DM - Assignment 03

Problem 1

Part (1)

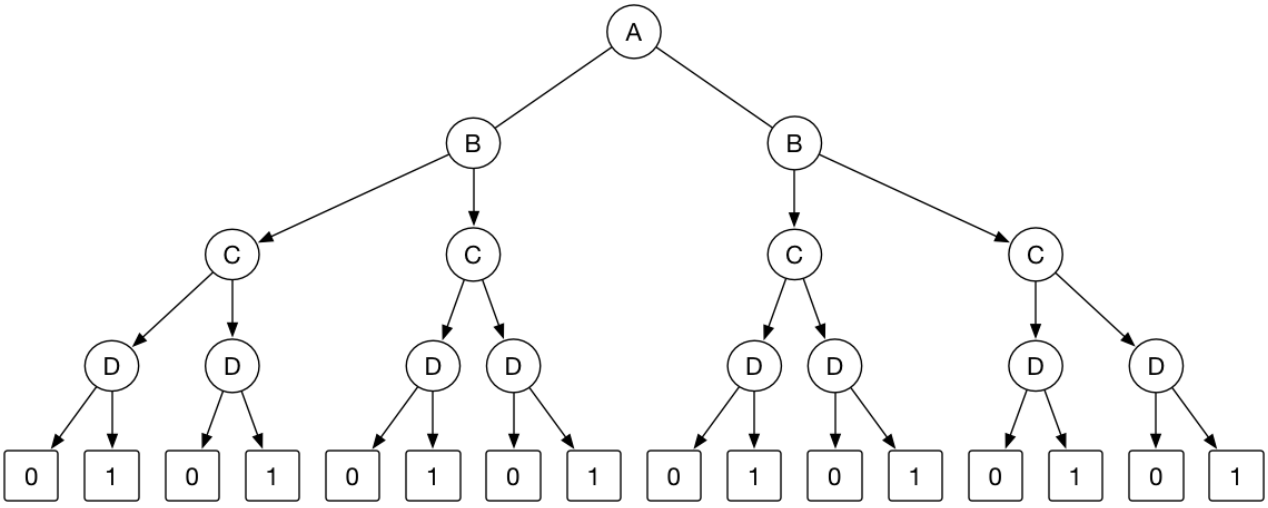
As split of node based on Information Gain with Gini Index favors equal-sized partitions with equal purity, the best strategy would be:

First, sort the samples on that attribute's value using some O(NlogN) method, such as quick sort or merge sort.

Then we have to compute N-1 splits to determine the optimal one; as each one of these is a binary split, so calculating its Gini Index & Info Gain will be O(N) time complexity. But we don't have to do O(N) for each split from small split value to larger ones, which is a O(N^2); instead, each time when we move the split value to a larger one in the sorted attribute array, the left subnode will include one more element, while the right subnode exclude that element, leading to one eccential fact that this modification in recalculating Gini Info Gain is O(1), and computing all N-1 splits to find the optimal is simply O(N).

So the overall performance could be still O(NlogN).

