Lab 1: Decision Trees

DD2421

Yuqi Zheng Jingxuan Mao

Assignment 0

Intuitively, MONK-2 is the most difficult for the decision tree algorithm to learn. Because each attribute is relevant. For example, when we find $a_1 = a_2 = 1$, we still need to check whether a_3, a_4, a_5, and a_6 are equal to 1. Hence, this will result in a rather complex tree which is hard to learn. In contrast, the attributes of MONK-1 and MONK-3 have lower relevance, which makes MONK-1 and MONK-3 much easier to learn.

This is also validated later.

Assignment 1

Dataset	Entropy
MONK-1	1.0000
MONK-2	0.9571
MONK-3	0.9998

Assignment 2

$$\text{Entropy}(S) = -\sum_i p_i \log_2 p_i$$

Considering the result of an exam, there are two possibilities:

Pass or Fail

Assume :

P(Pass) = p, P(Fail) = 1-p

The curve Entropy vs. p is shown in Fig.1
So we can see the highest entropy 1 is reached when p = 1-p = 0.5

When p = 0.1, P(Pass) = 0.1, P(Fail) = 0.9
Entropy = 0.4690

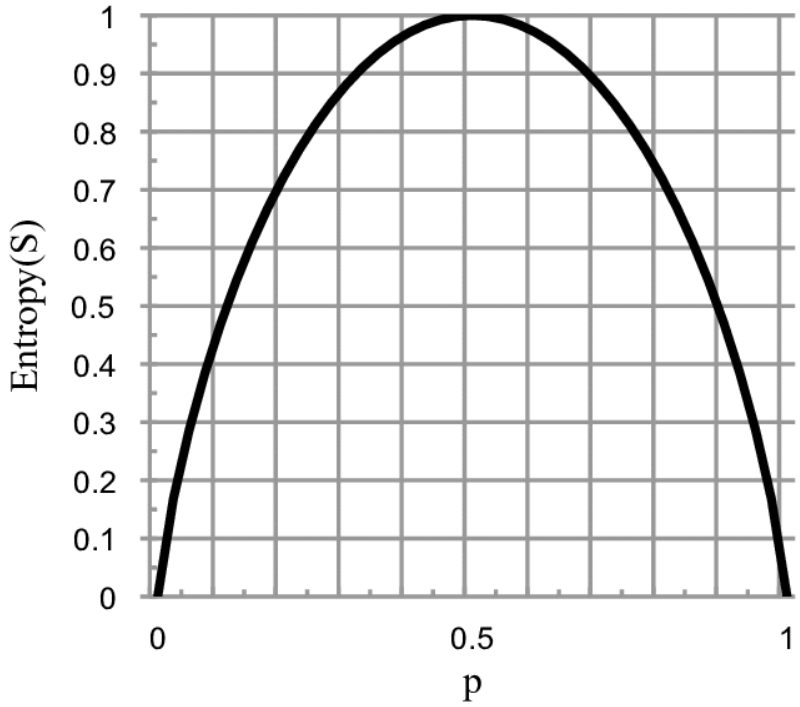


Fig.1

Assignment 3

Dataset	a1	a2	a3	a4	a5	a6	
MONK-1	0.07527	0.00584	0.00471	0.02631	0.28703	0.0076	**a5**
MONK-2	0.00376	0.00246	0.00106	0.01566	0.01728	0.00625	**a5**
MONK-3	0.00712	0.29374	0.00083	0.00289	0.25591	0.00708	**a2**

