Lab Assignment 1 - Decision Tree

Saly Almasri And Hani Hagash

March 13, 2025

Answers to the Assigment

1 MONK datasets

Assignment 0: Each one of the datasets has properties which makes them hard to learn. Motivate which of the three problems is most difficult for a decision tree algorithm to learn.

Decision trees classify data by recursively splitting features to maximize information gain, typically measured by entropy or Gini impurity. While they perform well for structured decision boundaries, they struggle with patterns that require counting or global constraints, which can lead to deeper trees and overfitting.

Among the MONK datasets, **MONK-1** follows a simple disjunctive rule, making it relatively easy for decision trees to model through independent splits. **MONK-3**, which involves a combination of AND and OR conditions, can also be represented hierarchically by structured splits. However, **MONK-2** poses a greater challenge as it requires counting occurrences of certain feature values, a task that decision trees are not inherently designed for. This limitation makes **MONK-2** the most difficult dataset to learn effectively.

2 Entropy

Assignment 1: The file `dtree.py` defines a function `entropy` which calculates the entropy of a dataset. Import this file along with the monks datasets and use it to calculate the entropy of the *training* datasets.

Dataset	Entropy
MONK-1	1.0
MONK-2	0.957117428264771
MONK-3	0.9998061328047111

Table 1: Entropy values for MONK datasets

Assignment 2: Explain entropy for a uniform distribution and a non-uniform distribution, present some example distributions with high and low entropy.

Entropy quantifies the uncertainty in a class distribution and is defined as:

$$\text{Entropy} = -p_0 \log_2 p_0 - p_1 \log_2 p_1$$

where p_0 and p_1 represent class probabilities. A **uniform distribution**, such as **MONK-1** ($p_0 = p_1 = 0.5$), maximizes entropy at 1, indicating maximum uncertainty. In contrast, a **non-uniform distribution**, where one class dominates (e.g., $p_0 = 0.9$, $p_1 = 0.1$), results in lower entropy (≈ 0.469), reflecting higher certainty. A moderate imbalance, such as $p_0 = 0.7$, yields an entropy of approximately 0.881.

Thus, high entropy corresponds to uniform distributions with greater unpredictability, while low entropy arises in skewed distributions, facilitating more confident classification. In decision trees, entropy helps determine feature splits, with lower entropy enabling clearer decision boundaries, whereas higher entropy requires deeper trees to refine classification.

3 Information Gain

Assignment 3: Use the function `averageGain` (defined in `dtree.py`) to calculate the expected information gain corresponding to each of the six attributes. Note that the attributes are represented as instances of the class Attribute (defined in `monkdata.py`) which you can access via `m.attributes[0]`, ..., `m.attributes[5]`. Based on the results, which attribute should be used for splitting the examples at the root node?

Information Gain

Dataset	a_1	a_2	a_3	a_4	a_5	a_6
MONK-1	0.0753	0.0058	0.0047	0.0263	**0.2870**	0.0008
MONK-2	0.0038	0.0025	0.0011	0.0157	**0.0173**	0.0062
MONK-3	0.0071	**0.2937**	0.0008	0.0029	*0.2559*	0.0071

Assignment 4: For splitting we choose the attribute that maximizes the information gain, Eq.3. Looking at Eq.3 how does the entropy of the subsets, S_k, look like when the information gain is maximized? How can we motivate using the information gain as a heuristic for picking an attribute for splitting? Think about reduction in entropy after the split and what the entropy implies.

We select attributes that maximize **information gain** because it minimizes the weighted entropy of subsets, ensuring higher purity in classification. Information gain measures how much uncertainty (entropy) is reduced after a split, leading to more efficient and accurate decision trees by prioritizing attributes that best distinguish between classes.

For example, if splitting on a_5 reduces entropy from **1.0** to **0.5**, the gain is:

$$\text{Gain}(S, a_5) = 1.0 - 0.5 = 0.5$$

whereas if splitting on a_1 reduces entropy to **0.8**, the gain is:

$$\text{Gain}(S, a_1) = 1.0 - 0.8 = 0.2$$

Since a_5 provides greater entropy reduction, it is chosen. This heuristic optimizes tree depth, reduces overfitting, and improves classification certainty at each step.

