A Omitted Proofs

A.1 Proof of Lemma 3.1

Proof. Computing a single one dimensional projection takes time $O(nd)$ and computing a k-nearest neighbor graph can be done in time $O(nk)$ after sorting the points in $O(n \log n)$ time. Note that this *crucially* depends on the fact that we perform a one dimensional projection as nearest neighbors are determined by adjacent points on the real line. It is computationally expensive to compute such a graph in arbitrary dimensions larger than 1. Finding the sparsest prefix cut after sorting as is done in line 5 of Algorithm 1 takes linear time once the k-nearest neighbor graph has been constructed. Overall, the runtime of each of the T procedures is $O(nd + n \log n + nk)$. □

A.2 Proof of Lemma 3.2

Proof. The proof follows from the fact that if a cut does not respect the sorted ordering, then we can switch two points in opposite parts of the cut to reduce the number of edges across the cut. □

A.3 Proof of Lemma 3.4

Proof. Consider the computational cost of building `ClusterTree` as a tree. We claim that at every level of the tree, we do $O(nd + n \log n + nk)$ work. To verify this, we note that $cn \log(cn) + (1-c)n \log((1-c)n \le cn \log n + (1-c)n \log n = n \log n$. Then from Assumption 3.3, there are $O(\log n)$ levels of the tree, leading to the stated runtime. (One can also use the Akra-Bazzi method to arrive at the same conclusion, see Akra & Bazzi (1998) or Leighton (1996)). □

A.4 Proof of Lemma 3.6

Proof. Fix the dataset X. First note that for any fixed points $x, y \in X$, we have

$$\mathbb{E}\|P(x-y)\|^2 = \|x-y\|^2$$

since P is independent of X. Now for any $x \in X$, the average distance squared from Px to its k-nearest neighbors in PX is at most the average distance squared from Px to the points that were originally its k-nearest neighbors in X. This gives that $\alpha_{PX} \le \alpha_X$. Finally, note that that the expected value of the sum of all pairwise distances after the projection is the same as the sum of all pairwise distances from our observation above. This proves $\beta_{PX} = \beta_X$, as desired. □

A.5 Proof of Lemma 3.7

Proof. To prove Lemma 3.7, we will need the following auxiliary result.

Lemma A.1. *Suppose $x \sim \mathcal{N}(\mu_1, \Sigma_1)$ and $y \sim \mathcal{N}(\mu_2, \Sigma_2)$ Then*

$$\mathbb{E}[\|x-y\|^2] = \|\mu_1 - \mu_2\|^2 + \operatorname{tr}(\Sigma_1) + \operatorname{tr}(\Sigma_2).$$

Proof. Note that $x-y$ is distributed as $\mathcal{N}(\mu_1 - \mu_2, \Sigma_1 - \Sigma_2)$ and thus, $x - y \sim \mu_1 - \mu_2 + Az$ where $z \sim \mathcal{N}(0, I)$ and A satisfies $AA^T = \Sigma_1 + \Sigma_2$. Thus,

$$\|x-y\|^2 = \|\mu_1 - \mu_2\|^2 + 2(\mu_1 - \mu_2)^T Az + z^T A^T Az.$$

Since $\mathbb{E}[z] = 0$, we have

$$\mathbb{E}[\|x-y\|^2] = \|\mu_1 - \mu_2\|^2 + \mathbb{E}[z^T A^T Az]$$

and

$$\mathbb{E}[z^T A^T Az] = \sum_{i,j} \mathbb{E}[z_i z_j](A^T A)_{ij} = \sum_i (A^T A)_i = \operatorname{tr}(A^T A) = \operatorname{tr}(AA^T) = \operatorname{tr}(\Sigma_1 + \Sigma_2).$$

Putting together the above calculations gives the desired result. □

Note that we can upper bound α_X by the expected distance squared between two points drawn form the same component. This is because the distance to the k-th nearest neighbor from a fixed point will always be smaller than the distance to another point drawn from the same component (assuming our hypothesis that at least k points are drawn from each component). From Lemma A.1, it follows that $\alpha_X \le 2\max(\mathrm{tr}(\Sigma_1), \mathrm{tr}(\Sigma_2))$.