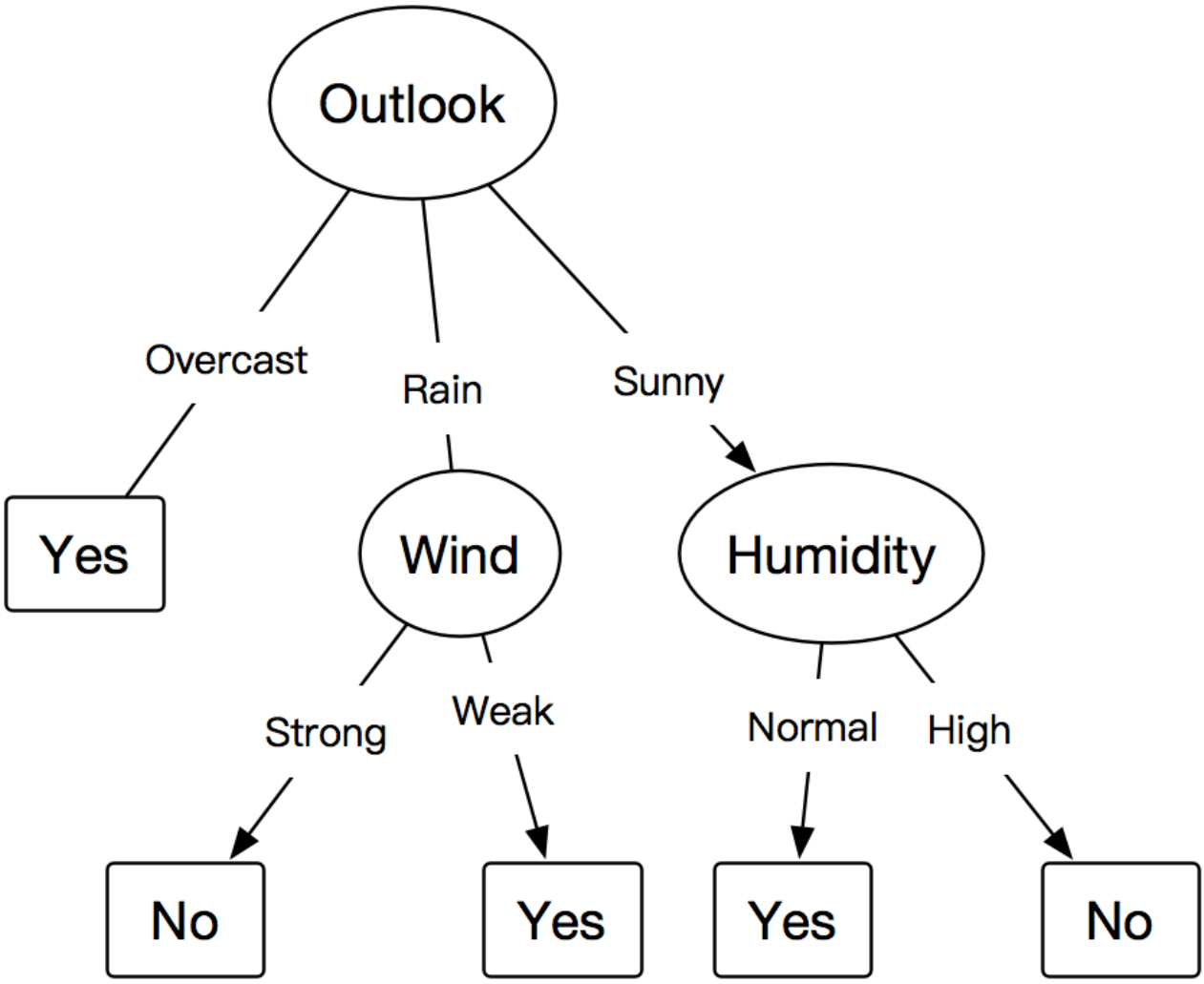
Part (2)



On every left path of tree node is labeled 1, and correspondingly the right paths are labeled 0.

The tree cannot be pruned without undermining its classification performance, because there aren't any pair of (binary) attributes that divide the original dataset into two subsets in an exact same way, meaning pruning any node will fail to separate some observations with different class labels.

Part (3)



Justifications for Each Attribute's Selection:

Outlook

Attr	Info_gain
Outlook	**0.1163**
Temperature	0.1139
Humidity	0.0918
Wind	0.0306

Wind

Attr	Info_gain
Temperature	0.0133
Humidity	0.0133
Wind	**0.48**

Humidity

Attr	Info_gain
Temperature	0.0133
Humidity	**0.48**
Wind	0.0133

Problem 2

Part (1)

attr_0	attr_1	class_label
1	0	1
1	0	0
1	0	0
1	0	1
1	0	1
0	1	1
0	0	1
1	0	1
1	1	1
0	1	0

