Trees with Attention for Set Prediction Tasks

Roy Hirsch [1] **Ran Gilad-Bachrach** [2]

Abstract

In many machine learning applications, each record represents a set of items. For example, when making predictions from medical records, the medications prescribed to a patient are a set whose size is not fixed and whose order is arbitrary. However, most machine learning algorithms are not designed to handle set structures and are limited to processing records of fixed size. *Set-Tree*, presented in this work, extends the support for sets to tree-based models, such as Random-Forest and Gradient-Boosting, by introducing an attention mechanism and set-compatible split criteria. We evaluate the new method empirically on a wide range of problems ranging from making predictions on sub-atomic particle jets to estimating the redshift of galaxies. The new method outperforms existing tree-based methods consistently and significantly. Moreover, it is competitive and often outperforms Deep Learning. We also discuss the theoretical properties of Set-Trees and explain how they enable item-level explainability.

1. Introduction

In this study, we focus on predictive tasks where each record contains a set of items. The number of items in a set may vary and predictions are expected to be independent of the items' ordering. Most modern machine learning algorithms are not designed for processing such structures, and very few works explicitly model them (Sutherland et al., 2012; Oliva et al., 2013; Zaheer et al., 2017). Problems where sets of items emerge abound in many fields, from particle physics (Qu & Gouskos, 2020) and cosmology (Rykoff et al., 2014) to statistics (Oliva et al., 2013) and computer graphics (Qi et al., 2017a). In this work, we present a novel tree-based algorithm for processing sets.

[1]Department of EE, Tel-Aviv University, Israel [2]Department of Bio-Medical Engineering, Tel-Aviv University, Israel and the Edmond J. Safra Center for Bioinformatics. Correspondence to: Roy Hirsch <royhirsch@mail.tau.ac.il>, Ran Gilad-Bachrach <rgb@tauex.tau.ac.il>.

Proceedings of the 38^{th} International Conference on Machine Learning, PMLR 139, 2021.

Tree-based models, such as Decision Tree (DT), Random Forest (RF) and Gradient Boosting Decision Tree (GBT) are broadly used in practice (Breiman et al., 1984; Breiman, 2001; Friedman, 2001). In a survey conducted by Kaggle among data scientists in 2020, RFs and GBTs were ranked as two of the three most commonly used ML algorithms (used by 78.1% and by 61.4% of the data scientists respectively) (Kaggle, 2020). They continue to serve as the state of the art in many tabular data predictive tasks (Sandulescu & Chiru, 2016; Munkhdalai et al., 2019; Desai et al., 2020a). Despite their ubiquity, relatively little progress has been made in extending trees to handle varying length inputs, with some exceptions for time series prediction tasks (Meek et al., 2002; Douzal-Chouakria & Amblard, 2012). Few recent works explicitly extend trees for sets, but these are restricted to categorical features and were designed for specific tasks, such as text classification (Lucena, 2020; Guillame-Bert et al., 2020).

Our method, Set-Tree, is a tree-based general framework for processing sets. We introduce set compatible split criteria and an attention mechanism that allows applying them to subsets of the input set which we refer to as *attention-sets*. In a theoretical analysis we show that composing split-criteria that are possibly applied over different attention-sets allows expressing complex set-functions. Learning from unordered sets is attracting growing attention in the field of Deep Learning (DL). Recent studies have focused on the theory of learning from sets (Segol & Lipman, 2019; Zaheer et al., 2017) or specifically designed for computer vision or computer graphics problems (Li et al., 2018; Qi et al., 2017b). We empirically prove that GBeST, a gradient boosting algorithm applied over Set-Trees, is competitive and often outperforms DL solutions. Moreover, it provides *item-level* explainability that allows quantifying the relative importance of each item in the input set.

Our contribution can be summarized as follows: (1) Set-Tree, a new framework based on attention and set-compatible split criteria, that allows applying tree-based models to problems defined over sets. (2) A theoretical analysis of Set-Tree. (3) Empirical evaluation in diverse domains that shows that Set-Tree consistently outperforms other tree-based approaches and, in most cases, is on a par or better than DL-based methods. (4) Introduce a new, item-level explainabiity method, derived from the attention-sets.

2. Related Work

Tree-based learning. Introduced by Breiman et al. (1984), decision trees have become widespread in various domains (Rodriguez-Galiano et al., 2015; Krauss et al., 2017; Zhang et al., 2017; Han et al., 2018; Munkhdalai et al., 2019; Desai et al., 2020b). Tree learning is a greedy process that sequentially picks split criteria based, in most cases, on individual features, to decrease a loss function. To enhance the performance of individual trees, a common approach is to employ ensemble methods such as Random Forests (Ho, 1998; Breiman, 2001) and Gradient Boosting Trees (Friedman, 2001; 2002; Friedman & Meulman, 2003). Gradient Boosting packages such as Catboost (Prokhorenkova et al., 2018), LightGBM (Ke et al., 2017) and XGBoost (Chen & Guestrin, 2016) are highly popular within machine learning practitioners. Despite the popularity and effectiveness of tree-based methods, they are applicable to tabular data alone with a very few exceptions.

Decision tree extensions. Several studies have extended decision trees to handle inputs of varying lengths, most of which are devoted to time series classification (González & Diez, 2000; Meek et al., 2002; Douzal-Chouakria & Amblard, 2012; Guillame-Bert & Dubrawski, 2017). Recently, several scholars have proposed methods to support *categorical-sets*, a setup where the input features are categorical and take discrete values from a known vocabulary (Lucena, 2020; Guillame-Bert et al., 2020). The very recent work of Guillame-Bert et al. (2020) defines a categorical-set split criterion for text classification. This work supports data in the form of sets, but only sets of categorical features such that $\mathbf{S} \subseteq \mathbb{S}$ where $\mathbb{S}$ is a finite set of non-ordinal objects. In a sense, Set-Tree extends the capabilities of their algorithm to support sets of real valued vectors.

Multi Instance Learning (MIL). In MIL setup, labels are assigned to *bags* of items (sets) where it is assumed that the bag's prediction is an aggregation of its items' predictions (Dietterich et al., 1997). The vast majority of tree-based MIL methods are based on item-level prediction (Melki et al., 2018; Zucker & Chevaleyre, 2000; Leistner et al., 2010). These methods are incapable of learning features derived from the whole bag, or from its subsets. BLRT (Komárek & Somol, 2018) operates directly on the entire bag. Set-Tree generalizes BLRT's split criterion, and can potentially learn more complex tasks using the attention-sets.

Permutation-invariance in deep learning. The seminal work of Zaheer et al. (2017) defines a framework for learning neural-networks that respect set symmetries. This work was extended to incorporate attention mechanism by Lee et al. (2019) and by others (Segol & Lipman, 2019; Maron et al., 2020). Another line of research proposes set-processing methods for point clouds (Qi et al., 2017a;b; Li et al., 2018; Xu et al., 2018; Liu et al., 2019).