To lower bound β_X, note that since $\min(w, 1-w) = \Omega(1)$, the expected distance squared between two uniformly random points is at least asymptotically the expected distance squared between two points from separate components. Again using Lemma A.1, it follows that $\beta_X = \Omega(\|\mu_1 - \mu_2\|^2 + \mathrm{tr}(\Sigma_1 + \Sigma_2))$. □

A.6 Proof of Lemma 3.8

Proof. Lemma 3.7 tells us that

$$\frac{\alpha_X}{\beta_X} \lesssim \frac{\mathrm{tr}(\Sigma_1 + \Sigma_2)}{\|\mu_1 - \mu_2\|^2} \lesssim \frac{1}{c^2}$$

where we have used the fact that $d\,\lambda_1(\Sigma) \ge \mathrm{tr}(\Sigma)$ for a covariance matrix $\Sigma \in \mathbb{R}^{d\times d}$. □

A.7 Proof of Lemma B.1

We first need the following auxiliary results from Indyk & Naor (2007).

Lemma A.2. *Let $x \in S^{d-1}$ and let P be a random one-dimensional Gaussian projection. Then for all $t > 0$,*

$$\Pr(|\|Px\| - 1| \ge t) \le \exp(-t^2/8), \tag{1}$$

$$\Pr(\|Px\| \le 1/t) \le \frac{3}{t}. \tag{2}$$

We now proceed with the proof of Lemma B.1.

Proof. We first claim that the diameters of S and S' don't increase by a large factor after a random projection. Fix $x, y \in S$. By Eq. equation 1, the probability that $\|P(x-y)\|$ increases by a factor of t is at most $\exp(-t^2/8)$. Thus for a suitable constant c, we have that the probability $\|P(x-y)\|$ is larger by a $c(\sqrt{\log|S|} + \log(1/\epsilon))$ factor is at most $\epsilon/(3|S|^2)$. Union bounding across all pairs in S and using a similar argument for S' gives us that with probability at least $1 - 2\epsilon/3$, we have that $\mathrm{diam}(PS) \lesssim \sqrt{\log|S|}\,\mathrm{diam}(S)$ and $\mathrm{diam}(PS') \lesssim \sqrt{\log|S'|}\,\mathrm{diam}(S')$.

We now claim that the sets PS and PS' don't come 'too' close together. Indeed, take any point $x \in S$ and $y \in S'$. We have that $\|x - y\| \ge d(S, S')$. Thus by Eq. equation 2, the probability that $\|P(x-y)\|$ shrinks by a factor of $\Omega(1/\epsilon)$ is at most $O(\epsilon)$.

Altogether, we know that with probability at least $1 - \epsilon$, all three of the following events occur:

1. $\mathrm{diam}(PS) \lesssim \sqrt{\log|S|}\,\mathrm{diam}(S)$,
2. $\mathrm{diam}(PS') \lesssim \sqrt{\log|S'|}\,\mathrm{diam}(S')$,
3. $d(PS, PS') \ge \epsilon\, d(S, S') - \mathrm{diam}(PS) - \mathrm{diam}(PS')$.

Thus by our assumption that S and S' are r-apart for the value of r in the lemma statement, it follows that

$$\mathrm{diam}(PS) \lesssim \sqrt{\log|S|}\,\mathrm{diam}(S) - \mathrm{diam}(PS) - \mathrm{diam}(PS') \lesssim \epsilon\, d(S, S') \lesssim d(PS, PS')$$

and a similar statement holds for S', proving the lemma. □

A.8 Proof of Lemma B.2

Proof. The proof follows from Lemma B.1 as every point in PS will be closer to any other point in PS than any other point in PS'. A similar symmetric statement holds for PS'. Thus, any edges of the k-nearest neighbor graph starting from any point in PS must have its other vertex in PS as well. This implies that there is an empty cut between PS and PS', as desired. $\square$

Lemma A.3 (Follows from corollary 5 and 6 in Kushnir et al. (2019))**.** *Consider two c-separated Gaussian distributions in $\mathbb{R}^d$ with means μ_1, μ_2 and covariance matrices Σ_1 and Σ_2. Define $T(c', d)$ as in Lemma 3.9. Let $\gamma := 2d(c')^2\lambda_{\max}/\|\mu_1 - \mu_2\|^2$, where $\lambda_{\max}$ denotes the largest eigenvalue of the matrix $\Sigma_1 + \Sigma_2$. Then*

$$\lim_{d\to\infty} T(c', d) \le \frac{1}{2Q(\sqrt{\gamma})}.$$

B Hierarchical Clustering

In this section, we discuss the performance of `ClusterTree` for hierarchical clustering. Since our tree is designed to preserve the underlying cluster structure of the dataset, it is very natural to use it for clustering applications, such as hierarchical clustering. In hierarchical clustering, the goal is to design a tree over the input dataset which hopefully captures 'multi-scale' clustering relationships of the dataset.