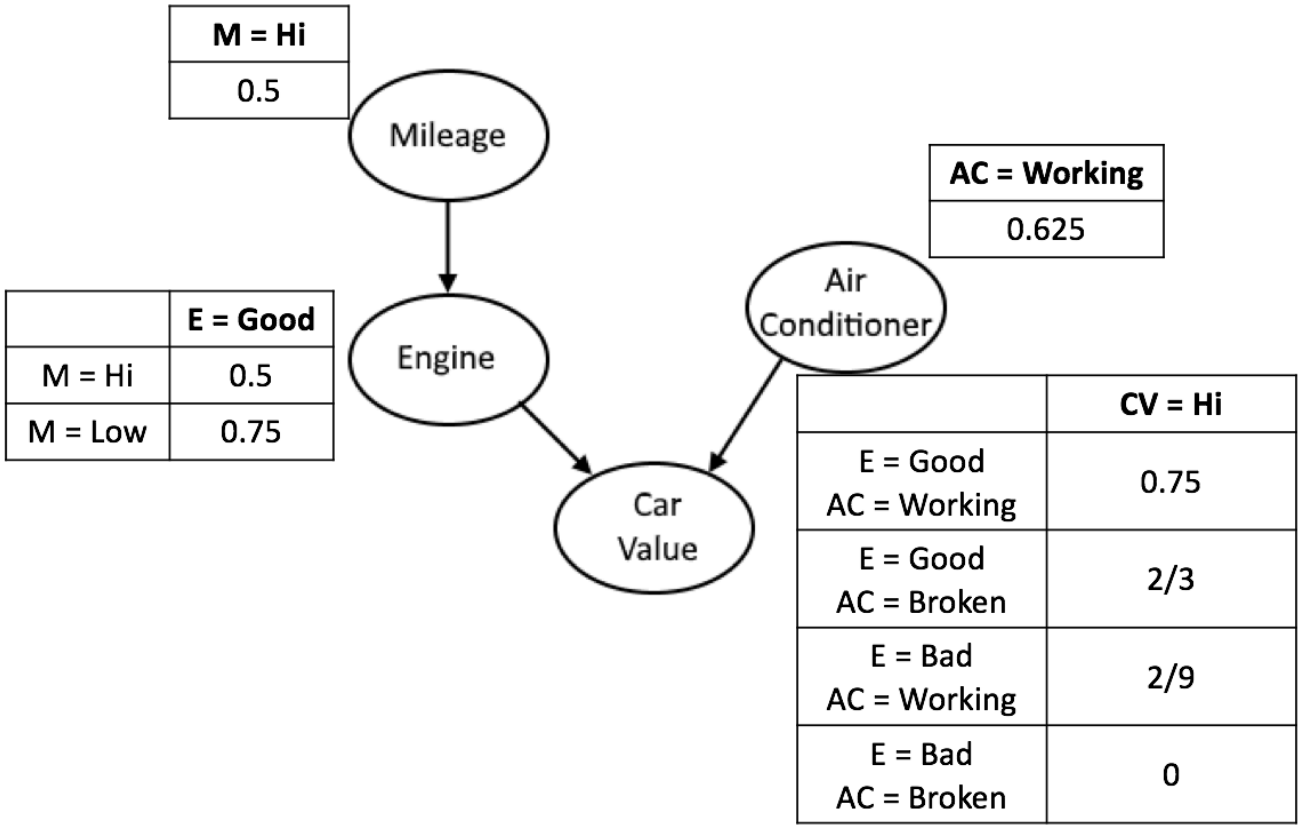
Based on the above test case, Entropy-based Info_gain will select attr_0 for division cuz its split gain is 0.0016177510177042276, slightly larger than attr_1's 0.0016177510177041721; while Gini-based Info_gain will select attr_1 for division cuz its split gain is 0.0009523809523810656, slightly larger than attr_1's 0.0009523809523810378.

Part (2)

The problem is to prove Info_gain(t, t1, t2, …) = Entropy(t) - $\sum_i$ (#pts(t_i) / #pts(t)) * Entropy(t_i) ≥ 0 always stands, for any node t that multi-splits into t_1, t_2, … . As Entropy(t) = -$\sum_c$ p(c|t)log(p(c|t)), if we first define f(x) = xlog(x), then the problem could be converted to proving f(c) ≤ $\sum_i$ (#pts(t_i) / #pts(t)) * f(c_i) for each c, where $\sum_i c_i$ = c (c_i stands for ratio of c class samples in sub-node t_i, and c stands for ratio of c class samples in father node t). This formula is in exact Jensen's inequality form, so it is easily proved correct.

