Machine Learning
Lab1 Decision Trees

Catherine Weldon, Qiao Ren

Assignment 0

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.

- From first glance, dataset #3 seems the hardest to learn because it has 5% noise
- We also expect the dataset with the most entropy to be hard to learn as it has the most uncertainty

Assignment 1

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.

```
#calculate entropy of data set
entropym1 = dtree.entropy(m.monk1)
print(entropym1)
entropym2 = dtree.entropy(m.monk2)
print(entropym2)
entropym3 = dtree.entropy(m.monk3)
print(entropym3)
```

```
1.0
0.957117428264771
0.9998061328047111
```

Assignment 2

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

A uniform distribution will have higher entropy because there is more uncertainty in the sample.

Uniform example: Normal dice – each side has equal probability of landing upwards. In this case we would expect a higher entropy.

Non-uniform example: Unfair dice – one side is more likely to land face up, making there less uncertainty in the outcome and decreasing the entropy.

Assignment 3

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?

- For Monk 1, the highest information gain is at a5 (0.2870)
- For Monk 2, the highest information gain is also at a5 (0.0173)
- For Monk 3, the highest information gain is at a2 (0.2937)

information Gain						
	a1	a2	a3	a4	a5	a6
monk1	0.0753	0.0058	0.0047	0.0263	**0.2870**	0.0008
monk2	0.0038	0.0025	0.0011	0.0157	**0.0173**	0.0062
monk3	0.0071	**0.2937**	0.0008	0.0029	0.2559	0.0071

Assignment 4

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.

```
entropy11 = dtree.entropy(monk11)
entropy12 = dtree.entropy(monk12)
entropy13 = dtree.entropy(monk13)
entropy14 = dtree.entropy(monk14)
print(entropy11)
print(entropy12)
print(entropy13)
print(entropy14)
```

```
0.0
0.9383153522334069
0.9480782435939054
0.9081783472997051
```

- Heuristic: split each data subset by attribute with highest information gain
- After splitting Monk1 on a5 we see entropy decrease (figure on right) because the split allowed us to gain more information

