# Random Forest Induced Diverse Subgroup Discovery

Djamey Hermelijn (s3548880)

Supervisors:
Matthijs van Leeuwen & Lincen Yang

Bachelor Thesis– Data Science and Artificial Intelligence

Leiden Institute of Advanced Computer Science (LIACS)
liacs.leidenuniv.nl

August 23, 2026

**Abstract**

Subgroup discovery is a data mining technique used to find subsets of data where the target class distribution deviates from the norm. This offers experts an interpretable way to explore large datasets that would otherwise be difficult to analyze. A common flaw in traditional subgroup discovery is that the final results are often redundant, with similar subgroups being described repeatedly. Existing methods combat this by diversifying candidate generation and by using diversity-aware selection algorithms. This paper builds on the diverse subgroup set discovery (DSSD) algorithm, which uses a diversity-aware beam search to generate candidates, and diversity-aware selection to construct a high-quality, non-redundant result set. While DSSD is a powerful SD tool, several aspects of its design leave room for improvement. Namely, DSSD requires a considerable amount of time to run, even though it uses a heuristic approach. Furthermore, its candidate generation is quality-driven at its core, meaning classes with lower-quality subgroups on average will not be well represented in the final result. While this is a reasonable design choice if the goal is to find globally high-quality and diverse subgroups, it may be limiting if a general collection of subgroups for every class is desired. This paper proposes RF-DSSD, a diverse subgroup discovery algorithm designed with DSSD's limitations in mind. It uses a random forest as a diverse candidate generation mechanism, paired with three selection algorithms, top-k, greedy Jaccard, and coverage-based. The selection algorithms are applied per class using a proportional budget that reflects each class's distribution in the dataset. As a result, RF-DSSD achieved full class coverage, compared to DSSD's 72.6% average class coverage. To evaluate RF-DSSD against DSSD, multiple experiments assessed the final quality and diversity of the collected subgroups. However, DSSD's average quality remained substantially better, though its maximum quality was more comparable. This suggests that certain classes pull RF-DSSD's global average down, a pattern later confirmed by restricting the comparison to classes covered by both methods. While RF-DSSD does not match DSSD's raw quality, it demonstrates a viable, fast approach to diverse subgroup discovery that achieves full class coverage, at a fraction of DSSD's runtime, with clear directions identified for narrowing the remaining quality gap.

# Contents

# 1 Introduction

As datasets grow in size and complexity, interpretability can be lost due to the increased difficulty of finding meaningful patterns by hand. To tackle this issue, pattern mining algorithms are used to find such patterns using a systematic approach, providing useful insights to domain experts where it would otherwise be unfeasible. A prime example of such an algorithm is subgroup discovery (SD) [4], a method that attempts to find subsets of data whose class distribution deviates from that of the global dataset. This deviation can be seen as an interestingness metric, as it measures how unusual, or surprising, a certain subgroup may be. Early approaches implemented a naive top-k selection method [6, 14], meaning that after a collection of subgroups had been found, the final output would simply consist of the top-k subgroups based on their interestingness. While this provides a collection of high-quality subgroups, a common issue with this approach is that the collection eventually contains multiple redundant subgroups. This redundancy means that the top ranking subgroups describe very similar parts of the data, leading to fewer genuinely unique and novel subgroups than desired. Such redundancy motivated a new form of SD to combat this redundancy, aptly named diverse subgroup discovery.

A relatively recent example of a diverse subgroup discovery algorithm is that of diverse subgroup set discovery (DSSD) [12], which creates candidates while applying various processing techniques meant to motivate diversity in the final result. While DSSD outputs a collection of high-quality, diverse subgroups, a few characteristics stand out: DSSD takes a considerable amount of time to finish. Furthermore, its framework doesn't guarantee full class coverage, meaning that it is not certain that every class in the dataset will have a corresponding subgroup in DSSD's final output. This could be resolved by performing a separate run per class, though at a substantial cost to runtime. RF-DSSD, the algorithm this paper introduces, was designed to combat these issues. As a diverse subgroup discovery algorithm, it aims to reduce runtimes and make full class coverage more reasonable to achieve, while also attempting to uphold DSSD's level of quality and diversity in the final output. Random forests [2] are used as a candidate generation technique, chosen primarily for being a fast, heuristic approach.

To understand why random forests are a suitable candidate generation mechanism, it is useful to first outline their basic structure. A random forest consists of a collection of unique decision trees, also referred to as estimators, and a decision tree consists of a number of nodes. The nodes in a decision tree contain a number of instances in the data, which each belong to their respective class. How evenly these classes are mixed within a node is referred to as its impurity, where a node containing mostly one class is considered pure. This property is used as a filtering criterion when deciding which nodes are eligible for inclusion in the candidate pool, the collection of subgroups from which RF-DSSD's final output is selected. Both of these properties, namely the number of estimators and the impurity threshold, are modulated throughout the experiments in this thesis. Random forests are by nature a global candidate generation technique, meaning they produce subgroups found in all different classes in a dataset, rather than requiring a new candidate generation step per class. The variability amongst different trees also assists in a more diverse result. Considering these factors, and the fact that random forests are known to be relatively computationally cheap, they make for an interesting potential candidate generator.

Further sections outline the exact goals that were set to help with the creation of RF-DSSD, as

well as the main methods of measuring its success. Various experiments were performed in order to find a reasonable setup for RF-DSSD, allowing it to be compared to DSSD. These experiments led to several notable patterns, showing that impurity and imbalance both affect WRAcc, the quality measure used throughout this thesis, via their effect on subgroup size. Despite the efforts made, RF-DSSD's average WRAcc does not match that of DSSD, motivating multiple promising directions for improvement, discussed later in this thesis.

### 1.1 Research Questions

This section introduces the research questions behind this paper.

RQ1: How do the quality and diversity of subgroups selected from a candidate pool generated from only leaf nodes compare to one generated from all nodes when using RF-DSSD?

RQ2: How does modulating the estimator count, impurity threshold, and selection-method-specific parameters affect the quality and diversity of RF-DSSD's output?

RQ3: How does RF-DSSD compare to DSSD in terms of quality, diversity, runtime, and class coverage?

RQ4: What amount of the quality disparity between RF-DSSD and DSSD stems from DSSD's incomplete class coverage, and does restricting the comparison to DSSD's covered classes, with and without attempting to emulate its output's class ratio, close the gap?

## 2 Background

This chapter introduces several key concepts used in this thesis, providing a necessary background for the later chapters. It starts by explaining subgroup discovery (SD) itself [4, 6, 14], covering its main use cases, discussing its relevance, and various shortcomings that motivate this thesis's research. This is followed by the quality metric used throughout this paper, WRAcc [7], as well as diverse subgroup discovery, the specific archetype of SD this thesis builds upon, alongside the diversity metrics associated with it. Since RF-DSSD uses random forests (RFs) as a candidate generation technique, this chapter also introduces the fundamentals of RFs [2]. Finally, DSSD [12] is discussed as both the primary point of comparison and a partial inspiration for RF-DSSD, given its diversity-aware candidate generation and selection algorithms.

### 2.1 Subgroup Discovery

Subgroup discovery (SD) is a descriptive data mining method used to make large collections of data more interpretable by finding interesting subsets of the data. An early implementation of SD is EXPLORA [6], a system developed by Klösgen to find subgroups based on their interestingness. More specifically, it evaluates how much a subgroup's target distribution deviates from that of the global population. Other early related approaches include Webb's OPUS [13] and Wrobel's MIDOS [14]. The following formalizes the notation used throughout this thesis, building up to a precise definition of a subgroup and its quality measure, before grounding these concepts in a concrete example.

The data used for SD are represented by a dataset $D$ consisting of instances described by a set of $m$ descriptive attributes

$$X = \{X_1, \ldots, X_m\},$$

and a target attribute $T$. For each descriptive attribute $X_i \in X$ let $Dom(X_i)$ denote the set of possible values of $X_i$. The target attribute can take values from a set of $n$ possible classes

$$C = \{c_1, \ldots, c_n\}.$$

For a particular subgroup, its target attribute takes the value $T = c$ for some $c \in C$. A subgroup description consists of a conjunction of conditions, where individual conditions are denoted by $q_i$. The number of conditions $k$ in a subgroup's description is referred to as its description length. These conditions are based on the various attributes found in $D$. Each condition compares an attribute $X_i$ with a value $v \in Dom(X_i)$ using a comparative operator. In this thesis, conditions are restricted to comparisons using $\{\leq, >\}$ (further explained in Section 3). A resulting subgroup description including its target class would be in the form:

$$s : q_1 \wedge q_2 \wedge \cdots \wedge q_k \implies c$$

Every subgroup $s$ has a cover $G \subseteq D$, the set of instances satisfying all conditions in its description:

$$G = \{d \in D : q_1(d) \wedge q_2(d) \wedge \cdots \wedge q_k(d)\}$$

Lastly, there exists a function called a quality measure $\varphi : S \to R$ that assigns a numeric value to a subgroup $s$, where $S$ denotes the set of all possible subgroups in $D$. This value indicates how interesting or unusual a subgroup is, typically by measuring how much its target distribution deviates from that of the global mean.

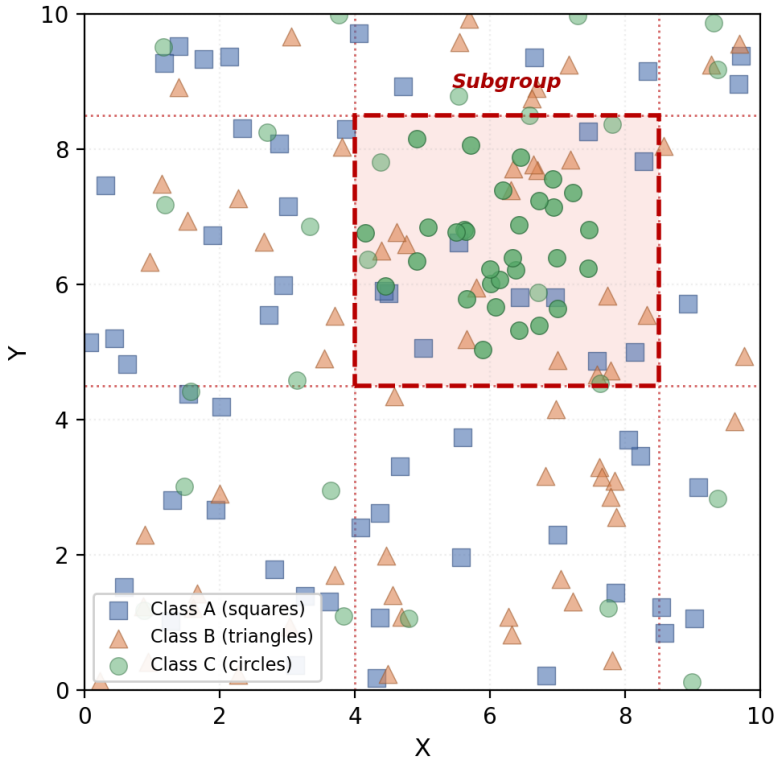



Figure 1: A figure showing an example of a possible subgroup, where X and Y denote the coordinates in the graph. The graph contains three classes: squares, triangles, and circles. Circles are the target class of the subgroup

As shown in Figure 1, the graph contains a collection of data points, where the shape determines the class of the individual (circle, triangle, square). The circle class is distributed quite sparsely throughout most of the graph, whereas the other classes have a more even distribution. Overall, the class distribution is fairly uniform, meaning every class makes up $\sim 33\%$ of the data. The graph shows a subgroup which could be described by the following rule: $s : X > 4 \wedge X \leq 8.5 \wedge Y > 4.5 \wedge Y \leq 8.5 \rightarrow$ Class C (circles). Its cover $G$ is every data point within the highlighted area, including instances whose class is not that of the target class, giving it a description length of four.

Now that subgroups have been sufficiently described, the next step is to specify how said subgroups can be found in data. In principle, this can be done exhaustively by evaluating every possible subgroup description. However, the search space grows exponentially with the number of features and maximum description length, as each additional condition and feature multiplies the number of possible descriptions. When exhaustive search is not feasible, subgroup discovery methods often rely on heuristics such as beam search [4], which iteratively refines a set of promising candidates rather than expanding all candidates. These methods result in a candidate pool from which the final result will be selected. Traditional methods utilize a top-k approach, meaning that they greedily select the k best subgroups based on the quality metric that they use. While this can lead to high-quality subgroups, subgroups with different descriptions may have similar covers, meaning the final subgroup set ends up obtaining some level of redundancy.

### 2.1.1 Weighted Relative Accuracy

Many variations of subgroup quality metrics exist. Early approaches typically combined accuracy and size, computing the ratio of data points in the cover belonging to the predicted class weighted by the size of the cover [14, 6]. Though alternative metrics were quickly developed, Lavrač et al. (2004) introduced Weighted Relative Accuracy (WRAcc) [7] as a refinement of these accuracy and size based measures, which has since become a standard in subgroup discovery and is the primary quality metric used in this paper. The formula is:

$$\text{WRAcc}(G) = \frac{|G|}{|D|}\left(\frac{|G \cap T|}{|G|} - \frac{|T|}{|D|}\right), \tag{1}$$

where $G$ is the subgroup cover, $D$ is the full dataset, $T$ is the set of instances belonging to the target class, $\frac{|G|}{|D|}$ is the relative size of the subgroup, $\frac{|G \cap T|}{|G|}$ is the precision of the subgroup (the fraction of covered instances belonging to the target class), and $\frac{|T|}{|D|}$ is the prior probability of the target class in the dataset. By subtracting the prior probability of the target class, WRAcc captures how much more prevalent the target class is within the subgroup compared to the general population, while also considering the subgroup's cover size relative to the size of the full dataset.

## 2.2 Diverse Subgroup Discovery

While top-k subgroup discovery is good for selecting high quality subgroup sets, as previously mentioned the final result suffers from the fact that the resulting subgroups have very similar conditions and cover largely the same data. This issue was addressed by the creation of diverse subgroup set discovery (DSSD) [12], which was created to assure a more diverse and non-redundant final result.

DSSD was introduced by van Leeuwen and Knobbe (2012) [12]. It utilizes a beam search for its candidate generation mechanism, while also using a diversity-aware beam selection to prevent redundancy during search. Beam search initializes an empty rule and iteratively refines it by adding one condition at a time until a stopping condition is met. The two main parameters that beam search has to control this process are the beam width $w$, which controls how many candidates are kept at each level, and maximum depth which controls the maximum number of conditions a subgroup description can contain. More specifically, the beam width only expands and refines the top $w$ candidates at each depth.

After this generation process different post-processing methods are applied to enable more diversity-aware selection. DSSD also applies post-processing steps such as dominance pruning, which reduces overly specific subgroup descriptions, resulting in a lower average description length while retaining most of the subgroup quality [12].

For its final step DSSD uses a diversity-aware selection procedure, and while it has numerous selection algorithms, the algorithm that was used in this paper's baseline comparisons is the coverage-based method. This method works by iteratively selecting the candidate that maximizes a coverage-weighted quality score, as candidates that overlap heavily with already selected subgroups get a penalty to their scores. Coverage-based selection is further explained in Section 3.2.3, including how candidates are scored during selection.

Related work has built on DSSD to address some of its limitations, specifically with regards to its runtime and memory efficiency [1]. This work was not, however, used as a central point of comparison in this paper, but illustrates that clear improvements can be made to DSSD, and that it remains an active reference point for further research.

### 2.2.1 Mean Jaccard Overlap

Jaccard overlap is a classical method used in set theory to describe the similarity between two sets [5]. Within subgroup discovery it can be used on separate subgroups for a more local comparison, but it can also be used to calculate the mean pairwise overlap of the final result [12], where a high overlap would indicate low diversity and vice versa. The exact formula used on a set A and B goes as follows.

$$J(A, B) = \frac{|A \cap B|}{|A \cup B|} \tag{2}$$

where $A$ and $B$ are two subgroup covers, $|A \cap B|$ is the number of instances covered by both subgroups, and $|A \cup B|$ is the number of instances covered by either subgroup. When there is no overlap between $A$ and $B$, $J(A, B) = 0$, and when their covers are identical, $J(A, B) = 1$.

### 2.2.2 Cover Redundancy

Cover redundancy is a diversity metric that measures how evenly a collection of subgroups' coverage is distributed across a dataset. This is computed by determining how many times each data point is covered across all subgroups, then measuring how far each coverage count deviates from the average, and finally averaging this across the whole dataset. A high cover redundancy score indicates a low level of diversity, as it reflects a more uneven distribution of coverage across the dataset. Uncovered instances are also considered in the calculation, meaning a subgroup set which covers more instances will tend to have a better score.

$$CR(S) = \frac{1}{|D|} \sum_{t \in D} \frac{|c(t, S) - \hat{c}|}{\hat{c}} \tag{3}$$

where $S$ is the selected subgroup set, $D$ is the full dataset, $c(t, S)$ is the number of subgroups in $S$ whose cover contains instance $t$, and $\hat{c}$ is the average value of $c(t, S)$ over all instances in $D$.

