CART-ELC: Oblique Decision Tree Induction via Exhaustive Search

Andrew D. Laack
University of Wisconsin-Superior

andrew@laack.co

Abstract

Oblique decision trees have attracted attention due to their potential for improved classification performance over traditional axis-aligned decision trees. However, methods that rely on exhaustive search to find oblique splits face computational challenges. As a result, they have not been widely explored. We introduce a novel algorithm, Classification and Regression Tree - Exhaustive Linear Combinations (CART-ELC), for inducing oblique decision trees that performs an exhaustive search on a restricted set of hyperplanes. We then investigate the algorithm's computational complexity and its predictive capabilities. Our results demonstrate that CART-ELC consistently achieves competitive performance on small datasets, often yielding statistically significant improvements in classification accuracy relative to existing decision tree induction algorithms, while frequently producing shallower, simpler, and thus more interpretable trees.

1 Introduction

Decision tree induction algorithms are among the most widely used algorithms for training interpretable machine learning models. This interpretability is one of the reasons they are often preferred over other models with better predictive capabilities (Mienye & Jere, 2024; Drucker & Cortes, 1995). While there has been much research on axis-aligned decision trees (Kotsiantis, 2011; Breiman et al., 1984; Mienye & Jere, 2024), research on oblique decision trees has been less extensive (Wickramarachchi et al., 2016; Murthy et al., 1994; Breiman et al., 1984).

Where traditional decision trees use axis-aligned splits to partition the feature space, oblique decision trees use linear combinations of features that are not restricted to axis-alignment. This allows oblique decision trees to be stronger learners with shallower trees while still being fairly interpretable (Murthy et al., 1994). However, this increased flexibility results in a larger search space as the number of hyperplanes that uniquely partition a dataset when considering the relative positions of samples with respect to the hyperplane is at most $2^m \cdot \binom{n}{m}$, where m is the dimensionality of the feature space and n is the number of samples (Murthy et al., 1994).

1.1 Related Works

1.1.1 CART

The Classification and Regression Tree (CART) algorithm (Breiman et al., 1984) induces axis-aligned decision trees by recursively partitioning the dataset. At each step, it evaluates a set of candidate axis-aligned splits, selects the best one, and partitions the dataset using this split. These steps are then performed recursively for each partition until a stopping criterion is met.

1.1.2 CART-LC

The Classification and Regression Tree - Linear Combinations (CART-LC) algorithm (Breiman et al., 1984) is a deterministic hill climbing algorithm for inducing oblique decision trees. The algorithm starts by median-centering the features and dividing them by their interquartile range. It then identifies the best axis-aligned

split and iteratively adjusts the coefficients of the corresponding hyperplane to improve the split. Once improvements fall below a threshold, the algorithm converts the refined split from the transformed feature space to the original feature space, and uses this split to partition the dataset. This process is then applied recursively for each partition until a stopping criterion is met.

1.1.3 OC1

The Oblique Classifier 1 (OC1) algorithm (Murthy et al., 1994) is a stochastic hill climbing algorithm for inducing oblique decision trees. It begins by initializing a random hyperplane and deterministically perturbs the hyperplane's coefficients to improve its splitting score. If no improvements can be made, a new hyperplane is created by adding random values to the coefficients in an attempt to escape the local optimum. The new hyperplane is then evaluated against the previous one. If the new score is better, the process repeats, starting with this new hyperplane as the initial hyperplane. If it is not better, another random hyperplane is initialized, and the entire procedure starts again. This process of applying perturbations and then attempting to escape local optima is applied z times, where z is a hyperparameter of the algorithm. After z iterations, the split with the best splitting score is then used to partition the dataset. The algorithm then recursively applies the above procedure to each partition until a stopping criterion is met.

1.1.4 HHCART

The Householder Classification and Regression Tree (HHCART) algorithm (Wickramarachchi et al., 2016) induces oblique decision trees by leveraging class-specific orientation. HHCART(D), a variant of the algorithm, computes the dominant eigenvector of the covariance matrix for each class. Each of these dominant eigenvectors is then individually used to perform a Householder transformation to align the corresponding class orientation with a coordinate axis. In the resulting feature space, the algorithm searches for the best axis-aligned split, which is then mapped back to the original feature space as an oblique split. The best oblique split found after evaluating each dominant eigenvector is used to partition the dataset. This process is then applied to each of the resulting partitions until a stopping criterion is met. Wickramarachchi et al. (2016) defined two variants of HHCART: HHCART(D), which only uses the dominant eigenvector for each class, and HHCART(A), which uses all eigenvectors for each class.

1.2 Limitations

A limitation of the CART algorithm is that it only considers axis-aligned splits. These splits can be suboptimal when the data is linearly separable, as described by Breiman et al. (1984). Algorithms like CART-LC and OC1 are not constrained to axis-aligned splits, but they often converge to local optima due to their hill climbing nature. While OC1 attempts to escape local optima by adding random values to the coefficients of the splitting hyperplane after its improvements have stagnated, this strategy does not guarantee escaping local optima. HHCART is not limited in either of these regards, but it does only test splits that are axis-aligned in specific transformed feature spaces. This subset of oblique splits can be overly limited in cases where it is computationally feasible to evaluate more oblique splits.

Additionally, while CART-LC, OC1, and HHCART can generate shallower trees relative to CART (Murthy et al., 1994), the resulting trees often use splits that are linear combinations of many features. This results in oblique trees that can be difficult to interpret, and more costly to perform predictions with.

2 CART-ELC

2.1 Description

We introduce CART-ELC to address the limitations described in subsection 1.2. CART-ELC has one hyperparameter, r, defining both the minimum number of samples each candidate hyperplane must pass through and the maximum number of non-zero coefficients the hyperplane's normal vector can have. We define m as the dimensionality of the feature space. With these definitions, we now discuss the two cases for the algorithm.

Case 1: r = m

In the general case for $r = m$, CART-ELC performs an exhaustive search of all unique hyperplanes that pass through at least r samples. This is done by evaluating each hyperplane according to a splitting criterion, tracking the best splitting hyperplane, and after all hyperplanes have been evaluated, the algorithm partitions the dataset using the best split. This process is then recursively applied to both partitions, stopping when a stopping criterion is met. The general case for $r = m$ is when each selection of r samples uniquely identifies a hyperplane. In such cases, this process guarantees the greedily optimal hyperplane passing through at least r samples will be used to partition the dataset at each splitting node. This guarantee does not apply in degenerate cases where samples don't uniquely identify a hyperplane, but when floating-point vectors are sampled from a continuous distribution, it is unlikely a selection of r samples is affinely dependent. In such cases, the normal vector for a hyperplane passing through the r samples is chosen according to the deterministic tie-breaking behavior of the underlying linear algebra solver's implementation.

