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 here, 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 an *oracle*, a highly accurate probabilistic model on the training data. The uncertainty in the oracle's predictions are used to learn a sampling distribution for the training data. The interpretable model is then trained on a data sample obtained using this distribution, leading often to significantly greater accuracy.

We formulate the sampling strategy as an optimization problem. Our solution[1] possesses the following key favorable properties: (1) it uses a fixed number of seven optimization variables, irrespective of the dimensionality of the data (2) it is *model agnostic* - in that both the interpretable model and the oracle may belong to arbitrary model families (3) it has a flexible notion of model size, and can accommodate vector sizes (4) it is a *framework*, enabling it to benefit from progress in the area of optimization.

We also present the following interesting observations: (a) In general, the optimal training distribution for a model when its size is small, is different from the test distribution; (b) This effect exists even when the interpretable model and the oracle are from highly disparate model families: we show this on a text classification task, by using a *Gated Recurrent Unit* network as an oracle to improve the sequence classification accuracy of a Decision Tree that uses character n-grams; (c) Our technique may be used to identify an *optimal training sample* of a given sample size, for a model.

Empirical results using multiple real world datasets, various oracles and interpretable models with different notions of model sizes, are presented. We observe significant relative improvements in the *F1-score* in most cases, occasionally seeing improvements greater than 100% over baselines.

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, especially in domains where the cost

1. Available as a Python package (Ghose, 2021).

of being wrong is prohibitively high, e.g., medicine and healthcare (Caruana et al., 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 transparency 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 (T. 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 et al., 2018) and 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 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 et al., 2017) and Ablation-CAM (Desai & Ramaswamy, 2020), influence functions (Koh & Liang, 2017) and feature attribution based on Shapley values (Lundberg & Lee, 2017; Ancona, Oztireli, & Gross, 2019).

Anecdotally it seems that interpretable models are preferably small in *size*: a linear model with 10 terms, against one with 100 terms, or a decision tree (DT) of $depth = 5$ as opposed to one of $depth = 50$, is easier to parse by humans. This relationship between interpretability and model size has been scientifically studied as well :

- User studies indicate that small model size is one of a few important factors that makes a model interpretable: Lage et al. (2019) show in the context of decision sets that small model sizes aid interpretability (although it's not the most important property do so); Poursabzi-Sangdeh, Goldstein, Hofman, Wortman Vaughan, and Wallach (2021) find that smaller model sizes aid in certain tasks that require a human subject to have understood how a model works; Feldman (2000) notes that longer Boolean formulae are harder to learn by humans.

  While model size is important, Kulesza et al. (2013) caution against focusing on size in isolation, arguing smaller model sizes can be detrimental to understanding if they are too simplistic. Freitas (2014) highlights this aspect as well.

- This role of model size is variously acknowledged in the design and analysis of interpretable models: Herman (2017) refers to this as low *explanation complexity*, this is seen as important for *simulability* - ease of simulating the reasoning process of a model by a human (Lipton, 2018; Murdoch, Singh, Kumbier, Abbasi-Asl, & Yu, 2019) - and

is often listed as a desirable property in interpretable model representations (Ribeiro et al., 2016; Lakkaraju et al., 2016; Angelino et al., 2017; Murdoch et al., 2019; Bertsimas, Delarue, Jaillet, & Martin, 2019).

Based on existing literature, we note that it is desirable for interpretable models to be small in size. Multiple algorithms explicitly account for it in their model training objective. This is the aspect of interpretability our current work focuses on: *we provide a technique to improve the accuracy of a model of a given size. This technique may be used to produce accurate models of small sizes; which are likely to be easier to interpret than larger models.*

While many existing algorithms constrain model size for a specific model family, *we propose a generic solution for this common requirement, i.e., our technique works on models from an arbitrary model family for a flexible notion of model size.* This also signifies the practical value of our work: *instead of picking an interpretable model family based on accuracy, one may first construct an accurate but possibly large model from a preferred model family, and then use our method to make it compact.*

The challenge with constraining models to small sizes is that size is typically inversely proportional to its bias, and therefore, such a model often trades off accuracy for interpretability. Our technique precisely minimizes this trade-off using a novel form of *adaptive sampling*:

1. We first learn an *oracle*: a highly accurate, possibly black-box, *probabilistic* model trained on the training data. It produces a probability distribution over labels for an instance x:

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

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

2. Next, we try to incorporate the oracle's implicit representation of class boundaries into our interpretable model. The mechanism used is to sample points from the training data based on a learned distribution over the *uncertainty* in the oracle's predictions[2].

3. The interpretable model is then trained on this sample.

We empirically show that this often leads to significant improvements in the classification accuracy, especially when the interpretable model size is small.

Figure 1 depicts our technique on a two-dimensional two-label dataset. The dataset is shown in Figure 1(a). Figure 1(b) visualizes the generalization learned by a *Gradient Boosted Model (GBM)* using this dataset. This serves as our oracle with an $F1$ score of $= 0.84$. Figure 1(c) shows what a *CART* (Breiman et al., 1984) decision tree of $depth = 5$ learns; here $F1 = 0.63$. Finally, Figure 1(d) shows what a CART decision tree of $depth = 5$ learns, when we supply the GBM as an oracle to our technique. There is a significant improvement with $F1 = 0.77$. Visually, we see the boundaries approximating the ones learned by the oracle in Figure 1(b).

These are the key contributions of this work:

2. This is different from *Knowledge Distillation*; discussed in Section 2.2.

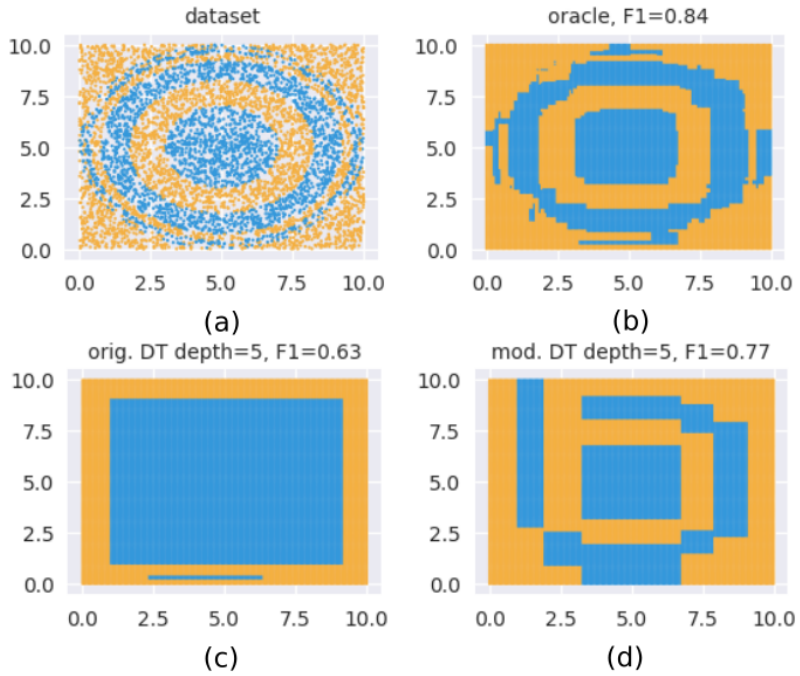


Figure 1: A demo of our technique using a GBM as an oracle. See text for explanation.

1. We propose an algorithm to find a sampling distribution over a training dataset that is optimal in terms of achieving high test accuracy, for a provided model family and model size.

2. This algorithm is *model-agnostic*[3] in that both the interpretable model and the oracle may belong to arbitrary model families, e.g., these can be *Linear Probability Model* and a *Random Forest*, or even a decision tree and a *Gated Recurrent Network (GRU)* respectively. It also admits a flexible notion of model size, e.g., depth of a decision tree, number of terms with non-zero coefficients in a linear model, number of trees *and* maximum depth per tree in a GBM model.

3. The sampling algorithm internally solves an optimization problem to identify the optimal distribution; however, in our formulation only a fixed number of *seven optimization variables* are required irrespective of the dimensionality of the data.

We also present the following significant findings:

1. The proposed technique may be used as a tool to identify and study the optimal training data for a given data size, for a model.

3. We adopt the common usage of the term (Ribeiro et al., 2016; Lundberg & Lee, 2017; Chen, Song, Wainwright, & Jordan, 2018) to imply our technique is agnostic to model *families*.

2. Based on extensive experiments with this algorithm, we make this, possibly counter-intuitive, observation: *in general, the optimal training distribution is not the same as the test distribution, especially at small model sizes*[4].

   We also explicitly show that as model size increases the optimal training distribution progressively approximates the test distribution.

The remainder of the paper is structured as follows: in Section 2, we present details such as terminology and prior work. Section 3 discusses the algorithm in detail while Section 4 presents extensive experimental validation using real-world datasets. In Section 5 we discuss the results and their implications. Section 6 discusses directions for future work and Section 7 concludes the paper.

2. Overview

In this section, we present a formal statement of the problem we are solving, followed by a review of previous work, and then a discussion of where our technique fits in within the standard model-building workflow. Finally, we establish the terminology relevant to the remaining paper.

2.1 Formal Statement

We extend the terminology of Ghose and Ravindran (2020) to rigorously state the outcomes we achieve in this work. 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 sample generated by our learned distribution 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,

4. This "small model effect" reaffirms the observations in our previous work (Ghose & Ravindran, 2020).

- 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 distribution.
  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 using 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, q and p 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:

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

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

2.2 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; Chawla, Bowyer, Hall, & Kegelmeyer, 2002; He, Bai, Garcia, & Li, 2008; Santhiappan, Chelladurai, & Ravindran, 2018), our previous work on the topic (Ghose & Ravindran, 2020) is the only one we are aware of that applies the principle to learn size-constrained models. Here, we compare and contrast with various techniques that have some similarity with ours:

1. **Density Tree based sampling**: In our previous work (Ghose & Ravindran, 2020), we addressed a similar problem of enhancing the accuracy of a size-constrained model. The approach taken was to develop a specific form of decision tree, known as a *density tree*, to capture neighborhood information, and sample from its various nodes to obtain an optimal training distribution. Our current method may be considered a significant evolution of the approach, as it adds the following flexibilities:

(a) The choice of the oracle was restricted to a forest of density trees[5] in our previous work. Here, we might use an oracle from an arbitrary model family. 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 as-is, to improve the accuracy of an interpretable model.

(b) The density trees and the interpretable model had to be constructed on the same (or very similar) feature space. Here, this is not required, and the oracle might be a sequence model that classifies text, while the interpretable model may be an n-gram based classifier. This considerably broadens the scope of our technique. We look at an example in Section 4.3.1.

2. **Knowledge Distillation (KD)**: KD looks at using powerful "teacher" models (similar to our oracle) to learn a smaller "student" model (Gou, Yu, Maybank, & Tao, 2021). The key differences with KD are:

   (a) Unlike KD our goal is not to approximate the oracle's performance. In fact, we ignore the oracle's label assignments entirely. This is in contrast to KD methods that may use teacher-assigned labels (Bucilă, Caruana, & Niculescu-Mizil, 2006) or distribution of label confidences (Hinton, Vinyals, & Dean, 2015a)[6], or in general, focus on extracting "dark knowledge" from the oracle in some form. Instead, our goal is to evolve the smaller model towards a more accurate version.

   (b) A lot of KD research focuses on Neural Networks, e.g., *FitNets* (Romero et al., 2015), *DistilBERT* (Sanh, Debut, Chaumond, & Wolf, 2019). In contrast, our technique is model-agnostic.

   (c) Methodological differences aside, our observations suggest that there might be *no oracle required* for obtaining the optimal training distribution (discussed in Section 6).

3. **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.

5. We don't refer to the density trees as oracles in our previous work, but they play a role similar to the oracle here.
6. While we use the uncertainty in the oracle's prediction, note that we don't know which labels is the oracle more or less uncertain about, i.e., we ignore label *identity*.

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.

4. **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 (Thrun & Mitchell, 1994), Markov Logic Networks (Mihalkova & Mooney, 2006) or task-specific neural networks like *BERT* (Devlin, Chang, Lee, & Toutanova, 2019) or *ULMFiT* (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[7], 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; W. 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.

2.3 Workflow

Figure 2 compares (a) a standard workflow to our (b) model building workflow. The arrows represent flow of information. 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" 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.

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

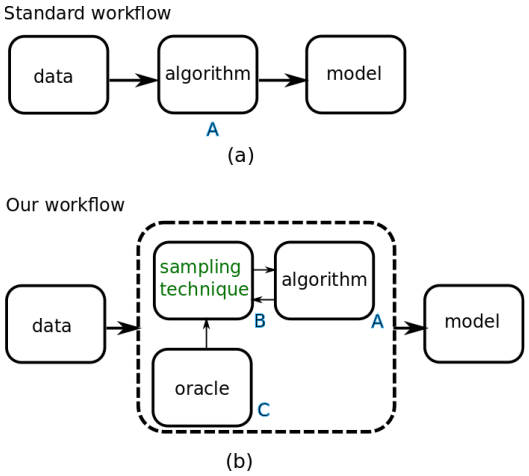


Figure 2: Modified workflow. Arrows denote flow of information. Our sampler B receives uncertainty information from the oracle C, which it then uses to iteratively learn a distribution, using the performance of A as its objective function.

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.

Only the first criteria above is required for using our technique. The second criteria reflects the usefulness of the technique for interpretability.

It must be noted that the notion of model size is subjective. Consider a 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 used:

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. The distribution of test datasets or held-out datasets is the original distribution for *all models discussed in this paper.*

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 look at our methodology.

3. Methodology

We describe our methodology in this section. We begin with the intuition, and then look at the algorithm and various implementation details.

3.1 Intuition

Our intuition here builds upon certain observations from our previous work (Ghose & Ravindran, 2020). There, to find an optimal training distribution:

1. We learn a specific form of decision trees we refer to as *density trees*, to capture neighborhood information in the input space.

2. A node sampling distribution, defined over both internal nodes and leaves, is iteratively learned in the following manner: nodes are sampled based on the current distribution, and then data is sampled from within them. A classifier trained on this data is evaluated on a held-out set. This accuracy is used to modify the node sampling distribution, so it leads to greater accuracy in the next iteration.

In our analysis of the results, we had observed that the learned distribution uses nodes from different depths. Since different depths represent varying levels of information, and therefore, uncertainty, about the location of class boundaries - information increases from the root towards the leaves - this indicates that a sampling distribution using this information *necessarily needs to be learned* as opposed to using simple heuristics like only sampling from regions of high uncertainty. This observation informs the key intuition behind this work: we view the problem of sampling training data as one of learning a distribution over uncertainty scores provided by an oracle. Of course, we validate this assumption empirically in Section 4.2.

We now discuss the various algorithmic details.

3.2 Measuring Uncertainty

We begin by discussing the measurement of uncertainty, since our technique critically depends on this quantity. We denote the uncertainty of prediction by a model M on an instance x by $u_M(x)$, where $u_M(x) \in [0, 1]$. A good uncertainty metric for our application (a) should not exclusively consider the confidence of the predicted label (b) should result in a high value even if the model is uncertain between two labels in a multi-class problem.

The **margin uncertainty** (Scheffer, Decomain, & Wrobel, 2001) metric satisfies these criteria. This is computed as:

$$u_M(x) \leftarrow 1 - (p_{C_1} - p_{C_2}) \tag{5}$$

Here, p_{C_1} and p_{C_2} denote the probabilities of the most confident and next most confident classes, provided by model M for instance x. Lower differences between the top two probabilities lead to higher scores for this metric. We calibrate (J. C. Platt, 1999) our oracles for reliable probability estimates.

See Section A.4 for a discussion on suitability of other uncertainty metrics.

3.3 Sampling based on Uncertainty

Since we want to learn a distribution over uncertainties, $p(u_M(x))$ needs to have a flexible representation. 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 $u_M(x) \in [0, 1]$.

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 (GMM)*, 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. Further discussed in Section 3.5.

The *Infinite Beta Mixture Model (IBMM)* (Ghose & Ravindran, 2020) satisfies the above requirements.

