Learning Interpretable Models Using an Oracle

Abhishek Ghose   ABHISHEK.GHOSE.82@GMAIL.COM
Dept. of Computer Science and Engineering, IIT Madras
Chennai, India

Balaraman Ravindran   RAVI@CSE.IITM.AC.IN
Dept. of Computer Science and Engineering, IIT Madras
Robert Bosch Centre for Data Science and AI, IIT Madras
Chennai India

Abstract

As Machine Learning (ML) becomes pervasive in various real world systems, the need for models to be understandable, either by being *interpretable* or *explainable*, has increased. We focus on interpretability, noting that models often need to be constrained in size for them to be considered interpretable, e.g., a decision tree of depth 5 is easier to interpret than one of depth 50. But smaller models also tend to have high bias. This suggests a trade-off between interpretability and accuracy. We propose a *model agnostic* technique to minimize this trade-off. Our strategy is to first learn a powerful, possibly black-box, probabilistic model - which we refer to as the *oracle* - on the training data. Uncertainty in the oracle's predictions are used to learn a sampling distribution for the training data. The interpretable model is trained on a sample obtained using this distribution. We demonstrate that such a model often is significantly more accurate than one trained on the original data.

Determining the sampling strategy is formulated as an optimization problem. Our solution to this problem possesses the following key favorable properties: (1) the number of optimization variables is independent of the dimensionality of the data: a fixed number of seven variables are used (2) our technique is *model agnostic* - in that both the interpretable model and the oracle may belong to arbitrary model families.

Results using multiple real world datasets, using *Linear Probability Models* and *Decision Trees* as interpretable models, with *Gradient Boosted Model* and *Random Forest* as oracles, are presented. We observe significant relative improvements in the *F1-score* in most cases, occassionally seeing improvements greater than 100%. Additionally, we discuss an interesting application of our technique where a *Gated Recurrent Unit* network is used as an oracle to improve the sequence classification accuracy of a Decision Tree that uses character n-grams as features.

1. Introduction

In recent years, Machine Learning (ML) models have become increasingly pervasive in various real world systems. In many of these applications, such as movie and product recommendations, it is sufficient that the ML model is accurate. However, there is a growing emphasis on models to be *understandable* as well, in domains where the cost of being wrong is prohibitively high, e.g., medicine and healthcare (Caruana, Lou, Gehrke, Koch, Sturm, & Elhadad, 2015; Ustun & Rudin, 2016), defence applications (Gunning, 2016), law enforcement (Angwin, Larson, Mattu, & Kirchner, 2016; Larson, Mattu, Kirchner, & Angwin, 2016) and banking (Castellanos & Nash, 2018). It is expected that soon model trans-

parency would be mandated by law within systems involving digital interactions (Goodman & Flaxman, 2017; Clarke, 2019).

Contemporary research in this area may be categorized into two broad approaches:

1. *Interpretability*: this area looks at building models that are considered easy to understand as-is, e.g., rule lists (Letham, Rudin, McCormick, & Madigan, 2013; Angelino, Larus-Stone, Alabi, Seltzer, & Rudin, 2017), decision trees (Breiman et al., 1984; Quinlan, 1993; Quinlan, 2004; Hu, Rudin, & Seltzer, 2019), sparse linear models (Ustun & Rudin, 2016), decision sets (Lakkaraju, Bach, & Leskovec, 2016), rule sets (Wang, 2018), pairwise interaction models that may be linear (Lim & Hastie, 2015) or additive (Lou, Caruana, Gehrke, & Hooker, 2013), task-specific interpretable models like neural-symbolic models for visual question-answering (Yi, Wu, Gan, Torralba, Kohli, & Tenenbaum, 2018), rules for negation scope detection in natural language (Pröllochs, Feuerriegel, & Neumann, 2019).

2. *Explainability*: this area looks at techniques that may be used to understand the workings of models that do not naturally lend themselves to a simple interpretation, e.g., locally interpretable explanations as provided by LIME and Anchors (Ribeiro, Singh, & Guestrin, 2016, 2018), visual explanations for Convolutional Neural Networks such as Grad-CAM (Selvaraju, Cogswell, Das, Vedantam, Parikh, & Batra, 2017) and Ablation-CAM (Desai & Ramaswamy, 2020), influence functions (Koh & Liang, 2017), feature attribution based on Shapley values (Lundberg & Lee, 2017; Ancona, Oztireli, & Gross, 2019).

We focus on interpretablity in this work; specifically, in improving the accuracy of a model from an arbitrary model family that is considered interpretable, e.g., decision trees, linear models, rules.

Interpretable models are preferably small in *size*: this is anecdotally seen in how a linear model with 10 terms may be preferred over one with 100 terms, or in how a decision tree (DT) of $depth = 5$ is easier to understand than one of $depth = 50$. This property is variously acknowledged in the area of interpretability: Herman (2017) refers to this as low *explanation complexity*, this is seen as a form of *simulability* in Lipton (2018), and is often listed as a desirable property in interpretable model representations (Ribeiro et al., 2016; Lakkaraju et al., 2016; Angelino et al., 2017; Bertsimas, Delarue, Jaillet, & Martin, 2019). The preference for small-sized models points to an obvious problem: since size is usually inversely proportional to model bias, such a model often trades accuracy for interpretability.

We propose a novel adaptive sampling technique to minimize this trade-off.

We first learn a highly accurate, possibly black-box, *probabilistic* model - referred to as the *oracle* - on our training data. There are no size constraints imposed on the oracle and it is assumed to be more accurate than our interpretable model. We refer to a model as "probabilistic" if it can produce a probability distribution over labels during prediction:

$$p(y_i|x), \forall y_i \in \{1, 2, ..., C\} \tag{1}$$

Here, $\{1, 2, ..., C\}$ is the set of labels. The probabilities $p(y_i|x)$ are commonly construed as *confidences* of predicting labels y_i for instance x.

Next, we try to incorporate the oracle's view of the input space into our interpretable model. The mechanism we use is to sample points from the training data based on how *uncertain* the oracle is in their classification. We make the non-trivial observation here that the optimal sampling probabilities are not directly proportional to uncertainties in general, and their relationship needs to be learned.

The interpretable model is then trained on this sample. Our experiments demonstrate this often leads to significant improvements in the classification accuracy, especially when the interpretable model size is small.

Utilizing uncertainty information provides a way to avoid directly modeling the sampling distribution; this makes our technique independent of the dimensionality of the data, using a fixed set of seven parameters. Additionally, using sampling as an exclusive means to pass information from the oracle to the interpretable model enables us to be model agnostic.

To state the above effect rigorously, we extend the terminology of (Ghose & Ravindran, 2020) and let:

1. $accuracy(M, p)$ be the classification accuracy of model M on data represented by the joint distribution $p(X, Y)$ of instances X and labels Y. The term "accuracy" is used in a generic sense to represent prediction accuracy; depending on the application, this might be *F1-score, AUC, lift*, etc.

2. $train_{\mathcal{F},f}(p, \eta)$ produce a model obtained using a specific training algorithm f, e.g., CART (Breiman et al., 1984), for a given model family $\mathcal{F}$, e.g., DTs, where the model size is fixed at η, e.g., trees with $depth = 5$. p represents the joint distribution $p(X, Y)$ of instances X and labels Y. $train_{\mathcal{F},f}(p, *)$ denotes there are no constraints imposed on the model size.

Then, we claim, and empirically demonstrate, that *the interpretable model trained on the generated sample is at least as accurate as one learned on the original training data, and is up to as accurate as the oracle*:

$$accuracy(M_{\mathcal{I}p\eta}, p) \leq accuracy(M_{\mathcal{I}q\eta}, p) \leq accuracy(M_{\mathcal{O}p*}, p) \tag{2}$$

where,

$$M_{\mathcal{I}p\eta} = train_{\mathcal{I},g}(p, \eta)$$
$$M_{\mathcal{I}q\eta} = train_{\mathcal{I},g}(q, \eta)$$
$$M_{\mathcal{O}p*} = train_{\mathcal{O},h}(p, *)$$

Here,

- For a model named M_{ABC}, this is what the subscripts denote:

  1. A signifies if the model is an oracle or an interpretable model, with symbols $\mathcal{O}$ and $\mathcal{I}$ respectively.
  2. B denotes the training distirbution.
  3. C is the model size.

- g and h represent specific training algorithms, e.g., *CART* for DTs, *rmsprop* (Graves, 2013) for neural networks. These are omitted in model names for brevity, and are made clear by context.

- We refer to $M_{\mathcal{I}p\eta}$ as the "baseline model", since this is the standard way of training a model against which we evaluate our approach.

- p and q both denote joint distributions of X and Y. $p(X, Y)$ is the distribution we are provided, and all our models use this as the *test* distribution. $q(X, Y)$ is the distribution we learn over the uncertainty scores provided by the oracle $M_{\mathcal{O}p*}$.

  Note that, typically, the train and test distributions are identical for a model, as in the terms $accuracy(M_{\mathcal{I}p\eta}, p)$ and $accuracy(M_{\mathcal{O}p*}, p)$. However, for the middle term in Equation 2 - $accuracy(M_{\mathcal{I}q\eta}, p)$ - the train and test distributions, p and q respectively, are different.

We also show that Equation 2 can be further refined into two size-regimes: the interpretable model trained on the new sample is more accurate than the baseline model only until a model size η^*. At sizes greater than η^* the model performances are equal:

$$accuracy(M_{\mathcal{I}p\eta}, p) < accuracy(M_{\mathcal{I}q\eta}, p) \leq accuracy(M_{\mathcal{O}p*}, p), \text{ when } \eta \leq \eta^* \quad (3)$$

$$accuracy(M_{\mathcal{I}p\eta}, p) = accuracy(M_{\mathcal{I}q\eta}, p) \leq accuracy(M_{\mathcal{O}p*}, p), \text{ when } \eta > \eta^* \quad (4)$$

We summarize our key contributions as follows:

1. We provide a practical technique to increase the accuracy of an interpretable model using an oracle model. This technique possesses the following favorable properties:

   (a) It is model agnostic, w.r.t. both the oracle and the interpretable model families.

   (b) It has a fixed set of seven parameters irrespective of the dimensionality of the data.

2. We provide a novel way to utilize uncertainty information from a classifier. It is also shown that exclusively sampling highly uncertain points is not optimal - which is not intuitively obvious.

As a corollary, we reaffirm the "small model effect" reported in Ghose and Ravindran (2020): a more accurate classifier may be obtained by using a training distribution different from the test distribution ($q(X, Y)$ and $p(X, Y)$ respectively in Equation 2), especially when the model size is small. This challenges the conventional wisdom that train and test distributions must be identical.

We identify no specific criteria for interpretability that model family $\mathcal{I}$ must satisfy. Our technique maybe used with any family; however, it is *useful* for applications requiring interpretability since small models are preferred. Thus, our emphasis on interpretability is use-case driven. Model compression is possibly another application, that we briefly mention in Section 6.

The remainder of the paper is structured as follows: we present an overview of our technique in Section 2. Sections 3 and 4 discuss the working of our technique in detail and Section 5 presents extensive experimental validation using real-world datasets. Sections 6 and 7 mention directions for future work and conclude the paper.

2. Overview

We provide an overview of our technique by using an illustrative example, followed by a discussion of the modelling workflow and prior work. We also define the terminology that is used in the subsequent sections.

2.1 A Toy Example

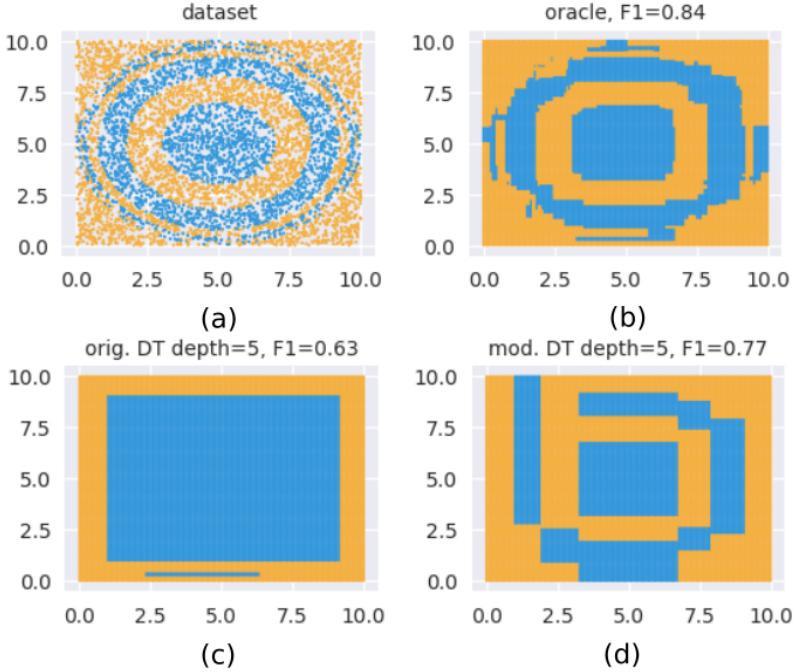Figure 1 demonstrates our technique on a toy dataset.

Figure 1: A demo of our technique using a Gradient Boosted Model as an oracle.

Fig 1(a) shows a dataset with 2 labels we want to classify. Fig 1(b) visualizes the generalization learned by a *Gradient Boosted Model (GBM)* using this dataset. This seems fairly accurate with a F1 score = 0.84. Fig 1(c) shows what a *CART* (Breiman et al., 1984) decision tree of *depth* = 5 learns from the data; this has F1 score = 0.63. Finally, Fig 1(d) shows what a CART decision tree of *depth* = 5 learns, when we supply the GBM as an oracle to our technique. The F1 score improves significantly to 0.77. Visually, we see the boundaries approximating the ones learned by the oracle, as shown in Figure 1(b).

2.2 Workflow

Figure 2 compares (a) a standard workflow to our (b) model building workflow. In the standard setup, a model training algorithm, A, accepts training data and produces a model that maximizes some pre-defined prediction accuracy metric. Our workflow adds two new components - the adaptive sampling technique, B, and an oracle, C. The oracle provides information to the sampling technique, that enables it to identify a potentially "better"
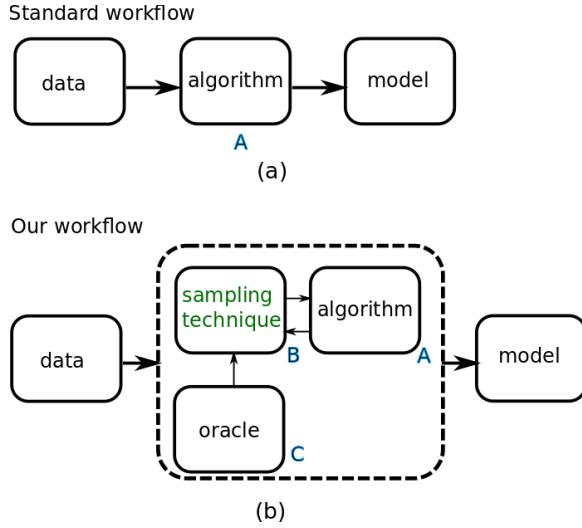
Figure 2: Modified workflow.

sample from the training data for input to algorithm A. Here, a "better" sample is the one that leads A to produce a model with the higher accuracy (measured on a held-out dataset), compared to training on the provided data as-is. Determining this sample is an iterative process; at each iteration, B modifies the sample based on the current accuracy of the model from A. The information from the oracle is conveyed to the sampling technique only once, before the beginning of the iterative interaction between A and B.

2.3 Related Work

While there is precedent to using different train and test distributions, such as when there is class imbalance in the data (Japkowicz & Stephen, 2002), the only other work we are aware of that discusses learning accurate interpretable classifiers by identifying a specific training distribution is our earlier research on the topic (Ghose & Ravindran, 2020). There, our strategy was to first construct a decision tree on the data, to obtain the fragmentation of the input space defined by the leaves. A distribution is then imposed on these fragments, according to which they are sampled from, to assemble a larger sample that is used to train the interpretable model. The parameters of this distribution are learned. Our current method may be considered a significant enhancement of this approach since it adds the following flexibility in modelling:

1. We can use an arbitrary oracle to guide the training of the interpretable model. This also has the practical benefit that the oracle need not be learned from scratch: if there is already a pre-trained probabilistic model like a *deep neural network* available for a dataset, it may be conveniently plugged into our algorithm to quickly enhance the accuracy of an interpretable model.

2. The oracle and the interpretable model might represent the data with different features, e.g., the oracle might be a sequence model that classifies text, while the in-

terpretable model may be a n-gram based classifier. This considerably broadens the scope of our technique. We look at an example in Section 5.3.

Our technique also has some similarities to ideas in the areas of *active learning* and *transfer learning*:

1. **Active Learning**: In the case of active learning too, a predictive model maybe learned on a distribution $q(X, Y)$ that is different from the test distribution $p(X, Y)$. However, some significant differences are:

    (a) Active learning works in the setting where only some or none of the labels of the training data are initially known, and there is an explicit label acquisition cost. We work within the traditional supervised setting where labels of all training instances are known.

    (b) The goal of an active learner is to minimize the total label acquisition cost, while being as accurate as a supervised learner that has access to complete label information. This is very different from our goal of performing *better* than a supervised learner, especially when the model size is small, assuming complete label information.

    (c) Although *uncertainty sampling* (Lewis & Gale, 1994), a heuristic sampling strategy based on uncertainties[1] is a popular technique in active learning, we show that it often does not produce an optimal sample for our problem[2] (Section 5.2.2).

    It must be noted that the term "oracle" in the active learning literature might refer to either a model or a human labeler; in our work, it exclusively refers to a model.

2. **Transfer Learning**: Transfer learning studies informing the training process of a "target" learner, given a "source" learner (Torrey & Shavlik, 2009; Pan & Yang, 2010; Weiss, Khoshgoftaar, & Wang, 2016). Our technique is ostensibly similar as we have an oracle (our source learner) informing the interpretable model (our target learner). However, here are some key differences:

    (a) The typical application of transfer learning is in settings where the source learner has access to more data than the model it must transfer knowledge to; here transfer learning is seen as a way to overcome the data shortage by directly having the source learner convey knowledge, in some form, to the target model. This is different from our setting where the same data is available to both the oracle and the interpretable model.

    (b) Transfer learning techniques usually make some assumptions about the model family. Some examples are Boolean concepts in (Thrun & Mitchell, 1994), Markov Logic Networks in (Mihalkova & Mooney, 2006) or task-specific neural networks like *BERT* (Devlin, Chang, Lee, & Toutanova, 2019) or *ULMFiT*

---

1. Several other ways of using uncertainty information have since been explored (Tong & Koller, 2002; Anderson & Moore, 2005; Gal, Islam, & Ghahramani, 2017; Beluch, Genewein, Nurnberger, & Kohler, 2018; Hafner, Tran, Lillicrap, Irpan, & Davidson, 2019); these are beyond the scope of our discussion.
2. It suffers from certain biases even when used for active learning, limiting it utility (Dasgupta & Hsu, 2008; Dasgupta, 2011).

(Howard & Ruder, 2018) for Natural Language Processing, and *VGG networks* (Simonyan & Zisserman, 2015) for image recognition. In comparison, our technique is model agnostic, both w.r.t. the oracle and the interpretable model.

(c) Although instance re-weighting techniques have been investigated as a means of transfer learning[3], their objective is to perform effective learning in situations where the data distribution available in the source task/domain is different from that in the target task/domain (Liao, Xue, & Carin, 2005; Dai, Yang, Xue, & Yu, 2007; Kamishima, Hamasaki, & Akaho, 2009). In our case, these two distributions, as provided, are identical; we *choose* to use a different training distribution in the interest of improving accuracy.

Thus, while the areas of active learning and transfer learning have some overlap with our problem, there is no simple or direct correspondence to it.

2.4 Terminology and Notation

We first define the notion of *model size* since it is critical for subsequent discussions. Model size is a model parameter with the following properties:

1. $model\_size \propto bias^{-1}$

2. The interpretability of a model decreases with increasing model size.

As mentioned before, only the first criteria above is required for using our technique. The second criteria reflects our emphasis on interpretability.

It must be noted that the notion of model size is subjective. Consider a *Gradient Boosted Model (GBM)* with DTs as base classifiers: here, the depth of the individual trees, or the number of trees, or both collectively may be seen as representing size. Even for a given notion of size, the value up to which a model is considered interpretable may be a matter of opinion. For example, some might consider a DT with $depth = 15$ to be interpretable, while some might decide $depth = 10$ to be the limit for interpretability. However, as long as the notion of size satisfies the criteria above, the discussion in this paper applies.

We now introduce the notations we follow:

1. We denote a dataset, D, by a set of instance-label pairs, i.e., $D = \{(x_1, y_1), (x_2, y_2), ..., (x_N, y_N)\}$, where x_i is the feature vector representing an instance and y_i is its corresponding label.

   Sometimes, we use *multisets*, when instance-label pairs may be repeated. Such usage is explicitly called out.

2. While we have referred earlier to the joint distribution of instances and labels, e.g., $p(X, Y)$ in Equation 2, this is understood to represent the dataset that we are actually given, in the form of a finite number of instance-label pairs.

3. We use the term *original*, as in *original distribution* or *original data* to denote the data that we are given. This is in contrast with samples we generate.

---

3. We specifically mention this since instance re-weighting maybe seen as a form of sampling.

4. The terms $accuracy()$ and $train_{\mathcal{F},f}()$ are overloaded to accept a dataset as input in lieu of a distribution:

   - $accuracy(M, D)$ denotes the accuracy of model M with the dataset D as the test set.

   - $train_{\mathcal{F},f}(D, \eta)$ denotes a specific training algorithm f, for a model family $\mathcal{F}$, that accepts as input a dataset D, and trains a model of size η.

5. The terms *pdf* and *pmf* denote *probability density function* and *probability mass function* respectively. The term probability distribution may refer to either, and is made clear by the context.

□

Next, we begin to look at our methodology.

3. Methodology

We describe our methodology, at a high level, in this section. Since some of our ideas here are based on our previous work (Ghose & Ravindran, 2020), we review it first, referring to the technique as a *density tree* based approach.

3.1 Review: Sampling using Density Trees

In our earlier work, we begin by asking why should we expect to be able to improve the accuracy of a small model at all? We hypothesize that models are not always optimal for a *given size*. All training algorithms make heuristic choices to make learning tractable, e.g., local search based techniques such as *rmsprop* (Graves, 2013) and *Adam* (Kingma & Ba, 2015) for learning neural networks, one-step look-ahead in the CART algorithm for learning DTs. This creates a *gap* between the *representational* and *effective* capacities of models; ideally a binary DT learning algorithm constrained to learn trees of up to $depth = 5$ should be able to produce full binary trees with $2^5 = 32$ leaves if needed (for example, in Figure 1(c)), but this is rarely seen in practice. Allowing the model to grow to an arbitrary size works around this issue; we may eventually end up with a DT of $depth = 10$ that solves our problem (albeit with number of leaves much lesser than 2^{10}). At a given model size, decreasing this gap gives us an opportunity to improve accuracy. One mechanism to decrease the gap is to focus the training algorithm on regions of the input space that most impact learning.

