Assignment

COMP9318

By: Arth Sanskar Patel
z5228942

Question 1

(1) The table is as follows

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

(2) Assuming table is called ‘Sales’, then Equivalent SQL Query to compute the cube without ‘CUBE BY’ clause:

```
SELECT Location, Time, Item, SUM(Quantity)
FROM Sales
GROUP BY Location, Time, Item

UNION ALL

SELECT Location, Time, NULL, SUM(Quantity)
FROM Sales
GROUP BY Location, Time

UNION ALL

SELECT Location, NULL, Item, SUM(Quantity)
FROM Sales
GROUP BY Location, Item

UNION ALL

SELECT NULL, Time, Item, SUM(Quantity)
FROM Sales
GROUP BY Time, Item

UNION ALL

SELECT Location, NULL, NULL, SUM(Quantity)
FROM Sales
GROUP BY Location

UNION ALL

SELECT NULL, Time, NULL, SUM(Quantity)
FROM Sales
GROUP BY Time

UNION ALL

SELECT NULL, NULL, Item, SUM(Quantity)
FROM Sales
GROUP BY Item

UNION ALL

SELECT NULL, NULL, NULL, SUM(Quantity)
FROM Sales
```

(3) Output of the given ice-berg cube query in part 3 of question 1 is:

Location	Time	Item	Quantity
NULL	NULL	PS2	2900
Sydney	NULL	PS2	2900
NULL	NULL	NULL	5100
Sydney	NULL	NULL	3400
NULL	2005	NULL	3100
Sydney	2006	NULL	2000
NULL	2006	NULL	2000

(4) We have the functions as follows:
Function 1: $f(x) = 9 * f_{Location} + 3 * f_{Time} + f_{Item}$
Function 2: $f(x) = 16 * f_{Location} + 4 * f_{Time} + f_{Item}$

As seen in the table below, Function 1 causes some duplicate values for different tuples (Highlighted in yellow and orange). The idea is to have a function which does a 1 to 1 mapping. Because of the duplicate values in the generated offset, the cross back to the stored database will return 2 different tuples for the same offset value. To avoid this, we should choose the function which has 1 to 1 mapping and hence no repeating values of offset. Function 2 ($f(x) = 16 * f_{Location} + 4 * f_{Time} + f_{Item}$) is the appropriate choice and more feasible.

Location	Time	Item	Quantity	Function 2	Function 1
1	1	1	1400	21	13
1	1	0	1400	20	12
1	2	1	1500	25	16
1	2	3	500	27	18
1	2	0	2000	24	15
1	0	1	2900	17	10
1	0	3	500	19	12
1	0	0	3400	16	9
2	1	2	1700	38	23
2	1	0	1700	36	21
2	0	2	1700	34	20
2	0	0	1700	32	18
0	1	1	1400	5	4
0	2	1	1500	9	7
0	0	1	2900	1	1
0	2	3	500	11	9
0	0	3	500	3	3
0	1	2	1700	6	5
0	0	2	1700	2	2
0	1	0	3100	4	3
0	2	0	2000	8	6
0	0	0	5100	0	0

Question 2

(1) In this we will use Gini Index as a cost function to evaluate the splits for the dataset. The lower Gini Index for the attributes will be chosen before the attributes with higher Gini Index. We have the table as follows:

Patient ID	Gender	Smoke	Chest Pain	Cough	Lung Cancer
1	Female	Yes	Yes	Yes	Yes
2	Male	Yes	No	Yes	Yes
3	Male	No	No	No	Yes
4	Female	No	Yes	Yes	No
5	Male	Yes	Yes	No	Yes
6	Male	No	Yes	Yes	No

In this data set there are 4 Attributes (Gender, Smoke, Chest Pain and Cough) from which, it is predicted the attribute Lung Cancer. The Attribute Lung cancer contains 2 (Yes and No) classes. All the other attributes contain 2 classes as follows:

Gender	Smoke	Chest Pain	Cough
Male	Yes	Yes	Yes
Female	No	No	No

Lets first calculate Gini Index for all the attributes.

a. Gender:

Distribution for Gender and Lung Cancer		Gender	
		Male	Female
Lung Cancer	Yes	3	1
	No	1	1
Total		**4**	**2**

Male = 4

For Male and Lung Cancer = Yes: $3/4$
For Male and Lung Cancer = No: $1/4$

Gini (3, 1) = $1 - ((3/4)^2 + (1/4)^2)$ = 0.375

Female = 2

For Female and Lung Cancer = Yes: $1/2$
For Female and Lung Cancer = No: $1/2$

Gini (1, 1) = $1 - ((1/2)^2 + (1/2)^2)$ = 0.5

By Adding weight and sum of each Gini indices:

Gini (Target, Gender) = $(4/6) * (0.375) + (2/6) * (0.5)$ = 0.4167

b. Smoke:

Distribution for Smoke and Lung Cancer		Smoke	
		Yes	No
Lung Cancer	Yes	3	1
	No	0	2
Total		**3**	**3**

Yes = 3

For Yes and Lung Cancer = Yes: $3/3$
For Yes and Lung Cancer = No: $0/3$

Gini (3, 0) = $1 - ((3/3)^2 + (0/3)^2)$ = 0

No = 3

For No and Lung Cancer = Yes: $1/3$

For No and Lung Cancer = No: $2/3$

Gini (1, 2) = $1 - ((1/3)^2 + (2/3)^2)$ = 0.4444

By Adding weight and sum of each Gini indices:

Gini (Target, Smoke) = $(3/6) * (0) + (3/6) * (0.4444)$ = 0.2222

c. Chest Pain

Distribution for Chest Pain and Lung Cancer		Chest Pain	
		Yes	No
Lung Cancer	Yes	2	2
	No	2	0
Total		4	2

Yes = 4

For Yes and Lung Cancer = Yes: $2/4$
For Yes and Lung Cancer = No: $2/4$

Gini (2, 2) = $1 - ((2/4)^2 + (2/4)^2)$ = 0.5

No = 2

For No and Lung Cancer = Yes: $2/2$

For No and Lung Cancer = No: $0/2$

Gini (2, 0) = $1 - ((2/2)^2 + (0/2)^2)$ = 0

By Adding weight and sum of each Gini indices:

Gini (Target, Chest Pain) = $(4/6) * (0.5) + (2/6) * (0)$ = 0.3333

d. Cough

Distribution for Cough and Lung Cancer		Cough	
		Yes	No
Lung Cancer	Yes	2	2
	No	2	0
Total		4	2

Yes = 4

For Yes and Lung Cancer = Yes: $2/4$
For Yes and Lung Cancer = No: $2/4$

Gini (2, 2) = $1 - ((2/4)^2 + (2/4)^2)$ = 0.5

No = 2

For No and Lung Cancer = Yes: $2/2$

For No and Lung Cancer = No: $0/2$

Gini (2, 0) = $1 - ((2/2)^2 + (0/2)^2)$ = 0

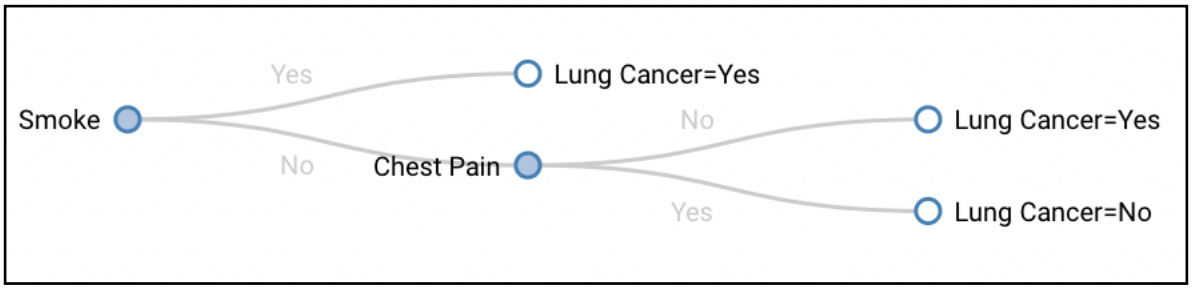
By Adding weight and sum of each Gini indices:

Gini (Target, Cough) = $(4/6) * (0.5) + (2/6) * (0)$ = 0.3333

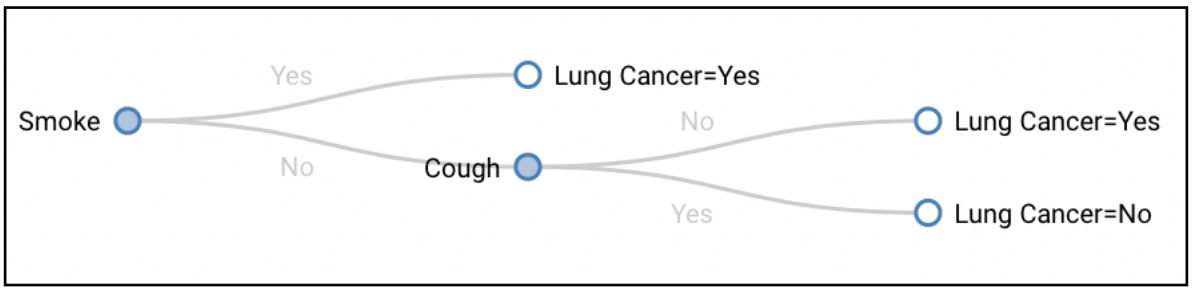
So we have the Gini Indices for the attributes as follows:

Attribute	Gini Index
Gender	0.4167
Smoke	0.2222
Chest Pain	0.3333
Cough	0.3333

As clearly obvious from the table above, the lowest Gini Index is for ‘Smoke’ attribute. So we will choose it as the first split. After that ‘Chest Pain’ and ‘Cough’ have the same gini indices so any of them can be chosen for second spilt. The final decision tree will look as follows:



If we replace the ‘Chest Pain’ with ‘Cough’ in the above tree, the tree will remain exactly same because they have the exact same gini indices. The Tree with ‘Cough’ attribute instead of ‘Chest Pain’ will look like this:



(2) The decision rules from the decision tree above will be:

IF *Smoke* = 'Yes' **THEN** *Lung Cancer* = 'Yes'
IF *Smoke* = 'No' **AND** *Chest Pain* = 'Yes' **THEN** *Lung Cancer* = 'No'
IF *Smoke* = 'No' **AND** *Chest Pain* = 'No' **THEN** *Lung Cancer* = 'Yes'

If we prefer to use the 'Cough' attribute instead of 'Chest Pain', the decision rules will be:

IF *Smoke* = 'Yes' **THEN** *Lung Cancer* = 'Yes'
IF *Smoke* = 'No' **AND** *Cough* = 'Yes' **THEN** *Lung Cancer* = 'No'
IF *Smoke* = 'No' **AND** *Cough* = 'No' **THEN** *Lung Cancer* = 'Yes'

Question 3

(1) To prove that Naive Bayes classifier can be transformed into linear classifier, let's first state the NB Classifier given in the question.

Bayes rule is as follows:

$$P(y = n \mid \overrightarrow{X}) = \frac{P(\overrightarrow{X} \mid y = n) * P(y = n)}{P(\overrightarrow{X})}$$

Here 'n' is the class (In our case since it's a binary classification, 'n' is either 1 or 0) and $\overrightarrow{X} = \langle x_1, x_2, x_3 \ldots x_d \rangle$ where 'd' is the number of dimensions of feature vectors.

So according to Naive Bayes classifier, we have:

$$NB(\overrightarrow{X}) = \begin{cases} 1 & \text{if } \frac{P(y = 1 \mid \overrightarrow{X})}{P(y = 0 \mid \overrightarrow{X})} \geq 1 \\ 0 & \text{if } \frac{P(y = 1 \mid \overrightarrow{X})}{P(y = 0 \mid \overrightarrow{X})} < 1 \end{cases} \qquad \text{(EQ 1)}$$

Lets start by simplifying the $\frac{P(y = 1 \mid \overrightarrow{X})}{P(y = 0 \mid \overrightarrow{X})}$ part of the classifier above.

According to Naive Bayes, we have:

$$\frac{P(y = 1 \mid \overrightarrow{X})}{P(y = 0 \mid \overrightarrow{X})} = \frac{P(y = 1) * P(\overrightarrow{X} \mid y = 1)}{P(y = 0) * P(\overrightarrow{X} \mid y = 0)}$$

$$\frac{P(y = 1 \mid \overrightarrow{X})}{P(y = 0 \mid \overrightarrow{X})} = \frac{P(y = 1) \prod_{i=1}^{d} P(x_i \mid y = 1)}{P(y = 0) \prod_{i=1}^{d} P(x_i \mid y = 0)}$$

$$\frac{P(y = 1 \mid \overrightarrow{X})}{P(y = 0 \mid \overrightarrow{X})} = \frac{P(y = 1)}{P(y = 0)} \prod_{i=1}^{d} \frac{P(x_i \mid y = 1)}{P(x_i \mid y = 0)} \qquad \text{(EQ 2)}$$