3. Notation and Motivation

In this work we consider classification and regression problems where the input record is a finite multi-set $\mathbf{S} = \{x_1, x_2, \ldots, x_n\}$ such that $x_i \in \mathbb{R}^d$ and $n = |\mathbf{S}|$. A function f is a set-function if it is defined for sets of any finite size and is *permutation-invariant* in the sense that for every permutation π: $f(\{x_1, x_2, \ldots, x_n\}) = f(\{x_{\pi(1)}, x_{\pi(2)}, \ldots, x_{\pi(n)}\})$. We note that it is possible to extend the work presented here to settings in which each record consists of several sets or contains non-set features along with set features. However, for the sake of brevity we consider only the single set setting.

Our goal is to learn set-functions from the data. To achieve this we focus on learning trees. Trees are comprised of split nodes, each of which performs a prediction on a single feature. A record traverses through the tree's nodes following the predictors' results until it reaches an end node, a leaf, whose value is the model's prediction. For a tree to be a set-function the split criteria should be a set-functions.

Section 4.2 introduces a class of split criteria that are set-functions. However, many natural set-functions cannot be represented by trees using these split criteria unless we also allow them to apply to subsets of the set $\mathbf{S}$. For example, imagine the case where $\mathbf{S}$ is a set of real numbers and the function f evaluates to True if the number of positive items in $\mathbf{S}$ is at least 10 and False otherwise. To evaluate this function using a tree, it should be able to locate the subset of positive items and be able to count them. Therefore, we denote by $A(\mathbf{S}) \subseteq \mathbf{S}$ the *attention-set* to which the split criterion is applied. In the problem introduced earlier a node in the tree can define the attention-set that consists of all positive numbers and another node can count the items in this set. To preserve the tree as a set-function, we require that the attention-sets be defined by set-functions. That is, the inclusion/exclusion criterion for an element in $\mathbf{S}$ to belong to $A(\mathbf{S})$ must be invariant to the permutations of $\mathbf{S}$.

4. Set-Tree

Since vanilla decision trees cannot handle sets, a common practice is to use feature augmentation (Ramoser et al., 2000; Bevan et al., 2014). For example, it is possible to extract values for each feature in the set, such as the minimum, maximum, average, sum, and etc., and apply the tree using these features. However, as demonstrated previously, feature augmentation alone creates trees that are limited in their descriptive power. The mechanism of attention we introduce addresses this issue.

4.1. Attention

Internal nodes of trees contain split criteria which control the traversal of the tree. In most manifestations of trees

the split criterion is a binary function that operates on a single feature. In Set-Tree, the criterion is a triplet $(c, A, \bar{A})$. Similar to vanilla trees, c is a binary function that controls the traversal of the tree. Unlike vanilla decision trees, in Set-Trees the binary function operates over sets. For example, this function can apply the max operator to the value of the j'th feature in all items in the set and compare it to a threshold θ. Alongside with the function c, the criterion defines a function A that operates on the input set and returns a subset of its items. Intuitively speaking, the resulting subset contains the most significant items in the input that trigger the function c to be True. For example, when using c in the example above, the set A would consist of all items whose j'th feature is $\geq \theta$. $\bar{A}$ is the complementary set to A.

The tree learning algorithm assigns to each node a split criterion as well as the attention-set to which the criterion should be applied. The attention-set can be either the original set or a subset of this set generated by the function A or $\bar{A}$ in one of the nodes in the path leading to the current node. Note that this creates a dependency between the nodes. For example if node 1 is evaluated on the original set $\mathbf{S}$ and generates the attention-set $A_1(\mathbf{S})$ and node 2 uses this attention-set then it generates the attention-set $A_2(A_1(\mathbf{S}))$. Attention serves to specify dependencies that cannot be specified using feature augmentation.

4.2. Set Compatible Split Criteria

The attention mechanism can operate with diverse split criteria. Here we propose a family of criteria inspired by the idea of sample moments in statistics. Given a set $\mathbf{S}$ we define the decision criterion to be:

$$|\mathbf{S}|^{-\beta} \sum_{x \in \mathbf{S}} (x_j)^{\alpha} \geq \theta \tag{1}$$

Where $\alpha \in [-\infty, \infty]$ controls the degree of the moment, $\beta \in \{0, 1\}$ controls whether to normalize by the size of the set, j points to the index of the feature to be used for the evaluation and θ is a threshold. The criterion defined in (1) generalizes several familiar functions. When $\alpha = 1$ and $\beta = 0$ the criterion uses the sum, whereas when $\alpha = 1$ and $\beta = 1$ the average is used. When $\alpha = \beta = 0$ the size of the set is used. With $\alpha = -1$ and $\beta = 1$ the inverse of the harmonic mean is used.

When α is finite and $\neq 0$ we can take the power of $1/\alpha$ from both sides of (1) to rewrite the criterion as $\left(|\mathbf{S}|^{-\beta} \sum_{x \in \mathbf{S}} (x_j)^{\alpha}\right)^{1/\alpha} \geq \theta$. We use this to also define the split criterion in the limits. For $\alpha = \infty$ we define the split criterion to be the maximal value and for $\alpha = -\infty$ it is the minimum. For $\alpha = 0$ and $\beta = 1$ it is the geometric mean.

There are certain cases in which the split criterion is not well defined. These include the cases when the set is empty or when $(x_j)^{\alpha}$ is not a real number. In these cases we define the L.H.S. of (1) to be smaller than the R.H.S. regardless of the value of θ (meaning, the condition is evaluated as False). As discussed in Section 4.1, in Set-Trees every split criterion should also be accompanied by a function A that returns the attention-set. For the criterion defined here we define the function A as follows:

$$A(\mathbf{S}) = \left\{ x \in \mathbf{S} \ : \ (x_j)^{\alpha} \geq \frac{\theta}{|\mathbf{S}|^{1-\beta}} \right\} \tag{2}$$

The rationale behind this definition is that if every $x \in \mathbf{S}$ is such that $(x_j)^{\alpha} \geq \theta / |\mathbf{S}|^{1-\beta}$ then $|\mathbf{S}|^{-\beta} \sum_{x \in \mathbf{S}} (x_j)^{\alpha} = \theta$. For example, for $\alpha = 1$ and $\beta = 0$ the split criterion tests if the sum $\geq \theta$ and $A(\mathbf{S}) = \{x \in \mathbf{S} \ : \ x_j \geq \theta / |\mathbf{S}|\}$ while if $\beta = 1$ the split criterion tests whether the average $\geq \theta$ and $A(\mathbf{S}) = \{x \in \mathbf{S} \ : \ x_j \geq \theta\}$.

4.3. Implementation Details

Parameters: using attention and the split criterion described above, every node in the tree is parameterized by the tuple $(\alpha, \beta, j, \theta, h, p)$. The decision function (Eq.1) is characterized by α, β, j and θ, while h and p control the node's attention-set. h points to one of the nodes in the path to the current node (by convention, if this pointer is null then the entire set is used) and p is the boolean polarity signal that indicates whether A or $\bar{A}$ should be used.