Problem 3

Part (1)



Pr(E=Bad, AC=Broken) = Pr(E=Bad) * Pr(AC=Broken) = (Pr(E=Bad | M=Hi) * Pr(M=Hi) + Pr(E=Bad | M=Low) * Pr(M=Low)) * Pr(AC=Broken) = (0.5 * 0.5 * 0.25 * 0.5) * 0.375 = 0.140625.

Cuz given no evidence, E & AC are d-separate.

Part (2)

(1)

According to the formula of the joint distribution probability in Bayesian Belief Network:

```
Pr(B = good, F = empty, G = empty, S = yes)
= Pr(B = good, F = empty) * Pr(G = empty, S = yes | B = good, F = empty)
= Pr(B = good) * Pr(F = empty) *
Pr(G = empty | B = good, F = empty) * Pr(S = yes | B = good, F = empty)
= 0.9 * 0.2 * 0.8 *0.2 = 0.0288.
```

(2)

The same method with the part above:

```
Pr(B = bad, F = empty, G = not empty, S = no)
= Pr(B = bad) * Pr(F = empty) *
Pr(G = not empty | B = bad, F = empty) * Pr(S = no| B = bad, F = empty)
= 0.1 * 0.2 * 0.1 * 1 = 0.002.
```

(3)

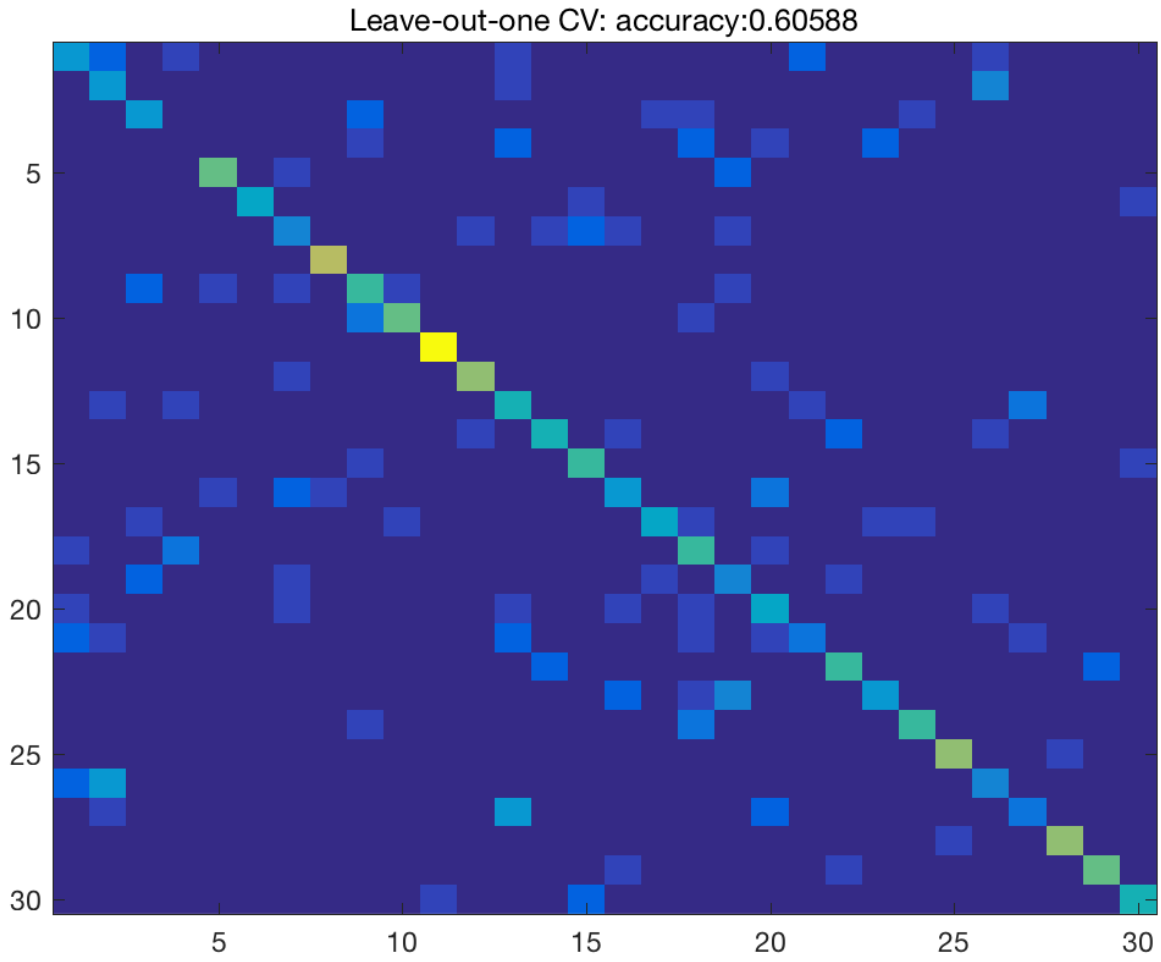
```
Pr(S = yes | B = bad)
= Pr(F = empty) * Pr(S = yes | B = bad, F = empty) +
Pr(F = not empty) * Pr(S = yes | B = bad, F = not empty)
= 0.2 * 0 + 0.8 * 0.1 = 0.08.
```