The IBMM is a *Dirichlet Process (DP) 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 DP is described by the *concentration parameter* $\alpha \in \mathbb{R}_{>0}$, which identifies the components that have at least one point assigned to them[8]. The shape parameters of

8. In theory, the DP has an infinite number of components, with only a finite number of them actually representing instances in the data

all the *Beta* components are drawn from shared prior distributions, which themselves are *Beta* distributions. Use of a DP, with shared 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 common 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 the probabilities across instances 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 3.4.

The above procedure is summarized in Algorithm 1. 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).

3.4 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 technique uses two additional parameters:

1. $p_o \in [0, 1]$, proportion of instance-label pairs from the original training data. This parameter serves two purposes: (1) it acts as a "shortcut" for the optimizer to sample from the original distribution, as opposed to determining the right Ψ to do so (2) the relationship of p_o and model size enables us to study the correlation between model size and effectiveness of the original distribution during training.

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 $\Psi = \{\alpha, a, b, a', b'\}$.

Algorithm 1: Sample based on uncertainties and Ψ

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'

Our technique randomly initializes Φ, creates a sample based on Algorithm 1 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 the parameters Φ, and repeats the process. Our stopping criteria is an iteration budget T. Algorithm 2 lists these steps.

Some details to note in Algorithm 2:

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 the oracle training step in 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.

Algorithm 2: 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 splits $D_{train}, D_{val}, D_{test}$ from D, stratified wrt labels
2 $M_O \leftarrow train_{\mathcal{O},h}(D_{train}, *)$
3 **for** $t \leftarrow 1$ **to** T **do**
4 $\quad \Phi_t \leftarrow suggest(s_1, ...s_{t-1}, \Phi_1, ..., \Phi_{t-1})$ // randomly initialize at $t = 1$
$\quad$ // 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 $\quad N_o \leftarrow p_{o,t} \times N_{s,t}$
6 $\quad N_u \leftarrow N_{s\_t} - N_o$
7 $\quad D_o \leftarrow$ uniformly sample, with replacement, N_o points from D_{train}
8 $\quad D_u \leftarrow$ sample N_u points from D_{train} using **Algorithm 1** with input $(N_u, M_O, D_{train}, \Psi_t)$.
9 $\quad D_s \leftarrow D_o \uplus D_u$ // D_o, D_u are assumed to be multisets
10 $\quad M_t \leftarrow train_{\mathcal{I},g}(D_s, \eta)$
11 $\quad s_t \leftarrow accuracy(M_t, D_{val})$
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^*

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.

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 and GBMs).

It is important to note here that D_{val} *and* D_{test} *are not modified by our algorithm in any way*, and therefore s_t and s_{test} measure the accuracy on the original distribution.

Algorithm 2 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.

3.5 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 2. 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* 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 *Kernel Density Estimators (KDE)*; 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 (Feurer & Hutter, 2019; Turner et al., 2021).

Among BO techniques, of which there are many today, e.g., (Hutter, Hoos, & Leyton-Brown, 2011; Bergstra, Bardenet, Bengio, & Kégl, 2011; Malkomes & Garnett, 2018; Z. 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[10] and has a popular and mature library: *Hyperopt* (Bergstra, Yamins, & Cox, 2013).

We note here that *TPE* supports conditional parameter spaces, which would have allowed us to use a finite mixture model such as GMMs, setting the number of mixture components as the top level optimization variable. However, our design choice of a fixed parameter space effectively makes our technique a **framework**: any optimizer that satisfies the above criteria may be used. *This enables us to make Algorithm 2 faster and better as newer optimizers become available.* For example, any of the BO algorithms from the *black-box optimization challenge*, NeurIPS2020 (Turner et al., 2021), may be used to implement *suggest*() in Algorithm 2.

9. The connotation of this term here is different from what we have seen before: it denotes variance in the response surface model.
10. The runtime complexity of a naive BO algorithm is *cubic* in the number of evaluations (Shahriari, Swersky, Wang, Adams, & de Freitas, 2016).

3.6 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 2. Since BOs model the response surface of the actual objective function using a finite number of evaluations (s_t in Algorithm 2), 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 1, 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" the distribution[11] within $[0, 1]$. Our transformation is simple: we divide the interval $[0, 1]$ into B bins, and map approximately $|D_{train}|/B$ uncertainty scores to each bin, while maintaining order between the original and mapped scores. Within a bin, the mapped scores are linearly spread across its range. This distributes the mapped scores approximately uniformly in the range $[0, 1]$. The algorithm is detailed in Section A.5.

The alternative to flattening is to identify a suitable parameter for the BO algorithm, e.g., a suitable *kernel* for Gaussian Process based BO. However, this introduces additional hyperparameters; hence we prefer flattening.

Figure 3 visualizes the process of flattening. The original and modified uncertainty distributions for the datasets `Sensorless` and `covtype.binary` are shown in Figure 3(a) and 3(b) respectively.

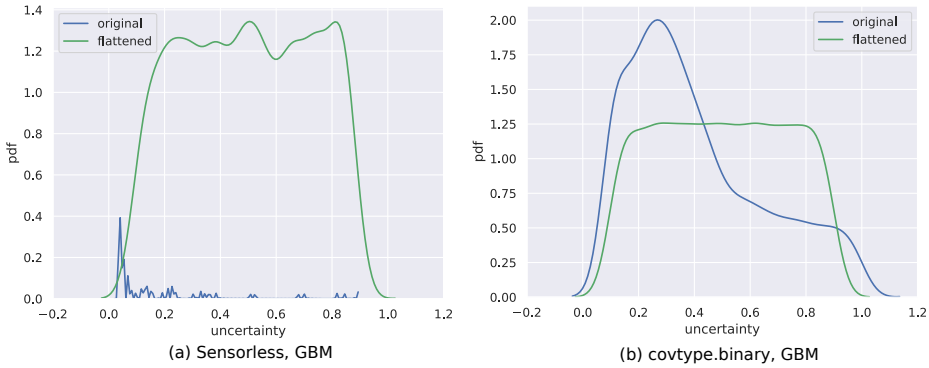


Figure 3: Example of curve-flattening, for datasets (a) `Sensorless` and (b) `covtype.binary`. The uncertainty scores shown are obtained using the *GBM* oracle.

While `Sensorless` appears to have a non-smooth distribution, and flattening here might help, this seems redundant for `covtype.binary`. *However, since this step is computationally*

11. 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 distribution transformation applied to a loss landscape (Santurkar, Tsipras, Ilyas, & Madry, 2018).

inexpensive, we perform this for all our experiments, saving us the effort of assessing its need. The effect of flattening in our experiments is discussed in Section 5.

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 1. Instead of sampling based on the actual oracle uncertainty scores:

$$p_j \leftarrow Beta(u_{M_O}(x_j); A_i, B_i) \tag{6}$$

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

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

The use of the transformation is optional, since Algorithm 2 does not critically depend upon it, but makes it robust (discussed in Section 5).

□

This concludes our discussion of algorithmic details. In summary, we require seven parameters $\Phi = \{\Psi, N_s, p_o\}$, where $\Psi = \{\alpha, a, b, a', b'\}$. Hyperparameters are discussed in Section 4.1.5. Our experimental validation of the technique is discussed next.

4. Experiments

We now look at extensive evaluation of our technique. Our experiments maybe categorized into the following types:

1. **Validation**, Section 4.1: this set of experiments exhibit statistically significant improvements across multiple datasets, using different models and oracles (Section 4.1.6). Various properties of the learned distributions are analyzed (Section 4.1.7). The relationship between model capacity and the efficacy of our technique is also discussed (Section 4.1.8).

2. **Comparisons**, Section 4.2: here we compare the improvements produced by our technique with (a) a supervised version of uncertainty sampling and (b) using density trees.

3. **Additional applications**, Section 4.3: fundamentally, our technique learns a sampling distribution that leads to effective training. This can be used as a tool for the following interesting applications - (a) different feature representations may be used across the interpretable model and the oracle, e.g., a DT as the interpretable model with n-grams as input, and a Gated Recurrent Unit the oracle, that operates on a sequence of tokens, (b) a minimal sample for effective learning maybe identified using our technique and (c) a multivariate notion of model size may be used.

The section on validation experiments is the most comprehensive, establishing various aspects of our technique.

4.1 Validation

To empirically 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. The experimental setup is described in terms of the data, models and oracles, metrics used and the optimization search space explored.

4.1.1 Data

We use *13* real-world datasets to validate our technique. Table 1 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 1. 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 2). The data splits are stratified wrt class labels.

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

$$\text{Label Entropy} = \sum_{j \in \{1,2,..,C\}} -p_j \log_C p_j \tag{8}$$

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

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

4.1.2 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 et al., 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 1, 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) the technique is well validated for the case of linear classification by any standard linear classifier.

Table 1: We use the following datasets available on the LIBSVM website (Chang & Lin, 2011). Their original source is mentioned in the "Description" column. 10000 instances from each dataset are used. A $train : val : test$ split ratio of $60 : 20 : 20$ is used for D_{train}, D_{val} and D_{test} in Algorithm 2. The splits are stratified wrt labels.

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 (J. 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 et al., 2013). We use the transformations from (C.-C. Wang et al., 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).

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 1).

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 a tree (with no size constraints) with the highest *F1-score* (macro) using standard 5−fold cross-validation. We refer to this as the optimal tree T_{opt}, and its depth is denoted by $depth(T_{opt})$. We then experiment up to a model size of $min(depth(T_{opt}), 15)$. This is controlled by setting the values of CART's max_depth parameter to: $\{1, 2, ..., min(depth(T_{opt}), 15)\}$.

   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[12]. This is important to remember since 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*, although the parameter controlled is max_depth.

4.1.3 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 et al., 2017) is used in our experiments. Effective parameters were determined using a validation set. **NOTE:** This is *not* D_{val} from Algorithm 2, since that would constitute *data leakage*. A sample, stratified by labels, from within D_{train} was held out for learning good GBM parameters.

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

The above oracles were calibrated (J. C. Platt, 1999) for reliable probability estimates.

4.1.4 Metrics

We measure two quantities - improvements in model accuracy and their statistical significance. These are the metrics we use:

12. The training phase may be declared complete before growing till max_depth, based on other settings like leaf purity, minimum number of samples required at a leaf, etc.

1. To measure *accuracy*() as in Equation 2 or Algorithm 2, 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:

   $$\delta F1 = \frac{100 \times (F1_{new} - F1_{baseline})}{F1_{baseline}} \tag{9}$$

   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 2, 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 9.

   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. To measure statistical significance of our results we use the *Wilcoxon signed-rank test*, where the paired set of samples are $F1_{baseline}$ and $F1_{new}$ scores (from Equation 9) for a dataset. The *p-value* is reported. This test is separately performed for different model sizes.

4.1.5 Optimization Search Space

The optimizer we use, TPE, requires *box constraints*. Here we specify our search space for the optimization variables, Φ in Algorithm 2:

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:

   $$E[k|\alpha] \approx 5\alpha + 2 \tag{10}$$

We empirically estimated 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 3.6) 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.

Hyperparameters: In theory, the box constraints and the iteration budget required by the optimizer constitute our hyperparameters, which may be tuned for a specific task. However, as we note above, we don't need to estimate a range for p_o and reasonable defaults may be applied to N_s, $\{a, b, a', b'\}$, *scale* and α. This results in the practical convenience of having to set the value for only a single hyperparameter: T, the iteration budget. This was set to $T = 1000$ for LPMs and $T = 3000$ for DTs based on limited search. Since the LPMs we use construct multiple *one-vs-rest* classifiers, higher iteration budgets are computationally expensive to use.

□

This completes our discussion of the experimental setup; we present our observations next.

4.1.6 Improvements in Accuracy

Figure 4 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 4.1.4.

Table 2 enumerates the observations corresponding to the plots in Figure 4. 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 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 2 see the entry for `letter`,

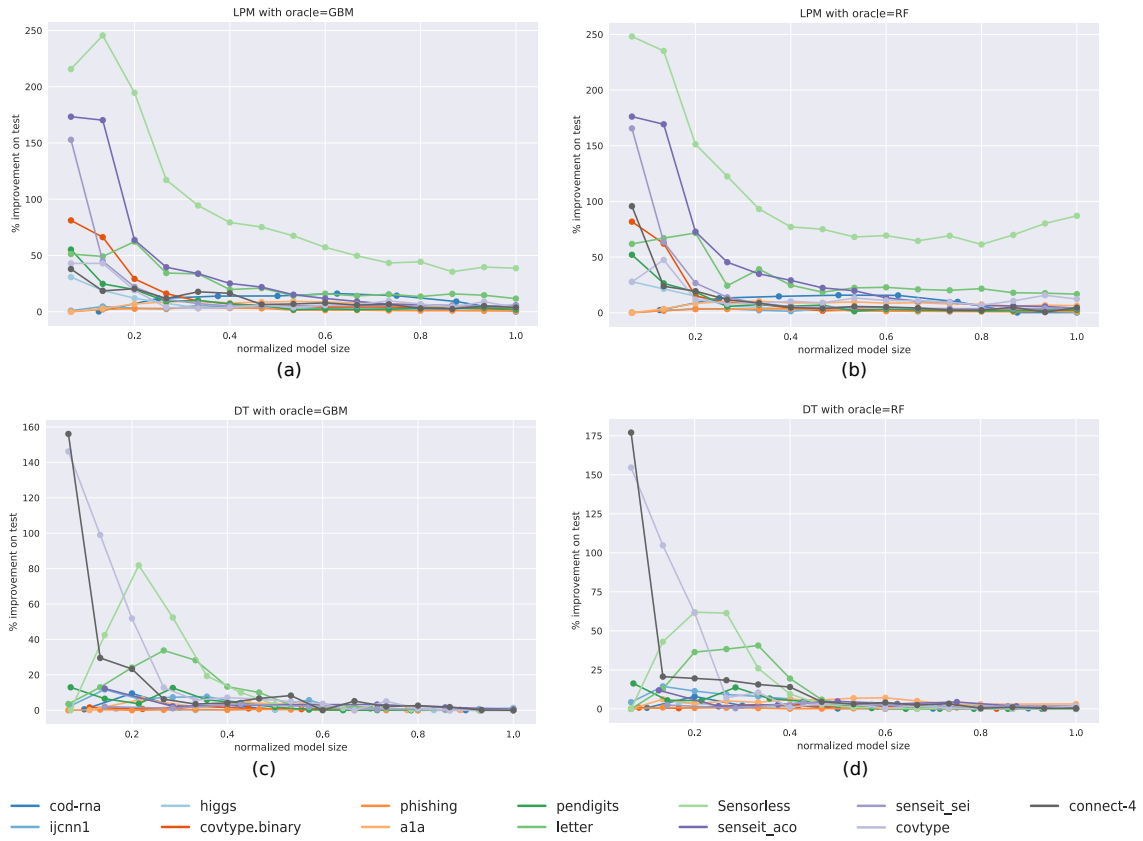


Figure 4: For different combinations of models and oracles: $\{LPM, DT\} \times \{GBM, RF\}$, these plots show improvements, $\delta F1$, seen for different model sizes and data. Table 2 shows the corresponding improvement scores.

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.

The statistical significance of the difference between $F1_{baseline}$ and corresponding $F1_{new}$ scores were verified using the *one-sided* version of the paired *Wilcoxon signed-rank test* - this is detailed in Section A.6.

Before we conclude this section, we present an additional way to visualize improvements: create a correspondence of model sizes, without and with our technique, for the same accuracy. See Figure 5 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. See Section A.8 for more compaction profiles.

Table 2: 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**

4.1.7 Learned Distributions

It is also instructive to analyse the distributions we have learned: this includes both the parameter p_o and the parameters for the IBMM $\Psi = \{\alpha, a, b, a', b'\}$.