Assignment 5

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

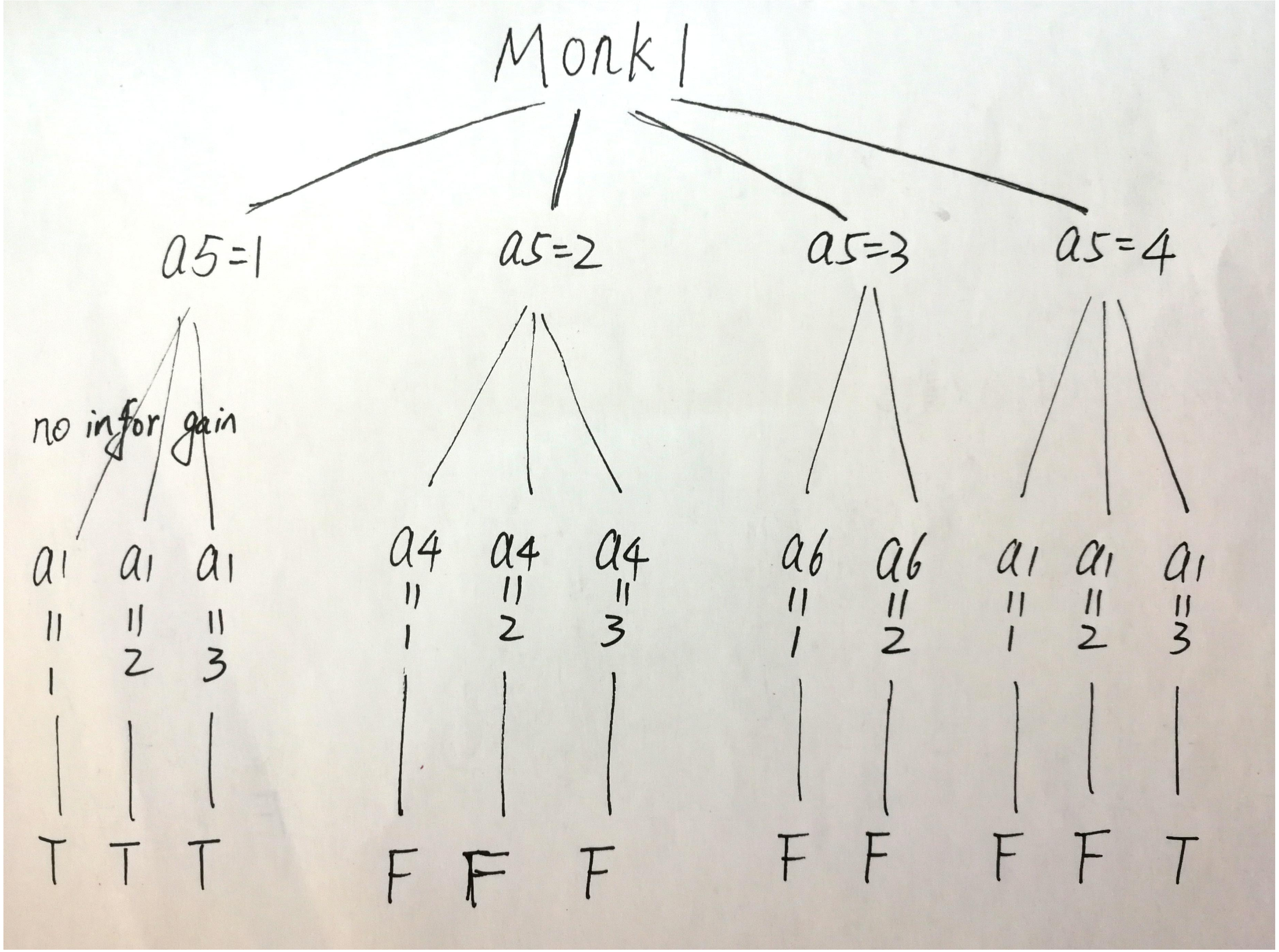
```
import monkdata as m
import dtree as d

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.

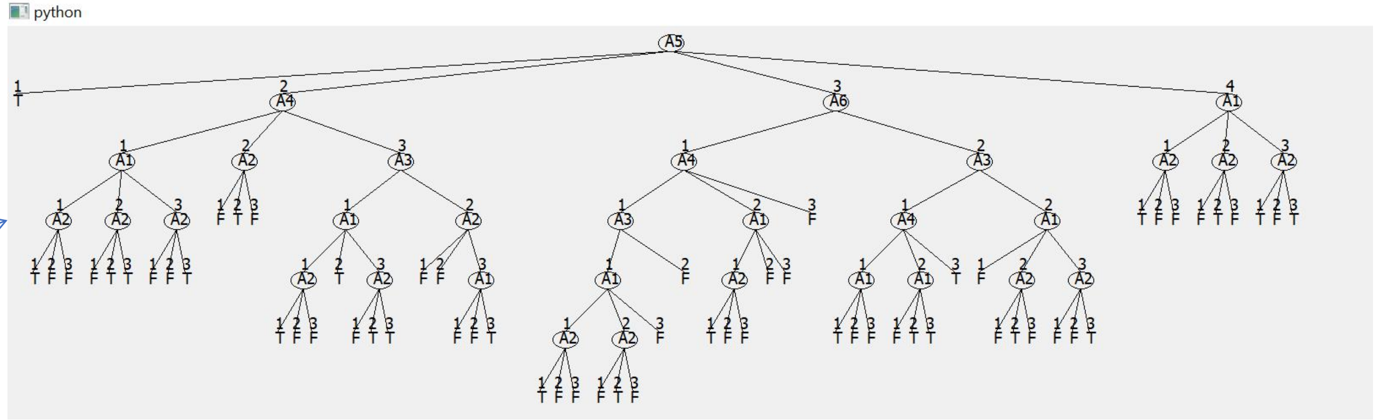
- The test set errors tell us that the Monk2 data set tree performed the worst for the test set
- This was unexpected as Monk3 had the most noise