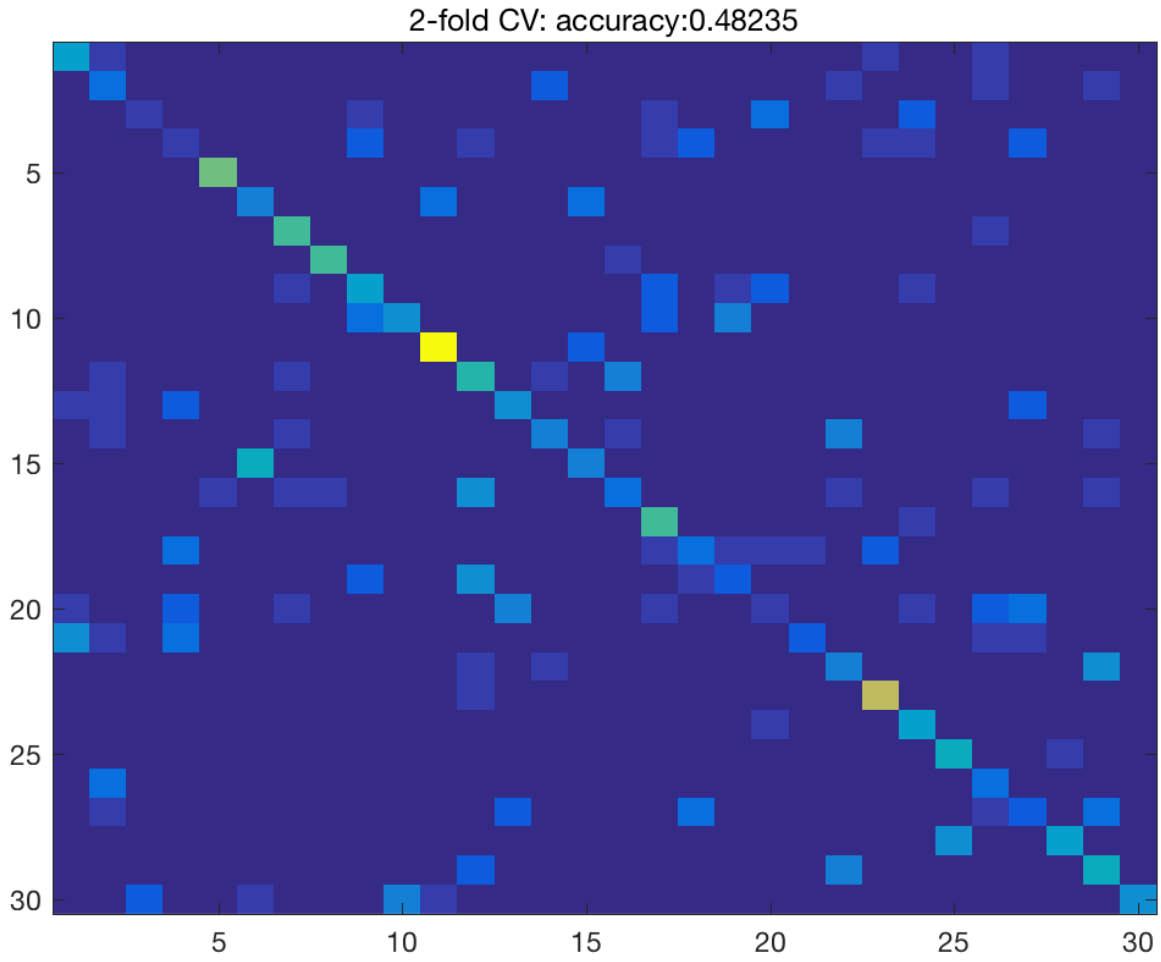
This induction is based on Total Probability Rule, which is $P(B) = \sum_i P(A_i)P(B|A_i)$.