### 2.2.3 Joint Entropy

Lastly, joint entropy is a diversity metric used in diverse subgroup discovery originating from information theory [3]. For subgroup sets, it measures diversity by calculating the entropy over the joint distribution of subgroup covers [12]. A higher joint entropy indicates that the subgroup covers are more diverse, since JE measures the unpredictability of an instance's coverage pattern. The following is the exact formula for joint entropy:

$$H(S) = -\sum_{v \in \{0,1\}^{|S|}} p(v) \log_2 p(v) \tag{4}$$

where $v$ is a binary vector of length $|S|$ indicating which subgroups cover a given instance, and $p(v)$ is the proportion of instances in $D$ with the coverage pattern $v$.

To summarize the difference between the aforementioned diversity metrics, Jaccard overlap and cover redundancy both function as redundancy metrics by penalizing similarity or repeated covering. Joint entropy, on the other hand, more directly measures the variety of covers across subgroups. In other words, the first two metrics measure the absence of redundancy, while JE measures the presence of diversity.

### 2.3 Random Forests

Decision trees and random forests are traditionally used as supervised predictive models, trained to classify instances based on their descriptive attributes. Despite this predictive origin, this paper repurposes random forests as a candidate generation mechanism [2], with the aim of providing a diverse candidate pool while hopefully maintaining a high average WRAcc.

A decision tree consists of nodes, where the initial node represents the entire dataset. From this node onward, a split occurs, meaning a certain condition divides the node into two nodes with distinct individuals. This split aims to maximize the predictive power of the tree by minimizing the impurity at each split, which is done by choosing the split condition that results in the most uniform class distribution.

However, since decision trees are deterministic and prone to overfitting on their training data, random forests introduce two new techniques in order to create a more robust method. The first mechanism, called feature sub-sampling, makes it so that only a random subset of features is considered per split. The second technique is called bootstrapping, and ensures that every decision tree is trained on a different subset of the data. Bootstrapping randomly samples with replacement from the training set for every decision tree, resulting in each tree's training subset including multiple copies of certain instances, while omitting others.

By using bootstrapping and feature sub-sampling, a random forest offers greater predictive power than a decision tree. More importantly, it is able to capture the true signal more accurately, allowing it to generalize to some extent to unseen data. These techniques introduce randomness into the data and features each tree is trained on, causing different splits to be chosen across the forest, resulting in trees that differ structurally from one another. Since each tree describes various regions of the instance space, the random forest in its entirety describes a larger and more varied collection of data than a single decision tree otherwise would.

## 3 Methods

This chapter describes RF-DSSD's methodology, including the motivation behind its key design decisions. The algorithm uses random forests as a candidate generation technique and is thus named RF-DSSD, as it is inspired by DSSD [12].

The first step of RF-DSSD is training a random forest, after which rules are extracted from every decision tree in the forest by utilizing a depth-first search traversal approach. Only the rules that meet certain node type and impurity criteria are stored for later use, in order to reduce the size of the candidate pool and thus the runtime. A rule corresponding to a certain node is constructed from the split conditions found along the path from the root of the tree to that node. DFS is well

suited for this, as it naturally keeps track of the conditions leading up to any given node during traversal, making it a simple but effective extraction method.

Following this step, the extracted rules are added to a candidate pool, after which one of three different selection algorithms is used to perform the final step of subgroup discovery. The three methods used are top-k, greedy Jaccard selection, and coverage-based selection, found in Section 3.2.1, Section 3.2.2, and Section 3.2.3 respectively. The RF-DSSD pipeline is shown in Figure 2.

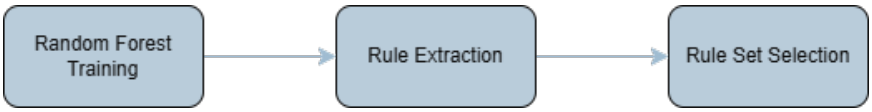



Figure 2: Overview of the RF-DSSD pipeline.

## 3.1 Candidate Generation

### 3.1.1 Random Forest Training

Traditionally, random forests are used as a predictive method. For this reason the data they are being trained on is often split into a train and test set to check whether the model generalizes to unseen data. However, in the case of subgroup discovery, which is a descriptive method, there is no need to generalize to unseen data, as the aim is instead to capture the structure present in the dataset itself. This is consistent with the broader convention in subgroup discovery, where quality measures such as WRAcc are already relied upon to discourage overly specific results [7]. This thesis therefore trains the random forest on the full dataset.

### 3.1.2 Rule Extraction

After the random forest has been trained, the next step is to extract and convert its rules into a candidate pool of subgroups. Within a decision tree, any node can be viewed as a subgroup in the same sense described in Section 2 (illustrated for a specific example in Figure 3). Whether or not this is a good subgroup is determined by the quality measure, which the selection algorithm then uses to make its choices.

To extract rules from a decision tree, a depth-first-search approach is used, where during the traversal the conditions leading to the current node are stored in a path list. By using backtracking, the list can be updated to accurately reflect the path of conditions leading to certain nodes. However, for a path to be added to the candidate pool, certain criteria must be met, namely an impurity threshold $\tau$ and a minimum samples threshold. $\tau$ checks whether or not the current node being considered has an impurity index below a specified amount, usually with a value in the range of $\tau \in [0, 1]$. Pure nodes have a class distribution concentrated around one or a few classes, whereas impure nodes tend to have a more even distribution across classes. Pure nodes would thus lead to subgroups with higher accuracy, which would directly increase the WRAcc, as a result of the accuracy component found in Formula 1. While this is a reasonable approach, accuracy is but one component in WRAcc, and size plays another direct role in the calculation, which motivates experimenting with different values of $\tau$.

A minimum instance count of 5 was chosen as a requirement for a node to be added to the candidate pool. This threshold is intentionally kept low, as it is only meant to filter out candidates with an extremely low instance count, given that WRAcc benefits from larger subgroup sizes [7]. A higher threshold could exclude genuine, but naturally smaller subgroups, especially when an underrepresented class is present, going against RF-DSSD's diversity goal. By doing this for every decision tree in the model, the final candidate pool is generated.

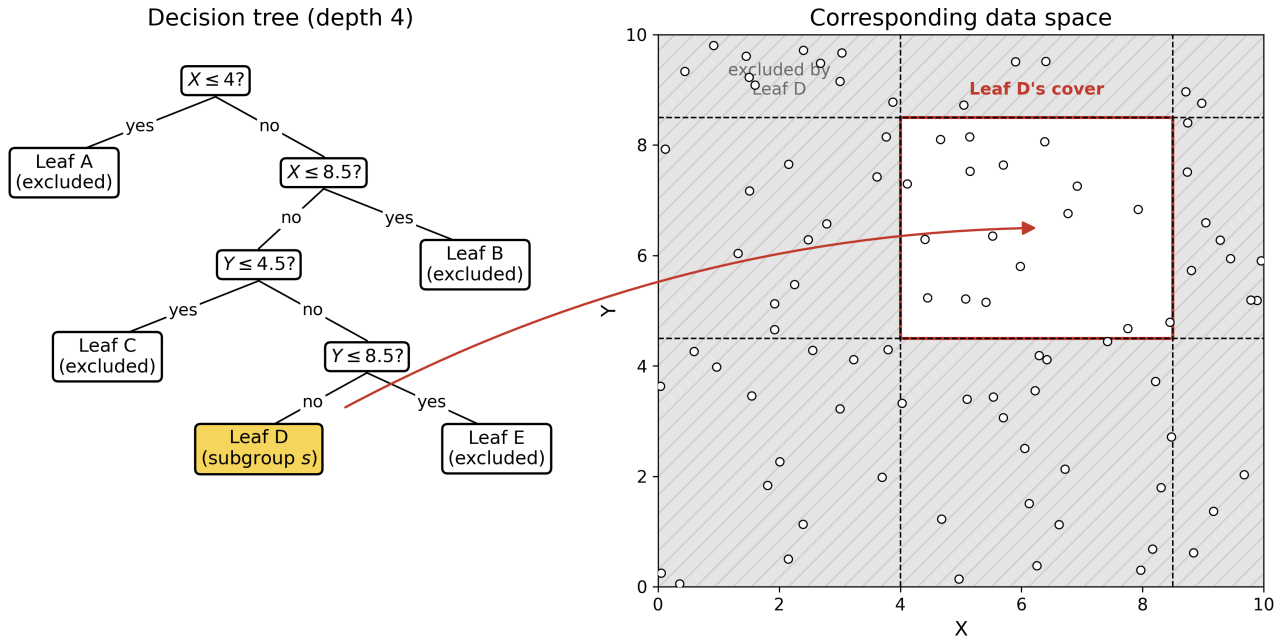



Figure 3: Figure showing an example of a decision tree, and its corresponding subgroup in the data space. The left panel shows a depth-4 decision tree, with each split narrowing the region until Leaf D corresponds to the bounded rectangle $s : X > 4 \wedge X \leq 8.5 \wedge Y > 4.5 \wedge Y \leq 8.5 \rightarrow$ Class C (circles). The right panel shows this same region within the data space; the hatched grey area indicates instances excluded from Leaf D's cover.

## 3.2 Candidate Selection

RF-DSSD performs a per-class selection, meaning that only one class is considered per selection run. Unlike DSSD, which requires a new candidate generation step for every class, RF-DSSD's random forest only has to be generated once. Following training, all that remains is a selection step for every class, after which the results are aggregated. This approach considers global diversity using metrics like mean overlap, cover redundancy, and joint entropy, and also leads to total class coverage in the final output, which is the primary motive for a separate selection per class. An alternative would be to select subgroups across all classes in a single run. This would require a quality measure comparable across classes, such as MWRAcc, which sums the absolute value of one-vs-rest WRAcc across all classes into a single combined score [12]. However, some classes are easier to describe than others, for example due to being larger or distinct from other classes, which allows subgroups targeting them to more easily achieve high accuracy and coverage, directly inflating their WRAcc. Given that these classes will contribute more to the combined score, subgroups whose target class

is smaller and inherently harder to describe are less likely to be considered during selection, even if they are a relatively good subgroup for their respective class. Per-class selection avoids this issue entirely. Performing a separate run per class creates a one-vs-rest scenario overall, making WRAcc a natural fit as its quality measure.

Although selection is performed separately per class, covers and WRAcc are still calculated relative to the full training set. This means that the coverage of a rule depends on the proportion of the complete dataset it describes, rather than only on instances of the target class. Calculating WRAcc separately using class-specific dataset sizes would place classes on different scales and could inflate scores for smaller classes, making cross-class comparisons less meaningful. This is especially important because the reported quality and diversity metrics are macro-averaged, meaning each metric is first calculated per class and then averaged across classes in order to keep the per-class scores comparable. Given that selection is performed per class, the diversity metrics are likewise computed separately for each class's selected subgroup set and then macro-averaged.

RF-DSSD has a desired results parameter similarly to DSSD which dictates the final size of the output. The standard value for both DSSD and RF-DSSD has been set to 100, providing enough subgroups to perform quantitative analysis on while also keeping runtime realistic. RF-DSSD attempts to generate subgroups relative to their size in the data, which is done by calculating the ratio of data points in a class and the amount of data points in the whole dataset. Given that a very small class might not be covered at all using this method, a lower bound of 1 subgroup per class was set, resulting in the following formula:

$$\frac{|C|}{|S|} \cdot n \tag{5}$$

where $|C|$ is the number of instances in class $C$, $|S|$ is the total number of instances in the dataset, and $n$ is the desired total number of results.

### 3.2.1 Top-k

Top-k selection is a quality-based selection method that selects the candidate with the highest quality score until the desired output length is reached. Similarly to the other selection methods, this gets done for every class in order to maintain an equal class distribution. As it is a quality-based selection method, it provides a reasonable upper bound on the quality achievable by RF-DSSD's subgroups.

### 3.2.2 Greedy Jaccard

This selection algorithm builds upon the top-k selection, as it still selects the candidate with the highest WRAcc that it can. Unlike top-k, however, it is a diversity-aware selection method based on the Jaccard overlap between candidates. The following shows the procedure behind this algorithm:

**Step 1:** From the candidate pool select the subgroup $G$ with the highest quality score $\varphi(G)$, remove it from the candidate pool and add it to the selection set Sel. Ties are resolved using the candidate ordering.

**Step 2:** Remove all candidates whose Jaccard overlap with $G$ exceeds the threshold parameter $\theta \in [0, 1]$.

**Step 3:** Repeat step 1 and step 2 using the remaining candidate pool.

**Step 4:** Terminate the selection when the desired number of candidates has been selected, or the candidate pool is empty.

This process is performed separately for every class using their respective candidate pool. Modulating $\theta$ allows experimentation into its effects on the final quality and diversity of the output (See section 5.4).

#### 3.2.3 Coverage-based

Coverage-based selection is the final selection algorithm used and follows essentially the same process that is described in the DSSD paper [12]. Taking into account that DSSD also uses a coverage-based selection algorithm allows for comparisons to be made between RF-DSSD and DSSD's results using the same selection process. Coverage-based selection considers how much of a new subgroup's cover has already been covered by previously selected subgroups. It does this using the function $\Omega$, which takes in a candidate subgroup $G$, and the set of already selected subgroups $Sel$, and applies the following formula:

$$\Omega(G, Sel) = \frac{1}{|G|} \sum_{t \in G} \alpha^{c(t,Sel)} \tag{6}$$

The symbols $t$ and $\alpha \in [0, 1]$ denote an isolated instance, and a weighting parameter, respectively. The formula is performed by first summing the discounts $\alpha^{c(t,Sel)}$ of all instances $t \in G$, where $c(t, Sel)$ denotes the number of times $t$ has already been covered in $Sel$, and finally dividing this by the subgroup's size. The score with which subgroups are then considered during a selection run is the average sum of discounts $\Omega(G, Sel) \cdot \varphi(G)$. To perform a selection run, the following procedure is followed:

**Step 1:** From the candidate pool select the subgroup $G$ with the highest $\Omega(G, Sel) \cdot \varphi(G)$.

**Step 2:** Remove the selected candidate from the candidate pool, and add it to $Sel$.

**Step 3:** Update the $\Omega(G, Sel)$ value for every subgroup.

**Step 4:** Repeat the previous steps until either the desired number of results is achieved, or no subgroups exist in the candidate pool.

This process is performed separately for every class using their respective candidate pool. Later experiments modulate $\alpha$ in order to observe its effects on both quality and diversity (See section 5.4).

# 4 Experimental Setup

## 4.1 Experimental setup

To create the random forest component in RF-DSSD, the scikit-Learn library was used, which is a Python library consisting of numerous machine learning algorithms [9]. Specifically, the random forest classifier package was used. This package allows you to train a random forest with any dataset

consisting of numeric tabular data, though categorical data can also be described using one-hot encoding. Table 1 shows all datasets used in the various experiments performed in this paper. It denotes the dataset name, how many instances, features and classes it has and also contains an imbalance ratio to show the degree of class imbalance in a particular dataset.

| Dataset | Instances | Features | Classes | Imbalance Ratio |
|---|---|---|---|---|
| Adult | 48,842 | 14 | 2 | 3.18 |
| Mushroom | 8,124 | 22 | 2 | 1.07 |
| Letters | 20,000 | 16 | 26 | 1.11 |
| Segment | 2,310 | 19 | 7 | 1.00 |
| Shuttle | 58,000 | 9 | 7 | 4558.60 |
| Covertype | 110,393 | 54 | 7 | 38.60 |
| Bank marketing | 45,211 | 16 | 2 | 7.55 |

Table 1: Dataset statistics.

For the initial experiments the Adult and Mushroom datasets were used. Adult and Mushroom are widely used datasets in subgroup discovery research, making them well suited for the earlier, non-exhaustive experiments in this thesis. Most of the remaining datasets are similarly typical for SD, with the exception of Segment and Letters, which are more commonly used for image recognition tasks. Segment and Letters are nonetheless included to provide a broader, more representative average across the full dataset evaluation.

RF-DSSD is evaluated across numerous parameter settings for every selection method, with the aim of finding a parameter setup for a fair comparison with DSSD. The same datasets are also used to establish a DSSD baseline, where the relevant metrics are calculated for comparison. These experiments examine how the diversity, quality, and runtime change when using different settings. All of the metrics shown in the Results (See section 5) and Appendix (See Section 9) sections have been calculated on a per-class basis and then macro averaged.

In order to have a comparable evaluation, DSSD's generated subgroups were grouped by their predicted class, after which the same per-class, macro-averaged metrics used for RF-DSSD were computed for DSSD's results as well. Since DSSD evaluates subgroups using MWRAcc, which measures value across classes rather than with respect to one fixed target class, a target class must be assigned afterward. The assigned class is the one that maximizes the WRAcc score. This is more consistent with the quality-based selection process used in DSSD than simply assigning the dominant class within a subgroup. The reason for not using a class dominance-based approach is possible imbalance in the dataset, meaning that an underrepresented class will end up not being used as a target class because a more common class is more likely to make up most of the subgroup. Additionally, the fact that a more common class has a higher prior is not penalized by the class dominance approach, while choosing the class with the highest WRAcc circumvents this issue.