To formalize this, we first define a natural hierarchical clustering model and then prove results which suggest that `ClusterTree` is naturally suited to recover such a clustering. Note that traditional algorithms for hierarchical clustering, such as computing the minimum spanning tree, require $\Omega(n^2)$ time, which is prohibitive for large datasets, whereas `ClusterTee` construction is nearly linear time.

Definition B.1 (Hierarchical Clustering Model)**.** *Let X be our dataset and P be a parameter. We assume there is a tree $\mathcal{T}$ over X such that the following is satisfied:*

- *The leaves of $\mathcal{T}$ are disjoint subsets of X of size at most some parameter P and together include all points of X,*
- *Level $i \ge 1$ of $\mathcal{T}$ is a union of two subsets in level $i-1$ of the tree where level 0 denotes the leaves. We assume that each subset at level $i-1$ contributes to exactly one subset in level i of the tree*
- *The largest level of the tree is the entire dataset X.*

Note that the above definition naturally describes a hierarchical clustering model over the dataset X where going up the tree indicates larger scale cluster structure over the dataset X. We further assume a separability criteria for our hierarchical clustering model.

Definition B.2. *Let $\mathrm{diam}(S)$ denote the diameter of the subset $S \subseteq X$ and $d(S, S')$ denote the distance between two subsets S, S':*

$$d(S, S') = \min_{x\in S, y\in S'} \|x - y\|.$$

*We say that subsets S, S' are r-**apart** if*

$$Cr\max(\mathrm{diam}(S), \mathrm{diam}S') \le d(S, S')$$

for some constant C.

If we assume the above definition applies to a pair of subsets of the tree $\mathcal{T}$ at any some fixed level, then intuitively we are requiring the two subsets constitute well separated clusters.

Given such an assumption, we want to argue that repeated application of Algorithm 1 can successfully recover the underlying tree $\mathcal{T}$. The intuition behind this is that if the subsets are projected, they will also be separated after a random projection with high probability. The following lemma shows that it is indeed the case.

Lemma B.1. *Suppose subsets S and S' satisfy Definition B.2 with $r \geq \sqrt{\log(|S| + |S'|)/\epsilon}$. Let PS and PS' respectively denote a random one-dimensional projection of the two subsets. Then with probability at least $1 - \epsilon$, we have*

$$c \max(\mathrm{diam}(PS), \mathrm{diam}(PS')) \leq d(PS, PS')$$

for some constant $c > 1$.

Lemma B.1 hints that with a sufficient separability assumption, the k-nearest neighbor graph in one-dimension will mostly have edges within a given cluster which leads to sparse cuts between different subsets. Thus, we can reasonably expect Algorithm 1 to separate the distinct clusters in $\mathcal{T}$ since it optimizes for sparse cuts. Formally, we can prove the following statement.

Lemma B.2. *Let S and S' be two r-apart subsets of dataset X for the value of r in Lemma B.1 and P be a random one-dimensional projection. If $\min(|S|, |S'|) \geq k$, then the k nearest neighbor graph of $PS \cup PS'$ will have a cut that separates PS and PS' with probability $1 - \epsilon$.*

Now consider the hierarchical clustering model given in Definition B.1 and let $\mathcal{T}$ denote the implicit tree over a dataset X. Consider a node of v of $\mathcal{T}$ and at some level i let S and S' denote the subsets at level $i-1$ that comprise v. If S and S' are r-apart for a sufficiently large value of r, then Lemma B.2 states that S and S' will have an empty cut between them after a random one-dimensional projection. If we further assume that each of the two pieces of the k-nearest neighbor graph is connected, Algorithm 1 will exactly split apart S and S', as intended in the tree $\mathcal{T}$.