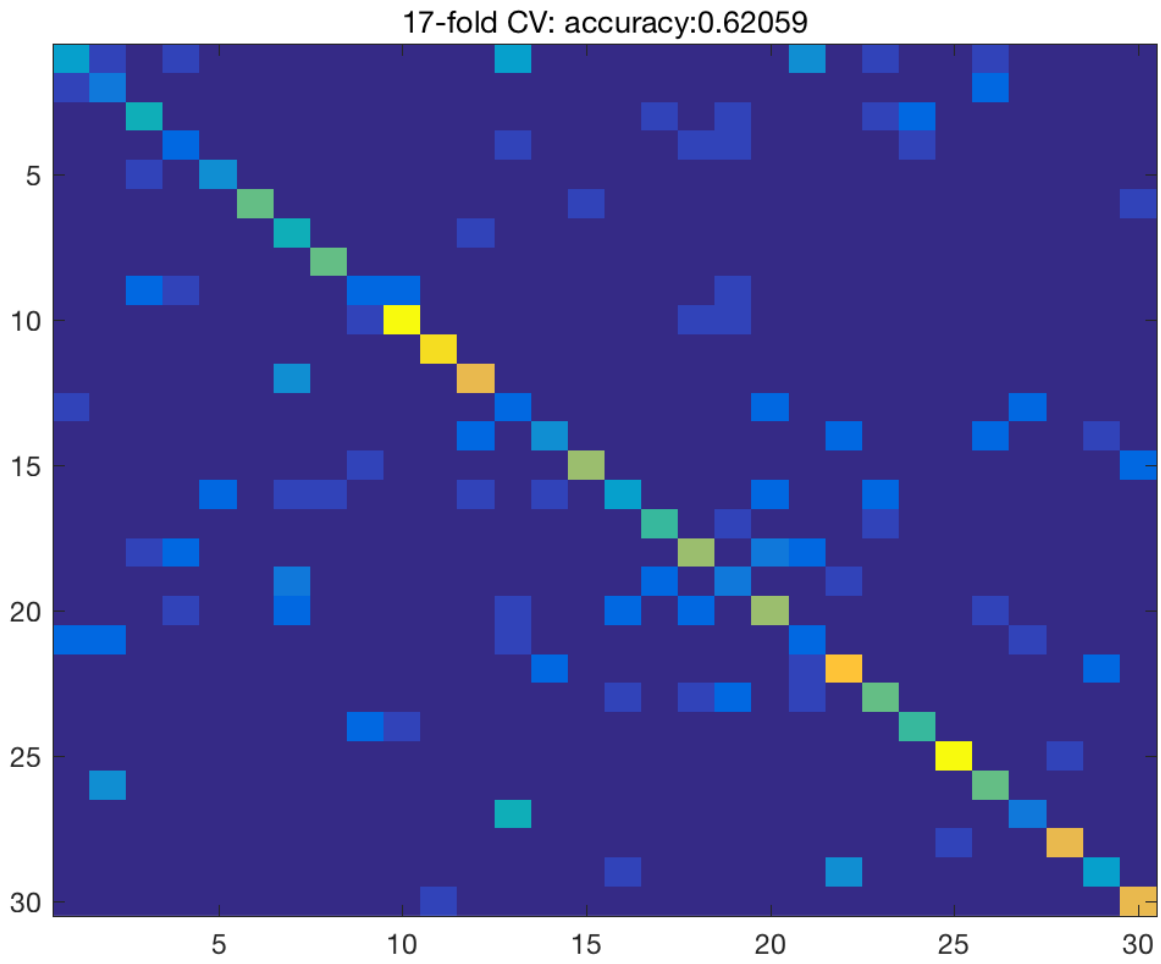
Problem 4

See script4.py included.