This approach is further supported by an initial experiment that examined whether or not any other classes received a WRAcc within 10 percent of the highest WRAcc's value for a particular class, which would indicate a close tie. The results showed that in all datasets except Segment and Letters, there were no subgroups found with ambiguous target classes. Given that the method used

to calculate the target class remains unambiguous for the more typical subgroup discovery datasets (Segment and Letters are normally used for image recognition tasks), it was deemed an acceptable approach.

| Parameter | Estimator Sweep | Impurity Sweep | Parameter Sweep | Full Evaluation |
|---|---|---|---|---|
| Estimators | 50–500 | 300 | 300 | 300 |
| $\tau$ | 0.1 | 0.0–1.0 | 1.0 | 1.0 |
| $\theta$ (Jaccard) | 0.6 | 0.6 | 0.0–1.0 | 0.8 |
| $\alpha$ (Coverage) | 0.6 | 0.6 | 0.0–1.0 | 0.9 |

Table 2: Parameter configurations used across the different experiments in this thesis. The first column lists the parameter names; each remaining column corresponds to an experiment, in order: Estimator Sweep (Section 5.1), Impurity Sweep (Section 5.2), Parameter Sweep (Section 5.4), and Full Evaluation (Section 6.5). Each row shows the value used for a given parameter in that experiment, where a range indicates the start and stop points of that parameter's sweep.

The random forest model has numerous parameters dictating the specifics of the way training is to be done as can be seen in Table 2. The parameter controlling the maximum depth of the random forest maintains a value of 5 throughout all experiments performed in this paper.

## 4.2 Experimental Questions

This section covers more specific experiment-level questions, which are meant to answer the research questions stated in Section 1.1.

Q1: How does increasing the number of estimators in the random forest used as a candidate source affect the quality and diversity of the various RF-DSSD variants?

Q2: How does modulating $\tau$ used in the candidate extraction process affect the quality and diversity of the various RF-DSSD variants?

Q3: How does modulating $\tau$ compare to increasing the number of estimators in terms of its effect on the quality and diversity of various RF-DSSD variants?

Q4: Which selection-method-specific parameters provide the best balance between quality and diversity for RF-DSSD?

Q5: How does a reasonable parameter setup for RF-DSSD fare against DSSD in terms of quality, diversity, and runtime?

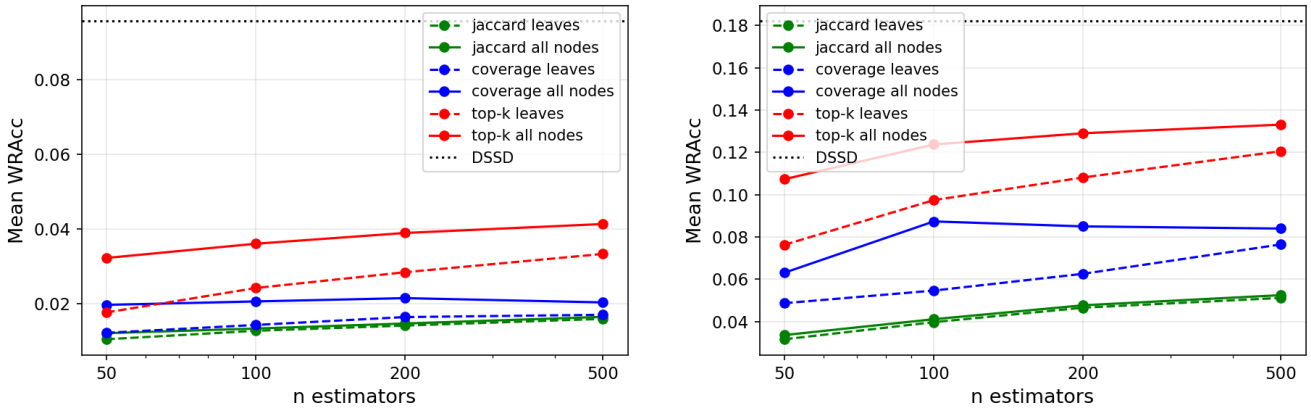
Q6: To what extent does the quality gap between DSSD and RF-DSSD narrow when only comparing metrics on classes they both cover?

# 5 Results

## 5.1 Estimator Sweep

The following section covers the results of an initial experiment examining the effect of increasing the estimator count on various metrics. The metrics of the final output of the RF-DSSD are presented for varying numbers of estimators, specifically models using 50, 100, 200, and 500 estimators to generate their respective candidate pool while also using $\tau = 0.1$. $\tau = 0.1$ was used to focus on promising nodes during extraction, as purer nodes could potentially have a higher predictive quality leading to higher WRAcc candidates being extracted.

Full results are shown in Appendix A.

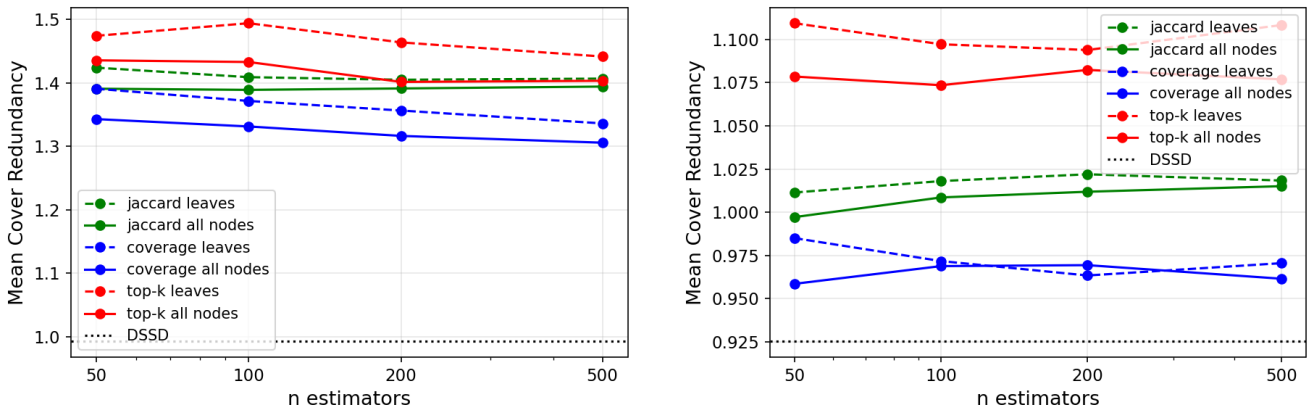



(a) Adult

(b) Mushroom

Figure 4: Mean WRAcc across estimator counts for (a) Adult and (b) Mushroom. The x-axis shows the number of estimators and the y-axis shows the mean WRAcc averaged across classes. Each line corresponds to a method and node type combination as shown in the legend. The dotted black line indicates the DSSD baseline.

Figure 4 shows the average WRAcc scores across two datasets. Three different selection methods are used, each with two variations, namely candidate generation using rule extraction only from leaves, vs all nodes. Both plots show that as the number of estimators increases, so does the quality. However, in both plots the average quality never surpasses that of DSSD, which contrasts with the fact that RF-DSSD does achieve a higher maximum quality score than DSSD (see Appendix A). This improvement coincides with an increase in runtime (see Appendix A), though RF-DSSD remains consistently faster than DSSD throughout.




(a) Adult

(b) Mushroom

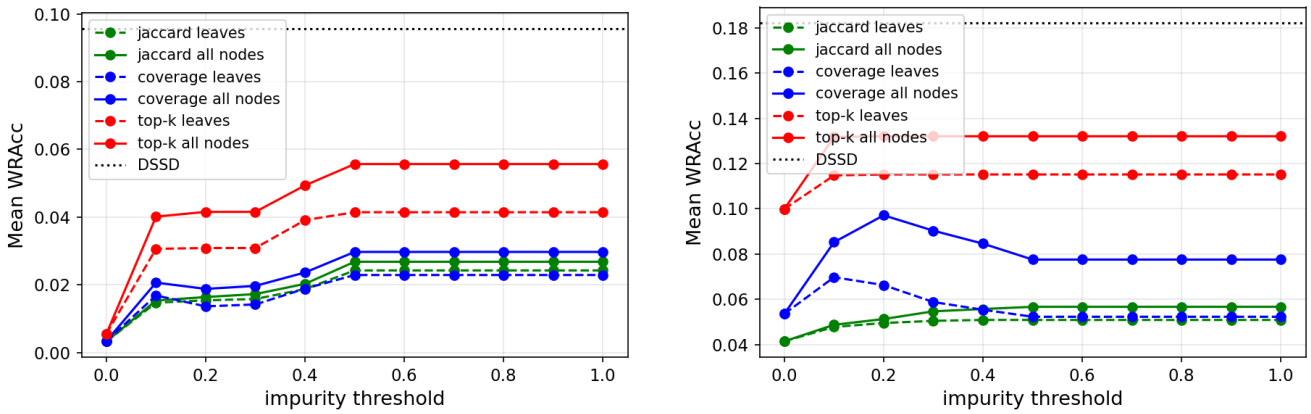
Figure 5: Mean cover redundancy across estimator counts for (a) Adult and (b) Mushroom. The x-axis shows the number of estimators and the y-axis shows the mean cover redundancy averaged across classes. Lower values indicate more diverse subgroup sets. Each line corresponds to a method and node type combination as shown in the legend. The dotted black line indicates the DSSD baseline.

The results discussed in the previous paragraph largely hold for all the listed selection algorithms except one, namely the Jaccard selection method. This selection method's mean WRAcc is the lowest across all estimator counts, even though the maximum WRAcc is universal across methods. This result co-occurs with the fact that both the Jaccard- and Coverage-based selection methods achieve good JE, and Jaccard overlap scores (see Appendix A), while the coverage-based method achieves the lowest CR.

For all selection algorithms, the CR is sizably higher than that of DSSD, and also occasionally increases as the number of estimators grows (see Plot 5), though this is more common for the top-k and Jaccard selection methods. Interestingly, the selection methods used on subgroups extracted solely from leaf nodes consistently achieve better or equally good overlap and JE scores, while for CR, the reverse holds. The results shown illustrate a good increase in quality as the number of estimators grows, while the diversity metrics mostly get worse, giving further motivation for other experimentation where both quality and diversity improve.

## 5.2 Impurity Threshold Sweep

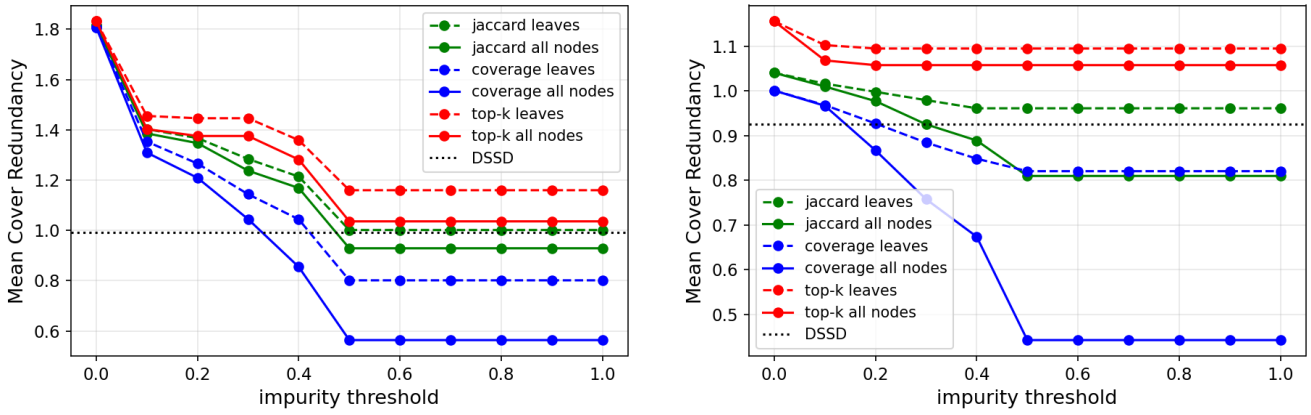
The following section covers the results of an initial experiment examining the effect of increasing $\tau$ on various metrics. The metrics of the final output of RF-DSSD are presented for $\tau$ ranging from 0.0 to 1.0 in steps of 0.2, while using 300 estimators. Full results are shown in Appendix B.




(a) Adult (b) Mushroom

Figure 6: Mean WRAcc across $\tau$ values for (a) Adult and (b) Mushroom. The x-axis shows $\tau$ and the y-axis shows the mean WRAcc averaged across classes. Each line corresponds to a method and node type combination as shown in the legend. The dotted black line indicates the DSSD baseline.

Figure 6 shows two plots, (a) and (b), that show average WRAcc scores across two datasets. As seen in the plots, the average quality of subgroups rises steadily as $\tau$ is raised, with the exception of the results shown in plot (b) of the coverage-based selection method. This discrepancy follows a similar pattern to the diversity plot, where most selection methods plateau, the coverage-based method can be seen having sharp improvements in cover redundancy.




(a) Adult (b) Mushroom

Figure 7: Mean cover redundancy across $\tau$ values for (a) Adult and (b) Mushroom. The x-axis shows $\tau$ and the y-axis shows the mean cover redundancy averaged across classes. Lower values indicate more diverse subgroup sets. Each line corresponds to a method and node type combination as shown in the legend. The dotted black line indicates the DSSD baseline.

Figure 7 shows two plots, (a) and (b), showing average CR scores across two datasets. Both plots (a) and (b) show a decrease in CR, with coverage-based standing out as the method with the lowest

score, similar to what was shown in the previous experiment. RF-DSSD's diversity-aware methods score better than DSSD on the diversity metrics when using a high $\tau$, save for a few cases where a selection type is slightly worse.

These improvements correspond to an increase in the runtime of RF-DSSD, where in most cases RF-DSSD terminates faster than DSSD, with only the all-nodes coverage-based method having a runtime close or slightly worse than DSSD. Though from these results, besides the fact that the coverage-based method's runtime increases sharply for higher $\tau$, it could seem possible that impurity improves most metrics more efficiently. This claim is further explored in the next experiment, where the improvement of metrics per second of runtime added is compared between the results gained from the estimator sweep, and the impurity sweep.

Lastly, the pattern observed in the previous section, where it was noted that both JE and Jaccard overlap are better for selection methods using a leaf-only candidate pool, while CR is better when extracting from all nodes, still holds in this experiment.

## 5.3 Estimator vs Impurity Efficiency

This section describes the results gained from an experiment where the rate of improvement per added second of runtime in the estimator sweep is compared against that of the impurity sweep. The difference of these numbers is presented in Table 3, where the numbers shown can be viewed as impurity sweep rate of improvement − estimator sweep rate of improvement.

| Mode | Nodes | WRAcc/s | JE/s | CR/s | Overlap/s |
|---|---|---|---|---|---|
| *Adult* | | | | | |
| coverage-based | all nodes | 0.0000 | -0.0006 | -0.0008 | 0.0000 |
| coverage-based | leaves | -0.0000 | 0.0157 | -0.0030 | -0.0012 |
| greedy jaccard | all nodes | -0.0000 | 0.0243 | -0.0055 | -0.0024 |
| greedy jaccard | leaves | -0.0000 | 0.0609 | -0.0125 | -0.0054 |
| top-k | all nodes | -0.0001 | 0.1908 | -0.0085 | -0.0135 |
| top-k | leaves | 0.0003 | 0.2840 | -0.0339 | -0.0318 |
| *Mushroom* | | | | | |
| coverage-based | all nodes | -0.0011 | -0.0039 | -0.0039 | -0.0082 |
| coverage-based | leaves | -0.0076 | 0.4611 | -0.0423 | -0.0410 |
| greedy jaccard | all nodes | -0.0032 | 0.0241 | -0.0310 | -0.0189 |
| greedy jaccard | leaves | -0.0011 | 0.2109 | -0.0630 | -0.0024 |
| top-k | all nodes | -0.0149 | 1.6212 | -0.0171 | -0.1779 |
| top-k | leaves | -0.0109 | 2.2743 | -0.0731 | -0.1492 |

Table 3: Impurity vs estimator efficiency (impurity - estimators). The table shows the difference of improvement per second (denoted by "/s") between data from the estimator sweep and the impurity sweep.

The results in Table 3 show that the difference in WRAcc improvement per second is fairly equal for the diversity-aware methods, with there only being relatively small differences in the Mushroom

dataset. The top-k method's WRAcc clearly improves less efficiently by increasing the number of estimators when running on the Mushroom dataset.

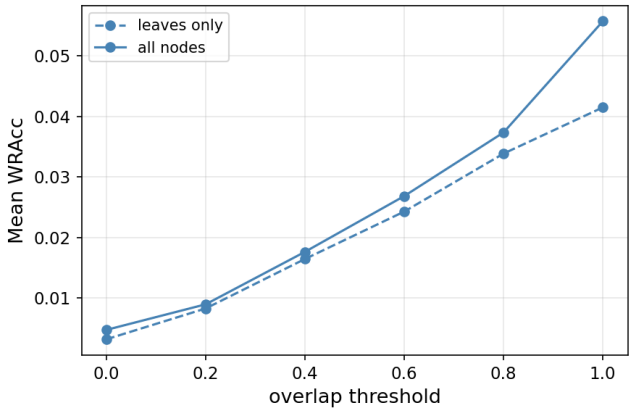
Contrasting a slower WRAcc improvement is that of the diversity metrics, where the changes in $\tau$ clearly bring more efficient improvements. This can be seen by the fact that the differences shown in the table are mostly in the favorable direction for their corresponding metric, though it should be noted that for some metrics, the actual value still deteriorates, only at a slower rate. An example of better diversity seen when modulating $\tau$, was a consistent and substantial reduction in CR (See Plot 23). This is unlike what is found when performing the estimator sweep, which shows a fairly flat plot, as seen in Figure 16.