Optimization: tree-learning algorithms are based on exhaustive search for the optimal nodes' parameters. The parameters (α, β, h, p), introduced by Set-Tree, add complexity to the learning process. β and p are boolean variables, in practice we limit the number of nodes scanned along the path to N_h and the number of scanned α, β pair values to $N_{[\alpha,\beta]}$. This creates an overhead factor of $2N_{[\alpha,\beta]}(N_h + 1)$ compared to vanilla decision tree. This overhead can be further reduced by caching and by sampling techniques. More details about running time and acceleration techniques are discussed in the supplementary material. Moreover, since Set-Tree differs from vanilla tree only by the decision rule, it can be combined with any existing tree-learning algorithm (CART (Breiman et al., 1984), ID3 (Quinlan, 1986), ...) or ensemble method.

We release an implementation of our proposed method for the community and for reproducibility. The code is available at: `https://github.com/TAU-MLwell/Set-Tree`.

4.4. Theoretical Properties

The following theorem shows that if we restrict the domain of a set function, then Set-Trees can express any set-function.

Theorem 1. *Let $\mathbf{R} \subseteq \mathbb{R}$ be a finite set. Let f be a set-function defined over $\mathbf{R}$. For every M there exists a Set-Tree*

T such that for every set $\mathbf{S}$ of size at most M from $\mathbf{R}$ it holds that $T(\mathbf{S}) = f(\mathbf{S})$.

Note that throughout the paper the term "set" refers to multi-sets; that is, each item may appear multiple times. Therefore, when we measure the size of a set we count all the items and their repetitions. It should be noted that Set-Trees are not limited to expressing the set-functions described in Theorem 1. Rather, this theorem shows that Set-Trees can express a rich collection of set-functions.

Proof. Let $\mathbf{R} = \{r_1, \ldots, r_m\}$ be a finite set. We can identify a set over $\mathbf{R}$ with $(r_1^{\times n_1}, \ldots, r_m^{\times n_m})$ where n_i is the number of times the item r_i appears in the set and therefore the size of the set is $\sum n_i$.

For a set over $\mathbf{R}$, let f be a set-function. W.l.o.g. we can assume that for every $\mathbf{S}$ such that $|\mathbf{S}| > M$ it holds that $f(\mathbf{S}) = 0$. Therefore, we can write f as:

$$f(\mathbf{S}) = \sum_{n_1,\ldots:\sum n_i \leq M} f((r_1^{\times n_1}, \ldots, r_m^{\times n_m})) \prod_i I_{N_i(\mathbf{S})=n_i}$$

where $I_{N_i(\mathbf{S})=n_i}$ is the function that returns 1 if the number of copies of r_i in $\mathbf{S}$ is n_i and returns 0 otherwise.

Set-Trees are closed under addition and multiplication (proof provided in the supplementary material) and therefore, to complete the proof it is sufficient to show that there is a tree that computes the function $I_{N_i(S)=n_i}$. Let δ be the minimal distance between two items in $\mathbf{R}$. Consider a tree with the following 4 decision nodes: [Node 1]: $\max(\mathbf{S}) \geq r_i - \delta/2$. [Node 2]: $\min(A_1(\mathbf{S})) < r_i + \delta/2$ where $A_1(\mathbf{S})$ is the attention-set generated by Node 1. [Node 3]: $size(A_2(A_1(\mathbf{S}))) > n_i - 0.5$ where $A_2(A_1(\mathbf{S}))$ is the attention-set generated by Node 2. [Node 4]: $size(A_2(A_1(\mathbf{S}))) < n_i + 0.5$.

All the leaves of this tree are assigned the value 0 with the only exception being the leaf that is reached if all the decision nodes evaluate to True. This node is assigned the value of 1. It is easy to see that the first two nodes select only the elements in the set that have the value r_i and the latter nodes will both evaluate to True only if the number of items in this selection is n_i. Therefore, this tree computes the function $I_{N_i(\mathbf{S})=n_i}$ which completes the proof. □

However, in the most general case, without additional restrictions, Set-Trees cannot express every set-function.

Theorem 2. *Let $\mathbb{C}$ be the class of circuits that operate on sets and evaluate to a Boolean value such that every gate in this circuit has a polynomial computational complexity with respect to the size of the set and the number of features in each element in the set. If $P \neq NP$ then there exists a set function f that cannot be computed by any circuit in $\mathbb{C}$.*

Since a Set-Tree is a circuit in $\mathbb{C}$, defined in the theorem, and also every ensemble of Set-Trees is a circuit in that class, this implies that there are set-functions that cannot be expressed by Set-Trees or ensembles of such trees. Full proof is provided in the supplementary material.

5. Empirical Evaluation

In this section, we examine the applicability of Set-Trees to various tasks. We consider synthetic and real-world tasks from diverse domains and a variety of objective functions (see Table 1 for a quick summary of these tasks). We primarily focus on comparing Set-Trees to tree-based algorithms but include comparison to deep-learning algorithms too. We describe here the main algorithms used.

Gradient Boosted Set Tree (GBeST): gradient boosting applied over Set-Trees. We limit the subset of (α, β) pairs which correspond to the following group of operators: $\mathbf{O}$ ={*max, min, mean, sum, harmonic mean, geometric mean, second moment mean*}. To further reduce the computational complexity, the attention-set is either the entire set or one that was generated in the last 5 nodes leading to the current node.

Gradient Boosting Tree (GBT): gradient boosting applied over vanilla trees using XGBoost (Chen & Guestrin, 2016). Features augmentation is used to allow prediction over sets. We use $\mathbf{O}$, the same group of operators as in GBeST, to represent the set. This baseline can also be viewed as an ablated version of Set-Tree, without using the attention-sets.

For all the tree-based models (GBT and GBeST) and for all the tasks, we scanned a pre-defined hyperparameter search space. we used 10% of the training data as validation for the hyperparameters tuning. The trees' maximal depth was chosen within $\{5, 6, 8, 10\}$, the number of estimators was chosen within $\{50, 100, 200, 300\}$ and the learning rate was chosen within $\{0.2, 0.1, 0.05\}$. We also applied known tree regularization techniques, the fraction of train records sampled per tree was chosen within $\{1, 0.8, 0.5\}$ and the fraction of features sampled per tree was chosen within $\{1, 0.8, 0.5\}$ (where 1 is using all the records).

DeepSets: (Zaheer et al., 2017) a popular deep learning framework for learning functions on sets. The framework is made up of two sub-networks. The first is a network ϕ that operates on each element of the set independently. The second is a network ρ that operates on the sum of the outputs of ϕ such that the entire network computes $\rho(\sum_{x \in \mathbf{S}} \phi(x))$. In the experiments ϕ and ρ are Multi Layer Perceptrons (MLPs) (Hastie et al., 2009)). This model can be further extended when using different set-compatible aggregation operators. We scanned the following operators {*sum, max, mean*} and report their results.

Table 1. Summary of experiments: **# Items**: number of items in a set. **# Features**: number of features in an item. **Task**: 'BC' - binary classification, 'MC' - multi-class classification, 'R' - regression.