The problem of finding such regions is formulated as determining an optimal training distribution. Since representing a distribution in d dimensions requires at least $O(d)$ variables, e.g, when using a *Gaussian Mixture Model* with a diagonal covariance matrix, this leads to a computationally expensive optimization problem for most real-world data. We offer an indirect representation as a solution. Our key observation is the fragmentation of the input space produced by a DT possesses the property that smaller fragments are typically located close to class boundaries. This may be seen in Figure 3 - the leaves of a DT learned (with no size constraints) on the dataset from Figure 1 are visualized as rectangles.
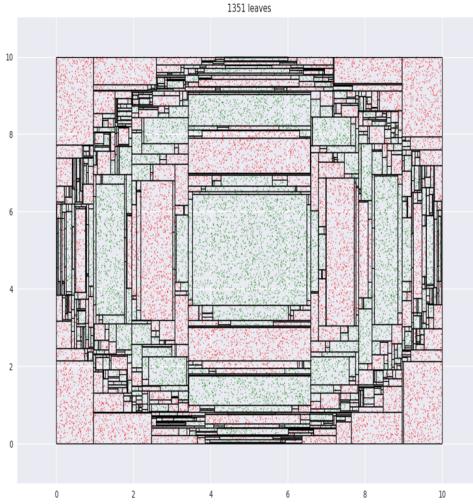
1351 leaves

Figure 3: Fragmentation of input space by decision tree. Source: Ghose and Ravindran (2020)

Given a dataset, a DT is first constructed to use as a guide for sampling[4]. This DT is known as a *density tree*[5]. The sampling technique consists of these main components:

1. Use a *pmf* to pick leaf regions to sample from; within a leaf, instances are uniformly sampled. A prior favoring leaves with small volume is used. This enables the optimizer to operate at the abstraction of sampling from regions based on their proximity to class boundaries.

2. The sampling doesn't need to be limited to the leaf level. We may see the different depths of a DT as corresponding to differing amounts of information about class boundaries: with no information at the root, and complete information at the leaves. Depending on how much information is required for an optimal solution, the sampler may pick any combination of depths to sample from, using a different *pmf* to sample from nodes at a depth. This combination is represented by a mixture model *pdf* called the "depth distribution".

3. Not all points are sampled using the density tree. With a probability p_o, the sampler may also uniformly sample the original training data. This option is provided so that optimization algorithm may (partially) fall back on the original training data, if that may contribute to an optimal sample.

Because of the use of shared parameters, the above sampling technique requires only eight parameters, including p_o. These parameters constitute the distribution representation,

---

4. It is recommended to construct multiple trees for low variance, but we ignore this detail for simplicity, and assume exactly one tree is constructed.
5. To avoid confusion, we abbreviate decision trees as DTs, but the term density tree is used as-is.

and they are optimized to maximize accuracy (on a validation set) of a model trained on a sample drawn from this distribution.
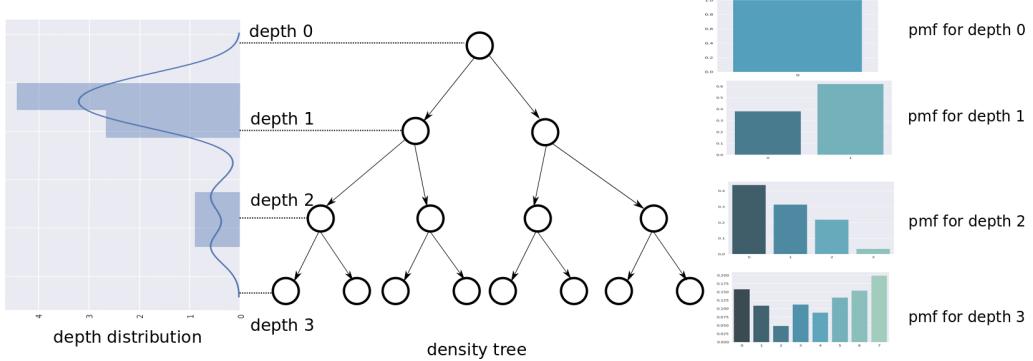

Figure 4: The *depth distribution* and the depth specific *pmf*s for the density tree approach are shown.

Figure 4 shows a typical density tree. Every node corresponds to some region of the input space. A depth distribution is shown to the left of the tree, and for different depths, the corresponding *pmf*s are shown to the right of the tree. To sample N_s points, $p_o \times N_s$ points are randomly sampled from the original distribution. Each of the remaining $(1 - p_o) \times N_s$ points is sampled in the following manner: (1) use the depth distribution to first select a depth (2) then use the *pmf* at that depth to sample a node (3) and finally, uniformly sample a point from the region defined by that node.

It is instructive to look at the optimal depth distributions that are obtained. This is concisely visualized in Figure 5 for interpretable models of different sizes (here, DTs of different depths), across different datasets; the x-axis shows density tree depths normalized to lie within $[0, 1]$ (so depth distributions over density trees of various heights may be compared[6]), where 0 and 1 indicate the root level and leaf level respectively. The curve for a specific dataset is plotted in the following manner:

1. Interpretable models for a range of sizes are built for the dataset. Let's say the sizes are $k \in \{1, 2, ..., K\}$. For a size k, we denote the relative improvement - using a density tree compared to not using it - by Δ_k.

2. n_k points are sampled from the optimal depth distribution obtained for the model of size k, where $n_k \propto \Delta_k$. Points sampled thus, across the K model sizes, are pooled together.

3. A *Kernel Density Estimator (KDE)* is fit to this sample and plotted.

Weighting w.r.t. Δ_k aggregates the depth distributions for different sizes into a single KDE plot, allowing each distribution to shape it in proportion to the improvement it has produced.

6. One dataset corresponds to one density tree; but here we're comparing across datasets, each of which has its own density tree.

Figure 5: Weighted depth distribution for multiple datasets. Source: (Ghose & Ravindran, 2020)

Interestingly, a common pattern across the KDE plots for different datasets may be observed: most of the sampling either occurs near the root or near the leaves. This pattern is fairly consistent across various experiments we have performed. Our hypothesis is that since the intermediate tree levels are noisy w.r.t. to information content - the class boundaries have not been fully discovered yet, but we have moved away from the original distribution - the interpretable model avoids sampling here.

The sampling technique turns out to be highly effective, as is indicated by the results presented in the paper. We leave out various other technical details, e.g., handling of axis-parallel boundaries, incomplete density trees, since they are not relevant here.

3.2 Sampling using a Oracle

In order to extend the above technique to use oracles, we consider how the information provided by the density tree may be extracted from an arbitrary classifier. Here, our key insights are (1) uncertainty information from probabilistic models may be used as an alternative to the class boundary information provided by a density tree, and (2) the root node information is redundant and the intermediate depths may be ignored for an approximate solution. We elaborate these further.

1. **Using uncertainty information**: The uncertainty score (precisely defined later) for a prediction made by a model quantifies its ambiguity in making the prediction. Intuitively, for a good classifier, it is expected that the instances with high uncertainty scores lie close to class boundaries. This is similar to the role played by leaves with low volumes in the density tree based approach.

   Figure 6 illustrates this correspondence. Figure 6(a) reproduces Figure 3, showing the fragmentation generated by a DT. Figure 6(b) visualizes uncertainty scores (darker is

Figure 6: (a) This is the same as Figure 3 (b) Uncertainty scores from a SVM are shown, where darker shades imply greater uncertainty. Note how low volume regions in (a) correspond to darker regions in (b).

higher) across the input space, for a *Support Vector Machine (SVM)* using a *Radial Basis Function* kernel, trained on the same dataset. We see that leaves with low volumes in Figure 6(a) roughly correspond to regions with high uncertainties in Figure 6(b).

Using uncertainty scores provides an alternative way to obtain class boundary information. Since most classifiers generate some form of prediction score, based on which an uncertainty score may be calculated, this provides us with a way to utilize a wide variety of oracles for the sampling process.

Naive uncertainty based sampling, when used for active learning, is known to suffer from sample bias. We want to visit this issue here to see if it is pertinent to our technique.

In active learning, our goal is to learn a model when we are given none or few of the labels of our training data, but we are allowed to query for labels for a cost (Settles, 2009). This is helpful in scenarios where acquiring labels is expensive, and instead of asking for labels for a random 1000 points to train on, we could ask for the labels of a specific 200 points, chosen in some manner, that leads to comparable model accuracy. *Uncertainty Sampling* was introduced in (Lewis & Gale, 1994) to solve this problem. We begin by requesting the labels of small batch of randomly sampled points - this is the labelled subset of the data. The following steps are then repeated:

(a) Construct a classifier on the current labelled subset.

(b) Use it to provide uncertainty scores for unlabelled points in the data, and then request labels for the top b (the precise value of b may be task specific) uncertain points. These now become part of the labelled subset.

13

Although intuitive, this approach was shown to suffer from sample bias (Dasgupta & Hsu, 2008; Dasgupta, 2011). We illustrate this in Figure 7.

Figure 7: Uncertainty estimates from uncertainty sampling compared to those from an oracle. The former presents an incomplete picture because of the active learning setting.

We consider the simple case where our data is located on a line, has two labels (denoted by red and green in the figure) and most of the data is located at the extremes of the line segment, as shown by blocks P and Q, each of which represent 45% of the overall data. Here, learning a classifier is equivalent to identifying a single point on the line, and the classification rule is we assign labels green and red, to left and right of this point, respectively. B and C show two possible classifiers.

In the active learning setup, we observe only the points but not their labels. To use uncertainty sampling, we pick our first small batch of points randomly and query their labels. Because of the distribution of the data, its highly likely that we would only see points from P and Q. The best classifier on this sample is C, which is midway between P and Q. Plot A shows what the uncertainty across the input space looks like according to C. In the next iteration, we will sample close to C, since that's where the highest uncertainties are, and the new classifier constructed would again be at location C. Subsequent iterations would further reinforce the "correctness" of C. Here, the classification error of C is 5%, but the optimal classifier is B, with an error of 2.5%, which uncertainty sampling fails to discover. The key problem here is we may never see some boundaries, like those defined by R, because of the combination of initial sample bias and subsequent aggressive sampling.

The plot at the bottom, D, shows the uncertainty landscape according to our oracle. Since ours is a supervised setting, the oracle has access to the complete joint distribution of instances and labels during training. The uncertainty plot correctly captures three peaks at points on the line where the label changes.

Thus, the sample bias of the classic uncertainty sampling technique doesn't apply to the supervised setting.

However, we make the interesting observation in our experiments in Section 5.2.2 that the absence of this bias is not enough: even with uncertainties accurately represented, sampling exclusively from highly uncertain regions is not an optimal strategy.

2. **Alternatives to the root node and intermediate depths of the density tree:** Aside from the leaf level, the density tree also provides information at its root node and intermediate depths, i.e., depths other than the root and leaf levels. Lets consider how an arbitrary oracle may account for such information:

   (a) **Root node**. According to Figure 5, there is high sampling density at the root node (x-axis= 0) which makes it a significant source of information. We require an equivalent source when using an oracle.

   Here we note that the root node represents the training data as-is, i.e., there is no fragmentation at this point, and uniformly sampling here is equivalent to uniformly sampling from the original distribution with an appropriate p_o. In this sense, sampling at the root node is redundant, if one is also allowed to sample from the original distribution. This suggests a simple strategy for an oracle based approach: retain the option to sample from the original training data[7].

   (b) **Intermediate depths.** To simulate the intermediate depths, we consider the following possibilities:

      i. In principle, we could create oracle classifiers of varying strengths and use the equivalent of a "depth distribution" over these classifiers. Sampling would work similar to the density tree approach: first pick a classifier of a specific strength using this distribution, and then sample based on the uncertainty scores of this classifier.

      An example of oracle classifiers of varying strengths is a neural network trained till a varying number of epochs, or a GBM trained up to a varying number of boosting rounds.

      ii. Referring to Figure 5, we note that the contribution of the intermediate depths to the optimal sample is relatively low. We may ignore these depths altogether.

   We adopt the second approach because its simple, and importantly, our experiments validate that ignoring the intermediate depths still lead to good results

---

7. In the density tree approach, even though the root node and the original data provide equivalent sample spaces, the parameter p_o is introduced to study the composition of the optimal sample, i.e., how much of it comes from the original sample. We believe there might be a computational advantage too; this makes it convenient for the optimizer to use the original data: its faster to tune just the parameter p_o, than to tune the multiple parameters for the depth distribution to obtain a component favouring the root.

(bench-marked against the density tree based approach). The first approach is challenging in multiple ways: it is hard to concretely define what varying strengths may mean for different model families, it is not clear how many such classifiers we need to construct, and also, it doesn't support the use-case of being able to use a previously trained oracle, i.e., an oracle of only one strength.

Table 1 summarizes how the broadly defined information sources compare across the density tree and oracle based approaches.

Table 1: Information sources compared across density tree and oracle based approaches.

	Density Tree	**Oracle**
1	Original distribution, controlled by p_o	Original distribution, controlled by p_o
2	Root node	Approximated by original distribution, controlled by p_o
3	Intermediate depths	Ignored, owing to low impact
4	Leaf nodes	Uncertainty information

□

We now have a high-level strategy to approximate the density tree based approach with an oracle. In the next section, we look at algorithmic realizations of these ideas.

4. Algorithm Details

This section discusses the implementation details of the ideas presented in Section 3.2. We begin by precisely defining uncertainty, and then look at the representation for the sampling distribution, the actual algorithm, and details of the optimization.

4.1 Measuring Uncertainty

Our technique critically depends on how we measure uncertainty. We denote the uncertainty of prediction by a model M on an instance x by $u_M(x)$, where $u_M(x) \in [0, 1]$. Some popular metrics used to measure uncertainty are:

1. **Least confident**: we calculate the extent of uncertainty w.r.t. the class we are most confident about:

$$u_M(x) = 1 - \max_{y_i \in \{1,2,...,C\}} M(y_i|x) \tag{5}$$

   Here, we have C classes, and $M(y_i|x)$ is the probability score produced by the model[8].

2. **Margin**: this score is decided by the difference of the top two probabilities - lower the difference the more uncertain the model is. This was introduced in (Scheffer, Decomain, & Wrobel, 2001). Algorithm 1 describes how this is calculated.

---

8. The possibly confusing name "least confident" for this idea originated within the context of uncertainty sampling, where we are interested in sampling the most uncertain point, $x^* = \arg\min_x [\max_{y_i \in \{1,2,...,C\}} M(y_i|x)]$, which may be considered to be the instance with the "least most confident label".

3. **Entropy**: this is the standard Shannon entropy measure calculated over class prediction confidences:

$$u_M(x) = \sum_{y_i \in \{1,2,...,C\}} -M(y_i|x) \log M(y_i|x) \tag{6}$$

Algorithm 1: Compute uncertainty of prediction, $u_M(x)$, for instance x by model M

Data: x, M

Result: $u_M(x)$

1 Calculate the probability $p_i = M(y_i|x)$ for predicting each class
   y_i, where $y_i \in \{1, 2, ..., C\}$

2 Let $p_{j_1} \leq p_{j_2} \leq ... \leq p_{j_{C-1}} \leq p_{j_C}$

3 $u_M(x) \leftarrow 1 - (p_{j_C} - p_{j_{C-1}})$

4 **return** $u_M(x)$

We use the *margin* uncertainty metric in our work. We do not use the *least confident* metric since it completely ignores confidence distribution across labels. While *entropy* is quite popular, and does take into account the confidence distribution, we do not use it since it reaches its maximum for only points for which the classifier must be equally ambiguous about *all* labels; for datasets with many labels (one of our experiments uses a dataset with 26 labels - see Table 2) we may never reach this maximum. Also we want to highly penalize points with even *two* likely labels, something that the margin uncertainty allows.

There is no best uncertainty metric in general, and the choice is usually application specific (Settles, 2009).

Fig 8 visually shows what uncertainty values look like for the different metrics. Panel (a) displays a dataset with 4 labels. A probabilistic *linear Support Vector Machine (SVM)* is learned on this, and uncertainty scores corresponding to the metrics "margin", "least confident" and "entropy" are visualized in panels (b), (c) and (d) respectively. Darker shades of gray correspond to high uncertainty. Observe that only the "margin" metric in panel (b) achieves scores close to 1 at the two-label boundaries.

We calibrate (Platt, 1999) our oracles for reliable probability estimates.

4.2 Sampling based on Uncertainty

We want to sample instances based on their uncertainty scores, but we don't want to rely on a heuristics based relationship between the sampling probabilities and the scores. We want our algorithm to learn this relationship. Essentially, we need a flexible distribution for $p(u_M(x))$, whose parameters we want to optimize. A desiderata for such a distribution is:

1. Since we want to avoid any assumptions, we want the distribution to be able to assume an arbitrary "shape", unlike, say using a normal distribution that is unimodal, and the mode is centered.

2. It should be defined over the bounded interval $[0, 1]$ since $u_M(x) \in [0, 1]$.

Figure 8: Visualizations of different uncertainty metrics. (a) shows a 4-label dataset on which linear SVM is learned. (b), (c), (d) visualize uncertainty scores based on different metrics, as per the linear SVM, where darker shades imply higher scores.

3. A fixed set of parameters is preferred over a conditional parameter space. An example of a distribution with a conditional parameter space is the popular *Gaussian Mixture Model*, where the number of parameters is determined by the number of components.

   We list this requirement since the parameters of this distribution are to be learned via optimization, and there are many more optimizers that can handle fixed than conditional parameter spaces. This affords us the flexibility of exploring a much wider variety of optimizers.

   This design consideration was instrumental in our current implementation, as it allowed us to conveniently experiment with different optimizers (further discussed in Section 4.4).

The *Infinite Beta Mixture Model (IBMM)* used in Ghose and Ravindran (2020) satisfies the above requirements. This is used for the depth distribution, which has very similar requirements: while reasons 1 and 3 above are identical in the context of density trees, the upper bound of the interval there is the density tree depth.

The IBMM is a mixture model with *Beta* components. It may be seen as a variation of the *Infinite Gaussian Mixture Model* (Rasmussen, 1999). A mixture model allows us to model an arbitrary distribution, satisfying our first requirement. Using *Beta* components enables support for a bounded interval - this satisfies our second requirement. The IBMM uses a *Dirichlet Process (DP)*, described by the *concentration parameter* $\alpha \in \mathbb{R}_{>0}$, to determine the number of components (alternatively referred to as *partitions* or *clusters*), and assignments of points to them. The parameters of the *Beta* components are sampled

from common priors, which themselves are *Beta* distributions. Use of a DP, with common parameter priors, gives us a fixed parameter space; this satisfies our third requirement.

This is how we sample N_s points, from a dataset D, using an oracle M_O:

1. Determine partitioning over the N_s points induced by the DP. We use *Blackwell-MacQueen* sampling (Blackwell & MacQueen, 1973) for this. Let's assume this step produces k partitions $\{c_1, c_2, ..., c_k\}$ and quantities $n_i \in \mathbb{N}$ where $\sum_{i=1}^{k} n_i = N$. Here, n_i denotes the number of points that belong to partition c_i.

2. We determine the $Beta(A_i, B_i)$ component for each c_i. We assume the priors for the *Beta* parameters are also represented by *Beta* distributions, i.e., $A_i \sim scale \times Beta(a, b)$ and $B_i \sim scale \times Beta(a', b')$. Since samples from the standard *Beta* are within $[0, 1]$, we use a parameter *scale* as a multiplier to obtain a wide range of A_i, B_i.

   Thus we have exactly two prior *Beta* distributions associated with our IBMM. Here, a, b, a', b' are positive reals.

3. Repeat for each c_i: for each instance-label pair (x_j, y_j) in our training dataset, we calculate the oracle uncertainty score, $u_{M_O}(x_j)$. We then calculate $p_j = Beta(u_{M_O}(x_j)|A_i, B_i)$. We scale these probabilities to sum to 1. These quantities are used as sampling probabilities for various (x_j, y_j), and n_i points are sampled with replacement based on them.

The parameters for the IBMM are collectively denoted by $\Psi = \{\alpha, a, b, a', b'\}$. The best values for Ψ are learned via an optimization process detailed in Section 4.3.

The above procedure is summarized in Algorithm 2. Note that $temp$ and D' are multisets in the algorithm, since we sample with replacement. Accordingly, line 13 uses the **multiset sum**, $\uplus$: if (x_i, y_i) occurs m times in D' and n times within $temp$, then $D' \leftarrow D' \uplus temp$ has $m + n$ occurrences of (x_i, y_i).

4.3 Learning Interpretable Models using an Oracle

We tie together the various individual pieces in this section. We have already discussed the parameters Ψ for the IBMM. Our framework uses two additional parameters:

1. $p_o \in [0, 1]$, proportion of instance-label pairs from the original training data. As discussed in Section 3.2, we want to retain the option of uniformly sampling from the original training data.

2. $N_s \in \mathbb{N}$, sample size. Since the sample size can have a significant effect on model performance, we allow the optimizer to determine its best value. N_s is constrained to be at least as large as what is needed for statistically significant results.

The complete set of parameters is denoted by $\Phi = \{\Psi, N_s, p_o\}$, where the IBMM parameters are denoted by $\Psi = \{\alpha, a, b, a', b'\}$.

Our technique randomly initializes Φ, creates a sample based on Algorithm 2 and the original training data (based on p_o), learns an interpretable model of size η on this sample, and evaluates it on a validation set. Based on the validation score, an optimizer modifies

---

Algorithm 2: Sample based on uncertainties

Data: Sample size N_s, oracle M_O, dataset $D = \{(x_1, y_1), (x_2, y_2), ..., (x_N, y_N)\}$,
    IBMM parameters $\Psi = \{\alpha, a, b, a', b'\}$

Result: Sample D', where $|D'| = N_s$

1 $D' = \{\}$ `// assumed to be a multiset`
2 $\{(c_1, n_1), (c_2, n_2), ..., (c_k, n_k)\} \leftarrow$ partition N_s using the DP `// Here $\sum_{i=1}^{k} n_i = N_s$.`
3 **for** $i \leftarrow 1$ **to** k **do**
4 $\quad$ $A_i \sim scale \times Beta(a, b)$
5 $\quad$ $B_i \sim scale \times Beta(a', b')$
6 $\quad$ **for** $j \leftarrow 1$ **to** N **do**
7 $\quad\quad$ $p_j \leftarrow Beta(u_{M_O}(x_j); A_i, B_i)$
8 $\quad$ **end**
9 $\quad$ **for** $j \leftarrow 1$ **to** N **do**
10 $\quad\quad$ $p'_j \leftarrow c \cdot p_j$, where $c = 1/\sum_{j=1}^{N} p_j$ `// normalize the probabilities`
11 $\quad$ **end**
12 $\quad$ $temp \leftarrow$ sample with replacement n_i instance-label pairs based on p'_j `// assumed to be a multiset`
13 $\quad$ $D' \leftarrow D' \uplus temp$ `// $\uplus$ is the multiset sum`
14 **end**
15 **return** D

---

the parameters Φ, and repeats the process. Our stopping criteria is an iteration budget T. Algorithm 3 lists these steps.

Some details to note in Algorithm 3:

1. The optimizer is represented by the function call $suggest()$ which takes as input all past parameter values and validation scores. $suggest()$ denotes a generic optimizer; not all optimizers require this extent of historical information.

2. While the training algorithm for the oracle, $train_{\mathcal{O}, h}()$ is taken as input, a pre-constructed oracle M_O may also be used. This would eliminate line 2.

3. $accuracy()$ on the validation data, D_{val}, serves as both the objective and fitness function.

4. Evaluation on the test set, D_{test} is done only once, in line 16, with the model that produces the best validation score.

5. Since we sample with replacement, both temporary datasets D_o and D_u, procured from uniformly sampling the original training data and sampling based on uncertainties respectively, are multisets. Accordingly, line 9 uses the multiset sum operator $\uplus$ to combine them.

6. Since the validation score s_t (line 11) needs to be reliable, in our implementation we repeat lines 7-10 *thrice* and use the averaged validation score as s_t.