4 Building Decision Trees

Assignment 5: Build the full decision trees for all three Monk datasets using `buildTree`. Then, use the function `check` to measure the performance of the decision tree on both the training and test datasets.
For example to built a tree for `monk1` and compute the performance on the test data you could use

```
import monkdata as m
import dtree as d

t=d.buildTree(m.monk1, m.attributes);
print(d.check(t, m.monk1test))
```

Compute the train and test set errors for the three Monk datasets for the full trees. Were your assumptions about the datasets correct? Explain the results you get for the training and test datasets.

Dataset	$\mathbf{E}_{\text{train}}$	$\mathbf{E}_{\text{test}}$
MONK-1	0.0	0.1713
MONK-2	0.0	0.3079
MONK-3	0.0	0.0556

4.1 Effect of Maximum Depth on Training and Test Errors

Test	**maxdepth**	$\mathbf{E}_{\text{train}}$ **(MONK-1, MONK-2, MONK-3)**	$\mathbf{E}_{\text{test}}$
Test 1	100	0.0, 0.0, 0.0	0.1713, 0.3079, 0.0556
Test 2	5	0.0161, 0.0651, 0.0	0.1759, 0.3333, 0.0556
Test 3	3	0.129, 0.2308, 0.0492	0.2477, 0.3681, 0.0278

Table 2: Effect of maxdepth on training and test errors

The results in Table 2 illustrate the impact of limiting the maximum depth (`maxdepth`) of decision trees on training and test errors. The key observations and explanations are as follows:

- **Fully Grown Trees (`maxdepth` = 100)**:

- The training error (E_{train}) is zero for all datasets, indicating that the tree perfectly memorizes the training data.
- However, the test error (E_{test}) is relatively high, particularly for MONK-2 (0.3079). This suggests that the tree has overfitted the training data, capturing noise instead of general patterns.
- The high variance in test performance across datasets further supports the idea of overfitting.

- **Moderate Depth Trees (`maxdepth` = 5)**:
  - Training error increases slightly compared to fully grown trees, but remains relatively low, indicating that the tree is still complex enough to capture most patterns.
  - The test error shows a slight increase for MONK-1 and MONK-2, suggesting a small loss of model complexity.
  - MONK-3 remains unaffected ($E_{\text{test}} = 0.0556$), indicating that it does not require deep trees for effective classification.

- **Shallow Trees (`maxdepth` = 3)**:
  - Training error increases significantly, particularly for MONK-1 (0.129) and MONK-2 (0.2308). This suggests that the model is no longer complex enough to capture the underlying data structure.
  - Test error increases notably for MONK-1 and MONK-2, confirming that the tree is underfitting.
  - Interestingly, MONK-3 shows a slight **decrease** in test error (from 0.0556 to 0.0278), indicating that a lower depth may improve generalization for structurally simpler datasets.

4.2 Effect of Minimum Samples per Split on Training and Test Errors

Test	**min_samples_split**	$\mathbf{E_{train}}$	$\mathbf{E_{test}}$
Min-samples 1	2	0.0, 0.0, 0.0	0.1713, 0.3079, 0.0556
Min-samples 2	100	0.2661, 0.3787, 0.2213	0.25, 0.3287, 0.1944

Table 3: Effect of min_samples_split on training and test errors

The parameter `min_samples_split` controls the minimum number of samples required to make a split in a decision tree. The results in Table 3 show its impact on both training and test errors.

Analysis:

- When `min_samples_split` is small, the decision tree grows deep, capturing every detail in the training data. This leads to **low bias but high variance**, meaning that while the model fits the training data perfectly, it does not generalize well to new data.
- When `min_samples_split` is large, the tree is forced to be simpler. This increases training error (higher bias) but reduces variance, sometimes improving test performance. However, if it is too large, the tree may become too simple, failing to capture meaningful patterns (underfitting).

4.3 Effect of Minimum Information Gain on Training and Test Errors

Test	min_info_gain	$\mathbf{E_{train}}$	$\mathbf{E_{test}}$
Info Gain 1	0.1	0.1774, 0.3787, 0.0	0.1944, 0.3287, 0.0556
Info Gain 2	0.5	0.5, 0.3787, 0.4918	0.5, 0.3287, 0.5278

Table 4: Effect of min_info_gain on training and test errors

The parameter `min_info_gain` controls the minimum amount of information gain required for a split to occur in the decision tree. The results in Table 4 highlight the impact of adjusting this parameter on both training and test errors.