Assignment 4

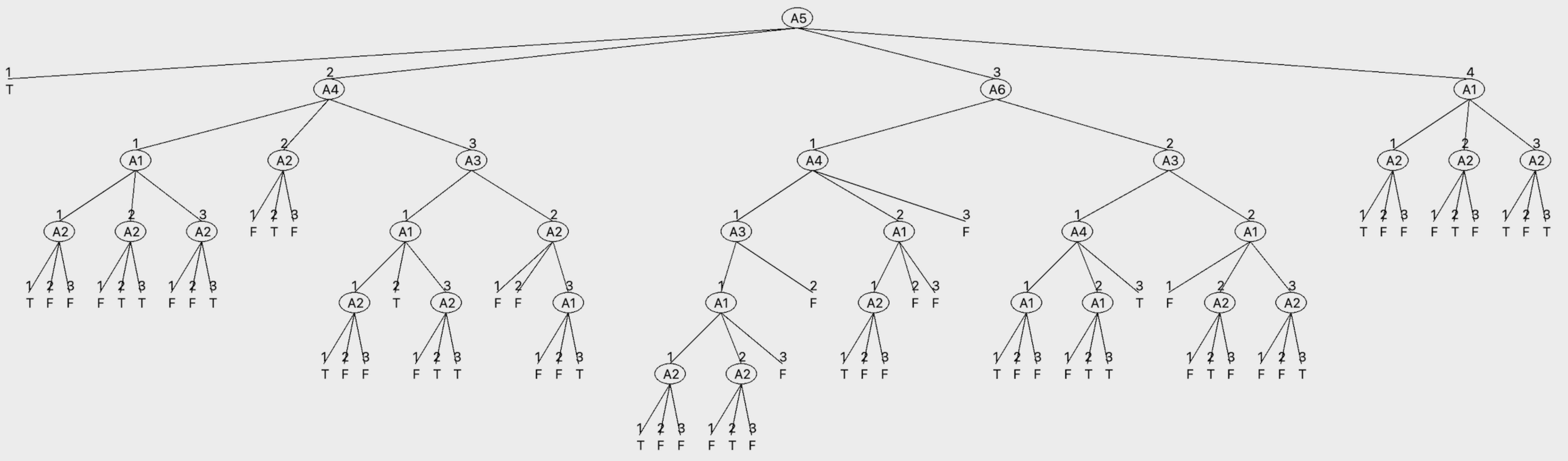
$$\text{Gain}(S, A) = \text{Entropy}(S) - \sum_{k \in \text{values}(A)} \frac{|S_k|}{|S|} \text{Entropy}(S_k) \qquad (3)$$

Entropy indicates the uncertainty in the dataset. The attribute with the highest gain can maximize the expected reduction of the entropy, therefore the uncertainty in its subsets is also reduced most. Which means the purity of the subsets increases and we can keep moving down for further splitting in the most effective way.

Assignment 5

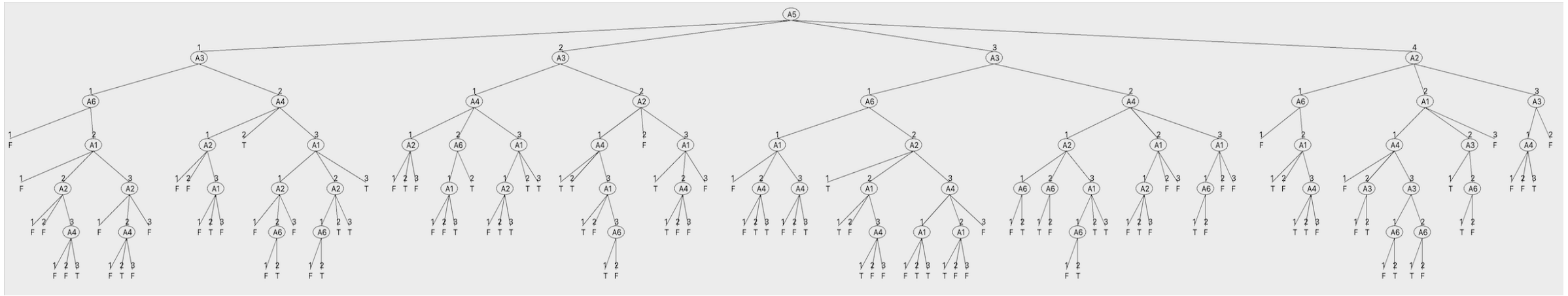
	E_train	E_test
MONK-1	0.0000	0.1713
MONK-2	0.0000	0.3079
MONK-3	0.0000	0.0556