Experiment	# Items	# Features	Task
First quadrant	20-300	2/100	BC
Drug prescriptions	2-100	18	B/MC
Redshift	24-309	17	R
Jets	1-99/1-148	7	BC
Two-sample	4-50	1	BC
Point clouds	100	3	MC
Poker hands	5	2	B/MC

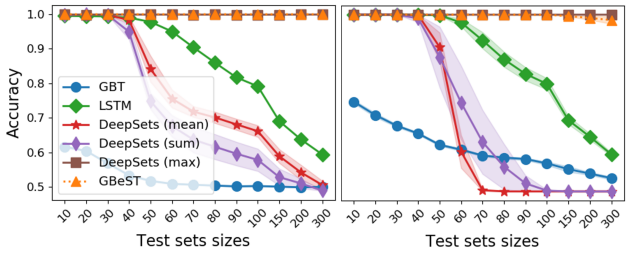


Figure 1. First quadrant classification accuracy, *Left*: results for sets of 2D points, *right*: results for sets of 100D points. All models were trained with a constant set size of $n = 20$ and tested with varying sets sizes. Shaded area represents the standard deviation.

We also compare to other relevant baselines that are tuned for specific tasks, these are described separately for each experiment. All DNN-based models were trained using Adam optimizer (Kingma & Ba, 2014) and a learning rate of 1e-3. We used early stopping while monitoring the validation loss with a patience of 3 epochs. Unless specified otherwise, we used 10% of the training data as validation for hyperparameters tuning. The test results for the models with the best validation performances are reported. Each experiment was repeated five times, we report the mean metrics and standard deviations for all of the experiments. The supplementary material contains more details about the data processing and training protocols for all of the experiments.

5.1. First Quadrant

We tested the item-level discriminative power of the different methods using a synthetic anomaly detection task. We uniformly sampled sets of items from the unit cube $[-1, 1]^D \subset \mathbb{R}^D$. Positive and negative sets differed in that in negative sets all items in the 1^{st} quadrant were removed (items where all coordinates are positive), while for the positive sets exactly one item in the 1^{st} quadrant was kept. Positive and negative sets were sampled evenly. The training set consisted of $100K$ records, each with 20 items. We tested over $10K$ test records with various set sizes ($n \in \{10, 20, 30, ..., 100, 150, 200, 300\}$). We ran the experiments with two configurations where $D = 2, 100$.

In this experiment, LSTM Neural-Networks were used as additional benchmark (Hochreiter & Schmidhuber, 1997). The results, presented at Figure 1, show that GBeST significantly outperformed the of the tested baselines and is on par with DeepSets (max). It achieved perfect accuracy in the $2D$ case. In the $100D$ case it achieved perfect accuracy when the test set size was < 150 there was a slight degradation of accuracy. GBT, in comparison, failed to achieve an accuracy > 0.7 on any sets size. LSTM and DeepSets with sum and mean operators had perfect accuracy for small set sizes (< 50) but as the set sizes increased their accuracies dropped.

5.2. Drug Prescription Errors

Next, we considered the task of detecting drug prescription errors. Prescription errors are estimated to account for 1 out of 854 inpatient deaths in the U.S. (Kohn LT, 2000). Recent studies claim that machine-learning holds promise for improving drug prescription errors detection (Segal et al., 2019; Rozenblum et al., 2020).

Data were generated using the MIMIC-III database (Johnson et al., 2016), a large database containing medical records of patients admitted to critical care units. We randomly split the $39,360$ patients into train and test ($90\%/10\%$). Each patient could have multiple hospital stays, we only considered hospital stays where 2-100 drugs were prescribed and ended up with $49,811$ stays. We excluded drugs that were prescribed fewer than 10 times or more than $10K$ times. Each stay was converted into one positive record and one negative record. The positive records contained the set of drugs prescribed during the hospital stay. The negative records contained the same set of drugs but n_{neg} of them were replaced by uniformly sampled ones from the catalog. We considered two setups: *Single Item Replacement (SIR)* where $n_{neg} = 1$, and *Multiple Items Replacement (MIR)* where n_{neg} uniformly sampled from $\{1, 2, 3, 4, 5\}$.

Each item (drug) in the set was represented as a 18-dimensional vector corresponding to the 18 top-level categories of the International Classification of Diseases code (ICD-9). We used the fact that every hospital stay is associated with an ICD-9 code and counted the number of times each drug in the training set was prescribed for each of the 18 diagnosis categories. These 18 counts are normalized to sum to one and are used as the feature vector for the drug.

We compared GBeST to GBT, DeepSets (Zaheer et al., 2017) and against bidirectional recurrent neural networks: BiLSTM and BiGRU (Huang et al., 2015) (which outperformed unidirectional recurrent neural networks in this task). Due to space limitations more details are deferred to the supplementary material. Table 2 summarizes the results

Table 2. Mean and std. of test AUC for drugs prescription errors prediction. We consider two setups: Single Item Replacement (SIR), and Multiple Items Replacement (MIR).

Model	*SIR* AUC	*MIR* AUC
BiLSTM	$.804 \pm .005$	$.884 \pm .006$
BiGRU	$.793 \pm .01$	$.896 \pm .02$
DeepSets (sum)	$.804 \pm 0.01$	$.914 \pm .06$
DeepSets (mean)	$.809 \pm 0.06$	$.910 \pm .06$
DeepSets (max)	$\mathbf{.845 \pm 0.02}$	$.916 \pm .03$
GBT	$.791 \pm .003$	$.898 \pm 0.002$
GBeST	$.825 \pm .002$	$\mathbf{.922 \pm .003}$

for the experiment. GBeST outperforms GBT by a margin for the two setups. GBeST outperfoms GBT and the DL contenders by $1-3$ percentage points for the *MIR* setup, DeepSets (max) outperfoms GBeST by 2 points for the *SIR* setup. A critical component of our model, in compare to DeepSets, is its natural interpretability. In Section 6 a way to use the attention mechanism to make the predictions of GBeST intelligible is presented. In medical applications, this property is of particular importance.

5.3. Redshift Estimation in Galaxies Clusters

Cosmological redshift is a physical phenomenon where wavelengths become longer when observing objects that are moving apart. Redshift is used to measure cosmological objects' recession velocity and distance. Two sources of observations are used to measure redshift: photometric and spectroscopic. The latter is assumed to provide a more accurate estimate, but is more expensive to acquire. We considered the problem of estimating a galaxy's redshift from photometric measurements using regression models (Connolly et al., 1995). Since galaxies are clustered (Rykoff et al., 2014), information about the cluster can be used to improve the predictions. For each galaxy, 17 photometric features were collected from the RedMaPPer DR8 catalog (Rykoff & S., 2014). The catalog contains photometric measurements for 70,505 galaxies and their corresponding spectroscopic redshift in 26,111 clusters of 24 to 309 galaxies.

To measure the contribution of cluster information, two experiments were conducted. In the first, cluster information was not used and every record was a single galaxy, a vector in $\mathbb{R}^{17}$. We refer to this experiment as the *single* setup. In the second experiment cluster information was provided, and a record was a set of galaxies' representations. We refer to this experiment as the *cluster* setup. In the cluster setup, we added a binary feature for every galaxy that was set to 1 for the galaxy whose redshift was to be predicted and 0 for any other galaxy in the cluster.