Lets take $p = P(y = 1)$ then $1 - p = P(y = 0)$ since $\{P(y = 1) + P(y = 0) = 1\}$
By similar logic we have:
$a_i = P(x_i = 1 \mid y = 1)$ and $1 - a_i = P(x_i = 0 \mid y = 1)$
$b_i = P(x_i = 1 \mid y = 0)$ and $1 - b_i = P(x_i = 0 \mid y = 0)$

From the equations above we have:

$P(x_i \mid y = 1) = a_i^{x_i}(1 - a_i)^{(1-x_i)}$, and
$P(x_i \mid y = 0) = b_i^{x_i}(1 - b_i)^{(1-x_i)}$

Plugging the values we have into EQ 2, we get

$$\frac{P(y = 1 \mid \vec{X})}{P(y = 0 \mid \vec{X})} = \frac{p}{1-p} \prod_{i=1}^{d} \frac{a_i^{x_i}(1 - a_i)^{(1-x_i)}}{b_i^{x_i}(1 - b_i)^{(1-x_i)}}$$

$$\frac{P(y = 1 \mid \vec{X})}{P(y = 0 \mid \vec{X})} = \frac{p}{1-p} \prod_{i=1}^{d} \frac{a_i^{x_i}(1 - b_i)^{x_i}(1 - a_i)}{b_i^{x_i}(1 - a_i)^{x_i}(1 - b_i)}$$

Taking 'log' on both sides, we get:

$$\log \frac{P(y = 1 \mid \vec{X})}{P(y = 0 \mid \vec{X})} = \log \frac{p}{1-p} \prod_{i=1}^{d} \frac{a_i^{x_i}(1 - b_i)^{x_i}(1 - a_i)}{b_i^{x_i}(1 - a_i)^{x_i}(1 - b_i)}$$

$$\log \frac{P(y = 1 \mid \vec{X})}{P(y = 0 \mid \vec{X})} = \log \frac{p}{1-p} + \sum_{i=1}^{d} \log \frac{1 - a_i}{1 - b_i} + \sum_{i=1}^{d} x_i \log \frac{a_i(1 - b_i)}{b_i(1 - a_i)}$$

Here let $c = \log \frac{p}{1-p} + \sum_{i=1}^{d} \log \frac{1 - a_i}{1 - b_i}$

Let $\vec{w} = \begin{pmatrix} w_1 \\ w_2 \\ w_3 \\ \cdots \\ w_d \end{pmatrix}$ where $w_i = \log \frac{a_i(1 - b_i)}{b_i(1 - a_i)}$

Therefore taking $\vec{x} = \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ \cdots \\ x_d \end{pmatrix}$ we have

$$\log \frac{P(y = 1 \mid \vec{X})}{P(y = 0 \mid \vec{X})} = c + \vec{w}^T \vec{x}$$

This is basically a linear classifier in d + 1-dimension space.

Putting everything in EQ 1 we have

$$NB(\overrightarrow{X}) = \begin{cases} 1 & \text{if } c + \overrightarrow{w}^T \overrightarrow{x} \geq \log 1 \\ 0 & \text{if } c + \overrightarrow{w}^T \overrightarrow{x} < \log 1 \end{cases}$$

Or our final Naive Bayes classifier transformed to Linear Classifier will look like:

$$NB(\overrightarrow{X}) = \begin{cases} 1 & \text{if } c + \overrightarrow{w}^T \overrightarrow{x} \geq 0 \\ 0 & otherwise \end{cases}$$

(2) The reason Naive Bayes learning ($\overrightarrow{w}_{NB}$) is much easier than Logistic regression learning ($\overrightarrow{w}_{LR}$) is due to the following reasons:

a. Naive Bayes is much faster to train. It reaches the asymptotic faster when compared with Logistic Regression. Generative Models (Naive Bayes) have typical time complexity of $O(\log(n))$ compared to $O(n)$ of discriminative models (Logistic regression). This was proved by Professors Andrew Ng and Michael I Jordan when they performed the comparison on random 15 datasets from the UCI ML repository.

b. Naive Bayes has a simple prediction process which relies on using trained values in equations like $\frac{P(y=1 \mid \overrightarrow{X})}{P(y=0 \mid \overrightarrow{X})} = \frac{P(y=1)}{P(y=0)} \prod_{i=1}^{d} \frac{P(x_i \mid y=1)}{P(x_i \mid y=0)}$. On the other hand, Logistic Regression uses complex methods like Gradient Descent which internally have complexities of order $O(kn^2)$ which complicates the whole process.

Due to the above mentioned reasons, $\overrightarrow{w}_{NB}$ is much easier than $\overrightarrow{w}_{LR}$.