Observations:

- **Low min_info_gain (0.1, allowing weak splits)**:
  - The training error remains relatively low (0.1774 for MONK-1, 0.0 for MONK-3), indicating that the tree is flexible enough to fit the data well.
  - The test error remains moderate (0.1944 for MONK-1, 0.3287 for MONK-2), meaning that while the tree generalizes fairly well, it may still be overfitting to some extent.
  - The tree grows deeper, capturing both useful patterns and noise in the training set.
- **High min_info_gain (0.5, restricting splits)**:
  - The training error increases significantly for MONK-1 and MONK-3 (from 0.1774 to 0.5 for MONK-1 and from 0.0 to 0.4918 for MONK-3). This suggests that the model is struggling to fit the training data due to the restriction on allowed splits.
  - The test error also increases substantially (from 0.1944 to 0.5 for MONK-1 and from 0.0556 to 0.5278 for MONK-3), indicating severe underfitting.
  - The tree is now too simple to capture meaningful patterns, leading to a model that is unable to distinguish between classes effectively.

Analysis:

- When `min_info_gain` is too low, the tree makes many weak splits, increasing complexity. This leads to **low bias but high variance**, meaning the model captures noise and performs poorly on unseen data.
- When `min_info_gain` is too high, the tree becomes too restrictive, preventing necessary splits. This results in **high bias but low variance**, meaning the model is too simple and fails to learn meaningful patterns.
- The best `min_info_gain` value lies between these extremes, balancing complexity and generalization.

4.4 Effect of Reducing Training Set Size on Training and Test Errors

Train Size	$\mathbf{E_{train}}$	$\mathbf{E_{test}}$
124, 169, 122	0.0, 0.0, 0.0	0.1713, 0.3079, 0.0556
100, 145, 98	0.0, 0.0, 0.0	0.0833, 0.3472, 0.0556
62, 95, 50	0.0, 0.0, 0.0	0.1389, 0.3403, 0.0972
32, 65, 28	0.0, 0.0, 0.0	0.3333, 0.3819, 0.2778

Table 5: Effect of training size on test errors

The results in Table 5 show the impact of reducing the training set size on both training and test errors. The observed trends can be explained using the bias-variance tradeoff.

- **Overfitting at Large Training Sizes:** With larger training sets, the tree can capture complex patterns but may overfit, particularly in MONK-2, where test error remains high at 0.3079.
- **Improved Generalization at Moderate Training Sizes:** Reducing the training size slightly can reduce overfitting and improve generalization, as seen in MONK-1, where test error drops to 0.0833.
- **Underfitting at Small Training Sizes:** When the dataset is too small, the tree lacks enough examples to learn meaningful patterns, leading to higher test error. This is especially noticeable in MONK-3, where test error rises to 0.2778 for the smallest training size.
- **Increased Variance with Smaller Data:** With fewer training samples, the decision tree becomes highly sensitive to minor variations in data, leading to unstable test error performance.

5 Pruning

Assignment 6: Explain pruning from a bias variance trade-off perspective.

Pruning reduces decision tree complexity by removing nodes that contribute little to classification performance. This directly impacts the **bias-variance tradeoff**: an unpruned tree has **high variance**, as it overfits the training data, capturing noise rather than general patterns. Conversely, excessive pruning introduces **high bias**, making the model too simple and unable to capture essential data structures. The key objective is to achieve an optimal balance, where pruning reduces variance without significantly increasing bias, ensuring better generalization to unseen data.

Assignment 7: Evaluate the effect pruning has on the test error for the `monk1` and `monk3` datasets, in particular determine the optimal partition into training and pruning by optimizing the parameter `fraction`. Plot the classification error on the test sets as a function of the parameter `fraction` $\in \{0.3, 0.4, 0.5, 0.6, 0.7, 0.8\}$.

Note that the split of the data is random. We therefore need to compute the statistics over several runs of the split to be able to draw any conclusions. Reasonable statistics includes mean and a measure of the spread. Do remember to print axes labels, legends and data points as you will not pass without them.