It should be noted that CR and JE are the metrics which show the most improvement under a higher $\tau$, as Jaccard only deteriorates more slowly, and does not become better than what was found during the estimator sweep. Interestingly, top-k seems to be the method which benefits the most from a higher $\tau$ when considering diversity. The most apparent change is that of the JE, where the joint entropy was seen decreasing in the estimator sweep (See figure A) and almost always increasing in the impurity sweep (See figure B). This could possibly have resulted in the large JE/s stat in Table 3. Given this, while the speed at which WRAcc changes does not meaningfully improve, neither does it deteriorate much, leading to a clear benefit from using a higher $\tau$ with better diversity as the end goal.

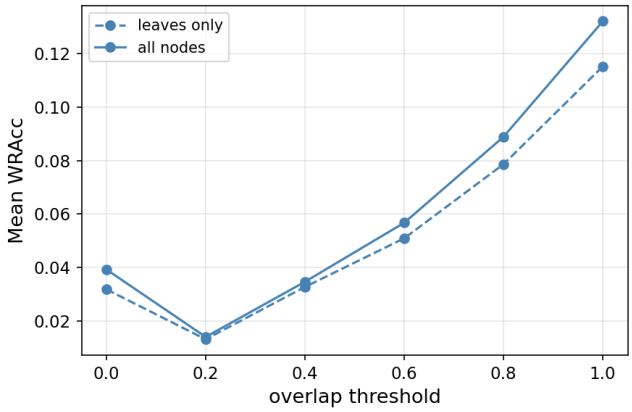
When comparing the different selection methods, coverage-based all nodes benefits the least from the impurity sweep compared to the others. This coincides with the fact that coverage-based all nodes has the longest runtime by far on both datasets. However, it does achieve the lowest CR when not taking into account the added runtime, with the other diversity metrics being similar to that of the Greedy Jaccard method.

## 5.4 Parameter Sweep

This section describes the results gained from a parameter sweep that was performed to determine reasonable parameter values for the greedy Jaccard, and coverage-based selection methods. Full results are shown in Appendix C.

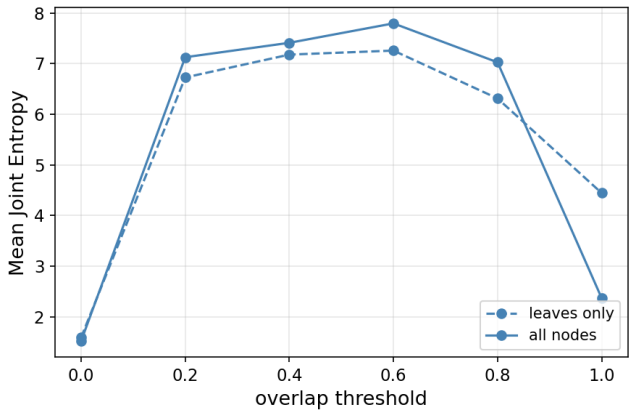



(a) Parameter sweep done on Greedy Jaccard using the Adult dataset. The x-axis shows $\theta$, and the y-axis shows the mean WRAcc.

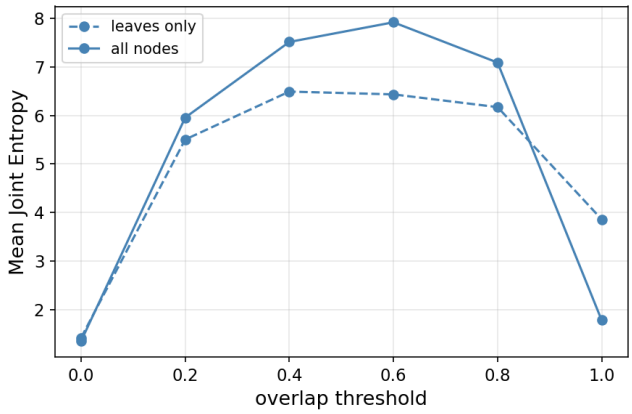



(b) Parameter sweep done on Greedy Jaccard using the Mushroom dataset. The x-axis shows $\theta$, and the y-axis shows the mean WRAcc.

Figure 8: Mean WRAcc across overlap thresholds.



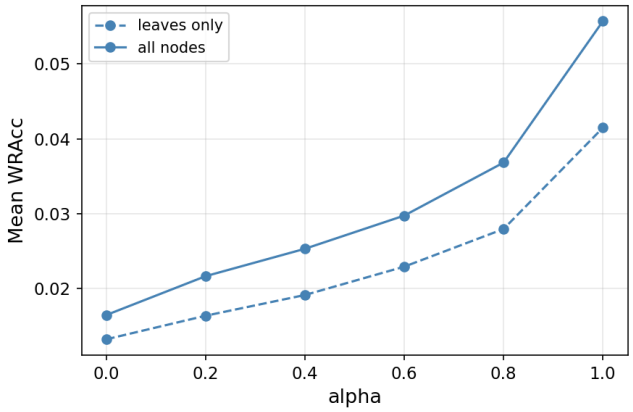

(a) Parameter sweep done on Greedy Jaccard using the Adult dataset. The x-axis shows $\theta$, and the y-axis shows the mean joint entropy.



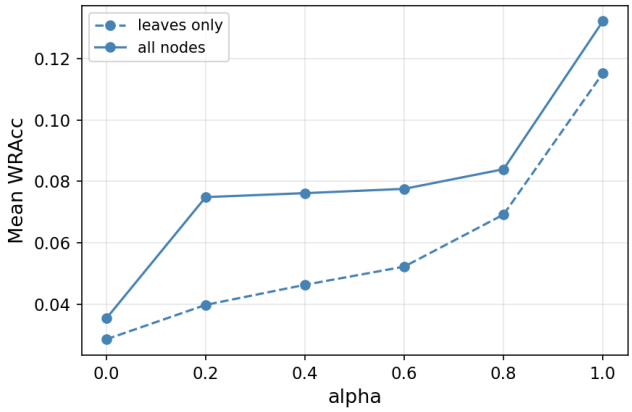

(b) Parameter sweep done on Greedy Jaccard using the Mushroom dataset. The x-axis shows $\theta$, and the y-axis shows the mean joint entropy.

Figure 9: Mean joint entropy across overlap thresholds.

Figure 8 shows that higher overlap-thresholds lead to higher average quality subgroups, with only a small initial decrease being present in the sweep performed on the Mushroom dataset. Figure 9 on the other hand, shows a steep initial growth in JE, after which it remains relatively stable until collapsing back to a similar initial value. As RF-DSSD aims to find a diverse set of subgroups, the extra WRAcc gain after 0.8 is not considered to be beneficial, given the fact that the diversity metrics drop steeply afterwards (See Appendix C).

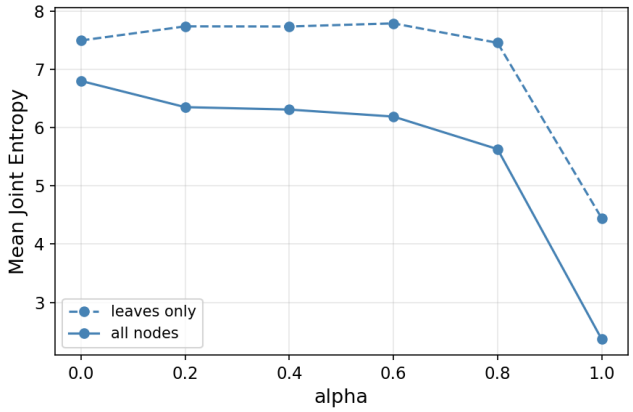



(a) Parameter sweep done on Coverage-based selection using the Adult dataset. The x-axis shows $\alpha$, and the y-axis shows the mean WRAcc.

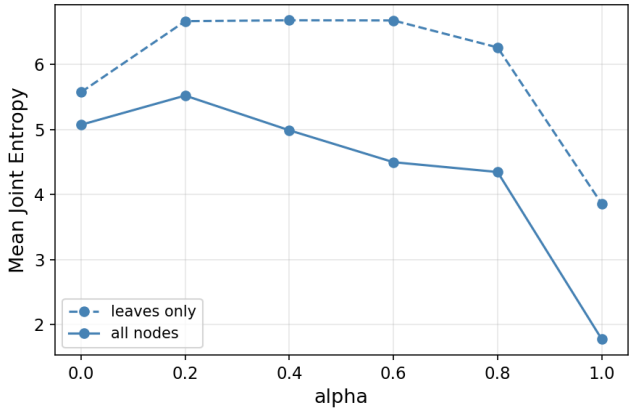



(b) Parameter sweep done on Coverage-based selection using the Mushroom dataset. The x-axis shows $\alpha$, and the y-axis shows the mean WRAcc.

Figure 10: Mean WRAcc across alpha values.




(a) Parameter sweep done on Coverage-based selection using the Adult dataset. The x-axis shows $\alpha$, and the y-axis shows the mean joint entropy.




(b) Parameter sweep done on Coverage-based selection using the Mushroom dataset. The x-axis shows $\alpha$, and the y-axis shows the mean joint entropy.

Figure 11: Mean joint entropy across alpha values.

Figure 10 shows that a higher $\alpha$ leads to higher average quality subgroups. Figure 11 shows a slowly decreasing JE, with the collapse occurring specifically at the extreme value of 1.0. As RF-DSSD aims to find a diverse set of subgroups, the extra WRAcc gain after 0.8 is not considered to be beneficial, given the fact that the diversity metrics drop steeply afterwards (See Appendix C). However, considering that DSSD uses 0.9 in its selection, and that the collapse only occurs at the extreme value of 1.0, 0.9 was used as the value for $\alpha$ moving forward.

## 5.5 Full Dataset Evaluation

The following section describes the results gained from averaging the different metrics across datasets. This is done for every RF-DSSD selection method, as well as the DSSD baseline, to allow for a general comparison between approaches.

| Method | WRAcc | Max WRAcc | Jaccard | CR | JE | Runtime (s) |
|---|---|---|---|---|---|---|
| coverage all nodes | 0.0400 | 0.0752 | 0.5854 | 1.0832 | 3.0699 | 915.2163 |
| coverage leaves | 0.0386 | 0.0743 | 0.5108 | 1.2964 | 3.6841 | 152.2610 |
| jaccard all nodes | 0.0373 | 0.0752 | 0.4753 | 1.2757 | 4.3878 | 75.3687 |
| jaccard leaves | 0.0334 | 0.0743 | 0.4332 | 1.3709 | 3.9303 | 32.8510 |
| top-k all nodes | 0.0517 | 0.0752 | 0.8130 | 1.4070 | 1.3445 | 26.3021 |
| top-k leaves | 0.0457 | 0.0743 | 0.6873 | 1.4315 | 2.5260 | 15.1918 |
| DSSD | 0.0855 | 0.1029 | 0.8562 | 0.9380 | 2.3755 | 1389.8571 |

Table 4: Average metrics for each RF-DSSD selection method and extraction type, macro-averaged across all datasets at 300 estimators and $\tau = 1.0$ (see Section 5.3 for the reasoning behind this choice of $\tau$), along with the DSSD baseline.

Table 4 shows that for the different RF-DSSD selection methods, they score highest in their respective specializations. This means that Greedy Jaccard has the lowest mean overlap, Coverage-based has the lowest CR, and top-k has the highest mean WRAcc. DSSD outperforms all RF-DSSD methods when it comes to WRAcc and CR, though this comes at the cost of added runtime, with all RF-DSSD being substantially faster than DSSD. The diversity-aware selection methods and top-k leaves achieve a better JE and overlap score however, which is interesting considering that top-k leaves is still a quality-based selection method. Another point worth mentioning is that of class coverage, as RF-DSSD achieves 100% class coverage, whereas DSSD achieves 72.6% class coverage.

Furthermore, as was previously observed within selection types, the ones using all nodes extraction score better on quality and CR, whereas the leaves only subgroups score better on JE and mean overlap. As expected, the maximum WRAcc achieved for RF-DSSD methods depends on the extraction type as well, with all leaves extraction resulting in a slightly higher max WRAcc.

## 5.6 Data Subset Comparison

This section describes the results gained from an experiment where the subgroups RF-DSSD generated were further pruned to match the DSSD class distribution. The first method, dubbed class-subset restriction, runs the metric calculations on RF-DSSD's subgroups again, only this time, just subgroups predicting classes that DSSD covered are included. The second method attempts to further this pruning by also taking into account the class ratios found in DSSD's results. This is done by using said ratios to extract a final result from RF-DSSD's subgroups that best matches the ratio calculated in DSSD's results.

It is to be noted that when a class does not have enough subgroups to adhere to the ratio, the achieved distribution can fall short of DSSD's true ratio. Despite this, Table 5 shows that

ratio-matching still improved the macro mean WRAcc over class-subset restriction alone for both method groups. Furthermore, comparing the class distributions using total variation distance [8] confirmed that the ratio-matched subgroups more closely resemble DSSD's own distribution than the class-subset restriction alone. This further justifies using ratio-matching over subset restriction, as it achieves both a higher macro mean WRAcc and a closer match to DSSD's class ratios.

| Group | RM % vs Full | RM % vs Covered | DSSD % vs Full | DSSD % vs RM |
|---|---|---|---|---|
| Div. | +30.7% | +6.5% | +128.9% | +75.1% |
| Qual. | +27.6% | +1.1% | +75.4% | +37.4% |

Table 5: RF-DSSD mean WRAcc under DSSD-covered-class restriction and ratio-matched (RM) per-class selection, averaged across node types, methods, and datasets within each method group. '% vs X' denotes the percentage change relative to X.

Table 5 shows that ratio-matching substantially increases the average WRAcc, though when comparing the percentage increase of ratio-matching to class-subset restriction, the changes are noticeably smaller. This indicates that while ratio-matching does further increase the average WRAcc, it is not as substantial as going from the full subgroup average to the class-subset restricted average.

As previously seen, DSSD outperforms RF-DSSD in absolute WRAcc even when using ratio-matching. This is expected given the results from Table 4 where it was shown that the max-WRAcc that DSSD achieved was substantially higher than that of RF-DSSD. Lastly, the top-k selection method benefits the least from both class-subset restriction, and ratio-matching. Though, as expected, top-k using ratio matching does get the closest to DSSD's average WRAcc.

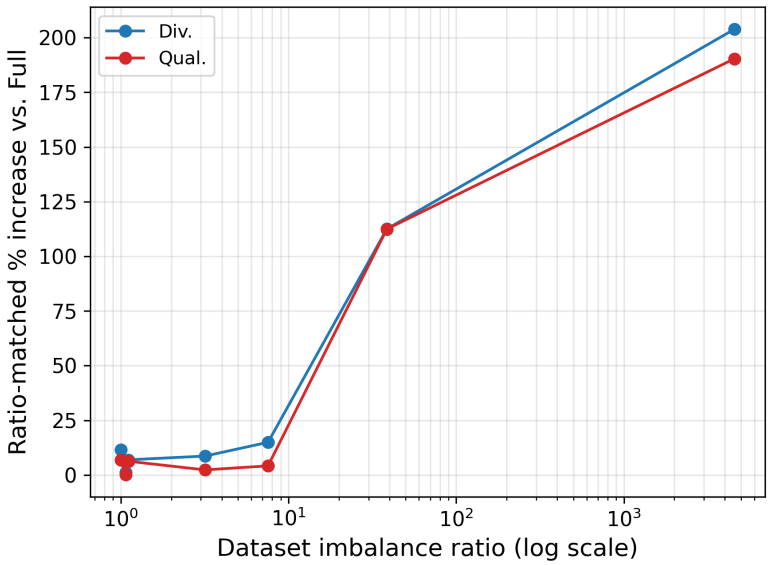



Figure 12: Effect of ratio matching across different imbalance ratios. The x-axis represents the imbalance ratio described in Section 4, and the y-axis shows the percentage increase resulting from ratio matching. Div. denotes diversity-aware selection methods, whereas Qual. denotes quality-driven selection methods. The corresponding datasets can be identified by comparing the imbalance ratios with those reported in Table 1.

Figure 12 shows the results from an experiment where class imbalance was measured against the WRAcc increase that ratio-matching provided. The plot shows that for datasets with an imbalance ratio under 10, the WRAcc increase is fairly consistent. However, two outliers with far greater imbalance ratios exist, both of which benefit considerably more from ratio-matching. This suggests that beyond a certain level of imbalance, quality-related issues may begin to emerge.

# 6 Discussion

## 6.1 Estimator Sweep

As the number of estimators increases, the candidate pool grows, and mean and max WRAcc improve accordingly (See Figure 4). The exact mechanism $\tau$ attempts to affect is that of accuracy, as a higher accuracy leads to a higher WRAcc, as seen in Formula 1. The all-nodes method plateaus in terms of max-quality as the number of estimators increases (See Appendix A), whereas the leaves-only method shows consistent growth. A reason for this could be that all-nodes extraction results in more subgroups being extracted, which increases the chance of finding high-quality subgroups. It is also possible that the higher purity of leaf nodes is detrimental to the quality, rather than beneficial. This is supported by the fact that all-nodes extraction always leads to a higher mean WRAcc and a higher max WRAcc.

