when used as a nearest-prototype classifier sometimes outperform the original tree ensemble. This suggests that to the extent that tree ensemble proximity is an intuitive measure of similarity, users with longer-term experience with prototypes may attain even better accuracy at anticipating the behavior of the tree ensemble. Another intriguing possibility is that we ought to consider a slightly different question: if a nearest-prototype classifier is the superior classifier, should we perhaps analyse its interpretability in its own right, as opposed to the original tree ensemble?

Another open question is the impact of the number of prototypes on interpretability, as opposed to accuracy alone. Intuition suggests that humans should be better at dealing with a smaller number of prototypes, but how does this trade off against the generally greater accuracy nearest-prototype classifiers tend to exhibit with more prototypes? Subjectively, our adaptive prototype selection methods that can select different numbers of prototypes per class tend to lead to better coverage for classes that are more diverse, but our analysis shows that there do exist some datasets on which they do not benefit classification. A quantitative analysis of the relationship between prototype count and human predictions in multiple settings could provide further arguments for or against adaptive methods, and constitutes an attractive direction for future work.

ACKNOWLEDGMENTS

We thank Jacob Bien and Albert Gordo for helpful discussion. GH was supported by NSF grant DMS-1712554. MTW was supported by NIH grants R01 GM135926 and U19 AI111143.

REFERENCES

[1] Umang Bhatt, Alice Xiang, Shubham Sharma, Adrian Weller, Ankur Taly, Yunhan Jia, Joydeep Ghosh, Ruchir Puri, Jose MF Moura, and Peter Eckersley. 2020. Explainable machine learning in deployment. In *FAT**.

[2] Jacob Bien and Robert Tibshirani. 2011. Prototype selection for interpretable classification. *The Annals of Applied Statistics* (2011).

[3] Leo Breiman. 2001. Random forests. *Machine Learning* 45, 1 (2001).

[4] Leo Breiman and Adele Cutler. 2002. Random Forests Manual. https://www.stat.berkeley.edu/~breiman/RandomForests. Accessed July 6, 2019. Year 2002 based on copyright year indicated in the authors' Fortran code.

[5] Kay Henning Brodersen, Cheng Soon Ong, Klaas Enno Stephan, and Joachim M Buhmann. 2010. The Balanced Accuracy and Its Posterior Distribution. In *International Conference on Pattern Recognition*.

[6] Rich Caruana, Hooshang Kangarloo, John David Dionisio, Usha Sinha, and David Johnson. 1999. Case-based explanation of non-case-based learning methods.. In *Proceedings of the AMIA Symposium*. American Medical Informatics Association.

[7] Rich Caruana and Alexandru Niculescu-Mizil. 2006. An Empirical Comparison of Supervised Learning Algorithms. In *ICML*.

[8] Finale Doshi-Velez and Been Kim. 2017. Towards A Rigorous Science of Interpretable Machine Learning. *arXiv preprint arXiv:1702.08608* (2017).

[9] Julia Dressel and Hany Farid. 2018. The accuracy, fairness, and limits of predicting recidivism. *Science Advances* (2018). Data accessed from www.cs.dartmouth.edu/farid/downloads/publications/scienceadvances17.

[10] Alex A Freitas. 2014. Comprehensible classification models: a position paper. *ACM SIGKDD Explorations* (2014).

[11] Jerome H Friedman. 2001. Greedy function approximation: a gradient boosting machine. *Annals of Statistics* (2001).

[12] Salvador Garcia, Joaquin Derrac, Jose Ramon Cano, and Francisco Herrera. 2011. Prototype selection for nearest neighbor classification: Taxonomy and empirical study. *TPAMI* (2011).

[13] Ryan Gomes and Andreas Krause. 2010. Budgeted Nonparametric Learning from Data Streams. In *ICML*.

[14] Joseph K Goodman, Cynthia E Cryder, and Amar Cheema. 2013. Data Collection in a Flat World: The Strengths and Weaknesses of Mechanical Turk Samples. *Journal of Behavioral Decision Making* 26 (2013).

[15] Ben Green and Yiling Chen. 2019. Disparate interactions: An algorithm-in-the-loop analysis of fairness in risk assessments. In *FAT**.

[16] Gregory Griffin, Alex Holub, and Pietro Perona. 2006. Caltech-256 Object Category Dataset. *Technical Report, California Institute of Technology* (2006).

[17] Raia Hadsell, Sumit Chopra, and Yann LeCun. 2006. Dimensionality Reduction by Learning an Invariant Mapping. In *CVPR*.