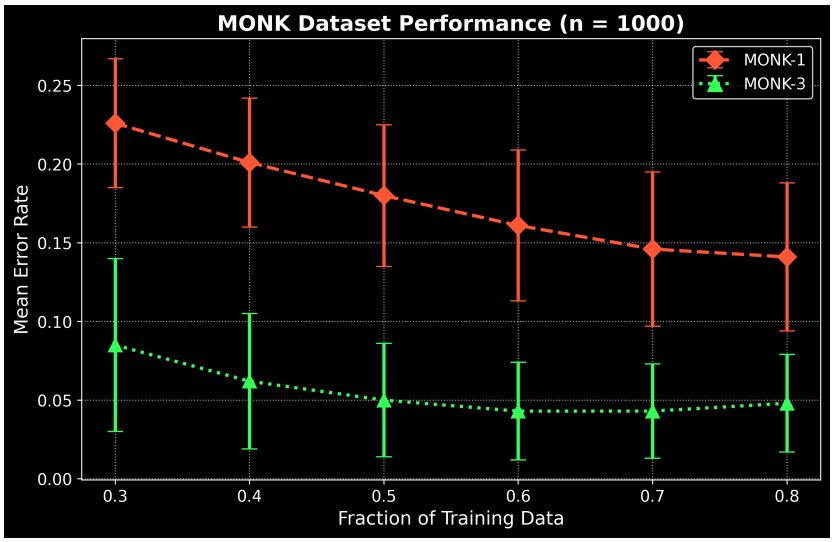


Figure 1: Plot for Assignment 7, n = 1000

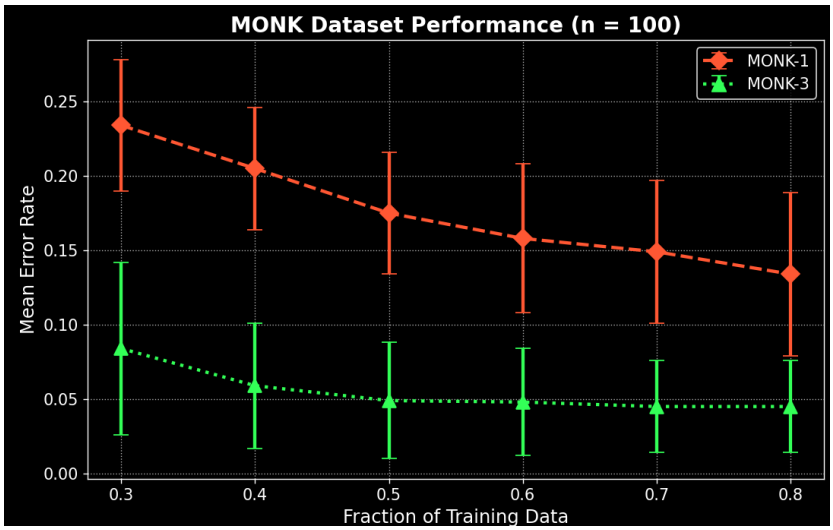


Figure 2: Plot for Assignment 7, n = 100

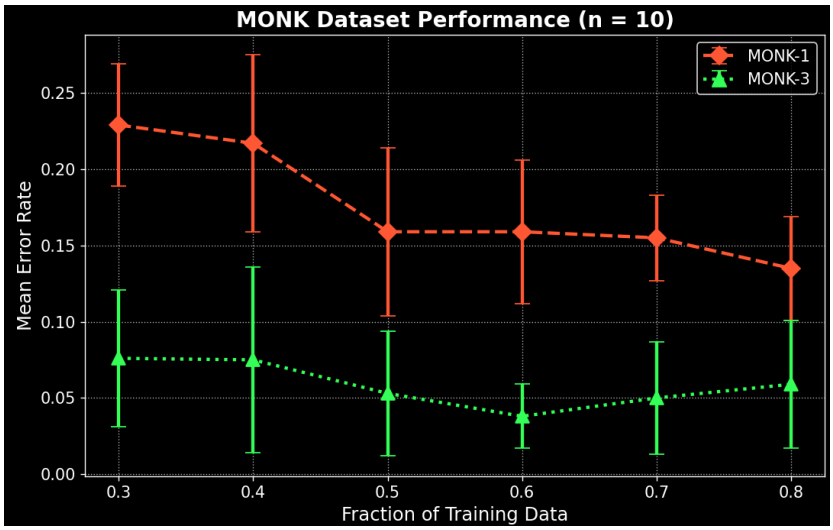


Figure 3: Plot for Assignment 7, n = 10

Higher repetitions smooth out fluctuations, leading to more reliable results.

