Learning Interpretable Models Using an Oracle

1st Abhishek Ghose
Dept. of Computer Science and Engineering
IIT Madras
Chennai, India
abhishek.ghose.82@gmail.com

2nd Balaraman Ravindran
Dept. of Computer Science and Engineering
Robert Bosch Centre for Data Science and AI, IIT Madras
Chennai, India
ravi@cse.iitm.ac.in

***Abstract*—As Machine Learning (ML) becomes pervasive in various real world systems, the need for models to be *interpretable* or *explainable* has increased. We focus on interpretability, noting that models often need to be constrained in size for them to be considered understandable, e.g., a decision tree of depth 5 is easier to interpret than one of depth 50. This suggests a trade-off between interpretability and accuracy. We propose a technique to minimize this tradeoff. Our strategy is to first learn a powerful, possibly black-box, probabilistic model on the data, which we refer to as the oracle. We use this to adaptively sample the training dataset to present data to our model of interest to learn from. Determining the sampling strategy is formulated as an optimization problem that, independent of the dimensionality of the data, uses only seven variables. We empirically show that this often significantly increases the accuracy of our model. Our technique is *model agnostic* - in that, both the interpretable model and the oracle might come from any model family. Results using multiple real world datasets, using *Linear Probability Models* and *Decision Trees* as interpretable models, and *Gradient Boosted Model* and *Random Forest* as oracles are presented. Additionally, we discuss an interesting example of using a sentence-embedding based text classifier as an oracle to improve the accuracy of a term-frequency based bag-of-words linear classifier.**

***Index Terms*—machine learning**

I. 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 *interpretable* or *explainable* as well, in domains where the cost of being wrong is prohibitively high, e.g., medicine and healthcare [11], [63], defence applications [21], law enforcement [3], [29], banking [12]. It is expected that soon model transparency would be mandated by law within systems involving digital interactions [20].

Contemporary research in this area has adopted two broad approaches:

1) *Interpretability*: this area looks at building models that are considered easy to understand as-is, e.g., rule lists [2], [31], decision trees [9], [46], [47], sparse linear models [63], decision sets [28], pairwise interaction models that may be linear [35] or additive [37].
2) *Explainability*: this area looks at techniques that may be used to understand the workings of models that do not naturally lend themselves to a simple interpretation, e.g., locally interpretable models such as LIME, Anchors [51], [52], visual explanations for Convolutional Neural Networks such as Grad-CAM [55], influence functions [27].

We focus on interpretablity in this work; specifically, in improving the accuracy of the existing vast majority of models that are considered interpretable, e.g., linear models, rules.

Interpretable models are preferably small in *size*: this is anecdotally seen in how a linear model with 10 terms may be preferred over one with 100 terms, or in how a decision tree (DT) of $depth = 5$ is easier to understand than one of $depth = 50$. This property is variously acknowledged in the area of interpretability: [22] refers to this as low *explanation complexity*, this is seen as a form of *simulability* in [36], and is often listed as a desirable property in interpretable model representations [2], [28], [51]. The preference for small-sized models points to an obvious problem: since size is usually inversely proportional to model bias, such a model often trades accuracy for interpretability.

We propose a novel adaptive sampling technique to minimize this trade-off. We first learn a highly accurate, possibly black-box, *probabilistic* model on our training data. We refer to a model as "probabilistic" if it can produce a probability distribution over labels during prediction:

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

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

We refer to this model as our *oracle*. There are no size constraints imposed on the oracle.

Next, we try to incorporate the oracle's understanding of the data/input space into our interpretable model. The mechanism to do so is via *adaptive sampling*: we let the oracle suggest data for the interpretable model training algorithm to learn from.

Figure 1 compares a standard workflow (top) to our model building workflow (bottom). In the standard setup, a model training algorithm, A, accepts training data and produces a model that minimizes some pre-defined error 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 the "best" sample from the training data for input to

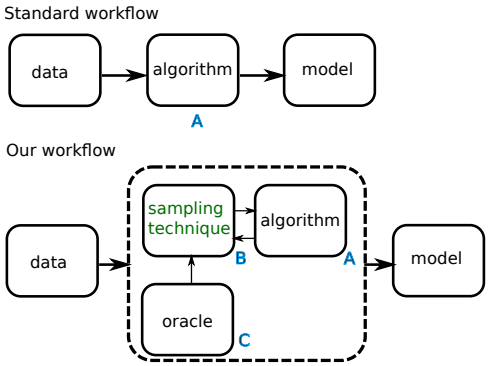


Fig. 1. Modified workflow.

algorithm A. Here, the "best" sample is the one that leads A to produce a model with the lowest error (measured on a held-out dataset). Determining this sample is an iterative process; at each iteration, B modifies the sample based on the current error of the model that A learns. The information from the oracle is conveyed to the sampling technique only once.

Our technique is *model agnostic*: since we assume no analytic/functional form for either the oracle or the interpretable model, they may come from different arbitrary model families.

Figure 2 provides a quick demonstration of our technique on a toy dataset.

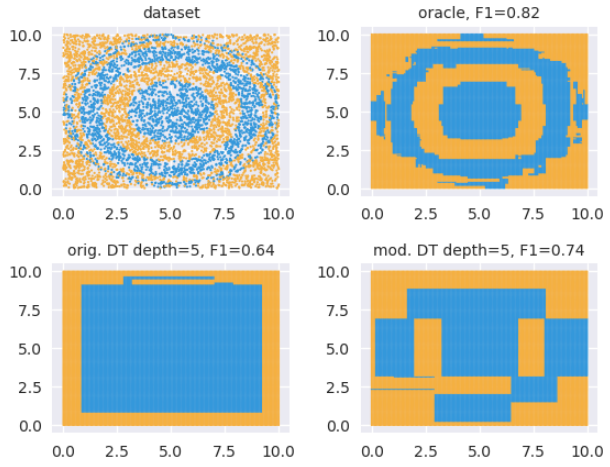


Fig. 2. Adaptive sampling using a Gradient Boosted Model as an oracle.

The top-left panel in Fig 2 shows a dataset with 2 labels we want to classify. The top-right panel visualizes what a *Gradient Boosted Model (GBM)* learns using the dataset. This seems quite accurate with a F1 score $= 0.82$. The bottom-left panel shows what a *CART* [9] decision tree of $depth = 5$ learns from the data; this has F1 score $= 0.64$. Finally, the bottom-right panel shows what a CART decision tree of $depth = 5$ learns, when we supply the GBM as an oracle to our technique. The F1 score improves significantly to 0.74.

Our adaptive sampling technique is an extension of the one discussed in [19]. For convenience, we refer to the latter as a being a *density tree* based approach - this is reviewed in a later section. *Our extension is non-trivial since we change critical aspects of the data density representation - using uncertainty information instead of explicit class boundary information - that the sampling technique relies on*. This enables to use an arbitrary oracle while performing better on average compared to the density tree approach.

We note, as in [19], that there is no standard definition of "model size" across model families. Even for a specific model family, there might be different notions of size, e.g., for *Random Forests (RF)*, both the number of trees and the maximum/average depth across trees may be seen as representing the size of a RF model. Often there are conventionally accepted notions of sizes, e.g., the depth of a DT, number of non-zero terms in a linear model. However, in such cases too, their preferred values may be subjective: some might consider DTs only up to a $depth = 10$ interpretable, while others might find $depth = 15$ to be a reasonable limit. Irrespective of such variances, as long as a specific notion of model size is inversely proportional to model bias, the discussion here applies. Referring to this general notion, we say most interpretable models are preferably *small*.

Our key contribution in this paper is to **provide a *model-agnostic* and *practically effective technique* to use an oracle to improve the accuracy of small interpretable models**. Since our work extends the density tree based approach, we list our specific contributions relative to it:

1) The ability to pick any oracle makes our technique flexible in terms of:
   a) The oracle may be arbitrarily powerful, e.g., we may choose between using a linear *Support Vector Machine (SVM)* or a *Deep Neural Network (DNN)* based on how accurate we want the oracle to be. We show that we outperform the density tree based approach on an average.
   Other concerns also might dictate the choice of an oracle, e.g., pre-trained models readily available, hardware resources and libraries, etc.
   b) Data representation to use - the oracle and the small model might represent the data with different features. We look at an example later.
2) Faster run-times on average.

Our technique may be used for improving the accuracy of any small model, and not exclusively interpretable models. The discussion here centres around interpretability since this is where we see a need for using small models.