Problem 5


Leave-out-one CV: accuracy:0.60588
5
10
15
20
25
30
5
10
15
20
25
30


2-fold CV: accuracy:0.48235
5
10
15
20
25
30
5
10
15
20
25
30



For tree, see `dtree_view.png` as included.

Problem 6

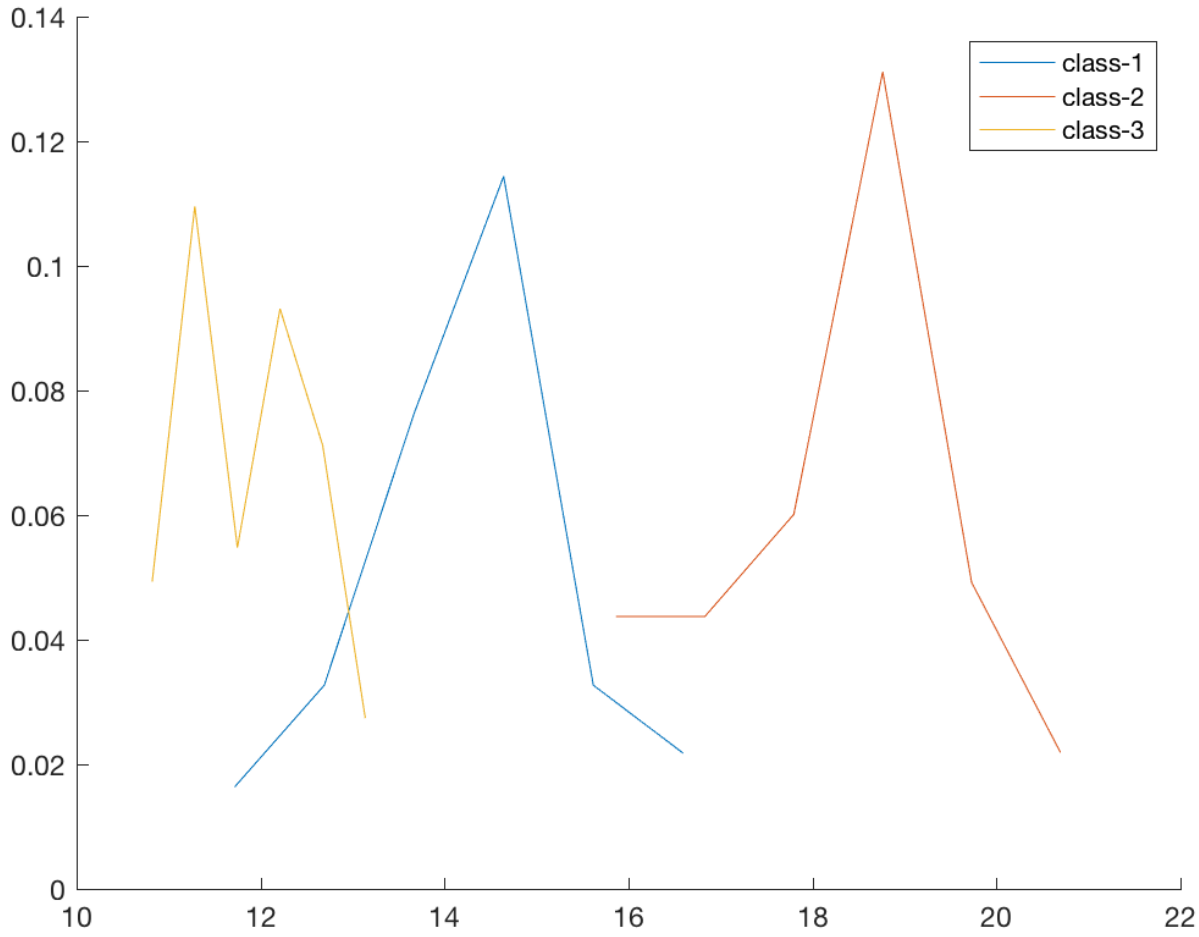
When setting random permutation seed to 0 (`rng(0)`):