Algorithm 3: Learning interpretable model using oracle

Data: Dataset D, model size η, $train_{\mathcal{O},h}()$, $train_{\mathcal{I},g}()$, iterations T

Result: Optimal parameters Φ^*, test set accuracy s_{test} at Φ^*, and interpretable model M^* at Φ^*

1 Create stratified splits $D_{train}, D_{val}, D_{test}$ from D

2 $M_O \leftarrow train_{\mathcal{O},h}(D_{train}, *)$

3 for $t \leftarrow 1$ **to** T **do**

4 $\Phi_t \leftarrow suggest(s_1, ...s_{t-1}, \Phi_1, ..., \Phi_{t-1})$ `// randomly initialize at` $t=1$

      `// Note:` $\Phi_t = \{\Psi_t, N_{s\_t}, p_{o\_t}\}$ `where` $\Psi_t = \{\alpha_t, a_t, b_t, a'_t, b'_t\}$.

5 $N_o \leftarrow p_{o,t} \times N_{s,t}$

6 $N_u \leftarrow N_{s\_t} - N_o$

7 $D_o \leftarrow$ uniformly sample, with replacement, N_o points from D_{train}

8 $D_u \leftarrow$ sample N_u points from D_{train} using M_O and Ψ_t as inputs to Algorithm 2

9 $D_s \leftarrow D_o \uplus D_u$ `//` D_o`,` D_u `are assumed to be multisets`

10 $M_t \leftarrow train_{\mathcal{I},g}(D_s, \eta)$

11 $s_t \leftarrow accuracy(M_t, D_{val})$

12 end

13 $t^* \leftarrow \arg\max_t \{s_1, s_2, ..., s_{T-1}, s_T\}$

14 $\Phi^* \leftarrow \Phi_{t^*}$

15 $M^* \leftarrow M_{t^*}$

16 $s_{test} \leftarrow accuracy(M^*, D_{test})$

17 return Φ^*, s_{test}, M^*

---

7. Class imbalance is accounted for in our implementation when training model M_t in line 10. We either balance the data by sampling (this is the case with a *Linear Probability Model*), or an appropriate cost function is used to simulate balanced classes (this is the case with DTs).

   It is important to note here that D_{val} and D_{test} are not artificially balanced (in fact, they are not modified beyond line 1), and therefore s_t and s_{test} reflect the accuracy on the original distribution.

Algorithm 3 presents the core contribution of the paper. Quite significantly, the optimization loop has a fixed set of seven variables, *irrespective* of the dimensionality of the data; this makes our technique practical for use on real-world datasets.

Clearly, the choice of the optimizer $suggest()$ is crucial - we discuss this next.

4.4 Choice of Optimizer

We begin by listing below the challenges faced by our optimizer:

1. **Black-box objective function**: Our objective function is $accuracy()$, which depends on the interpretable model produced by $train_{\mathcal{I},g}()$ in Algorithm 3. Since we want our technique to be model agnostic, nothing is assumed about the form of $train_{\mathcal{I},g}()$. This effectively makes our objective a black-box function.

2. **Noisy objective function**: The interpretable model is trained on a *sample* drawn based on the current parameters Φ_t. This implies two models constructed for the same Φ_t may not be identical. There might be other sources of noise intrinsic to the learning algorithm too, e.g,, local search used for training.

3. **Expensive objective function**: Every evaluation of the objective function requires an interpretable model to be trained, which is expensive. We want our optimizer to be conservative in its calls to the objective function.

We use *Bayesian Optimization (BO)* to implement *suggest*(). BOs build their own model of the response surface as a function of the optimization variables, over multiple iterations. They optimize this *surrogate* objective. This strategy enables them to work with black-box objective functions, satisfying our first requirement. BOs explicitly quantify the uncertainty[9] of the response surface model, by using appropriate representations such as *Gaussian Processes (GP)* or KDEs; this helps them to account for reasonable amounts of noise, which satisfies our second requirement. The evolving response surface (over iterations) allows BOs to balance *exploitation and exploration* to make well-informed decisions about the best point on which to next evaluate the objective function - making it conservative in its calls to $accuracy()$, and therefore $train_{\mathcal{I},g}()$. This satisfies our third requirement. See reference Brochu, Cora, and de Freitas (2010) for details.

While there exist other promising candidates for optimization, e.g., evolutionary algorithms such as *Covariance Matrix Adaptation Evolution Strategy (CMA-ES)* (Hansen & Ostermeier, 2001; Hansen & Kern, 2004) or bandit-based algorithms such as *Parallel Optimistic Optimization* (Grill, Valko, Munos, & Munos, 2015), we choose BO because of their continued success for *hyperparameter optimization*, a domain with similar optimization challenges.

Among BO techniques, of which there are many today, e.g., (Hutter, Hoos, & Leyton-Brown, 2011; Bergstra, Bardenet, Bengio, & Kégl, 2011; Malkomes & Garnett, 2018; Dai, Yu, Low, & Jaillet, 2019), we use the *Tree Structured Parzen Estimator (TPE)* algorithm (Bergstra et al., 2011) since it scales linearly with the number of evaluations (the runtime complexity of a naive BO algorithm is *cubic* in the number of evaluations (Shahriari, Swersky, Wang, Adams, & de Freitas, 2016)) and has a popular and mature library: *Hyperopt* (Bergstra, Yamins, & Cox, 2013).

We note here that while *TPE* supports conditional parameter spaces, our design of a fixed parameter space enabled us to conveniently experiment with various other optimizers like *DNGO* (Snoek, Rippel, Swersky, Kiros, Satish, Sundaram, Patwary, Prabhat, & Adams, 2015) and *Spearmint* (Snoek, Larochelle, & Adams, 2012) and compare their practical benefits.

4.5 Smoothing the Optimization Landscape

A final but key consideration in our optimization is to make it easier to discover the global maximum: Φ^* in Algorithm 3. Since BOs model the response surface of the actual objec-

---

9. The connotation of this term here is different from what we have seen before: it denotes variance in the response surface model.

tive function using a finite number of evaluations (s_t in Algorithm 3), a certain degree of smoothness is assumed (Shahriari et al., 2016; Brochu et al., 2010).

Here, the optimization variables Φ_t influence the sampling in Algorithm 2, which directly affects the score s_t that the BO consumes. Empirically, we have observed that the distribution of uncertainty scores produced by an oracle do not always form a smooth distribution. Consequently, neighboring values of Φ may pick drastically different samples leading to large differences in s_t.

To address this, we "flatten" out the distribution[10] within $[0, 1]$. Our transformation is simple: given N scores, we divide the interval $[0, 1]$ into B equal bins, and allocate N/B uncertainty scores to each bin, ordered by the scores, e.g., the lowest N/B scores are assigned to the bin for the interval $[0, 1/B)$, the next N/B scores are assigned to the bin for $[1/B, 2/B)$, and so on. Within a bin, the uncertainty scores are linearly mapped within the boundaries of the bin. This produces a transformation that looks like the uniform distribution, while preserving the ordering of the original scores. We prefer the likeness to the uniform distribution since it makes all regions within the interval $[0, 1]$ equally easy to discover.

The original and modified uncertainty distributions for the datasets `Sensorless` and `covtype.binary` are shown in Figure 9(a) and 9(b) respectively. While `Sensorless` appears to have a non-smooth distribution, and flattening here might help, this seems redundant for `covtype.binary` (further discussed in Section 6). However, since this step is computationally inexpensive, we perform this for all our experiments, saving us the effort of manually assessing its need.

(a) Sensorless, GBM   (b) covtype.binary, GBM

Figure 9: Example of curve-flattening, for datasets (a) `Sensorless` and (b) `covtype.binary`. The uncertainty scores shown are obtained using the *GBM* oracle.

---

10. Distribution transformations have a long history in statistics, e.g., *power transforms* like the *Box-Cox* (Box & Cox, 1964) and *Yeo-Johnson* (Yeo & Johnson, 2000) transforms. Within ML, *Batch Normalization* (Ioffe & Szegedy, 2015) is a popular example of a statistical correction to a loss landscape.

Our transformation is invertible, which is useful in analyzing the observations from our experiments. Note however, it is not differentiable because of the discontinuities at the bin-boundaries; we also don't require this property.

The transformation affects line 7 in Algorithm 2. Instead of sampling based on the actual oracle uncertainty scores:

$$p_j \leftarrow Beta(u_{M_O}(x_j); A_i, B_i) \tag{7}$$

we sample based on the transformed uncertainty scores, $u'_{M_O}(x_j)$:

$$p_j \leftarrow Beta(u'_{M_O}(x_j); A_i, B_i) \tag{8}$$

The use of the transformation is optional, since Algorithm 3 does not critically depend upon it, but makes it robust (discussed in Section 6).

$\square$

This concludes our discussion of algorithmic details. Our experimental validation of the technique is discussed next.

5. Experiments

To experimentally validate our technique we consider different real-world datasets, on which we train *Linear Probability Models (LPM)* and DTs, using *Gradient Boosted Models (GBM)* and *Random Forests (RF)* as oracles.

All four combinations of models and oracles $\{LPM, DT\} \times \{GBM, RF\}$ are explored (Section 5.2.1). We also compare our technique with an uncertainty sampling based approach (Section 5.2.2) and the density tree based approach (Section 5.2.3). Additionally, we present an application of the technique where the input feature representations to the oracle and interpretable models, a *Gated Recurrent Unit (GRU)* and a DT respectively, are different (Section 5.3). All results are reported over three trials. Collectively, these results provide rigorous empirical validation of the utility of our technique.

We begin by describing the setup for our experiments.

5.1 Setup

In this section we describe the following aspects of our experiments: datasets and metrics used, details regarding our models and oracles, and finally, the search space identified for optimization.

5.1.1 Data

We use *13* real-world datasets to validate our technique. Table 2 lists relevant details. These are picked to vary in their dimensions, number of labels and label distribution, enabling a broad validation of our technique. Although we use the version of data available on the *LIBSVM* website (Chang & Lin, 2011), we mention their original source in Table 2. 10000 instances from each dataset are used. We use a $train : val : test$ split ratio of $60 : 20 : 20$ to create D_{train}, D_{val} and D_{test} in all our experiments (line 1, Algorithm 3).

In terms of the label distribution, we are interested in knowing whether a dataset is balanced across its labels. We quantify this with the "Label Entropy", which is computed for a dataset with N instances and C classes in the the following manner:

$$\text{Label Entropy} = \sum_{j \in \{1,2,...,C\}} -p_j \log_C p_j \tag{9}$$

$$\text{Here, } p_j = \frac{|\{x_i | y_i = j\}|}{N}$$

Label Entropy $\in [0, 1]$ where values close to 1 denote the dataset is nearly balanced, and values close to 0 represent relative imbalance.

5.1.2 METRICS

We broadly measure two quantities - improvements in model accuracy and comparative benefits to using our technique over a different one. These are the metrics we use:

1. To measure $accuracy()$ as in Equation 2 or Algorithm 3, our metric of choice is the $F1$ (macro) score, evaluated on D_{test}. We use this since it accounts for class imbalance, e.g., it doesn't allow good results for a majority class to eclipse poor results for a minority class.

   To measure the *improvements* obtained from our technique, we record the percentage *relative improvement* in the $F1$ score compared to the *baseline* of training the model on the original distribution:

   $$\delta F1 = \frac{100 \times (F1_{new} - F1_{baseline})}{F1_{baseline}} \tag{10}$$

   Since the original distribution is part of the optimization search space, i.e., when $p_o = 1$, the lowest improvement we report is 0%, i.e., $\delta F1 \in [0, \infty)$.

   All reported values of $\delta F1$ represent averaging over **three** runs of Algorithm 3, where we average the baseline and new scores *first*, and then calculate the improvement. In other words, if the runs are indexed by i, $F1_{new}$ and $F1_{baseline}$ are replaced by $\overline{F1}_{new} = \sum_{i=1}^{3} F1_{new,i}/3$ and $\overline{F1}_{baseline} = \sum_{i=1}^{3} F1_{baseline,i}/3$ respectively, in Equation 10.

   We take an average of the scores first since $F1_{baseline}$ can be a small value, especially at smaller model sizes, and being in the denominator, slight changes to it across runs can produce outsize differences in the per-run $\delta F1$ scores.

2. For performing comparative analysis, e.g., with the density based approach or uncertainty sampling, we introduce the *Scaled Difference in Improvement (SDI)* score.

   If the improvement produced by our method and the alternative method are denoted by $\delta F1_{ora}$ and $\delta F1_{alt}$ respectively, then:

   $$SDI = \begin{cases} \frac{\delta F1_{ora}}{H} - \frac{\delta F1_{alt}}{H}, & \text{if } H > 0 \\ 0, & \text{if } H = 0 \end{cases} \tag{11}$$

   $$\text{where } H = \max\{\delta F1_{alt}, \delta F1_{ora}\}$$

Table 2: Datasets: we use the dataset versions available on the LIBSVM website (Chang & Lin, 2011). However, we have mentioned the original source in the "Description" column.

S.No.	Dataset	Dimensions	# Classes	Label Entropy	Description
1	cod-rna	8	2	0.92	Predict presence of non-coding RNA common to a pair of RNA sequences, based on individual sequence properties and their similarity (Uzilov, Keegan, & Mathews, 2006).
2	ijcnn1	22	2	0.46	Time series data produced by an internal combustion engine is used to predict normal engine firings vs misfirings (Prokhorov, 2001). Transformations as in (Chang & Lin, 2001).
3	higgs	28	2	1.00	Predict if a particle collision produces Higgs bosons or not, based on collision properties (Baldi, Sadowski, & Whiteson, 2014).
4	covtype.binary	54	2	1.00	Modification of the *covtype* dataset (see row 12), where classes are divided into two groups (Collobert, Bengio, & Bengio, 2002).
5	phishing	68	2	0.99	Various website features are used to predict if the website is a *phishing* website (Mohammad, Thabtah, & McCluskey, 2012). Transformations used as in (Juan, Zhuang, Chin, & Lin, 2016)
6	a1a	123	2	0.80	Predict whether a person makes over 50K a year, based on census data variables (Dua & Graff, 2017). Transformations as in (Platt, 1998).
7	pendigits	16	10	1.00	Classify handwritten digit samples into the digits 0-9. (Alimoglu & Alpaydin, 1996; Dua & Graff, 2017).
8	letter	16	26	1.00	Images of the capital letters A-Z were produced by random distortion of these characters from 20 fonts. The task is to classify these character images as one of the original letters (Michie, Spiegelhalter, Taylor, & Campbell, 1995). Transformations as in (Hsu & Lin, 2002).
9	Sensorless	48	11	1.00	Based on phase current measurements of an electric motor, predict different error conditions (Paschke, Bayer, Bator, Mnks, Dicks, Enge-Rosenblatt, & Lohweg, 2013). We use the transformations from (Wang, Tan, Chen, Lin, Keerthi, Mahajan, Sundararajan, & Lin, 2018).
10	senseit_aco	50	3	0.95	Predict vehicle type using acoustic data gathered by a sensor network (Duarte & Hu, 2004).
11	senseit_sei	50	3	0.94	Predict vehicle type using seismic data gathered by a sensor network (Duarte & Hu, 2004).
12	covtype	54	7	0.62	Predict forest cover type from cartographic variables (Dean & Blackard, 1998; Dua & Graff, 2017).
13	connect-4	126	3	0.77	Predict if the first player wins, loses or draws, based on board positions of the board game *Connect Four* (Dua & Graff, 2017).

The key idea here is that the improvement possible across these competing techniques is $\in [0, H]$, and the SDI score measures the difference between the fraction of this range realized by either technique. Note that $H \geq 0$ since $\delta F1_{ora} \geq 0$ and $\delta F1_{alt} \geq 0$. This score has the following intuitive properties:

(a) $SDI \in [-1, 1]$

(b) $SDI > 0$ when $\delta F1_{ora} > \delta F1_{alt}$

(c) $SDI = 0$ when $\delta F1_{ora} = \delta F1_{alt}$

(d) $SDI < 0$ when $\delta F1_{ora} < \delta F1_{alt}$

For ease of interpretation, we average the SDI scores at the level of a dataset, across model sizes, for a given model and oracle. This averaged score is denoted by $\overline{SDI}$. Additionally, we also report the percentage of times $\delta F1_{ora} > \delta F1_{alt}$ across these model sizes. This is denoted as *pct_better*. While $\overline{SDI}$ quantifies the *extent* to which one technique might be better, *pct_better* denotes how many times does this occur. We believe both these metrics are of practical relevance in selecting one technique over another.

We consider the oracle-based approach to be a meaningful contribution **if $\overline{SDI} > 0$ and** *pct_better* **> 50%** compared to alternatives.

5.1.3 Models

For interpretable models $\mathcal{I}$, we consider the following model families:

1. *Linear Probability Model (LPM)* (Mood, 2010): This is a linear classifier. We use the commonly accepted notion of model size here: the number of terms in the model, i.e., features from the original data, with non-zero coefficients. We use the *Least Angle Regression* (Efron, Hastie, Johnstone, & Tibshirani, 2004) algorithm, that grows the model one term at a time, to enforce the size constraint. We use our own implementation based on the *scikit-learn* library (Pedregosa, Varoquaux, Gramfort, Michel, Thirion, Grisel, Blondel, Prettenhofer, Weiss, Dubourg, Vanderplas, Passos, Cournapeau, Brucher, Perrot, & Duchesnay, 2011).

   Since LPMs inherently handle only binary class data, for a multiclass problem, we construct a *one-vs-rest* model, comprising of as many binary classifiers as there are distinct labels. The given size is enforced for *each* binary classifier. For instance, consider the dataset *letter* in Table 2, with 26 classes. A model size of 10 implies we construct 26 binary classifiers, each with 10 terms. We have not used the more common *Logistic Regression* classifier because: (1) from the perspective of interpretability, LPMs provide a better sense of variable importance (Mood, 2010) (2) we believe our technique is well validated by any linear classifier.

   Sizes: For a dataset with dimensionality d, we construct models of sizes: $\{1, 2, ..., min(d, 15)\}$. We end up with sizes less than 15 only for the dataset *cod-rna*, which has $d = 8$. All other datasets have $d > 15$ (see Table 2).

2. *Decision Trees (DT)*: We use the implementation of CART in the *scikit-learn* library. Our notion of size here is the depth of the tree.

Sizes: For a dataset, we first learn an optimal tree T_{opt} based on the *F1-score* (macro), without any size constraints. Denote the depth of this tree by $depth(T_{opt})$. We then try our algorithm for these settings of CART's *max_depth* parameter: $\{1, 2, ..., min(depth(T_{opt}), 15)\}$, i.e., we experiment only up to a model size of 15, stopping early if we encounter the optimal tree size. Stopping early makes sense since the model is saturated in its learning from the data; changing the input distribution is not helpful beyond this point.

Note that while our notion of size is the *actual* depth of the tree produced, the parameter we vary is *max_depth*; this is because decision tree libraries do not allow specification of an exact tree depth. This is important to remember since CART produces trees with actual depth up to as large as the specified *max_depth*, and therefore, we might not see actual tree depths take all values in $\{1, 2, ..., min(depth(T_{opt}), 15)\}$, e.g., *max_depth* = 5 might give us a tree with *depth* = 5, *max_depth* = 6 might also result in a tree with *depth* = 5, but *max_depth* = 7 might give us a tree with *depth* = 7. We report improvements *at actual depths*.

The choice of the above model families also enable convenient comparison to the density tree based approach, as we had previously reported results using them (Ghose & Ravindran, 2020).

5.1.4 Oracles

We want our oracle models $\mathcal{O}$ to be fairly accurate (so that the derived uncertainty information is reliable), hence we pick the following model families:

1. *Gradient Boosted Models (GBM)*: We used a gradient boosting model with DTs as our base classifiers. The *LightGBM* library (Ke, Meng, Finley, Wang, Chen, Ma, Ye, & Liu, 2017) is used in our experiments. Effective parameters were determined using a validation set. **NOTE:** This is *not* D_{val} from Algorithm 3, since that would constitute *data leakage*. A stratified sample from within D_{train} was held out for learning good parameters.

2. *Random Forests (RF)*: We used the implementation available in *scikit-learn*. Parameters were learned using 5-fold cross-validation over D_{train}.

As mentioned earlier, the above oracles were calibrated (Platt, 1999) for reliable probability estimates.

5.1.5 Optimization Search Space

The optimizer we use, TPE, requires *box constraints*. Here we specify our search space for the optimization variables, Φ in Algorithm 3:

1. p_o: We want to allow the algorithm to pick an arbitrary fraction of samples from the original data; we set $p_o \in [0, 1]$.

2. N_s: We set $N_s \in [400, 10000]$. The lower bound ensures we have statistically significant results. The upper bound is set to a reasonably large value.

3. $\{a, b, a', b'\}$: Each of these parameters are allowed a range $[0.1, 10]$ to allow for a wide range of shapes for the component *Beta* distributions.

4. *scale*: We fix *scale* $= 10000$ for our experiments, to allow for A_i and B_i to model skewed distributions where shape parameter large values might be required. For small values, the algorithm adapts by learning the appropriate $\{a, b, a', b'\}$.

5. α: For a DP, $\alpha \in \mathbb{R}_{>0}$. We use a lower bound of 0.1.

   To determine the upper bound, we rely on the following empirical relationship (Ohlssen, Sharples, & Spiegelhalter, 2007) between the number of components k and α:

$$E[k|\alpha] \approx 5\alpha + 2 \tag{12}$$

   Using trial-and-error, we determined a fairly inclusive upper bound on the number of components to be 500, which provides us the α upper bound of 99.6. Thus, we use $\alpha \in [0.1, 99.6]$.

   We draw a sample from the IBMM using *Blackwell-MacQueen* sampling (Blackwell & MacQueen, 1973).

   We use a flattening transformation (discussed in Section 4.5) on the original uncertainty distributions, with a fixed number of 20 bins. *However, all visualizations of distributions in the following sections were prepared after performing an inverse transformation*; hence, in studying them, it might be convenient to assume that no transformation was applied.

5.2 Observations

5.2.1 Improvements in Accuracy

In our first set of experiments, we study how models built using an oracle compare to baseline models, based on the $\delta F1$ score (Equation 10).

Figure 10 shows the improvements for different combinations of interpretable and oracle models, $\{LPM, DT\} \times \{GBM, RF\}$. The model size is on the x-axis, and is normalized to be in $[0, 1]$, so that performance across datasets may be conveniently compared in the same plot.

For LPMs, the model sizes for a dataset, i.e., number of non-zero terms, are multiplied by $1/min(d, 15)$, where d is the dimensionality of the data. For DTs, the model sizes are multiplied by $1/min(depth(T_{opt}), 15)$. All $\delta F1$ values are averaged over three runs, in the manner described in Section 5.1.2.

Table 3 enumerates the observations corresponding to the plots in Figure 10. The column *model_ora* represents the model and oracle combination used. For example, *dt_gbm* implies DT was used as the model and GBM as an oracle.

We observe that the oracle based approach indeed works on a variety of datasets, across different combinations of interpretable and oracle models. In some cases, such as the dataset `Sensorless`, for the LPM and RF combination, improvements are as high as $\delta F1 = 248.12\%$. The general trend seems to be that $\delta F1$ decreases as model sizes increase, with eventually $\delta F1 \approx 0$. This decrease seems to be faster for DTs, which makes intuitive sense

Figure 10: For different combinations of models and oracles: $\{LPM, DT\} \times \{GBM, RF\}$, these plots show improvements, $\delta F1$, seen for different model sizes and data. Table 3 shows the corresponding improvement scores.

given that a unit increase in size for a DT adds more representational power (a layer of nodes) than for an LPM (another term), making it harder to beat the baseline performance of DTs.

This decrease empirically verifies the property expressed by Equations 3 and 4.

We note that $\delta F1$ does not strictly monotonically decrease for all datasets, possibly due to the optimization terminating at a local maxima, e.g., in Table 3 see the entry for letter, lpm_rf, $size = 2$ ($improvement = 67.06\%$) and $size = 3$ ($improvement = 71.08\%$). But it largely appears to follow the general trend of decrease even in these cases.