The only work we are aware of that discusses model improvement in the small size regime, using adaptive sampling as a mechanism, is [19]. Using sampling as a strategy to improve model accuracy, in general, has been studied in the areas of *active learning* [56] and *core-set* identification [4], [41] although they focus on different problems. The term "oracle" is commonly used in active learning to indicate a source of correct label assignments; typically a human labeler. Our work also bears resemblance to *transfer learning* [42], [62] in that there is transfer of domain knowledge from the

oracle to the interpretable model. However, it differs from techniques in the area in two key respects:

1) Transfer learning techniques usually make some assumptions about the learning problem, e.g., the model family, such as Boolean concepts in [61] or Markov Logic Networks in [39], the learning framework, such as *Reinforcement Learning* in [15], [45], [59].
2) Although *instance-based transfer learning* looks at instance re-weighting, sampling etc., they typically study how to deal with change in data distributions available to different models [16], [23], [34].

The layout of the remaining paper is as follows: we begin by discussing our methodology in detail next, in Section II. Section III presents our experiments on multiple real-world datasets that validate the effectiveness of our technique. Finally, we conclude with our thoughts around future work in Section IV.

II. Methodology

We describe our methodology in this section. We begin by reviewing the core ideas of the density tree based approach since our technique builds on it.

A. Review: Density Tree Based Adaptive Sampling

Reference [19] begins by asking why should we expect to be able to improve the accuracy of a small model at all? The answer is that models are not always optimal for a *given size*. All training algorithms make heuristic choices to make learning tractable, e.g., local search based techniques such as *RMSProp* [65] and *Adam* [26] for learning neural networks, one-step look-ahead in the CART algorithm for learning DTs, . This creates a gap between the *representational* and *effective* capacities of models; ideally a DT learning algorithm constrained to learn trees of up to $depth = 5$ should be able to produce full binary trees with $2^5 = 32$ leaves if needed (for ex, in the bottom-left panel of Fig 2), but this is rarely seen in practice. Allowing the model to grow to an arbitrary size works around this issue; we may eventually end up with a DT of $depth = 10$ and 32 leaves that solves our problem. The possibility of decreasing this gap gives us an opportunity to improve small model accuracy. Selecting examples (from a larger training dataset) to focus the training algorithm on regions of the input space that most impact learning, is one mechanism to decrease the gap.

The authors formulate the problem of finding the best sample to learn from as one of determining an optimal sampling distribution over the training dataset. As a practically tractable approach, they first construct a DT on the data, with no depth restriction; information about the input space encoded in the tree structure is then used to determine this distribution. This tree is known as a *density tree*. Their key observation is that a DT tends to create leaves of small volume around class boundaries.

This is seen in Figure 3 where a rectangle represents a leaf. We note that most small rectangles are indeed located around class boundaries. Since we intend to solve a classification

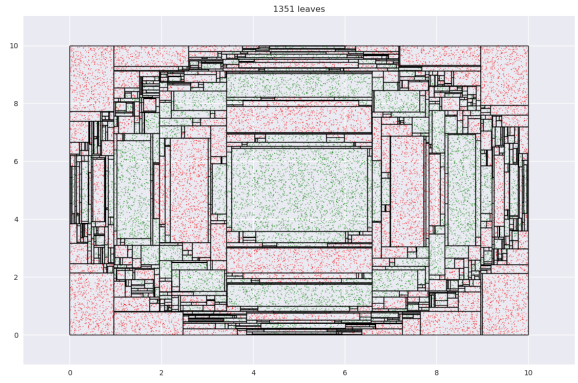


Fig. 3. Fragmentation of input space by decision tree. Source: [19].

problem, our interpretable model *at least* needs to know where the class boundaries are, and therefore, which points define these boundaries. These are now easily identifiable by locating leaves with small volumes. The model might additionally require points not near the class boundaries; since these are hard to characterize, a mechanism is setup to enable the interpretable model to select the following kinds of points: (a) points that define the class boundaries (b) optionally, points elsewhere in the input space. The purpose of categorizing points in this manner is that it is now easier for the model to choose points that it most likely needs - the boundary points. This fact need not be discovered ab initio while searching for the optimal distribution, thus significantly shrinking the parameter search space.

The mechanism to select points is realized by defining a "depth sampling distribution" over the depths of the density tree.

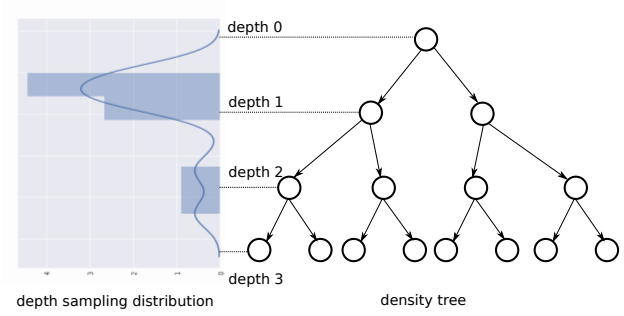


Fig. 4. Sampling using an IBMM from different levels in a density tree. Source: [19].

See Figure 4. The sampling technique samples a value of the density tree depth from the current distribution, and then samples points from nodes at this depth, where a preference is given to leaves with small volumes (at the leaf level this implies we get more points near class boundaries). In a way, the different tree levels represent different amounts of information about the class distribution, ranging from no information at the root, to complete information at the leaves;

and the sampling technique allows the interpretable model to select points at relevant information levels, by learning the parameters of the depth distribution.

The distribution family used is a *Infinite Beta Mixture Model (IBMM)* - a variation of the relatively popular *Infinite Gaussian Mixture Model (IGMM)* [49]. This allows for representing a wide variety of distributions with a fixed number of parameters.

This technique turns out to be highly effective, as is indicated by results in [19].

B. Proposed Extension

It is interesting to observe the depth distributions that are eventually learned. This is concisely visualized in Figure 5 for different datasets; the x-axis shows density tree depths normalized to lie within $[0, 1]$ (so IBMMs over density trees of various heights may be compared). The curve for a specific dataset is plotted in the following manner:

1) Interpretable models for a range of sizes are built for the dataset. Let's say the sizes are $k \in \{1, 2, ..., K\}$. For a size k, we assume the relative improvement - using a density tree compared to not using it - seen is Δ_k.
2) n_k points are sampled from the IBMM corresponding to the model of size k, where $n_k \propto \Delta_k$. Points sampled thus, across the K sizes, are pooled together.
3) A *Kernel Density Estimator (KDE)* is fit to this sample and plotted.

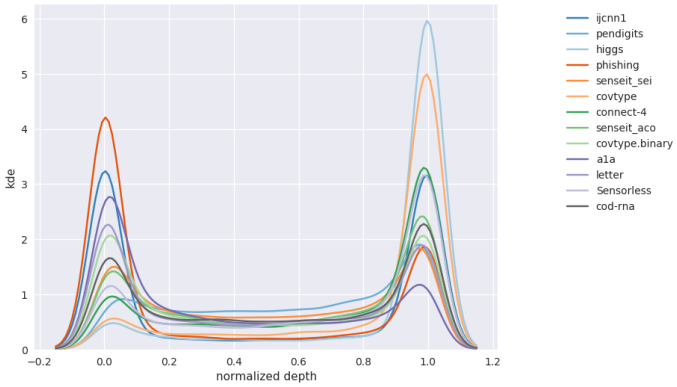


Fig. 5. Distribution over depths. Source: [19].

The weighting wrt Δ_k is performed so the KDE plot predominantly reflects the IBMMs where the greatest improvements are seen.

We observe that most of the sampling either occurs near the root or near the leaves. The authors point out that this pattern is fairly consistent across their other experiments. The reason hypothesized is since the intermediate tree levels are noisy wrt to information content - the class boundaries have not been fully discovered yet, but we have moved away from the original distribution - the interpretable model avoids sampling here.

Given that the density tree is apparently primarily useful at the leaf-level (the root level has the original data as-is), we ask if there is a different way to encode the information of class boundaries.

Obviously, *all* classifiers implicitly possess this information for classification to work; the non-trivial aspect is to be able to make this information *explicit* - as in leaves with small volumes in density trees.

Here, we consider probabilistic classifiers. Since they provide confidence of prediction, one way to identify points near class boundaries is to measure how diffuse the confidences are across different labels. Such a metric is known as *uncertainty* and has been studied in the *active learning* community [32], [53], [56].