We compared GBeST to GBT, DeepSets, MLP model presented by Zaheer et al. (2017) and a to a regression model presented by Connolly et al. (1995). DeepSets with sum operator is the only configuration to which we present results, as the other configurations yielded inferior results. The data were randomly split into 90% train and 10% test and the models were optimized to minimize the square-loss. Table 3 presents the average square-loss on the test set and the average scatter where scatter is $|z_{spec}-z_{pred}|/(1+z_{spec})$, where z_{spec} is the accurate spectroscopic measurement and z_{pred} is the model's photometric estimation. The avg. scatter of GBeST was the smallest, 0.0147, compared to 0.023 for DeepSets (a reduction of 34%). It is interesting to note that the tree-based models outperformed DL models even in the single setup in which no clusters information was provided (avg. scatter of 0.0190 compared to 0.026 for MLP - a reduction of 27%).

Table 3. Results for the redshift regression problem. The *single* experimental setup did not use cluster information whereas the *cluster* setup did use this additional information. Results for MLP and DeepSets are taken from (Zaheer et al., 2017) and the results for Regression model are taken from (Connolly et al., 1995).

Setup	Model	MSE	Avg. Scatter
Single	**Regression**	-	0.025
	MLP	-	0.026
	GBT	0.002 (4e-5)	**0.0190 (4e-3)**
Cluster	**DeepSets**	-	0.023
	GBT	0.001 (6e-5)	0.0157 (3e-3)
	GBeST	0.001 (5e-5)	**0.0147 (4e-3)**

Table 4. Jets classification results for two tasks: *Quark-Gluon tagging* and *Top Tagging*. The reported results for PFN and ParticleNet are from (Qu & Gouskos, 2020).

	Quark-Gluon		*Top tagging*	
Model	**Acc.**	**AUC**	**Acc.**	**AUC**
PFN	-	0.8911	-	0.9819
ParticleNet	**0.826**	**0.8993**	**0.937**	**0.9844**
GBT	0.7823	0.8559	0.8459	0.9199
GBeST	0.8117	0.8863	0.9226	0.9765

5.4. Jets Tagging

A jet is a narrow cone of scattered particles produced in a collision event that occurs in particle collider. Jets are measured in particle detectors and studied to determine the properties of sub-atomic particles. One of the most important questions about a jet is what kind of elementary particle initiates it (known as jet tagging). Many deep-learning-based solutions have been proposed for this task (Cogan et al., 2015; Almeida et al., 2015; Guest et al., 2016; Louppe et al., 2019). Recent works have suggested modeling the data as sets (Komiske et al., 2019; Qu & Gouskos, 2020), an alternative that is more natural for this data structure.

We used two popular jet classification datasets: *Quark-Gluon tagging* (Komiske et al., 2019) and *Top Tagging* (Kasieczka et al., 2019). The two are synthetic datasets generated by particle physics simulation software and are widely used. The *Quark-Gluon tagging* dataset is comprised of jets initiated by quarks and by gluons. The number of particles in each jet varies from 1 to 148 and each particle is characterized by a four-momentum vector. The data were split into 1.6M/200k/200k records for train/validation/test. The *Top Tagging* dataset included jets derived from hadronically decaying top quarks. It contained 2M jets with 1-99 particles. The data were split into 1.2M/400K/400K for train/validation/test. Each particle was characterized by seven features following Qu & Gouskos (2020).

Table 4 presents the results of these experiments. GBeST and GBT are compared to two DL-based solutions: Particle Flow Network (PFN) (Komiske et al., 2019) which is derived from DeepSets, and ParticleNet (Qu & Gouskos, 2020) which is derived from Dynamic Graph Convolutional Neural Network (Wang et al., 2019). GBeST outperformed GBT baseline and was within 1 percentage point of the DL solutions that were optimized for these specific tasks.

5.5. Two-Sample Hypothesis Testing

The next experiment used the two-sample hypothesis testing task. Here, a record was formed from two subsets of i.i.d. items, and the task was to identify whether the two subsets were drawn from the same distribution or not. We added to each item an additional binary feature that marked whether this item belonged to the first or the second subset.

We experimented with a few variants of this task (1) **Different distributions families**: the subsets were drawn from *Normal(0, 1)* or from *Laplace(0, 1)*. In half of the cases, the two subsets were drawn from the same distribution, and in half of the cases the two were drawn from the two different distributions. Given a record, the task was to predict whether the same distribution was used in both subsets or not. (2) **Normal distribution with different means**: The two subsets were drawn from *Normal(μ_1, 1)* and *Normal(μ_2, 1)*. μ_1 and μ_2 were uniformly sampled from $[0, 1]$. For half of the records, μ_1 equaled μ_2, and the task was to discriminate whether the two subsets shared the same mean. (3) **Normal distribution with different variances**: items were drawn from *Normal(0, σ_1^2)* and *Normal(0, σ_2^2)*, the task was to discriminate whether the two subsets shared the same σ^2. σ_1^2 and σ_2^2 were uniformly sampled from $[0, 1]$. We repeated each variant several times with different numbers of items in each set. Each experiment involved $50K$ train and $5K$ test records.

Figure 2 presents the test predictions' accuracies of GBeST, GBT and DeepSets. For brevity we only report the results for the best performing DeepSets configurations (using sum, mean and sum aggregation operations for the three tasks accordingly). GBeST consistently outperformed GBT and DeepSets on all tasks and all set sizes. The differences were more pronounced as sets grew in size. When sets consisted of 50 items, GBeST was $7-12$ percentage points more accurate than DeepSets.

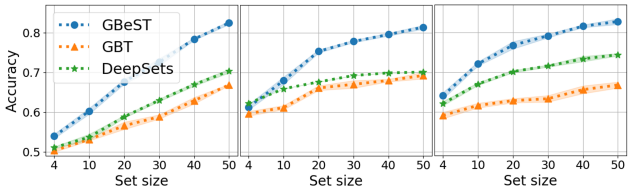


Figure 2. Results for two-sample hypothesis testing for varying set sizes. Each point is a model trained with a different number of items within a set. *Left*: different distributions families (*Normal(0, 1)* and *Laplace(0, 1)*). *Center*: normal distributions with different means. *Right*: normal distributions with different variances

Table 5. Poker hands test classification accuracy. Results for the binary classification setup (**BC**) and for the multi-class setup (**MC**). We report results for two GBT baselines, a setup where we preprocessed the data and extracted its statistical moments (GBT) and a setup with the flattened raw features (GBT-flat).

Model	BC Acc. (%)	MC Acc. (%)
MLP	49.1 ± 0.8	49.8 ± 0.3
DeepSets (sum)	66.3 ± 0.2	60.2 ± 0.1
DeepSets (mean)	66.9 ± 0.1	55.6 ± 0.1
DeepSets (max)	62.2 ± 0.2	67.5 ± 0.5
GBT-flat	85.9 ± 0.7	75.9 ± 0.3
GBT	90.6 ± 0.2	88.1 ± 0.2
GBeST	**100**	$\mathbf{98.9 \pm 0.1}$

5.6. Point Clouds Classification

Point cloud is a set of points in space that represent a 3D shape. Point clouds are extensively used in computer graphics and robotics as a compact way to represent 3D objects. We experimented with a multi-class point cloud classification task based on the ModelNet40 dataset (Wu et al., 2015). The task consisted of $9,843$ training and $2,468$ test instances belonging to 40 classes of objects. Each instance was a set of $10K$ points, where each point was a 3D vector (x, y, z). Following Zaheer et al. (2017) 100 points were sampled from the $10K$ points for each instance. GBeST achieved an accuracy of $72.8 \pm 0.5\%$ and outperformed GBT ($68.7 \pm 0.7\%$). However DeepSets significantly outperformed GBeST with an accuracy of $82 \pm 2\%$. We acknowledge the superiority of DeepSets over our method, however it is important to note that GBeST results are not trivial. As far as our knowledge goes, this is the first attempt at processing raw point clouds using a tree-based models.