Case 2: r < m

CART-ELC limits the search space to hyperplanes with at most r non-zero coefficients and one bias term. In the general case, the algorithm considers each unique hyperplane defined by selecting a combination of r features passing through at least r samples where the remaining $m - r$ features have coefficients of zero. Like the $r = m$ case, each candidate is then evaluated using a splitting criterion with the best split chosen to partition the dataset. This process is then recursively applied to both partitions until a stopping criterion is met. Additionally, like the $r = m$ case, the deterministic tie-breaking behavior of the underlying linear algebra solver's implementation is used to select a normal vector for a hyperplane that passes through the r samples when the selection of samples and features is affinely dependent.

Hyperplane Orientation

Every hyperplane has two unit normal vectors in $\mathbb{R}^n$ for $n \in \mathbb{N}$. Since we assign samples to the left partition when samples are on or above the hyperplane, the selection of a normal vector can change how samples that lie on the hyperplane contribute to the splitting score. To eliminate this nondeterminism, one may enforce a canonical sign rule. Given this, optimality guarantees for hyperplane splitting only hold under the deterministic orientation of the hyperplanes.
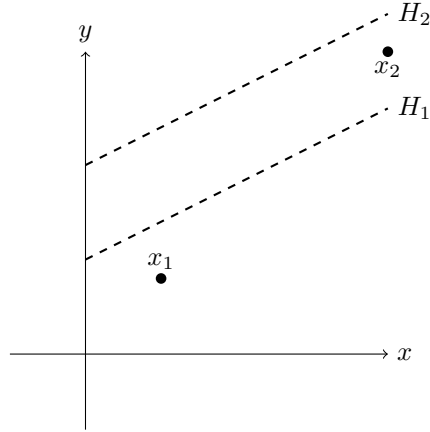

Figure 1: Hyperplanes resulting in unique partitionings of two samples.

In the general case for cases 1 and 2, the algorithm guarantees finding the best splitting hyperplane with at most r non-zero coefficients and a bias term that passes through at least r samples at each splitting node, but in neither case can it guarantee this is the greedily optimal oblique split. An illustration of this is shown in Figure 1. Based on the samples in Figure 1, if we define $r = m$, only one hyperplane will be evaluated: the hyperplane that passes through both samples and places them in the same partition. This hyperplane will result in the same partitioning as the hyperplane H_2. However, a hyperplane that results in

the same partitioning as hyperplane H_1 is better if x_1 and x_2 have different labels. Since CART-ELC does not evaluate a hyperplane that partitions the samples in this way, the algorithm can not guarantee greedily optimal oblique splits.

Additionally, there may be multiple selections of features and samples that describe the same hyperplane. While computing the splitting score for each hyperplane is not necessary, the algorithm will evaluate each. When floating-point vectors are sampled from a continuous distribution, it is infrequent different selections of samples and features will result in the same hyperplane.

The algorithm also defines two stopping criteria. The first stopping criterion is class homogeneity: if a partition contains samples that all belong to the same class, the algorithm will stop because subsequent splits would not change the model's predictions.

The second criterion is when a partition is empty. If no special handling is implemented, the algorithm will attempt to partition samples indefinitely in cases where one partition is empty and the other is not homogeneous. As such, the algorithm must verify both partitions are not empty after each partitioning.

Finally, the choice to not specify a splitting criterion, additional stopping criteria, specific hyperplane tie-breaking behavior, or pruning was made to ensure future research can use this algorithm without limiting its usefulness to specific domains.

2.2 Pseudocode

Algorithm 1: CART-ELC

1: **function** FIT(samples, labels, m, r)
2:     **if** homogeneous(labels) **then**
3:         **return** Node(majorityClass(labels))
4:     **end if**
5:     bestSplit, bestSplittingScore ← None, worstSplittingScore()
6:     **for** selectedSamples in combinations(samples, r) **do**
7:         **for** selectedFeatures in combinations(m, r) **do**
8:             vectorsToPassThrough ← featureSubset(selectedSamples, selectedFeatures)
9:             currentSplit ← findHyperplanePassingThrough(vectorsToPassThrough)
10:            currentSplittingScore ← evaluateSplit(currentSplit, samples)
11:            **if** isBetterThan(currentSplittingScore, bestSplittingScore) **then**
12:                bestSplittingScore, bestSplit ← currentSplittingScore, currentSplit
13:            **end if**
14:         **end for**
15:     **end for**
16:     left, right ← splitDataByBestSplit(samples, labels, bestSplit)
17:     **if** left is empty or right is empty **then**
18:         **return** Node(majorityClass(labels))
19:     **end if**
20:     leftSubtree ← fit(left.samples, left.labels, m, r)
21:     rightSubtree ← fit(right.samples, right.labels, m, r)
22:     tree ← Node(bestSplit)
23:     tree.left, tree.right ← leftSubtree, rightSubtree
24:     **return** tree
25: **end function**

2.3 Splitting Criteria

To evaluate the time complexity of our algorithm and perform an empirical analysis, we must define some splitting criteria. The three we implemented and will define are the twoing criterion (Breiman et al., 1984), the Gini criterion (Breiman et al., 1984), and information gain (Quinlan, 1986). The default splitting

criterion for our implementation of CART-ELC is the Gini criterion. This decision is based on our empirical analysis in subsection 3.4, which showed the twoing and Gini criteria often result in trees that yield the same classification accuracy, but the Gini criterion is computationally cheaper. Despite this, our empirical evaluation in subsection 3.3 uses the twoing criterion to maintain consistency with prior research.

2.3.1 Twoing Criterion

The twoing criterion is a splitting criterion that evaluates the quality of a split by considering the number of samples on each side of the splitting hyperplane and the difference in the distribution of classifications between the two partitions. Trees using the twoing criterion tend to be well-balanced because the splitting score is maximized when the number of samples in each partition is equal.