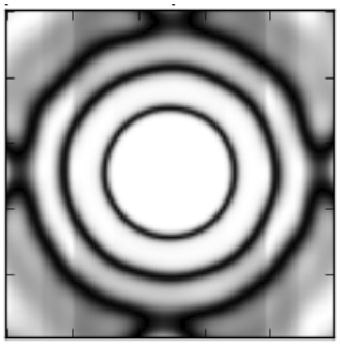


Fig. 6. Darker shades show higher uncertainty.

See Figure 6 for intuition. We use a dataset similar to the one used in Figure 3, learn a SVM with a *Radial Basis Function (RBF)* kernel on it, and show the uncertainty values across the input space. Between the visualizations in Figures 3 and 6, we see that leaves with low volumes and high uncertainties respectively indicate approximately the same regions.

This is the key insight behind our extension: we replace the density tree with an *arbitrary* probabilistic classifier, with the explicit leaf level class boundary information being substituted by uncertainty scores. *The value of our work is in showing this substitution is indeed effective.*

The remnant of the sampling problem stays the same: while we believe the boundary points are more likely to be used for learning, it is hard to characterize additional non-boundary points that may be required. So, we use a IBMM to represent a distribution over the uncertainty values of points in the training dataset. Learning the parameters of this distribution is formulated as an optimization problem. Note the subtle difference in the role of the IBMM: in the density tree based approach, it represents different depths, and thus different amounts of information about class boundaries; here, the IBMM is over the uncertainty scores, which already represent the complete information from the oracle.

C. Algorithm Details

We now go into additional details around the high level ideas presented in the previous section.

1) Measuring Uncertainty: We begin with the uncertainty metric since our technique critically depends on it. Some popular metrics used to measure uncertainty are:

1) Least confident: we look at the confidence of the most probable class alone. Given a model M and an instance x, the uncertainty $u_M(x)$ is measured as:

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

where we have C classes, and $M(y_i|x)$ is the probability score[1] produced by the model.

2) Entropy: this is the standard entropy measure:

$$u_M(x) = \sum_{y_i \in \{1,2,...,C\}} -M(y_i|x) \log M(y_i|x) \tag{3}$$

3) Margin: difference of the top two probabilities. Introduced in [53]. See Algorithm 1.

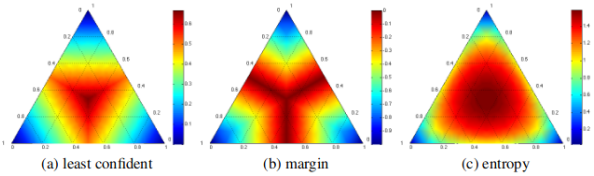
Algorithm 1: Compute uncertainty of prediction, $u_M(x)$, for instance x by model M

Data: x, M
Result: $u_M(x)$
Calculate probabilities for classes y_i, $p_i = M(y_i|x)$, where $y_i \in \{1, 2, ..., C\}$
Let $p_{j_1} \leq p_{j_2} \leq ... \leq p_{j_{C-1}} \leq p_{j_C}$
$u_M(x) \leftarrow 1 - (p_{j_C} - p_{j_{C-1}})$
return $u_M(x)$

Note that all the metrics $u_M(x) \in [0, 1]$. Also, all our oracles are **calibrated** with a *sigmoid* [44].

We use the *margin* uncertainty metric. We do not use the *least confident* metric since it ignores confidence distribution across labels. While *entropy* is quite popular, we do not use it since it reaches its maximum for only points for which the classifier must be uncertain about *all* labels; for datasets with many labels (one of our experiments uses a dataset with 26 labels - see Table I) we might never reach this maximum; also we want to highly penalize points with 2 ambiguous labels too.

There is no best uncertainty metric in general, and the choice is usually application specific [54], [56].



(a) least confident (b) margin (c) entropy

Fig. 7. Heatmap visualizations of uncertainty scores. Source: [56]

Fig 7 visually shows what the uncertainty values look like for the various metrics on a triangular plane; points with the highest confidences lie near the vertices, where each vertex represents a different class. We note that only the *margin* metric reaches peak uncertainty scores at the 2-label boundaries.

2) Sampling from the Oracle: As mentioned before we use a IBMM to represent the *probabiity density function (pdf)* over uncertainties. Our reasons to use it are:

1) It can represent an arbitrary *pdf*. This is important since we want to discover the optimal *pdf* over uncertainties. It seems natural to assume that we *exclusively* require data points with high uncertainty, but our experiments show this is not optimal.
2) Uses a fixed set of parameters to represent arbitrary *pdfs*. This is convenient since most optimizers support parameter spaces of fixed size. Contrast this representation with *Gaussian Mixture Models (GMM)* where the parameter space is conditional: number of mixture components decide the total number of parameters.

The number of components in the IBMM are represented using a *Dirichlet Process (DP)*. The DP is characterized by a *concentration parameter* α, which determines both the number of components (also known as *partitions* or *clusters*) and association of a data point to a specific component. The parameters for these components are drawn from prior distributions; the parameters of these prior distributions comprise our (fixed set of) variables.

Since $u_M(x) \in [0, 1]$ we need mixture components that are also constrained within $[0, 1]$; so the $Beta$ components in the IBMM fit our purpose well. This is a univariate distribution since the only random variable is the uncertainty score.

This is how we sample N points from the oracle:

1) Determine partitioning over the N points induced by the DP. We use *Blackwell-MacQueen* sampling [7] 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 these $Beta$ are also represented by $Beta$ distributions: $A_i \sim Beta(a, b)$ and $B_i \sim Beta(a', b')$. Thus we have two prior $Beta$ distributions associated with our IBMM.
3) Repeat for each c_i: for each instance x_j in our training dataset, we fetch the uncertainty score, $u_{M_O}(x_j)$, assigned to it by the oracle M_O. We calculate $p_j = Beta(u_{M_O}(x_j)|A_i, B_i)$. We scale these probabilities to sum to 1. Then use these probabilities as sampling probabilities in the training dataset to sample n_i points.

The parameters for the IBMM are collectively denoted by $\Psi = \{\alpha, a, b, a', b'\}$.

This procedure is summarized in Algorithm 2.

3) Learning an Interpretable Model using an Oracle: We tie together the various individual pieces in this section. Before we proceed, we mention the parameters we use in addition to the IBMM parameters:

1) $N_s \in \mathbb{N}$, sample size. The sample size, obviously, might have a significant effect on training the interpretable

[1]We use the non-standard symbol M to denote the model score so as to be consistent across various pseudo-codes in this paper.

Algorithm 2: Sample based on uncertainties

Data: sample size N, model M, data (X, y), Ψ
Result: $D = \{(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), ..., (x^{(N)}, y^{(N)})\}$

$D = \{\}$
for $i \leftarrow 1$ **to** $|X|$ **do**
  $p_i = p(u_M(X_i); \Psi)$
end
$\Theta_i \leftarrow c \cdot p_i$, where $c = 1/\sum_i p_i$
for $j \leftarrow 1$ **to** N **do**
  Sample $(x^{(j)}, y^{(j)}) \sim p(x; \Theta)$
  $D \leftarrow D \cup \{(x^{(j)}, y^{(j)})\}$
end
return D

model. We let the optimizer determine the best sample size to learn from. We constrain N_s to be larger than the minimum number of points needed for statistically significant results.

2) $p_o \in [0, 1]$ - proportion of the sample from the original distribution. Given a value for N_s, we sample $(1-p_o)N_s$ points using the oracle and $p_o N_s$ points from our training data (X_{train}, y_{train}). The latter sample is stratified wrt y_{train}. We use this parameter since, as in [19], we expect the original distribution to be optimal to learn from at some model sizes; this parameter allows for that possibility.

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

Our sampling technique is presented as Algorithm 3.

We begin by creating the 3 stratified splits of our dataset $(X_{train}, y_{train}), (X_{val}, y_{val}), (X_{test}, y_{test})$. We train our oracle, M_O, on (X_{train}, y_{train}) and obtain uncertainty scores for all points in X_{train}. Now, we begin to learn the optimal parameters, Φ^*, as an iterative process, with an optimization budget of T iterations. At each iteration t, based on our current parameters Φ_t, we create a sample (X_t, y_t) to learn our intepretable model, M_t. Our objective function is $accuracy()$, which produces a prediction accuracy score, s_t, over the validation set (X_{val}, y_{val}). The history of scores and parameters obtained so far is passed as input to the optimizer, $suggest()$, which generates parameters for the next iteration, Φ_{t+1}.

The optimal parameters, Φ^*, are defined as the ones in the iteration where M_t achieved the highest score (across the T iterations) on the validation set. The final model is trained with Φ^* and its accuracy on the test set, (X_{test}, y_{test}), is reported.

