Name:Tong ZHANG zID: z5235242

Q1

(1)

Location	Time	Item	SUM(Quantity)
Sydney	2005	PS2	1400
Sydney	2006	PS2	1500
Sydney	2006	Wii	500
Melbourne	2005	XBox 360	1700
Sydney	2005	ALL	1400
Sydney	2006	ALL	2000
Melbourne	2005	ALL	1700
Sydney	ALL	PS2	2900
Sydney	ALL	Wii	500
Melbourne	ALL	XBox 360	1700
ALL	2005	PS2	1400
ALL	2005	XBox 360	1700
ALL	2006	Wii	500
ALL	2006	PS2	1500
Sydney	ALL	ALL	3400
Melbourne	ALL	ALL	1700
ALL	ALL	PS2	2900
ALL	ALL	Wii	500
ALL	ALL	Xbox 360	1700
ALL	2005	ALL	3100
ALL	2006	ALL	2000
ALL	ALL	ALL	5100

(2)

SELECT Location, Time, Item, SUM(Quantity) FROM R GROUP BY Location, Time, Item
UNION ALL
SELECT Location, Time, ALL, SUM(Quantity) FROM R GROUP BY Location, Time
UNION ALL
SELECT Location, ALL, Item, SUM(Quantity) FROM R GROUP BY Location, Item
UNION ALL
SELECT ALL, Time, Item, SUM(Quantity) FROM R GROUP BY Time, Item
UNION ALL
SELECT Location, ALL, ALL, SUM(Quantity) FROM R GROUP BY Location
UNION ALL
SELECT ALL, ALL, Item, SUM(Quantity) FROM R GROUP BY Item
UNION ALL
SELECT ALL, Time, ALL, SUM(Quantity) FROM R GROUP BY Time
UNION ALL
SELECT ALL, ALL, ALL, SUM(Quantity) FROM R

(3)

Location	Time	Item	SUM(Quantity)
Sydney	2006	ALL	2000
Sydney	ALL	PS2	2900
Sydney	ALL	ALL	3400
ALL	2005	ALL	3100
ALL	2006	ALL	2000
ALL	ALL	PS2	2900
ALL	ALL	ALL	5100

(4)

The second function is feasible. This is because that the function should ensure that the original data would still be recovered after the mapping, which means the function should be an injective function(one to one).
Based on (1) and the first mapping functions, we have

Location	Time	Item	SUM(Quantity)	Offset
1	1	1	1400	21
1	2	1	1500	25
1	2	3	500	27
2	1	2	1700	38
1	1	0	1400	20
1	2	0	2000	24
2	1	0	1700	36
1	0	1	2900	17
1	0	3	500	19
2	0	2	1700	34
0	1	1	1400	5
0	1	2	1700	6
0	2	3	500	11
0	2	1	1500	9
1	0	0	3400	16
2	0	0	1700	32
0	0	1	2900	1
0	0	3	500	3
0	0	2	1700	2
0	1	0	3100	4
0	2	0	2000	8
0	0	0	5100	0

Then we can draw the MOLAP cube (sorted by the index(offset) in an ascending order):

Index	SUM(Quantity)
0	5100
1	2900
2	1700
3	500
4	3100
5	1400
6	1700
8	2000
9	1500
11	500
16	3400
17	2900
19	500
20	1400
21	1400
24	2000
25	1500
27	500
32	1700
34	1700
36	1700
38	1700

It is shown that the function is injective. If we use the first function, it would be seen that the function is not injective.

Q2

(1)

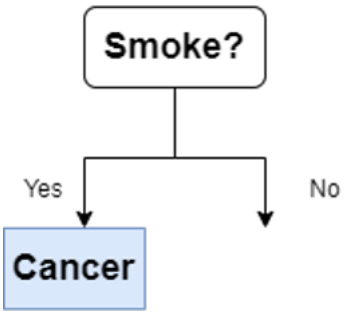
$$Gini(whole\_data) = 1 - \left(\frac{4}{6}\right)^2 - \left(\frac{2}{6}\right)^2 = 0.44$$

There are four features (`Gender, Smokes?, Chest pain?, Cough?`) in total.

$$Gini_{split}\left(Gender\right) = \frac{4}{6}Gini\left(3,1\right) + \frac{2}{6}Gini\left(1,1\right) = 0.42$$

$$Gini_{split}(Smoke) = \frac{3}{6}Gini(3,0) + \frac{3}{6}Gini(1,2) = 0.22$$
$$Gini_{split}(Chest\ pain) = \frac{4}{6}Gini(2,2) + \frac{2}{6}Gini(2,0) = 0.33$$
$$Gini_{split}(Cough) = \frac{4}{6}Gini(2,2) + \frac{2}{6}Gini(2,0) = 0.33$$