	Etrain	Etest
Monk1	0	17 13%
Monk2	0	30.79%
Monk3	0	5.56%

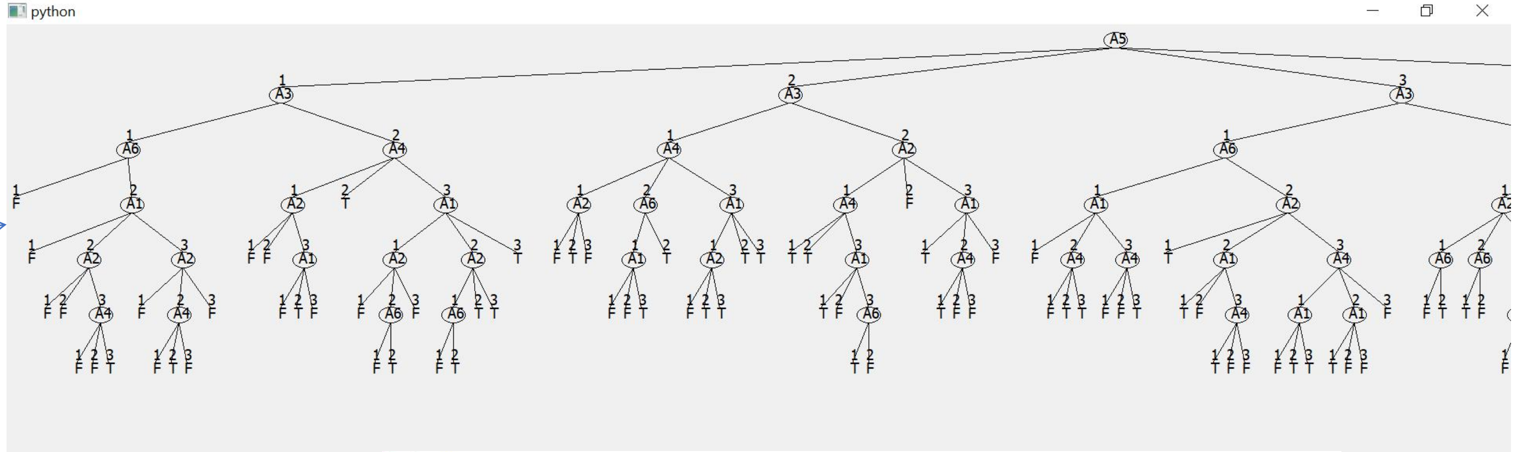

Monk1
a5=1
a5=2
a5=3
a5=4
no infor gain
a1 = 1
a1 = 2
a1 = 3
a4 = 1
a4 = 2
a4 = 3
a6 = 1
a6 = 2
a1 = 1
a1 = 2
a1 = 3
T T T
F F F
F F
F F T

visualize the tree

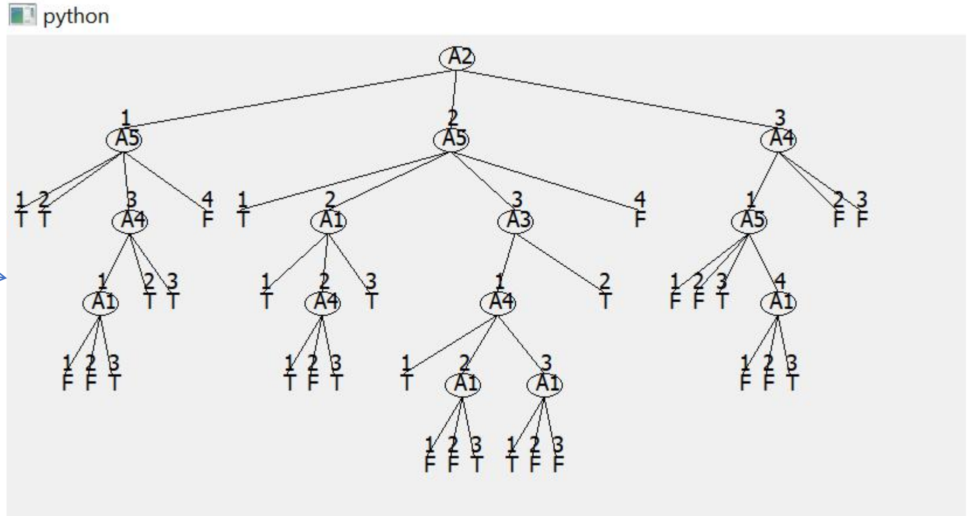
- monk1



- monk2



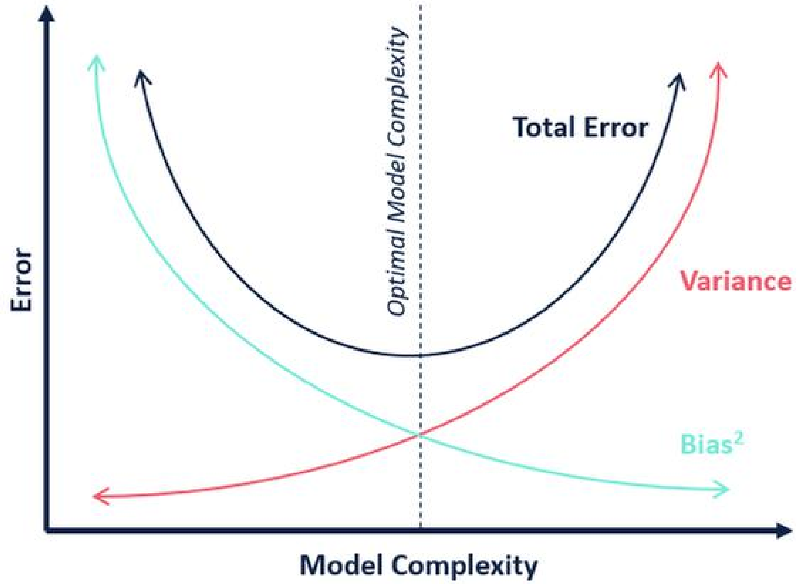
- monk3



Assignment 6