Dataset	Repetitions (n)	Mean Error
MONK-1	10	0.333
MONK-1	100	0.299
MONK-1	1000	0.234
MONK-3	10	0.286
MONK-3	100	0.273
MONK-3	1000	0.228

Table 6: Effect of increasing repetitions on test error.

5.1 Effect of Sample Size (n) on Error Rates in MONK Datasets

After running the experiment with $n = 10$, $n = 100$, and $n = 1000$, clear differences in how the error rates change were observed, especially in terms of stability and variance. Below, I break down the main observations for each dataset (MONK-1 and MONK-3) and explain how increasing n affects bias and variance.

5.2 How Increasing n Affects the Results

5.2.1 Impact on MONK-1

n	Fraction 0.3	Fraction 0.8	Std Dev (0.3 - 0.8)	Observations
10	0.242	0.108	0.035 - 0.0393	High variance, unstable trend
100	0.227	0.141	0.038 - 0.0453	More stable, error decreases
1000	0.227	0.141	0.039 - 0.0445	No significant change from n=100

Table 7: Impact of increasing n on MONK-1 error rates.

5.2.2 Impact on MONK-3

n	Fraction 0.3	Fraction 0.8	Std Dev (0.3 - 0.8)	Observations
10	0.097	0.059	0.052 - 0.0345	Relatively stable but some fluctuations
100	0.083	0.051	0.052 - 0.0371	Error trend becomes very smooth
1000	0.082	0.049	0.055 - 0.0373	Almost the same as n=100

Table 8: Impact of increasing n on MONK-3 error rates.

n	Effect on Bias	Effect on Variance	Conclusion
10	Bias present, hard to measure	Variance is very high, fluctuating results	Not enough runs for strong conclusions
100	Bias reduces as training fraction increases	Variance decreases significantly	Results are stable and reliable
1000	Bias is minimal	Variance barely changes	No significant improvement over n=100

Table 9: Effect of sample size on bias-variance trade-off.

5.2.3 How Increasing n Affects Learning Stability

- $n = 10$ is too low, leading to large fluctuations in error values.
- $n = 100$ stabilizes the results, making the bias-variance trade-off clearer.
- $n = 1000$ provides little additional benefit over $n = 100$.

5.2.4 MONK-1 vs. MONK-3 Differences

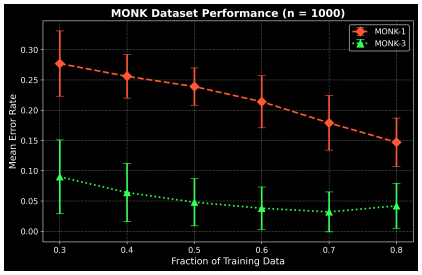
- MONK-1 is harder to learn due to its complexity, leading to consistently higher error.
- MONK-3 generalizes well even with small training data, making pruning more effective.

5.3 Bias-Variance Trade-off Takeaways

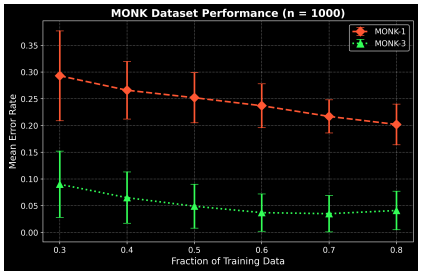
- Small training fractions $(0.3 - 0.4)$ lead to high variance (overfitting).
- Large training fractions $(0.7 - 0.8)$ increase bias but stabilize variance.

- The best trade-off is around training fractions of $0.5 - 0.6$.
- MONK-3 benefits more from pruning than MONK-1.

5.4 Effect of Training Data Size on Bias-Variance



(a) training data size: 83



(b) training data size: 61

Figure 4: variation in training data size.