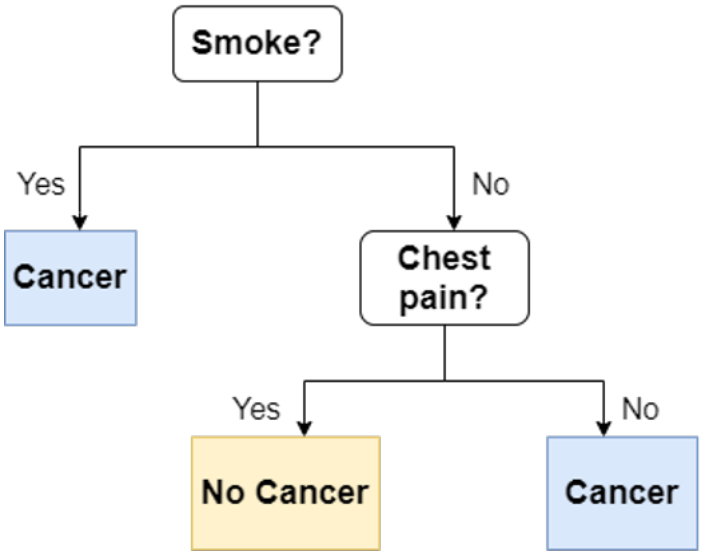
Thus **'Smoke'** would be chosen to split the node.



Then,

$$Gini_{split}(smoke = No, Gender) = \frac{2}{3}Gini(1,1) + \frac{1}{3}Gini(0,1) = 0.33$$
$$Gini_{split}(smoke = No, Chest\ pain) = \frac{2}{3}Gini(0,2) + \frac{1}{3}Gini(1,0) = 0$$

$Gini_{split}(smoke = No, Chest\ pain) = 0$, means that whether has a lung cancer could be concluded at this stage. So the decision tree could be constructed:



(2)

If the patient smokes, then he/she has lung cancer.
If the patient does not smoke and has chest pain, then he/she does not have lung cancer.
If the patient does not smoke and does not have chest pain, then he/she has lung cancer.

Q3

(1)

$x = (x_1, x_2, \ldots, x_d)$

$C = \{C_0 : y = 0,\ C_1 : y = 1\}$

$$P\left(C_j | x_1, x_2, \ldots, x_n\right) = \frac{P\left(x_1, x_2, \ldots, x_n, C_j\right)}{P\left(x\right)} = \frac{P\left(C_j\right) P\left(x_1, x_2, \ldots, x_d | C_j\right)}{P\left(x\right)}$$

According the assumption for Native Bayes(conditional independence), we have

$$P\left(x_1, x_2, \ldots, x_d | C_j\right) = \prod_{i=1}^{d} P\left(x_i | C_j\right)$$

Then we have

$$P\left(C_j | x_1, x_2, \ldots, x_d\right) = \frac{P\left(C_j\right) \prod_{i=1}^{d} P\left(x_i | C_j\right)}{P\left(x\right)} \qquad (1)$$

Since x is a binary vector, we have

$$\prod_{i=1}^{d} P\left(x_i | Cj\right) = \prod_{i=1}^{d} P_{ji}^{x_i} \left(1 - P_{ji}\right)^{1-x_i} \text{ , where } P_{ji} = P\left(x_i = 1 | C_j\right) \qquad (2)$$

Based on (1) and (2), we have

$$P\left(C_j | x_1, x_2, \ldots, x_d\right) \propto P\left(C_j\right) \prod_{i=1}^{d} P\left(x_i | C_j\right) = P\left(C_j\right) \prod_{i=1}^{d} P_{ji}^{x_i} \left(1 - P_{ji}\right)^{1-x_i} \qquad (3)$$

$$log\ (3) = logP\left(C_j\right) + \sum_{i=1}^{d} x_i log P_{ji} + \sum_{i=1}^{d} \left(1 - x_i\right) log\left(1 - P_{ji}\right)$$

$$= \sum_{i=1}^{d} x_i \left(lopP_{ji} - log\left(1 - P_{ji}\right)\right) + \sum_{i=1}^{d} log\left(1 - P_{ji}\right) + logP\left(C_j\right) \text{ , where } P_{ji} = P\left(x_i = 1 | C_j\right)$$

So the Native Bayes classifier is

$$argmax_{C_j \in C} \left[\sum_{i=1}^{d} x_i \left(lopP_{ji} - log\left(1 - P_{ji}\right)\right) + \sum_{i=1}^{d} log\left(1 - P_{ji}\right) + logP\left(C_j\right)\right], \qquad (4)$$

$$\text{where } P_{ji} = P\left(x_i = 1 | C_j\right)$$

Obviously, (4) = $w_j^T x + b$, where

$$w = logP_{ji} - log\left(1 - P_{ji}\right),\ b = \sum_{i=1}^{d} log\left(1 - P_{ji}\right) + lopP\left(C_j\right),$$

Thus the Native Bayes classifier is a linear classifier in a d+1 – dimension space.

(2)

Learning w_{NB} is much easier than learning w_{LR}, this mainly results from the assumption in Native Bayes Inference, i.e. attributes are conditionally independent. This means there is no dependence relation between attributes given the class, which reduces the computation cost, since we only need to estimate $P\left(x_k \mid C_j\right)$.

Compared with Native Bayes, Linear Regression updates the weight by gradient descent in general, the cost of which tends to be much greater.