MONK-1

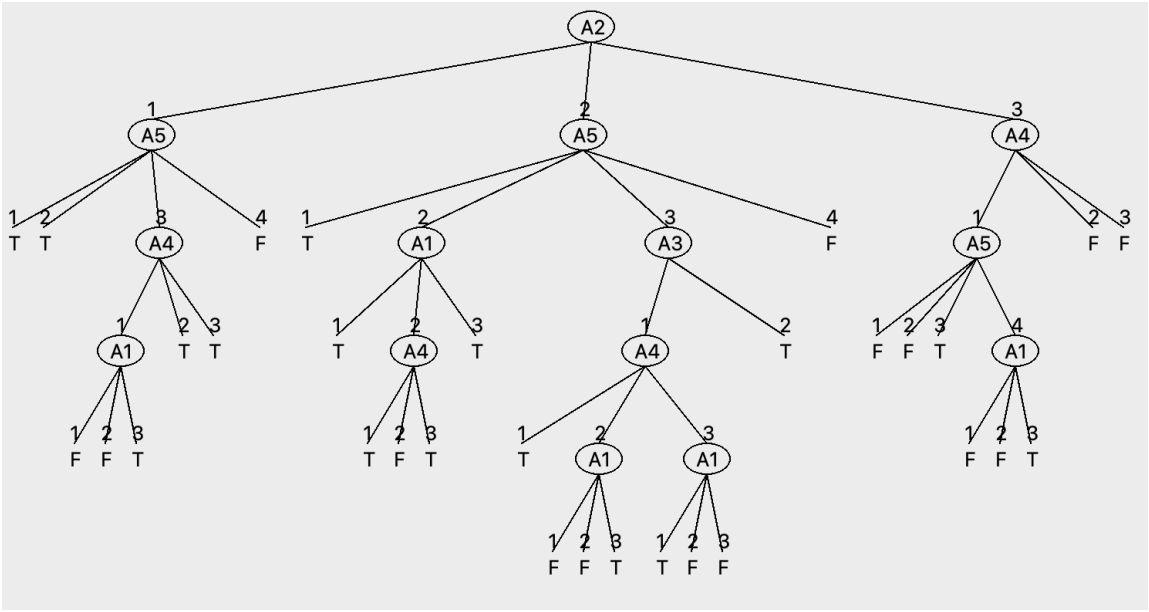


Assignment 5

MONK-2



MONK-3



Assignment 6

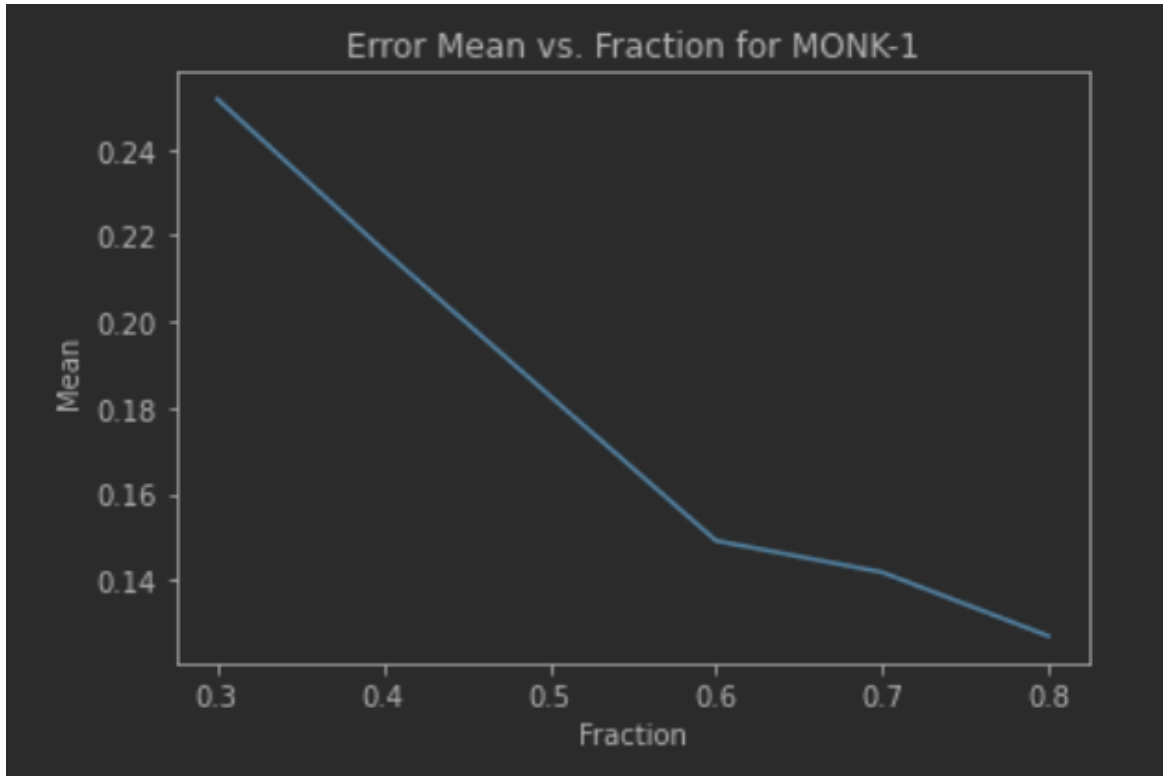
Variance depends on the depth of the decision tree. When the decision tree is pruned, the structure of it is simpler. Hence, the variance will be reduced (also the overfitting will be avoided). However, less precise classification would lead to greater bias, which is a trade-off we need to consider in practical application.

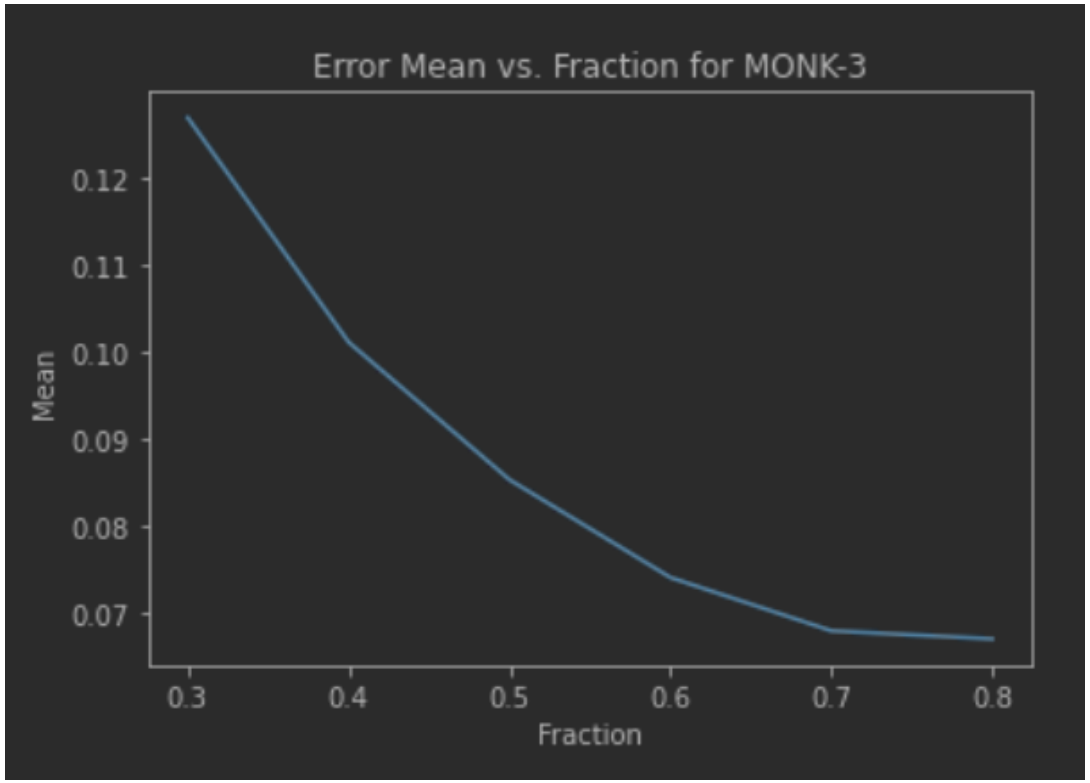
Assignment 7

```python
def best_prune(dataset, fraction):
    train, val = partition(dataset, fraction)
    t_o = d.buildTree(train, m.attributes)
    while():
        E_old = d.check(t_o, val)
        t_new = []
        t_new = d.allPruned(t_o)
        Emax = -1
        for i in range(len(t_new)):
            E = d.check(t_new[i], val)
            if E > Emax:
                best_idx = i
                Emax = E

        E_new = Emax

        if E_new <= E_old:
            break
        else:
            t_o = t_new[best_idx]

    return t_o
```