Preserving Cluster Structure of the Dataset. We empirically validate the hypothesis that `ClusterTree` is superior to RP trees in finding partitions that preserve the underlying cluster structure of the dataset. We designed two related experiments to demonstrate this. For the first set of experiments, we measured the diameter of the leaves (weighted by the leaf sizes) of each class of trees as the parameter P increases. Again the intuition here is that if the diameter of the leaves are small, then it mostly contains points that are well-clustered together while conversely, if the diameter is large, then the tree has bucketed together points that belong to different clusters. Our results are shown in Figure 8. Indeed, we see that for most datasets `ClusterTree` results in leaves that are much more tightly clustered than RP Trees, which again demonstrates that `ClusterTree` is adaptive to the underlying cluster structure of the dataset.

We present additional experiments on how the weighted radius of leaves of various tree-based algorithms varies as a function size in Figure 9. See Section 4 for more details on experimental setting. Overall, we see that `ClusterTree` has a smaller radius as a function of cluster size for the Gaussian Mixture, Spam, RNA, and KDD Cup datasets. Note that for Spam, 2-means tree was too costly to run and for Gaussian Mixture, the 2-means tree and `ClusterTree` have very identical curves for weighted radius as a function of candidate size.

C Omitted Experimental Results

We give additional experimental results in this section.

Distance to k-th Nearest Neighbors. We created instances of `ClusterTree` and RP trees for all of our datasets where we set the leaf size, the parameter P in Algorithm 2, to be equal to 10% of n. We then computed the distance from a query to the k-th nearest neighbor among the candidates returned by a tree for various values of k and averaged this across all queries. The intuition here is that if a leaf node of a tree contains points from multiple *distinct* clusters, then there will be a substantial increase in this metric at some intermediate value of k. Indeed, this is what we observe in Figure 10. For example in the Gaussian Mixture, KDD Cup, and Spam datasets, there is a noticeable 'jump' in the plots for RP trees as it is 'mixing' multiple clusters in the leaf nodes while for `ClusterTree`, the relationship is much smoother.

To recap, the intuition here is that if a leaf node of a tree contains points from multiple *distinct* clusters, then there will be a substantial increase in this metric at some intermediate value of k. For example, suppose that the leaf of a node contains points from two distinct well-separated clusters and consider a query that