The formula for the twoing criterion defines the following variables:

- p_L: The probability a sample is in the left partition.
- p_R: The probability a sample is in the right partition.
- C: The set of all classes in the union of the left and right partitions.
- p_{Lj}: The probability a random sample from the left partition has a classification of j.
- p_{Rj}: The probability a random sample from the right partition has a classification of j.

$$T = \frac{p_L p_R}{4} \left(\sum_{j \in C} |p_{Lj} - p_{Rj}| \right)^2$$

2.3.2 Gini Criterion

The Gini criterion is the most popular splitting criterion for decision trees. Trees induced using the Gini criterion minimize class impurity, resulting in trees that may be less balanced than the twoing criterion. The splitting score is calculated by computing the Gini impurity for both partitions, weighting them based on the number of samples in each partition, and summing the results.

The formula for the Gini criterion defines the following variables:

- p_L: The probability a sample is in the left partition.
- p_R: The probability a sample is in the right partition.
- C: The set of all classes in the union of the left and right partitions.
- p_{Lj}: The probability a random sample from the left partition has a classification of j.
- p_{Rj}: The probability a random sample from the right partition has a classification of j.

$$G = p_L \left(1 - \sum_{j \in C} p_{Lj}^2 \right) + p_R \left(1 - \sum_{j \in C} p_{Rj}^2 \right)$$

2.3.3 Information Gain

Information gain is a criterion based on reducing class impurity. Unlike the Gini criterion, information gain calculates impurity using entropy.

The formula for information gain defines the following variables where $H(S)$ is the entropy of the set S (Shannon, 1948):

5

- S: The set of all samples before the split.

- S_L: The set of all samples in the left partition.

- S_R: The set of all samples in the right partition.

$$IG = H(S) - \left(\frac{|S_L|}{|S|} H(S_L) + \frac{|S_R|}{|S|} H(S_R) \right)$$

2.4   Time Complexity

2.4.1   Asymptotic Time Complexity Analysis

To evaluate the algorithm's asymptotic time complexity, we concentrate on the cost of finding and executing a split at a single decision node. Although a complete decision tree often contains multiple splitting nodes, the computational cost is most significant for the first split. Once the dataset is partitioned, subsequent splits operate on smaller subsets and thus require less computation. This is shown in Table 1 by considering the computational differences between the $n = 1000$ and $n = 500$ columns.

The homogeneity calculation, which requires at most n operations per label, is dominated by the complexity of the loop over each hyperplane. As such, it does not need to be considered for the final asymptotic time complexity. Next, we consider the evaluation of hyperplanes. Assume the feature matrix is $n \times m$, and

$$0 < r \leq n, m.$$

Hyperplane construction requires selecting r samples from n, which can be done in $\binom{n}{r}$ ways, and for each selection of samples, there are $\binom{m}{r}$ ways to choose r features. Therefore, the total number of hyperplanes to evaluate is

$$\binom{n}{r} \cdot \binom{m}{r}.$$

To derive the equation of the hyperplane passing through the r samples, an $r \times r$ matrix is constructed based on the r selected features and samples. Subsequently, its column-wise means are computed; these means are then subtracted from each row of the matrix, and the covariance matrix is calculated. Eigen decomposition is then performed on the resulting matrix to find the hyperplane's normal vector, and the hyperplane's bias term is determined by taking the dot product between the normal vector and the column-wise mean vector. These calculations are sufficient to derive the hyperplane's equation in normal form. The calculation of the covariance matrix and the eigen decomposition dominate the complexity of finding the equation for the hyperplane. Each is in the complexity class of $\Theta(r^3)$, assuming the use of a naïve matrix multiplication approach (Golub & Van Loan, 2013).

We then evaluate the split using our splitting criterion. Assume our splitting criterion is the Gini criterion. To compute our splitting score, we evaluate

$$G = p_L \left(1 - \sum_{j \in C} p_{Lj}^2 \right) + p_R \left(1 - \sum_{j \in C} p_{Rj}^2 \right).$$

The complexity of calculating the splitting score is dominated by determining which side of the hyperplane each sample lies on. This requires calculating the dot product between the sample and the hyperplane's normal vector, adding the bias term, and checking if the result is positive, negative, or zero. As such, the evaluation of one hyperplane is in the complexity class of $\Theta(rn)$, and since $|C| \leq n$, the number of classes does not impact the asymptotic time complexity of evaluating each hyperplane.

6

To split the dataset on the best split, we recompute the side of the hyperplane each sample lies on and add each sample to the corresponding list for samples that are either above or on the hyperplane or strictly below it. While this recomputation can be avoided by caching the results, it does not impact the overall asymptotic time complexity of the algorithm. The evaluation of all samples will take $\Theta(rn)$ time, and since we need to move each sample to the appropriate list, and each sample has m features, this will take $\Theta(nm)$ time. Thus, the asymptotic time complexity of splitting the dataset is in the complexity class of

$$\Theta(rn + nm).$$

Since splitting on the best split is only done once, its complexity is dominated by the complexity of evaluating all hyperplanes. As such, this final split does not impact the overall asymptotic time complexity.

Since there are $\binom{n}{r} \cdot \binom{m}{r}$ splits to evaluate, each split takes $\Theta(r^3)$ operations to compute the hyperplane, and $\Theta(rn)$ time to evaluate, we find the asymptotic time complexity of finding and executing a split at a single decision node is in the complexity class of

$$\Theta\left(\binom{n}{r} \cdot \binom{m}{r} \cdot r(r^2 + n)\right).$$

2.4.2 Operations for a Single Split

Table 1: Operations required for CART-ELC to identify and execute a split at a single decision node, assuming a multiplicative factor of one for the asymptotic time complexity, no additive constants, and $r = m$.

r	n					
	100	500	1000	5000	10000	20000
1	1.01e+04	2.50e+05	1.00e+06	2.50e+07	1.00e+08	4.00e+08
2	1.03e+06	1.26e+08	1.00e+09	1.25e+11	1.00e+12	8.00e+12
3	5.29e+07	3.16e+10	5.03e+11	3.13e+14	5.00e+15	8.00e+16
4	1.82e+09	5.31e+12	1.68e+14	5.22e+17	1.67e+19	5.34e+20
5	4.71e+10	6.70e+14	4.23e+16	6.53e+20	4.17e+22	2.67e+24
6	9.73e+11	6.77e+16	8.50e+18	6.54e+23	8.35e+25	1.07e+28
7	1.67e+13	5.71e+18	1.43e+21	5.46e+26	1.39e+29	3.56e+31
8	2.44e+14	4.13e+20	2.05e+23	3.90e+29	1.99e+32	1.02e+35
9	3.10e+15	2.62e+22	2.59e+25	2.44e+32	2.49e+35	2.55e+38
10	3.46e+16	1.47e+24	2.90e+27	1.36e+35	2.77e+38	5.66e+41

While the assumptions underlying Table 1 are idealized, it offers insights into the computational cost for CART-ELC to identify and split on a single decision node. These values indicate using CART-ELC on small datasets ($n \leq 1000$) when r values are small ($r \leq 3$) remains manageable, but as r and n values increase, the computational costs quickly grow reaching $3.13 \cdot 10^{14}$ operations at $r = 3$, $n = 5000$ for a single split.