We use a Bayesian Optimizer (BO) to implement $suggest()$ since it fulfills the following desiderata for our problem[2]:

1) It should be ale to optimize a black-box function. Our objective function is $accuracy()$, but this uses M_t, which is purposefully kept unspecified since we want our technique to be model-agnostic.

[2]which, given the similarity in representations, overlap considerably with the ones mentioned for the density tree based approach in [19]

Algorithm 3: Adaptive sampling using oracle

Data: Data (X, y), iterations T
Result: Φ^*, s_{test}

Create stratified splits $(X_{train}, y_{train}), (X_{val}, y_{val}), (X_{test}, y_{test})$ from (X, y)
$M_O \leftarrow$ learn calibrated probabilistic oracle on (X_{train}, y_{train})
for $t \leftarrow 1$ **to** T **do**
  $\Phi_t \leftarrow suggest(s_1, ...s_{t-1}, \Phi_1, ..., \Phi_{t-1})$ `// randomly initialize at t = 1`
  $N_o \leftarrow p_{o,t} \times N_{s,t}$ where $p_{o,t}, N_{s,t} \in \Phi_t$
  $N_p \leftarrow N_s - N_o$
  $D_o \leftarrow$ sample N_o points from (X_{train}, y_{train})
  $D_p \leftarrow$ sample N_p points from (X_{train}, y_{train}) using M_O, Ψ_t where $\Psi_t \in \Phi_t$ `// Algorithm 2`
  $(X_t, y_t) = D_o \cup D_p$
  $M_t \leftarrow train((X_t, y_t))$
  $s_t \leftarrow accuracy(M_t(X_{val}), y_{val})$
end
$t^* \leftarrow \arg\max_t \{s_1, s_2, ..., s_{T-1}, s_T\}$
$\Phi^* \leftarrow \Phi_{t^*}$
Sample (X^*, y^*) based on Φ^*
$M^* \leftarrow train((X^*, y^*))$
$s_{test} \leftarrow accuracy(M^*(X_{test}), y_{test})$
return Φ^*, s_{test}

2) It must be resilient to noisy evaluations of the objective function. The noise might come from the fact that we use a sample (X_t, y_t) to train on, or $train()$ itself might possess an element of stochasticity, e.g., if it uses *Stochastic Gradient Descent (SGD)* as part of the training step.
3) It should minimize calls to $train()$; this is clearly desirable since learning M_t are expensive.

Over multiple iterations, BOs build their own model of the response surface (here, accuracy score s) as a function of the optimization variables (here, Φ). This enables them to work with black-box objective functions. Since they explicitly quantify the robustness of the response surface model, by using appropriate representations such as *Gaussian Processes (GP)* or KDEs, they can handle reasonable amounts of noise. The response surface model allows BOs to balance *exploitation and exploration* to make well-informed choices about what point in the optimization space to next evaluate the objective function on - making it conservative in its calls to $train()$. See Reference [10] for details.

Among various BO algorithms available, e.g., [6], [24], [30], [33], [38], [48], [57], [58], [64], we use the *Tree Structured Parzen Estimator (TPE)* [6] (as implemented in the *hyperopt* package [5]), since its runtime is linear in its input (history of suggested parameters and objective function values), and it has a mature library.

□

TABLE I
DATASETS

dataset	dimensions	# classes
cod-rna	8	2
ijcnn1	22	2
higgs	28	2
covtype.binary	54	2
phishing	68	2
a1a	123	2
pendigits	16	10
letter	16	26
Sensorless	48	11
senseit_aco	50	3
senseit_sei	50	3
covtype	54	7
connect-4	126	3

This concludes the description of our methodology. We discuss our experiments next.

III. EXPERIMENTS

We perform two sets of experiments to validate our approach. These are described in this section.

A. Multiple Oracles on Real Datasets

We use multiple real world datasets to perform classification on, to assess the practical effectiveness of our technique. Our datasets are obtained from the *LIBSVM* [14] website and are listed in Table I. We have intentionally used the same datasets (and interpretable models) as in [19] for convenient comparison of results.

The following model families are used to learn interpretable models:

1) *Linear Probability Model (LPM)*: This is a linear classifier. Our notion of size is the number of non-zero terms in the model, i.e., features from the original data with non-zero coefficients. For more than two classes, we construct *one-vs-rest (ovr)* models.
   The *Least Angle Regression* [18] algorithm is used to learn the LPMs, since it grows the models one term at a time, allowing us to conveniently enforce the size constraint.
   For a dataset, the LPM model sizes tried out are $\{1, 2, .., min(d, 15)\}$ where d is the dimensionality of the dataset.
2) *DT*: We consider the tree depth as its size. We use the implementation of CART in the *scikit-learn* library [43]. Unlike the LPM model, note that DTs don't allow you to enforce an exact depth, but only a *maximum* depth. This implies, occasionally, we don't have models for the exact specified size - this shows up as missing values in the results table for DTs in Table IV.
   If the best performing DT (no size constraints) for a dataset has depth $depth_{opt}$, then we construct DTs of sizes $\{1, 2, ..., depth_{opt}\}$ for it.

The following model families are used to learn oracles:

1) Random Forests (RF): we use the implmentation provided in *scikit-learn*.
2) Gradient Boosted Model (GBM): we use the *LightGBM* implementation [25].

The BO solves a constrained optimization problem, so we set the parameters to these ranges:

1) $p_o \in [0, 1]$.
2) $N_s \in [1000, 10000]$. The lower bound ensures that 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 admit various shapes for the $Beta$ distributions.
4) α: For a DP, $\alpha \in \mathbb{R}^+$. We use a lower bound of 0.1. We estimate the upper bound using the following property: for a DP describing N points, the expected number of components is $O(\alpha H_N)$, where H_N is the N^{th} *harmonic sum* [60]. Setting the expected number of *Beta* components to a high number (we use 60; from Figure 9 we see this much more than what we need), we compute the upper bound of α to be $60/H_{1000}$ (we use the lower bound of N_s for H_N since we are interested in the upper bound for α, and $\alpha \propto 1/H_N$).

The optimization budget, T, was set to 3000 when the interpretable model is DT. For LPM, for 2-class problems, $T = 3000$, but for higher number of classes we decrease the budget to $T = 1000$, to account for the increase in run time (more models because of ovr). [19] used identical budgets so our results are comparable.

For the experiments with DT and LPM At each model size, we record the percentage *relative improvement* in the $F1$ score on (X_{test}, y_{test}) compared to the *baseline* of training with the original distribution. This score is calculated as:

$$\delta F1 = \frac{100 \times (F1_{new} - F1_{baseline})}{F1_{baseline}} \tag{4}$$

Since we can never perform worse than training on the original distribution, the lowest improvement we report is 0%.

The results of the LPM experiments are shown in Table II.

The oracle based scores are indicated against the original density tree based scores from [19] for easy comparison. The highest scores for a dataset and model size combination is highlighted - if this score is obtained using the density tree based approach, we highlight it in red, else in green, for the relevant oracle. The F1 scores of the oracles are mentioned in the *oracle_type* column. We also compute a *RMSE* score using the $\delta F1$ from the density tree and the higher of the oracle scores, to indicate how much the scores from each of these approaches vary.

We observe that the oracles outperform the density tree scores most of the time (visually: more green highlights than red) - 88.77% of the time to be precise. $RMSE = 21.26$ for this experiment, which implies that in addition to doing better often, the oracle based runs do better by a wide margin.