5.7. Poker Hands Classification

In this task each record was a poker hand consisting of 5 cards drawn from a standard 52 card deck. We used the poker hands dataset introduced in Cattral et al. (2002) which contains $25K$ train records and $1M$ test records. Each card is represented by 5 features: a one-hot-encoding of the card's suit and another feature indicating the value of the card in the range of 1 to 13. A 10 class classification of the type of hand is defined (empty-hand, one-pair, two-pairs,...). We also derived a binary classification task of predicting whether the hand was empty or non-empty.

We compare GBeST to GBT and DeepSets with different configurations. Since the set size is fixed and small we also compared to a MLP baseline, from Yang et al. (2018). We also compared to a GBT baseline that was trained over the flattened fixed-size features (without the statistical moments extraction, we named this setup GBT-flat). The results are presented at Table 5. GBeST achieved perfect results for the binary setup and near-perfect results (98.9%) for the multi-class case. GBT achieved an accuracy of $90.6\%/88.1\%$ in the two setups whereas the best performing DeepSets achieved $66.9\%/67.5\%$ and the MLP achieved only $49.1\%/49.8\%$.

6. Item-Level Explainability

When the input record is a set of items, an important question arises: *which items make a significant impact on the prediction?* Set-Trees enable this item-level explainability by measuring the frequency each item occurs in the attention-sets. By examining the items' importance in different problems, we show that they provide useful insights and help in explaining the models' predictions.

First we rank the items according to their frequency of occurrence on a single Set-Tree's attention-sets. For a set $\mathbf{S}$, let $A_1, \ldots, A_k$ be all the attention-sets in the path of $\mathbf{S}$ when evaluated by the tree. Let $\text{rank}(x|\mathbf{S}) = r$ if $x \in \mathbf{S}$ is the r most frequent item to appear in the attention-sets. I.e. if $c(x) = \#\{A_i \ : \ x \in A_i\}$ and assume that $\text{rank}(x)$ is the position of $c(x)$ in the sorted list of $[c(x') \ : \ x' \in \mathbf{S}]$ (in descending order).[1]

When acquiring ranks from an ensemble of trees, we first compute the ranks for each tree individually. We then weight the rankings based on the importance of each tree. Let $\text{rank}_m(x|\mathbf{S})$ be the rank of x in the m'th tree in the ensemble. Since the magnitude of leaf values may be different between trees, especially when gradient boosting is used (Vinayak & Gilad-Bachrach, 2015), the ranks are weighted according to the leafs values. Let v_m be the prediction that the m'th tree assigns to $\mathbf{S}$. We define the importance score that the ensemble model assigns to $x \in \mathbf{S}$ as:

$$\text{Importance}(x|\mathbf{S}) = \frac{\sum_m |v_m| 2^{-\text{rank}_m(i|\mathbf{S})}}{\sum_{x' \in \mathbf{S}} \sum_m |v_m| 2^{-\text{rank}_m(x'|\mathbf{S})}}$$

such that the importance is non-negative and sums to 1.

The importance score can be applied to any type of data and visualized according to the task and data type. In the following we present applications of the importance score to a few of the experiments detailed in Section 5 (Figure 3 presents corresponding visualizations to most of them).

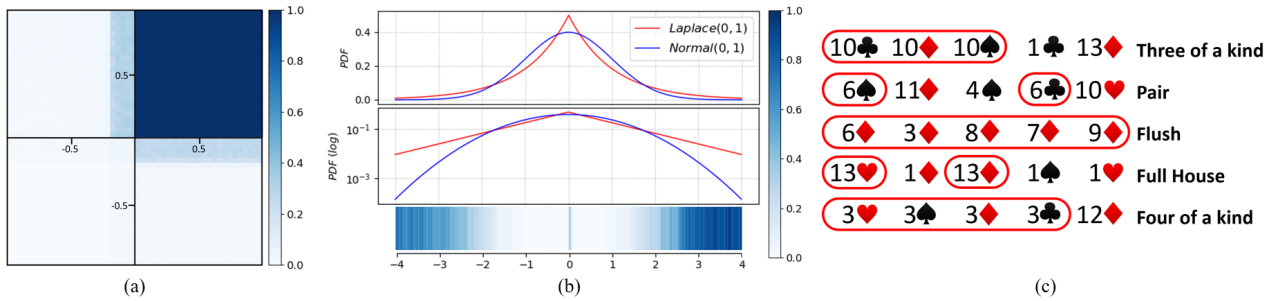
First quadrant (Sec. 5.1). We studied the 2D task of identifying the existence of a point in the 1^{st} quadrant. Figure 3(a) shows the average importance scores on all points in the test set. For visualisation purposes the region $[-1, 1] \times [-1, 1]$ is quantized into 50×50 bins. As expected, points in the first quadrant were the most important and points on the boarderline of this region had moderate importance.

Two-sample problem (Sec. 5.5). This task was to identify whether two subsets were sampled from the same distribution or not. We used *Laplace(0, 1)* or *Normal(0, 1)* as the possible source distributions. For visualization, the $[-4, 4]$ interval was broken to 800 bins. The mean importance score of the points in range are presented in Figure 3(b). The visualization shows that the model is focused on the tails of the distributions and on their mod (around 0). The model emphasizes those regimes since the differences between the two PDFs are more evident in those areas.

Binary poker hands classification (Sec. 5.7). In this task, the model was trained to distinguish between empty and non-empty poker hands without additional knowledge of poker rules. As the set size was constant and relatively small, we aggregated the importance scores per items' tuple (and not per individual items). That is, instead of looking at the importance of each item, we looked for the most prominent attention-set. For example, for a hand that contains a 'pair' (two cards with the same rank and different suit) the highest ranked subset includes exactly the two cards (see Figure 3(c) for more examples). Moreover, when the hand was empty, in 87% of the cases the most important subset consisted of only one card. For non-empty hands, the most important subset consisted of two or more cards 77% of the time.

Drug prescription errors (Sec. 5.2). In this task we trained GBeST to distinguish between compatible and incompatible sets of drugs prescriptions. We measured how often an incompatible drug appeared in the top k ranked items for $k = 1, 5$. We considered a 'hit' if an incompatible item appeared among the k most important items. The *mean hit rate (HR@k)* is reported as the average hit rate among all the incompatible test records. For the *Multiple Item Replacement* setup, *HR@1/5*$= 73.2\%/93.3\%$, and for the

[1] If there is a tie such that all the items from position r_1 to r_2 have the same count then each of them is ranked $^{(r_1+r_2)}/_2$.