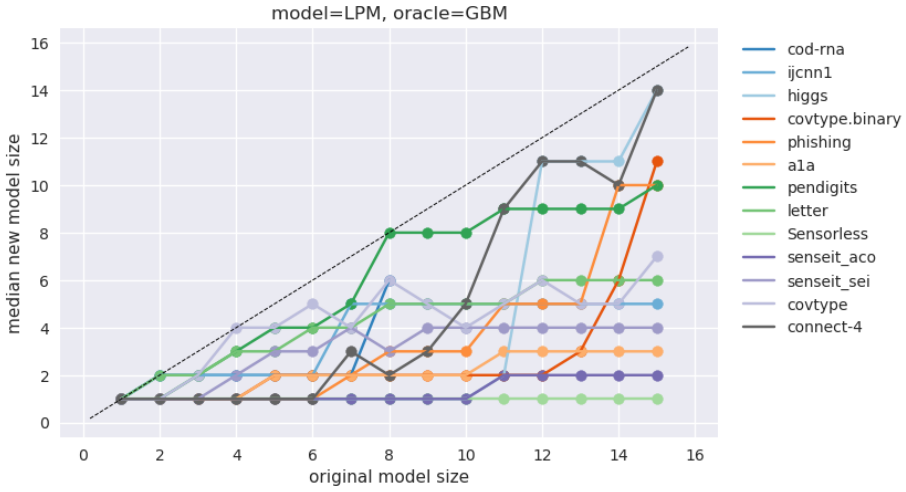


Figure 5: The compaction profile of LPM models using GBM as an oracle. A point (x, y) denotes the minimum size y of a model obtained using our technique that is *at least* as accurate as the baseline model of size x.

Figure 6 shows how p_o varies with normalized model size when the interpretable model is DT and the oracle is (a) GBM or (b) RF. This plot ignores the datasets where the largest tree depth explored was less than $depth(T_{opt})$ - so we can compare distributions in size regimes where our technique is effective against when it is not (recall, at sizes close to $depth(T_{opt})$ we expect $\delta F1 \approx 0$). The datasets ignored are[13]: `a1a, ijcnn1, covtype, connect-4`. Here, we clearly see $p_o \to 1$ as model size increases, thus implying the training algorithm tends to use more of the original distribution[14]. This observation is a **key contribution** of this work, since it challenges the conventional wisdom that the training data must be drawn from the same distribution as the test data, for effective learning. This reinforces a similar observation from our previous work (Ghose & Ravindran, 2020).

We now consider the IBMM distributions over the uncertainty values. These are difficult to concisely visualize since one IBMM is learned for *each* model size. Hence, we propose the following plot that aggregates distributions across model sizes for a dataset:

1. We set a value for N; the number of points to sample.

2. For a model size η_i, we sample n_i points from its corresponding IBMM, where $n_i \propto \delta F1_i$, the improvement seen at this size. For example, let's say we have explored two model sizes η_1, η_2, and these have led to improvements of $\delta F1_1 = 10\%$ and $\delta F1_2 = 20\%$, respectively. Then, $n_1 = 0.33N$ and $n_2 = 0.67N$.

13. These datasets are easy to identify in Table 2: the ones where the last column(s) is neither ≈ 0 nor "-".
14. In theory, the parameters Ψ could have been learned such that they mimic the original distribution, but we hypothesize that it is easier for the optimizer to learn the appropriate value of one parameter p_o as opposed to equivalent values of the multiple parameters Ψ. This is why we see the clear pattern in Figure 6.

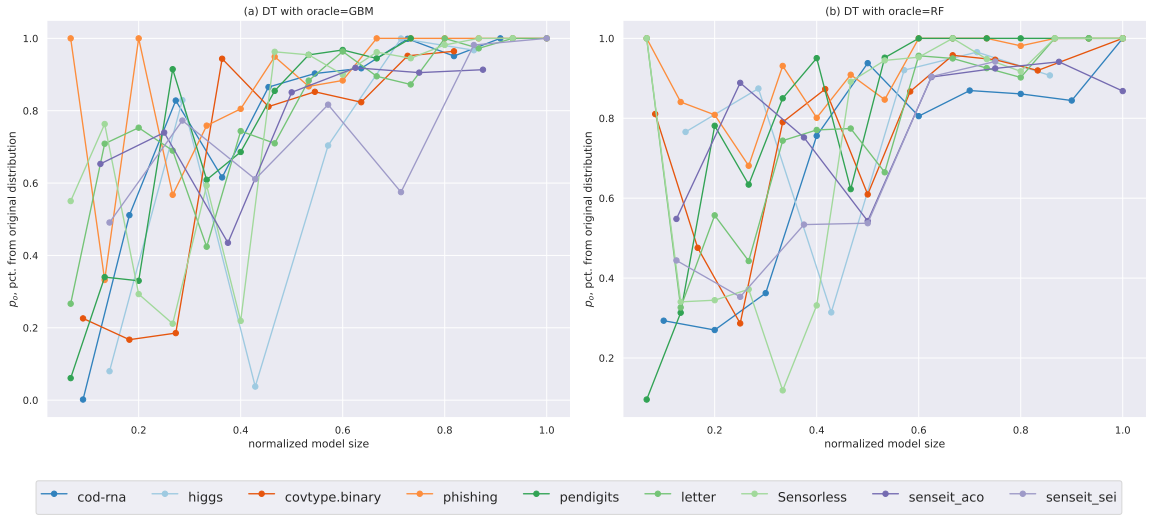


Figure 6: These plot shows the effect of increasing model size on p_o, when the interpretable model is DT. These plots strongly indicate that larger model sizes learn better with the original distribution. Some datasets are ignored - see text for explanation.

3. The various samples of sizes n_i are pooled together and a *Kernel Density Estimator (KDE)* fit on this data is visualized.

The KDE thus obtained is predominantly shaped by the distributions that resulted in high $\delta F1$. For the case of the LPM these are visualized in Figure 7 for both oracles.

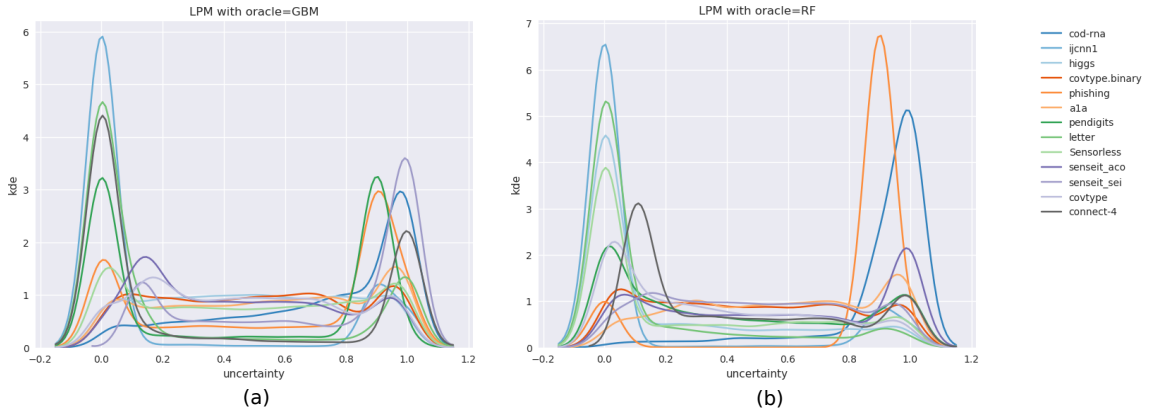


Figure 7: The aggregated IBMMs are visualized for LPMs, when the oracle is a (a) GBM or (b) RF. The corresponding plots for DTs are presented in Section A.7.

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.

Going a step further we might wonder what the aggregated 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 Figure 7 simply because most points receive a high uncertainty score.

We use the following technique to visualize such an *adjusted* aggregate distribution:

1. We again pick N, the number of points to sample. Exactly like in the previous case: we pool together samples from IBMMs for different model sizes, where the relative sample sizes are decided by the respective $\delta F1$ scores. We fit a KDE to this data, which we refer to as A.

2. We fit another KDE to the uncertainty values produced by the oracle for the training data. Let's call this B.

3. For K uniformly spaced values of uncertainty $u_k \in [0, 1], 1 \leq k \leq K$, we calculate the ratio $p_A(u_k)/p_B(u_k)$, and plot a scaled version of it $c \cdot p_A(u_k)/p_B(u_k)$. The scaling factor c is picked to transform the ratios into probability masses, i.e., $\sum_{k=1}^{K} c \cdot p_A(u_k)/p_B(u_k) = 1$.

   Essentially, we *normalize* the sampling probability $p_A(u_k)$ at u_k, with $p_B(u_k)$, a quantity representing the number of instances with uncertainty u_k.

These plots are shown in Figure 8. The corresponding plots for the DT are shown in Figure 23, Section A.7.

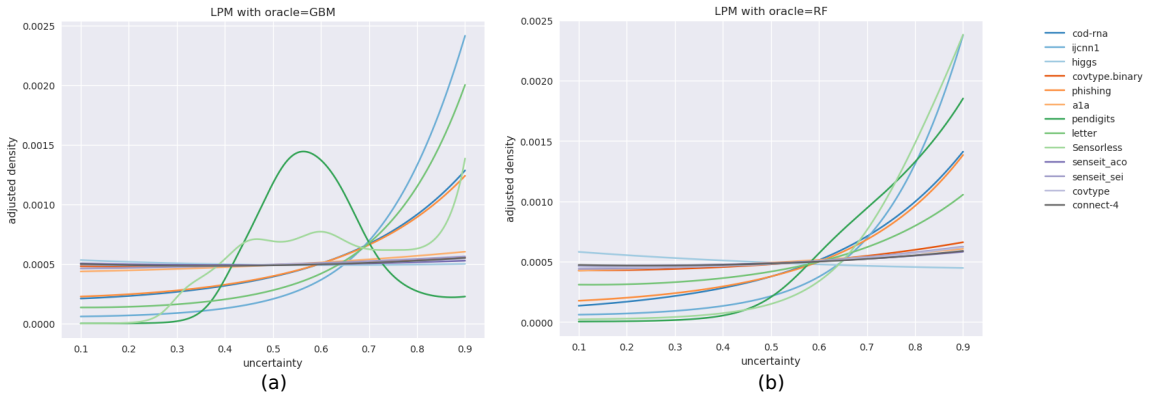
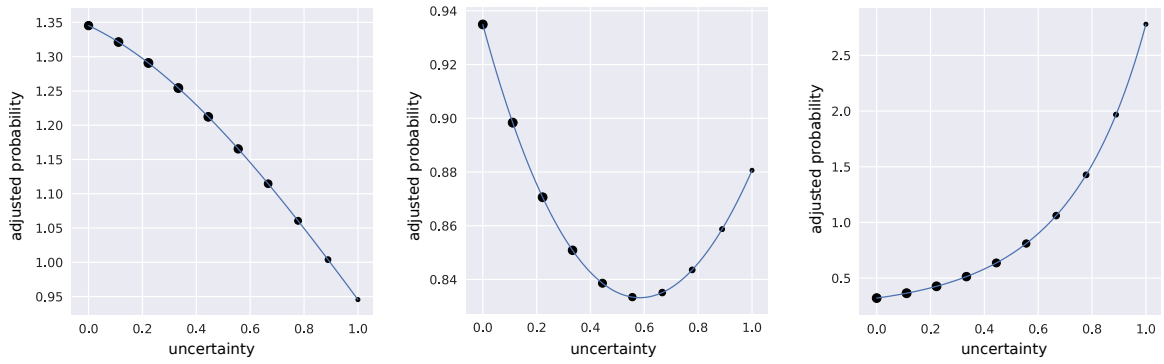


Figure 8: Aggregated IBMMs, adjusted for the uncertainty distribution. These plots are for the LPM, using a (a) GBM or (b) RF as an oracle. The corresponding plots for DTs may be found in Section A.7.

While the plots in Figure 7 are indicative of the individual distributions they aggregate (most of the individual distributions have similar shapes; see Figure 25 in Section A.9), this is not true for the adjusted plots in Figure 8 - there are diverse variations that are averaged out. We show some of them in Figure 9, for different datasets and model sizes, for $model = LPM, oracle = GBM$. The size of the dots on the curve represent $p_B(u_k)$ at



(a) a1a, size=2 (b) letter, size=2 (c) letter, size=3

Figure 9: 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.

the corresponding value of u_k on the x-axis. These are intended to signify robustness of the adjustment, since they occur in the denominator of the scaled ratios.

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 7.

4.1.8 Effect of Model Capacity

If we closely look at the improvements in Table 2, we would note that the improvements for DTs diminish faster than LPMs, as model size increases. This naturally leads to the question: how does model capacity influence improvements? This is difficult to answer in general since (a) there isn't a standard way to easily quantify capacity across model families, and (b) the notion of model size is subjective. And while the LPM vs DT data indicates a trend, we want to isolate this effect in a manner that is not affected by differences in the model families.

To that end, we adopt the following approach: we use two different instances of GBMs, where the notion of model size is the number of DTs in a GBM (or equivalently, the number of boosting rounds), and their model capacities are decided by the maximum depth of the constituent DTs; these are set to 2 and 5 for these GBM instances. We refer to these as the *GBM-2* and *GBM-5* "pseudo model families" respectively, where we understand *GBM-5* to possess higher capacity than *GBM-2*. Since the training algorithms and model representations are identical for *GBM-2* and *GBM-5*, this setup allows us to sidestep challenges with quantifying capacity for different model families.

The oracle used is another GBM, with no size/capacity restrictions, learned on the training dataset. The model sizes explored are $\{1, 2, ..., 10\}$. Figure 10 shows how $\delta F1$ varies with model size (denoted as "num_trees") for the datasets `senseit-aco, senseit-sei, cod-rna` and `higgs`, for each of the models *GBM-2* and *GBM-5*.

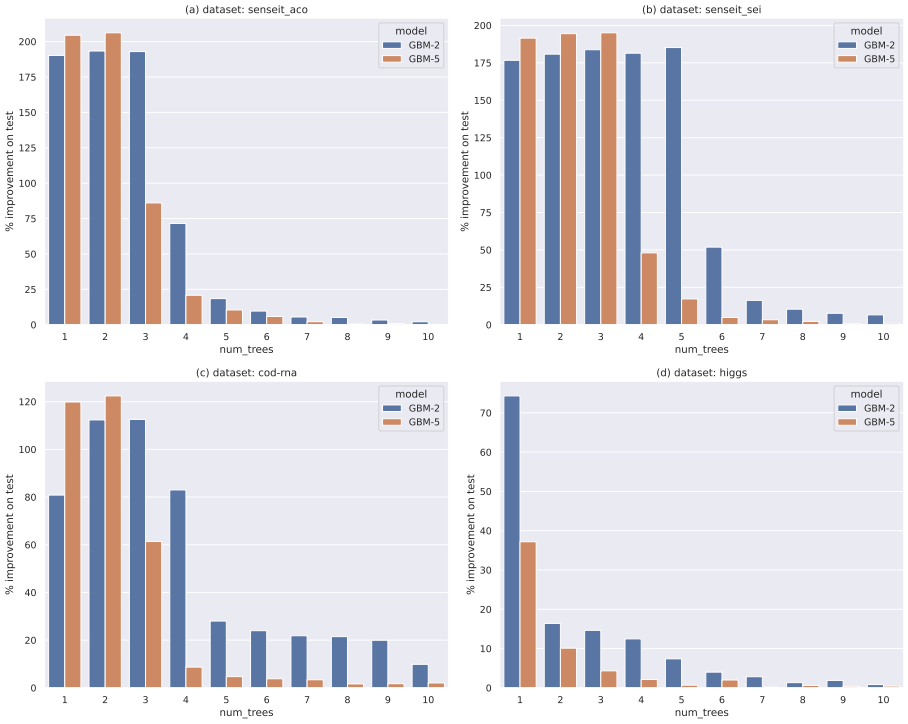


Figure 10: The above plots show how the capacity of a model family influences improvements, for different datasets. With a higher *max_depth* setting for GBMs, the improvements decline faster with an increase in number of trees.

As we might expect, we observe that improvements for *GBM-5*, the model with the higher capacity, diminish faster with increasing size, compared to *GBM-2*.

4.2 Comparisons

In this section, we present a comparative evaluation of our technique. We compare against the following techniques:

1. *Supervised* uncertainty sampling: the interpretable model, of a given size, is iteratively trained on a growing subset of training data; this subset starts with the b most uncertain points in the training data, with b-sized batches of the most uncertain points from the remaining training data being progressively added to it. At every iteration, the model is evaluated on a validation set, and the one with the highest $F1$-macro score is picked for comparison. We compare against this technique because:

   (a) This explores an obvious possibility: can a heuristic-driven, simple algorithm outperform our algorithm?