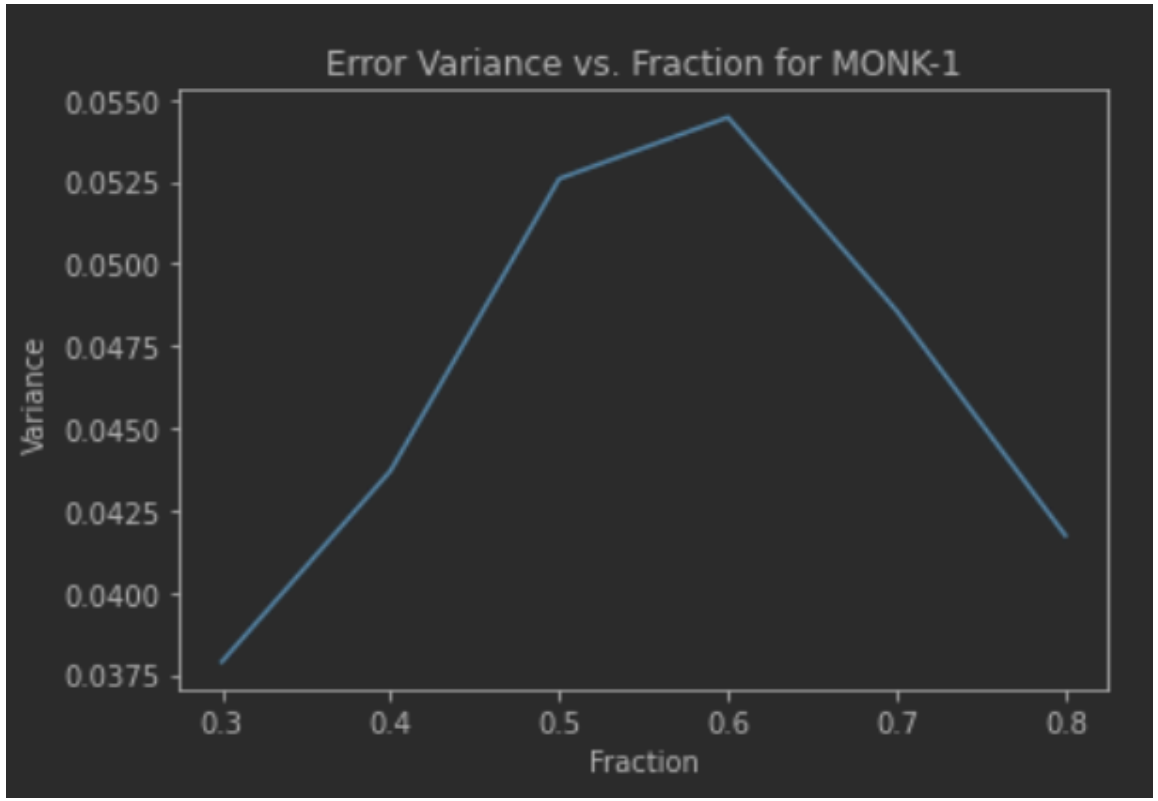
```python
def best_frac(dataset, testset, runtime):

    fraction = [0.3, 0.4, 0.5, 0.6, 0.7, 0.8]
    E_mean = np.zeros(len(fraction))
    E_var = np.zeros(len(fraction))

    for i in range(len(fraction)):
        f = fraction[i]
        E = np.zeros(runtime)
        for k in range(runtime):
            t_best = best_prune(dataset, f)
            E[k] = 1 - d.check(t_best, testset)

        E_mean[i] = np.mean(E)
        E_var[i] = np.std(E)

    print(E_mean)
    print(E_var)


MONK1 = m.monk1
MONK3 = m.monk3
best_frac(MONK1, m.monk1test, 30)
best_frac(MONK3, m.monk3test, 30)
```