RF-DSSD's diversity metrics are consistently worse than those of DSSD. Additionally, only JE is shown to improve consistently during the sweep (See Appendix A). For the metrics that show

improvements during the sweep, every improvement comes from roughly doubling the candidate pool's size. Consequently, maintaining this rate of improvement would require exponential growth in runtime (See Appendix A). Furthermore, previous research shows that RFs suffer from diminishing returns when increasing the estimator count [10], which might also manifest when using RFs as a candidate generation mechanism. Despite this, all RF-DSSD configurations are faster than DSSD, possibly due to a difference in candidate pool size.

Beyond these general trends, individual selection methods also show interesting behavior. Greedy Jaccard's mean WRAcc is the lowest, though the max WRAcc is solely dependent on whether extraction used all-nodes or leaves-only (See Appendix A). A possible reason for this is that Greedy Jaccard excludes subgroups entirely once they exceed $\theta$, unlike coverage-based, which merely discounts them. Another consequence of Greedy Jaccard's aggressive pruning method is its inflated JE, likely due to the selection pool only consisting of subgroups describing many different parts of the data.

A recurring pattern can be found when comparing leaves-only and all-nodes extraction, where all-nodes resulted in higher mean and max WRAcc. As previously stated, this likely reflects subgroup size, not just purity, driving quality. There exists a pattern in the diversity metrics as well (See Appendix A), where all-nodes scores are worse in terms of JE and mean overlap, and higher in terms of CR. This likely stems from non-leaf nodes being larger on average, which increases the likelihood of overlap and possibly of similarity between subgroups. Larger covers also produce a more even coverage spread, giving all-nodes a better CR than leaves-only, at the cost of worse JE and mean overlap. While a higher estimator count improves average quality, its diminishing returns, combined with the finding that purity alone does not drive WRAcc, motivate testing $\tau$ directly.

## 6.2 Impurity Threshold Sweep

In general, increasing $\tau$ improves all metrics except mean overlap. As discussed previously, the increase in mean Jaccard overlap and WRAcc may result from larger subgroup covers at higher values of $\tau$. Impure nodes sit high up in a decision tree, with the purest nodes being leaf nodes. Because nodes which are split fewer times are larger, raising $\tau$ adds larger subgroups to the candidate pool. RF-DSSD's diversity and quality improvements plateau at $\tau = 0.5$ (See Appendix B) [1]. However, because runtime also plateaus and remains at or below that of DSSD, the threshold was set to 1, allowing for any possible improvements beyond 0.5.

Visual inspections of the results suggest that raising $\tau$ could be more efficient than increasing the number of estimators, though it is unclear if this holds for WRAcc. Both CR and JE improve rather than worsen during the impurity sweep, in contrast to the estimator sweep. These facts motivate the next experiment, which will compare the rate at which the quality and diversity metrics improve in both sweeps.

[1] It is to be noted that the runtimes still show some variability after the plateau (See Appendix B). This is likely not a consequence of RF-DSSD itself, but rather artifacts from some performance fluctuations on the device used to run the sweep.

## 6.3 Estimator vs Impurity Efficiency

The rate of growth in terms of quality is fairly equal, as only top-k, when run on the Mushroom dataset (See Table 3), shows that adding estimators increases the quality significantly better than raising $\tau$. This is deemed to be acceptable given the fact that top-k is used as a baseline method, meant to measure the maximal quality an RF candidate generation technique achieves. Coverage-based all-nodes also shows a comparable pattern in both sweeps, with its mean quality beginning to dip past a certain point, an effect most visible on Mushroom. Since this occurs consistently across both the estimator and impurity sweeps, rather than being isolated to one setting, it likely reflects a deeper, method-specific cause, though the Mushroom dataset may also be amplifying this effect.

The diversity metrics are shown to improve more efficiently for the impurity sweep, barring mean overlap, which simply worsens slower. For CR, the previously discussed idea, stating that allowing for a higher $\tau$ leads to an increase in size, leading to more equal coverage, holds. The top-k method shows a disproportionate increase in JE, likely due to an estimator increase leading to worse JE, whereas raising $\tau$ increases it. A possible reason that a higher $\tau$ leads to these improvements is that raising the threshold allows for more extensive extraction per tree, rather than requiring entirely new trees to be added. This increases the candidate pool's size without introducing the same redundancy risk associated with adding additional estimators [10].

These results show that raising $\tau$ rather than the number of estimators is the more productive decision overall. Although top-k's WRAcc growth on Mushroom, and coverage-based all-nodes' dip in both sweeps, likely amplified by the Mushroom dataset, appear to suggest otherwise, this exception does not change the overall conclusion. Adult is larger and has a genuine class imbalance, which led its results to be more seriously considered. Furthermore, coverage-based all-nodes' quality dip was accompanied by a significant reduction in CR, suggesting this loss in quality was not without benefit. Thus, the final experiments used a maximum $\tau$ of 1.0 and the same estimator count as the impurity sweep. These parameters balance diversity, quality, and runtime.

## 6.4 Parameter Sweep

The plot showing the JE that Greedy Jaccard achieved was shown to be quite symmetrical (See Figure 9). The initially low JE may result from Greedy Jaccard's aggressive pruning, which can select fewer candidates than desired per run (See Appendix C). The coverage-based method avoids this issue, since candidates are never removed, simply discounted. For both methods, when their respective parameters are set to 1, they theoretically revert to a greedy top-k approach, explaining the dip in JE.

In the final experiments, the parameters chosen for Greedy Jaccard and Coverage-based are 0.8 and 0.9, respectively. For Greedy Jaccard, the reasoning behind choosing 0.8 is that the rate of JE decline noticeably increases beyond 0.7. Given the marginal WRAcc gain beyond this point, 0.8 offers a reasonable balance between quality and diversity. As for $\alpha$, the JE collapse likely occurs at the extreme value of 1, where discounting is fully removed. Thus, a value of 0.9 achieves a higher WRAcc and reasonable JE, while matching DSSD's own configuration.

### 6.5 Full Dataset Evaluation

The same extraction type pattern is found in the aggregated results (See Table 4), which suggests subgroups extracted from impure nodes tend to be of higher quality, even across various datasets. As for the class coverage discrepancy between RF-DSSD and DSSD: DSSD would be able to close this gap, but it would require an entirely separate candidate generation and selection step per class. In contrast, RF-DSSD only requires one candidate generation step, after which candidates can be selected per class. The reason for DSSD's lack of global class coverage could be that its candidate generation technique is still quality-driven. This means that the final candidate pool could lack certain underrepresented, and thus lower quality classes.

Globally, DSSD outperforms every RF-DSSD variant in quality and CR, but has a far higher mean overlap. This builds on the earlier size-driven explanation, but points to a more fundamental distinction: mean overlap is a local, pairwise measure of similarity between individual subgroups, while CR reflects a global, dataset-wide distribution of coverage. Additionally, an uneven class coverage would lead to subgroups being more concentrated in individual classes. The combination of these factors, including the fact that DSSD optimizes for CR, explains the striking difference in metrics.

Overall, while RF-DSSD does not match DSSD's raw quality under any configuration tested, it consistently offers substantially better runtime, full class coverage, and, depending on the selection method used, comparable or better diversity. While it is not possible for RF-DSSD's WRAcc to exceed that of DSSD, given that DSSD's mean WRAcc is higher than RF-DSSD's max WRAcc, the experiment may still provide insight into how much underrepresented classes affect the final quality score.

### 6.6 Data Subset Comparison

For both class restriction methods, DSSD still outperforms RF-DSSD in terms of WRAcc. Although top-k's WRAcc came a fair bit closer to DSSD, the difference remains large enough to suggest that the current extraction-selection combination limits the quality of the final result.

An additional observation shows that imbalanced datasets benefit the most from class-subset restrictions (See Figure 12). This is likely due to the fact that for datasets with very underrepresented classes, it can be hard to generate high-quality subgroups, given the fact that WRAcc is partially based on subgroup size. Prior work similarly shows that cover size and class balance have an intricate influence on subgroup performance metrics [11], suggesting that the underrepresentation of certain classes may similarly affect subgroup quality in this context.

## 7 Limitations

This chapter details the various limitations encountered in the making of this thesis. In the discussion various claims have been made with regards to possible mechanisms causing certain patterns to manifest in the results. These claims, while supported by the results and known phenomena, have not always been rigorously checked. Some examples of this include, but are not limited to: the claim made in Section 6.5 explaining the reasoning behind the overlap/CR

discrepancy in DSSD and RF-DSSD, as well as a claim made in the same section, where it was stated that DSSD's lack of full class coverage is due to its candidate generation mechanism, which is likely still quality-driven beam search. While these claims are founded on logical assumptions, they are still only assumptions, and further experimentation would have to be done to confirm this.

Although Mushroom and Adult were chosen as two representative datasets with different characteristics, the method of deciding which parameters to select for RF-DSSD was only performed on these two datasets. This means that the final selection of parameter values may reflect dataset-specific behavior, rather than settings that generalize well across different datasets. This is a genuine limitation, and ideally the parameter selection should be performed on a larger and more diverse set of datasets. This is especially important given the fact that Mushroom and Adult presented contradicting results in terms of quality gain during the impurity sweep, suggesting that even across these two datasets the preferred parameter behavior is not fully consistent. The final aggregated result of RF-DSSD (See Table 4) was done using 7 datasets. While this is a reasonable amount to derive an average from, two of the datasets are not typically used in SD, motivating further research to experiment on a larger collection of datasets. Additionally, for this average result, as well as the class restriction experiment, CIs are absent. This makes it unclear whether the observed effects reflect genuine patterns or fall within the natural variance of the underlying data.

Regarding the comparison with DSSD, there exist a few other design decisions which could possibly lead to a level of unfairness in the final results. Aside from the claim made regarding DSSD's quality-driven candidate generation, a decision was made to perform DSSD runs on the OneVsRest setting, leading to the uneven class coverage. This could have been avoided by performing a single run per class. However, this was deemed too computationally intensive for the scope of this thesis, as it would likely result in a relatively large increase in runtime. It is also worth noting that RF-DSSD's own candidates are generated and selected based on their predicted class, rather than being optimized against a cross-class-comparable measure such as MWRAcc, unlike DSSD's candidates.

Alongside this, the way the target class is assigned to the subgroups found using DSSD is only an estimation based on which class obtains the highest WRAcc. This is necessary because DSSD's own quality measure, MWRAcc, never computes or requires a per-subgroup target class, meaning DSSD's results do not include this information natively, and some subgroups' assigned targets could be inaccurate as a result. However, the test which checked to see if any subgroup's target class could be ambiguous showed that only non-standard SD datasets resulted in ambiguous targets, suggesting the max WRAcc is likely a serviceable approximation.

# 8 Future Work

This chapter discusses various directions for future work which could prove to be interesting in the domain of SD. When it comes to RF-DSSD, a limitation which was observed using the approach outlined in this paper was that of quality. DSSD remains ahead of RF-DSSD in terms of quality by a substantial amount. To address this issue, two new ideas will be proposed.

First, with the current candidate extraction approach, subgroups only get added to a class's candidate pool if the node it was extracted from is predicting said class. While this is a reasonable

approach, as it would mean that the subgroup is likely to be accurate, another approach would be to order all subgroups on the basis of WRAcc. More specifically, per class, the entire candidate pool would be considered, with ultimately only the higher-scoring subgroups being kept. This approach could potentially lead to higher quality subgroups being added to certain classes' candidate pools that the current approach might not. Currently, a node might contain a good description for an underrepresented class, but is excluded from that class's pool because a more dominant class holds the node's majority. This would likely be more commonly observed in imbalanced datasets, as small priors lower the number of candidates required for a positive WRAcc.

While the final candidate pool would be much larger, it is unclear whether this would lead to more or less redundancy. It is possible that the newly added subgroups are different enough that they result in a more diverse final result, but it is also possible that they would end up increasing the candidate pool size while potentially introducing more redundancy.

Second, most RF-DSSD variants had far faster runtimes when compared to DSSD. Given this, another way to close the gap is to further increase the depth and number of estimators in the RFs. As previously stated, this is not likely to result in closing the gap fully, but is worth exploring nonetheless. One factor to consider is that by increasing the depth, the description length increases as well. This results in the interpretability of the final result decreasing. Considering this, it would prove useful to introduce dominance pruning into the process as well. By doing this, the description length would stay lower, while increasing the average WRAcc.

While the following conception is not based on the workings of RF-DSSD, it provides a suggestion for improving DSSD's framework in terms of full class coverage. DSSD's candidate generation is guided by MWRAcc, which sums one-vs-rest WRAcc across all classes into a single combined score per candidate [12]. This means that certain candidates belonging to naturally higher quality classes can dominate the beam, which reduces the chance that candidates that more accurately describe lower-quality classes make it through the pruning step, even though they may be relatively good descriptions for their respective class. A method to fix this would be to add class normalized quality weighting to the per-class WRAcc terms before they are summed, essentially weighting every class more, or less, depending on whether that class's average WRAcc is lower, or higher, respectively. To avoid including extremely bad quality subgroups simply on the merit of its target class having a low average WRAcc, normalized quality weighting would need a parameter to control how substantial the changes are. It is to be noted that this added overhead will increase the runtime, and further research would have to be performed in order to discern whether or not this added runtime is deemed acceptable.

# 9 Conclusion

RF-DSSD was created in order to achieve a quick and efficient diverse subgroup discovery algorithm, using random forests for candidate generation, and three selection methods, one of which is a DSSD-inspired coverage-based approach, to produce a diverse output. After numerous experiments, a few things were made clear. First, RF-DSSD's average quality requires much improvement. It did not reach a similar level of quality to DSSD, though it finished far faster. This could be addressed by no longer relying on a node's own predicted class to determine a subgroup's target, and separately, by conducting more extensive parameter sweeps across estimators, depth, and other

settings on a wider range of datasets.

Second, modulating the random forest's estimator count and impurity threshold, as well as each selection method's own parameters, was found to meaningfully affect both quality and diversity, with impurity generally having a greater effect. Using the resulting parameter setup, RF-DSSD was found to run substantially faster than DSSD and achieve full class coverage, though at the cost of lower overall quality and CR.

Third, a clear pattern was observed across the experiments, first noticed when comparing candidates extracted from leaf nodes versus all nodes: the coverage-size term in WRAcc was central to how impurity affects all metrics. This same mechanism also explains why imbalanced datasets benefit more from class-subset restriction, which closed the quality gap with DSSD, though not fully. Both imbalance and impurity, when reduced to their basic effects, indicate how important size is for a subgroup's quality. More specifically, impurity correlates with the number of preceding splits a node has, and whether or not it is a leaf node, allowing larger subgroups to be added to the candidate pool and thus increasing WRAcc. For imbalance, the opposite holds: inherently smaller subgroups lead to lower WRAcc, meaning imbalanced datasets benefit most when restricted to classes with fewer, smaller subgroups.

Lastly, this thesis also identified a possible improvement for DSSD itself. DSSD's candidate generation relies on MWRAcc, where, simply put, only globally good subgroups are found, regardless of their target's average quality. Applying class-normalized quality weighting could allow DSSD to achieve better coverage, while keeping its framework largely the same. However, this remains a direction for future work rather than something tested here. To conclude, while RF-DSSD does not yet match DSSD's raw quality, it demonstrates that a fast, full class-coverage approach to diverse subgroup discovery is viable, with several concrete directions identified for narrowing the remaining quality gap.

# References


[1] Adnene Belfodil, Aimene Belfodil, Anes Bendimerad, Philippe Lamarre, Céline Robardet, Mehdi Kaytoue, and Marc Plantevit. FSSD - a fast and efficient algorithm for subgroup set discovery. In *2019 IEEE International Conference on Data Science and Advanced Analytics (DSAA)*, pages 91–99, 2019. doi: 10.1109/DSAA.2019.00023.

[2] Leo Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001.

[3] Thomas M. Cover and Joy A. Thomas. *Elements of Information Theory*. Wiley, 2nd edition, 2006.

[4] Francisco Herrera, Cristóbal José Carmona, Pedro González, and María José del Jesus. An overview on subgroup discovery: Foundations and applications. *Knowledge and Information Systems*, 2011. doi: 10.1007/s10115-010-0356-2.

[5] Paul Jaccard. The distribution of the flora in the alpine zone. *New Phytologist*, 11(2):37–50, 1912.

[6] Willi Klösgen. Explora: A multipattern and multistrategy discovery assistant. In *Advances in Knowledge Discovery and Data Mining*, pages 249–271. AAAI Press, 1996.

[7] Nada Lavrač, Branko Kavšek, Peter Flach, and Ljupčo Todorovski. Subgroup discovery with CN2-SD. *Journal of Machine Learning Research*, 5:153–188, 2004.

[8] David A. Levin and Yuval Peres. *Markov Chains and Mixing Times*. American Mathematical Society, Providence, RI, second edition, 2017. With contributions by Elizabeth L. Wilmer.

[9] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Édouard Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[10] Philipp Probst and Anne-Laure Boulesteix. To tune or not to tune the number of trees in random forest. *Journal of Machine Learning Research*, 18(181):1–18, 2018.

[11] Tom Siegl, Kutalmış Coşkun, Bjarne C. Hiller, Amin Mirzaei, Florian Lemmerich, and Martin Becker. Subroc: Auc-based discovery of exceptional subgroup performance for binary classifiers. *arXiv preprint arXiv:2505.11283*, 2025.