Another way to visualize the improvements is to create a correspondence of model sizes, without and with using our technique, for the same accuracy. See Figure 11 as an example. The point $(12, 2)$ for senseit_aco implies that the accuracy of a LPM with 2 non-zero terms produced by our technique equals, or is greater than, the accuracy of a baseline LPM with 12 non-zero terms. The model size on the y-axis is the median of three runs. We refer to such a plot as the *compaction profile* for a model-oracle combination. We still prefer the visualizations in Figure 10 since they clearly show the *extent* of the improvement obtained. See Section B, Appendix, for more compaction profiles.

Figure 11: The compaction profile of LPM models using GBM as an oracle. A point (x, y) denotes the minimum size y of a model obtained using our technique that is *at least* as accurate as the baseline model of size x.

It is also instructive to see the IBMMs we have learned. For the case of the LPM these are visualized in Figure 12 for both oracles. The visualization technique is the same as the one used for Figure 5 in Section 3.1: for a dataset, IBMMs for different sizes are aggregated into one KDE plot, weighted by the improvement produced.

It is interesting to see that the optimal strategy, in general, turns out to be to sample from *both* regions of low and high uncertainties. One must be careful in comparing these plots to those in Figure 5; although they look similar, they don't describe the same

Table 3: This table shows the improvement, $\delta F1$, over the averaged baseline and improved scores across three runs. This is shown for different combinations of models and oracles: $\{LPM, DT\} \times \{GBM, RF\}$. The best improvement for a model size and oracle is indicated in bold

dataset	model_ora	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
cod-rna	lpm_gbm	0.24	11.82	13.91	14.03	**16.16**	**14.29**	**9.07**	0.17	-	-	-	-	-	-	-
	lpm_rf	**2.40**	**13.20**	**14.75**	**15.82**	15.62	9.81	0.19	**0.26**	-	-	-	-	-	-	-
	dt_gbm	0.51	**9.50**	1.41	**2.89**	0.45	0.99	1.03	0.05	0.01	**0.00**	-	-	-	-	-
	dt_rf	**0.67**	7.95	**2.08**	2.39	0.19	0.23	0.16	**0.68**	**0.31**	0.00	-	-	-	-	-
ijcnn1	lpm_gbm	**0.93**	**4.67**	3.14	2.40	**5.67**	**4.05**	3.53	3.78	3.67	2.06	**3.36**	3.27	2.90	**4.02**	**3.93**
	lpm_rf	0.26	1.84	**3.64**	**3.59**	2.24	1.40	3.36	3.51	2.81	**3.05**	1.42	**3.37**	**3.12**	2.75	3.13
	dt_gbm	2.51	11.73	6.41	7.33	7.68	4.13	2.92	**5.70**	0.35	0.18	0.04	0.26	0.67	0.30	-
	dt_rf	**4.26**	**14.40**	**11.41**	**9.18**	**7.98**	**6.34**	**4.31**	2.22	**1.88**	**1.26**	**1.75**	**1.49**	**0.80**	**0.73**	1.21
higgs	lpm_gbm	**30.78**	18.95	11.99	**7.50**	3.46	3.19	**4.20**	**3.71**	**3.61**	**2.84**	2.15	1.99	2.22	**2.32**	0.75
	lpm_rf	27.88	**21.43**	**14.95**	6.09	**4.84**	**3.27**	2.19	2.36	1.83	1.23	**2.89**	**3.11**	**3.28**	1.27	**0.89**
	dt_gbm	0.77	0.07	0.47	0.70	0.01	**1.41**	-	-	-	-	-	-	-	-	-
	dt_rf	**3.99**	0.95	**2.07**	1.66	**1.66**	1.32	-	-	-	-	-	-	-	-	-
covtype.binary	lpm_gbm	81.18	**66.29**	**29.22**	**16.13**	**9.55**	**7.11**	**5.30**	**5.19**	**4.21**	**4.19**	**4.23**	**3.80**	**3.45**	**2.59**	**2.20**
	lpm_rf	**81.81**	62.16	15.23	9.19	7.65	4.24	1.85	2.59	2.94	2.38	2.25	2.24	2.30	2.04	1.80
	dt_gbm	**1.59**	**0.71**	**2.28**	1.03	0.59	1.11	0.17	0.18	0.06	-	-	-	-	-	-
	dt_rf	0.80	0.42	1.67	**2.73**	**1.83**	**1.40**	**1.37**	**1.48**	**0.98**	0.07	-	0.00	-	-	-
phishing	lpm_gbm	0.00	**2.03**	2.74	3.02	**3.31**	3.36	3.03	1.45	1.35	**1.43**	1.07	0.91	0.90	0.76	0.62
	lpm_rf	**0.00**	1.96	**3.21**	**3.30**	3.27	**3.66**	**3.06**	**1.71**	**1.42**	1.37	**1.15**	**1.02**	**0.93**	**1.29**	**1.12**
	dt_gbm	**0.00**	0.33	0.00	0.22	0.42	**0.17**	**0.59**	**0.35**	**0.19**	0.00	**0.00**	0.00	0.00	0.00	**0.00**
	dt_rf	0.00	**0.91**	**0.56**	**0.72**	**0.48**	0.07	0.14	0.32	0.00	0.00	0.00	**0.11**	**0.06**	**0.02**	0.00
a1a	lpm_gbm	**0.00**	2.86	6.88	8.81	9.40	7.80	8.59	9.17	**8.73**	7.79	6.95	6.47	4.29	5.27	4.13
	lpm_rf	0.00	**3.58**	**8.48**	**10.13**	**10.42**	**8.89**	**8.91**	**9.79**	8.65	**9.03**	**7.93**	**7.57**	**6.04**	**6.82**	**6.42**
	dt_gbm	**0.02**	**6.24**	1.90	4.16	3.29	2.18	0.13	0.20	0.27	-	-	-	-	-	-
	dt_rf	0.00	6.17	**3.37**	**5.04**	**4.23**	**5.89**	**5.72**	**6.85**	**7.08**	5.06	3.02	2.94	-	-	3.16
pendigits	lpm_gbm	**55.28**	24.79	**19.76**	**9.17**	**10.11**	**7.00**	5.44	1.88	2.49	2.05	**2.27**	**3.00**	**3.05**	**3.42**	1.91
	lpm_rf	52.08	**26.43**	18.69	5.64	7.38	5.92	**6.92**	1.40	**3.22**	**2.57**	2.01	1.67	1.91	1.96	**2.66**
	dt_gbm	12.94	**6.50**	3.66	12.59	5.86	4.04	1.77	0.31	**0.02**	**0.07**	**0.00**	-	**0.00**	**0.00**	-
	dt_rf	**16.27**	5.49	**5.28**	**13.72**	**6.63**	**4.76**	**2.57**	**0.48**	0.00	0.04	0.00	0.00	0.00	0.00	-
letter	lpm_gbm	51.26	49.14	62.29	**34.39**	33.64	19.75	**21.00**	14.19	16.08	14.09	15.54	13.50	15.88	14.70	11.68
	lpm_rf	**61.85**	**67.06**	**71.68**	24.31	**39.12**	**24.79**	18.53	**22.40**	**22.87**	**21.02**	**20.21**	**21.64**	**17.99**	**17.65**	**16.75**
	dt_gbm	**3.55**	**12.97**	24.10	33.77	28.24	13.33	**10.03**	**2.85**	**3.31**	**1.99**	**1.04**	**0.52**	**0.05**	0.00	0.00
	dt_rf	0.00	12.75	**36.38**	**38.30**	**40.57**	**19.41**	5.85	1.87	3.09	1.19	0.58	0.23	0.00	0.00	0.00
Sensorless	lpm_gbm	215.68	**245.56**	**194.63**	117.14	**94.42**	**79.37**	**75.26**	67.50	57.27	49.72	43.39	44.35	35.55	39.70	38.71
	lpm_rf	**248.12**	235.13	151.32	**122.50**	93.19	77.13	75.06	**68.10**	**69.37**	**64.62**	**69.18**	**61.36**	**69.95**	**80.24**	**87.07**
	dt_gbm	**0.02**	42.49	**81.85**	52.37	19.33	**10.00**	**3.29**	**2.02**	**1.18**	**0.69**	**0.41**	**0.06**	**0.00**	0.00	-
	dt_rf	0.00	**42.99**	61.92	**61.36**	**25.97**	9.43	2.81	1.08	0.42	0.00	0.18	0.04	0.00	-	0.00
senseit_aco	lpm_gbm	173.34	**170.28**	63.78	39.58	33.88	25.15	21.91	15.15	11.78	9.16	6.80	6.11	5.43	5.23	**5.65**
	lpm_rf	**176.19**	169.29	**72.81**	**45.43**	**34.94**	**29.12**	**22.38**	**19.79**	**13.41**	**10.40**	**9.59**	**6.41**	**6.21**	**5.63**	4.15
	dt_gbm	**12.25**	**2.18**	**2.92**	3.18	3.22	1.63	0.57	-	-	-	-	-	-	-	-
	dt_rf	11.89	1.71	2.76	**4.80**	**3.22**	**4.39**	**1.66**	0.25	-	-	-	-	-	-	-
senseit_sei	lpm_gbm	152.84	44.99	22.03	11.00	7.25	5.16	**5.43**	5.03	**5.68**	**5.31**	**5.64**	**5.08**	**5.56**	**5.08**	**5.39**
	lpm_rf	**165.59**	**63.54**	**26.69**	**13.95**	**8.19**	**5.50**	4.89	**5.04**	4.81	4.06	3.78	3.67	3.77	4.06	4.25
	dt_gbm	2.04	**1.06**	3.54	2.05	0.49	0.36	0.00	-	-	-	-	-	-	-	-
	dt_rf	**2.46**	0.43	**3.98**	**2.69**	**1.33**	**1.77**	**1.91**	-	-	-	-	-	-	-	-
covtype	lpm_gbm	**42.93**	42.99	**19.04**	3.86	2.93	3.83	6.01	3.81	5.80	6.38	**9.68**	**6.85**	4.22	8.91	3.86
	lpm_rf	27.67	**47.49**	11.85	**8.15**	**8.28**	**10.34**	**8.93**	**13.25**	**11.11**	**10.85**	9.44	6.81	**10.65**	**15.72**	**12.34**
	dt_gbm	146.18	99.02	51.83	**12.79**	5.68	**7.12**	**6.35**	**4.93**	**3.68**	0.00	**5.02**	**0.97**	0.00	**0.00**	0.00
	dt_rf	**154.63**	**104.78**	**61.40**	7.12	**10.47**	1.05	3.33	2.96	0.44	**1.84**	0.64	0.00	**0.39**	0.00	**2.01**
connect-4	lpm_gbm	37.97	18.54	**20.45**	**11.81**	**17.75**	**16.38**	**6.48**	6.90	**7.86**	**5.96**	**6.63**	**3.05**	2.52	**5.06**	3.81
	lpm_rf	**95.72**	**23.80**	19.45	11.74	8.71	4.73	3.76	5.66	5.14	4.32	2.43	2.31	**4.95**	0.60	**4.63**
	dt_gbm	156.06	**29.54**	**23.31**	6.22	3.47	4.10	**6.66**	**8.28**	0.00	**5.15**	2.08	**2.61**	**1.78**	0.11	0.00
	dt_rf	**177.05**	20.63	19.53	**18.35**	**15.63**	**14.00**	4.48	3.04	**4.06**	2.43	**3.33**	0.39	1.00	**0.32**	**0.42**

quantities! These distributions map closely to the *pmfs* at the leaf level (not discussed in Ghose and Ravindran (2020)), whereas Figure 5 shows the distributions along the height of the density tree. The corresponding plots for the DT are shown in Figure 22, Section A, Appendix.

Figure 12: The aggregated IBMMs are visualized for LPMs, when the oracle is a (a) GBM or (b) RF.

Going a step further we might wonder what the aggregated sampling distribution would look like if adjusted for the number of instances with a given uncertainty value. For example, we might see a peak on the extreme right for a dataset in the plots in Figure 12 simply because most points receive a high uncertainty score.

To visualize such an *adjusted distribution*, we slightly modify the logic used for the IBMM plots:

1. Let's say the model sizes are $k \in \{1, 2, ..., K\}$ for a dataset. For a size k, we denote the relative improvement by Δ_k.

2. n_k points are sampled from the IBMM obtained for size k, where $n_k \propto \Delta_k$. Points sampled thus, across the K model sizes, are pooled together into a set A.

3. n_k points are also sampled from the uncertainty distribution as obtained from the oracle. These points are pooled together into a set B.

4. Distributions p_A and p_B are learned over the sets A and B respectively. We use KDEs to fit A, B.

5. For a range of uncertainty values $u \in [0, 1]$, we compute $p_A(u)/p_B(u)$. We scale these values with a constant c s.t. $\sum_u c \cdot p_A(u)/p_B(u) = 1$ and plot them.

The scaling in the last step helps to visualize the plots across datasets on the same axes. These plots are shown in Figure 13. The corresponding plots for the DT are shown in Figure 22, Section A, Appendix.

While the plots in Figure 12 are indicative of the individual distributions for a model size (see Figure 24 in Section C, Appendix) owing to most of the individual distributions having similar shapes, this is not true for the adjusted plots in Figure 13 - there are diverse variations that are averaged out. We show some of them in Figure 14, for different datasets and model sizes, for the combination $model = LPM, oracle = GBM$. The size of the

33

Figure 13: Aggregated IBMMs, adjusted for the uncertainty distribution on the training data. These plots are for the LPM, using a (a) GBM or (b) RF as an oracle.

dots on the curve represent the density of the original uncertainty distribution around the corresponding x-value. These are intended to signify robustness of the adjustment.

It is probably important to point out here that typical discussions of uncertainty sampling, such as the classic version (Lewis & Gale, 1994), imply the non-adjusted distributions shown in Figure 12.

5.2.2 Comparison with Uncertainty Sampling

In this section, we compare our approach to an uncertainty sampling algorithm based on the heuristic strategy of favoring points with high uncertainties. However, we do not compare to the classic algorithm (Lewis & Gale, 1994), since that would not be fair given we have complete label information, and a full view of the uncertainty distribution from the oracle. We use a modified version, shown in Algorithm 4. We refer to this as *Supervised Uncertainty Sampling*, since it can use the *GBM* or *RF* oracles to guide its sampling.

In Algorithm 4:

1. The loop in lines 5-11 runs $\lceil |D_{train}|/b \rceil$ times, where every iteration adds the b most uncertain points to the current training dataset D_t. If b doesn't evenly divide $|D_{train}|$, the last iteration picks all remaining points.

2. In our implementation, $u_{M_\mathcal{O}}(x_i)$ in line 6 is precomputed and stored as a lookup table to reduce execution time.

3. In our experiments, we use a batch size $b = 10$. Note that this gives us optimal models as per Algorithm 4, for all batch sizes of the form $10k$, where $k \in \{1, 2, ..., \lfloor |D_{train}|/10 \rfloor\}$

The modified algorithm is a **significantly** more powerful version compared to the ones typically used in Active Learning setups, due to the following reasons:

1. We do not assume a cost for procuring or applying the oracle, which contrasts with the typical active learning setup. Thus, our oracle utilizes complete label information

Figure 14: Adjusted IBMMs for some model sizes and datasets, for $model = LPM, oracle = GBM$. We observe that fairly different distributions may be learned across our experiments.

Algorithm 4: Supervised Uncertainty Sampling

Data: Dataset D, model size η, $train_{\mathcal{O},h}()$, $train_{\mathcal{I},g}()$, batch size b

Result: Test set accuracy s_{test}, and interpretable model M^*

1 Create stratified splits $D_{train}, D_{val}, D_{test}$ from D

2 $M_O \leftarrow train_{\mathcal{O},h}(D_{train}, *)$

3 $I_{remaining} \leftarrow \{1, 2, ..., |D_{train}|\}$ be an index set of D_{train}

4 $I_{current} \leftarrow \{\}$

5 for $t \leftarrow 1$ **to** $\lceil |D_{train}|/b \rceil$ **do**

6 $I_U \leftarrow$ set of top b entries from $I_{remaining}$, based on $u_{M_{\mathcal{O}}}(x_i), i \in I_{remaining}$

7 $I_{remaining} \leftarrow I_{remaining} - I_U$

8 $I_{current} \leftarrow I_{current} \cup I_U$

9 $D_t \leftarrow \{D_{train,i} | i \in I_{current}\}$

10 $M_t \leftarrow train_{\mathcal{I},g}(D_t, \eta)$

11 $s_t \leftarrow accuracy(M_t, D_{val})$

12 end

13 $t^* \leftarrow \arg\max_t \{s_1, s_2, ..., s_{T-1}, s_T\}$

14 $M^* \leftarrow M_{t^*}$

15 $s_{test} \leftarrow accuracy(M^*, D_{test})$

16 return s_{test}, M^*

---

and our model has access to reliable uncertainty scores; this avoids the sample bias discussed in Section 3.2 (visualized in Figure 7).

2. Since we have complete label information, we have a validation set D_{val} available to us. In active learning, a validation set would be created from within the current labelled subset of data, which often makes it statistically insignificant or non-representative of the true distribution, especially at early iterations.

3. We do not have to estimate how many times the loop in lines 5-11 must run - this is executed till all data from D_{train} has been used up to train the model. Estimating the number of iterations is required when performing active learning since every iteration incurs a cost - that of calling the oracle to compute I_U. Consequently, here, we have the liberty of being able to *pick* the best model based on a validation set D_{val}.

Table 4 compares $LPMs$ and DTs learned via our technique with the ones produced by Algorithm 4. For each dataset, model and oracle combination we present two scores: (1) $\overline{SDI}$ and (2) *pct_better*, as defined in Section 5.1.2. Scores are compared across the same oracles, i.e., a score using oracle GBM in our method, is compared to a score from Supervised Uncertainty Sampling using a GBM.

Further points to note:

1. We introduce two special groupings:

- **ANY**: For each model size, the SDI score considered is the higher of the ones obtained from using the GBM or RF as oracles. The $\overline{SDI}$ and *pct_better* scores

Table 4: LPM, DT compared with Supervised Uncertainty Sampling

dataset	LPM			DT		
	GBM	RF	**ANY**	GBM	RF	**ANY**
cod-rna	0.23, 87.50%	-0.21, 50.00%	0.36, 87.50%	0.14, 50.00%	0.26, 60.00%	0.57, 90.00%
ijcnn1	0.24, 66.67%	0.10, 60.00%	0.44, 80.00%	-0.29, 35.71%	0.25, 80.00%	0.29, 80.00%
higgs	0.83, 100.00%	0.10, 60.00%	0.86, 100.00%	-0.05, 33.33%	0.52, 83.33%	0.63, 83.33%
covtype.binary	0.41, 93.33%	-0.05, 33.33%	0.48, 100.00%	-0.08, 22.22%	0.17, 45.45%	0.28, 54.55%
phishing	0.41, 86.67%	0.25, 100.00%	0.54, 100.00%	0.24, 40.00%	-0.15, 33.33%	0.41, 53.33%
a1a	0.04, 66.67%	-0.05, 40.00%	0.10, 73.33%	-0.31, 11.11%	0.53, 91.67%	0.61, 100.00%
pendigits	0.76, 100.00%	0.85, 100.00%	0.89, 100.00%	0.29, 61.54%	0.21, 50.00%	0.31, 57.14%
letter	0.95, 100.00%	0.95, 100.00%	0.98, 100.00%	0.50, 73.33%	0.14, 46.67%	0.62, 73.33%
Sensorless	0.02, 60.00%	0.44, 93.33%	0.46, 100.00%	0.65, 78.57%	0.44, 64.29%	0.65, 73.33%
senseit_aco	-0.01, 46.67%	0.07, 80.00%	0.11, 86.67%	0.79, 100.00%	0.47, 75.00%	0.79, 87.50%
senseit_sei	-0.11, 0.00%	0.02, 60.00%	0.07, 60.00%	0.08, 28.57%	-0.02, 42.86%	0.34, 57.14%
covtype	0.76, 100.00%	0.61, 93.33%	0.85, 100.00%	0.52, 66.67%	0.48, 60.00%	0.70, 73.33%
connect-4	0.17, 60.00%	0.10, 53.33%	0.44, 93.33%	0.04, 53.33%	0.21, 66.67%	0.45, 80.00%
OVERALL	0.37, 73.94%	0.26, 71.81%	0.51, 90.96%	0.21, 52.03%	0.26, 60.51%	0.50, 73.42%

are computed based on these scores. This grouping represents the ideal way to use our technique in practice: try multiple oracles and pick the best.

- **OVERALL**: This averages results across datasets, to provide an aggregate view of the comparison.

The entries defined by **OVERALL** and **ANY** provide comparison numbers aggregated over datasets, model sizes and oracles.

2. Favorable outcome values - $\overline{SDI} > 0$ or $pct\_better > 50$ - are colored green, unfavorable outcomes are colored red, and tied values are unformatted.

The color coding enables a quick overview: a table with a lot of green numbers imply the oracle based approach is better on an average, and vice-versa.

We observe our technique does better than Supervised Uncertainty sampling on average. Comparing the **OVERALL** +**ANY** entries, it seems like our technique works better for $LPMs$ than for DTs.

It is instructive to look at some specific adjusted IBMMs in the context of the relative performance of techniques. Figure 15 shows the plots from Figure 14 annotated with SDI scores. These are for $LPMs$ using GBM as the oracle.

The top row - (a), (b), (c) in Figure 15 - shows instances where our technique did much better ($SDI > 0$); it would seem that these are cases where sampling exclusively at high uncertainties is not an optimal distribution. Figure 15(d) shows a case where the optimal distribution is composed exclusively of high uncertainty points - so its not surprising that uncertainty sampling is at par with our technique ($SDI = 0$). (e) and (f) show similar trends.

While these plots are helpful in developing intuition for the underlying process, we would like to add the caveat that they are not conclusive in isolation. An example of this is (c) - it is not clear why uncertainty sampling does so poorly here. Possibly, instances with low uncertainties need to be sampled in a very specific manner that cannot be approximated by selecting the top n uncertain points, for any n.
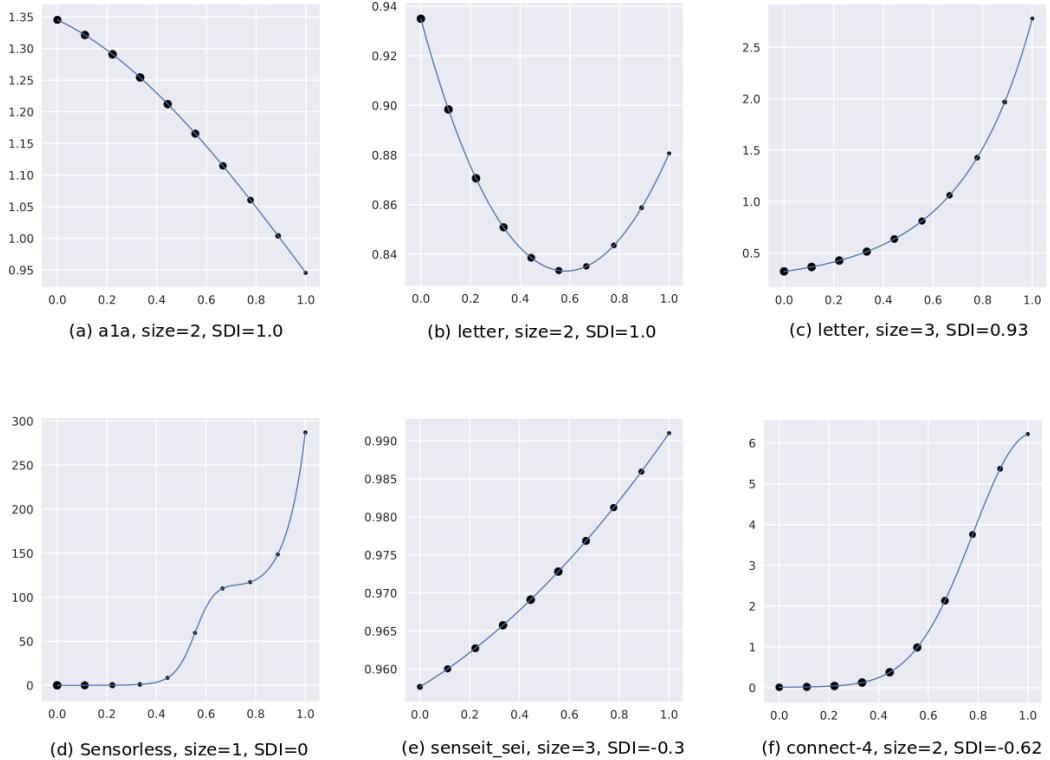
Figure 15: The same distributions as in Figure 14 are shown, now labelled with their *SDI* scores, measured against supervised uncertainty sampling. The top row - (a), (b), (c) - shows instances where our technique performed relatively better, and the bottom row shows cases where uncertainty sampling was competitive - (d) - or better - (e), (f).