   (b) Although we borrow this technique from Active Learning (Lewis & Gale, 1994), this version is significantly more powerful, primarily because of the oracle's supervision: we have reliable uncertainty scores from a powerful model. Because of this, we are able to avoid sampling bias arising due to a partial view of the uncertainty distribution (detailed in Section A.2).

   (c) Even within the Active Learning community, uncertainty sampling is a strong baseline for Logistic Regression (Yang & Loog, 2018), and by extension, we expect it to be a strong baseline for learning LPMs.

   We use a batch size of $b = 10$. The algorithm is described in detail in Section A.1.

2. Density Trees: We also compare against our previous work on density trees since it uses a similar philosophy of determining an optimal distribution to build accurate small models. We use the parameter search space described in Ghose and Ravindran (2020).

4.2.1 Setup

The experimental setup is identical to the one used for the validation experiments in terms of the datasets (see Section 4.1.1), models (Section 4.1.2), oracles (Section 4.1.3) and optimization search space (Section 4.1.5). The metrics differ, and these are described next.

4.2.2 Metrics

To compare techniques, we wish to measure the following outcomes over multiple trials:

1. The extent to which a technique is better.
2. The proportion of times a technique is better.

The following properties are desirable for a metric that measures the first kind of outcomes:

1. It should be bounded, so that scores across different data, model sizes, etc., are on the same scale.

2. It should be easy to infer which approach is better.

We introduce a score called the *Scaled Difference in Improvement (SDI)*, that possesses these properties. The SDI is defined in terms of the improvement produced by our method, $\delta F1_{ora}$, and the alternative method, $\delta F1_{alt}$:

$$SDI = \begin{cases} \frac{\delta F1_{ora}}{H} - \frac{\delta F1_{alt}}{H}, & \text{if } H > 0 \\ 0, & \text{if } H = 0 \end{cases} \quad (11)$$
$$\text{where } H = \max\{\delta F1_{alt}, \delta F1_{ora}\}$$

The central idea here is that the improvements possible across the competing techniques are in $[0, H]$, and the SDI score measures the difference between the fractions 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:

1. $SDI \in [-1, 1]$

2. $SDI > 0$ when $\delta F1_{ora} > \delta F1_{alt}$

3. $SDI = 0$ when $\delta F1_{ora} = \delta F1_{alt}$

4. $SDI < 0$ when $\delta F1_{ora} < \delta F1_{alt}$

The SDI score may be seen as the *Mean Signed Deviation*[15] *(MSD)*: $\delta F1_{ora} - \delta F1_{alt}$, normalized with the maximum possible improvement H. We don't directly use MSD as $\delta F1 \in [0, \infty)$ makes it unbounded.

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 $\overline{SDI}$.

To measure the second kind of outcomes, we report the percentage of times $\delta F1_{ora} > \delta F1_{alt}$ across these model sizes. This is denoted as *pct_better*.

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

4.2.3 Observations

Table 3 and Table 4 compare our approach to Supervised Uncertainty Sampling and the Density Tree based approach, respectively. All $\delta F1_{ora}$ and $\delta F1_{alt}$ scores used are the *average over three runs*. This is the presentation format followed:

1. For each dataset, model and oracle combination we present two scores: (1) $\overline{SDI}$ and (2) *pct_better*.

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

15. `https://en.wikipedia.org/wiki/Mean_signed_deviation`

Table 3: LPM, DT compared to Supervised Uncertainty Sampling

	LPM			DT		
dataset	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%

3. In the case of Supervised Uncertainty Sampling, Table 3, 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.

4. Unlike supervised uncertainty sampling, there is no notion of an oracle in the Density Tree based approach. In Table 4, 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.

5. We also 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 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 identified by **OVERALL** and **ANY** provide comparison numbers aggregated over datasets, model sizes and oracles.

The predominant amount of values colored green, indicate that our technique performs better in most settings. In both cases, the **OVERALL** +**ANY** entries indicate that our technique works better on average. The *pct_better* scores in these entries also indicate that we seem to do better much more frequently in the case of $LPMs$ than DTs.

We note here that the space of sampling distributions modeled by our technique subsume the ones modeled by either competing technique:

1. Supervised Uncertainty Sampling assumes high uncertainty points are favorable; this may be modeled with an IBMM with appropriate parameters.

Table 4: LPM, DT compared to the Density Tree approach.

	LPM			DT		
dataset	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%

2. Density Trees learn distributions that are based on the proximity of instances to class boundaries; since uncertainty values also correlate with distance from class boundaries - a high uncertainty value for an instance indicates it's near a class boundary and vice versa - this too is well within the scope of what an IBMM may represent.

Our hypothesis as to when the competing techniques outperform our technique is that the optimal sampling distribution is easier to discover given their distributional assumptions. For example, if the optimal distribution indeed turns out to be one where instances with high uncertainty are preferred, the Supervised Uncertainty Sampling technique would quickly discover this, while our technique would need to navigate a larger search space to converge to this solution. Our technique would likely do better on such problems with a larger iteration budget or an appropriately defined prior; we leave this analysis for future work.

Both Supervised Uncertainty Sampling and our technique use distributions over uncertainty values. This makes it interesting to contrast them, and is reviewed in Section A.3.

4.3 Additional Applications

Viewing our technique purely as a tool to find the optimal distribution for effective learning, we explore some additional interesting applications of it in this section.

4.3.1 Different Feature Spaces

In our previous experiments, the feature vector representation was identical for the oracle and the interpretable model. This is also what Algorithm 2 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 et al., 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 an n-gram vocabulary, which is then used to create a sparse binary vector per surname - 1s and 0s indicating the presence and absence of an n-gram respectively.

Figure 11 shows a schematic of the setup.

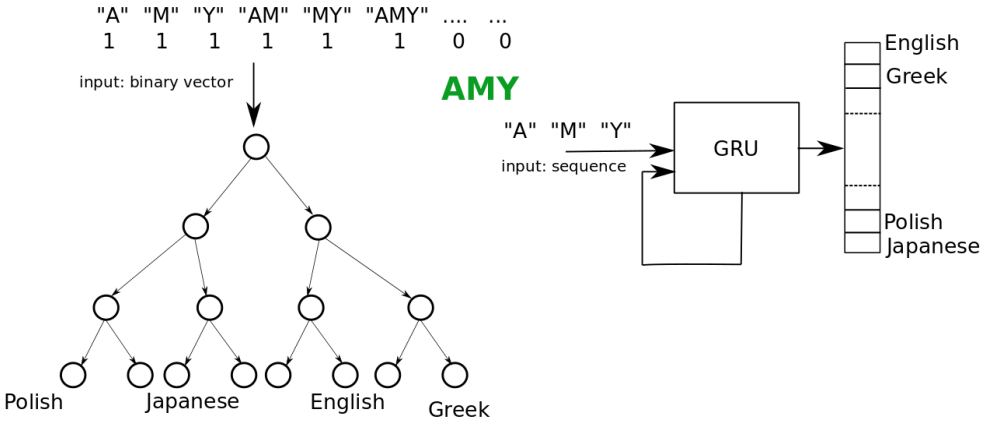


Figure 11: The feature representations for the oracle and the interpretable model may be different. Consider the name "Amy": the GRU is provided its letters, one at a time, in sequence, while the DT is given an n-gram representation of the name.

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 A.11 for details). DTs of different *depth* ≤ 15 were trained. A budget of $T = 3000$ iterations was used (the search space for Φ is the same as in Section 4.1.5), and the relative improvement in the $F1$ macro score (as in Equation 9) is reported, averaged over three runs. Figure 12 shows the results.

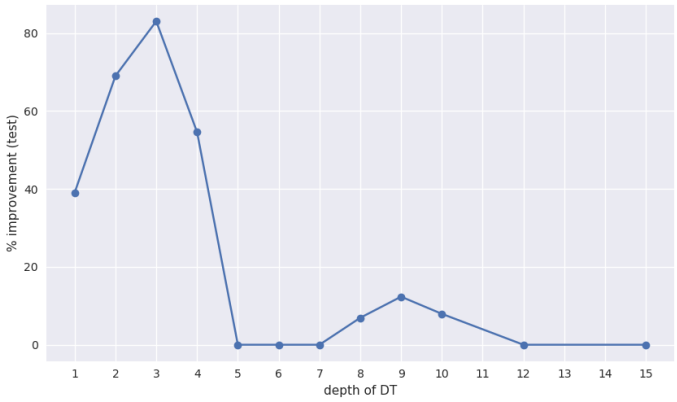


Figure 12: Improvements $\delta F1$ are shown for different depths of the DT.

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.

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 $F1 = 23.08\%$.

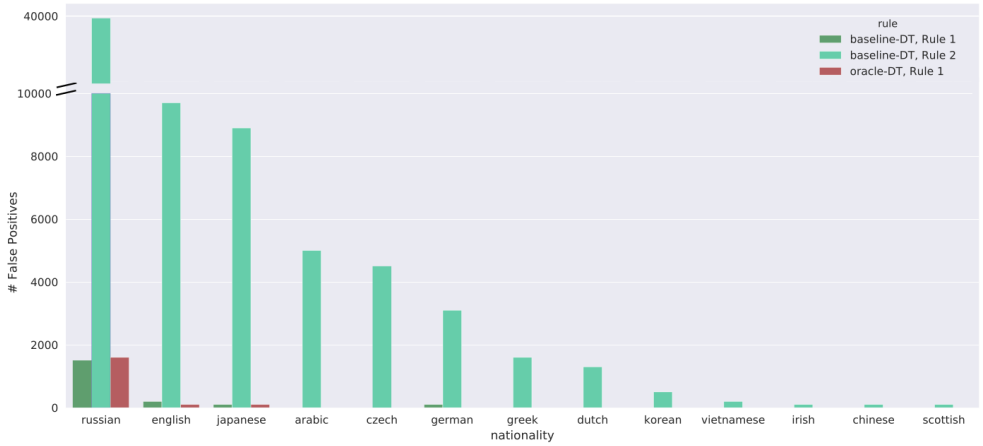


Figure 13: The distribution of nationalities in false positive predictions for the baseline and oracle based models, shown for predicting *Polish* names. Only nationalities with non-zero counts are shown.

The difference in rules may also be visualized by comparing the distribution of nationalities represented in their false positives, as in Figure 13. 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.

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.

4.3.2 Size-Constrained Training Sample

Recall from Section 3.4, we make use of a parameter N_s, denoting sample size, that we had constrained to $\in [400, 10000]$ (Section 4.1.5) in our experiments. But it is possible to set this to much smaller values to study the sampling distribution for patterns, significance of regions in the input space, etc. Figure 14 shows an example of this: we set $N_s \in [50, 50]$ (so it can take exactly one value, 50), and for the dataset shown in Figure 14(a), we visualize the sampling distribution when the model is a DT of $depth = 2$ in Figure 14(b) vs when $depth = 4$ in Figure 14(c). The dataset is balanced, and the oracle used is a GBM.

We see the following interesting patterns: (a) at $depth = 2$, the DT picks points from both regions where $label = 1$, but the larger region shows higher density. This is possibly because owing to its limited capacity, the model is able to effectively parameterize only one region, and therefore it prioritizes correct classification of points around the larger region, (b) at $depth = 4$, we see increased sampling density in the smaller region with $label = 1$ as well.

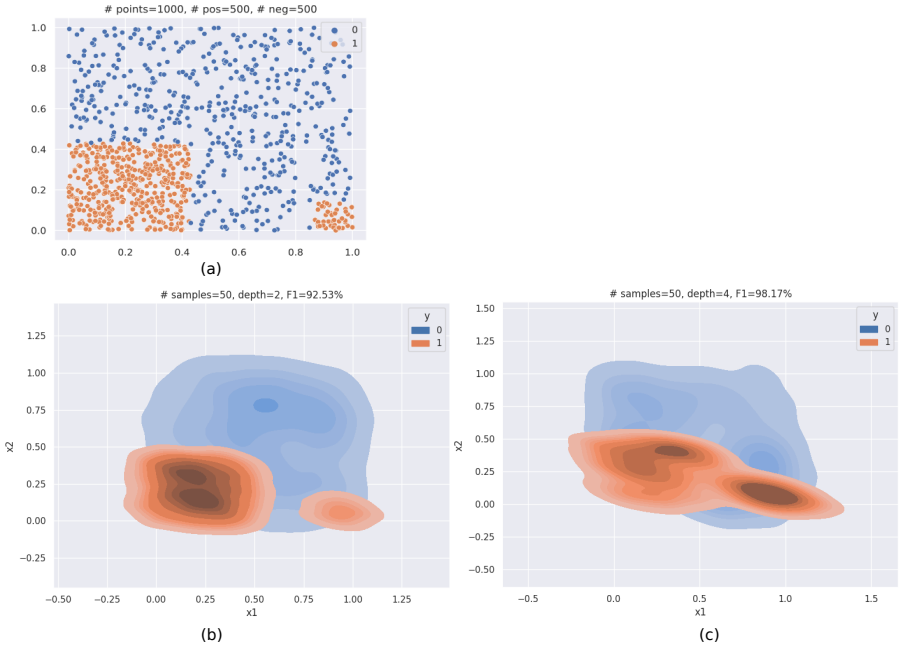


Figure 14: Our technique might be used to identify the optimal sample of a given size. (a) shows the original dataset. (b) and (c) visualize the learned distribution of points, using a KDE, for DTs with $depth = 2$ and $depth = 4$ respectively, for a sample size of 50. NOTE: the connection shown in (c), between the two originally disjoint regions with $label = 1$, is an artifact of the KDE.

4.3.3 Vector Model Size

Although we have been using a scalar notion of model size - depth for DT, number of terms for LPM, number of trees for a GBM - Algorithm 2 doesn't restrict us from using a vector-valued model size η. For example, in the case of GBMs, we may consider the notion of model size $\eta = [max\_depth, num\_trees]$, where the quantities respectively denote the maximum depth allowed for each constituent DT in a GBM, and the number of DTs in the GBM. In Figure 15 we show how improvements for GBMs vary when $1 \leq max\_depth \leq 5$ (x-axis) and $1 \leq num\_trees \leq 5$ (y-axis); the oracle used is a GBM as well (unconstrained in size), and results for these datasets are shown: (a) `higgs` (b) `cod-rna` (c) `senseit-sei` and (d) `senseit-aco`. The improvements are averaged over three runs. We observe the familiar pattern that as model sizes increase, in terms of both *max_depth* and *num_trees*, improvements decrease.

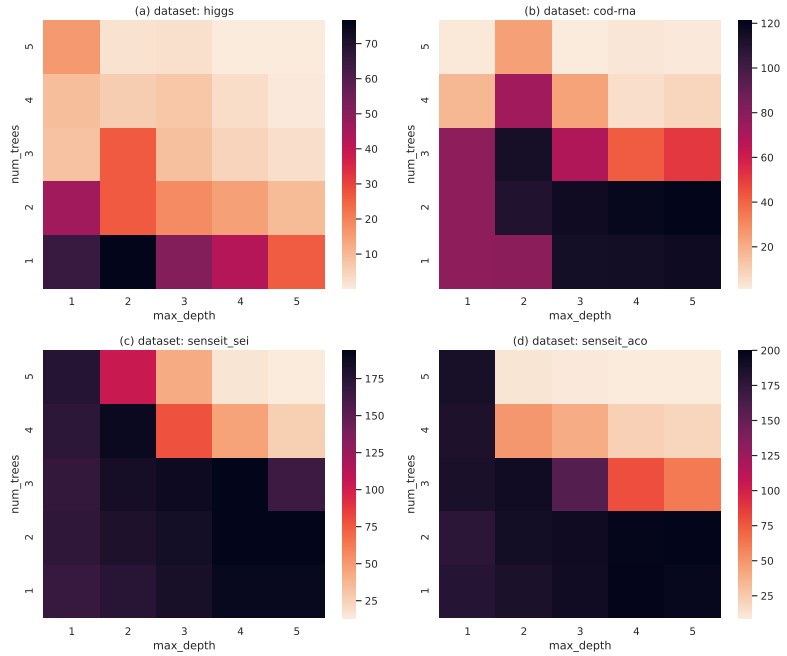


Figure 15: Improvements in test $F1$-macro for multiple datasets for different sizes of GBM models are shown. Here, model size is the combination of *max_depth* and *number of trees* in the GBM model. Greater improvements are seen at lower sizes.