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

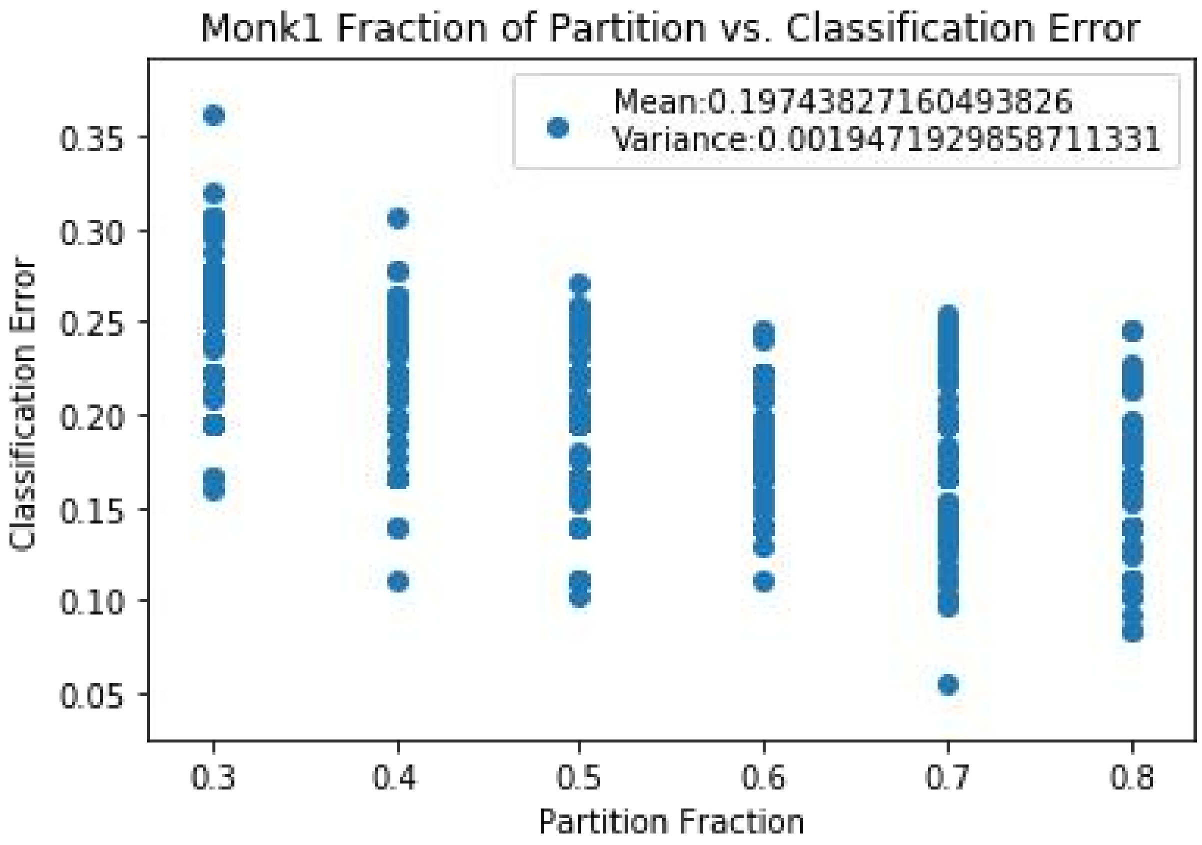
- Pruning may increase the bias on the training set but will decrease the variance and hopefully improve the error rate on the test set
- If we prune too much, the tree may be over-simplified and the bias will increase

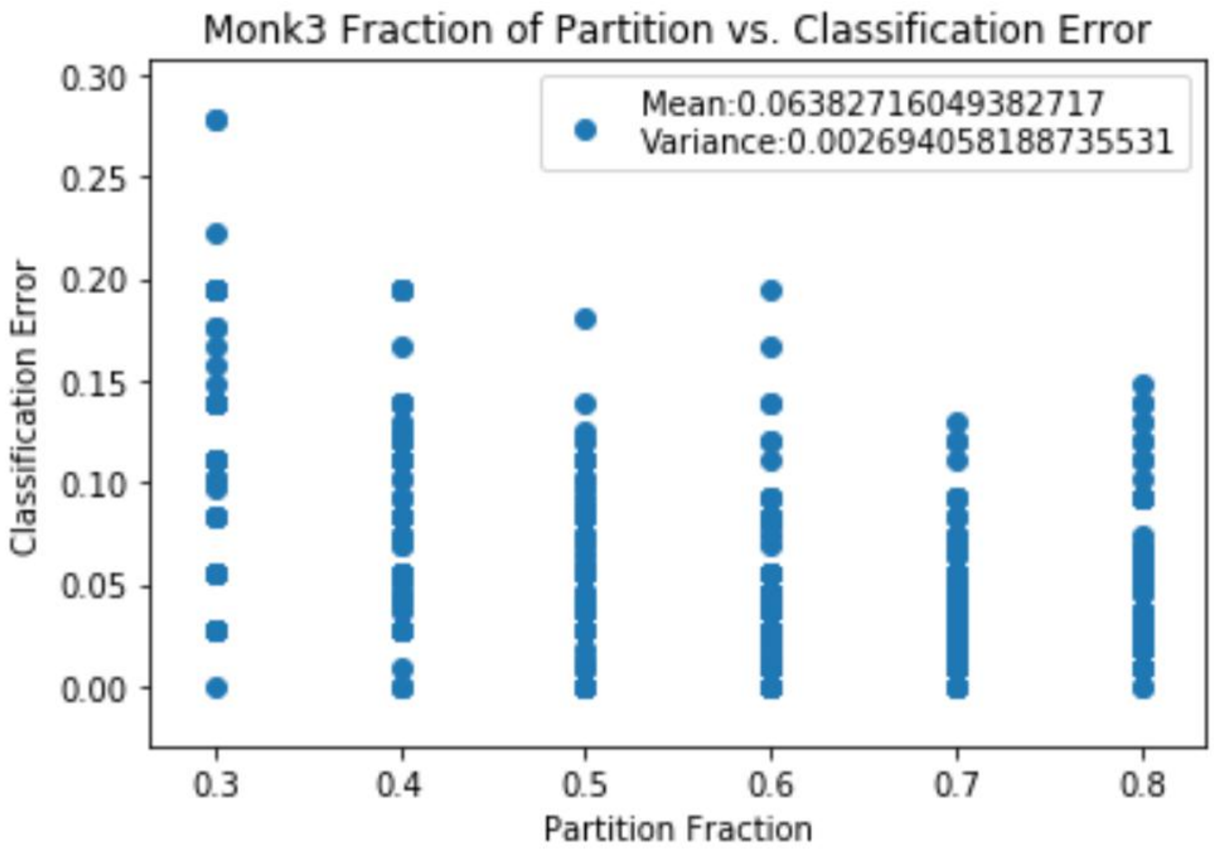