5.2.3 Comparison with Density Tree Based Approach

We compare our technique against the density tree based approach to validate the information equivalences we have assumed between the techniques (discussed in Section 3.2, summarized in Table 1).

The comparison numbers for $LPMs$ and DTs are shown in Table 5. We follow the same format as in Table 4. Unlike the uncertainty sampling comparison, there is no notion of an oracle in the density tree approach, so, for a combination of dataset, model and model size, improved scores from using either the GBM or RF as the oracle are compared to the same reference score from the density tree based approach.

Table 5: LPM, DT compared to the Density Tree approach.

dataset	LPM			DT		
	GBM	RF	**ANY**	GBM	RF	**ANY**
cod-rna	-0.66, 0.00%	-0.69, 0.00%	-0.62, 0.00%	0.42, 66.67%	0.26, 66.67%	0.65, 88.89%
ijcnn1	0.30, 86.67%	0.08, 73.33%	0.34, 93.33%	0.26, 57.14%	0.74, 100.00%	0.74, 100.00%
higgs	-0.04, 33.33%	-0.11, 33.33%	0.03, 40.00%	-0.27, 20.00%	0.59, 100.00%	0.59, 100.00%
covtype.binary	-0.12, 40.00%	-0.36, 26.67%	-0.12, 40.00%	0.10, 55.56%	0.35, 80.00%	0.47, 90.00%
phishing	0.59, 93.33%	0.72, 100.00%	0.72, 100.00%	0.06, 26.67%	0.06, 33.33%	0.26, 46.67%
a1a	-0.11, 46.67%	-0.02, 60.00%	-0.02, 60.00%	-0.05, 55.56%	0.42, 72.73%	0.51, 81.82%
pendigits	0.62, 100.00%	0.55, 86.67%	0.64, 100.00%	0.14, 58.33%	0.20, 61.54%	0.20, 61.54%
letter	0.78, 100.00%	0.82, 100.00%	0.82, 100.00%	-0.07, 33.33%	-0.30, 13.33%	-0.01, 40.00%
Sensorless	0.49, 80.00%	0.65, 100.00%	0.66, 100.00%	-0.13, 28.57%	-0.31, 14.29%	-0.10, 26.67%
senseit_aco	0.54, 100.00%	0.58, 100.00%	0.59, 100.00%	0.46, 85.71%	0.29, 62.50%	0.30, 75.00%
senseit_sei	0.63, 93.33%	0.65, 100.00%	0.68, 100.00%	-0.15, 28.57%	0.49, 85.71%	0.58, 85.71%
covtype	-0.02, 46.67%	0.35, 86.67%	0.38, 86.67%	0.40, 66.67%	0.27, 53.33%	0.50, 73.33%
connect-4	0.55, 100.00%	0.45, 93.33%	0.62, 100.00%	-0.23, 33.33%	-0.20, 33.33%	0.05, 60.00%
OVERALL	0.31, 73.40%	0.32, 76.60%	0.40, 81.38%	0.08, 46.58%	0.17, 54.61%	0.33, 67.32%

Here too, the entries for the **OVERALL** and **ANY** groupings indicate that our improvements are better than the density tree based approach in general. We also observe that the performance gap is larger for $LPMs$ than for DTs.

5.3 Extension: Different Feature Spaces

We now consider an extension of our technique that showcases its flexibility. In our previous experiments, the feature vector representation was identical for the oracle and the interpretable model. This is also what Algorithm 3 implicitly assumes. Here, we consider the possibility of going a step further and using different feature vectors. If $f_{\mathcal{O}}$ and $f_{\mathcal{I}}$ are the feature vector creation functions for the oracle and the interpretable model respectively, and x_i is a "raw data" instance, then:

1. The oracle is trained on instances $f_{\mathcal{O}}(x_i)$, and provides uncertainties $u_{\mathcal{O}}(f_{\mathcal{O}}(x_i))$.

2. The interpretable model is provided with data $f_{\mathcal{I}}(x_i)$, but the uncertainty scores available to it are $u_{\mathcal{O}}(f_{\mathcal{O}}(x_i))$.

The motivation for using different feature spaces is that the combination $(\mathcal{O}, f_{\mathcal{O}})$ may be known to work well together and/or a pre-trained oracle might be available only for this combination.

We illustrate this application with the example of predicting nationalities from surnames of individuals. Our dataset (Rao & McMahan, 2019) contains examples from 18 nationalities: *Arabic, Chinese, Czech, Dutch, English, French, German, Greek, Irish, Italian, Japanese, Korean, Polish, Portuguese, Russian, Scottish, Spanish, Vietnamese.* The representations and models are as follows:

1. The oracle model is a *Gated Recurrent Unit (GRU)* (Cho, van Merriënboer, Gulcehre, Bahdanau, Bougares, Schwenk, & Bengio, 2014), that is learned on the sequence of characters in a surname. The GRU is calibrated with *temperature scaling* (Guo, Pleiss, Sun, & Weinberger, 2017).

2. The interpretable model is a DT, where the features are character n-grams, $n \in 1, 2, 3$. The entire training set is initially scanned to construct a n-gram vocabulary, which is then used to create a sparse binary vector per surname - 1s and 0s indicating the presence and absence of a n-gram respectively.

Figure 16 shows a schematic of the setup.

Figure 16: The feature representations for the oracle and the interpretable model may be different. Consider the name "Amy": the GRU is provided its letters, one at a time, in sequence, while the DT is given a n-gram representation of the name.

The n-gram representation leads to a vocabulary of ~ 5000 terms, that is reduced to 600 terms based on a χ^2-test in the interest of lower running time (see Section E, Appendix, for details). DTs of different $depth \leq 15$ were trained. A budget of $T = 3000$ iterations was used, and the relative improvement in the $F1$ macro score (as in Equation 10) is reported, averaged over three runs. Figure 17 shows the results.

We see large improvements at small depths, that peak with $\delta F1 = 83.04\%$ at $depth = 3$, and then again at slightly larger depths, which peak at $depth = 9$ with $\delta F1 = 12.34\%$.

To obtain a qualitative idea of the changes in the DT using a oracle produces, we look at the prediction rules for *Polish* surnames, when DT $depth = 3$. For each rule, we also present examples of true and false positives.

Figure 17: Improvements $\delta F1$ are shown for different depths of the DT.

Baseline rules - $precision = 2.99\%, recall = 85.71\%, F1 = 5.77\%$:

Rule 1. $k \wedge ski \wedge \neg v$

- True Positives: *jaskolski, rudawski*
- False Positives: *skipper (English), babutski (Russian)*

Rule 2. $k \wedge \neg ski \wedge \neg v$

- True Positives: *wawrzaszek, koziol*
- False Positives: *konda (Japanese), jagujinsky (Russian)*

Oracle-based DT rules - $precision = 25.00\%, recall = 21.43\%, F1 = 23.08\%$:

Rule 1. $ski \wedge \neg(b \vee kin)$

- True Positives: *jaskolski, rudawski*
- False Positives: *skipper (English), aivazovski (Russian)*

We note that the baseline rules are in conflict w.r.t. the literal "ski", and taken together, they simplify to $k \wedge \neg v$. This makes them extremely permissive, especially *Rule 2*, which requires the literal "k" while needing "ski" and 'v' to be absent. Not surprisingly, these rules have high recall ($= 85.71\%$) but poor precision ($= 2.99\%$), leading to $F1 = 5.77\%$.

In the case of the oracle-based DT, now we have only one rule, that requires the atypical trigram "ski". This improves precision ($= 25\%$), trading off recall ($= 21.43\%$), for a significantly improved $F1 = 23.08\%$.

The difference in rules may also be visualized by comparing the distribution of nationalities represented in their false positives, as in Figure 18. We see that the baseline DT rules, especially *Rule 2*, predict many nationalities, but in the case of the DT learned using the oracle, the model confusion is concentrated around *Russian* names, which is reasonable given the shared *Slavic* origin of many *Polish* and *Russian* names.

41

Figure 18: The distribution of nationalities in false positive predictions for the baseline and oracle based models, shown for predicting *Polish* names. Only nationalities with non-zero counts are shown.

We believe this is a particularly powerful and exciting application of our technique, and opens up a wide range of possibilities for translating information between models of varied capabilities.

5.4 Summary

We summarize our observations from our experiments here:

1. For all combinations of interpretable and oracle models - $\{LPM, DT\} \times \{GBM, RF\}$ - we see good improvements, $\delta F1$, especially at small sizes (Section 5.2.1). Sometimes these may be $> 100\%$. As model sizes increase beyond a point, $\delta F1 \to 0$.

2. We observed that the precise relationship of the sampling distribution and the uncertainty needs to be learned, and a heuristic strategy of exclusively sampling high uncertainty points is not optimal in general (Section 5.2.2). We believe this is an important result, especially given that this is true for the supervised version of uncertainty sampling, where the known biases of Figure 7, Section 3.2, do not apply.

3. Our approach produces better accuracy, in general, compared to both the supervised uncertainty sampling approach in Algorithm 4 and the density tree approach (Ghose & Ravindran, 2020). The latter set of results empirically justify the approximations made in extending the density tree based approach, especially points 1 and 2 in Table 1.

   The results from Sections 5.2.2 and 5.2.3 are summarized in Table 6. Recall that the combination **OVERALL + ANY** averages over datasets and oracles; Table 6 shows these summary statistics.

Table 6: Summary comparison results, **OVERALL + ANY**

compared to	model	$\overline{SDI}$	pct_better
supervised uncertainty sampling	LPM	0.51	90.96%
	DT	0.50	73.42%
density trees	LPM	0.40	81.38%
	DT	0.33	67.32%

We note that density trees are more competitive to our technique than supervised uncertainty sampling: smaller $\overline{SDI}$ and pct_better scores.

4. Section 5.3 showcases an exciting application of the technique where the feature representations between the interpretable model and the oracle need not be identical. This considerably broadens the scope of our work.

5. The oracle based approach is better than the density approach in the following ways:

   (a) Greater improvements to the baseline in general.

   (b) It is much more flexible, since:

   - an oracle from an arbitrary model family may be used.
   - the oracle need not use the same feature representation as the interpretable model.

6. Discussion

Having looked at both the theory and empirical outcomes, we revisit a few points of interest in this section.

1. **Effect of flattening**: We first consider the question: how much does flattening (Section 4.5) help? Table 7 shows the effect on *improved $F1$ scores* due to transforming the uncertainty distributions for the datasets `Sensorless` and `covtype.binary`, for model $size \in \{1, 2, 3\}$. This is shown for the combination of $model = LPM$ and $oracle = GBM$, with two different experiment settings. The scores were averaged over three trials. Setting 1 is what we have used in the experiments discussed in the previous sections: maximum allowed $Beta$ components are 500 and $scale = 10000$. Setting 2 looks at much lower values of these parameters: maximum allowed components is 50 and $scale = 10$.

We observe that while flattening influences results, other parameters may additionally decide the magnitude of the influence. At Setting 1, `Sensorless` is affected at $size = 1$, but at higher sizes the differences seem to be from random variations across trials. At Setting 2 however, the differences are seen for $size \in \{1, 2, 3\}$. For `covtype.binary` only $size = 2$ seems to be affected in either setting.

Recall we had noted in Figure 9 that the datasets `Sensorless` and `covtype.binary` have non-smooth and smooth uncertainty distributions respectively. The observations

43

Table 7: Improved scores averaged over three trials, shown for different parameter settings, with and without flattening. Here, Setting 1 is $\{max\_components = 500, scale = 10000\}$ and Setting 2 is $\{max\_components = 50, scale = 10\}$. "curr." signifies this is the current setting for our other experiments, while "low" signifies lower values of parameters relative to our current setting.

		Setting 1 (curr.)			Setting 2 (low)		
dataset	dist.	1	2	3	1	2	3
Sensorless	original	0.39	0.54	0.57	0.38	0.42	0.41
	flattened	0.44	0.53	0.55	0.43	0.54	0.59
covtype.binary	original	0.66	0.69	0.71	0.64	0.66	0.71
	flattened	0.68	0.73	0.73	0.65	0.71	0.71

in Table 7 align well with the expectation that `Sensorless` is positively affected by the transformation, while results for `covtype.binary` remain mostly unchanged.

Our takeaway is that for non-smooth uncertainty distributions, flattening makes our technique robust across parameter settings. It does not affect smooth distributions in a significant way.

2. **Alternative Parameterization**: Instead of using shape variables in our optimization, which lie in the interval $(0, \infty)$, one might wonder if its simpler to optimize based on the mean, $\mu \in [0, 1]$ (bounded by the range of uncertainty values), and standard deviation, $\sigma \in [0, 0.5]$ (a property of the *Beta* distribution). While this is appealing, we need to account for the fact that μ and σ do not independently identify a *Beta* distribution, unlike the case for the *Normal distribution*. In other words, not all values of $\mu \in [0, 1], \sigma \in [0, 0.5]$ lead to valid shape parameters. The optimization would need to model this dependence, and we would lose our current convenience of using only box constraints. The scatter plot in Figure 19 marks the different combinations of μ and σ for which valid *Beta* distributions exist. An example of this dependence is $\mu \to 1 \implies \sigma \to 0$.

Figure 19: Blue dots indicate a valid *Beta* distribution exists for the corresponding mean and standard deviation values.

3. **Measuring compaction**: We mentioned in the Introduction, Section 1, that a possible area of application of this work might be model compression. We would like to point out that the *compaction profile* (Figure 11, Figure 23) plots emphasize this use-case: they're essentially a tool to determine the minimal model size achievable using our technique, given a baseline model size.

To formalize this connection, we introduce the score *Compaction Index (CI)* that denotes the extent of model size decrease possible, up to a size where $\delta F1 \approx 0$. Figure 20 shows a sample compaction profile. The CI score, where $CI \in [0, 1]$, is the ratio of the area in red to the area in green.

Figure 20: Compaction Index

The more reduction in model size our technique can obtain, the closer the red curve is to the green boundary, and $CI \approx 1$. If no reduction is possible at any model size, the red line coincides with the diagonal and $CI = 0$. Clearly, this score is specific to a model family $\mathcal{F}$, a training algorithm f and a specific notion of model size. And ideally, this should be averaged over all possible datasets and oracles.

Here are the CI scores for our experiments :

- $LPM : CI = 0.57$
- $DT : CI = 0.16$

These scores indicate that $LPMs$ may be compacted better than DTs - this may also be seen from the plots in Figure 10, where the improvements for DTs decrease faster, with growing model size, than those for $LPMs$.

4. **Upper bound of improvements**: In Equation 2, and then in Equations 3 and 4, the improved accuracy of the interpretable model is shown bounded by the oracle accuracy. For example, see the rightmost term in Equation 2, reproduced below:

$$accuracy(M_{\mathcal{I}p\eta}, p) \leq accuracy(M_{\mathcal{I}q\eta}, p) \leq accuracy(M_{\mathcal{O}p*}, p) \qquad (13)$$

We empirically show this to be true now. In Figure 21, we show the distribution of relative difference between the improved accuracy of a LPM model and the accuracy of a GBM oracle.

Using the notation in the equation above, we calculate the relative difference $\Delta F1$ as:

$$\Delta F1 = \frac{accuracy(M_{\mathcal{I}q\eta}, p) - accuracy(M_{\mathcal{O}p*}, p)}{accuracy(M_{\mathcal{O}p*}, p)} \tag{14}$$

Here, of course, we measure accuracy using the $F1$ macro score.

Figure 21: Distribution of %age accuracy difference from the oracle accuracy.

There is one distribution plotted per dataset, where the distribution uses information from multiple runs, for multiple model sizes. It may be seen in Figure 21 that all relative differences are at most 0 (there is some spillover to the right of 0 due to the distribution fit).

For precise numbers, we look at Table 8, which lists the %age of cases where the interpretable model's accuracy exceeded that of the oracle, and the average value of the relative difference for the cases where it is positive. Such cases seem to be insignificant. See Figure 25, Appendix, for plots for other model-oracle combinations.

Table 8: The percentage of cases where we see positive relative difference w.r.t. oracle, and the mean of these positive difference are shown.

model	oracle	%age positive cases	mean positive value
LPM	GBM	0.00%	–
LPM	RF	6.03%	1.60
DT	GBM	3.86%	2.31
DT	RF	3.22%	0.83

7. Conclusion

In this paper we introduced a technique to learn an interpretable model, that reduces the trade-off between model size and accuracy. The practical implication of this work is: instead

of selecting among different interpretable model families on account of their accuracy, one might try our technique first to see if a preferred model might be improved.

Producing an accurate model is formulated as an optimization problem of identifying training data that maximizes learning. This process is aided by an oracle model. Our technique is shown to possess these favorable properties: (a) the optimization uses a fixed set of seven variables, irrespective of the dimensionality of the data, making it a practical tool (b) it is model-agnostic: the interpretable and oracle models may be from any model family (c) the technique may be used even when the feature spaces of the interpretable model and oracle are different.

We have provided extensive empirical validation to establish the utility of the technique.

At a deeper level, this work also presents some intriguing findings : (a) train and test distributions need not be identical for optimal learning (b) all our observations point to a "small model effect": most improvements occur when the model size is small. These observations confirm the results of our earlier work (Ghose & Ravindran, 2020).

We believe that the general theme of the proposed technique, that of shaping data density to influence accuracy, as well as the deeper results, offer promising directions for future research.

Appendices

A. Uncertainty Distribution for DT

The uncertainty distributions learned when using a DT with different oracles are shown in
Figure 22. The first row shows visualizes the aggregation of the IBMMs that were learned,
while the second row shows them adjusted with the uncertainty distribution from the oracle.
These are analogues of the LPM plots in Figure 12 and Figure 13.

Figure 22: The aggregated IBMMs visualized when using a DT as our interpretable model.
The top row shows the aggregated IBMMs for different oracles: GBM (left) and
RF (right). The bottom row visualizes the IBMMs adjusted for the uncertainty
distribution.

The patterns we observe here are similar to what we saw for LPMs:

1. Top-row: the IBMMs seem to prefer both low and high uncertainty regions.

2. Bottom-row: when adjusted with the oracle's uncertainty distribution, there is sam-
pling across the entire range of uncertainty values, with slight/occasional preference
for higher uncertainties.

B. Compaction Profiles

Figure 23 shows the compaction profiles for all model-oracle combinations. These are discussed in Section 5.2.1, in reference to Figure 11.

Figure 23: For different combinations of models and oracles: $\{LPM, DT\} \times \{GBM, RF\}$, these plots show the size of an improved model (y-axis), that may replace a traditionally trained model of a given size (x-axis). A model is considered as a replacement for another if its accuracy is at least as high as the latter.

C. IBMMs for Different Model Sizes

Figure 24 shows the IBMMs learned over uncertainties for individual model sizes of the LPM, with GBM as the oracle,. These are *not* adjusted with the density of the uncertainty distribution. The plot shows them for the datasets (a) `covtype.binary` and `Sensorless`. We observe that the unified IBMM weighted by improvements, shown in Figure 12, are indicative of the individual distributions in this Figure 24.

49

Figure 24: IBMM distributions for model sizes $\{1, 2, ..., 15\}$, for the datasets (a) `covtype.binary` and (b) `Sensorless`. These are for the combination of using LPM as the model with GBM as an oracle. Darker curves indicate higher model sizes.

D. Improvements Relative to Oracle

Some of the positive values we see in Figure 25 may be attributed to spillovers due to the *kde* fit. Their magnitudes and occurrences are typically small: these are detailed in Table 8.

E. Feature Selection for n-gram DT

For the experiments in Section 5.3, we perform feature selection to reduce their running time. After the n-gram ($n \in \{1, 2, 3\}$) vocabulary is created from the training data, we perform a χ^2-test to select the $k-$best features. The original number of features is 5308. To pick the smallest useful set of features, we test different values of $k \leq 1000$. A test constitutes of:

1. Construct a DT, for a given max_depth, on the original set of features. Obtain its test accuracy, $F1_{all}$.

2. Construct a DT, with the same max_depth, using only the k best features as per the χ^2-test, and obtain its test accuracy $F1_k$.

3. Report:
$$\delta F1 = 100 \times \frac{F1_k - F1_{all}}{F1_{all}}$$

We use the "macro" averaging for the $F1$ score to be consistent with other experiments in the paper. All reported $\delta F1$ are *averaged over ten runs*.

Figure 26 shows how $\delta F1$ varies with k.

Figure 25: These plots show the distribution of the percentage relative difference of a model's improved score w.r.t. to the accuracy of the oracle it is trained with. We note that this quantity is almost always non-positive as claimed in Equations 2, 3 and 4.

Figure 26: The relationship between $\delta F1$ and $k \leq 1000$. Each data point is an average over ten runs.

We observe that at around 600 features, $\delta F1 \approx 0\%$. The only exception is the case for $max\_depth = 3$, but that is admissible since $\delta F1 > 0$, i.e., we seem to be improving the accuracy .

References

Alimoglu, F., & Alpaydin, E. (1996). Methods of combining multiple classifiers based on different representations for pen-based handwritten digit recognition. In *Proceedings of the Fifth Turkish Artificial Intelligence and Artificial Neural Networks Symposium (TAINN 96*.

Ancona, M., Oztireli, C., & Gross, M. (2019). Explaining deep neural networks with a polynomial time algorithm for shapley value approximation. In Chaudhuri, K., & Salakhutdinov, R. (Eds.), *Proceedings of the 36th International Conference on Machine Learning*, Vol. 97 of *Proceedings of Machine Learning Research*, pp. 272–281, Long Beach, California, USA. PMLR.

Anderson, B., & Moore, A. (2005). Active learning for hidden markov models: Objective functions and algorithms. In *Proceedings of the 22Nd International Conference on Machine Learning*, ICML '05, pp. 9–16, New York, NY, USA. ACM.

Angelino, E., Larus-Stone, N., Alabi, D., Seltzer, M., & Rudin, C. (2017). Learning certifiably optimal rule lists. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '17, pp. 35–44, New York, NY, USA. ACM.

Angwin, J., Larson, J., Mattu, S., & Kirchner, L. (2016). Machine Bias. https://www.propublica.org/article/machine-bias-risk-assessments-in-criminal-sentencing.

Baldi, P., Sadowski, P., & Whiteson, D. (2014). Searching for exotic particles in high-energy physics with deep learning. *Nature Communications*, *5*(1), 4308.

Beluch, W. H., Genewein, T., Nurnberger, A., & Kohler, J. M. (2018). The power of ensembles for active learning in image classification. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 9368–9377, Los Alamitos, CA, USA. IEEE Computer Society.