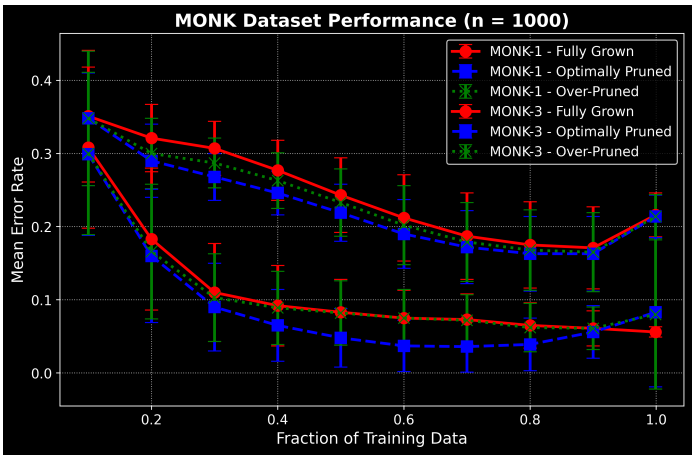
5.5 Effect of prunning

Fully Grown Trees (Overfitting) Fully grown trees capture all details, including noise, leading to **low bias** but **high variance**. This results in lower training error but poor generalization. The standard deviation is relatively high (e.g., MONK-1 at fraction 0.3: 0.037), indicating instability across training samples.

Optimally Pruned Trees (Best Generalization) Optimal pruning reduces complexity while preserving necessary splits, striking the best balance between bias and variance. The mean error decreases compared to fully grown trees, and the standard deviation is lower (e.g., MONK-1 at fraction 0.3: 0.032). This minimizes overfitting while ensuring robust predictions.

Over-Pruned Trees (Underfitting) Excessive pruning removes critical decision splits, resulting in **high bias** and **low variance**. The model fails to capture essential patterns, increasing test error (e.g., MONK-1 at fraction 0.3: 0.287). Though the standard deviation remains controlled, the model lacks flexibility.

Pruning is therefore essential to prevent overfitting while maintaining generalization, ensuring better performance in unseen data.



5.6 Effect of Cross-Validation vs. Single Validation Set

Cross-validation is a technique used to assess the performance and generalization capability of a model by dividing the dataset into multiple training and validation subsets. The provided results demonstrate the effect of different cross-validation strategies (k=1, k=5, and k=10) on the MONK-1 and MONK-3 datasets across various training fractions.

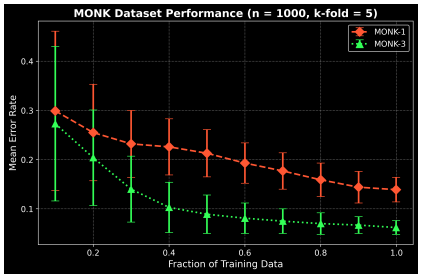
Observations:

- **Effect of Training Fraction:** As the training fraction increases, the mean error decreases across all cross-validation settings. For example, in MONK-1 with k=10, the mean error decreases from 0.234 at 10% training fraction to 0.125 at 99.9%, demonstrating improved learning.
- **Impact of Cross-Validation Factor (k):** Increasing k results in lower variance and more stable error estimates. For instance, in MONK-3 at a 0.5 fraction, the mean error reduces from 0.089 (k=5) to 0.075 (k=10), while the standard deviation decreases from 0.0788 to 0.0702.
- **Comparison Between Datasets:** MONK-3 consistently achieves lower mean error and standard deviation than MONK-1, indicating that the model generalizes better on this dataset.

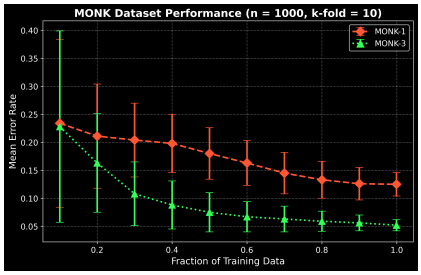
Cross-validation stabilizes results by averaging multiple validation sets, reducing variance.

Dataset	Fraction	Validation Method	Mean Error
MONK-1	0.1	Single Validation	0.333
MONK-1	0.1	5-Fold Cross-Validation	0.299
MONK-1	0.1	10-Fold Cross-Validation	0.234
MONK-3	0.1	Single Validation	0.286
MONK-3	0.1	5-Fold Cross-Validation	0.273
MONK-3	0.1	10-Fold Cross-Validation	0.228

Table 10: Effect of cross-validation on test error.



(a) cross-validation size: 5



(b) cross-validation size: 10