[18] Satoshi Hara and Kohei Hayashi. 2018. Making tree ensembles interpretable: A Bayesian Model Selection Approach. In *AISTATS*.

[19] Peter Hart. 1968. The condensed nearest neighbor rule. *IEEE Transactions on Information Theory* (1968).

[20] Hemant Ishwaran. 2007. Variable importance in binary regression trees and forests. *Electronic Journal of Statistics* (2007).

[21] Leonard Kaufman and Peter J Rousseeuw. 1987. Clustering by means of medoids. In *Statistical Data Analysis Based on the L1 Norm*. Birkhäuser Basel.

[22] Rajiv Khanna, Been Kim, Joydeep Ghosh, and Oluwasanmi Koyejo. 2019. Interpreting black box predictions using fisher kernels. In *AISTATS*.

[23] Been Kim, Rajiv Khanna, and Oluwasanmi O Koyejo. 2016. Examples are not enough, learn to criticize! criticism for interpretability. In *NIPS*.

[24] Been Kim, Cynthia Rudin, and Julie A Shah. 2014. The Bayesian Case Model: A generative approach for case-based reasoning and prototype classification. In *NIPS*.

[25] Pang Wei Koh and Percy Liang. 2017. Understanding black-box predictions via influence functions. In *ICML*.

[26] Vivian Lai and Chenhao Tan. 2019. On human predictions with explanations and predictions of machine learning models: A case study on deception detection. In *FAT**.

[27] Himabindu Lakkaraju and Osbert Bastani. 2020. " How do I fool you?" Manipulating User Trust via Misleading Black Box Explanations. In *AIES*.

[28] Jeff Larson, Surya Mattu, Lauren Kirchner, and Julia Angwin. 2016. How We Analyzed the COMPAS Recidivism Algorithm. ProPublica.

[29] Yann LeCun, Leon Bottou, Yoshua Bengio, and Patrick Haffner. 1998. Gradient-based learning applied to document recognition. In *Proceedings of the IEEE*.

[30] Oscar Li, Hao Liu, Chaofan Chen, and Cynthia Rudin. 2018. Deep learning for case-based reasoning through prototypes: A neural network that explains its predictions. In *AAAI*.

[31] Andy Liaw and Matthew Wiener. 2002. Classification and regression by randomForest. *R News* (2002).

[32] Hui Lin and Jeff Bilmes. 2011. A class of submodular functions for document summarization. In *ACL*.

[33] Yi Lin and Yongho Jeon. 2006. Random forests and adaptive nearest neighbors. *J. Amer. Statist. Assoc.* (2006).

[34] Zachary C. Lipton. 2016. The Mythos of Model Interpretability. *arXiv preprint arXiv:1606.03490* (2016).

[35] Gilles Louppe. 2014. Understanding random forests: From theory to practice. *arXiv preprint arXiv:1407.7502* (2014).

[36] Scott M Lundberg, Gabriel Erion, Hugh Chen, Alex DeGrave, Jordan M Prutkin, Bala Nair, Ronit Katz, Jonathan Himmelfarb, Nisha Bansal, and Su-In Lee. 2020. From local explanations to global understanding with explainable AI for trees. *Nature Machine Intelligence* 2 (2020).

[37] Scott M Lundberg and Su-In Lee. 2017. A Unified Approach to Interpreting Model Predictions. In *NIPS*.

[38] Laurens van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-SNE. *JMLR* (2008).

[39] Baharan Mirzasoleiman, Amin Karbasi, Rik Sarkar, and Andreas Krause. 2013. Distributed submodular maximization: Identifying representative elements in massive data. In *NIPS*.

[40] George L Nemhauser, Laurence A Wolsey, and Marshall L Fisher. 1978. An analysis of approximations for maximizing submodular set functions—I. *Mathematical Programming* (1978).

[41] Christos H Papadimitriou. 1981. Worst-Case and Probabilistic Analysis of a Geometric Location Problem. *SIAM J. Comput.* (1981).

[42] Forough Poursabzi-Sangdeh, Daniel G Goldstein, Jake M Hofman, Jennifer Wortman Vaughan, and Hanna Wallach. 2018. Manipulating and measuring model interpretability. *arXiv preprint arXiv:1802.07810* (2018).

[43] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. 2016. "Why Should I Trust You?": Explaining the Predictions of Any Classifier. In *KDD*.

[44] Michael M Richter and Agnar Aamodt. 2005. Case-based reasoning foundations. *The Knowledge Engineering Review* (2005).

[45] Erwan Scornet. 2016. Random forests and kernel methods. *IEEE Transactions on Information Theory* (2016).