Bergstra, J., Yamins, D., & Cox, D. D. (2013). Making a science of model search: Hyperparameter optimization in hundreds of dimensions for vision architectures. In *Proceedings of the 30th International Conference on International Conference on Machine Learning - Volume 28*, ICML'13, pp. I–115–I–123. JMLR.org.

Bergstra, J., Bardenet, R., Bengio, Y., & Kégl, B. (2011). Algorithms for hyper-parameter optimization. In *Proceedings of the 24th International Conference on Neural Information Processing Systems*, NIPS'11, pp. 2546–2554, USA. Curran Associates Inc.

Bertsimas, D., Delarue, A., Jaillet, P., & Martin, S. (2019). The price of interpretability. *CoRR*, *abs/1907.03419*.

Blackwell, D., & MacQueen, J. B. (1973). Ferguson distributions via polya urn schemes. *Ann. Statist.*, *1*(2), 353–355.

Box, G. E. P., & Cox, D. R. (1964). An analysis of transformations. *Journal of the Royal Statistical Society. Series B (Methodological)*, *26*(2), 211–252.

Breiman, L., et al. (1984). *Classification and Regression Trees*. Chapman & Hall, New York.

Brochu, E., Cora, V. M., & de Freitas, N. (2010). A tutorial on bayesian optimization of expensive cost functions, with application to active user modeling and hierarchical reinforcement learning. *CoRR*, *abs/1012.2599*.

Caruana, R., Lou, Y., Gehrke, J., Koch, P., Sturm, M., & Elhadad, N. (2015). Intelligible models for healthcare: Predicting pneumonia risk and hospital 30-day readmission. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '15, pp. 1721–1730, New York, NY, USA. ACM.

Castellanos, S., & Nash, K. S. (2018). Bank of America Confronts AI's 'Black Box' With Fraud Detection Effort. `https://blogs.wsj.com/cio/2018/05/11/bank-of-america-confronts-ais-black-box-with-fraud-detection-effort/`.

Chang, C.-C., & Lin, C.-J. (2001). Ijcnn 2001 challenge: Generalization ability and text decoding. In *In Proceedings of IJCNN. IEEE*, pp. 1031–1036.

Chang, C.-C., & Lin, C.-J. (2011). LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, *2*, 27:1–27:27. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

Cho, K., van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., & Bengio, Y. (2014). Learning phrase representations using RNN encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 1724–1734, Doha, Qatar. Association for Computational Linguistics.

Clarke, Y. D. (2019). Algorithmic Accountability Act of 2019. `https://www.congress.gov/bill/116th-congress/house-bill/2231`.

Collobert, R., Bengio, S., & Bengio, Y. (2002). A parallel mixture of svms for very large scale problems. In Dietterich, T. G., Becker, S., & Ghahramani, Z. (Eds.), *Advances in Neural Information Processing Systems 14*, pp. 633–640. MIT Press.

Dai, W., Yang, Q., Xue, G.-R., & Yu, Y. (2007). Boosting for transfer learning. In *Proceedings of the 24th International Conference on Machine Learning*, ICML '07, pp. 193–200, New York, NY, USA. ACM.

Dai, Z., Yu, H., Low, B. K. H., & Jaillet, P. (2019). Bayesian optimization meets Bayesian optimal stopping. In Chaudhuri, K., & Salakhutdinov, R. (Eds.), *Proceedings of the 36th International Conference on Machine Learning*, Vol. 97 of *Proceedings of Machine Learning Research*, pp. 1496–1506, Long Beach, California, USA. PMLR.

Dasgupta, S. (2011). Two faces of active learning. *Theor. Comput. Sci.*, *412*(19), 1767–1781.

Dasgupta, S., & Hsu, D. (2008). Hierarchical sampling for active learning. In *Proceedings of the 25th International Conference on Machine Learning*, ICML '08, pp. 208–215, New York, NY, USA. ACM.

Dean, D. J., & Blackard, J. A. (1998). Comparison of neural networks and discriminant analysis in predicting forest cover types..

Desai, S., & Ramaswamy, H. G. (2020). Ablation-cam: Visual explanations for deep convolutional network via gradient-free localization. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*.

Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2019). BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pp. 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Dua, D., & Graff, C. (2017). UCI machine learning repository..

Duarte, M. F., & Hu, Y. H. (2004). Vehicle classification in distributed sensor networks. *J. Parallel Distrib. Comput.*, *64*(7), 826838.

Efron, B., Hastie, T., Johnstone, I., & Tibshirani, R. (2004). Least angle regression. *Ann. Statist.*, *32*(2), 407–499.

Gal, Y., Islam, R., & Ghahramani, Z. (2017). Deep bayesian active learning with image data. In *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, ICML'17, pp. 1183–1192. JMLR.org.

Ghose, A., & Ravindran, B. (2020). Interpretability with accurate small models. *Frontiers in Artificial Intelligence*, *3*, 3.

Goodman, B., & Flaxman, S. (2017). European union regulations on algorithmic decision-making and a "right to explanation". *AI Magazine*, *38*, 50–57.

Graves, A. (2013). Generating sequences with recurrent neural networks. *CoRR*, *abs/1308.0850*.

Grill, J.-B., Valko, M., Munos, R., & Munos, R. (2015). Black-box optimization of noisy functions with unknown smoothness. In Cortes, C., Lawrence, N. D., Lee, D. D., Sugiyama, M., & Garnett, R. (Eds.), *Advances in Neural Information Processing Systems 28*, pp. 667–675. Curran Associates, Inc.

Gunning, D. (2016). Explainable Artificial Intelligence. https://www.darpa.mil/program/explainable-artificial-intelligence.

Guo, C., Pleiss, G., Sun, Y., & Weinberger, K. Q. (2017). On calibration of modern neural networks. In *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, ICML17, p. 13211330. JMLR.org.

Hafner, D., Tran, D., Lillicrap, T. P., Irpan, A., & Davidson, J. (2019). Noise contrastive priors for functional uncertainty. In *Proceedings of the Thirty-Fifth Conference on Uncertainty in Artificial Intelligence, UAI 2019, Tel Aviv, Israel, July 22-25, 2019*, p. 332.

Hansen, N., & Kern, S. (2004). Evaluating the CMA evolution strategy on multimodal test functions. In Yao, X., et al. (Eds.), *Parallel Problem Solving from Nature PPSN VIII*, Vol. 3242 of *LNCS*, pp. 282–291. Springer.

Hansen, N., & Ostermeier, A. (2001). Completely derandomized self-adaptation in evolution strategies. *Evol. Comput.*, *9*(2), 159–195.

Herman, B. (2017). The promise and peril of human evaluation for model interpretability. Presented at NIPS 2017 Symposium on Interpretable Machine Learning. Available at: https://arxiv.org/abs/1711.09889v3.

Howard, J., & Ruder, S. (2018). Universal language model fine-tuning for text classification. In *ACL*. Association for Computational Linguistics.

Hsu, C.-W., & Lin, C.-J. (2002). A comparison of methods for multiclass support vector machines. *IEEE transactions on neural networks / a publication of the IEEE Neural Networks Council, 13*, 415–25.

Hu, X., Rudin, C., & Seltzer, M. (2019). Optimal sparse decision trees. In Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E., & Garnett, R. (Eds.), *Advances in Neural Information Processing Systems 32*, pp. 7265–7273. Curran Associates, Inc.

Hutter, F., Hoos, H. H., & Leyton-Brown, K. (2011). Sequential model-based optimization for general algorithm configuration. In *Proceedings of the 5th International Conference on Learning and Intelligent Optimization*, LION'05, pp. 507–523, Berlin, Heidelberg. Springer-Verlag.

Ioffe, S., & Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proceedings of the 32nd International Conference on International Conference on Machine Learning - Volume 37*, ICML15, p. 448456. JMLR.org.

Japkowicz, N., & Stephen, S. (2002). The class imbalance problem: A systematic study. *Intell. Data Anal., 6*(5), 429–449.

Juan, Y., Zhuang, Y., Chin, W.-S., & Lin, C.-J. (2016). Field-aware factorization machines for ctr prediction. In *Proceedings of the 10th ACM Conference on Recommender Systems*, RecSys 16, p. 4350, New York, NY, USA. Association for Computing Machinery.

Kamishima, T., Hamasaki, M., & Akaho, S. (2009). Trbagg: A simple transfer learning method and its application to personalization in collaborative tagging. In *Proceedings of the 2009 Ninth IEEE International Conference on Data Mining*, ICDM '09, pp. 219–228, Washington, DC, USA. IEEE Computer Society.

Ke, G., Meng, Q., Finley, T., Wang, T., Chen, W., Ma, W., Ye, Q., & Liu, T.-Y. (2017). Lightgbm: A highly efficient gradient boosting decision tree. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, pp. 3149–3157, USA. Curran Associates Inc.

Kingma, D. P., & Ba, J. (2015). Adam: A method for stochastic optimization. *CoRR, abs/1412.6980*.

Koh, P. W., & Liang, P. (2017). Understanding black-box predictions via influence functions. In Precup, D., & Teh, Y. W. (Eds.), *Proceedings of the 34th International Conference on Machine Learning*, Vol. 70 of *Proceedings of Machine Learning Research*, pp. 1885–1894, International Convention Centre, Sydney, Australia. PMLR.

Lakkaraju, H., Bach, S. H., & Leskovec, J. (2016). Interpretable decision sets: A joint framework for description and prediction. In *Proceedings of the 22Nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, pp. 1675–1684, New York, NY, USA. ACM.

Larson, J., Mattu, S., Kirchner, L., & Angwin, J. (2016). How We Analyzed the COMPAS Recidivism Algorithm. `https://www.propublica.org/article/how-we-analyzed-the-compas-recidivism-algorithm`.

Letham, B., Rudin, C., McCormick, T. H., & Madigan, D. (2013). Interpretable classifiers using rules and bayesian analysis: Building a better stroke prediction model. *CoRR*, *abs/1511.01644*.

Lewis, D. D., & Gale, W. A. (1994). A sequential algorithm for training text classifiers. In *Proceedings of the 17th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '94, pp. 3–12, New York, NY, USA. Springer-Verlag New York, Inc.

Liao, X., Xue, Y., & Carin, L. (2005). Logistic regression with an auxiliary data source. In *Proceedings of the 22Nd International Conference on Machine Learning*, ICML '05, pp. 505–512, New York, NY, USA. ACM.

Lim, M., & Hastie, T. (2015). Learning interactions via hierarchical group-lasso regularization. *J Comput Graph Stat*, *24*(3), 627–654. 26759522[pmid].

Lipton, Z. C. (2018). The mythos of model interpretability. *Queue*, *16*(3), 30:31–30:57.

Lou, Y., Caruana, R., Gehrke, J., & Hooker, G. (2013). Accurate intelligible models with pairwise interactions. In *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '13, pp. 623–631, New York, NY, USA. ACM.

Lundberg, S. M., & Lee, S.-I. (2017). A unified approach to interpreting model predictions. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., & Garnett, R. (Eds.), *Advances in Neural Information Processing Systems 30*, pp. 4765–4774. Curran Associates, Inc.

Malkomes, G., & Garnett, R. (2018). Automating bayesian optimization with bayesian optimization. In Bengio, S., Wallach, H., Larochelle, H., Grauman, K., Cesa-Bianchi, N., & Garnett, R. (Eds.), *Advances in Neural Information Processing Systems 31*, pp. 5984–5994. Curran Associates, Inc.

Michie, D., Spiegelhalter, D. J., Taylor, C. C., & Campbell, J. (Eds.). (1995). *Machine Learning, Neural and Statistical Classification*. Ellis Horwood, USA.

Mihalkova, L., & Mooney, R. (2006). Transfer learning with markov logic networks. In *Proceedings of the ICML-06 Workshop on Structural Knowledge Transfer for Machine Learning*, Pittsburgh, PA.

Mohammad, R. M., Thabtah, F., & McCluskey, L. (2012). An assessment of features related to phishing websites using an automated technique. In *2012 International Conference for Internet Technology and Secured Transactions*, pp. 492–497.

Mood, C. (2010). Logistic regression : Why we cannot do what we think we can do, and what we can do about it. *European Sociological Review*, *26*(1), 67–82.

Ohlssen, D. I., Sharples, L. D., & Spiegelhalter, D. J. (2007). Flexible random-effects models using bayesian semi-parametric models: applications to institutional comparisons. *Statistics in Medicine*, *26*(9), 2088–2112.

Pan, S. J., & Yang, Q. (2010). A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering, 22*(10), 1345–1359.

Paschke, F., Bayer, C., Bator, M., Mnks, U., Dicks, A., Enge-Rosenblatt, O., & Lohweg, V. (2013). Sensorlose zustandsberwachung an synchronmotoren. In *Proceedings of Computational Intelligence Workshop*.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., & Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research, 12*, 2825–2830.

Platt, J. (1998). Fast training of support vector machines using sequential minimal optimization. In *Advances in Kernel Methods - Support Vector Learning* (Advances in Kernel Methods - Support Vector Learning edition). MIT Press.

Platt, J. C. (1999). Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. In *ADVANCES IN LARGE MARGIN CLASSIFIERS*, pp. 61–74. MIT Press.

Prokhorov, D. (2001). IJCNN 2001 Neural Network Competition. `http://www.geocities.ws/ijcnn/nnc_ijcnn01.pdf`.

Pröllochs, N., Feuerriegel, S., & Neumann, D. (2019). Learning interpretable negation rules via weak supervision at document level: A reinforcement learning approach. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pp. 407–413, Minneapolis, Minnesota. Association for Computational Linguistics.

Quinlan, J. R. (1993). *C4.5: Programs for Machine Learning.* Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

Quinlan, J. R. (2004). C5.0. `https://rulequest.com/`.

Rao, D., & McMahan, B. (2019). *Natural Language Processing with PyTorch.* O'Reilly. `https://www.amazon.com/Natural-Language-Processing-PyTorch-Applications/dp/1491978236/` and `https://github.com/joosthub/PyTorchNLPBook`.

Rasmussen, C. E. (1999). The infinite gaussian mixture model. In *Proceedings of the 12th International Conference on Neural Information Processing Systems*, NIPS'99, pp. 554–560, Cambridge, MA, USA. MIT Press.

Ribeiro, M. T., Singh, S., & Guestrin, C. (2016). "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, pp. 1135–1144, New York, NY, USA. ACM.

Ribeiro, M. T., Singh, S., & Guestrin, C. (2018). Anchors: High-precision model-agnostic explanations..

Scheffer, T., Decomain, C., & Wrobel, S. (2001). Active hidden markov models for information extraction. In *Proceedings of the 4th International Conference on Advances in Intelligent Data Analysis*, IDA '01, pp. 309–318, London, UK, UK. Springer-Verlag.

Selvaraju, R. R., Cogswell, M., Das, A., Vedantam, R., Parikh, D., & Batra, D. (2017). Grad-cam: Visual explanations from deep networks via gradient-based localization. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pp. 618–626.

Settles, B. (2009). Active learning literature survey. Computer sciences technical report 1648, University of Wisconsin–Madison.

Shahriari, B., Swersky, K., Wang, Z., Adams, R. P., & de Freitas, N. (2016). Taking the human out of the loop: A review of bayesian optimization. *Proceedings of the IEEE*, *104*(1), 148–175.

Simonyan, K., & Zisserman, A. (2015). Very deep convolutional networks for large-scale image recognition. In *International Conference on Learning Representations*.

Snoek, J., Larochelle, H., & Adams, R. P. (2012). Practical bayesian optimization of machine learning algorithms. In Pereira, F., Burges, C. J. C., Bottou, L., & Weinberger, K. Q. (Eds.), *Advances in Neural Information Processing Systems 25*, pp. 2951–2959. Curran Associates, Inc.

Snoek, J., Rippel, O., Swersky, K., Kiros, R., Satish, N., Sundaram, N., Patwary, M. M. A., Prabhat, P., & Adams, R. P. (2015). Scalable bayesian optimization using deep neural networks. In *Proceedings of the 32Nd International Conference on International Conference on Machine Learning - Volume 37*, ICML'15, pp. 2171–2180. JMLR.org.

Thrun, S., & Mitchell, T. M. (1994). Learning one more thing. In *IJCAI*.

Tong, S., & Koller, D. (2002). Support vector machine active learning with applications to text classification. *J. Mach. Learn. Res.*, *2*, 45–66.

Torrey, L., & Shavlik, J. W. (2009). Chapter 11 transfer learning..

Ustun, B., & Rudin, C. (2016). Supersparse linear integer models for optimized medical scoring systems. *Machine Learning*, *102*(3), 349–391.

Uzilov, A. V., Keegan, J. M., & Mathews, D. H. (2006). Detection of non-coding rnas on the basis of predicted secondary structure formation free energy change. *BMC bioinformatics*, *7*, 173–173. 16566836[pmid].

Wang, C.-C., Tan, K. L., Chen, C.-T., Lin, Y.-H., Keerthi, S. S., Mahajan, D., Sundararajan, S., & Lin, C.-J. (2018). Distributed newton methods for deep neural networks. *Neural Comput.*, *30*(6), 16731724.

Wang, T. (2018). Multi-value rule sets for interpretable classification with feature-efficient representations. In Bengio, S., Wallach, H., Larochelle, H., Grauman, K., Cesa-Bianchi, N., & Garnett, R. (Eds.), *Advances in Neural Information Processing Systems 31*, pp. 10835–10845. Curran Associates, Inc.

Weiss, K., Khoshgoftaar, T. M., & Wang, D. (2016). A survey of transfer learning. *Journal of Big Data*, *3*(1), 9.

Yeo, I.-K., & Johnson, R. A. (2000). A new family of power transformations to improve normality or symmetry. *Biometrika*, *87*(4), 954–959.

Yi, K., Wu, J., Gan, C., Torralba, A., Kohli, P., & Tenenbaum, J. B. (2018). Neural-symbolic vqa: Disentangling reasoning from vision and language understanding. In

Proceedings of the 32Nd International Conference on Neural Information Processing Systems, NIPS'18, pp. 1039–1050, USA. Curran Associates Inc.

JAIR Experiments/Results

1. Experiments

1.1 Improvements in Accuracy

Figure 1: [Main Paper] The improvements, $\delta F1$, averaged over three runs, for different combinations of the models and oracles: $\{LPM, DT\} \times \{GBM, RF\}$. The x-axes show the normalized model sizes.

1.2 Comparison to Uncertainty Sampling

OTHER INFO: In Table 2, the oracle based approach is better 70.74% of the cases, and $RMSE = 25.88$ in these cases. Uncertainty based sampling is better in 29.26% of the cases, with $RMSE = 19.00$.

OTHER INFO: In Table 3, the oracle based approach is better 58.92% of the cases, and $RMSE = 13.77$ in these cases. Uncertainty based sampling is better in 41.08% of the cases, with $RMSE = 2.05$.

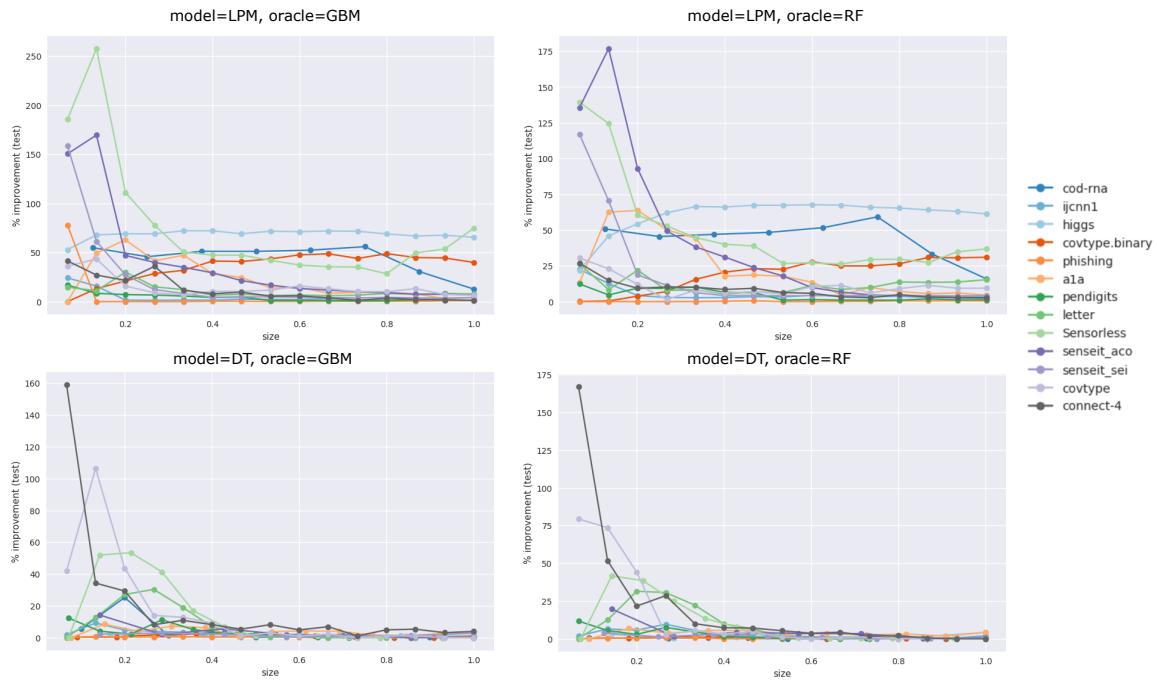Table 1: [Main Paper] This table shows the average improvements, $\delta F1$, over three runs for different combinations of models and oracles: $\{LPM, DT\} \times \{GBM, RF\}$. The best improvement for a model size and oracle is indicated in bold