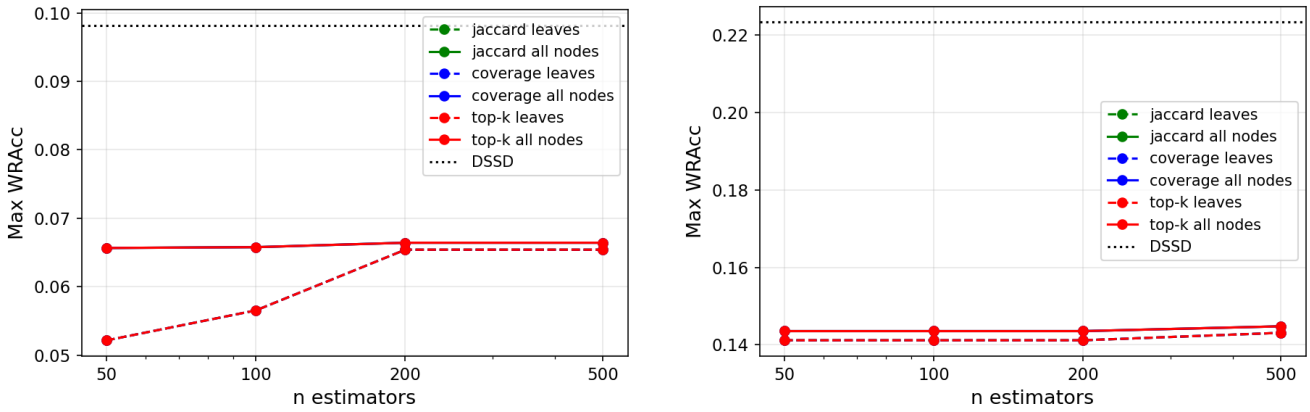
[12] Matthijs van Leeuwen and Arno Knobbe. Diverse subgroup set discovery. *Data Mining and Knowledge Discovery*, 2012. doi: 10.1007/s10618-012-0273-y.

[13] Geoffrey Webb. Opus: An efficient admissible algorithm for unordered search. *Journal of Artificial Intelligence Research*, 3:431–465, 1995.

[14] Stefan Wrobel. An algorithm for multi-relational discovery of subgroups. In *Proceedings of the 1st European Symposium on Principles of Data Mining and Knowledge Discovery*, volume 1263 of *Lecture Notes in Computer Science*, pages 78–87, 1997.

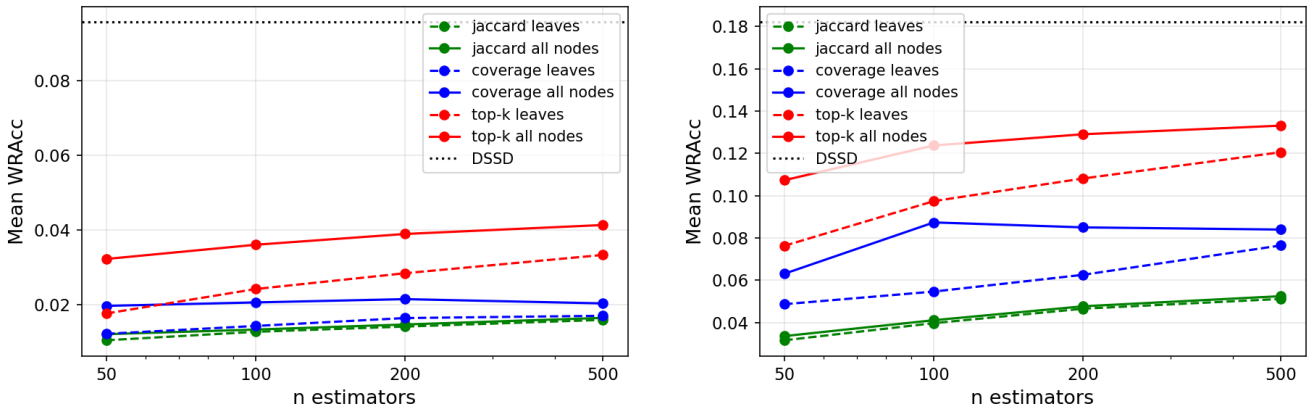
# A Estimator Sweep Plots

## Quality




(a) Adult (b) Mushroom

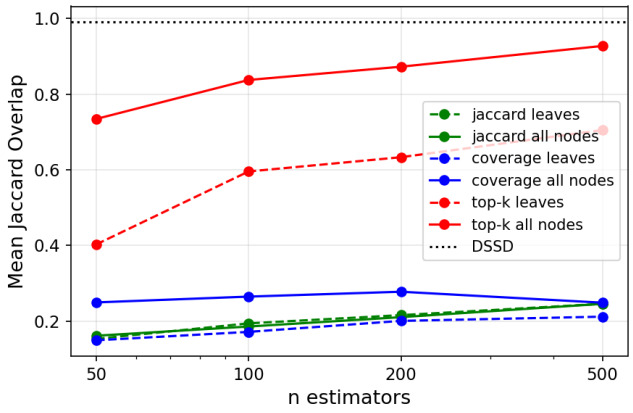
Figure 13: Max WRAcc across estimator counts.



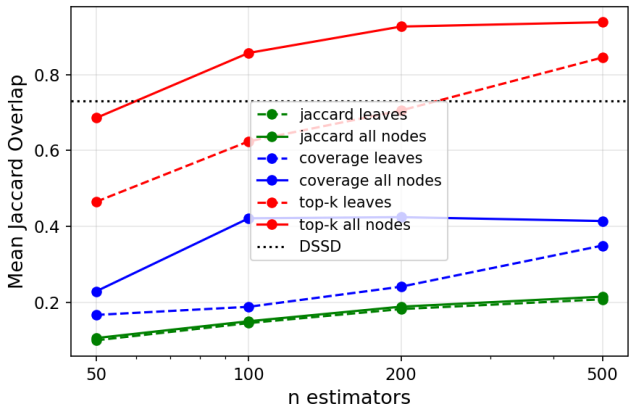

(a) Adult (b) Mushroom

Figure 14: Mean WRAcc across estimator counts.

## Diversity

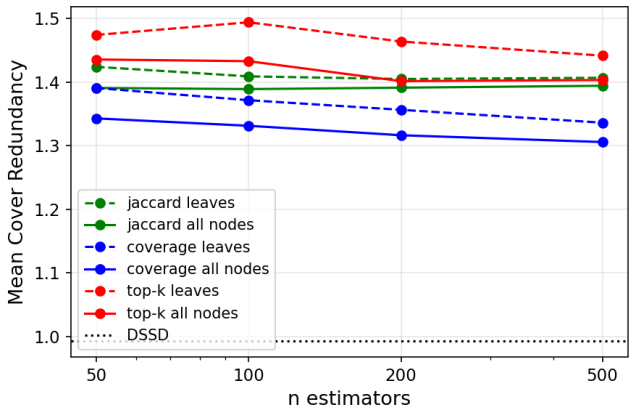



(a) Adult

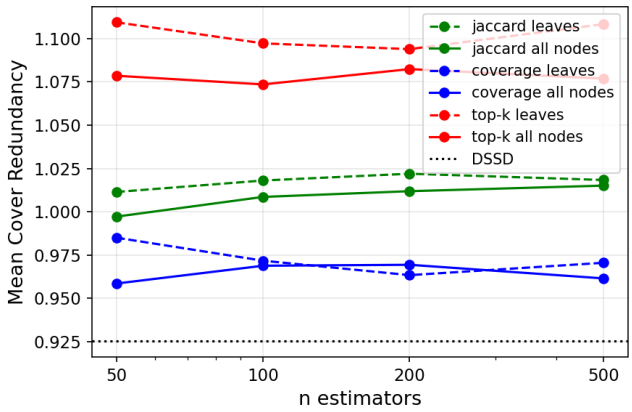



(b) Mushroom

Figure 15: Mean overlap across estimator counts.

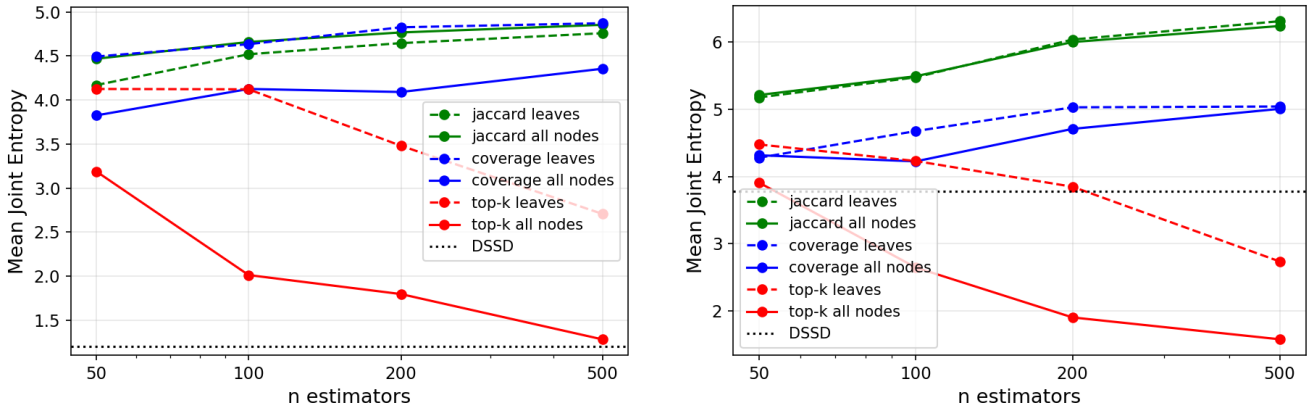



(a) Adult




(b) Mushroom

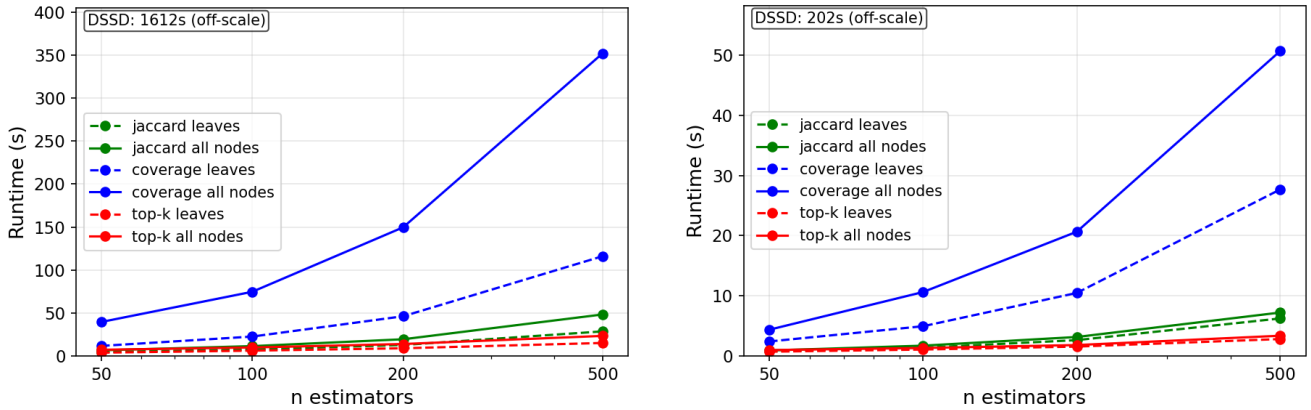
Figure 16: Mean cover redundancy across estimator counts.




(a) Adult (b) Mushroom

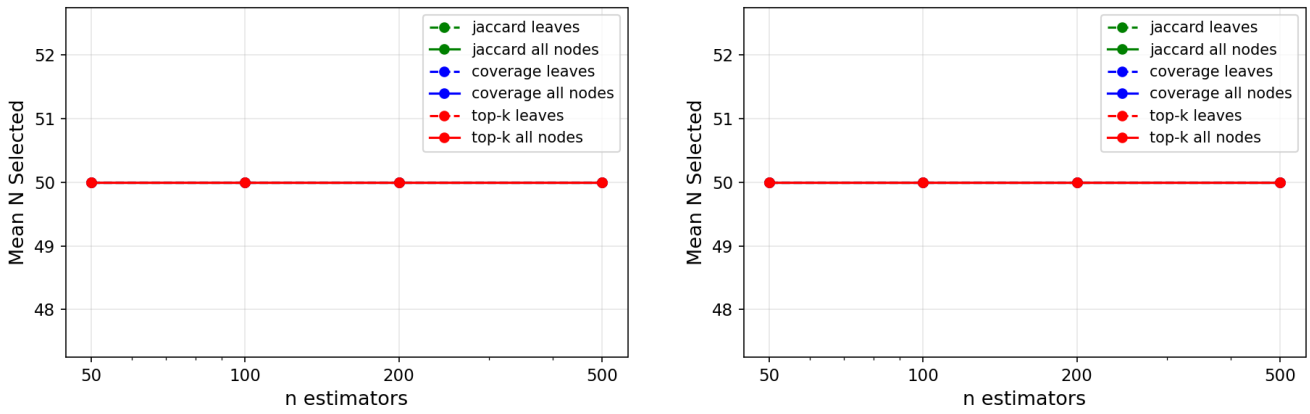
Figure 17: Mean joint entropy across estimator counts.

## Runtime & Amount Selected




(a) Adult (b) Mushroom

Figure 18: Runtime across estimator counts.

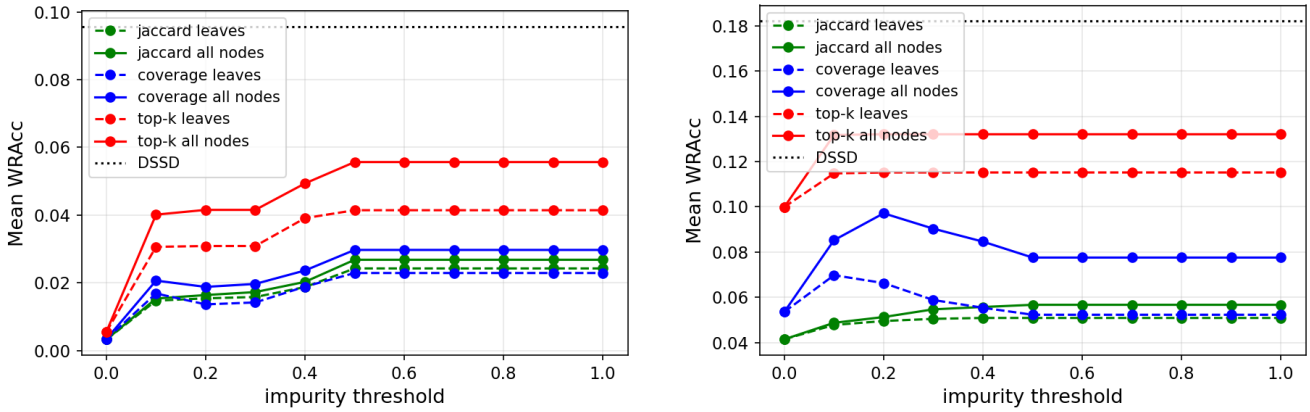



(a) Adult (b) Mushroom

Figure 19: Mean candidates selected across estimator counts.

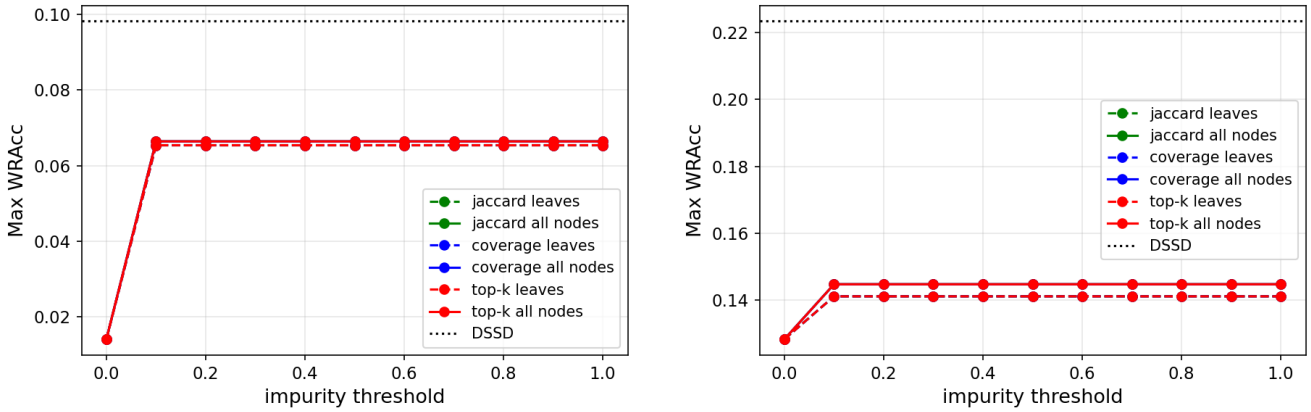
# B Impurity Threshold Sweep Plots

### Quality




(a) Adult (b) Mushroom

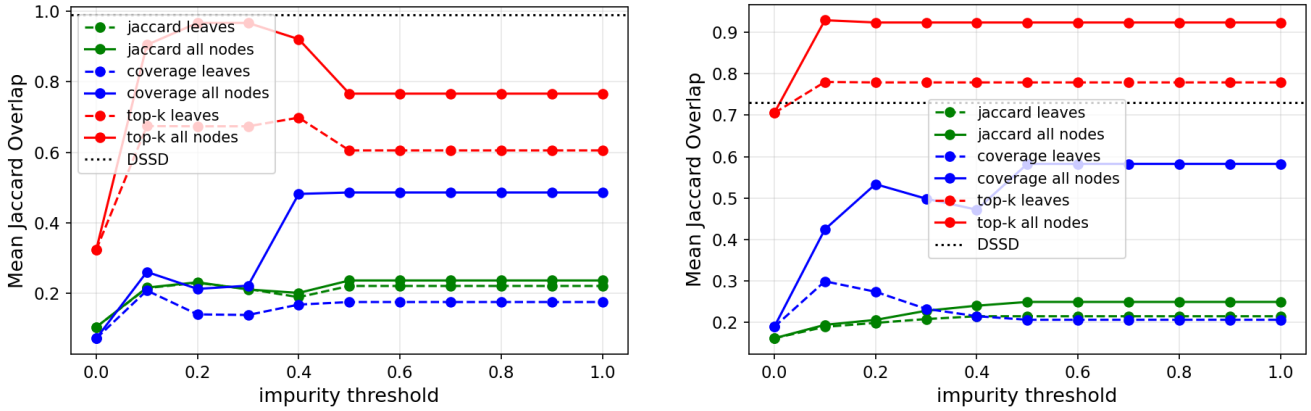
Figure 20: Mean WRAcc across impurity thresholds.