TABLE II
LINEAR PROBABILITY MODEL

dataset	oracle_type	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
cod-rna	density tree	13.95	29.93	**66.56**	**76.41**	**67.93**	15.30	2.44	3.76	-	-	-	-	-	-	-
	gbm (F1=0.93)	**53.34**	**45.13**	49.25	48.52	50.64	**57.42**	**65.35**	12.32	-	-	-	-	-	-	-
	rf (F1=0.93)	52.37	44.45	44.56	51.87	53.09	44.42	15.63	**15.48**	-	-	-	-	-	-	-
ijcnn1	density tree	19.68	7.09	0.00	2.06	0.20	1.15	0.20	0.41	1.03	0.87	0.41	1.61	2.20	0.15	**3.45**
	gbm (F1=0.89)	**23.63**	8.20	0.00	0.62	0.46	0.58	1.91	3.17	2.49	**3.09**	2.96	3.23	0.29	1.84	0.35
	rf (F1=0.88)	23.03	**10.46**	**1.42**	**2.26**	**3.27**	**3.23**	**1.94**	**3.19**	**3.41**	2.49	**3.19**	**4.15**	**2.96**	**2.01**	2.75
higgs	density tree	0.00	0.11	0.21	0.11	1.08	1.29	1.12	2.87	1.17	1.39	4.11	2.92	4.37	3.36	4.96
	gbm (F1=0.71)	**51.00**	**77.69**	**77.36**	**75.32**	**75.44**	**75.05**	**68.15**	**69.03**	**68.44**	**70.00**	**69.63**	**68.36**	**71.23**	**71.51**	**70.99**
	rf (F1=0.68)	11.34	50.73	51.86	62.87	67.17	67.59	68.03	64.86	66.65	65.86	66.83	63.79	61.79	58.46	60.16
covtype.binary	density tree	0.40	3.68	4.03	6.49	9.24	11.61	11.27	4.76	3.71	8.70	9.48	8.33	10.43	13.76	12.82
	gbm (F1=0.78)	0.00	**17.16**	**19.45**	**30.38**	**33.64**	**53.96**	**52.89**	**64.74**	**67.14**	**68.45**	**58.90**	**61.12**	**56.55**	**49.31**	**47.47**
	rf (F1=0.83)	**0.48**	1.45	3.41	1.57	17.22	25.63	22.56	21.48	29.65	23.46	24.59	27.79	35.34	34.50	38.24
phishing	density tree	0.00	**0.00**	0.00	**0.00**	**0.00**	0.15	**0.94**	0.00	0.00	0.01	0.00	0.00	0.00	0.00	0.33
	gbm (F1=0.95)	**171.74**	0.00	**0.26**	0.00	0.00	**0.56**	0.54	**0.62**	0.00	**0.30**	0.23	0.11	0.24	**1.04**	**0.90**
	rf (F1=0.96)	0.00	0.00	0.00	0.00	0.00	0.32	0.69	0.00	**0.04**	0.27	**0.57**	**0.94**	**0.73**	1.02	0.72
a1a	density tree	0.00	41.77	62.21	0.00	0.00	1.55	0.38	1.25	1.81	0.10	1.89	2.68	2.95	0.69	2.89
	gbm (F1=0.77)	0.00	20.55	61.76	43.30	28.30	**45.47**	**45.04**	**14.99**	**17.47**	**10.61**	**13.64**	**14.52**	**6.17**	3.40	**6.07**
	rf (F1=0.74)	0.00	**64.10**	**65.39**	**44.27**	**45.34**	19.74	20.41	4.03	4.50	4.07	12.37	6.36	3.23	**6.16**	1.89
pendigits	density tree	**11.36**	**9.94**	**10.38**	4.82	9.06	2.36	3.10	0.95	1.40	0.96	0.58	0.43	0.82	0.00	0.35
	gbm (F1=0.99)	10.00	9.68	7.79	**10.53**	4.34	3.95	6.44	0.00	0.36	0.90	0.19	1.77	2.87	2.68	1.21
	rf (F1=0.99)	2.78	4.94	5.44	8.92	**11.48**	**8.24**	**7.51**	**2.22**	**1.56**	**1.84**	**1.71**	**2.08**	**3.33**	**3.05**	**3.56**
letter	density tree	8.94	16.12	23.44	11.49	8.55	3.42	0.00	1.44	1.27	5.18	0.00	1.88	5.55	6.70	6.88
	gbm (F1=0.90)	24.71	**18.89**	**46.52**	**23.88**	**16.78**	**7.41**	**10.44**	**7.83**	4.33	5.78	4.87	6.13	3.68	7.65	6.30
	rf (F1=0.91)	**29.73**	7.48	31.26	5.37	13.85	3.72	7.21	5.31	**10.71**	**8.49**	**9.70**	**16.43**	**16.01**	**15.15**	**16.95**
Sensorless	density tree	59.59	60.09	19.07	26.65	26.38	34.96	38.89	47.82	37.05	**49.47**	34.26	**44.64**	46.68	40.58	44.13
	gbm (F1=0.99)	**210.99**	**162.03**	**123.18**	**77.72**	**66.45**	**67.08**	**64.13**	**63.02**	**52.46**	49.20	37.50	17.60	**68.29**	**78.97**	**74.65**
	rf (F1=0.98)	112.85	119.61	51.09	40.49	43.95	36.80	39.30	36.63	40.41	36.45	**42.36**	37.61	43.24	49.80	44.48
senseit_aco	density tree	19.27	79.89	68.00	32.05	13.35	16.77	8.49	6.00	1.87	3.64	1.30	1.67	2.25	0.34	0.13
	gbm (F1=0.72)	**158.32**	169.21	42.03	38.16	36.51	36.29	22.25	**18.62**	**15.90**	**14.85**	**15.59**	**15.07**	**12.33**	**13.50**	**13.67**
	rf (F1=0.72)	108.01	**178.22**	**88.84**	**57.70**	**52.07**	**42.01**	**30.59**	14.72	11.02	7.69	4.59	3.40	2.94	1.88	0.90
senseit_sei	density tree	137.91	44.84	**26.91**	7.67	2.25	0.59	0.00	2.13	0.95	0.89	0.71	1.70	0.08	0.40	0.48
	gbm (F1=0.67)	**161.17**	**80.72**	18.86	**10.05**	6.36	3.38	**5.29**	**3.88**	**4.05**	3.27	4.58	**4.03**	3.14	2.73	3.08
	rf (F1=0.68)	102.36	48.48	12.74	9.22	**7.25**	**4.72**	4.61	3.71	3.77	**4.02**	**4.75**	3.66	**3.89**	**4.22**	**4.27**
covtype	density tree	**43.36**	19.21	4.96	**6.01**	3.08	0.72	4.83	7.60	6.46	5.39	4.29	1.76	2.20	5.07	3.10
	gbm (F1=0.51)	34.46	**56.88**	**16.94**	1.94	**7.76**	**5.32**	**15.51**	**15.41**	**24.05**	**18.07**	**16.75**	**15.31**	**17.91**	9.79	8.16
	rf (F1=0.65)	29.43	23.71	7.84	0.00	2.60	4.37	3.35	5.19	5.32	10.76	6.33	9.52	15.44	**13.22**	**15.62**
connect-4	density tree	0.00	**38.59**	**50.94**	30.86	**8.71**	6.57	5.31	**13.10**	**10.80**	4.98	1.68	1.98	0.00	0.00	2.33
	gbm (F1=0.53)	**32.73**	22.48	23.84	**33.61**	7.22	8.39	**13.86**	7.61	7.74	5.42	2.08	3.17	**2.19**	**1.13**	1.78
	rf (F1=0.58)	6.31	1.02	2.93	7.60	2.32	**8.52**	2.63	5.01	0.79	**6.43**	**2.61**	**5.12**	0.00	0.00	**3.85**

An oracle performs better 88.77% of the time. $RMSE = 21.26$, computed with density tree score and best oracle score.

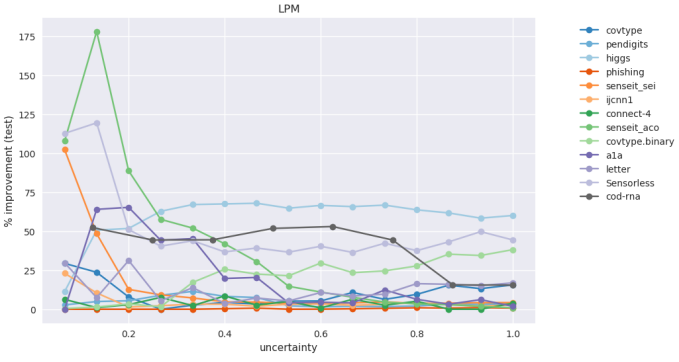


Fig. 8. Improvements in $F1$ score, when using a calibrated Random Forest as oracle.

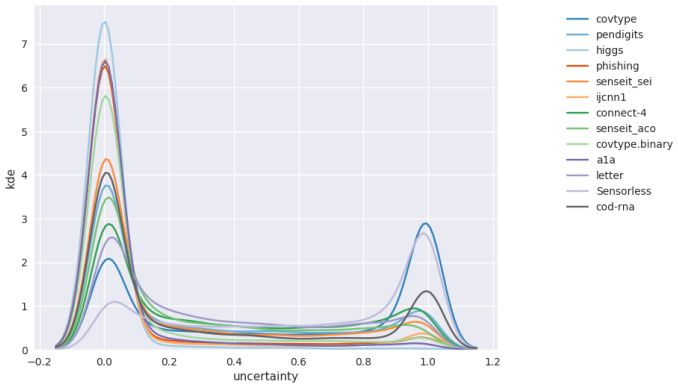


Fig. 9. Distribution over uncertainties, when using a calibrated Random Forest as oracle.