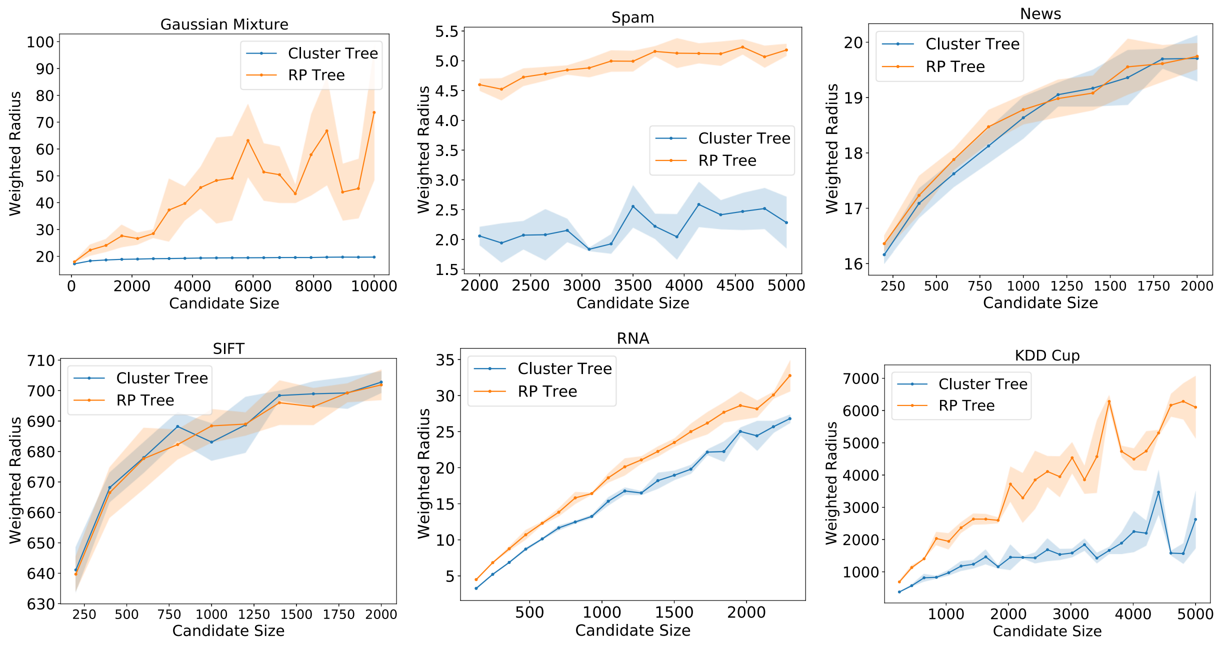

Figure 8: The leaves of `ClusterTree` have a smaller diameter than those of RP Trees.

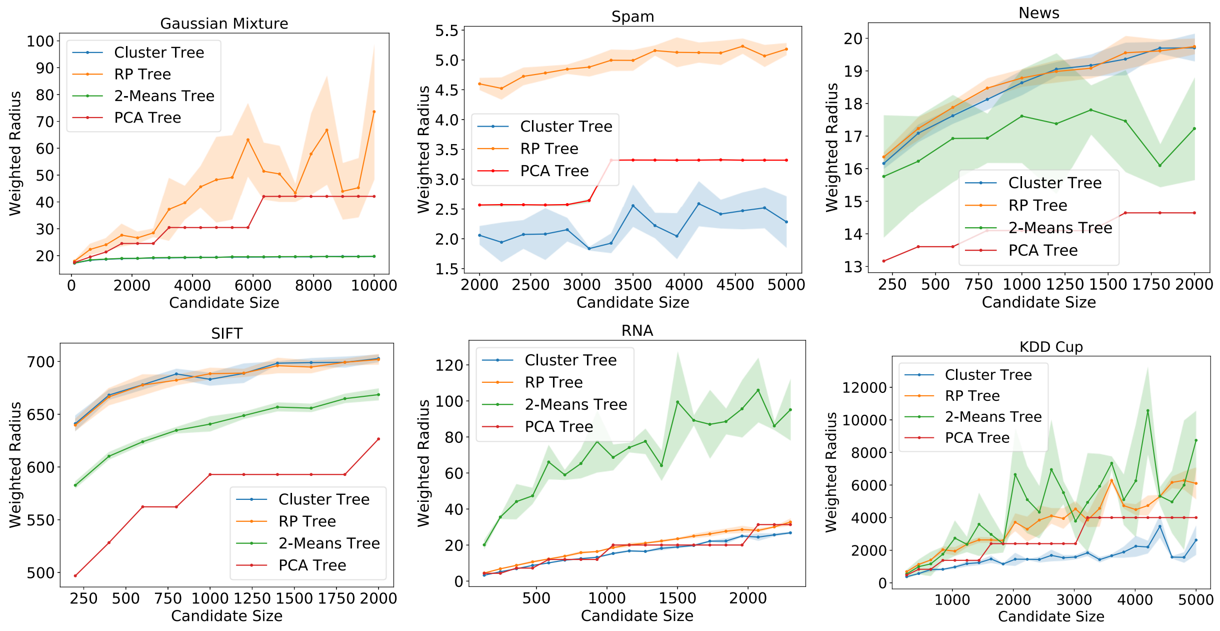

Figure 9: Expanded version of Figure 8 using all tree-based algorithms.

lands in this leaf but is closer to only one of the clusters. Then for a large enough value of k, the k-th nearest neighbor for this query would come from the far away cluster, leading to a significant increase in the distance to the k-th nearest neighbor in comparison to the $(k-1)$-th nearest neighbor. In contrast, if the leaf mostly contained points from one cluster, the distance would smoothly increase. To summarize, this is exactly the behaviour observed in Figure 10: in the Gaussian Mixture, KDD Cup, and Spam datasets, there is a noticeable 'jump' in the plots for RP trees as it is 'mixing' multiple clusters in the leaf nodes while for `ClusterTree`, the relationship is much smoother.