[46] Tao Shi and Steve Horvath. 2006. Unsupervised learning with random forest predictors. *Journal of Computational and Graphical Statistics* (2006).

[47] Daniel J Stekhoven. 2015. missForest: Nonparametric missing value imputation using random forest. *Astrophysics Source Code Library* (2015).

[48] Sandra Wachter, Brent Mittelstadt, and Chris Russell. 2017. Counterfactual explanations without opening the black box: Automated decisions and the GDPR. *Harvard Journal of Law & Technology* 31, 2 (2017).

[49] Dennis L Wilson. 1972. Asymptotic properties of nearest neighbor rules using edited data sets. *Transactions on Systems, Man and Cybernetics* (1972).

[50] Caiming Xiong, David Johnson, Ran Xu, and Jason J Corso. 2012. Random forests for metric learning with implicit pairwise position dependence. In *KDD*.

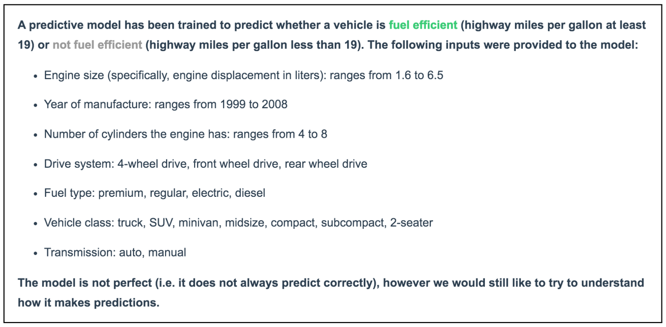
[51] Chih-Kuan Yeh, Joon Kim, Ian En-Hsu Yen, and Pradeep K Ravikumar. 2018. Representer point selection for explaining deep neural networks. In *NeurIPS*.

[52] Peng Zhao, Xiaogang Su, Tingting Ge, and Juanjuan Fan. 2016. Propensity score and proximity matching using random forest. *Contemporary Clinical Trials* (2016).

[53] Qi-Feng Zhou, Hao Zhou, Yong-Peng Ning, Fan Yang, and Tao Li. 2015. Two approaches for novelty detection using random forest. *Expert Systems with Applications* (2015).

[54] Yichen Zhou, Zhengze Zhou, and Giles Hooker. 2018. Approximation trees: Statistical stability in model distillation. *arXiv preprint arXiv:1808.07573* (2018).

A USER STUDY MATERIALS

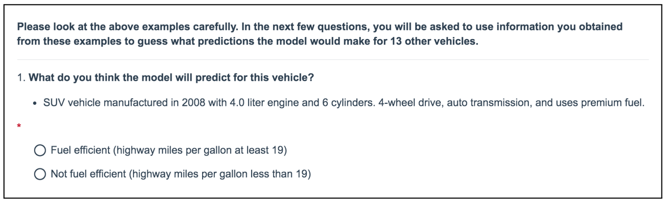


A predictive model has been trained to predict whether a vehicle is fuel efficient (highway miles per gallon at least 19) or not fuel efficient (highway miles per gallon less than 19). The following inputs were provided to the model:

- Engine size (specifically, engine displacement in liters): ranges from 1.6 to 6.5
- Year of manufacture: ranges from 1999 to 2008
- Number of cylinders the engine has: ranges from 4 to 8
- Drive system: 4-wheel drive, front wheel drive, rear wheel drive
- Fuel type: premium, regular, electric, diesel
- Vehicle class: truck, SUV, minivan, midsize, compact, subcompact, 2-seater
- Transmission: auto, manual

The model is not perfect (i.e. it does not always predict correctly), however we would still like to try to understand how it makes predictions.

Figure A1: Model inputs presented to users in both prototype and Shapley conditions.



Please look at the above examples carefully. In the next few questions, you will be asked to use information you obtained from these examples to guess what predictions the model would make for 13 other vehicles.