The improvements for various model sizes, when using a RF, are also visually depicted in Figure 8. The model sizes are standardized to be $\in [0, 1]$.

We plot the IBMMs learnt for various model sizes using a KDE, similar to Figure 5, where the IBMMs for different model sizes are weighted by the relative improvements. As mentioned before, its notable that its not just data instances with high uncertainty scores that contribute to the optimal training sample.

In the density tree based approach, we needed to sample from the density tree at each iteration. Here, in contrast, the

TABLE III
IMPROVEMENTS IN RUNTIME

dataset	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
cod-rna	1.21	0.85	1.10	0.80	0.74	0.79	0.88	1.04	-	-	-	-	-	-	-
ijcnn1	1.65	1.80	1.58	2.14	1.30	1.74	1.78	1.51	1.71	1.45	1.31	1.35	1.63	1.50	1.63
pendigits	1.09	1.78	1.40	1.46	1.57	1.13	1.18	1.31	1.36	1.41	1.20	1.48	1.23	1.49	1.34
letter	1.30	1.53	1.26	1.31	1.62	1.52	1.50	1.48	1.27	1.30	1.31	1.27	1.33	1.36	1.32

oracle may be queried for the uncertainty scores only once, which are then reused within the optimization loop. We expect this to be faster. Table III shows the improvements in LPM running times compared to density tree approach. If the density tree approach takes t_d seconds to run, and the RF oracle based approach takes t_o seconds, the number reported is t_d/t_o. We note that barring 4 of 53 instances (all for *cod-rna*), the oracle based approach is indeed faster. *cod-rna* is the smallest dataset we have - with 8 features - the speedups of the current approach are possibly only visible from slightly larger sizes onward.

We present the results with the DT interpretable model in the Appendix (Table IV) in the interest of space. We note that the density tree based approach does better relative to LPMs, with the oracle based approaches doing better only 63.5% of the time. However, the $RMSE = 9.33$ is not as high, which implies the density tree approach probably doesn't do substantially better on an average.

B. Text Classification

We look at an interesting application of our technique in this section. Our previous experiments looked at cases where the data, specifically the feature vector representation, was identical for the oracle and the interpretable model. This is also what Algorithm 3 implicitly assumes. Here, we explore the possibility of going a step further and changing the feature vectors between the oracle and the interpretable model. We consider the task of document classification, where our 'raw data' comprises of text documents from the *ag-news* dataset [66]. However, the feature vectors used by the models are different:

1) Oracle: Documents are converted to *Universal Sentence Encoder* Embeddings [13]. The oracle model is a RF built on these embeddings.
2) Interpretable model: We use a bag-of-words (BoW) feature vector, using term frequencies (tf) as feature values. A LPM model of $size = 5$ is used as the classifier.

We use 2 (of 4) labels from the dataset. The accuracy on our task is measured with the F1 score.

Figure 10 shows the improvements we obtain at various sizes of the dataset, with a LPM of $size = 5$. Clearly, the USE embeddings influence the effectiveness of the simpler tf based BoW representation. We believe this is a particular powerful and exciting application of our technique since:

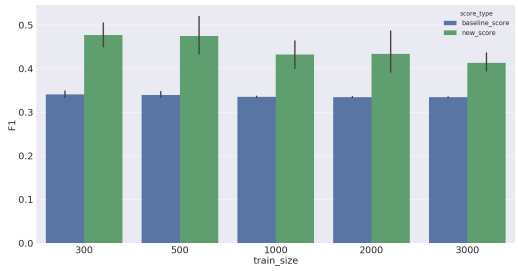


Fig. 10. LPM with 5 non-zero terms on tf-BoW document features, for different training data sizes.The blue bars show the original LPM F1-scores, the green bars show scores after learning from an oracle, which here is a Random Forest learned on USE embeddings.

1) Embeddings are an active and impactful area of research today [1], [8], [13], [17], [40], and our technique allows interpretable models to benefit from them even if they may not directly use them.
2) This can be extended to more general applications. For ex, in the case of image classification, we might use a *Convolutional Neural Network (CNN)* as our oracle, which would provide uncertainty scores to an interpretable model that uses superpixels [50] for image representation.

IV. CONCLUSION AND FUTURE WORK

In this work, we have demonstrated a practical technique for using an oracle - a powerful but possibly black-box model - to improve the predictive accuracy of a small-sized interpretable model. The ability to use an arbitrary oracle opens up a wide range of possibilities; we see this in the context of text classification, where we use sentence embeddings, a relatively recent technique, to improve a classifier based on the ubiquitous tf based BoW representation. The key element of our solution is density discovery, which typically requires solving for at least $O(d)$ variables, where d is the dimensionality of the data; but we show that using *uncertainty* information from the oracle is a feasible proxy for richer density information, that enables us to use a fixed number of only $seven$ variables.

The discussion here also suggests some interesting directions for further research:

1) Is there a way to quantify the upper bound of the improvement attainable by changing the distribution of

the input data? At what point can we declare a model to be "density-optimal", i.e., its accuracy cannot be decidedly increased further by changing input density?

2) What is a good way to characterize the conditions in which changing feature vectors would work (as in the text classification example in Section III-B)? This is clearly a powerful application, but its also easy to think of pathological examples: if $f : \mathcal{X}' \rightarrow \mathcal{X}''$ represents the mapping between the oracle feature vectors to the interpretable model feature vectors, and $|\mathcal{X}'| \gg |\mathcal{X}''|$, clearly much of the uncertainty information is useless since it cannot distinguish between feature vectors in $\mathcal{X}''$ well enough.

We had initially mentioned that we might expect to improve small model accuracy by decreasing the gap between representational and effective capacities; this might be obvious in hindsight, but the true value of our work is to provide a practical method to do so.

REFERENCES

[1] Sami Abu-El-Haija, Bryan Perozzi, Rami Al-Rfou, and Alex Alemi. Watch your step: Learning node embeddings via graph attention. In *Proceedings of the 32Nd International Conference on Neural Information Processing Systems*, NIPS'18, pages 9198–9208, USA, 2018. Curran Associates Inc.

[2] Elaine Angelino, Nicholas Larus-Stone, Daniel Alabi, Margo Seltzer, and Cynthia Rudin. Learning certifiably optimal rule lists. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '17, pages 35–44, New York, NY, USA, 2017. ACM.

[3] Julia Angwin, Jeff Larson, Surya Mattu, and Lauren Kirchner. Machine Bias. https://www.propublica.org/article/machine-bias-risk-assessments-in-criminal-sentencing, May 2016.

[4] Olivier Bachem, Mario Lucic, and Andreas Krause. Practical coreset constructions for machine learning. 03 2017.

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

[6] James Bergstra, Rémi Bardenet, Yoshua Bengio, and Balázs Kégl. Algorithms for hyper-parameter optimization. In *Proceedings of the 24th International Conference on Neural Information Processing Systems*, NIPS'11, pages 2546–2554, USA, 2011. Curran Associates Inc.

[7] David Blackwell and James B. MacQueen. Ferguson distributions via polya urn schemes. *Ann. Statist.*, 1(2):353–355, 03 1973.

[8] Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. Enriching word vectors with subword information. *arXiv preprint arXiv:1607.04606*, 2016.

[9] Leo Breiman et al. *Classification and Regression Trees*. Chapman & Hall, New York, 1984.

[10] Eric Brochu, Vlad M. Cora, and Nando de Freitas. A tutorial on bayesian optimization of expensive cost functions, with application to active user modeling and hierarchical reinforcement learning. *CoRR*, abs/1012.2599, 2010.

[11] Rich Caruana, Yin Lou, Johannes Gehrke, Paul Koch, Marc Sturm, and Noemie Elhadad. Intelligible models for healthcare: Predicting pneumonia risk and hospital 30-day readmission. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '15, pages 1721–1730, New York, NY, USA, 2015. ACM.

[12] Sara Castellanos and Kim S. Nash. 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/, May 2018.

[13] Daniel Cer, Yinfei Yang, Sheng-yi Kong, Nan Hua, Nicole Limtiaco, Rhomni St. John, Noah Constant, Mario Guajardo-Cespedes, Steve Yuan, Chris Tar, Yun-Hsuan Sung, Brian Strope, and Ray Kurzweil. Universal sentence encoder. *CoRR*, abs/1803.11175, 2018.