(a) (b) (c)

Figure 3. Visualizations of item-level importance. (a) First quadrant, mean importance for points in $[-1, 1]^2$. (b) Two-sample problem, mean importance for points in $[-4, 4]$. (c) Binary poker hands classification, the most important subsets are highlighted.

Single Item Replacement setup: *HR@1/5*= $48.1\%/77.2\%$,

The newly developed item-level explainability method is useful for quantifying the importance of each item in the set. In combination with existing tools for explaining tree-based models, it provides high transparency to the algorithmic decision-making process.

7. Conclusions

In this study, we introduced Set-Tree, to allow tree-based models to operate on sets by using an attention mechanism. We presented a theoretical analysis of the model and provided a comprehensive empirical evaluation. We demonstrated that learning using Set-Trees consistently outperformed other methods based on trees, and frequently outperformed deep learning methods. We also introduced a new item-level explainability method for Set-Trees and demonstrated how it makes the predictions of Set-Trees intelligible.

Since problems in which records containing sets are common, as demonstrated in this work, we believe that Set-Trees may have large influence on machine-learning practitioners. Moreover, the approaches presented here can be extended to other types of data structures such as graphs or "mixed" records with both set and non-set features.

Acknowledgements

We wish to thank Prof. Abner Soffer and Dr. Emilie Bertholet (Tel Aviv University, Department of Particle Physics) for their inspiring ideas and helpful discussions. The research reported in this work was supported by a grant from the Tel Aviv University Center for AI and Data Science (TAD).

References

Almeida, L. G., Backović, M., Cliche, M., Lee, S. J., and Perelstein, M. Playing tag with ann: boosted top identification with pattern recognition. *Journal of High Energy Physics*, 2015(7):1–21, 2015.

Bevan, A., Golob, B., Mannel, T., Prell, S., Yabsley, B., Abe, K., Aihara, H., Anulli, F., Arnaud, N., Aushev, T., et al. *The Physics of the B Factories*, volume 74, chapter 9, pp. 3026. Springer Science and Business Media LLC, 2014.

Breiman, L. Random forests. *Machine learning*, 45(1): 5–32, 2001.

Breiman, L., Friedman, J., Stone, C. J., and Olshen, R. A. *Classification and regression trees*. CRC press, 1984.

Cattral, R., Oppacher, F., and Deugo, D. Evolutionary data mining with automatic rule generalization. *Recent Advances in Computers, Computing and Communications*, 1(1):296–300, 2002.

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

Cogan, J., Kagan, M., Strauss, E., and Schwartzman, A. Jet-images: computer vision inspired techniques for jet tagging. *Journal of High Energy Physics*, 2015(2), Feb 2015.

Connolly, A. J., Csabai, I., Szalay, A. S., Koo, D. C., Kron, R. G., and Munn, J. A. Slicing through multicolor space: Galaxy redshifts from broadband photometry. *The Astronomical Journal*, 110:2655, Dec 1995.

Desai, R. J., Dejene, S., Jin, Y., Liu, J., and Kim, S. C. Comparative risk of diabetes mellitus in patients with rheumatoid arthritis treated with biologic or targeted synthetic disease-modifying drugs: A cohort study. *ACR Open Rheumatology*, 2(4):222–231, 2020a.

Desai, R. J., Wang, S. V., Vaduganathan, M., Evers, T., and Schneeweiss, S. Comparison of machine learning methods with traditional models for use of administrative claims with electronic medical records to predict heart failure outcomes. *JAMA Network Open*, 3(1):e1918962–e1918962, 2020b.

Dietterich, T. G., Lathrop, R. H., and Lozano-Pérez, T. Solving the multiple instance problem with axis-parallel rectangles. *Artificial intelligence*, 89(1-2):31–71, 1997.

Douzal-Chouakria, A. and Amblard, C. Classification trees for time series. *Pattern Recognition*, 45(3):1076–1091, 2012.

Friedman, J. H. Greedy function approximation: a gradient boosting machine. *Annals of statistics*, pp. 1189–1232, 2001.

Friedman, J. H. Stochastic gradient boosting. *Computational statistics & data analysis*, 38(4):367–378, 2002.

Friedman, J. H. and Meulman, J. J. Multiple additive regression trees with application in epidemiology. *Statistics in medicine*, 22(9):1365–1381, 2003.

González, C. J. A. and Diez, J. J. R. Time series classification by boosting interval based literals. *Inteligencia Artificial. Revista Iberoamericana de Inteligencia Artificial*, 4(11):2–11, 2000.

Guest, D., Collado, J., Baldi, P., Hsu, S.-C., Urban, G., and Whiteson, D. Jet flavor classification in high-energy physics with deep neural networks. *Physical Review D*, 94(11), Dec 2016.

Guillame-Bert, M. and Dubrawski, A. Classification of time sequences using graphs of temporal constraints. *The Journal of Machine Learning Research*, 18(1):4370–4403, 2017.

Guillame-Bert, M., Bruch, S., Mitrichev, P., Mikheev, P., and Pfeifer, J. Modeling text with decision forests using categorical-set splits. *arXiv preprint arXiv:2009.09991*, 2020.

Han, T., Jiang, D., Zhao, Q., Wang, L., and Yin, K. Comparison of random forest, artificial neural networks and support vector machine for intelligent diagnosis of rotating machinery. *Transactions of the Institute of Measurement and Control*, 40(8):2681–2693, 2018.

Hastie, T., Tibshirani, R., and Friedman, J. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer series in statistics. Springer, 2009.

Ho, T. K. The random subspace method for constructing decision forests. *IEEE transactions on pattern analysis and machine intelligence*, 20(8):832–844, 1998.

Hochreiter, S. and Schmidhuber, J. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

Huang, Z., Xu, W., and Yu, K. Bidirectional lstm-crf models for sequence tagging. *arXiv preprint arXiv:1508.01991*, 2015.

Johnson, A. E., Pollard, T. J., Shen, L., Lehman, L.-w. H., Feng, M., Ghassemi, M., Moody, B., Szolovits, P., Celi, L. A., and Mark, R. G. Mimic-iii, a freely accessible critical care database. *Scientific data*, 3:160035, 2016.

Kaggle. Kaggle state of machine learning and data science 2020, 2020.

Kasieczka, G., Plehn, T., Thompson, J., and Russel, M. Top quark tagging reference dataset, 2019. URL https://zenodo.org/record/2603256#.X-xjV9gzabg.

Ke, G., Meng, Q., Finley, T., Wang, T., Chen, W., Ma, W., Ye, Q., and Liu, T.-Y. Lightgbm: A highly efficient gradient boosting decision tree. In *Advances in neural information processing systems*, pp. 3146–3154, 2017.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Kohn LT, Corrigan JM, D. M. *To Err is Human: Building a Safer Health System*. National Academies Press (US), 2000.

Komárek, T. and Somol, P. Multiple instance learning with bag-level randomized trees. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pp. 259–272. Springer, 2018.