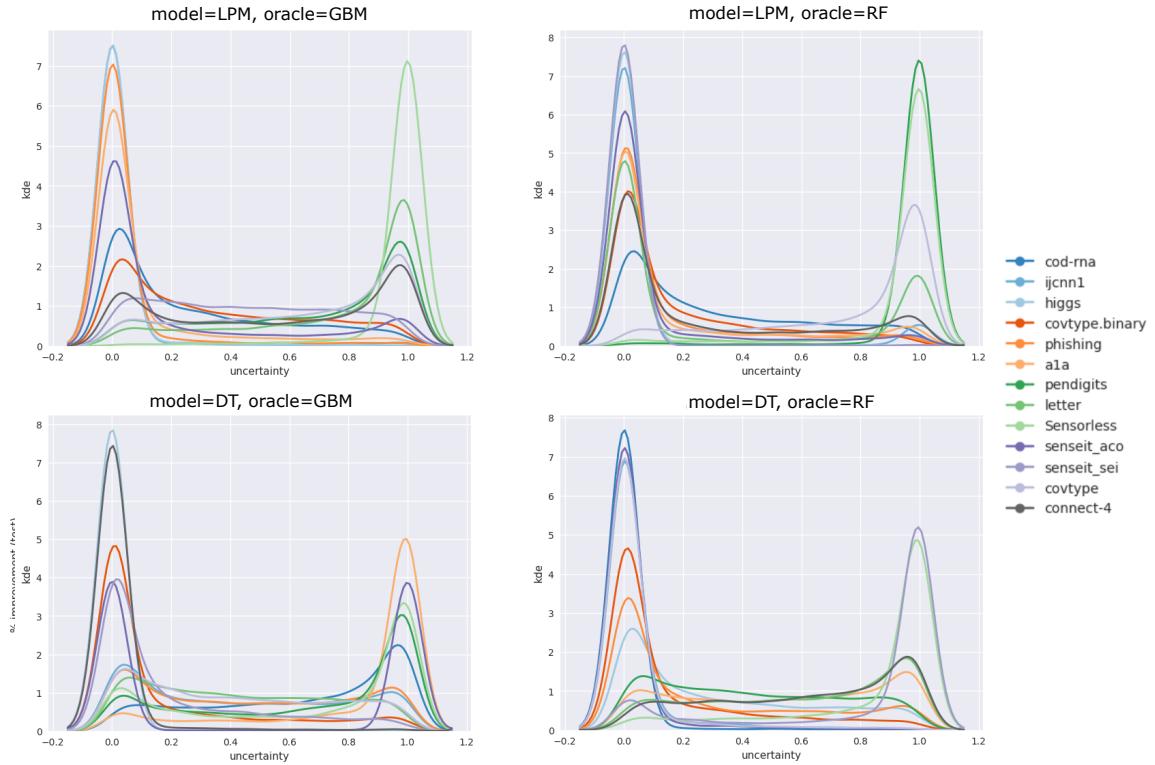| dataset	model_ora	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
cod-rna	lpm_gbm	**54.93**	**45.74**	**51.25**	**51.15**	**52.42**	55.98	30.62	12.78	-	-	-	-	-	-	-
	lpm_rf	50.81	45.51	47.01	48.39	51.67	**59.25**	**33.29**	**15.90**	-	-	-	-	-	-	-
	dt_gbm	**5.51**	25.25	**1.43**	**3.16**	0.19	**0.64**	**1.29**	**0.67**	**0.35**	**0.66**	-	-	-	-	-
	dt_rf	0.59	-	0.28	0.87	**0.66**	0.00	0.00	0.24	0.20	0.00	0.00	-	-	-	-
ijcnn1	lpm_gbm	**24.20**	**15.85**	1.55	1.37	1.63	1.46	2.41	2.21	2.12	3.06	1.95	1.58	0.46	1.06	1.17
	lpm_rf	22.61	12.61	**4.23**	**2.91**	**2.80**	**2.87**	**3.52**	**3.26**	**4.45**	**3.76**	**3.40**	**3.76**	**2.84**	**2.94**	**3.76**
	dt_gbm	1.84	**9.24**	4.79	3.92	3.41	0.65	0.30	1.53	**2.12**	0.95	0.63	0.67	**0.00**	**0.62**	0.87
	dt_rf	**1.93**	6.49	**5.74**	**9.63**	**5.58**	**1.93**	**3.41**	**2.11**	1.36	**1.19**	**1.34**	**1.14**	0.00	0.56	**2.00**
higgs	lpm_gbm	**52.88**	**67.99**	**69.08**	**69.09**	**72.21**	**72.17**	**69.14**	**71.75**	**71.25**	**71.89**	**71.70**	**68.93**	**66.88**	**67.61**	**65.53**
	lpm_rf	21.97	45.98	54.37	62.07	66.47	66.15	67.33	67.35	67.72	67.42	65.99	65.37	64.12	63.13	61.32
	dt_gbm	2.10	**0.55**	1.02	**1.93**	**1.42**	1.44	-	-	-	-	-	-	-	-	-
	dt_rf	**2.91**	0.36	**1.16**	1.40	1.31	**1.46**	4.17	-	-	-	-	-	-	-	-
covtype.binary	lpm_gbm	0.00	**13.24**	**21.01**	**28.60**	**31.91**	**41.35**	**40.89**	**43.51**	**47.67**	**48.72**	**43.96**	**48.86**	**44.89**	**44.49**	**39.89**
	lpm_rf	**0.16**	0.48	3.85	7.07	15.64	20.64	23.04	22.59	27.84	24.97	24.94	26.46	30.84	30.59	31.06
	dt_gbm	0.32	**0.42**	**1.85**	**2.50**	**2.00**	**2.04**	**1.00**	0.73	0.05	0.00	**0.83**	-	-	-	-
	dt_rf	**0.44**	0.41	1.36	1.03	1.24	0.92	0.11	**0.95**	**0.33**	**0.40**	0.42	-	-	-	-
phishing	lpm_gbm	**78.02**	0.00	**0.09**	0.00	0.05	**0.44**	0.40	**0.21**	**0.22**	**0.45**	**0.44**	0.43	0.52	**0.98**	**1.19**
	lpm_rf	0.00	**0.00**	0.00	**0.00**	**0.06**	0.29	**0.61**	0.00	0.05	0.23	0.22	**0.63**	**0.53**	0.69	0.53
	dt_gbm	**0.00**	**0.77**	**0.44**	0.43	**0.59**	**0.35**	**0.46**	**0.76**	**0.25**	0.03	**0.13**	**0.19**	**0.00**	**0.00**	**0.00**
	dt_rf	0.00	0.33	0.33	**0.56**	0.57	0.00	0.04	0.01	**0.05**	0.00	0.00	0.00	0.00	0.00	0.00
a1a	lpm_gbm	0.00	49.32	62.93	41.54	**47.20**	**28.58**	**24.54**	13.97	**14.32**	8.66	9.27	**9.41**	**7.92**	3.07	4.16
	lpm_rf	**13.31**	**62.55**	**63.72**	**50.26**	44.29	17.75	18.68	**17.71**	13.56	4.74	**9.87**	7.07	5.60	**6.09**	**4.54**
	dt_gbm	0.00	**8.44**	**3.89**	**7.24**	**6.21**	0.67	**3.83**	**4.24**	**3.97**	1.56	1.18	2.48	3.31	-	-
	dt_rf	**0.00**	6.88	3.75	5.50	5.61	**3.09**	3.47	2.57	3.14	**1.80**	**4.16**	-	-	-	-
pendigits	lpm_gbm	**16.91**	8.45	7.19	6.72	5.68	4.25	4.66	**1.17**	1.35	**1.15**	0.78	**1.19**	1.84	**1.89**	**1.44**
	lpm_rf	12.35	4.71	**9.42**	**9.20**	**10.14**	**6.49**	**6.26**	1.07	**1.41**	1.12	**1.07**	1.05	**1.94**	1.29	1.34
	dt_gbm	**12.31**	4.08	2.22	**11.20**	**5.11**	**2.76**	**2.32**	**0.78**	0.13	**0.08**	**0.00**	**0.00**	**0.00**	**0.00**	-
	dt_rf	11.71	**5.48**	**2.91**	7.55	4.52	1.64	0.89	0.23	**0.26**	0.00	0.00	0.00	0.00	-	0.00
letter	lpm_gbm	14.71	**11.08**	**30.11**	**15.11**	**12.19**	**6.58**	**7.63**	6.27	6.70	6.06	6.57	8.35	7.47	8.33	7.84
	lpm_rf	**26.27**	8.45	21.94	7.85	8.26	5.30	6.84	**6.39**	**11.23**	**8.54**	**9.84**	**13.73**	**13.48**	**13.71**	**15.34**
	dt_gbm	0.11	**12.63**	27.14	30.32	19.00	6.88	2.14	0.36	0.91	0.00	0.00	0.00	0.00	0.00	0.00
	dt_rf	0.10	12.50	**31.51**	**30.62**	**22.31**	**9.83**	**6.56**	**1.30**	**1.40**	**0.19**	0.00	0.00	0.00	0.00	0.00
Sensorless	lpm_gbm	**185.48**	**257.69**	**110.99**	**77.89**	**50.55**	**47.38**	**47.31**	**42.12**	**37.25**	**35.47**	**35.25**	28.42	**49.69**	**53.59**	**74.85**
	lpm_rf	139.45	124.40	60.32	52.83	44.74	40.04	38.94	26.72	27.08	26.24	29.43	**29.71**	27.40	34.88	36.84
	dt_gbm	0.00	**51.90**	**53.44**	**41.26**	**16.97**	6.79	0.94	**0.89**	**0.88**	0.20	0.00	0.25	0.03	**0.04**	-
	dt_rf	0.00	41.77	38.49	24.97	13.44	**7.76**	**1.54**	0.67	0.36	0.19	**0.29**	**0.27**	**0.15**	0.00	-
senseit_aco	lpm_gbm	**150.48**	169.61	47.04	39.94	34.99	29.36	21.38	16.80	**13.40**	**11.70**	**10.14**	**9.13**	**7.55**	**7.51**	**6.70**
	lpm_rf	135.54	**176.84**	**92.93**	**49.36**	**38.21**	**31.15**	**23.77**	**17.94**	9.78	6.74	4.75	4.17	3.74	2.94	2.57
	dt_gbm	14.18	**2.01**	**5.53**	1.79	1.62	0.00	0.00	-	-	-	-	-	-	-	-
	dt_rf	**19.73**	1.44	4.24	**3.56**	**3.42**	**0.12**	**0.83**	-	-	-	-	-	-	-	-
senseit_sei	lpm_gbm	**158.68**	61.33	**27.15**	**12.30**	**8.26**	**4.38**	**5.03**	**4.79**	**4.44**	4.15	3.77	**4.35**	3.88	3.64	4.18
	lpm_rf	116.95	**70.54**	18.65	11.39	6.27	4.23	4.36	4.09	**4.21**	**4.21**	**4.47**	4.20	**4.26**	**4.05**	**4.22**
	dt_gbm	2.22	**2.90**	2.20	0.43	0.85	**1.29**	**2.88**	-	-	-	-	-	-	-	-
	dt_rf	**3.93**	1.32	**2.87**	**1.36**	**1.10**	0.00	0.36	0.43	-	-	-	-	-	-	-
covtype	lpm_gbm	36.12	**43.42**	15.86	9.01	6.88	10.45	10.41	11.73	**16.08**	**13.08**	**10.31**	**10.13**	**13.23**	7.29	6.45
	lpm_rf	30.47	22.96	12.04	1.45	**8.08**	5.93	7.03	5.34	10.60	11.45	6.07	9.11	11.39	**9.37**	**9.37**
	dt_gbm	41.85	**106.35**	43.61	**14.04**	**12.74**	**9.17**	1.29	**2.91**	**1.08**	**2.14**	0.04	**0.97**	**1.13**	**0.00**	**0.00**
	dt_rf	**79.42**	**73.61**	**44.21**	2.76	5.40	3.14	**4.81**	1.08	0.00	1.17	**1.02**	0.00	0.00	0.00	0.00
connect-4	lpm_gbm	**41.54**	**27.06**	**21.68**	**35.97**	**11.55**	8.03	**9.71**	5.59	**5.96**	**3.84**	1.26	3.57	2.23	1.98	1.21
	lpm_rf	26.85	14.86	9.37	10.25	10.10	**8.53**	**9.33**	**6.34**	5.83	3.34	**2.71**	**4.94**	**3.25**	**2.68**	**2.71**
	dt_gbm	159.08	34.29	29.31	8.11	**10.87**	**8.18**	5.34	**8.14**	**4.88**	**6.88**	1.09	**4.90**	**5.33**	**3.13**	**3.92**
	dt_rf	**167.19**	**51.76**	21.68	**28.58**	10.01	7.31	**7.12**	5.32	3.65	4.37	**1.72**	2.13	0.39	0.00	0.00

Figure 2: [Main Paper] KDE plots of the IBMM density learned on uncertainty scale $[0, 1]$. Shown for different combinations of the models and oracles: $\{LPM, DT\} \times \{GBM, RF\}$.

Table 2: [Main Paper] This table compares the average improvement, $\delta F1$, from our oracle based approach with model=LPM to uncertainty sampling. The reported values are averaged over three runs. Please see text for explanation of color scheme and analysis.

dataset	model_ora	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
cod-rna	ora_gbm	54.93	45.74	51.25	51.15	52.42	55.98	30.62	12.78	-	-	-	-	-	-	-
	unc_gbm	60.16	42.45	41.35	41.34	40.59	61.06	55.08	7.89	-	-	-	-	-	-	-
	ora_rf	50.81	45.51	47.01	48.39	51.67	59.25	33.29	15.90	-	-	-	-	-	-	-
	unc_rf	61.47	46.00	44.99	44.04	41.42	52.40	56.95	8.27	-	-	-	-	-	-	-
ijcnn1	ora_gbm	24.20	15.85	1.55	1.37	1.63	1.46	2.41	2.21	2.12	3.06	1.95	1.58	0.46	1.06	1.17
	unc_gbm	0.00	2.68	1.65	1.11	1.45	3.42	0.51	0.74	0.82	4.39	2.54	2.05	2.13	1.74	3.93
	ora_rf	22.61	12.61	4.23	2.91	2.80	2.87	3.52	3.26	4.45	3.76	3.40	3.76	2.84	2.94	3.76
	unc_rf	0.00	0.27	1.13	2.31	0.93	0.36	1.28	0.03	1.16	2.59	1.56	0.43	0.00	2.65	0.66
higgs	ora_gbm	52.88	67.99	69.08	69.09	72.21	72.17	69.14	71.75	71.25	71.89	71.70	68.93	66.88	67.61	65.53
	unc_gbm	0.00	0.00	0.00	0.00	0.00	0.17	0.28	0.60	0.59	0.54	0.09	0.44	1.14	0.15	0.42
	ora_rf	21.97	45.98	54.37	62.07	66.47	66.15	67.33	67.35	67.72	67.42	65.99	65.37	64.12	63.13	61.32
	unc_rf	0.07	0.00	0.86	2.09	2.03	2.02	1.32	2.18	2.28	2.12	2.74	2.45	2.34	2.32	2.68
covtype.binary	ora_gbm	0.00	13.24	21.01	28.60	31.91	41.35	40.89	43.51	47.67	48.72	43.96	48.86	44.89	44.49	39.89
	unc_gbm	0.00	0.00	0.58	3.63	6.31	9.38	11.51	10.38	9.21	9.90	13.74	17.88	16.70	16.49	15.60
	ora_rf	0.16	0.48	3.85	7.07	15.64	20.64	23.04	22.59	27.84	24.97	24.94	26.46	30.84	30.59	31.06
	unc_rf	0.00	0.00	0.65	3.88	6.48	8.56	11.11	13.92	12.98	7.69	9.19	13.46	14.05	12.14	9.88
phishing	ora_gbm	78.02	0.00	0.09	0.00	0.05	0.44	0.40	0.21	0.22	0.45	0.44	0.43	0.52	0.98	1.19
	unc_gbm	125.89	0.00	0.00	0.00	0.04	0.26	0.95	0.65	0.00	0.26	0.03	0.20	0.63	0.81	0.73
	ora_rf	0.00	0.00	0.00	0.00	0.06	0.29	0.61	0.00	0.05	0.23	0.22	0.63	0.53	0.69	0.53
	unc_rf	64.74	0.00	0.00	0.00	0.00	0.00	0.00	0.65	0.12	0.18	0.58	0.78	1.15	1.30	1.34
a1a	ora_gbm	0.00	49.32	62.93	41.54	47.20	28.58	24.54	13.97	14.32	8.66	9.27	9.41	7.92	3.07	4.16
	unc_gbm	0.00	0.00	21.07	23.57	4.79	3.91	8.22	0.00	1.32	1.03	1.70	3.72	1.59	0.88	2.53
	ora_rf	13.31	62.55	63.72	50.26	44.29	17.75	18.68	17.71	13.56	4.74	9.87	7.07	5.60	6.09	4.54
	unc_rf	0.00	0.00	20.78	23.57	29.18	5.53	12.57	1.58	1.57	0.75	0.66	2.91	2.61	3.49	0.86
pendigits	ora_gbm	16.91	8.45	7.19	6.72	5.68	4.25	4.66	1.17	1.35	1.15	0.78	1.19	1.84	1.89	1.44
	unc_gbm	39.51	11.46	6.05	0.00	0.00	0.00	0.06	0.00	0.00	0.23	0.13	0.41	1.40	0.97	0.83
	ora_rf	12.35	4.71	9.42	9.20	10.14	6.49	6.26	1.07	1.41	1.12	1.07	1.05	1.94	1.29	1.34
	unc_rf	1.60	1.44	0.24	0.80	0.19	1.40	1.76	0.16	0.38	0.18	0.17	0.22	0.92	1.14	0.96
letter	ora_gbm	14.71	11.08	30.11	15.11	12.19	6.58	7.63	6.27	6.70	6.06	6.57	8.35	7.47	8.33	7.84
	unc_gbm	4.57	2.74	0.00	2.03	0.07	0.00	1.48	0.17	0.60	1.61	0.32	0.35	3.06	2.75	2.10
	ora_rf	26.27	8.45	21.94	7.85	8.26	5.30	6.84	6.39	11.23	8.54	9.84	13.73	13.48	13.71	15.34
	unc_rf	0.00	4.00	1.92	3.06	0.58	0.00	0.40	0.78	0.00	0.24	0.00	0.04	2.18	2.95	1.35
Sensorless	ora_gbm	185.48	257.69	110.99	77.89	50.55	47.38	47.31	42.12	37.25	35.47	35.25	28.42	49.69	53.59	74.85
	unc_gbm	175.28	257.12	166.11	77.70	67.31	55.86	55.31	41.40	37.74	28.65	24.73	21.51	28.35	19.73	21.18
	ora_rf	139.45	124.40	60.32	52.83	44.74	40.04	38.94	26.72	27.08	26.24	29.43	29.71	27.40	34.88	36.84
	unc_rf	164.03	236.19	148.01	67.17	47.82	41.31	39.55	32.86	28.99	32.73	28.44	34.84	46.07	51.61	55.86
senseit_aco	ora_gbm	150.48	169.61	47.04	39.94	34.99	29.36	21.38	16.80	13.40	11.70	10.14	9.13	7.55	7.51	6.70
	unc_gbm	64.82	156.09	50.18	40.51	24.90	19.36	14.61	12.87	10.43	9.76	6.35	6.03	5.54	4.94	4.61
	ora_rf	135.54	176.84	92.93	49.36	38.21	31.15	23.77	17.94	9.78	6.74	4.75	4.17	3.74	2.94	2.57
	unc_rf	103.74	152.13	51.15	38.02	29.18	22.51	13.85	11.52	9.47	7.69	4.82	4.61	3.67	3.56	3.40
senseit_sei	ora_gbm	158.68	61.33	27.15	12.30	8.26	4.38	5.03	4.79	4.44	4.15	3.77	4.35	3.88	3.64	4.18
	unc_gbm	151.82	92.94	29.87	11.26	5.91	5.98	5.41	5.34	4.78	4.95	5.63	5.74	5.82	5.77	5.66
	ora_rf	116.95	70.54	18.65	11.39	6.27	4.23	4.36	4.09	4.21	4.21	4.47	4.20	4.26	4.05	4.22
	unc_rf	151.45	84.09	34.24	15.58	9.41	7.00	6.62	6.09	6.38	5.99	6.08	6.49	6.08	6.15	6.19
covtype	ora_gbm	36.12	43.42	15.86	9.01	6.88	10.45	10.41	11.73	16.08	13.08	10.31	10.13	13.23	7.29	6.45
	unc_gbm	43.56	27.68	1.85	6.82	6.70	1.63	3.27	1.12	3.43	2.36	1.44	2.18	3.27	4.27	3.39
	ora_rf	30.47	22.96	12.04	1.45	8.08	5.93	7.03	5.34	10.60	11.45	6.07	9.11	11.39	9.37	9.37
	unc_rf	24.46	27.38	9.32	7.74	6.23	3.60	4.27	0.52	0.83	2.24	3.63	5.86	3.21	2.76	5.80
connect-4	ora_gbm	41.54	27.06	21.68	35.97	11.55	8.03	9.71	5.59	5.96	3.84	1.26	3.57	2.23	1.98	1.21
	unc_gbm	53.26	62.53	17.20	27.18	17.02	17.43	15.63	8.16	5.74	2.34	7.38	5.54	6.27	3.71	4.67
	ora_rf	26.85	14.86	9.37	10.25	10.10	8.53	9.33	6.34	5.83	3.34	2.71	4.94	3.25	2.68	2.71
	unc_rf	23.92	19.55	33.23	32.87	13.57	21.13	5.16	2.71	2.08	6.30	1.97	5.78	4.71	3.83	10.12

Table 3: [Appendix] This table compares the average improvement, $\delta F1$, from our oracle based approach with model=DT to uncertainty sampling. The reported values are averaged over three runs. Please see text for explanation of color scheme and analysis.