1. What do you think the model will predict for this vehicle?

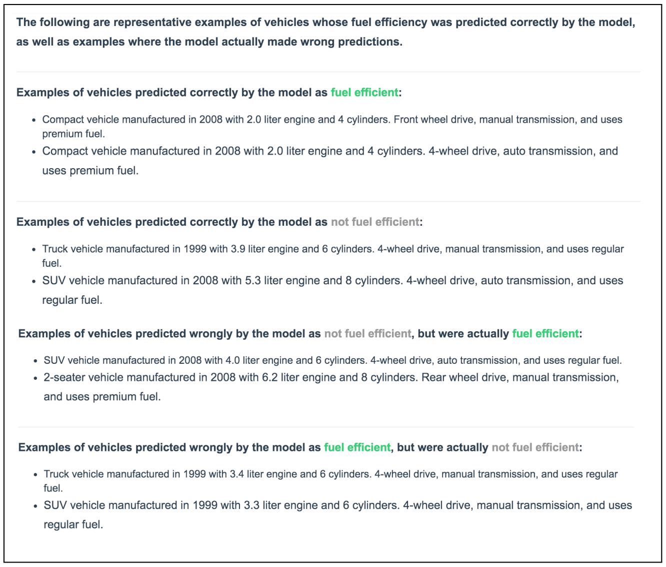
- SUV vehicle manufactured in 2008 with 4.0 liter engine and 6 cylinders. 4-wheel drive, auto transmission, and uses premium fuel.

*

○ Fuel efficient (highway miles per gallon at least 19)

○ Not fuel efficient (highway miles per gallon less than 19)

Figure A2: An example question answered by users in both conditions. Each question represents a vehicle. Each user is asked to evaluate 13 such questions.



The following are representative examples of vehicles whose fuel efficiency was predicted correctly by the model, as well as examples where the model actually made wrong predictions.

Examples of vehicles predicted correctly by the model as fuel efficient:

- Compact vehicle manufactured in 2008 with 2.0 liter engine and 4 cylinders. Front wheel drive, manual transmission, and uses premium fuel.
- Compact vehicle manufactured in 2008 with 2.0 liter engine and 4 cylinders. 4-wheel drive, auto transmission, and uses premium fuel.

Examples of vehicles predicted correctly by the model as not fuel efficient:

- Truck vehicle manufactured in 1999 with 3.9 liter engine and 6 cylinders. 4-wheel drive, manual transmission, and uses regular fuel.
- SUV vehicle manufactured in 2008 with 5.3 liter engine and 8 cylinders. 4-wheel drive, auto transmission, and uses regular fuel.

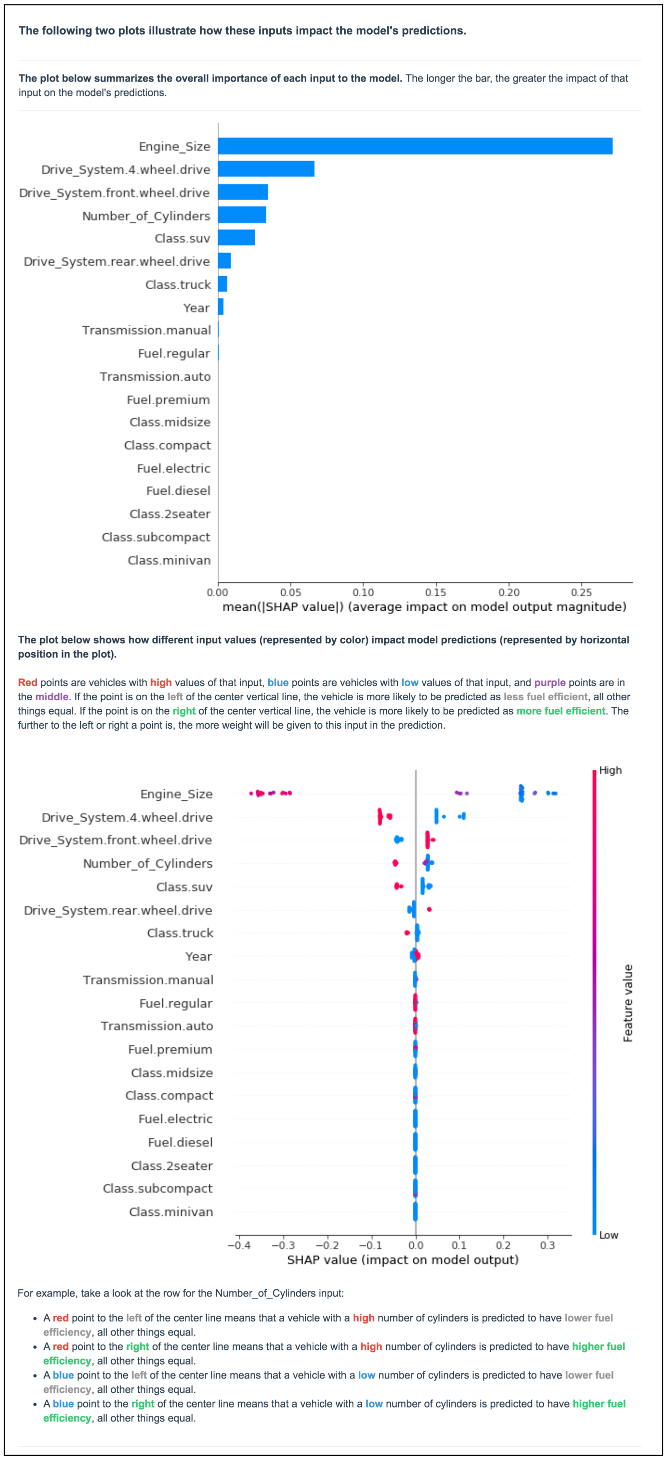
Examples of vehicles predicted wrongly by the model as not fuel efficient, but were actually fuel efficient:

- SUV vehicle manufactured in 2008 with 4.0 liter engine and 6 cylinders. 4-wheel drive, auto transmission, and uses regular fuel.
- 2-seater vehicle manufactured in 2008 with 6.2 liter engine and 8 cylinders. Rear wheel drive, manual transmission, and uses premium fuel.

Examples of vehicles predicted wrongly by the model as fuel efficient, but were actually not fuel efficient:

- Truck vehicle manufactured in 1999 with 3.4 liter engine and 6 cylinders. 4-wheel drive, manual transmission, and uses regular fuel.
- SUV vehicle manufactured in 1999 with 3.3 liter engine and 6 cylinders. 4-wheel drive, manual transmission, and uses regular fuel.

Figure A3: Prototypes presented to users in prototypes condition.



The following two plots illustrate how these inputs impact the model's predictions.

The plot below summarizes the overall importance of each input to the model. The longer the bar, the greater the impact of that input on the model's predictions.

The plot below shows how different input values (represented by color) impact model predictions (represented by horizontal position in the plot).

Red points are vehicles with high values of that input, blue points are vehicles with low values of that input, and purple points are in the middle. If the point is on the left of the center vertical line, the vehicle is more likely to be predicted as less fuel efficient, all other things equal. If the point is on the right of the center vertical line, the vehicle is more likely to be predicted as more fuel efficient. The further to the left or right a point is, the more weight will be given to this input in the prediction.

For example, take a look at the row for the Number_of_Cylinders input:

- A red point to the left of the center line means that a vehicle with a high number of cylinders is predicted to have lower fuel efficiency, all other things equal.
- A red point to the right of the center line means that a vehicle with a high number of cylinders is predicted to have higher fuel efficiency, all other things equal.
- A blue point to the left of the center line means that a vehicle with a low number of cylinders is predicted to have lower fuel efficiency, all other things equal.
- A blue point to the right of the center line means that a vehicle with a low number of cylinders is predicted to have higher fuel efficiency, all other things equal.

Figure A4: Shapley values feature attribution plots presented to users in Shapley condition.

B PROOF OF LEMMA 1

Lemma 1. *The objective function (4) is non-negative, monotone and submodular.*

Proof (Lemma 1). Observe that whenever $X \subseteq Y$, we have $f(X) \geq f(Y)$, since adding more points to a set can only make the closest point to a given point closer. From this, monotonicity and non-negativity is immediate, since $f(P) \geq f(P \cup M)$.

To establish submodularity, we will show that the function f of (3) satisfies

$$f(Y) - f(Y \cup \{t\}) \leq f(X) - f(X \cup \{t\})$$

whenever $X \subseteq Y \subseteq S$. The inequality of definition 3 then follows for g by plugging into its definition (4).

For any point $s \in S$, define $p_M(s)$ to be the closest point to s in M of the same class, that is,

$$p_M(s) = \underset{m \in M : c(m) = c(s)}{\arg\min} \quad d(s, m).$$

Then we can rewrite $f(M)$ as

$$\sum_{s \in S} d(s, p_M(s)),$$

and it suffices to show that

$$\begin{aligned} & d(s, p_Y(s)) - d(s, p_{Y \cup \{t\}}(s)) \\ \leq \quad & d(s, p_X(s)) - d(s, p_{X \cup \{t\}}(s)). \end{aligned}$$

for all $s \in S$. Both sides of this inequality are non-negative (+), since adding points can only shorten the distance to the closest point. Suppose $p_{Y \cup \{t\}}(s) \in Y$. Then it must be equal to $p_Y(s)$, since the closest point is present in Y, and so the first line is 0, and the inequality follows from (+).

Suppose instead $p_{Y \cup \{t\}}(s) \notin Y$. Then it must be t. So $p_{X \cup \{t\}}(s) = t$ as well (since $X \subseteq Y$), and the inequality reduces to $d(s, p_Y(s)) \leq d(s, p_X(s))$. But this is immediate, since $Y \supseteq X$ and adding more points can only shorten the distance to the closest point. □