Problem 2: Some practice on naïve Bayes classifier (discrete variables)

Pertinent readings for Problem #2:

Kubat: Introduction to machine learning (2nd ed)

Blackboard location: /Resources / Our main textbooks for this class

Ch. 2: Pages 19 - 27 (naïve Bayes classifier, discrete variables)

Page 26: The big example that you should really look at ! =)

Your homework tasks: (turn in the parts highlighted in yellow

Part 1: Practice coding things with categorical variables:

Let's revisit the "tissue biopsy" example that we did in class, where we worked on the data presented in Table 1:

A) Your dataset: 12 biopsies Total ($N = 12$)

Instance	$\vec{x}_1$ shape	$\vec{x}_2$ Radius	$\vec{x}_3$ Concavity	$\vec{x}_4$ Texture	$\vec{x}_5$ Color	Class label (malignant or not?)	
1	Circle	L	convex	Smooth	Dark	+	1
2		L	concave		Dark	+	1
3		L	flat		Red	+	1
4	Irregular	S	Concave	Rough	Dark	+	1
5	Circle	L	flat		Neutral	+	1
6		L	Concave		Dark	+	1
7		L	convex	Smooth	Neutral	−	0
8	Irregular	L	concave		Red	−	0
9	Triangle	S	convex	Rough	Dark	−	0
10		L	flat	smooth	Neutral	−	0
11	Irregular	L	concave		Dark	−	0
12	Irregular	L	concave		Red	−	0

malignant

Benign

Table 1: The tissue biopsy example that we did in class !

And as always, we aim to code a machine learning (ML) box so that:

1) Given a new biopsy sample with input attributes $x_1, x_2,\ x_3,\ x_4,$ and x_5

2) Our ML box can try to predict whether our new biopsy sample was either:

   a) Malignant (numerical output value = 1 , Class C_1), or
   b) Benign (numerical output value = 0 , Class C_2)

The overall scheme is depicted in Figure 2 below:

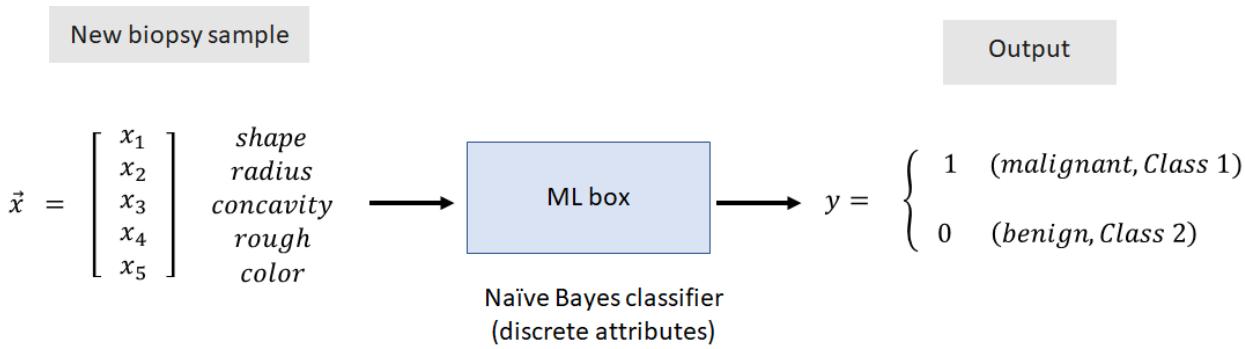

Figure 2: The ML box that we're gonna revisit in Part 1 of this problem =)

Your tasks for Part A: Previewing + reading in data files

1) Using matlab's *preview* and *readtable* functions , read in our biopsy training data file called "biopsy_data_missing_values.csv." You might want to try what we did in our Friday recitations (type in the lines below and check out how it works):

```
% -- Preview the file using the "detectImportOptions" command

opts = detectImportOptions('biopsy_data_missing_values.csv', 'NumHeaderLines', 1);
preview('biopsy_data_missing_values.csv', opts)

input('This is a preview of the CSV file..  press enter to continue !')


% -- Now, we will read in the table for real !

A = readtable('biopsy_data_missing_values.csv', 'NumHeaderLines', 1);
```

Dealing with missing values in tables:

If you echo your matrix "A," you will notice the "irregular" shapes are missing in column A.Var2….. =(And also, in the first column (biopsy ID number), we have missing entries in samples 6 and 7 (they're NaN's right now).