Komiske, P. T., Metodiev, E. M., and Thaler, J. Energy flow networks: deep sets for particle jets. *Journal of High Energy Physics*, 2019(1), Jan 2019.

Krauss, C., Do, X. A., and Huck, N. Deep neural networks, gradient-boosted trees, random forests: Statistical arbitrage on the s&p 500. *European Journal of Operational Research*, 259(2):689–702, 2017.

Lee, J., Lee, Y., Kim, J., Kosiorek, A., Choi, S., and Teh, Y. W. Set transformer: A framework for attention-based permutation-invariant neural networks. In *International Conference on Machine Learning*, pp. 3744–3753. PMLR, 2019.

Leistner, C., Saffari, A., and Bischof, H. Miforests: Multiple-instance learning with randomized trees. In *European Conference on Computer Vision*, pp. 29–42. Springer, 2010.

Li, Y., Bu, R., Sun, M., Wu, W., Di, X., and Chen, B. Pointcnn: Convolution on x-transformed points. In *Advances in neural information processing systems*, pp. 820–830, 2018.

Liu, X., Guo, Z., You, J., and Kumar, B. Attention control with metric learning alignment for image set-based recognition. *arXiv preprint arXiv:1908.01872*, 2019.

Louppe, G., Cho, K., Becot, C., and Cranmer, K. Qcd-aware recursive neural networks for jet physics. *Journal of High Energy Physics*, 2019(1), Jan 2019.

Lucena, B. Exploiting categorical structure using tree-based methods. In *International Conference on Artificial Intelligence and Statistics*, pp. 2949–2958. PMLR, 2020.

Maron, H., Litany, O., Chechik, G., and Fetaya, E. On learning sets of symmetric elements. In *International Conference on Machine Learning*, pp. 6734–6744. PMLR, 2020.

Meek, C., Chickering, D. M., and Heckerman, D. Autoregressive tree models for time-series analysis. In *Proceedings of the 2002 SIAM International Conference on Data Mining*, pp. 229–244. SIAM, 2002.

Melki, G., Cano, A., and Ventura, S. Mirsvm: multi-instance support vector machine with bag representatives. *Pattern Recognition*, 79:228–241, 2018.

Munkhdalai, L., Munkhdalai, T., Namsrai, O.-E., Lee, J. Y., and Ryu, K. H. An empirical comparison of machine-learning methods on bank client credit assessments. *Sustainability*, 11(3):699, 2019.

Oliva, J., Póczos, B., and Schneider, J. Distribution to distribution regression. In *International Conference on Machine Learning*, pp. 1049–1057, 2013.

Prokhorenkova, L., Gusev, G., Vorobev, A., Dorogush, A. V., and Gulin, A. Catboost: unbiased boosting with categorical features. In *Advances in neural information processing systems*, pp. 6638–6648, 2018.

Qi, C. R., Su, H., Mo, K., and Guibas, L. J. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 652–660, 2017a.

Qi, C. R., Yi, L., Su, H., and Guibas, L. J. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. *Advances in neural information processing systems*, 30:5099–5108, 2017b.

Qu, H. and Gouskos, L. Jet tagging via particle clouds. *Physical Review D*, 101(5), Mar 2020.

Quinlan, J. R. Induction of decision trees. *Machine learning*, 1(1):81–106, 1986.

Ramoser, H., Muller-Gerking, J., and Pfurtscheller, G. Optimal spatial filtering of single trial eeg during imagined hand movement. *IEEE transactions on rehabilitation engineering*, 8(4):441–446, 2000.

Rodriguez-Galiano, V., Sanchez-Castillo, M., Chica-Olmo, M., and Chica-Rivas, M. Machine learning predictive models for mineral prospectivity: An evaluation of neural networks, random forest, regression trees and support vector machines. *Ore Geology Reviews*, 71:804–818, 2015.

Rozenblum, R., Rodriguez-Monguio, R., Volk, L. A., Forsythe, K. J., Myers, S., McGurrin, M., Williams, D. H., Bates, D. W., Schiff, G., and Seoane-Vazquez, E. Using a machine learning system to identify and prevent medication prescribing errors: A clinical and cost analysis evaluation. *The Joint Commission Journal on Quality and Patient Safety*, 46(1):3 – 10, 2020.

Rykoff, E. R. and S., E. redmapper ii: X-ray and sz performance benchmarks for the sdss catalog. *The Astrophysical Journal*, 783(2):80, Feb 2014.

Rykoff, E. S., Rozo, E., Busha, M. T., Cunha, C. E., Finoguenov, A., Evrard, A., Hao, J., Koester, B. P., Leauthaud, A., Nord, B., and et al. redmapper. i. algorithm and sdss dr8 catalog. *The Astrophysical Journal*, 785(2), Apr 2014.

Sandulescu, V. and Chiru, M. Predicting the future relevance of research institutions-the winning solution of the kdd cup 2016. *arXiv preprint arXiv:1609.02728*, 2016.

Segal, G., Segev, A., Brom, A., Lifshitz, Y., Wasserstrum, Y., and Zimlichman, E. Reducing drug prescription errors and adverse drug events by application of a probabilistic, machine-learning based clinical decision support system in an inpatient setting. *Journal of the American Medical Informatics Association*, 26(12):1560–1565, 2019.

Segol, N. and Lipman, Y. On universal equivariant set networks. *arXiv preprint arXiv:1910.02421*, 2019.

Sutherland, D. J., Xiong, L., Póczos, B., and Schneider, J. Kernels on sample sets via nonparametric divergence estimates. *arXiv preprint arXiv:1202.0302*, 2012.

Vinayak, R. K. and Gilad-Bachrach, R. Dart: Dropouts meet multiple additive regression trees. In *Artificial Intelligence and Statistics*, pp. 489–497. PMLR, 2015.

Wang, Y., Sun, Y., Liu, Z., Sarma, S. E., Bronstein, M. M., and Solomon, J. M. Dynamic graph cnn for learning on point clouds. *Acm Transactions On Graphics (tog)*, 38 (5):1–12, 2019.

Wu, Z., Song, S., Khosla, A., Yu, F., Zhang, L., Tang, X., and Xiao, J. 3d shapenets: A deep representation for volumetric shapes. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1912–1920, 2015.

Xu, Y., Fan, T., Xu, M., Zeng, L., and Qiao, Y. Spidercnn: Deep learning on point sets with parameterized convolutional filters. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 87–102, 2018.

Yang, Y., Morillo, I. G., and Hospedales, T. M. Deep neural decision trees. *arXiv preprint arXiv:1806.06988*, 2018.

Zaheer, M., Kottur, S., Ravanbakhsh, S., Poczos, B., Salakhutdinov, R. R., and Smola, A. J. Deep sets. In *Advances in neural information processing systems*, pp. 3391–3401, 2017.

Zhang, C., Liu, C., Zhang, X., and Almpanidis, G. An up-to-date comparison of state-of-the-art classification algorithms. *Expert Systems with Applications*, 82:128–150, 2017.

Zucker, J.-D. and Chevaleyre, Y. *Solving multiple-instance and multiple-part learning problems with decision trees and decision rules. Application to the mutagenesis problem*. PhD thesis, LIP6, 2000.