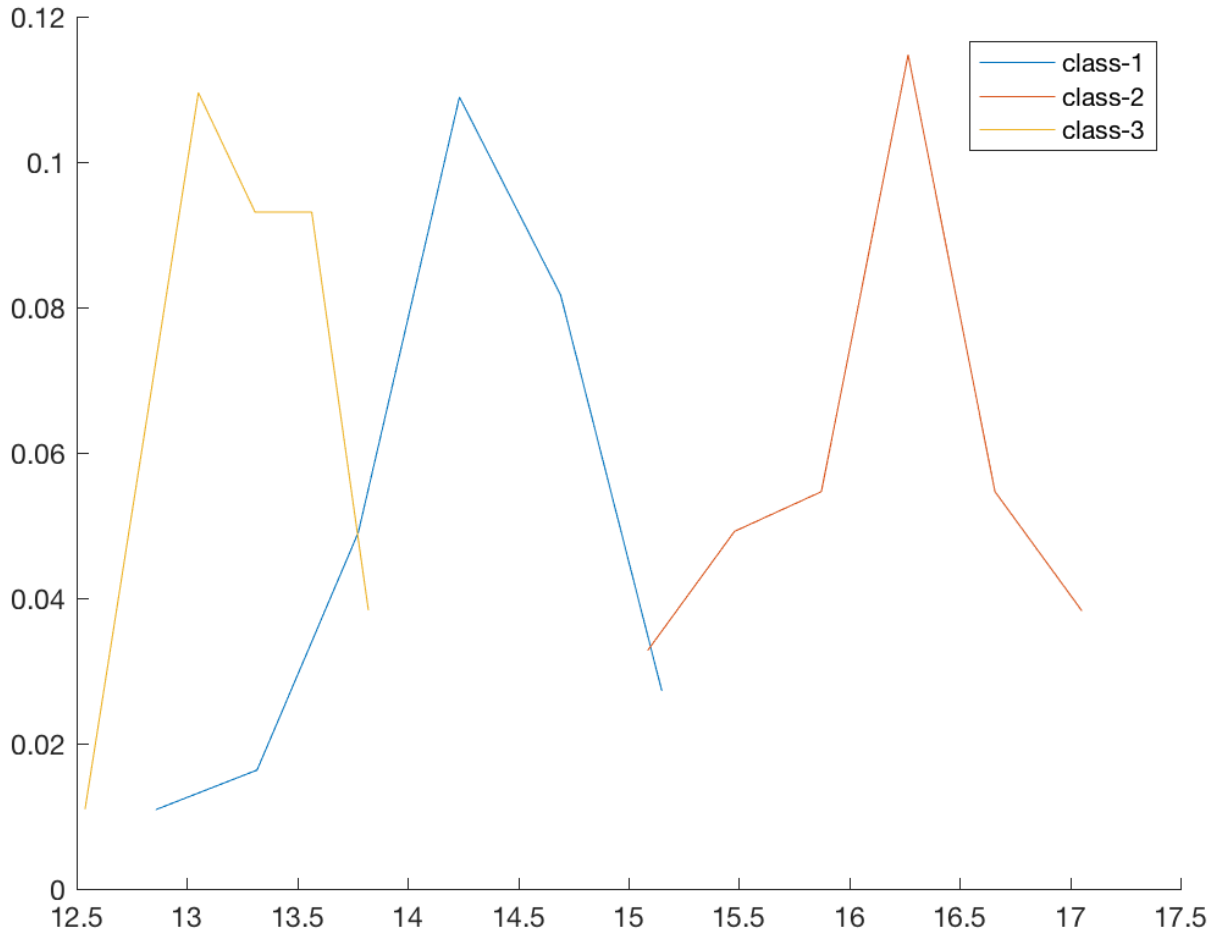
accuracy =

```
0.7333
```

confusion_matrix =

```
13     4    33
10    49     1
 0     0    70
```


0.14
0.12
0.1
0.08
0.06
0.04
0.02
0
10
12
14
16
18
20
22
class-1
class-2
class-3


0.12
0.1
0.08
0.06
0.04
0.02
0
12.5
13
13.5
14
14.5
15
15.5
16
16.5
17
17.5
class-1
class-2
class-3