While performing our evaluations shown in Table 2, we found r values of three or higher to be too costly in most cases. This was partially because the results shown in Table 1 assume $r = m$, but in Table 2 this was often not true, and thus there was an additional multiplicative factor of $\binom{m}{r}$.

3 Experiments and Results

In this section, we present our data collection methodology and provide a detailed description of the datasets used. We then compare CART-ELC with other decision tree induction algorithms, evaluate different splitting criteria, and discuss our findings.

3.1 Methodology

The results shown in Table 2 for OC1 and OC1-AP (Murthy et al., 1994), CART-LC and CART (Breiman et al., 1984), and C4.5 (Quinlan, 1993), where C4.5 and CART were collected using the IND 2.1 package (Buntine, 1993), were reported by Murthy et al. (1994). Alongside these, we present results from our implementations of HHCART (Wickramarachchi et al., 2016) and CART-ELC.[1] To ensure a fair comparison, we followed experimental procedures consistent with those of Murthy et al. (1994), differing only in our hyperparameter tuning strategy for CART-ELC and HHCART. The change to hyperparameter tuning was necessary due to our inclusion of a max depth hyperparameter for HHCART and CART-ELC, as well as the r hyperparameter for CART-ELC.

Similar to the experiments performed by Murthy et al. (1994), HHCART and CART-ELC used the twoing criterion as the splitting criterion, defined in subsection 2.3. Additionally, HHCART and CART-ELC did not apply any post-pruning or pre-pruning strategies.

For each experiment with CART-ELC we defined two variables, d and j, the largest max depth and r value to test respectively. We set $d = 5$ because increasing the max depth beyond five often led to identical tree structures or overfitting (see Appendix A). We also set $j = 2$ for all datasets except for the iris dataset where we set $j = 4$. These choices for j values were made due to computational constraints imposed by our algorithm's complexity and dataset sizes, outlined in subsection 2.4.

Shown below are the steps performed for each experiment.

1. Complete ten iterations of the following:

   (a) Randomly partition the dataset into five partitions of roughly equal size.
   (b) Perform a grid search over all integers $1 \leq i \leq j$ and $1 \leq k \leq d$ where k is defined as the current max depth hyperparameter, and i is defined as the current r hyperparameter. For each iteration:
      i. Build a tree using all partitions except for one as the training data.
      ii. Evaluate the accuracy of the tree on the unseen partition.
      iii. Repeat steps (i) and (ii) for each of the five partitions.
      iv. Calculate the mean accuracy and tree size across the five iterations where tree size is defined as the number of leaf nodes in the tree.

2. Calculate the mean and standard deviation of the tree sizes and accuracies based on the final results of each cross-validation (CV) (Kohavi, 1995) iteration for each hyperparameter pair.

3. Select the hyperparameter pair that offers the best balance between mean accuracy and tree size. The mean and standard deviation of the accuracy and tree size with the selected hyperparameter pair are the results of the experiment.

For our evaluation of both HHCART algorithms, we performed the same steps outlined above, but HHCART does not have an r hyperparameter, and thus our hyperparameter search was only performed for max depth values.

3.2 Datasets

3.2.1 Star/Galaxy Discrimination (Bright)

This dataset was derived from bright images collected by Odewahn et al. (1992) of stars and galaxies. The dataset is comprised of 2,462 samples, 14 features, and two classifications. The two classifications are star and galaxy. The 14 features were measurements defined by the original researchers that they deemed to be relevant for star/galaxy discrimination.

---

[1] Our implementations of HHCART and CART-ELC are available at `https://github.com/andrewlaack/cart-elc`.

3.2.2 Star/Galaxy Discrimination (Dim)

This dataset was derived from dim images collected by Odewahn et al. (1992) of stars and galaxies. The dataset is comprised of 4,192 samples, 14 features, and two classifications. The two classifications are star and galaxy. The 14 features were measurements defined by the original researchers that they deemed to be relevant for star/galaxy discrimination.

3.2.3 Breast Cancer Diagnosis

This dataset (Wolberg, 1990) is comprised of 699 samples with nine features and two classifications. The two classifications are malignant and benign. Similar to the evaluations performed by Murthy et al. (1994), we removed the 16 samples from this dataset that were missing the "Bare Nuclei" feature. Of the remaining 683 samples, 444 were labeled as benign and 239 were labeled as malignant.

3.2.4 Iris Classification

This dataset (Fisher, 1936) is comprised of 150 samples with four features and three classifications. Each of the three classifications specifies a different species of iris and contains 50 samples.

3.2.5 Boston Housing Cost

This dataset (Harrison & Rubinfeld, 1978) is comprised of 506 samples with 12 continuous features, one binary feature, and two classifications. While the original dataset contained a continuous label, we discretized the dataset by assigning the class of a given sample to be one if the continuous label's value was strictly less than 21,000 and two if it was greater than or equal to 21,000, as was done by Murthy et al. (1994). After this preprocessing, the dataset contained 260 samples with a classification of two and 246 samples with a classification of one. The dataset was also missing some features for select samples which we left as NaN for our evaluation of CART-ELC and performed mean imputation for HHCART as HHCART is unable to handle missing features.

3.2.6 Diabetes Diagnosis

This dataset (Smith et al., 1988) is comprised of 768 samples with eight features and two classifications. These two classifications represent testing positive and negative for diabetes. Of the 768 samples, 500 tested negative for diabetes, and 268 tested positive.

3.3 Empirical Algorithmic Comparison

Table 2: Accuracy and tree size comparison across decision tree induction algorithms.