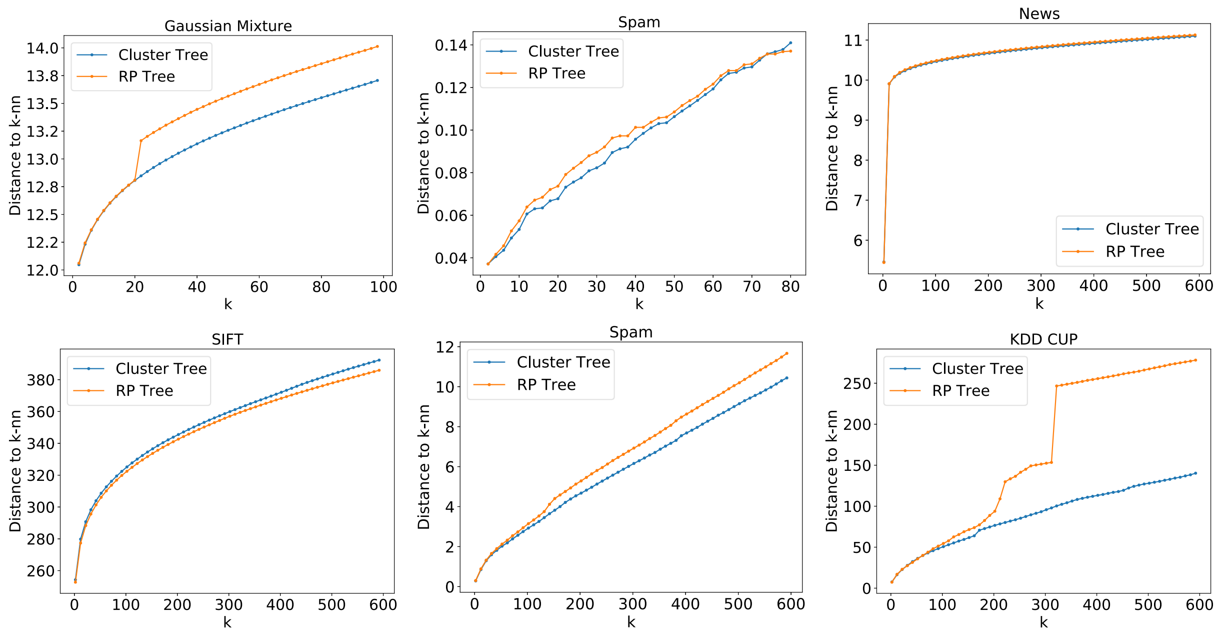

Figure 10: `ClusterTree` has a smoother trade-off curve for distance to the k-th neighbor as k increases.

Additional Parameter Selection Details. If there are multiple cuts that have conductance 0, i.e., multiple separated pieces in the k-nearest neighbor graph constructed in Algorithm 1, we pick the cut that is the most balanced. This is because any choice of the cuts would have been good with respect to preserving near neighbors so we should optimize for keeping the tree balanced.

Note that there have been recent works on improving RP trees using additional techniques such as sparse random projections (Sinha & Keivani, 2017), using auxiliary information when performing search over the tree (Keivani & Sinha, 2021), and other methods (Hyvönen et al., 2016). For simplicity, we did not use these techniques as they can be used identically for `ClusterTree` as for RP trees.

Finally, we highlight that both RP tree and `ClusterTree` are randomized algorithms. Therefore, they have the following additional benefit: in order to boost accuracy, we can initialize multiple instances of the data structure to create an ensemble of trees while keeping the overall number of candidates fixed. For example, instead of creating one tree with leaf size P, we can create two trees with leaf sizes $P/2$ each. In general, if we make a constant number of trees, this can be thought of as significantly boosting accuracy by increasing the amount of space used by only a constant factor. Note that 2-means trees and PCA trees are deterministic so they do not have this additional benefit. For simplicity however, we only compare single instantiation of each algorithm.

D Justifications for Assumption 3.3

We provide justification for Assumption 3.3 as its conditions hold true for one-dimensional datasets with very different structural properties: both uniform and clustered inputs.

- **Uniform Points**: Suppose the input to Algorithm 1 is a set of uniformly spaced points in one-dimensions. Then it is clear that the cut with the lowest conductance will split the dataset exactly in half due to symmetry.