4.4 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 4.1.6). Sometimes these may be $> 100\%$. For model sizes beyond a point, we observe $\delta F1 \approx 0$.

2. The results in Section 4.2 strongly indicate 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 . We believe this is an important result, especially given that this is true for the supervised version of uncertainty sampling, which is significantly more powerful than standard uncertainty sampling.

3. Our approach produces better accuracy, in general, compared to both supervised uncertainty sampling and the density tree based approach.

   The results in Table 3 and Table 4 from Section 4.2.3 are summarized in Table 5. Recall that the combination **OVERALL** + **ANY** averages over datasets and oracles; Table 5 shows these summary statistics.

Table 5: 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 observe that density trees are more competitive to our technique than supervised uncertainty sampling: smaller $\overline{SDI}$ and *pct_better* scores. This is to be expected since the density tree based approach is capable of learning flexible distributions over the input space.

4. A remarkable fact of practical value is that we *don't tune the parameters* Φ for a specific problem. The value ranges for these are fixed across tasks, with only the iteration budget T being changed - as described in Section 4.1.5. This highlights another strength of the technique: Φ need not be tuned for obtaining meaningful improvements, as long as it admits a broad enough set of uncertainty distributions.

5. Section 4.3 showcases the generality of the proposed technique: we successfully used it with differing feature spaces across the oracle and the interpretable model, to identify the optimal training sample for a given size, and with vector valued model sizes. These applications considerably broaden the impact of our work.

Importantly, the various positive results from this section should be seen as representative of the proposed *framework*, and not just our *implementation*. In other words, these results establish a lower bound for the outcomes, because they may be potentially improved by using different components within the larger framework, e.g., by using a different optimizer from among the ones discussed in Feurer and Hutter (2019) or Turner et al. (2021) .

Table 6: 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 experiments in Section 4, while "low" signifies lower values of parameters. Highlighted cells indicate positve effect of flattening.

		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

5. 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: does flattening (Section 3.6) help? Table 6 contrasts *improved* $F1$ scores obtained with (rows denoted as "original") and without (denoted "flattened") flattening the uncertainty distribution. This is shown for the datasets `Sensorless` and `covtype.binary`, for model $size \in \{1, 2, 3\}$, with $model = LPM$ and $oracle = GBM$. Two different parameter settings are used: (a) Setting 1 is what we have used in the experiments in Section 4: maximum allowed $Beta$ components are 500 and $scale = 10000$ (b) Setting 2 looks at much lower values of these parameters where maximum allowed components is 50 and $scale = 10$. The scores presented are the average over three trials.

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

   Recall we had noted in Figure 3 that the datasets `Sensorless` and `covtype.binary` have non-smooth and smooth uncertainty distributions respectively. The observations in Table 6 align well with the expectation that `Sensorless` is positively affected by the transformation, while results for `covtype.binary` remain mostly unchanged.

   Based on these tests, we hypothesize that for non-smooth uncertainty distributions, flattening makes our technique robust across parameter settings. It does not affect smooth distributions in a significant way. Of course, rigorous and extensive tests are required to conclusively establish this effect.

2. **Alternative Parameterization**: Instead of using shape variables $\{a, b, a', b'\}$ to characterize the IBMM (Section 3.4), which lie in the interval $(0, \infty)$, one might wonder if its simpler to parameterize based on the mean, $\mu \in [0, 1]$ (bounded by the

range of uncertainty values), and standard deviation, $\sigma \in [0, 0.5]$ (also bounded; this range is a property of the *Beta* distribution). While this is appealing, we need to consider that unlike the *Normal* distribution, μ and σ are not independent for a *Beta* distribution. For instance, $\mu \to 1 \implies \sigma \to 0$. The optimization would need to account for this dependence, and we would lose our current convenience of using only box constraints. The scatter plot in Figure 16 marks the different combinations of μ and σ for which valid *Beta* distributions exist.

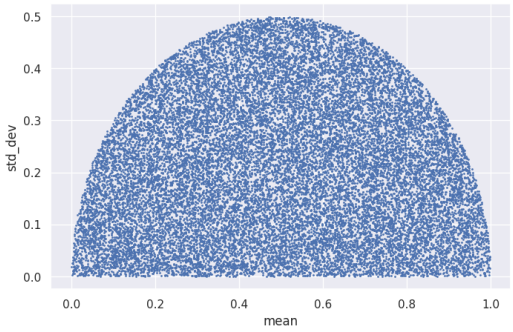


Figure 16: Blue dots indicate a valid *Beta* distribution exists for the corresponding mean and standard deviation values.

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 5, Figure 24) plots emphasize this use-case: they're a visual 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 17 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.

   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, for the respective notions of size we use here - this may also be seen from the plots in Figure 4, where the improvements for DTs decrease faster, with growing model size, than those for $LPMs$.

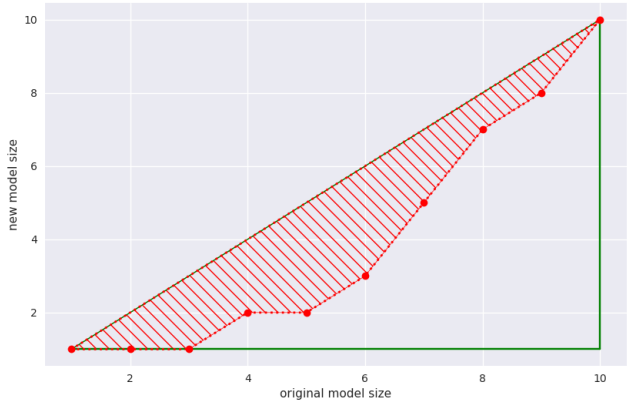


Figure 17: Compaction Index

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:

$$accuracy(M_{\mathcal{I}p\eta}, p) \leq accuracy(M_{\mathcal{I}q\eta}, p) \leq accuracy(M_{\mathcal{O}p*}, p) \tag{12}$$

We empirically show this to be true now. In Figure 18, 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:

$$\Delta F1 = \frac{accuracy(M_{\mathcal{I}q\eta}, p) - accuracy(M_{\mathcal{O}p*}, p)}{accuracy(M_{\mathcal{O}p*}, p)} \tag{13}$$

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

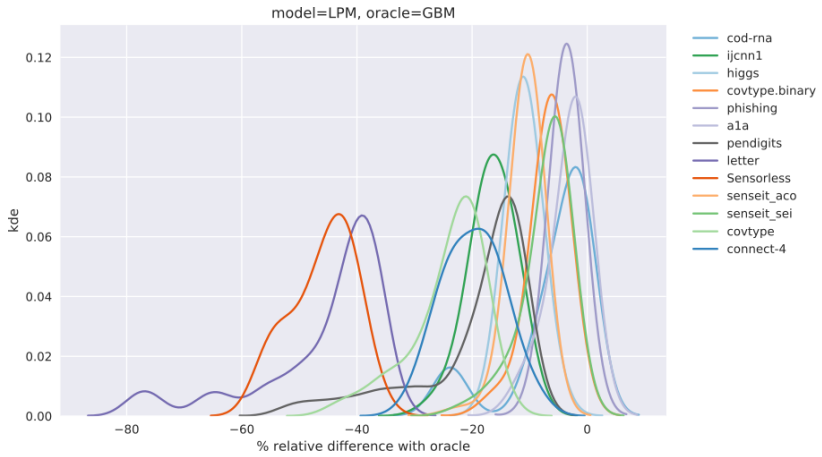


Figure 18: Distribution of %age accuracy difference from the oracle accuracy.

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 18 that all relative differences are at most 0 (there is some spillover to the right of 0 owing to the use of KDEs for visualization).

For precise numbers, we look at Table 7, 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 26 for plots for other model-oracle combinations.

Table 7: 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

6. Future Work

The results from our experiments suggest multiple promising avenues for future research. We list some of them here:

1. We might think of our technique as learning a sampling distribution $p'(x_i)$ indirectly on the input space: via $p(u_{M_O}(x_i))$, the distribution over uncertainty values produced by the oracle. An alternative view might be to directly learn instance weights w_i instead, where $w_i = p'(x_i)$. This approach clearly suffers from challenges in scaling - there are as many weights as training instances. However, recent work suggests that for *differentiable* model losses, this problem might be efficiently solved by formulating it as a *bi-level optimization* problem (Pedregosa, 2016; Lorraine, Vicol, & Duvenaud, 2020); which makes this a feasible direction to explore. The expected benefit is this might be faster[16], at least for moderately sized datasets.

   This approach brings its own challenges that future work would need to consider: (a) since gradient information is required, the loss function must be known; therefore this approach is not model-agnostic (b) even if an automatic differentiation framework is used, such as *JAX* (Bradbury et al., 2018), to generalize to unseen loss functions, model families like DTs remain out of scope since their loss isn't differentiable.

   NOTE that this aspect distinguishes our method from KD: beyond the uncertainty scores from the oracle, we do not require any additional information, e.g., "dark knowledge" (Hinton, Vinyals, & Dean, 2015b; Korattikara, Rathod, Murphy, & Welling, 2015).

16. Pedregosa (2016) compares this approach against BO for the task of hyperparameter tuning; these numbers should be assumed to be indicative only, since the BO algorithm used is not TPE.

2. The standard way to evaluate active learning algorithms is to evaluate model accuracy against the number of labeled training data instances. It is interesting to consider an alternative approach: for a given budget of labeled instances, measure the divergence between the sampling distribution our method learns (as in Section 4.3.2) and the one that an active learner proposes for labeling. Such an analysis is insightful since it can indicate precisely which points an active learner is *supposed* to label.

3. We noted that improvements from our technique diminish as model size grows (Section 4.1.6). For larger model sizes, a possible direction to explore might be to "chain" together multiple small models. This is similar to gradient boosting, and it would be especially informative to compare the two approaches.

4. An obvious question to ask is if our observations around the impact of training distribution on accuracy may be theoretically explained. There is some recent work in the area of KD that might serve as fruitful starting points: (a) Dao, Kamath, Syrgkanis, and Mackey (2021) provide theoretical tools to analyze distillation by treating it as a semi-parameteric inference problem (b) Menon, Rawat, Kumar, Reddi, and Kim (2021) propose a connection between the effectiveness of a teacher and its ability to approximate Bayes class-probabilities. (c) study of sample re-weighting on the effectiveness of distillation (Zhang, Hu, Qin, Xu, & Wang, 2021; Lu et al., 2021).

5. It would also be interesting to explore the connection between the sample of a given size our method finds (Section 4.3.2) and the *data Shapley value* (Ghorbani & Zou, 2019): a per-instance value quantifying the contribution of an instance to predictor accuracy. Some questions of interest are: (a) does an instance that has a high sampling probability across a range of sample sizes, per our method, also receive a high data Shapley value? (b) if there is indeed a correspondence between the two techniques, what algorithmic ideas may be borrowed from one technique to another?

7. Conclusion

In this paper we introduced a novel technique to learn an interpretable model, that reduces the trade-off between model size and accuracy. The practical implication of this work is that instead of picking an interpretable model family based on accuracy, one may use our method to construct accurate models for their preferred model family.

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 multiple favorable properties: (a) the optimization uses a fixed set of seven variables, irrespective of the dimensionality of the data (b) a reasonable choice of the search space produces good results across datasets (c) the technique is model-agnostic wrt both the interpretable and oracle models (d) it may be used even when the feature spaces of the interpretable model and oracle are different (e) its a framework, which leaves open the possibility of improving upon it. We have also shown some additional interesting applications for our technique.

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

This work also presents some intriguing deeper findings : (a) train and test distributions need not be identical for optimal learning (b) our observations point to a "small model effect": this difference in distributions exists for small model sizes, and it is in this model size regime that we observe most improvements.

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. Appendix A

A.1 Supervised Uncertainty Sampling

Algorithm 3: 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 $\quad I_U \leftarrow$ set of top b entries from $I_{remaining}$, based on $u_{M_\mathcal{O}}(x_i), i \in I_{remaining}$
7 $\quad I_{remaining} \leftarrow I_{remaining} - I_U$
8 $\quad I_{current} \leftarrow I_{current} \cup I_U$
9 $\quad D_t \leftarrow \{D_{train,i} | i \in I_{current}\}$
10 $\quad M_t \leftarrow train_{\mathcal{I},g}(D_t, \eta)$
11 $\quad s_t \leftarrow accuracy(M_t, D_{val})$
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^*

In Algorithm 3:

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 3, 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 and our model has access to reliable uncertainty scores; this avoids the sample bias discussed in Section A.2 (visualized in Figure 19).

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}.

A.2 Simple Uncertainty Sampling in Active Learning

In active learning, the 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:

1. Construct a classifier on the current labelled subset.

2. 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.

Although intuitive, this approach was shown to suffer from sample bias (Dasgupta & Hsu, 2008; Dasgupta, 2011). We illustrate this in Figure 19.

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 belief that C is the only class boundary. 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.

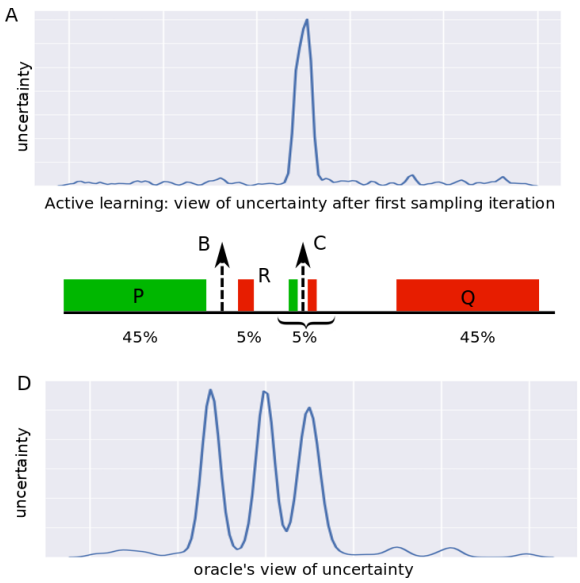


Figure 19: Uncertainty estimates from classifier after first iteration. Smaller boundaries are missed since the sample predominantly comes from P and Q.

This problem does not affect us since the oracle has access to the complete training data. Plot D shows the uncertainty distribution as per the oracle. However, as our results show, even with its complete view of uncertainty landscape, simple uncertainty sampling is not optimal.

A.3 Comparison of Uncertainty Distributions

It is instructive to look at some specific adjusted IBMMs in the context of the relative performance of techniques. Figure 20 shows the plots from Figure 9 annotated with SDI scores. These are for $LPMs$ using GBM as the oracle.

The top row - (a), (b), (c) in Figure 20 - 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 20(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.