Algorithm	Accuracy					
	S/G Bright	S/G Dim	Cancer	Iris	Housing	Diabetes
CART-ELC	**98.9 ± 0.2**	**95.2 ± 0.5**	96.3 ± 0.4	95.1 ± 0.8	83.5 ± 0.7	**74.5 ± 1.3**
HHCART(A)	98.3 ± 0.5	93.7 ± 0.8	**96.9 ± 0.3**	**95.5 ± 1.4**	**83.9 ± 0.8**	73.2 ± 1.2
HHCART(D)	98.1 ± 0.4	93.7 ± 0.9	**96.9 ± 0.3**	94.3 ± 1.5	82.2 ± 1.4	73.2 ± 1.2
OC1	**98.9 ± 0.2**	95.0 ± 0.3	96.2 ± 0.3	94.7 ± 3.1	82.4 ± 0.8	74.4 ± 1.0
OC1-AP	98.1 ± 0.2	94.0 ± 0.2	94.5 ± 0.5	92.7 ± 2.4	81.8 ± 1.0	73.8 ± 1.0
CART-LC	98.8 ± 0.2	92.8 ± 0.5	95.3 ± 0.6	93.5 ± 2.9	81.4 ± 1.2	73.7 ± 1.2
CART	98.5 ± 0.5	94.2 ± 0.7	95.0 ± 1.6	93.8 ± 3.7	82.1 ± 3.5	73.9 ± 3.4
C4.5	98.5 ± 0.5	93.3 ± 0.8	95.3 ± 2.0	95.1 ± 3.2	83.2 ± 3.1	71.4 ± 3.3

Algorithm	Tree Size					
	S/G Bright	S/G Dim	Cancer	Iris	Housing	Diabetes
CART-ELC	**3.7 ± 0.2**	**9.8 ± 4.2**	**2.0 ± 0.0**	4.8 ± 0.1	**4.0 ± 0.0**	**4.0 ± 0.0**
HHCART(A)	6.1 ± 0.3	14.6 ± 4.8	**2.0 ± 0.0**	**3.1 ± 0.1**	7.8 ± 0.2	**4.0 ± 0.0**
HHCART(D)	6.3 ± 0.4	14.9 ± 5.0	**2.0 ± 0.0**	4.7 ± 0.1	23.3 ± 0.8	**4.0 ± 0.0**
OC1	4.3 ± 1.0	13.0 ± 8.7	2.8 ± 0.9	**3.1 ± 0.2**	6.9 ± 3.2	5.4 ± 3.8
OC1-AP	6.9 ± 2.4	29.3 ± 8.8	6.4 ± 1.7	3.2 ± 0.3	8.6 ± 4.5	11.4 ± 7.5
CART-LC	3.9 ± 1.3	24.2 ± 8.7	3.5 ± 0.9	3.2 ± 0.3	5.8 ± 3.2	8.0 ± 5.2
CART	13.9 ± 5.7	30.4 ± 10.0	11.5 ± 7.2	4.3 ± 1.6	15.1 ± 10	11.5 ± 9.1
C4.5	14.3 ± 2.2	77.9 ± 7.4	9.8 ± 2.2	4.6 ± 0.8	28.2 ± 3.3	56.3 ± 7.9

The results presented in Table 2 demonstrate that CART-ELC, HHCART(A), HHCART(D), and OC1 obtained consistently strong accuracies across the datasets evaluated when compared with other decision tree induction algorithms. Additionally, of these top performers, CART-ELC had the smallest trees across most datasets, falling behind on the iris dataset where a higher max depth value was selected due to minor variances in accuracies across max depth and r values. Finally, we note the standard deviations in accuracy for CART-ELC, on average, were lower than those of OC1 and the variants of HHCART.

Table 3: P-values for accuracy comparisons between various models and CART-ELC. P-values strictly less than 0.05 are bolded. P-values reported as 0.000 are not zero, but are smaller than the reporting precision.

Algorithm	S/G Bright	S/G Dim	Cancer	Iris	Housing	Diabetes
HHCART(A)	**0.004**	**0.000**	**0.001**	0.446	0.250	**0.032**
HHCART(D)	**0.000**	**0.000**	**0.001**	0.159	**0.021**	**0.032**
OC1	1.000	0.296	0.536	0.701	**0.004**	0.849
OC1-AP	**0.000**	**0.000**	**0.000**	**0.012**	**0.000**	0.195
CART-LC	0.278	**0.000**	**0.000**	0.122	**0.000**	0.170
CART	**0.037**	**0.002**	**0.032**	0.303	0.244	0.612
C4.5	**0.037**	**0.000**	0.153	1.000	0.771	**0.017**

The results in Table 3, obtained by performing Welch's t-tests, compare the accuracies of each algorithm against CART-ELC. These results support the finding that CART-ELC, HHCART, and OC1 are the best-performing decision tree induction algorithms on the datasets evaluated. Additionally, CART-ELC had statistically significant performance improvements on both star/galaxy discrimination datasets and the diabetes dataset relative to both HHCART variants and only had statistically significant performance improvements when compared to OC1 on the housing dataset. The final result of interest is that both variants of HHCART performed better than CART-ELC on the cancer dataset by a statistically significant amount.

Table 4: Cohen's d effect size for accuracies between various models and CART-ELC. Comparisons with $p < 0.05$ are bolded.

Algorithm	S/G Bright	S/G Dim	Cancer	Iris	Housing	Diabetes
HHCART(A)	**1.576**	**2.249**	**-1.697**	-0.351	-0.532	**1.039**
HHCART(D)	**2.530**	**2.060**	**-1.697**	0.666	**1.175**	**1.039**
OC1	0.000	0.485	0.283	0.177	**1.463**	0.086
OC1-AP	**4.000**	**3.151**	**3.976**	**1.342**	**1.970**	0.604
CART-LC	0.500	**4.800**	**1.961**	0.752	**2.138**	0.639
CART	**1.050**	**1.644**	**1.115**	0.486	0.555	0.233
C4.5	**1.050**	**2.848**	0.693	0.000	0.133	**1.236**

The results in Table 4 show the effect sizes of the difference between accuracies with respect to CART-ELC. Negative values indicate the algorithm outperformed CART-ELC on the dataset, and positive values indicate CART-ELC outperformed the algorithm on the dataset. These results show the magnitude of CART-ELC's outperformance of HHCART on the star/galaxy datasets is greater than the magnitude of HHCART's outperformance of CART-ELC on the cancer dataset.

HHCART's strong performance relative to CART-ELC on the cancer dataset is likely due to the dataset being both linearly separable and having a sizable gap between the two classes. HHCART typically uses splits that pass through one sample in the transformed feature space, and thus it can find splits with a larger separation on both sides with respect to the classes on each side. This contrasts CART-ELC where each hyperplane generally passes through multiple samples and will thus generalize worse on some datasets because the separation on both sides of the hyperplane will be smaller.