- **Clustered Points**: Suppose the input consists of two well-separated clusters, with each cluster consisting of at least c-fraction of the total input size. Then the k-nearest neighbor graph for this input will be such that the edges of each cluster will be mostly to other points of the same cluster.

Thus, the cut separating the two clusters will be extremely sparse and hence have low conductance as well. See Figure 1 for an example.

Average Split Ratio. We now empirically validate Assumption 3.3. To do so, we compute `ClusterTree` for all of our datasets setting $P = 5\%$ of the size of the dataset in each case. The results across one run of Algorithm 2 are shown in Table 3. We observe that on average, each node of the tree splits the dataset into two approximately balanced parts.

Dataset	Avg. Split Ratio	Dataset	Avg. Split Ratio
KDD Cup	0.49(0.26)	Spam	0.50(0.27)
News	0.50(0.00)	SIFT	0.49(0.18)
RNA	0.45(0.29)	Gaussian Mixture	0.53(0.12)

Table 3: The average split ratio across all nodes in the tree with standard deviation in the parenthesis using 5% of the number of points as the parameter P (leaf size).

E Additional Datasets

We provide additional experiments for comparison between RP-trees and ClusterTree on a set of additional data sets: two clustering benchmark data sets from (Fränti & Sieranoja, 2018), 8776x13, 10126x15, Gas sensor data set 120kx128 (Fonollosa), stocks 94701x6 (Kaggle, 2023), mediamill 43907x120 (Repository), skyserver (Ahumada et al., 2020) 500Kx11.

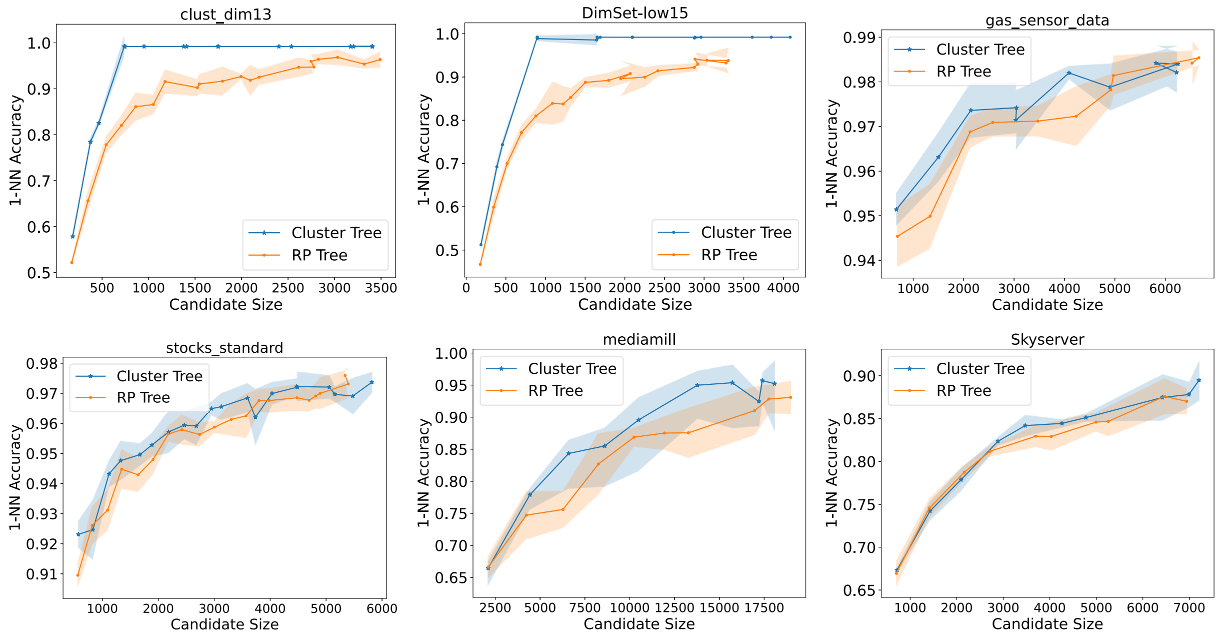

Figure 11: Candidate Size vs 1-NN Error for `ClusterTree` and RP tree.