[14] Chih-Chung Chang and Chih-Jen Lin. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2:27:1–27:27, 2011. Software available at http://www.csie.ntu.edu.tw/~cjlin/libsvm.

[15] Tom Croonenborghs, Kurt Driessens, and Maurice Bruynooghe. Learning relational options for inductive transfer in relational reinforcement learning. In Hendrik Blockeel, Jan Ramon, Jude Shavlik, and Prasad Tadepalli, editors, *Inductive Logic Programming*, pages 88–97, Berlin, Heidelberg, 2008. Springer Berlin Heidelberg.

[16] Wenyuan Dai, Qiang Yang, Gui-Rong Xue, and Yong Yu. Boosting for transfer learning. In *Proceedings of the 24th International Conference on Machine Learning*, ICML '07, pages 193–200, New York, NY, USA, 2007. ACM.

[17] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *NAACL-HLT*, 2018.

[18] Bradley Efron, Trevor Hastie, Iain Johnstone, and Robert Tibshirani. Least angle regression. *Ann. Statist.*, 32(2):407–499, 04 2004.

[19] Abhishek Ghose and Balaraman Ravindran. Optimal resampling for learning small models. *CoRR*, abs/1905.01520, 2019.

[20] Bryce Goodman and Seth Flaxman. European union regulations on algorithmic decision-making and a "right to explanation". *AI Magazine*, 38:50–57, 2017.

[21] David Gunning. Explainable Artificial Intelligence. https://www.darpa.mil/program/explainable-artificial-intelligence, July 2016.

[22] Bernease Herman. The promise and peril of human evaluation for model interpretability. *CoRR*, abs/1711.07414, 2017.

[23] Jiayuan Huang, Alexander J. Smola, Arthur Gretton, Karsten M. Borgwardt, and Bernhard Scholkopf. Correcting sample selection bias by unlabeled data. In *Proceedings of the 19th International Conference on Neural Information Processing Systems*, NIPS'06, pages 601–608, Cambridge, MA, USA, 2006. MIT Press.

[24] Frank Hutter, Holger H. Hoos, and Kevin Leyton-Brown. Sequential model-based optimization for general algorithm configuration. In *Proceedings of the 5th International Conference on Learning and Intelligent Optimization*, LION'05, pages 507–523, Berlin, Heidelberg, 2011. Springer-Verlag.

[25] Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. Lightgbm: A highly efficient gradient boosting decision tree. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, pages 3149–3157, USA, 2017. Curran Associates Inc.

[26] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2015.

[27] Pang Wei Koh and Percy Liang. Understanding black-box predictions via influence functions. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 1885–1894, International Convention Centre, Sydney, Australia, 06–11 Aug 2017. PMLR.

[28] Himabindu Lakkaraju, Stephen H. Bach, and Jure Leskovec. Interpretable decision sets: A joint framework for description and prediction. In *Proceedings of the 22Nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, pages 1675–1684, New York, NY, USA, 2016. ACM.

[29] Jeff Larson, Surya Mattu, Lauren Kirchner, and Julia Angwin. How We Analyzed the COMPAS Recidivism Algorithm. https://www.propublica.org/article/how-we-analyzed-the-compas-recidivism-algorithm, May 2016.

[30] Benjamin Letham, Brian Karrer, Guilherme Ottoni, and Eytan Bakshy. Constrained bayesian optimization with noisy experiments. *Bayesian Analysis*, 06 2017.

[31] Benjamin Letham, Cynthia Rudin, Tyler H. McCormick, and David Madigan. Interpretable classifiers using rules and bayesian analysis: Building a better stroke prediction model. *CoRR*, abs/1511.01644, 2013.

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

[33] Cheng Li, Sunil Gupta, Santu Rana, Vu Nguyen, Svetha Venkatesh, and Alistair Shilton. High dimensional bayesian optimization using dropout. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI-17*, pages 2096–2102, 2017.

[34] Xuejun Liao, Ya Xue, and Lawrence Carin. Logistic regression with an auxiliary data source. In *Proceedings of the 22Nd International Conference on Machine Learning*, ICML '05, pages 505–512, New York, NY, USA, 2005. ACM.

[35] Michael Lim and Trevor Hastie. Learning interactions via hierarchical group-lasso regularization. *J Comput Graph Stat*, 24(3):627–654, 2015. 26759522[pmid].

[36] Zachary C. Lipton. The mythos of model interpretability. *Queue*, 16(3):30:31–30:57, June 2018.

[37] Yin Lou, Rich Caruana, Johannes Gehrke, and Giles Hooker. Accurate intelligible models with pairwise interactions. In *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '13, pages 623–631, New York, NY, USA, 2013. ACM.

[38] Gustavo Malkomes and Roman Garnett. Automating bayesian optimization with bayesian optimization. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems 31*, pages 5984–5994. Curran Associates, Inc., 2018.

[39] Lilyana Mihalkova and Raymond Mooney. Transfer learning with markov logic networks. In *Proceedings of the ICML-06 Workshop on Structural Knowledge Transfer for Machine Learning*, Pittsburgh, PA, June 2006.

[40] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. Distributed representations of words and phrases and their compositionality. In *Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'13, pages 3111–3119, USA, 2013. Curran Associates Inc.

[41] Alexander Munteanu and Chris Schwiegelshohn. Coresets-methods and history: A theoreticians design pattern for approximation and streaming algorithms. *KI - Künstliche Intelligenz*, 32(1):37–53, Feb 2018.

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

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

[44] John C. Platt. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. In *ADVANCES IN LARGE MARGIN CLASSIFIERS*, pages 61–74. MIT Press, 1999.

[45] Bob Price and Craig Boutilier. Implicit imitation in multiagent reinforcement learning. In *Proceedings of the Sixteenth International Conference on Machine Learning*, ICML '99, pages 325–334, San Francisco, CA, USA, 1999. Morgan Kaufmann Publishers Inc.

[46] J. Ross Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1993.

[47] J. Ross Quinlan. C5.0. https://rulequest.com/, 2004.

[48] Santu Rana, Cheng Li, Sunil Gupta, Vu Nguyen, and Svetha Venkatesh. High dimensional Bayesian optimization with elastic Gaussian process. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 2883–2891, International Convention Centre, Sydney, Australia, 06–11 Aug 2017. PMLR.

[49] Carl Edward Rasmussen. The infinite gaussian mixture model. In *Proceedings of the 12th International Conference on Neural Information Processing Systems*, NIPS'99, pages 554–560, Cambridge, MA, USA, 1999. MIT Press.

[50] Xiaofeng Ren and Jitendra Malik. Learning a classification model for segmentation. In *Proceedings of the Ninth IEEE International Conference on Computer Vision - Volume 2*, ICCV '03, pages 10–, Washington, DC, USA, 2003. IEEE Computer Society.

[51] Marco Túlio Ribeiro, Sameer Singh, and Carlos Guestrin. "why should I trust you?": Explaining the predictions of any classifier. *CoRR*, abs/1602.04938, 2016.

[52] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. Anchors: High-precision model-agnostic explanations, 2018.

[53] Tobias Scheffer, Christian Decomain, and Stefan Wrobel. Active hidden markov models for information extraction. In *Proceedings of the 4th International Conference on Advances in Intelligent Data Analysis*, IDA '01, pages 309–318, London, UK, UK, 2001. Springer-Verlag.

[54] Andrew I. Schein and Lyle H. Ungar. Active learning for logistic regression: An evaluation. *Mach. Learn.*, 68(3):235–265, October 2007.

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

[56] Burr Settles. Active learning literature survey. Computer Sciences Technical Report 1648, University of Wisconsin–Madison, 2009.

[57] Jasper Snoek, Hugo Larochelle, and Ryan P Adams. Practical bayesian optimization of machine learning algorithms. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 2951–2959. Curran Associates, Inc., 2012.

[58] Jasper Snoek, Oren Rippel, Kevin Swersky, Ryan Kiros, Nadathur Satish, Narayanan Sundaram, Md. Mostofa Ali Patwary, Prabhat Prabhat, and Ryan P. Adams. Scalable bayesian optimization using deep neural networks. In *Proceedings of the 32Nd International Conference on International Conference on Machine Learning - Volume 37*, ICML'15, pages 2171–2180. JMLR.org, 2015.