Source: https://community.alteryx.com/t5/Data-Science-Blog/Bias-Versus-Variance/ba-p/351862

Assignment 7

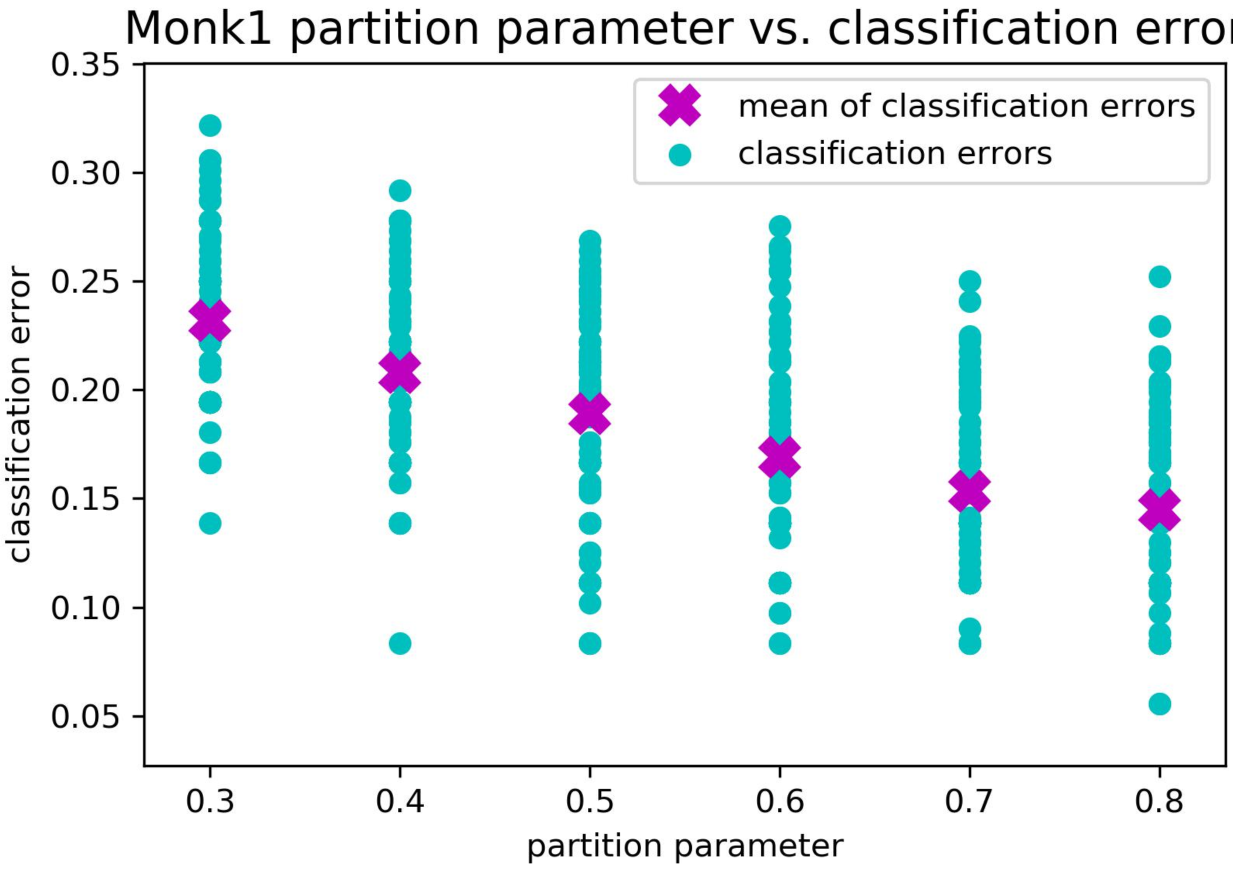
*Classification error decreases on both sample sets as partition increases
*We saw the variance decrease for Monk3 but not for Monk1

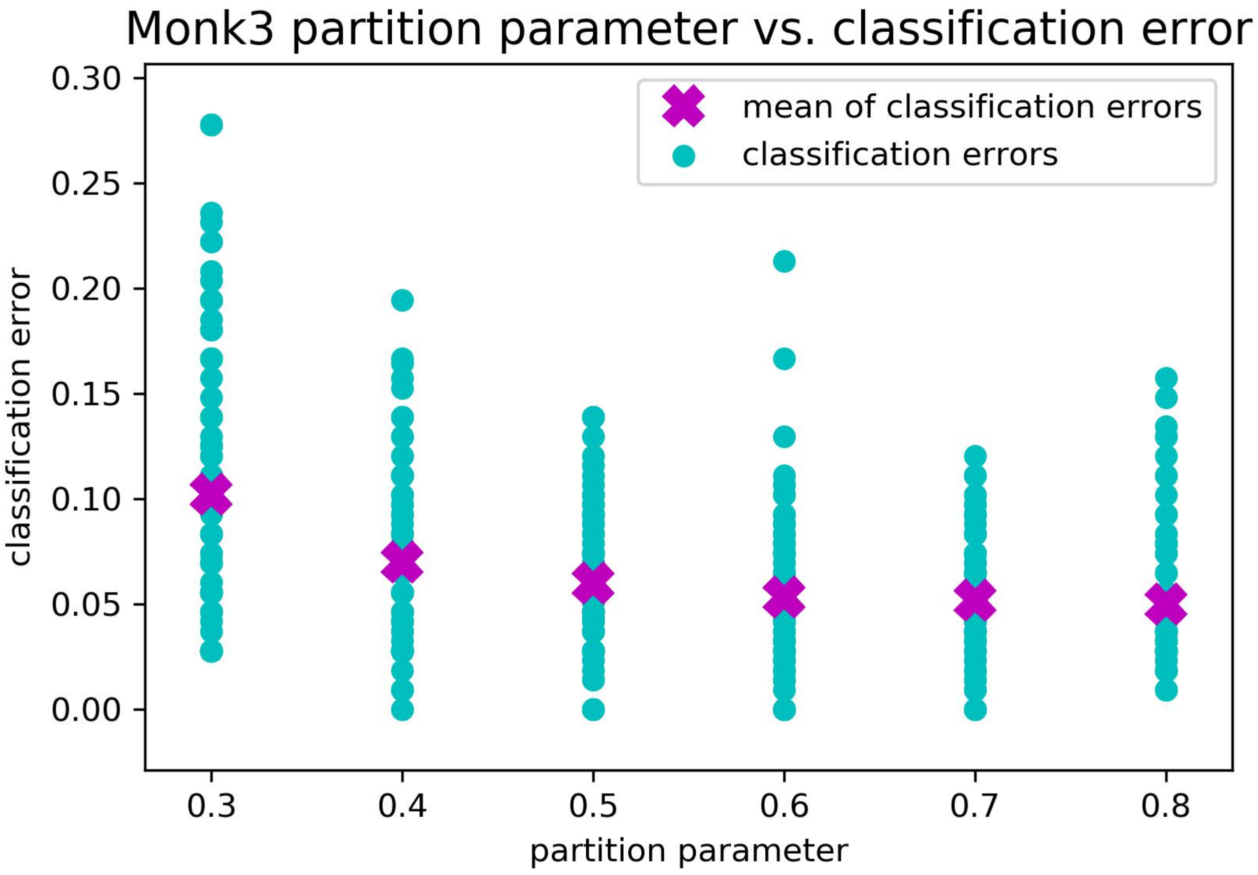




Assignment 7

*Classification error decreases on both sample sets as partition increases
*We saw the variance decrease for Monk3 but not for Monk1





the partition parameter which provides the lowest error: 0.8
mean: 0.1494
variance: 0.0015

the partition parameter which provides the lowest error: 0.8
mean: 0.0518
variance: 0.0013

Assignment 7 *mean and variance of classification errors*

- each partition parameter is run for 100 times. 100 classification errors (on test dataset) correspond to each partition para.

- monk1
  - partition: [0.3, 0.4, 0.5, 0.6, 0.7, **0.8**]
  - mean: [0.2379, 0.2138, 0.1808, 0.1687, 0.1512, **0.1494**]
  - var: [0.0025, 0.0018, 0.0017, 0.0017, 0.0018, **0.0015**]
- monk3
  - partition: [0.3, 0.4, 0.5, 0.6, 0.7, **0.8**]
  - mean: [0.0939, 0.0675, 0.0606, 0.0556, 0.0519, **0.0518**]
  - var: [0.0034, 0.0014, 0.0012, 0.0011, 0.0009, **0.0013**]