Despite this, a limitation of HHCART that makes CART-ELC perform better on certain datasets is its inability to handle outliers well. Outliers nominally impact CART-ELC as the splitting criterion is minimally impacted by them, but HHCART uses a change of basis derived from the eigen decomposition of the covariance matrix for each class, and these resulting eigenvectors can be greatly impacted by outliers. This results in HHCART searching for axis-aligned splits in suboptimal feature spaces. Similarly, when there are clusters of samples with the same classification throughout the feature space, this can also pose issues for HHCART as the covariance matrices will be impacted by the disparate groups more than CART-ELC will be.

3.4 Empirical Criteria Comparison

Table 5: Accuracy and tree size comparison across splitting criteria.

Splitting Criterion	Accuracy					
	S/G Bright	S/G Dim	Cancer	Iris	Housing	Diabetes
Twoing Criterion	**98.9 ± 0.2**	**95.2 ± 0.5**	**96.3 ± 0.4**	95.1 ± 0.8	**83.5 ± 0.7**	**74.5 ± 1.3**
Gini Criterion	**98.9 ± 0.3**	95.1 ± 0.5	**96.3 ± 0.4**	95.1 ± 0.9	**83.5 ± 0.7**	**74.5 ± 1.3**
Information Gain	**98.9 ± 0.2**	94.9 ± 0.5	96.0 ± 0.4	**95.3 ± 0.8**	81.8 ± 1.0	74.0 ± 1.1

Splitting Criterion	Tree Size					
	S/G Bright	S/G Dim	Cancer	Iris	Housing	Diabetes
Twoing Criterion	3.7 ± 0.2	**9.8 ± 4.2**	**2.0 ± 0.0**	4.8 ± 0.1	4.0 ± 0.0	**4.0 ± 0.0**
Gini Criterion	**3.6 ± 0.1**	12.4 ± 8.3	**2.0 ± 0.0**	3.0 ± 0.1	4.0 ± 0.0	**4.0 ± 0.0**
Information Gain	3.8 ± 0.2	9.9 ± 4.5	**2.0 ± 0.0**	4.8 ± 0.1	**2.0 ± 0.0**	**4.0 ± 0.0**

Our experimental results across splitting criteria, shown in Table 5, indicate the twoing criterion and Gini criterion are both strong splitting criteria, and information gain only falls behind nominally. Additionally,

on the dataset we evaluated that contained more than two classifications, information gain outperformed the other two criteria.

4 Conclusions and Future Work

CART-ELC achieves competitive accuracies across a variety of small datasets when compared with other decision tree algorithms despite only searching a subset of all unique hyperplanes. Additionally, while other oblique decision trees often use splits that are linear combinations of many features, CART-ELC's r hyperparameter limits the number of features in each linear combination, which can result in simpler decision boundaries. This, along with CART-ELC's ability to generate shallower trees, often results in trees that are both more interpretable and efficient when making predictions.

Despite this, the computational complexity of CART-ELC remains prohibitively high for large datasets. As such, future research should focus on strategies to mitigate these computational costs, such as developing methods to select a smaller subset of candidate hyperplanes or using CART-ELC with ensemble methods. Adapting CART-ELC for use in random forests (Breiman, 2001) and stochastic gradient boosting (Friedman, 2002) may be a promising direction, as these approaches could reduce computational costs by training on subsets of samples or by introducing stochastic elements into the training process. Finally, while our evaluation included six datasets, further experimentation on a wider variety of datasets and more extensive CV may be beneficial for assessing the algorithm's overall efficacy.

References

L. Breiman. Random forests. *Machine Learning*, 45(1):5–32, Oct 2001. ISSN 1573-0565. doi: 10.1023/A: 1010933404324.

L. Breiman, J. Friedman, R. A. Olshen, and C. J. Stone. *Classification and Regression Trees*. Chapman and Hall/CRC, 1st edition, 1984. doi: 10.1201/9781315139470.

W. L. Buntine. Tree classification software. In *Technology 2002: The Third National Technology Transfer Conference and Exposition, Volume 1*, 1993.

H. Drucker and C. Cortes. Boosting decision trees. In *Advances in Neural Information Processing Systems*, volume 8. MIT Press, 1995.

R. A. Fisher. Iris. UCI Machine Learning Repository, 1936. doi: 10.24432/C56C76.

J. H. Friedman. Stochastic gradient boosting. *Computational Statistics & Data Analysis*, 38(4):367–378, 2002. ISSN 0167-9473. doi: 10.1016/S0167-9473(01)00065-2.

G.H. Golub and C.F. Van Loan. *Matrix Computations*. Johns Hopkins Studies in the Mathematical Sciences. Johns Hopkins University Press, 2013. ISBN 9781421407944.

D. Harrison and D. L. Rubinfeld. Hedonic housing prices and the demand for clean air. *Journal of Environmental Economics and Management*, 5(1):81–102, 1978. ISSN 0095-0696. doi: 10.1016/0095-0696(78) 90006-2.

R. Kohavi. A study of cross-validation and bootstrap for accuracy estimation and model selection. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, pp. 1137–1143, Montreal, Canada, 1995.

S. B. Kotsiantis. Decision trees: a recent overview. *Artificial Intelligence Review*, 39(4):261–283, 2011. doi: 10.1007/s10462-011-9272-4.

I. D. Mienye and N. Jere. A survey of decision trees: Concepts, algorithms, and applications. *IEEE Access*, 12:86716–86727, 2024. doi: 10.1109/ACCESS.2024.3416838.

S. K. Murthy, S. Kasif, and S. L. Salzberg. A system for induction of oblique decision trees. *Journal of Artificial Intelligence Research*, 2:1–32, 1994. doi: 10.1613/jair.63.

S. C. Odewahn, E. B. Stockwell, R. L. Pennington, R. M. Humphreys, and W. A. Zumach. Automated star/galaxy discrimination with neural networks. *The Astronomical Journal*, 103:318, 1992. doi: 10.1086/116063.

J. R. Quinlan. Induction of decision trees. *Machine Learning*, 1(1):81–106, 1986. doi: 10.1007/BF00116251.

J. R. Quinlan. *C4.5: Programs for Machine Learning.* Morgan Kaufmann series in machine learning. Elsevier Science, 1993. ISBN 9781558602380.

C. E. Shannon. A mathematical theory of communication. *The Bell System Technical Journal*, 27(3): 379–423, 1948. doi: 10.1002/j.1538-7305.1948.tb01338.x.