[59] F. Tanaka and M. Yamamura. Multitask reinforcement learning on the distribution of mdps. In *Proceedings 2003 IEEE International Symposium on Computational Intelligence in Robotics and Automation. Computational Intelligence in Robotics and Automation for the New Millennium (Cat. No.03EX694)*, volume 3, pages 1108–1113 vol.3, July 2003.

[60] Yee Whye Teh. *Dirichlet Process*, pages 280–287. Springer US, Boston, MA, 2010.

[61] Sebastian Thrun and Tom Michael Mitchell. Learning one more thing. In *IJCAI*, 1994.

[62] Lisa Torrey and Jude W. Shavlik. Chapter 11 transfer learning. 2009.

[63] Berk Ustun and Cynthia Rudin. Supersparse linear integer models for optimized medical scoring systems. *Machine Learning*, 102(3):349–391, Mar 2016.

[64] Ziyu Wang, Masrour Zoghi, Frank Hutter, David Matheson, and Nando De Freitas. Bayesian optimization in high dimensions via random embeddings. In *Proceedings of the Twenty-Third International Joint Conference on Artificial Intelligence*, IJCAI '13, pages 1778–1784. AAAI Press, 2013.

[65] Matthew D. Zeiler. ADADELTA: an adaptive learning rate method. *CoRR*, abs/1212.5701, 2012.

[66] Xiang Zhang, Junbo Zhao, and Yann LeCun. Character-level convolutional networks for text classification. In *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 1*, NIPS'15, pages 649–657, Cambridge, MA, USA, 2015. MIT Press.

APPENDIX

TABLE IV
DECISION TREE

dataset	oracle_type	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
cod-rna	density tree	0.67	-	0.34	1.84	0.00	0.00	0.70	0.00	**0.46**	**0.00**	-	-	-	-	-
	gbm (F1=0.94)	**16.53**	-	**0.97**	**3.15**	0.00	0.00	**1.11**	0.00	0.00	-	-	-	-	-	-
	rf (F1=0.93)	0.25	-	0.08	0.00	**0.04**	0.00	0.00	0.00	0.00	0.00	-	-	-	-	-
ijcnn1	density tree	**4.88**	**19.38**	**12.01**	**16.12**	3.42	**0.91**	1.15	0.57	0.00	0.00	0.19	0.00	0.00	0.00	-
	gbm (F1=0.89)	2.66	9.70	6.96	8.12	**4.82**	0.27	0.00	0.64	0.00	0.00	0.00	**1.29**	0.00	0.76	-
	rf (F1=0.87)	0.03	0.15	5.94	7.73	2.21	0.00	**2.19**	**1.19**	0.00	0.00	**1.36**	1.19	0.00	**0.91**	-
higgs	density tree	**7.11**	**1.91**	0.00	0.00	0.00	0.94	-	-	-	-	-	-	-	-	-
	gbm (F1=0.68)	4.42	1.24	0.56	**3.96**	**2.72**	**1.45**	-	-	-	-	-	-	-	-	-
	rf (F1=0.69)	5.41	0.00	**2.08**	1.20	0.66	0.39	**4.17**	-	-	-	-	-	-	-	-
covtype.binary	density tree	0.00	0.00	0.01	0.68	0.47	0.00	0.00	0.55	**1.89**	**1.44**	-	-	-	-	-
	gbm (F1=0.77)	0.00	0.00	0.30	1.17	0.45	0.00	0.00	0.00	0.00	-	-	-	-	-	-
	rf (F1=0.82)	**0.44**	0.00	**1.17**	**1.69**	**2.63**	**1.55**	0.00	**0.95**	0.00	-	**0.85**	-	-	-	-
phishing	density tree	0.00	0.71	0.00	0.00	0.46	0.00	0.08	0.00	0.00	0.00	0.00	0.00	0.00	-	0.00
	gbm (F1=0.94)	**0.00**	0.17	0.00	0.00	**0.71**	0.00	**0.47**	**1.03**	**0.76**	0.00	**0.40**	**0.58**	0.00	**0.00**	0.00
	rf (F1=0.95)	0.00	**0.99**	**0.98**	**0.32**	0.55	0.00	0.00	0.02	0.00	0.00	0.00	0.00	0.00	0.00	0.00
a1a	density tree	0.00	2.50	**7.23**	-	4.91	**3.06**	**4.75**	0.88	0.87	-	0.85	-	**2.35**	-	**0.92**
	gbm (F1=0.76)	0.00	**8.06**	3.19	**6.63**	**6.13**	0.67	1.54	-	2.11	**1.56**	-	-	-	-	-
	rf (F1=0.76)	**0.00**	6.27	3.68	4.84	3.76	0.37	2.48	**2.63**	**3.44**	-	**4.16**	-	-	-	-
pendigits	density tree	10.14	3.46	**3.04**	6.71	**5.97**	**4.21**	1.61	0.00	0.00	0.00	0.00	0.00	0.00	-	-
	gbm (F1=0.98)	11.52	0.50	1.81	**7.82**	2.88	0.74	1.56	0.00	0.00	0.00	0.00	0.00	0.00	**0.01**	-
	rf (F1=0.98)	**12.61**	**6.94**	0.00	5.55	4.71	3.95	**1.84**	**0.03**	**0.78**	0.00	0.00	0.00	0.00	-	**0.00**
letter	density tree	**0.18**	9.71	31.91	**36.26**	**30.03**	**17.73**	**6.76**	**3.65**	1.90	**1.95**	0.00	0.00	0.00	0.00	0.00
	gbm (F1=0.91)	0.00	**14.72**	**32.26**	29.35	27.70	11.19	5.43	0.65	0.71	0.01	0.00	0.00	0.00	0.00	0.00
	rf (F1=0.89)	0.15	9.23	31.65	29.69	23.28	11.43	6.06	3.64	**4.14**	0.44	0.00	0.00	0.00	0.00	0.00
Sensorless	density tree	0.00	41.80	**85.34**	**65.99**	**27.63**	**14.84**	**7.40**	**5.29**	**2.31**	**1.43**	0.77	**0.47**	-	**0.84**	-
	gbm (F1=0.99)	0.00	**54.23**	50.72	32.59	16.65	9.12	0.31	0.00	0.00	0.00	0.00	0.00	**0.00**	0.00	-
	rf (F1=0.98)	0.00	53.41	52.94	16.75	12.01	8.02	1.45	0.93	0.00	0.31	**0.87**	0.41	0.00	0.00	-
senseit_aco	density tree	18.69	0.51	**5.29**	1.98	1.44	**0.56**	-	-	-	-	-	-	-	-	-
	gbm (F1=0.73)	18.27	**2.00**	4.57	1.78	2.00	0.00	**0.00**	-	-	-	-	-	-	-	-
	rf (F1=0.73)	**20.17**	0.50	2.78	**6.84**	**3.26**	-	-	-	-	-	-	-	-	-	-
senseit_sei	density tree	1.97	0.68	1.56	0.09	1.30	**0.10**	-	-	-	-	-	-	-	-	-
	gbm (F1=0.70)	2.95	**4.04**	1.46	0.39	0.00	0.00	**4.18**	-	-	-	-	-	-	-	-
	rf (F1=0.67)	**3.71**	1.61	**2.17**	**1.74**	**2.57**	0.00	0.71	**0.43**	-	-	-	-	-	-	-
covtype	density tree	16.89	**125.49**	16.73	5.29	**10.83**	7.31	-	**5.91**	**8.18**	**8.15**	**11.49**	0.00	0.00	**1.34**	**2.06**
	gbm (F1=0.57)	47.39	114.26	32.81	**10.70**	5.99	0.19	0.00	0.00	0.00	1.49	0.12	**2.69**	**2.12**	0.00	0.00
	rf (F1=0.63)	**119.19**	65.61	**34.15**	1.03	2.09	**7.74**	**6.45**	3.25	0.00	0.00	0.00	0.00	0.00	0.00	0.00
connect-4	density tree	181.76	22.50	18.55	**25.13**	-	**23.79**	-	4.20	0.00	3.08	**2.21**	1.92	0.00	0.00	0.68
	gbm (F1=0.55)	**193.93**	29.86	**52.09**	7.30	**8.09**	2.03	**5.88**	**11.41**	**8.14**	**9.38**	1.06	5.83	**7.83**	**4.93**	**8.51**
	rf (F1=0.56)	184.31	**30.63**	14.02	-	0.86	0.00	0.00	2.37	4.58	7.68	0.00	**6.12**	0.24	0.00	0.00

An oracle performs better 63.50% of the time. $RMSE = 9.33$, computed with density tree score and best oracle score.