dataset	model_ora	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
cod-rna	ora_gbm	5.51	25.25	1.43	3.16	0.19	0.64	1.29	0.67	0.35	0.66	-	-	-	-	-
	unc_gbm	0.14	0.14	0.17	1.66	0.67	0.85	0.72	0.95	0.83	0.20	0.23	0.69	-	-	-
	ora_rf	0.59	-	0.28	0.87	0.66	0.00	0.00	0.24	0.20	0.00	0.00	-	-	-	-
	unc_rf	0.00	1.59	0.48	1.32	0.20	0.58	1.06	0.47	0.00	0.00	0.00	-	-	-	-
ijcnn1	ora_gbm	1.84	9.24	4.79	3.92	3.41	0.65	0.30	1.53	2.12	0.95	0.63	0.67	0.00	0.62	0.87
	unc_gbm	2.55	7.88	5.76	5.41	2.51	2.64	1.30	1.00	1.20	1.20	1.11	1.23	1.10	1.45	1.66
	ora_rf	1.93	6.49	5.74	9.63	5.58	1.93	3.41	2.11	1.36	1.19	1.34	1.14	0.00	0.56	2.00
	unc_rf	2.65	11.07	7.87	8.99	8.72	8.94	3.57	2.08	1.55	0.75	0.45	1.09	1.86	2.02	1.78
higgs	ora_gbm	2.10	0.55	1.02	1.93	1.42	1.44	-	-	-	-	-	-	-	-	-
	unc_gbm	0.81	0.03	1.77	0.97	0.43	0.78	2.98	-	-	-	-	-	-	-	-
	ora_rf	2.91	0.36	1.16	1.40	1.31	1.46	4.17	-	-	-	-	-	-	-	-
	unc_rf	0.00	0.00	0.00	0.10	0.00	0.16	-	-	-	-	-	-	-	-	-
covtype.binary	ora_gbm	0.32	0.42	1.85	2.50	2.00	2.04	1.00	0.73	0.05	0.00	0.83	-	-	-	-
	unc_gbm	1.42	1.25	1.57	1.33	2.19	1.03	1.23	0.74	0.53	0.74	0.00	1.12	0.95	1.09	0.84
	ora_rf	0.44	0.41	1.36	1.03	1.24	0.92	0.11	0.95	0.33	0.40	0.42	-	-	-	-
	unc_rf	0.00	0.17	2.12	1.51	0.32	1.17	1.39	0.32	0.73	0.00	0.00	0.26	0.00	1.04	-
phishing	ora_gbm	0.00	0.77	0.44	0.43	0.59	0.35	0.46	0.76	0.25	0.03	0.13	0.19	0.00	0.00	0.00
	unc_gbm	0.00	1.17	0.00	0.62	0.08	0.00	0.27	0.30	0.00	0.14	0.02	0.01	0.19	0.08	0.00
	ora_rf	0.00	0.33	0.33	0.56	0.57	0.00	0.04	0.01	0.18	0.05	0.00	0.00	0.00	0.00	0.00
	unc_rf	0.00	0.64	0.00	0.09	0.71	0.10	1.12	0.36	0.79	0.67	0.45	0.74	0.20	0.60	0.76
a1a	ora_gbm	0.00	8.44	3.89	7.24	6.21	0.67	3.83	4.24	3.97	1.56	1.18	2.48	3.31	-	-
	unc_gbm	0.00	7.28	4.65	6.38	6.04	5.66	4.35	3.87	3.00	3.29	4.05	2.06	0.00	0.50	0.38
	ora_rf	0.00	6.88	3.75	5.50	5.61	3.09	3.47	2.57	3.14	1.80	4.16	-	-	-	-
	unc_rf	0.00	8.03	3.76	2.73	4.74	3.74	4.30	4.70	3.53	2.77	-	-	-	-	-
pendigits	ora_gbm	12.31	4.08	2.22	11.20	5.11	2.76	2.32	0.78	0.13	0.08	0.00	0.00	0.00	0.00	-
	unc_gbm	0.08	3.55	1.46	6.72	3.81	2.32	1.64	0.83	0.00	0.05	0.00	0.00	0.00	0.18	0.00
	ora_rf	11.71	5.48	2.91	7.55	4.52	1.64	0.89	0.23	0.26	0.00	0.00	0.00	0.00	-	0.00
	unc_rf	11.22	8.29	0.00	5.72	4.28	2.44	1.37	0.40	0.06	0.18	0.00	0.00	0.05	0.15	-
letter	ora_gbm	0.11	12.63	27.14	30.32	19.00	6.88	2.14	0.36	0.91	0.00	0.00	0.00	0.00	0.00	0.00
	unc_gbm	3.38	5.91	14.52	26.95	17.11	6.06	2.04	0.77	1.18	0.56	0.31	0.35	0.00	0.00	0.00
	ora_rf	0.10	12.50	31.51	30.62	22.31	9.83	6.56	1.30	1.40	0.19	0.00	0.00	0.00	0.00	0.00
	unc_rf	0.05	7.10	33.39	31.57	10.88	1.67	3.39	0.41	0.37	0.00	0.00	0.00	0.92	0.00	0.00
Sensorless	ora_gbm	0.00	51.90	53.44	41.26	16.97	6.79	0.94	0.89	0.88	0.20	0.00	0.25	0.03	0.04	-
	unc_gbm	0.05	24.06	13.95	19.03	13.20	5.18	1.42	1.11	0.21	0.09	0.11	0.00	0.07	0.04	0.12
	ora_rf	0.00	41.77	38.49	24.97	13.44	7.76	1.54	0.67	0.36	0.19	0.29	0.27	0.15	0.00	-
	unc_rf	0.00	26.39	12.99	23.83	5.34	2.43	0.00	0.08	0.37	0.00	0.00	0.04	0.00	0.00	0.00
senseit_aco	ora_gbm	14.18	2.01	5.53	1.79	1.62	0.00	0.00	-	-	-	-	-	-	-	-
	unc_gbm	0.13	0.54	2.17	3.05	0.73	1.11	-	-	-	-	-	-	-	-	-
	ora_rf	19.73	1.44	4.24	3.56	3.42	0.12	0.83	-	-	-	-	-	-	-	-
	unc_rf	0.36	1.75	2.30	1.93	1.83	1.11	0.00	0.00	0.67	-	-	-	-	-	-
senseit_sei	ora_gbm	2.22	2.90	2.20	0.43	0.85	1.29	2.88	-	-	-	-	-	-	-	-
	unc_gbm	1.53	0.63	0.48	1.18	2.17	0.00	0.26	0.00	0.13	0.00	-	-	-	-	-
	ora_rf	3.93	1.32	2.87	1.36	1.10	0.00	0.36	0.43	-	-	-	-	-	-	-
	unc_rf	2.66	0.46	0.27	1.42	0.21	1.42	2.24	2.51	-	-	-	-	-	-	-
covtype	ora_gbm	41.85	106.35	43.61	14.04	12.74	9.17	1.29	2.91	1.08	2.14	0.04	0.97	1.13	0.00	0.00
	unc_gbm	27.63	90.44	10.37	0.47	3.69	2.12	5.87	4.66	2.47	0.07	2.07	0.41	1.69	0.89	2.94
	ora_rf	79.42	73.61	44.21	2.76	5.40	3.14	4.81	1.08	0.00	1.17	1.02	0.00	0.00	0.00	0.00
	unc_rf	22.81	58.20	9.93	0.49	0.00	3.01	0.00	0.00	0.19	0.00	2.91	1.04	1.15	0.00	1.90
connect-4	ora_gbm	159.08	34.29	29.31	8.11	10.87	8.18	5.34	8.14	4.88	6.88	1.09	4.90	5.33	3.13	3.92
	unc_gbm	32.47	5.20	2.90	5.26	9.88	6.08	5.30	4.32	4.99	4.05	2.80	0.08	0.20	1.00	0.36
	ora_rf	167.19	51.76	21.68	28.58	10.01	7.31	7.12	5.32	3.65	4.37	1.72	2.13	0.39	0.00	0.00
	unc_rf	114.13	21.24	5.66	14.35	2.90	7.65	10.79	10.49	9.74	10.37	6.88	6.73	7.61	6.93	5.10

1.3 Comparison to Density Trees

Table 4: [Main Paper] This table compares the average improvement, $\delta F1$, from our oracle based approach with model=LPM to the density tree based approach. The reported values are averaged over three runs. Please see text for explanation of color scheme and analysis.

dataset	model_ora	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
cod-rna	lpm_gbm	54.93	45.74	51.25	51.15	52.42	55.98	30.62	12.78	-	-	-	-	-	-	-
	lpm_rf	50.81	45.51	47.01	48.39	51.67	59.25	33.29	15.90	-	-	-	-	-	-	-
	lpm_dentr	26.87	28.25	53.71	57.71	58.08	35.82	20.05	4.36	-	-	-	-	-	-	-
ijcnn1	lpm_gbm	24.20	15.85	1.55	1.37	1.63	1.46	2.41	2.21	2.12	3.06	1.95	1.58	0.46	1.06	1.17
	lpm_rf	22.61	12.61	4.23	2.91	2.80	2.87	3.52	3.26	4.45	3.76	3.40	3.76	2.84	2.94	3.76
	lpm_dentr	17.25	8.86	0.51	1.43	1.86	1.35	0.94	1.79	2.04	1.31	0.61	1.95	2.01	0.37	2.48
higgs	lpm_gbm	52.88	67.99	69.08	69.09	72.21	72.17	69.14	71.75	71.25	71.89	71.70	68.93	66.88	67.61	65.53
	lpm_rf	21.97	45.98	54.37	62.07	66.47	66.15	67.33	67.35	67.72	67.42	65.99	65.37	64.12	63.13	61.32
	lpm_dentr	0.00	0.04	0.97	1.75	2.53	3.42	3.36	2.99	3.48	4.90	6.12	4.99	4.69	4.33	4.61
covtype.binary	lpm_gbm	0.00	13.24	21.01	28.60	31.91	41.35	40.89	43.51	47.67	48.72	43.96	48.86	44.89	44.49	39.89
	lpm_rf	0.16	0.48	3.85	7.07	15.64	20.64	23.04	22.59	27.84	24.97	24.94	26.46	30.84	30.59	31.06
	lpm_dentr	0.13	2.22	4.24	6.73	9.40	12.29	14.67	7.66	9.70	10.06	11.60	12.43	10.73	11.44	9.08
phishing	lpm_gbm	78.02	0.00	0.09	0.00	0.05	0.44	0.40	0.21	0.22	0.45	0.44	0.43	0.52	0.98	1.19
	lpm_rf	0.00	0.00	0.00	0.00	0.06	0.29	0.61	0.00	0.05	0.23	0.22	0.63	0.53	0.69	0.53
	lpm_dentr	0.00	0.00	0.00	0.00	0.11	0.13	0.60	0.00	0.02	0.27	0.18	0.39	0.41	0.58	0.60
a1a	lpm_gbm	0.00	49.32	62.93	41.54	47.20	28.58	24.54	13.97	14.32	8.66	9.27	9.41	7.92	3.07	4.16
	lpm_rf	13.31	62.55	63.72	50.26	44.29	17.75	18.68	17.71	13.56	4.74	9.87	7.07	5.60	6.09	4.54
	lpm_dentr	0.00	20.62	41.00	29.02	22.53	12.10	9.03	13.28	10.08	4.33	4.24	3.14	2.55	0.61	1.96
pendigits	lpm_gbm	16.91	8.45	7.19	6.72	5.68	4.25	4.66	1.17	1.35	1.15	0.78	1.19	1.84	1.89	1.44
	lpm_rf	12.35	4.71	9.42	9.20	10.14	6.49	6.26	1.07	1.41	1.12	1.07	1.05	1.94	1.29	1.34
	lpm_dentr	10.08	8.54	10.09	6.14	8.36	3.83	4.80	0.67	1.06	0.48	0.46	0.46	1.14	0.37	0.37
letter	lpm_gbm	14.71	11.08	30.11	15.11	12.19	6.58	7.63	6.27	6.70	6.06	6.57	8.35	7.47	8.33	7.84
	lpm_rf	26.27	8.45	21.94	7.85	8.26	5.30	6.84	6.39	11.23	8.54	9.84	13.73	13.48	13.71	15.34
	lpm_dentr	12.36	8.90	22.35	12.27	9.33	2.87	0.94	1.42	1.58	3.05	1.60	1.57	3.97	6.70	5.57
Sensorless	lpm_gbm	185.48	257.69	110.99	77.89	50.55	47.38	47.31	42.12	37.25	35.47	35.25	28.42	49.69	53.59	74.85
	lpm_rf	139.45	124.40	60.32	52.83	44.74	40.04	38.94	26.72	27.08	26.24	29.43	29.71	27.40	34.88	36.84
	lpm_dentr	71.01	57.69	31.92	15.78	17.17	18.79	22.40	31.14	28.49	31.70	29.46	28.32	36.58	39.47	32.59
senseit_aco	lpm_gbm	150.48	169.61	47.04	39.94	34.99	29.36	21.38	16.80	13.40	11.70	10.14	9.13	7.55	7.51	6.70
	lpm_rf	135.54	176.84	92.93	49.36	38.21	31.15	23.77	17.94	9.78	6.74	4.75	4.17	3.74	2.94	2.57
	lpm_dentr	7.07	56.04	42.17	21.55	15.74	13.46	12.79	7.71	4.34	4.83	2.84	2.96	2.56	2.03	1.87
senseit_sei	lpm_gbm	158.68	61.33	27.15	12.30	8.26	4.38	5.03	4.79	4.44	4.15	3.77	4.35	3.88	3.64	4.18
	lpm_rf	116.95	70.54	18.65	11.39	6.27	4.23	4.36	4.09	4.21	4.21	4.47	4.20	4.26	4.05	4.22
	lpm_dentr	143.97	46.19	20.28	7.56	2.71	1.01	1.04	0.87	0.79	1.47	1.60	1.30	0.27	0.42
covtype	lpm_gbm	36.12	43.42	15.86	9.01	6.88	10.45	10.41	11.73	16.08	13.08	10.31	10.13	13.23	7.29	6.45
	lpm_rf	30.47	22.96	12.04	1.45	8.08	5.93	7.03	5.34	10.60	11.45	6.07	9.11	11.39	9.37	9.37
	lpm_dentr	30.10	20.05	4.68	3.24	1.24	6.46	2.57	4.66	5.43	5.03	6.92	6.02	2.29	9.65	9.03
connect-4	lpm_gbm	41.54	27.06	21.68	35.97	11.55	8.03	9.71	5.59	5.96	3.84	1.26	3.57	2.23	1.98	1.21
	lpm_rf	26.85	14.86	9.37	10.25	10.10	8.53	9.33	6.34	5.83	3.34	2.71	4.94	3.25	2.68	2.71
	lpm_dentr	117.62	32.09	20.29	17.47	7.07	6.41	6.15	5.67	6.80	4.72	3.87	1.82	1.62	0.12	2.31

OTHER INFO: In Table 4, the oracle based approach is better 93.62% of the cases, and $RMSE = 33.80$ in these cases. The density tree based approach is better 6.38% of the time, and $RMSE = 22.17$ in these cases.

OTHER INFO: In Table 6, the oracle based approach is better 67.97% of the cases, and $RMSE = 6.58$ in these cases. The density tree based approach is better 32.03% of the time, and $RMSE = 8.65$ in these cases.

Table 5: [Appendix] This table compares the average improvement, $\delta F1$, from our oracle based approach with model=DT to the density tree based approach. The reported values are averaged over three runs. Please see text for explanation of color scheme and analysis.

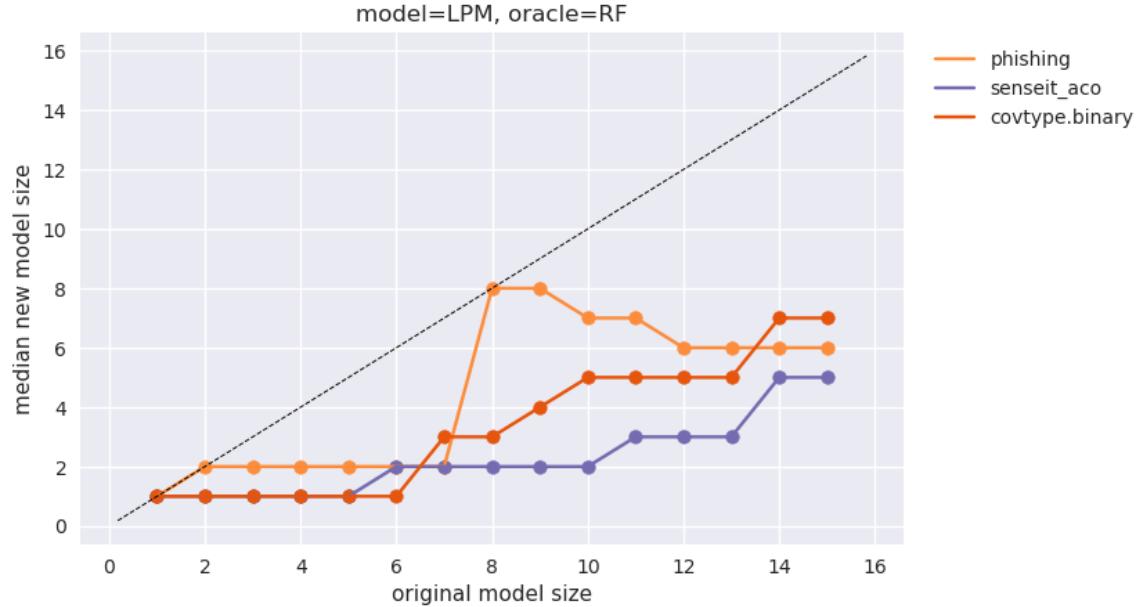| dataset	model_ora	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
cod-rna	dt_gbm	5.51	25.25	1.43	3.16	0.19	0.64	1.29	0.67	0.35	0.66	-	-	-	-	-
	dt_rf	0.59	-	0.28	0.87	0.66	0.00	0.00	0.24	0.20	0.00	0.00	-	-	-	-
	dt_dentr	0.22	0.82	0.36	2.20	0.23	0.17	0.35	0.30	0.28	0.00	-	-	-	-	-
ijcnn1	dt_gbm	1.84	9.24	4.79	3.92	3.41	0.65	0.30	1.53	2.12	0.95	0.63	0.67	0.00	0.62	0.87
	dt_rf	1.93	6.49	5.74	9.63	5.58	1.93	3.41	2.11	1.36	1.19	1.34	1.14	0.00	0.56	2.00
	dt_dentr	3.66	14.80	10.34	11.67	4.61	1.01	2.06	1.85	1.17	0.00	0.35	0.14	0.38	0.00	1.37
higgs	dt_gbm	2.10	0.55	1.02	1.93	1.42	1.44	-	-	-	-	-	-	-	-	-
	dt_rf	2.91	0.36	1.16	1.40	1.31	1.46	4.17	-	-	-	-	-	-	-	-
	dt_dentr	4.32	0.75	0.06	0.22	0.00	0.47	-	-	-	-	-	-	-	-	-
covtype.binary	dt_gbm	0.32	0.42	1.85	2.50	2.00	2.04	1.00	0.73	0.05	0.00	0.83	-	-	-	-
	dt_rf	0.44	0.41	1.36	1.03	1.24	0.92	0.11	0.95	0.33	0.40	0.42	-	-	-	-
	dt_dentr	0.16	0.01	0.60	0.89	0.48	0.58	0.05	0.27	1.27	0.72	0.00	1.64	-	-	-
phishing	dt_gbm	0.00	0.77	0.44	0.43	0.59	0.35	0.46	0.76	0.25	0.03	0.13	0.19	0.00	0.00	0.00
	dt_rf	0.00	0.33	0.33	0.56	0.57	0.00	0.04	0.01	0.18	0.05	0.00	0.00	0.00	0.00	0.00
	dt_dentr	0.00	0.76	0.03	0.09	0.51	0.13	0.19	0.52	0.12	0.13	0.00	0.00	0.00	0.00	0.00
a1a	dt_gbm	0.00	8.44	3.89	7.24	6.21	0.67	3.83	4.24	3.97	1.56	1.18	2.48	3.31	-	-
	dt_rf	0.00	6.88	3.75	5.50	5.61	3.09	3.47	2.57	3.14	1.80	4.16	-	-	-	-
	dt_dentr	0.00	5.04	5.77	5.45	3.60	1.73	2.97	0.82	1.63	-	0.65	0.00	2.35	-	0.92
pendigits	dt_gbm	12.31	4.08	2.22	11.20	5.11	2.76	2.32	0.78	0.13	0.08	0.00	0.00	0.00	0.00	-
	dt_rf	11.71	5.48	2.91	7.55	4.52	1.64	0.89	0.23	0.26	0.00	0.00	0.00	0.00	-	0.00
	dt_dentr	10.98	2.80	4.67	9.45	4.41	2.53	0.92	0.06	0.13	0.00	0.00	0.00	0.00	0.00	-
letter	dt_gbm	0.11	12.63	27.14	30.32	19.00	6.88	2.14	0.36	0.91	0.00	0.00	0.00	0.00	0.00	0.00
	dt_rf	0.10	12.50	31.51	30.62	22.31	9.83	6.56	1.30	1.40	0.19	0.00	0.00	0.00	0.00	0.00
	dt_dentr	0.16	10.68	32.39	44.58	39.66	19.20	9.37	4.75	3.77	1.72	0.35	0.00	0.00	0.00	0.00
Sensorless	dt_gbm	0.00	51.90	53.44	41.26	16.97	6.79	0.94	0.89	0.88	0.20	0.00	0.25	0.03	0.04	-
	dt_rf	0.00	41.77	38.49	24.97	13.44	7.76	1.54	0.67	0.36	0.19	0.29	0.27	0.15	0.00	-
	dt_dentr	0.00	40.93	70.98	82.53	39.72	17.07	7.97	4.77	2.26	1.44	0.88	0.33	0.75	0.84	-
senseit_aco	dt_gbm	14.18	2.01	5.53	1.79	1.62	0.00	0.00	-	-	-	-	-	-	-	-
	dt_rf	19.73	1.44	4.24	3.56	3.42	0.12	0.83	-	-	-	-	-	-	-	-
	dt_dentr	19.77	0.73	2.73	1.86	1.38	0.75	0.77	-	-	-	-	-	-	-	-
senseit_sei	dt_gbm	2.22	2.90	2.20	0.43	0.85	1.29	2.88	-	-	-	-	-	-	-	-
	dt_rf	3.93	1.32	2.87	1.36	1.10	0.00	0.36	0.43	-	-	-	-	-	-	-
	dt_dentr	1.78	0.63	2.08	0.03	1.52	1.32	0.00	-	-	-	-	-	-	-	-
covtype	dt_gbm	41.85	106.35	43.61	14.04	12.74	9.17	1.29	2.91	1.08	2.14	0.04	0.97	1.13	0.00	0.00
	dt_rf	79.42	73.61	44.21	2.76	5.40	3.14	4.81	1.08	0.00	1.17	1.02	0.00	0.00	0.00	0.00
	dt_dentr	34.44	115.25	13.54	9.68	5.38	3.31	2.02	2.00	2.73	3.36	4.30	2.79	0.00	1.14	1.03
connect-4	dt_gbm	159.08	34.29	29.31	8.11	10.87	8.18	5.34	8.14	4.88	6.88	1.09	4.90	5.33	3.13	3.92
	dt_rf	167.19	51.76	21.68	28.58	10.01	7.31	7.12	5.32	3.65	4.37	1.72	2.13	0.39	0.00	0.00
	dt_dentr	180.92	29.47	15.44	11.99	3.99	11.12	2.45	6.14	3.27	1.54	4.48	3.09	2.96	3.00	1.57

2. Compaction

Figure 3: [Main Paper] Alternative way to visualize results. The y-axis shows the new model size that has $F1\_score$ equal to or greater than the score at the original model size. The dashed line is the reference - ideally, no part of the plotted lines should be above it.

Table 6: Comparison of improvements in F1 after fixing the scikit bug.

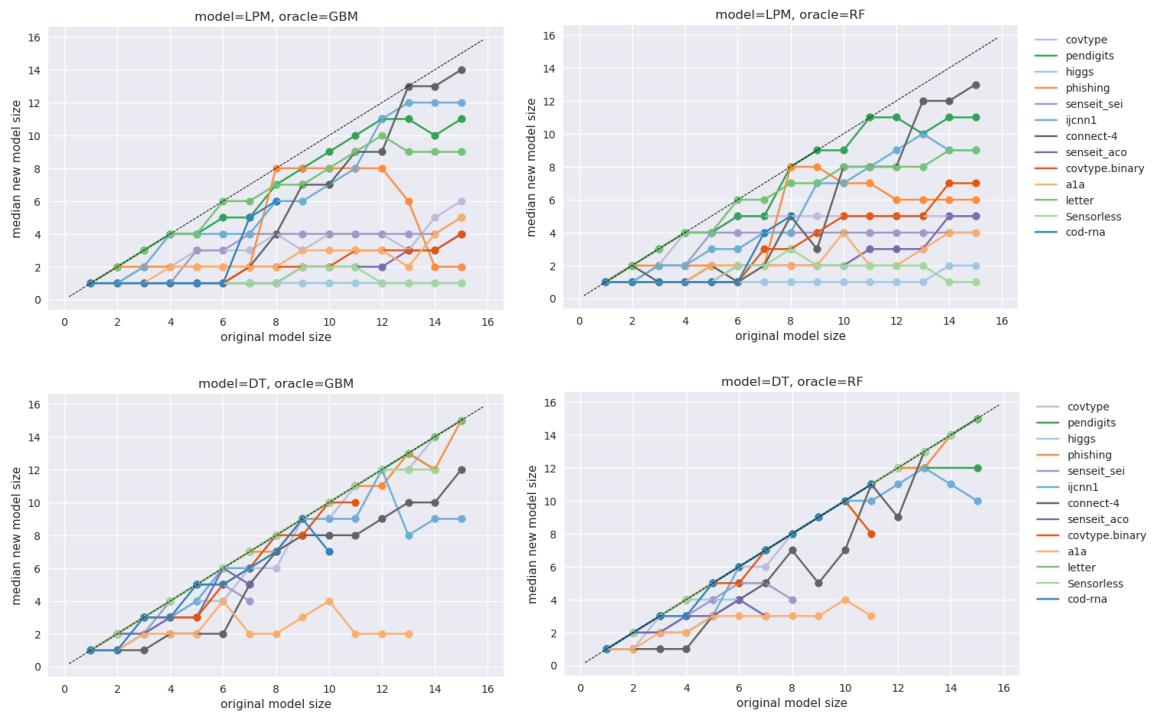| dataset	model_ora	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
cod-rna (before)	lpm_gbm	54.93	45.74	51.25	51.15	52.42	55.98	30.62	12.78	-	-	-	-	-	-	-
cod-rna (after)	lpm_gbm	4.53	13.04	12.32	13.90	13.70	12.81	0.00	0.00	-	-	-	-	-	-	-
Sensorless (before)	lpm_gbm	185.48	257.69	110.99	77.89	50.55	47.38	47.31	42.12	37.25	35.47	35.25	28.42	49.69	53.59	74.85
Sensorless (after)	lpm_gbm	189.73	208.38	142.16	46.49	36.44	29.41	38.52	16.44	24.49	33.39	44.09	49.10	68.33	75.56	61.94

Figure 4: [Appendix] Compaction profiles for different combinations of models and oracles: $\{LPM, DT\} \times \{GBM, RF\}$.