J. W. Smith, J. E. Everhart, W. C. Dickson, W. C. Knowler, and R. S. Johannes. Using the adap learning algorithm to forecast the onset of diabetes mellitus. In *Proceedings of the Annual Symposium on Computer Application in Medical Care*, pp. 261–265, 1988.

D. C. Wickramarachchi, B. L. Robertson, M. Reale, C. J. Price, and J. Brown. HHCART: An oblique decision tree. *Computational Statistics & Data Analysis*, 96:12–23, 2016. ISSN 0167-9473. doi: 10.1016/j.csda.2015.11.006.

W. Wolberg. Breast Cancer Wisconsin (Original). UCI Machine Learning Repository, 1990. doi: 10.24432/C5HP4Z.

A Hyperparameter Tuning Results for CART-ELC

Figures 2–4 show the accuracies and tree sizes for CART-ELC across each splitting criterion, dataset, r value, and max depth. These results show higher r values often result in higher accuracies at lower max depths but begin overfitting at lower max depths. We also see the accuracies and tree sizes across splitting criteria are quite consistent, often varying in accuracy but rarely in trends. Finally, on the S/G Dim dataset, accuracies appear to be plateauing around a max depth of five for $r = 2$ while on the other datasets, max depths of five either result in models overfitting or accuracies stagnating for $r \geq 2$.
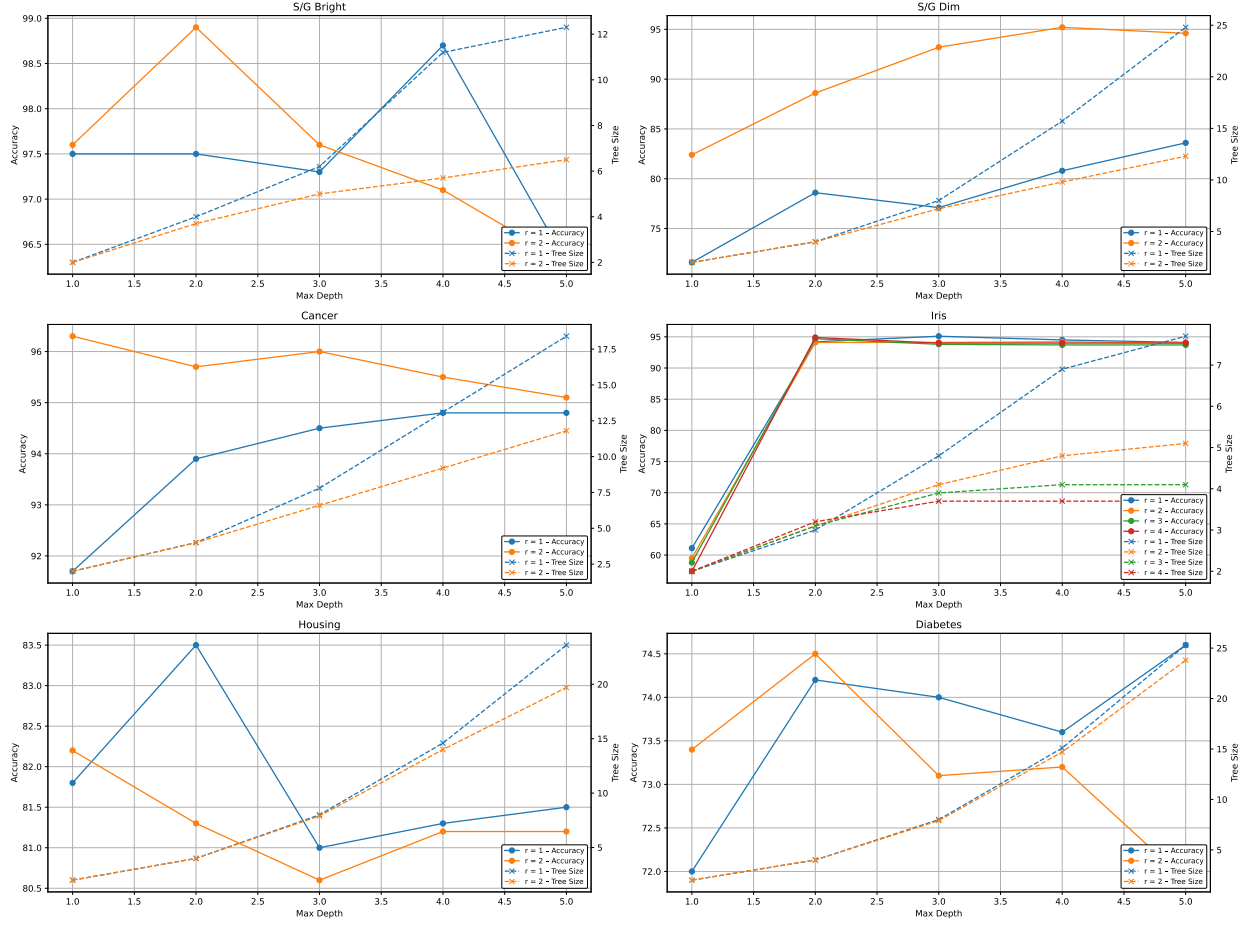