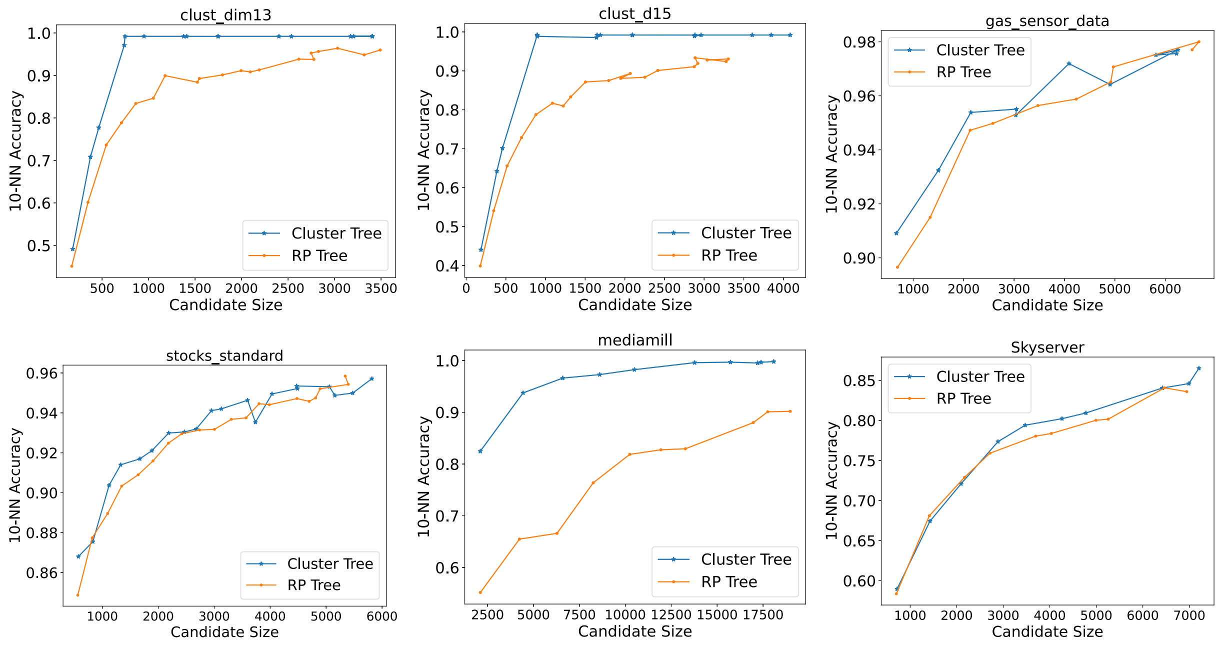

Figure 12: Candidate Size vs 10-NN Error for `ClusterTree` and RP tree.

F Additional Running-Time Accleration Analysis

Dataset	Accel. factor		CLTree bucket size		ClTree Acc. diff		PCATree Acc. diff		Comments
	min	**max**	min	max	min	max	min	max	
GMM	×**1.12**	×**1.35**	11180	17676	0.55	0.69	0.52	0.56	
SPAM	×**1.15**	×**2.7**	3×10^4	5×10^4	0.9980	0.9985	0.9968	0.9975	
News	**NA**	**NA**	NA	NA	1.0	1.0	0.9	0.95	no shared y values
RNA	**2.45**	**2.49**	379	92	0.99	0.99	0.12	0.17	
KDD Cup	× **0.53**	×**0.33**	1965	200	0.90	0.61	0.57	0.93	

Table 4: ClusterTree vs. PCATree query running-time acceleration min and max values for 10-NN.

Dataset	Accel. factor		CLTree bucket size		ClTree Acc. diff		KMsTree Acc. diff		Comments
	min	**max**	min	max	min	max	min	max	
GMM	**NA**	**NA**	NA	NA	NA	NA	NA	NA	no shared y and x values
SPAM	×**1.15**	×**2.7**	3×10^4	5×10^4	0.9980	0.9985	0.9968	0.9975	
News	**NA**	**NA**	NA	NA	1.0	1.0	0.44	0.15	no shared y values
RNA	**NA**	**NA**	NA	NA	0.98	0.96	0.15	0.05	no shared y values
KDD Cup	**NA**	**NA**	NA	NA	0.60	0.86	0.07	0.18	no shared y values

Table 5: ClusterTree vs. KMEANsTree query running-time acceleration min and max values for 10-NN.