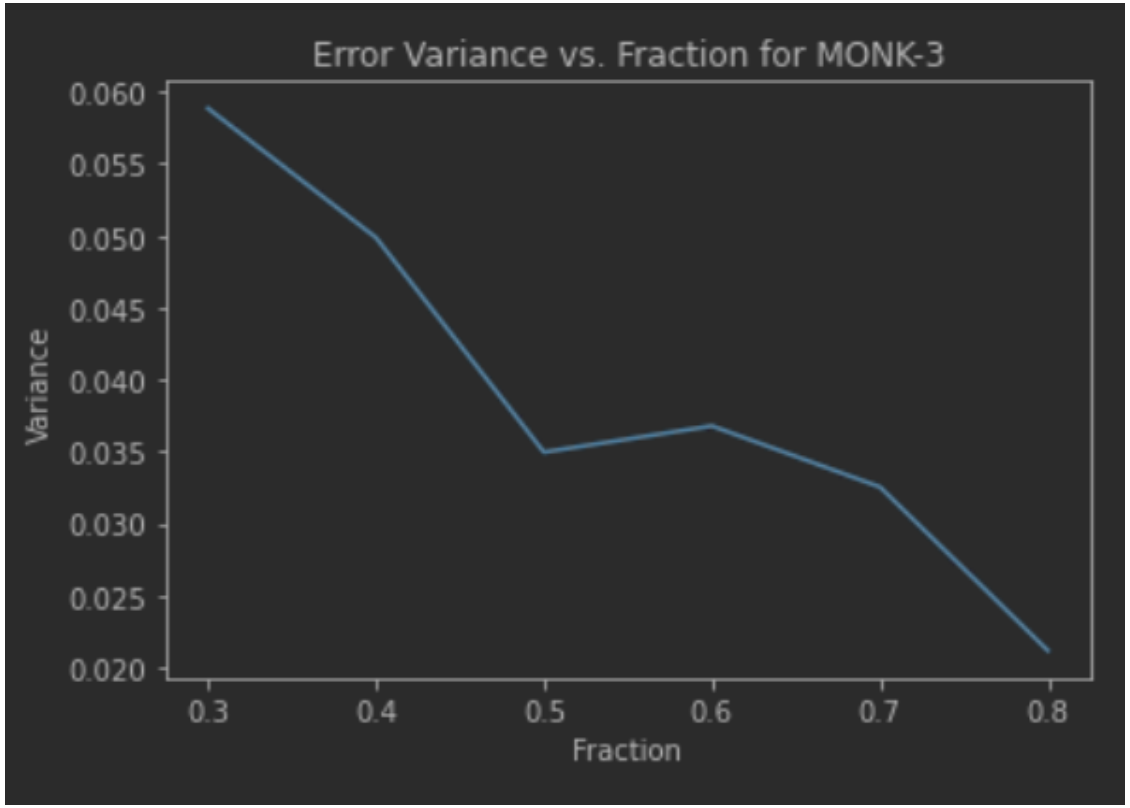
Assignment 7





Assignment 7





Thanks for listening :)