Figure 2: Max depth and r value results for CART-ELC when using the twoing criterion as the splitting criterion.
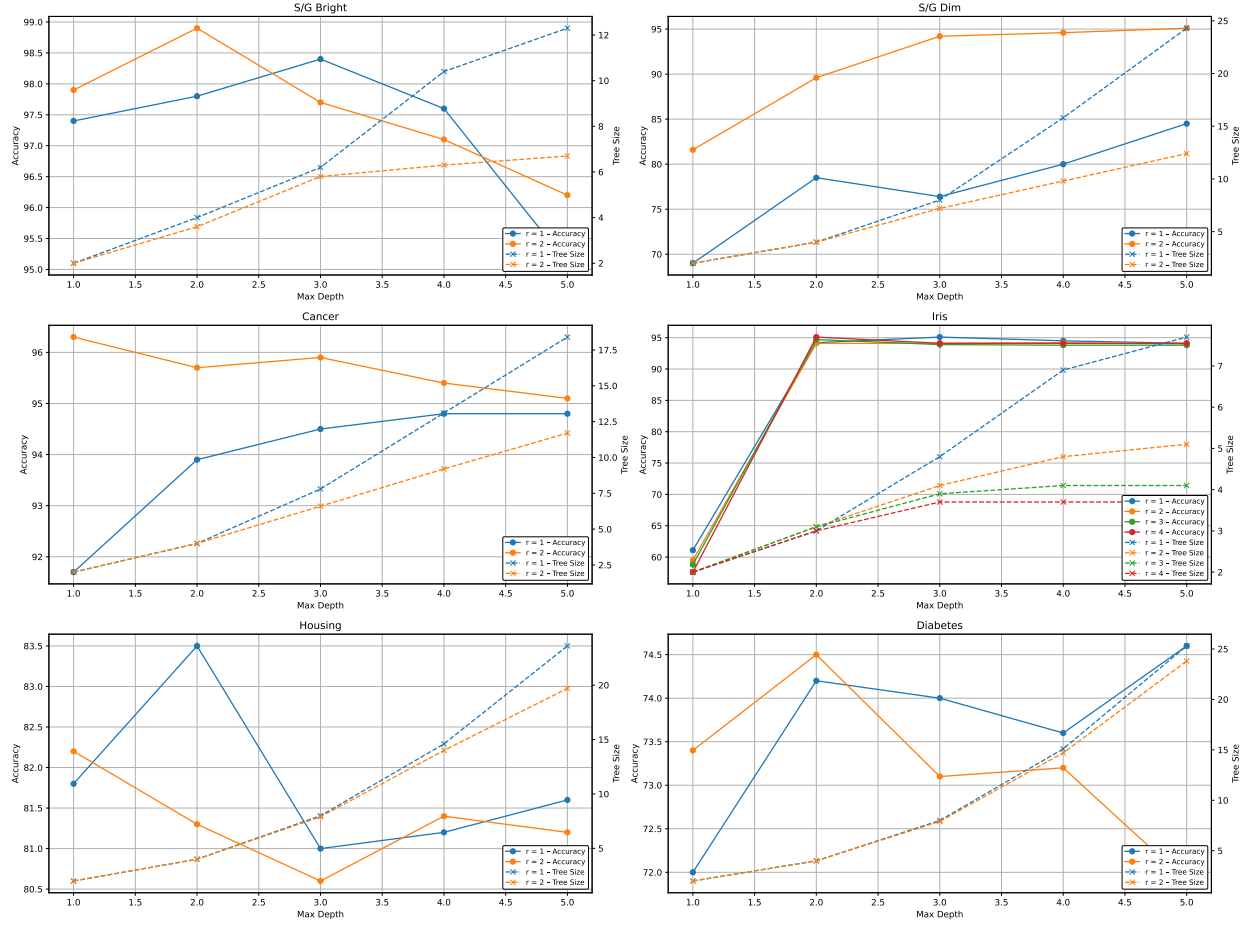
Figure 3: Max depth and r value results for CART-ELC when using the Gini criterion as the splitting criterion.
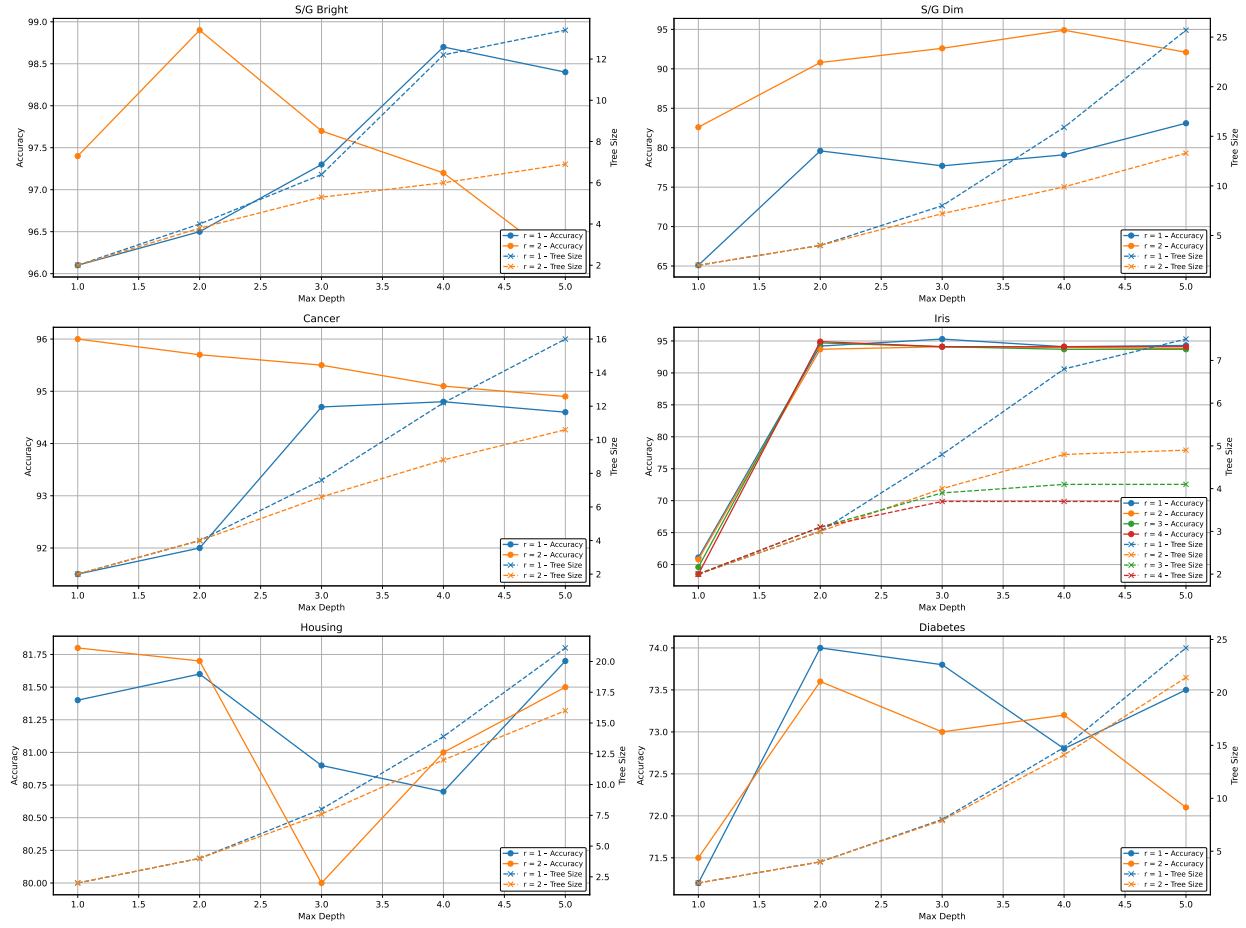
Figure 4: Max depth and r value results for CART-ELC when using information gain as the splitting criterion.