(a) Adult (b) Mushroom

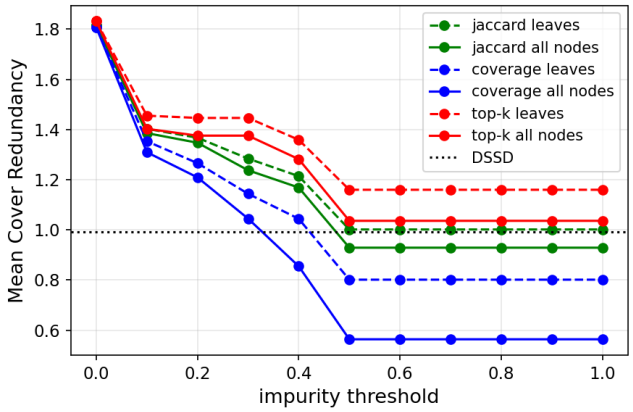
Figure 21: Max WRAcc across impurity thresholds.

## Diversity

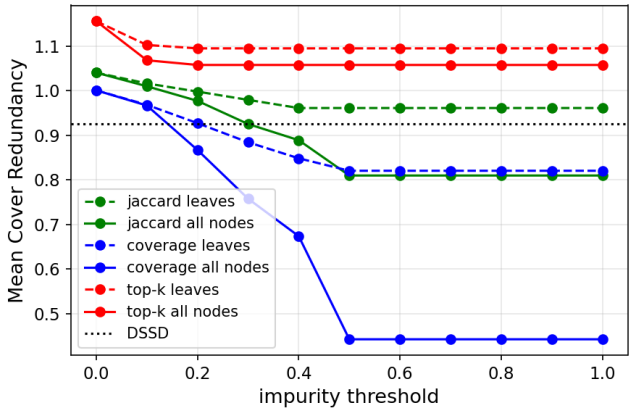



(a) Adult (b) Mushroom

Figure 22: Mean Jaccard overlap across impurity thresholds.

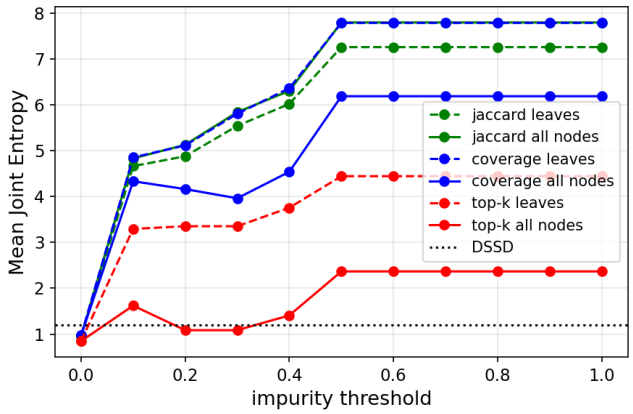



(a) Adult

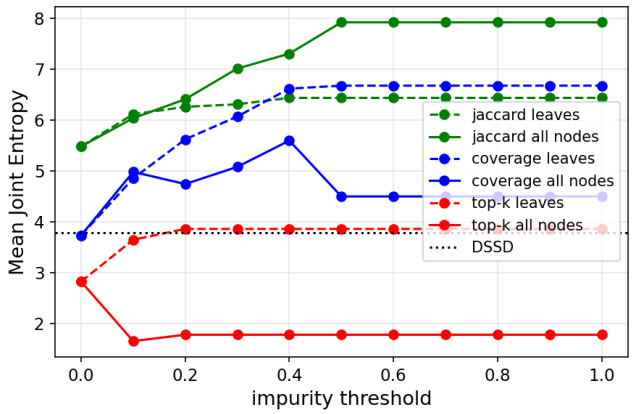



(b) Mushroom

Figure 23: Mean cover redundancy across impurity thresholds.



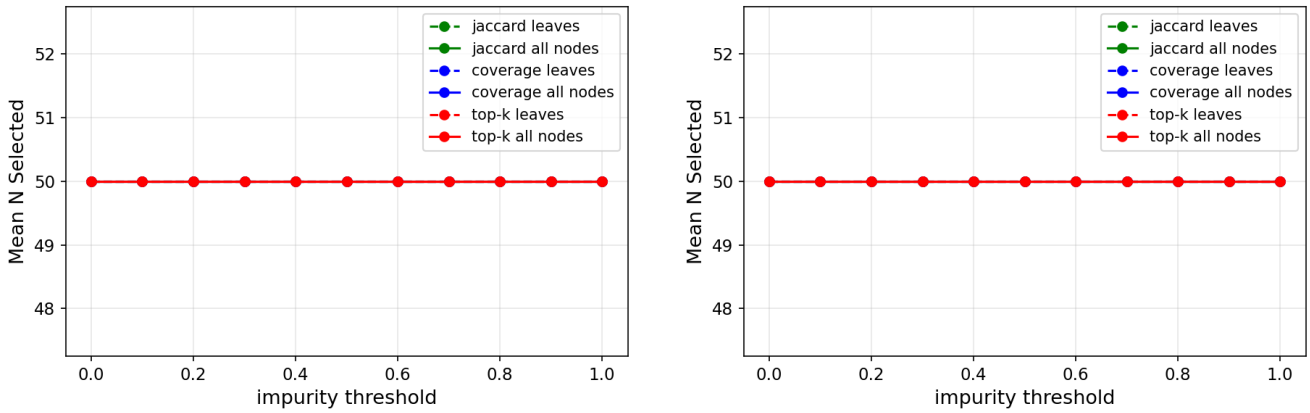

(a) Adult




(b) Mushroom

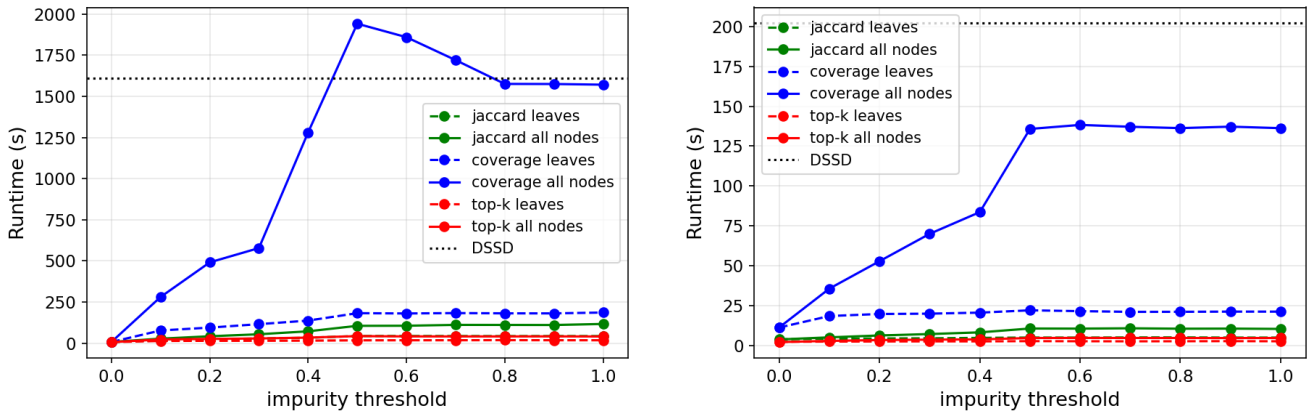
Figure 24: Mean joint entropy across impurity thresholds.

### Runtime & Amount Selected




(a) Adult (b) Mushroom

Figure 25: Mean candidates selected across impurity thresholds.



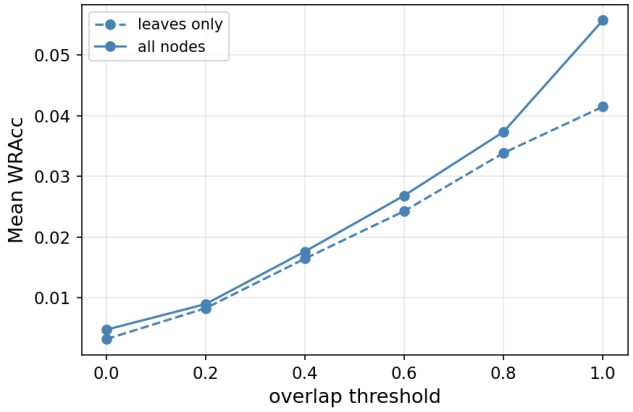

(a) Adult (b) Mushroom

Figure 26: Runtime across impurity thresholds.

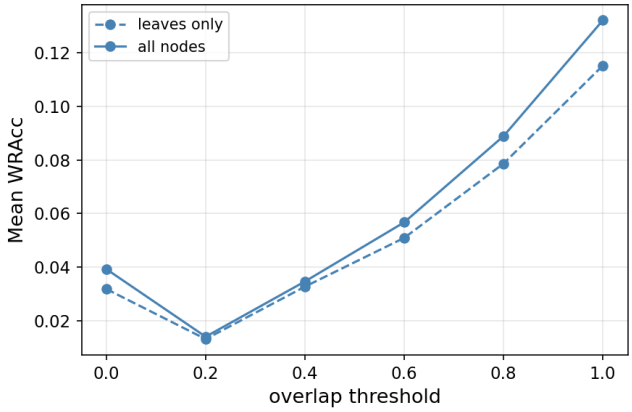
# C Parameter Sweep Plots

## Greedy Jaccard — Overlap Threshold

### Quality

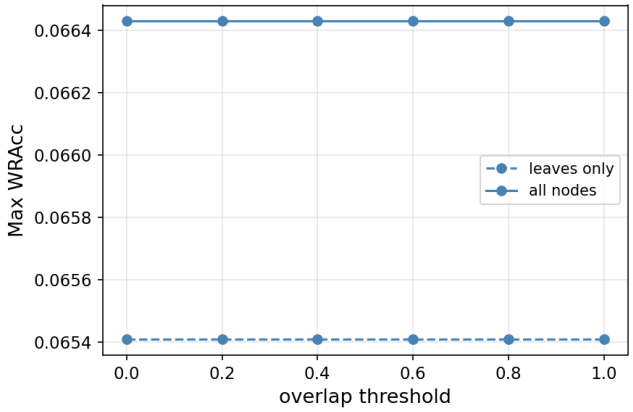



(a) Adult

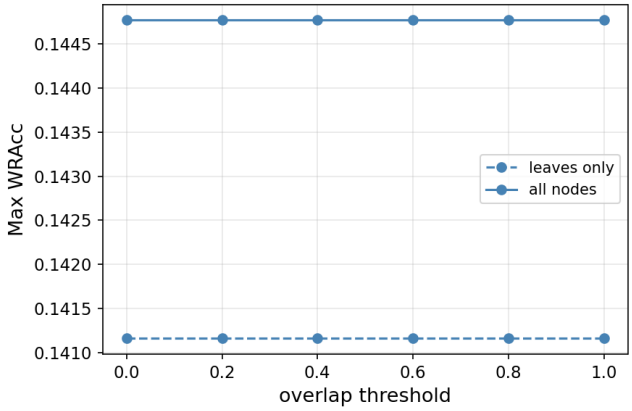



(b) Mushroom

Figure 27: Mean WRAcc across overlap thresholds.



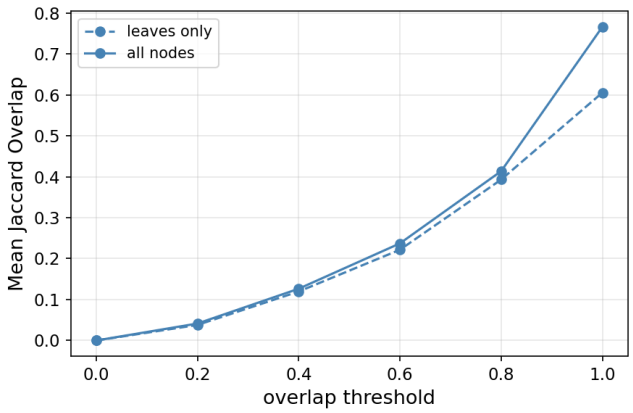

(a) Adult



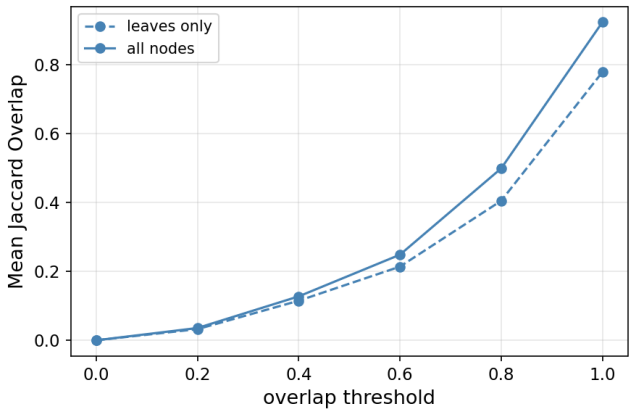

(b) Mushroom

Figure 28: Max WRAcc across overlap thresholds.

## Diversity

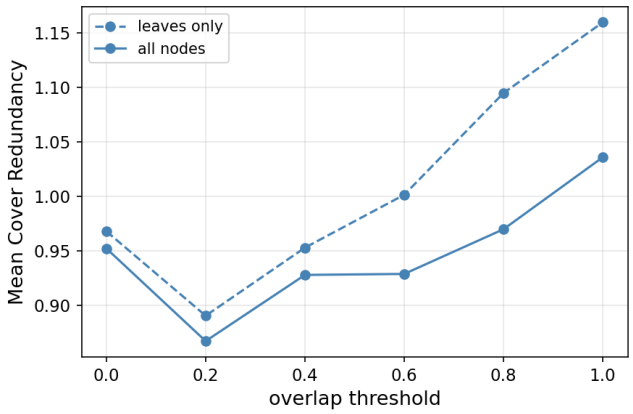



(a) Adult

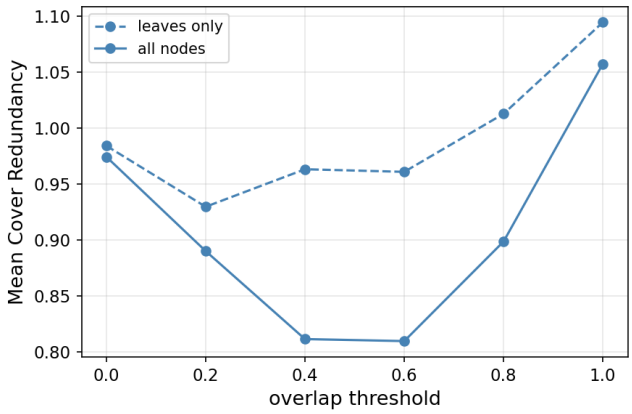



(b) Mushroom

Figure 29: Mean overlap across overlap thresholds.

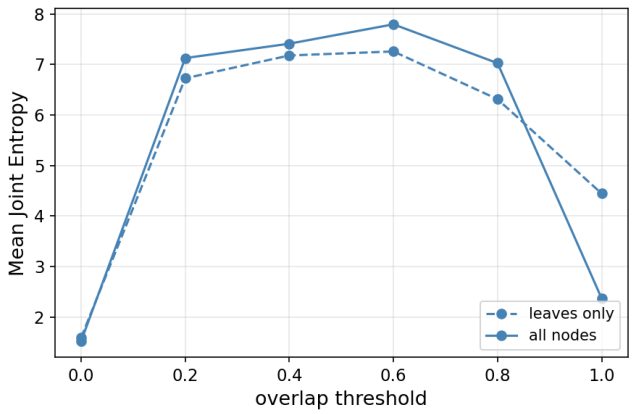



(a) Adult

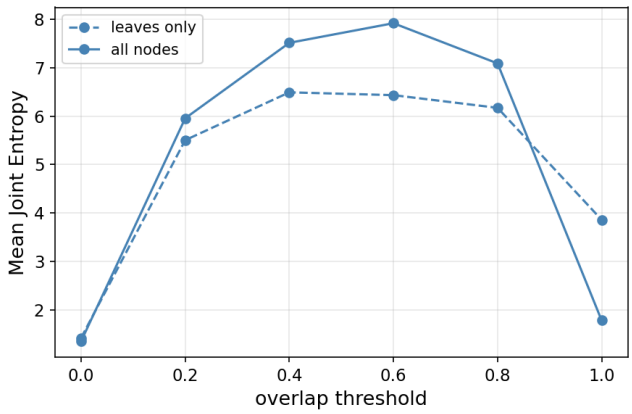



(b) Mushroom

Figure 30: Mean cover redundancy across overlap thresholds.