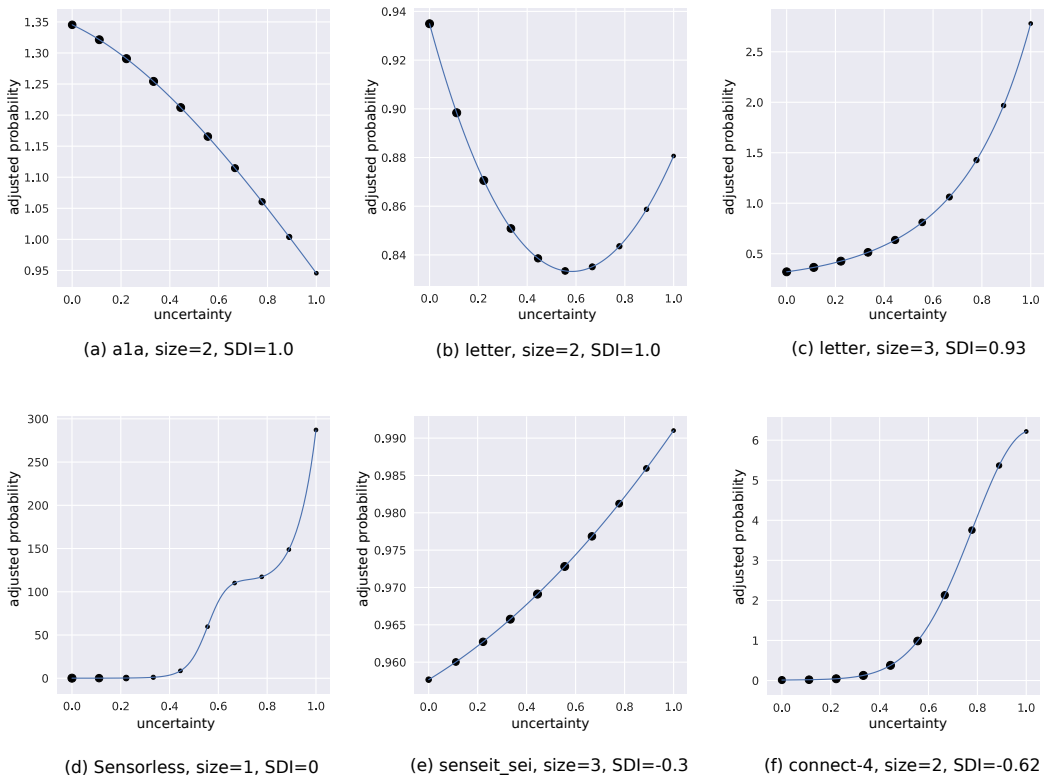


Figure 20: Examples of adjusted distributions are shown, and the *SDI* scores, measured against supervised uncertainty sampling, are mentioned. The plots in the top row are the same as in Figure 9. The top row - (a), (b), (c) - shows instances where our technique performed relatively better, and the bottom row shows cases where uncertainty sampling was competitive - (d) - or better - (e), (f).

A.4 Uncertainty Metrics

Some other popular uncertainty metrics are:

1. **Least confident**: we calculate the extent of uncertainty w.r.t. the class we are most confident about:
$$u_M(x) = 1 - \max_{y_i \in \{1,2,\ldots,C\}} M(y_i|x) \tag{14}$$
Here, we have C classes, and $M(y_i|x)$ is the probability score produced by the model[17].

2. **Entropy**: this is the standard Shannon entropy measure calculated over class prediction confidences:
$$u_M(x) = \sum_{y_i \in \{1,2,\ldots,C\}} -M(y_i|x) \log M(y_i|x) \tag{15}$$

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 1) we may never reach this maximum.

Fig 21 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.

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

17. 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, $x^* = \arg\min_x [\max_{y_i \in \{1,2,\ldots,C\}} M(y_i|x)]$, which may be considered to be the instance with the "least most confident label".

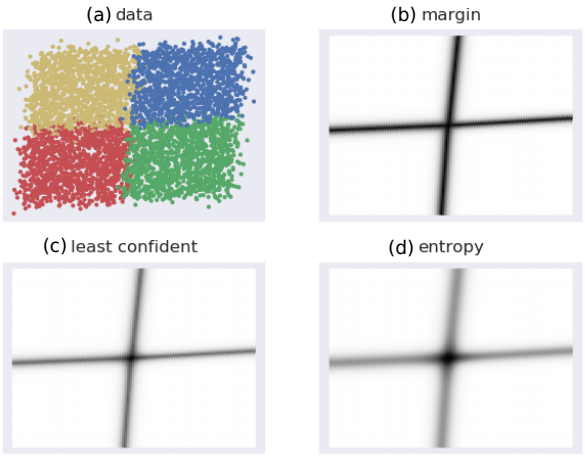


Figure 21: Visualizations of different uncertainty metrics. (a) shows a 4-label dataset on which linear SVM is learned. (b), (c), (d) visualize uncertainty scores based on different metrics, as per the linear SVM, where darker shades imply higher scores.

A.5 Flattening of the Uncertainty Distribution

Algorithm 4 details the flattening process mentioned in Section 3.6.

In lines 11 and 12 of Algorithm 4, we offset bin boundary limits by a small positive value δ to avoid assignment conflicts across adjacent bins at their boundaries.

This algorithm produces a transformation that looks like the uniform distribution. We prefer the likeness to the uniform distribution since it makes all regions within the interval $[0, 1]$ equally easy to discover.

Algorithm 4: Flatten distribution of uncertainty scores $\{u(x_1), u(x_2), ..., u(x_N)\}$

Data: $\{u(x_1), u(x_2), ..., u(x_N)\}$, number of bins B
Result: $\{u'(x_1), u'(x_2), ..., u'(x_N)\}$
$bin\_size \leftarrow \lceil N/B \rceil, bin\_range \leftarrow 1/B$
$bin\_min \leftarrow [\,], bin\_max \leftarrow [\,]$
Let $sortedIndex(i) \in \{1, 2, ..., N\}$ be the index of $u(x_i)$ in the sequence of scores ordered by non-decreasing values.
for $j \leftarrow 1$ **to** B **do**
  $bin\_min[j] \leftarrow \min\{u(x_i) | i \in \{1, 2, ..., N\} \wedge sortedIndex(i) = j\}$
  $bin\_max[j] \leftarrow \max\{u(x_i) | i \in \{1, 2, ..., N\} \wedge sortedIndex(i) = j\}$
end
for $i \leftarrow 1$ **to** N **do**
  $j \leftarrow sortedIndex(i)$
  $bin\_num \leftarrow \lceil j/bin\_size \rceil$
  $boundary\_low \leftarrow (bin\_num - 1) \times bin\_range + \delta$
  $boundary\_high \leftarrow bin\_num \times bin\_range - \delta$
  $u'(x_i) \leftarrow low + \frac{u(x_i) - bin\_min[j]}{bin\_max[j] - bin\_min[j]} \times (boundary\_high - boundary\_low)$
end
return $\{u'(x_1), u'(x_2), ..., u'(x_N)\}$

A.6 Statistical Significance of Improvements

To assess the statistical significance of the improvements presented in Section 4.1.6, we perform the *one-sided* version of the paired *Wilcoxon signed-rank test*, where the pairs of scores $F1_{baseline}$ and $F1_{new}$ across datasets are tested for the following hypotheses:

- $\mathbf{H_0}$, null hypothesis: accuracies of models trained using the oracle are not better.
- $\mathbf{H_1}$, alternate hypothesis: accuracies of models trained using the oracle are better.

Since the extent of improvement may vary with model size, we perform the test separately for different ranges or bins of normalized model sizes, where the bin boundaries are decided by the *Freedman-Diaconis* rule (Freedman & Diaconis, 1981). Normalized model sizes are used for convenient comparison with Figure 4. *p-values* of the significance tests, for different model-oracle combinations, are shown in Figure 22. A dashed line at $p = 0.05$ is provided for reference. A high *p-value* strongly indicates $\mathbf{H_0}$, i.e., using the oracle is not better than the baseline.

We observe that the improvements from using an oracle are indeed significant for most model sizes and model-oracle combinations, when measured across multiple datasets. In the case of DTs, we also observe that evidence in favor of retaining $\mathbf{H_0}$ is high at larger model sizes; this aligns with our earlier observations.

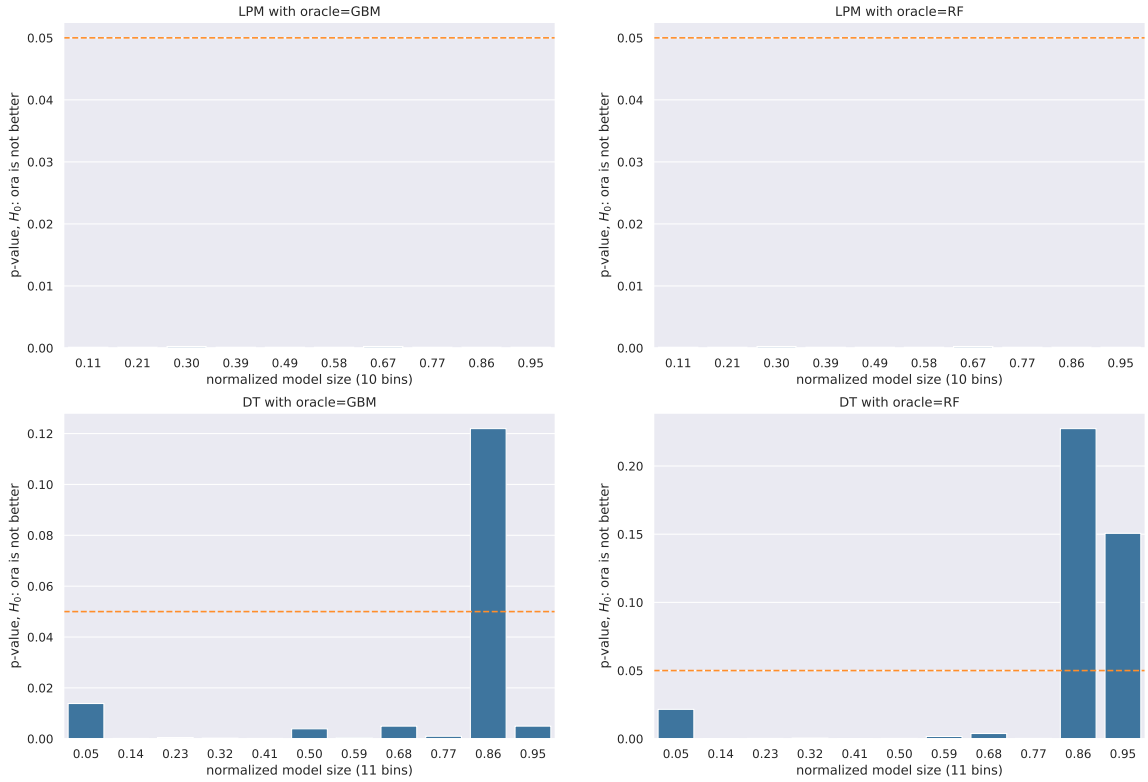


Figure 22: These plots show the *p-values* for the Wilcoxon signed-rank test, with the null hypothesis H_0: using the oracle does *not* produce better F1 test scores. The bin boundaries are selected using the *Freedman-Diaconis* rule (Freedman & Diaconis, 1981). Low *p-values* in most cases indicate that an oracle guided model is more accurate, especially at smaller model sizes. The significance level of 0.05 is shown with a dashed line for reference.

A.7 Uncertainty Distribution for DT

The uncertainty distributions learned when using a DT with different oracles are shown in Figure 23. 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 7 and Figure 8.

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 sampling across the entire range of uncertainty values, with slight/occasional preference for higher uncertainties.

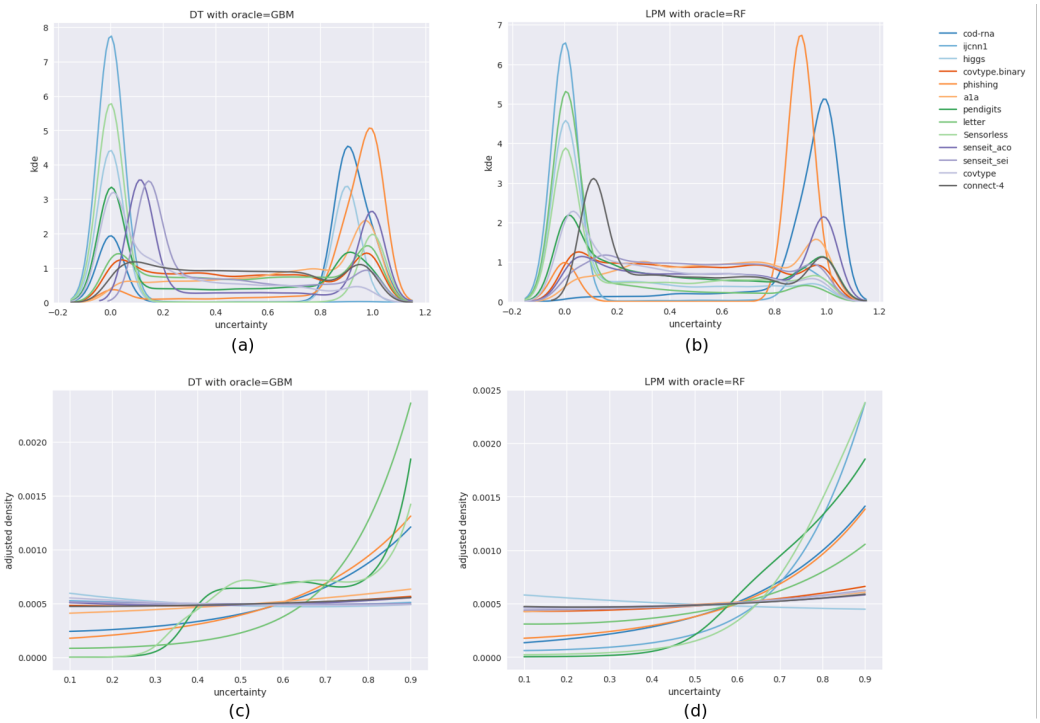


Figure 23: The aggregated IBMMs visualized when using a DT as our interpretable model. The top row shows the aggregated IBMMs for different oracles: GBM (left) and RF (right). The bottom row visualizes the IBMMs adjusted for the uncertainty distribution.

A.8 Compaction Profiles

Figure 24 shows the compaction profiles for all model-oracle combinations. These are discussed in Section 4.1.6, in reference to Figure 5.

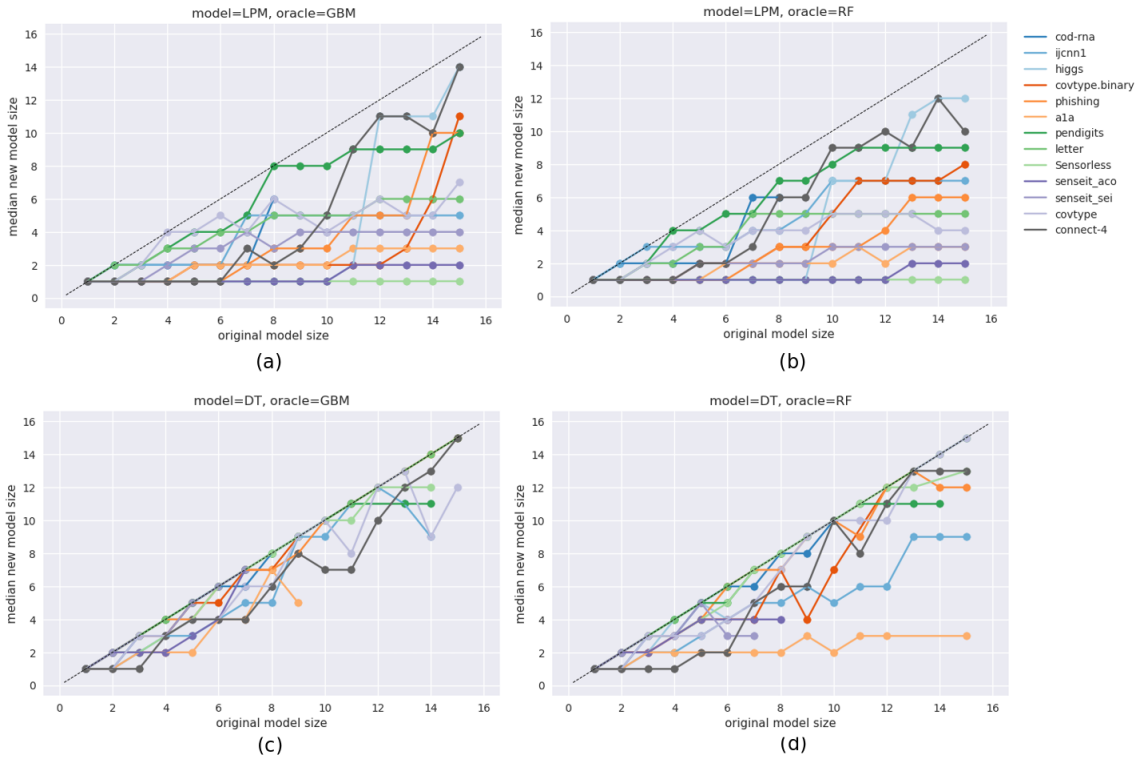


Figure 24: For different combinations of models and oracles: $\{LPM, DT\} \times \{GBM, RF\}$, these plots show the size of an improved model (y-axis), that may replace a traditionally trained model of a given size (x-axis). A model is considered as a replacement for another if its accuracy is at least as high as the latter.

A.9 IBMMs for Different Model Sizes

Figure 25 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 7, are indicative of the individual distributions in this Figure 25.

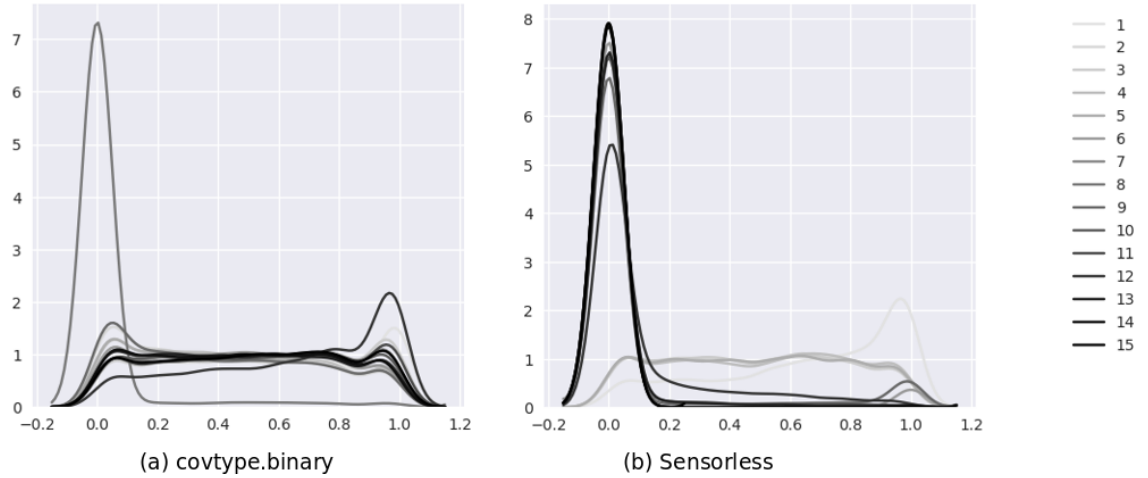


Figure 25: IBMM distributions for model sizes $\{1, 2, ..., 15\}$, for the datasets (a) `covtype.binary` and (b) `Sensorless`. These are for the combination of using LPM as the model with GBM as an oracle. Darker curves indicate higher model sizes.

A.10 Improvements Relative to Oracle

Some of the positive values we see in Figure 26 may be attributed to spillovers due to the *kde* fit. Their magnitudes and occurrences are typically small: these are detailed in Table 7.

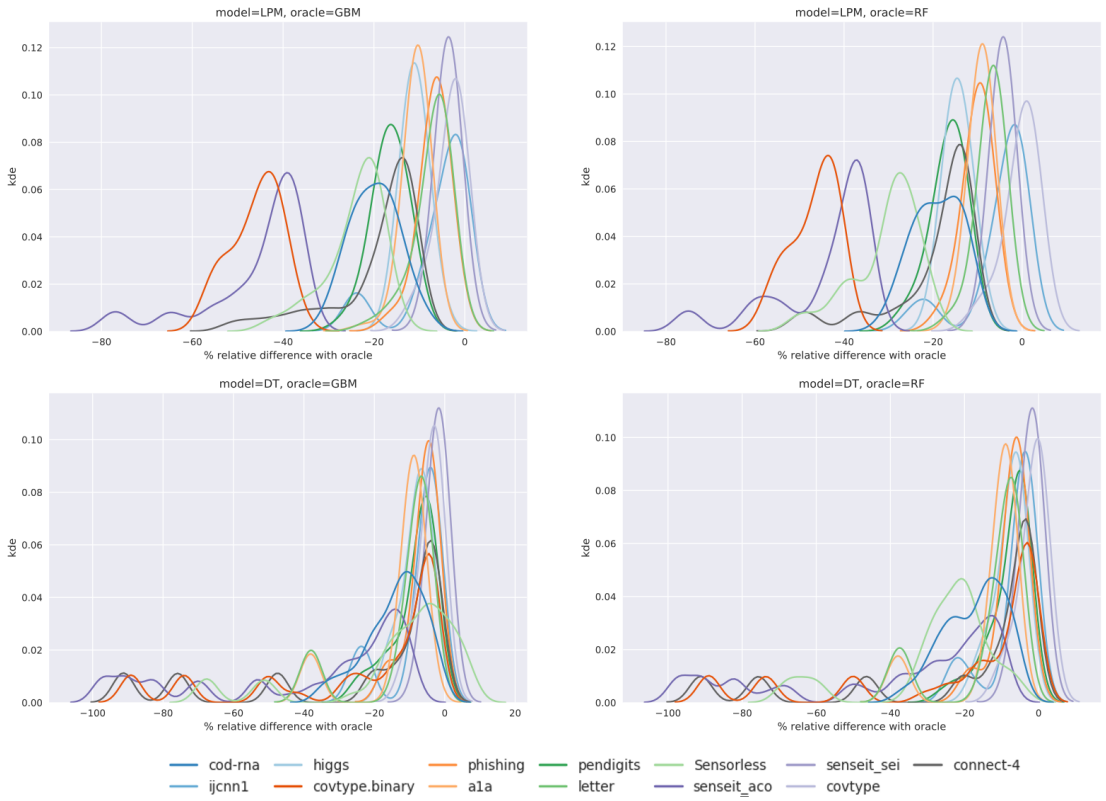


Figure 26: These plots show the distribution of the percentage relative difference of a model's improved score w.r.t. to the accuracy of the oracle it is trained with. We note that this quantity is almost always non-positive as claimed in Equations 2, 3 and 4.

A.11 Feature Selection for n-gram DT

For the experiments in Section 4.3.1, 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 27 shows how $\delta F1$ varies with k.

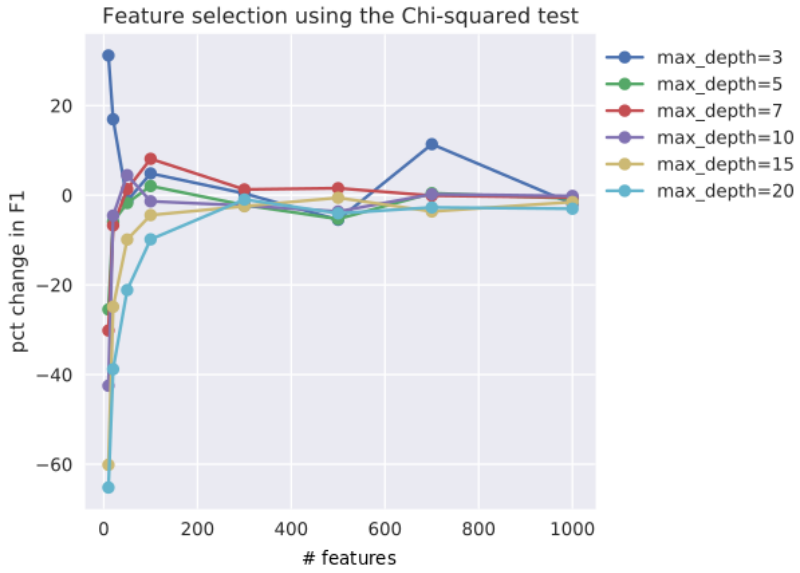


Figure 27: The relationship between $\delta F1$ and $k \leq 1000$. Each data point is an average over ten runs.

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, 09–15 Jun). Explaining deep neural networks with a polynomial time algorithm for shapley value approximation. In K. Chaudhuri & R. Salakhutdinov (Eds.), *Proceedings of the 36th international conference on machine learning* (Vol. 97, pp. 272–281). Long Beach, California, USA: PMLR. Retrieved from http://proceedings.mlr.press/v97/ancona19a.html

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* (pp. 35–44). New York, NY, USA: ACM. Retrieved from http://doi.acm.org/10.1145/3097983.3098047 doi: 10.1145/3097983.3098047

Angwin, J., Larson, J., Mattu, S., & Kirchner, L. (2016, May). *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. Retrieved from https://doi.org/10.1038/ncomms5308 doi: 10.1038/ncomms5308

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* (pp. 2546–2554). USA: Curran Associates Inc. Retrieved from http://dl.acm.org/citation.cfm?id=2986459.2986743

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* (pp. I-115–I-123). JMLR.org. Retrieved from http://dl.acm.org/citation.cfm?id=3042817.3042832

Bertsimas, D., Delarue, A., Jaillet, P., & Martin, S. (2019). The price of interpretability. *CoRR*, *abs/1907.03419*. Retrieved from http://arxiv.org/abs/1907.03419

Blackwell, D., & MacQueen, J. B. (1973, 03). Ferguson distributions via polya urn schemes. *Ann. Statist.*, *1*(2), 353–355. Retrieved from https://doi.org/10.1214/aos/1176342372 doi: 10.1214/aos/1176342372

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. Retrieved from http://www.jstor.org/stable/2984418

Bradbury, J., Frostig, R., Hawkins, P., Johnson, M. J., Leary, C., Maclaurin, D., ... Zhang, Q. (2018). *JAX: composable transformations of Python+NumPy programs* [Computer Software]. Retrieved from http://github.com/google/jax

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

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*.

Bucilă, C., Caruana, R., & Niculescu-Mizil, A. (2006). Model compression. In *Proceedings of the 12th acm sigkdd international conference on knowledge discovery and data mining* (p. 535–541). New York, NY, USA: Association for Computing Machinery. Retrieved from https://doi.org/10.1145/1150402.1150464 doi: 10.1145/1150402.1150464

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* (pp. 1721–1730). New York, NY, USA: ACM. Retrieved from http://doi.acm.org/10.1145/2783258.2788613 doi: 10.1145/2783258.2788613

Castellanos, S., & Nash, K. S. (2018, May). *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)

Chawla, N. V., Bowyer, K. W., Hall, L. O., & Kegelmeyer, W. P. (2002, June). Smote: Synthetic minority over-sampling technique. *J. Artif. Int. Res.*, *16*(1), 321–357.

Chen, J., Song, L., Wainwright, M. J., & Jordan, M. I. (2018). Learning to explain: An information-theoretic perspective on model interpretation. In J. G. Dy & A. Krause (Eds.), *Proceedings of the 35th international conference on machine learning, ICML 2018, stockholmsmässan, stockholm, sweden, july 10-15, 2018* (Vol. 80, pp. 882–891). PMLR. Retrieved from http://proceedings.mlr.press/v80/chen18j.html

Cho, K., van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., & Bengio, Y. (2014, October). 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. Retrieved from https://www.aclweb.org/anthology/D14-1179 doi: 10.3115/v1/D14-1179

Clarke, Y. D. (2019, Oct.). *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 T. G. Dietterich, S. Becker, & Z. Ghahramani (Eds.), *Advances in neural information processing systems 14* (pp. 633–640). MIT Press. Retrieved from http://papers.nips.cc/paper/1949-a-parallel-mixture-of-svms-for-very-large-scale-problems.pdf

Dai, W., Yang, Q., Xue, G.-R., & Yu, Y. (2007). Boosting for transfer learning. In *Proceedings of the 24th international conference on machine learning* (pp. 193–200). New York, NY, USA: ACM. Retrieved from http://doi.acm.org/10.1145/1273496.1273521 doi: 10.1145/1273496.1273521

Dai, Z., Yu, H., Low, B. K. H., & Jaillet, P. (2019, 09–15 Jun). Bayesian optimization meets Bayesian optimal stopping. In K. Chaudhuri & R. Salakhutdi-

nov (Eds.), *Proceedings of the 36th international conference on machine learning* (Vol. 97, pp. 1496–1506). Long Beach, California, USA: PMLR. Retrieved from http://proceedings.mlr.press/v97/dai19a.html

Dao, T., Kamath, G. M., Syrgkanis, V., & Mackey, L. (2021). Knowledge distillation as semiparametric inference. In *International conference on learning representations.*

Dasgupta, S. (2011, April). Two faces of active learning. *Theor. Comput. Sci.*, *412*(19), 1767–1781. Retrieved from http://dx.doi.org/10.1016/j.tcs.2010.12.054 doi: 10.1016/j.tcs.2010.12.054

Dasgupta, S., & Hsu, D. (2008). Hierarchical sampling for active learning. In *Proceedings of the 25th international conference on machine learning* (pp. 208–215). New York, NY, USA: ACM. Retrieved from http://doi.acm.org/10.1145/1390156.1390183 doi: 10.1145/1390156.1390183

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, March). 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, June). 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. Retrieved from https://www.aclweb.org/anthology/N19-1423 doi: 10.18653/v1/N19-1423

Dua, D., & Graff, C. (2017). *UCI machine learning repository.* Retrieved from http://archive.ics.uci.edu/ml

Duarte, M. F., & Hu, Y. H. (2004, July). Vehicle classification in distributed sensor networks. *J. Parallel Distrib. Comput.*, *64*(7), 826–838. Retrieved from https://doi.org/10.1016/j.jpdc.2004.03.020 doi: 10.1016/j.jpdc.2004.03.020

Efron, B., Hastie, T., Johnstone, I., & Tibshirani, R. (2004, 04). Least angle regression. *Ann. Statist.*, *32*(2), 407–499. Retrieved from https://doi.org/10.1214/009053604000000067 doi: 10.1214/009053604000000067

Feldman, J. (2000, 11). Minimization of boolean complexity in human concept learning. *Nature*, *407*, 630-3. doi: 10.1038/35036586

Feurer, M., & Hutter, F. (2019). Hyperparameter optimization. In F. Hutter, L. Kotthoff, & J. Vanschoren (Eds.), *Automated machine learning: Methods, systems, challenges* (pp. 3–33). Cham: Springer International Publishing. Retrieved from https://doi.org/10.1007/978-3-030-05318-5_1 doi: 10.1007/978-3-030-05318-5_1

Freedman, D., & Diaconis, P. (1981, Dec 01). On the histogram as a density estimator:l2 theory. *Zeitschrift für Wahrscheinlichkeitstheorie und Verwandte Gebiete*, *57*(4), 453-476. Retrieved from https://doi.org/10.1007/BF01025868 doi: 10.1007/BF01025868

Freitas, A. A. (2014, March). Comprehensible classification models: A position paper. *SIGKDD Explor. Newsl.*, *15*(1), 1–10. Retrieved from https://doi.org/10.1145/2594473.2594475 doi: 10.1145/2594473.2594475

Ghorbani, A., & Zou, J. (2019, 09–15 Jun). Data shapley: Equitable valuation of data

for machine learning. In K. Chaudhuri & R. Salakhutdinov (Eds.), *Proceedings of the 36th international conference on machine learning* (Vol. 97, pp. 2242–2251). PMLR. Retrieved from https://proceedings.mlr.press/v97/ghorbani19c.html

Ghose, A. (2021). *compactem: build accurate small models* [Computer Software]. https://compactem.readthedocs.io. Retrieved from https://compactem.readthedocs.io

Ghose, A., & Ravindran, B. (2020). Interpretability with accurate small models. *Frontiers in Artificial Intelligence*, *3*, 3. Retrieved from https://www.frontiersin.org/article/10.3389/frai.2020.00003 doi: 10.3389/frai.2020.00003

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

Gou, J., Yu, B., Maybank, S. J., & Tao, D. (2021, Jun 01). Knowledge distillation: A survey. *International Journal of Computer Vision*, *129*(6), 1789-1819. Retrieved from https://doi.org/10.1007/s11263-021-01453-z doi: 10.1007/s11263-021-01453-z

Graves, A. (2013). Generating sequences with recurrent neural networks. *CoRR*, *abs/1308.0850*. Retrieved from http://arxiv.org/abs/1308.0850

Grill, J.-B., Valko, M., Munos, R., & Munos, R. (2015). Black-box optimization of noisy functions with unknown smoothness. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, & R. Garnett (Eds.), *Advances in neural information processing systems 28* (pp. 667–675). Curran Associates, Inc. Retrieved from http://papers.nips.cc/paper/5721-black-box-optimization-of-noisy-functions-with-unknown-smoothness.pdf

Gunning, D. (2016, July). *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* (p. 1321–1330). JMLR.org.