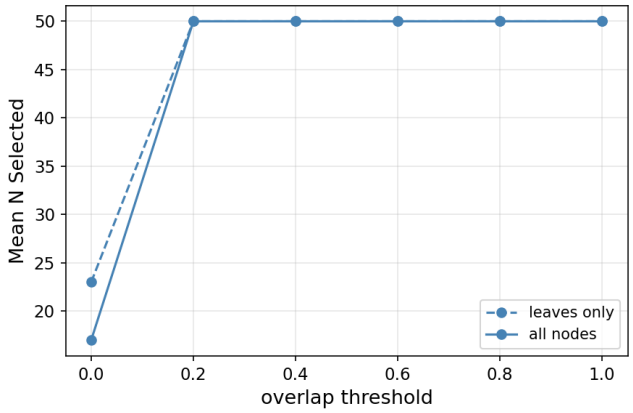

(a) Adult



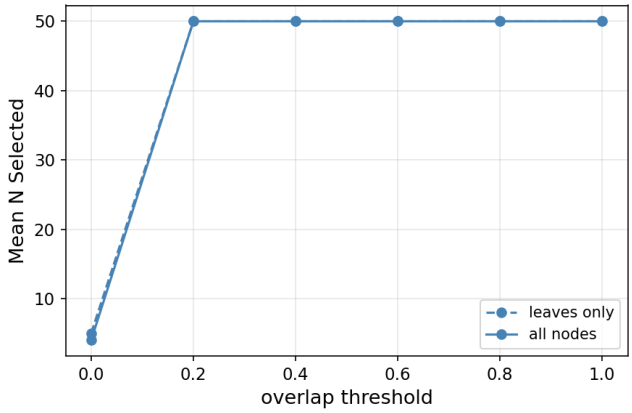

(b) Mushroom

Figure 31: Mean joint entropy across overlap thresholds.

### Runtime & Amount Selected

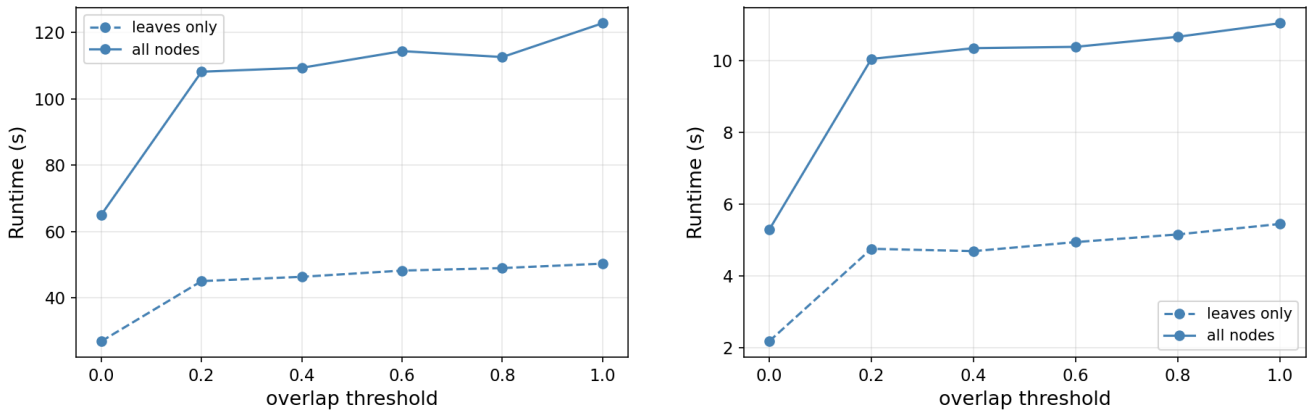



(a) Adult




(b) Mushroom

Figure 32: Mean candidates selected across overlap thresholds.

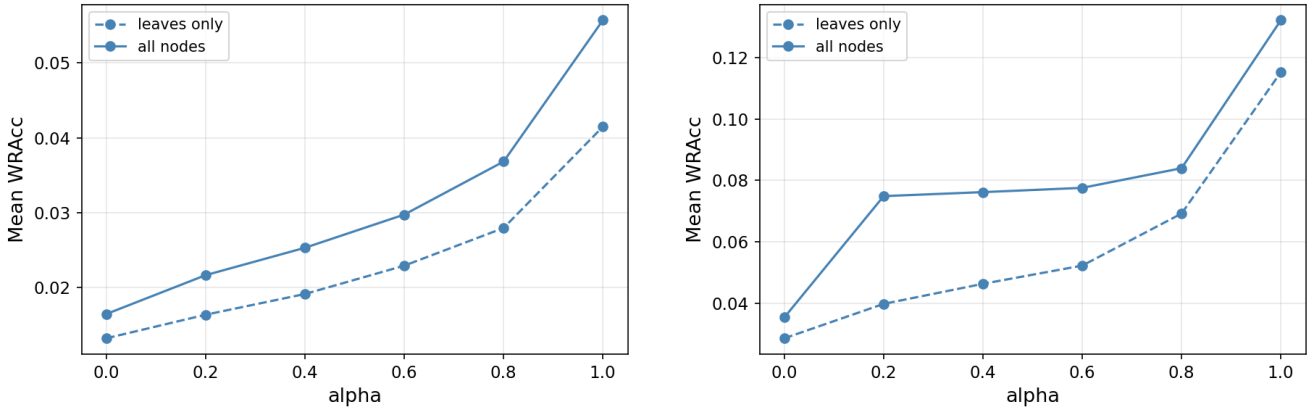



(a) Adult (b) Mushroom

Figure 33: Runtime across overlap thresholds.

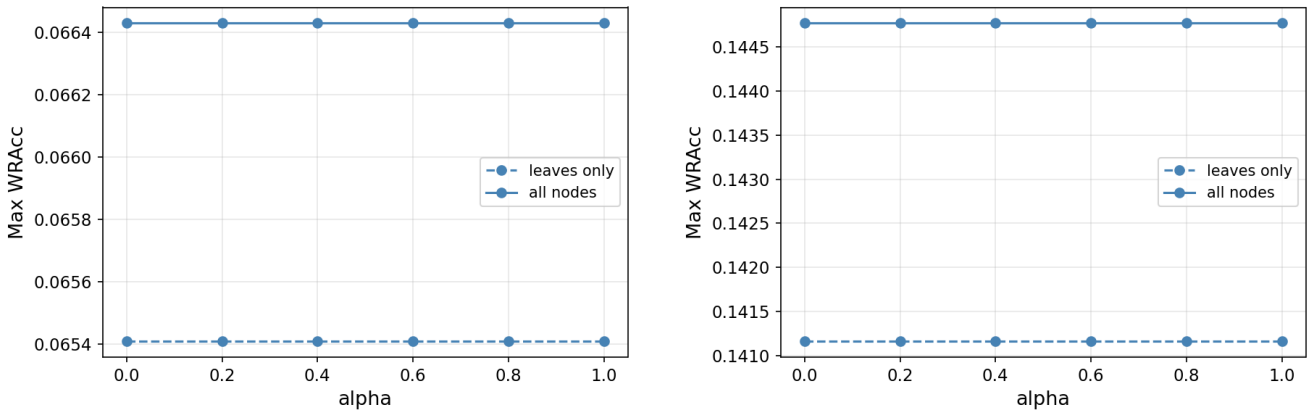
# Coverage-based — Alpha

## Quality




(a) Adult (b) Mushroom

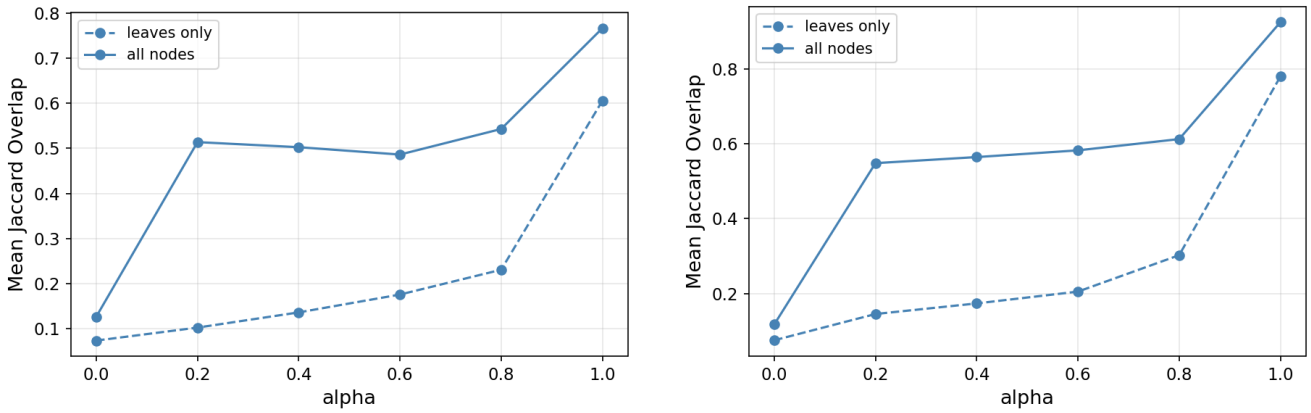
Figure 34: Mean WRAcc across alpha values.




(a) Adult (b) Mushroom

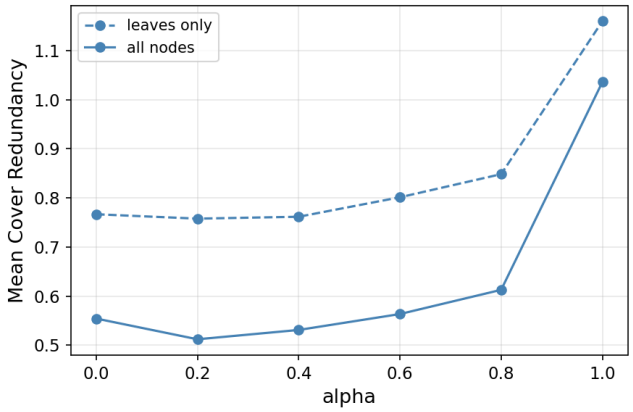
Figure 35: Max WRAcc across alpha values.

## Diversity

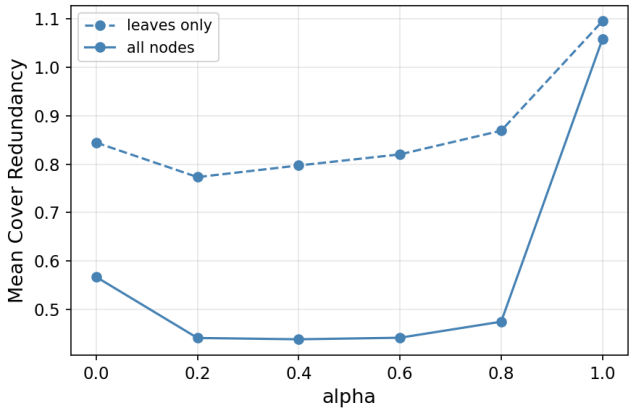



(a) Adult (b) Mushroom

Figure 36: Mean overlap across alpha values.

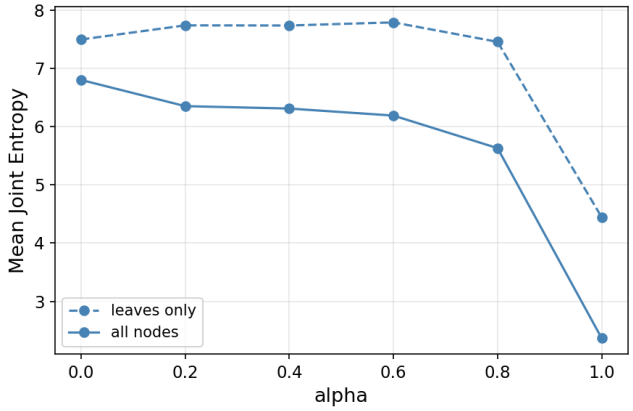



(a) Adult

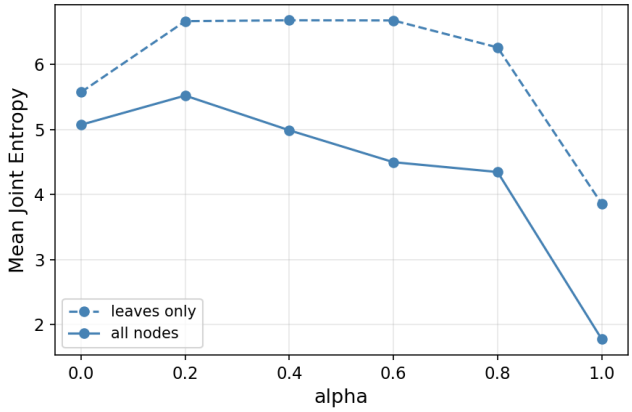



(b) Mushroom

Figure 37: Mean cover redundancy across alpha values.



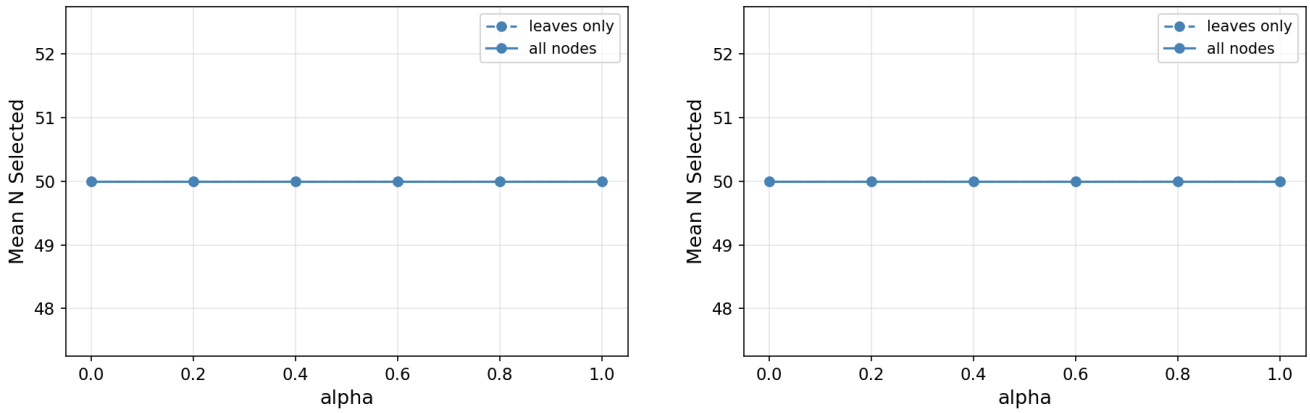

(a) Adult




(b) Mushroom

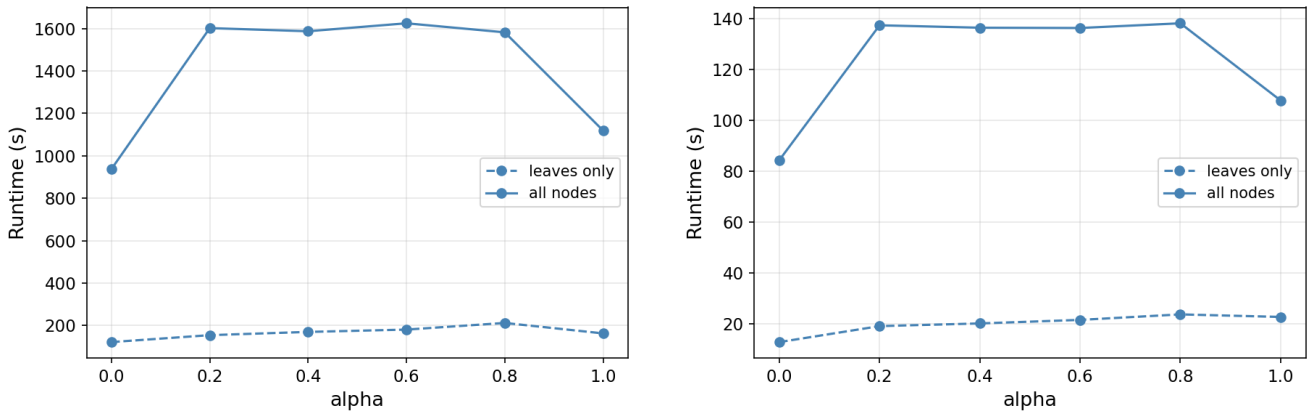
Figure 38: Mean joint entropy across alpha values.

### Runtime & Amount Selected




(a) Adult (b) Mushroom

Figure 39: Mean candidates selected across alpha values.




(a) Adult (b) Mushroom

Figure 40: Runtime across alpha values.