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

Hansen, N., & Ostermeier, A. (2001, June). Completely derandomized self-adaptation in evolution strategies. *Evol. Comput.*, *9*(2), 159–195. Retrieved from http://dx.doi.org/10.1162/106365601750190398 doi: 10.1162/106365601750190398

He, H., Bai, Y., Garcia, E. A., & Li, S. (2008). Adasyn: Adaptive synthetic sampling approach for imbalanced learning. In *2008 ieee international joint conference on neural networks (ieee world congress on computational intelligence)* (p. 1322-1328). doi: 10.1109/IJCNN.2008.4633969

Herman, B. (2017). *The promise and peril of human evaluation for model interpretability* (Vol. abs/1711.07414). Retrieved from http://arxiv.org/abs/1711.07414 (Presented at NIPS 2017 Symposium on Interpretable Machine Learning. Available at: https://arxiv.org/abs/1711.09889v3)

Hinton, G., Vinyals, O., & Dean, J. (2015a). Distilling the knowledge in a neural network. In *Nips deep learning and representation learning workshop.* Retrieved from http://arxiv.org/abs/1503.02531

Hinton, G., Vinyals, O., & Dean, J. (2015b). Distilling the knowledge in a neural network. In *Nips deep learning and representation learning workshop.* Retrieved from http://

arxiv.org/abs/1503.02531

Howard, J., & Ruder, S. (2018). Universal language model fine-tuning for text classification. In *Acl*. Association for Computational Linguistics. Retrieved from http://arxiv.org/abs/1801.06146

Hsu, C.-W., & Lin, C.-J. (2002, 02). 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. doi: 10.1109/72.991427

Hu, X., Rudin, C., & Seltzer, M. (2019). Optimal sparse decision trees. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, & R. Garnett (Eds.), *Advances in neural information processing systems 32* (pp. 7265–7273). Curran Associates, Inc. Retrieved from http://papers.nips.cc/paper/8947-optimal-sparse-decision-trees.pdf

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* (pp. 507–523). Berlin, Heidelberg: Springer-Verlag. Retrieved from http://dx.doi.org/10.1007/978-3-642-25566-3_40 doi: 10.1007/978-3-642-25566-3_40

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* (p. 448–456). JMLR.org.

Japkowicz, N., & Stephen, S. (2002, October). The class imbalance problem: A systematic study. *Intell. Data Anal.*, *6*(5), 429–449. Retrieved from http://dl.acm.org/citation.cfm?id=1293951.1293954

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* (p. 43–50). New York, NY, USA: Association for Computing Machinery. Retrieved from https://doi.org/10.1145/2959100.2959134 doi: 10.1145/2959100.2959134

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* (pp. 219–228). Washington, DC, USA: IEEE Computer Society. Retrieved from https://doi.org/10.1109/ICDM.2009.9 doi: 10.1109/ICDM.2009.9

Ke, G., Meng, Q., Finley, T., Wang, T., Chen, W., Ma, W., ... Liu, T.-Y. (2017). Lightgbm: A highly efficient gradient boosting decision tree. In *Proceedings of the 31st international conference on neural information processing systems* (pp. 3149–3157). USA: Curran Associates Inc. Retrieved from http://dl.acm.org/citation.cfm?id=3294996.3295074

Koh, P. W., & Liang, P. (2017, 06–11 Aug). Understanding black-box predictions via influence functions. In D. Precup & Y. W. Teh (Eds.), *Proceedings of the 34th international conference on machine learning* (Vol. 70, pp. 1885–1894). International Convention Centre, Sydney, Australia: PMLR. Retrieved from http://proceedings.mlr.press/v70/koh17a.html

Korattikara, A., Rathod, V., Murphy, K., & Welling, M. (2015). Bayesian dark knowledge. In *Nips'15 proceedings of the 28th international conference on neural information processing systems - volume 2* (Vol. 28, pp. 3438–3446).

Kulesza, T., Stumpf, S., Burnett, M., Yang, S., Kwan, I., & Wong, W.-K. (2013). Too much, too little, or just right? ways explanations impact end users' mental models. In *2013 ieee symposium on visual languages and human centric computing* (p. 3-10). doi: 10.1109/VLHCC.2013.6645235

Lage, I., Chen, E., He, J., Narayanan, M., Kim, B., Gershman, S. J., & Doshi-Velez, F. (2019, Oct.). Human evaluation of models built for interpretability. *Proceedings of the AAAI Conference on Human Computation and Crowdsourcing*, *7*(1), 59-67. Retrieved from https://ojs.aaai.org/index.php/HCOMP/article/view/5280

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* (pp. 1675–1684). New York, NY, USA: ACM. Retrieved from http://doi.acm.org/10.1145/2939672.2939874 doi: 10.1145/2939672.2939874

Larson, J., Mattu, S., Kirchner, L., & Angwin, J. (2016, May). *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* (pp. 3–12). New York, NY, USA: Springer-Verlag New York, Inc. Retrieved from http://dl.acm.org/citation.cfm?id=188490.188495

Liao, X., Xue, Y., & Carin, L. (2005). Logistic regression with an auxiliary data source. In *Proceedings of the 22nd international conference on machine learning* (pp. 505–512). New York, NY, USA: ACM. Retrieved from http://doi.acm.org/10.1145/1102351.1102415 doi: 10.1145/1102351.1102415

Lim, M., & Hastie, T. (2015). Learning interactions via hierarchical group-lasso regularization. *J Comput Graph Stat*, *24*(3), 627-654. Retrieved from https://www.ncbi.nlm.nih.gov/pubmed/26759522 (26759522[pmid]) doi: 10.1080/10618600.2014.938812

Lipton, Z. C. (2018, June). The mythos of model interpretability. *Queue*, *16*(3), 30:31–30:57. Retrieved from http://doi.acm.org/10.1145/3236386.3241340 doi: 10.1145/3236386.3241340

Lorraine, J., Vicol, P., & Duvenaud, D. (2020, 26–28 Aug). Optimizing millions of hyperparameters by implicit differentiation. In S. Chiappa & R. Calandra (Eds.), *Proceedings of the twenty third international conference on artificial intelligence and statistics* (Vol. 108, pp. 1540–1552). PMLR. Retrieved from https://proceedings.mlr.press/v108/lorraine20a.html

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* (pp. 623–631). New York, NY, USA: ACM. Retrieved from http://doi.acm.org/10.1145/2487575.2487579 doi: 10.1145/2487575.2487579

Lu, P., Ghaddar, A., Rashid, A., Rezagholizadeh, M., Ghodsi, A., & Langlais, P. (2021,

November). RW-KD: Sample-wise loss terms re-weighting for knowledge distillation. In *Findings of the association for computational linguistics: Emnlp 2021* (pp. 3145–3152). Punta Cana, Dominican Republic: Association for Computational Linguistics. Retrieved from https://aclanthology.org/2021.findings-emnlp.270

Lundberg, S. M., & Lee, S.-I. (2017). A unified approach to interpreting model predictions. In I. Guyon et al. (Eds.), *Advances in neural information processing systems 30* (pp. 4765–4774). Curran Associates, Inc. Retrieved from http://papers.nips.cc/paper/7062-a-unified-approach-to-interpreting-model-predictions.pdf

Malkomes, G., & Garnett, R. (2018). Automating bayesian optimization with bayesian optimization. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, & R. Garnett (Eds.), *Advances in neural information processing systems 31* (pp. 5984–5994). Curran Associates, Inc. Retrieved from http://papers.nips.cc/paper/7838-automating-bayesian-optimization-with-bayesian-optimization.pdf

Menon, A. K., Rawat, A. S., Kumar, S., Reddi, S., & Kim, S. (2021). A statistical perspective on distillation. In *International conference on machine learning (icml) 2021.*

Michie, D., Spiegelhalter, D. J., Taylor, C. C., & Campbell, J. (Eds.). (1995). *Machine learning, neural and statistical classification.* USA: Ellis Horwood.

Mihalkova, L., & Mooney, R. (2006, June). Transfer learning with markov logic networks. In *Proceedings of the icml-06 workshop on structural knowledge transfer for machine learning.* Pittsburgh, PA. Retrieved from http://www.cs.utexas.edu/users/ai-lab/?mihalkova:icml-wkshp06

Mohammad, R. M., Thabtah, F., & McCluskey, L. (2012, Dec). An assessment of features related to phishing websites using an automated technique. In *2012 international conference for internet technology and secured transactions* (p. 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. doi: 10.1093/esr/jcp006

Murdoch, W. J., Singh, C., Kumbier, K., Abbasi-Asl, R., & Yu, B. (2019). Definitions, methods, and applications in interpretable machine learning. *Proceedings of the National Academy of Sciences*, *116*(44), 22071–22080. Retrieved from https://www.pnas.org/content/116/44/22071 doi: 10.1073/pnas.1900654116

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. Retrieved from https://onlinelibrary.wiley.com/doi/abs/10.1002/sim.2666 doi: 10.1002/sim.2666

Pan, S. J., & Yang, Q. (2010, Oct). A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering*, *22*(10), 1345-1359. doi: 10.1109/TKDE.2009.191

Paschke, F., Bayer, C., Bator, M., Mönks, U., Dicks, A., Enge-Rosenblatt, O., & Lohweg, V. (2013, 12). Sensorlose zustandsüberwachung an synchronmotoren. In *Proceedings of computational intelligence workshop.*

Pedregosa, F. (2016). Hyperparameter optimization with approximate gradient. In *Proceedings of the 33rd international conference on international conference on machine learning - volume 48* (p. 737–746). JMLR.org.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., ... Duch-

esnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, *12*, 2825–2830.

Platt, J. (1998, January). 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 ed.). MIT Press. Retrieved from https://www.microsoft.com/en-us/research/publication/fast-training-of-support-vector-machines-using-sequential-minimal-optimization/

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.

Poursabzi-Sangdeh, F., Goldstein, D., Hofman, J., Wortman Vaughan, J., & Wallach, H. (2021, May). Manipulating and measuring model interpretability. In *Chi 2021*. Retrieved from https://www.microsoft.com/en-us/research/publication/manipulating-and-measuring-model-interpretability/

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

Pröllochs, N., Feuerriegel, S., & Neumann, D. (2019, June). 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. Retrieved from https://www.aclweb.org/anthology/N19-1038 doi: 10.18653/v1/N19-1038

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

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* (pp. 554–560). Cambridge, MA, USA: MIT Press. Retrieved from http://dl.acm.org/citation.cfm?id=3009657.3009736

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* (pp. 1135–1144). New York, NY, USA: ACM. Retrieved from http://doi.acm.org/10.1145/2939672.2939778 doi: 10.1145/2939672.2939778

Ribeiro, M. T., Singh, S., & Guestrin, C. (2018). Anchors: High-precision model-agnostic explanations.. Retrieved from https://aaai.org/ocs/index.php/AAAI/AAAI18/paper/view/16982

Romero, A., Ballas, N., Kahou, S. E., Chassang, A., Gatta, C., & Bengio, Y. (2015). Fitnets: Hints for thin deep nets. In Y. Bengio & Y. LeCun (Eds.), *3rd international conference on learning representations, ICLR 2015, san diego, ca, usa, may 7-9, 2015,*

conference track proceedings. Retrieved from http://arxiv.org/abs/1412.6550

Sanh, V., Debut, L., Chaumond, J., & Wolf, T. (2019). Distilbert, a distilled version of BERT: smaller, faster, cheaper and lighter. *CoRR*, *abs/1910.01108*. Retrieved from http://arxiv.org/abs/1910.01108

Santhiappan, S., Chelladurai, J., & Ravindran, B. (2018). A novel topic modeling based weighting framework for class imbalance learning. In *Proceedings of the acm india joint international conference on data science and management of data* (p. 20–29). New York, NY, USA: Association for Computing Machinery. Retrieved from https://doi.org/10.1145/3152494.3152496 doi: 10.1145/3152494.3152496

Santurkar, S., Tsipras, D., Ilyas, A., & Madry, A. (2018). How does batch normalization help optimization? In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, & R. Garnett (Eds.), *Advances in neural information processing systems* (Vol. 31). Curran Associates, Inc. Retrieved from https://proceedings.neurips.cc/paper/2018/file/905056c1ac1dad141560467e0a99e1cf-Paper.pdf

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* (pp. 309–318). London, UK, UK: Springer-Verlag. Retrieved from http://dl.acm.org/citation.cfm?id=647967.741626

Selvaraju, R. R., Cogswell, M., Das, A., Vedantam, R., Parikh, D., & Batra, D. (2017, Oct). Grad-cam: Visual explanations from deep networks via gradient-based localization. In *2017 ieee international conference on computer vision (iccv)* (p. 618-626). doi: 10.1109/ICCV.2017.74

Settles, B. (2009). *Active learning literature survey* (Computer Sciences Technical Report No. 1648). University of Wisconsin–Madison. Retrieved from http://axon.cs.byu.edu/~martinez/classes/778/Papers/settles.activelearning.pdf

Shahriari, B., Swersky, K., Wang, Z., Adams, R. P., & de Freitas, N. (2016, Jan). Taking the human out of the loop: A review of bayesian optimization. *Proceedings of the IEEE*, *104*(1), 148-175. doi: 10.1109/JPROC.2015.2494218

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

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

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

Turner, R., Eriksson, D., McCourt, M., Kiili, J., Laaksonen, E., Xu, Z., & Guyon, I. (2021). Bayesian optimization is superior to random search for machine learning hyperparameter tuning: Analysis of the black-box optimization challenge 2020. *CoRR*, *abs/2104.10201*. Retrieved from https://arxiv.org/abs/2104.10201

Ustun, B., & Rudin, C. (2016, Mar 01). Supersparse linear integer models for optimized medical scoring systems. *Machine Learning*, *102*(3), 349–391. Retrieved from https://doi.org/10.1007/s10994-015-5528-6 doi: 10.1007/s10994-015-5528-6

Uzilov, A. V., Keegan, J. M., & Mathews, D. H. (2006, Mar 27). Detection of noncoding rnas on the basis of predicted secondary structure formation free energy change. *BMC bioinformatics*, *7*, 173-173. Retrieved from https://www.ncbi.nlm.nih.gov/pubmed/16566836 (16566836[pmid]) doi: 10.1186/1471-2105-7-173

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

30(6), 1673–1724. Retrieved from https://doi.org/10.1162/neco_a_01088 doi: 10.1162/neco_a_01088

Wang, T. (2018). Multi-value rule sets for interpretable classification with feature-efficient representations. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, & R. Garnett (Eds.), *Advances in neural information processing systems 31* (pp. 10835–10845). Curran Associates, Inc. Retrieved from http://papers.nips.cc/paper/8281-multi-value-rule-sets-for-interpretable-classification-with-feature-efficient-representations.pdf

Weiss, K., Khoshgoftaar, T. M., & Wang, D. (2016). A survey of transfer learning. *Journal of Big Data*, *3*(1), 9. Retrieved from https://doi.org/10.1186/s40537-016-0043-6 doi: 10.1186/s40537-016-0043-6

Yang, Y., & Loog, M. (2018). A benchmark and comparison of active learning for logistic regression. *Pattern Recognit.*, *83*, 401–415. Retrieved from https://doi.org/10.1016/j.patcog.2018.06.004 doi: 10.1016/j.patcog.2018.06.004

Yeo, I.-K., & Johnson, R. A. (2000). A new family of power transformations to improve normality or symmetry. *Biometrika*, *87*(4), 954–959. Retrieved from http://www.jstor.org/stable/2673623

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* (pp. 1039–1050). USA: Curran Associates Inc. Retrieved from http://dl.acm.org/citation.cfm?id=3326943.3327039

Zhang, H., Hu, Z., Qin, W., Xu, M., & Wang, M. (2021). Adversarial co-distillation learning for image recognition. *Pattern Recognition*, *111*, 107659. Retrieved from https://www.sciencedirect.com/science/article/pii/S0031320320304623 doi: https://doi.org/10.1016/j.patcog.2020.107659