When you have missing values, matlab will fill them in as "empty" spaces, or '' (2 single quotes, no spaces in between them)… or if they are floating point numbers, they will be "NaNs" (not a number)

```
A =
 12×7 table
  Var1      Var2          Var3        Var4          Var5          Var6          Var7
  ____   ____________   _________   ___________   __________   ___________   ____________
    1    {'Circle' }    {'Large'}   {'Convex' }   {'Smooth'}   {'Dark'  }    {'Malignant'}
    2    {'Circle' }    {'Large'}   {'Concave'}   {'Smooth'}   {'Dark'  }    {'Malignant'}
    3    {'Circle' }    {'Large'}   {'Flat'  }    {'Smooth'}   {'Red'   }    {'Malignant'}
    4    {0×0 char }    {'Small'}   {'Concave'}   {'Rough' }   {'Dark'  }    {'Malignant'}
    5    {'Circle' }    {'Large'}   {'Flat'  }    {'Rough' }   {'Neutral'}   {'Malignant'}
  NaN    {'Circle' }    {'Large'}   {'Concave'}   {'Rough' }   {'Dark'  }    {'Malignant'}
    7    {'Circle' }    {'Large'}   {'Convex' }   {'Smooth'}   {'Neutral'}   {'Benign'  }
    8    {0×0 char }    {'Large'}   {'Concave'}   {'Smooth'}   {'Red'   }    {'Benign'  }
    9    {'Triangle'}   {'Small'}   {'Convex' }   {'Rough' }   {'Dark'  }    {'Benign'  }
  NaN    {'Circle' }    {'Large'}   {'Flat'  }    {'Smooth'}   {'Neutral'}   {'Benign'  }
   11    {0×0 char }    {'Large'}   {'Concave'}   {'Smooth'}   {'Dark'  }    {'Benign'  }
   12    {0×0 char }    {'Large'}   {'Concave'}   {'Smooth'}   {'Red'   }    {'Benign'  }
```

In order to detect whether you have missing values in your data columns, you have to use the *strcmp* function (string comparison) to see where they are ! For instance, type the following lines and see what you get:

```
% -- Compare the "circle" string in column A.Var2(1)
strcmp(A.Var2(1), 'Circle')

% -- Now try these 2 lines…
strcmp(A.Var2(4), 'Circle')
strcmp(A.Var2(4), '')              % -- No spaces between the quotation marks (empty string)  !!

% -- Now, try this line and see what you get  =)
find(strcmp(A.Var2, 'Circle'))

% -- If you want to replace an existing string, you can do the following (notice the BRACES { 1 }!!)
A.Var2{1} = 'omg_what_the_hell'

% -- Finally:  If you want to search for NaN's, you need to ask if the number is a "finite" number or not !
%    Notice we're trying to find the non-finite entries  (  the   ~   operator  =  logical NOT)
my_missing_IDs = find( ~isfinite(A.Var1))
```

Your next task in Part A:

2) Using matlab's *find* , *strcmp* , and *isfinite* functions

a) fill in the 4 missing entries in Column A.Var2 with the string "Irregular"

b) fill in the 2 missing entries in Column A.Var1 with integers "6" and "10," respectively

See next page for the next set of tasks !!

Part B: Coding the ML box for a new input biopsy sample ! =)

Now, we're ready to code a super-simple naïve Bayes classifier for your biopsy ML box !

New biopsy sample

Output

$$\vec{x} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{bmatrix} \begin{matrix} shape \\ radius \\ concavity \\ rough \\ color \end{matrix} \longrightarrow \boxed{\text{ML box}} \longrightarrow y = \begin{cases} 1 & (malignant, Class\ 1) \\ 0 & (benign, Class\ 2) \end{cases}$$

Naïve Bayes classifier
(discrete attributes)

Suppose you were presented with 5 new biopsy samples:

Attribute	New sample #1	New sample #2	New sample #3	New sample #4	New sample #5
$\vec{x_1}$ $(\boldsymbol{shape})$	Irregular	Irregular	Circle	Circle	Triangle
$\vec{x_2}$ $(\boldsymbol{radius})$	Large	Small	Large	Large	Large
$\vec{x_3}$ $(\boldsymbol{concavity})$	Convex	Flat	Concave	Convex	Concave
$\vec{x_4}$ $(\boldsymbol{texture})$	Rough	Rough	Smooth	Smooth	Smooth
$\vec{x_5}$ $(\boldsymbol{color})$	Neutral color	Red	Neutral color	Dark	Neutral color

We already did this one in Recitation #5
(see video / tablet scribbles for review) !!

And from our lectures, you know that given a new biopsy sample with an attribute vector $\vec{x}$, the big "positive" and "negative" questions you have to ask are (and *I don't care about the denominator term*!)

$$Big\ positive\ question\text{:}\quad P(\ C_1 \mid \vec{x}\ ) = \frac{P(\ x_1 \mid C_1) \cdot P(\ x_2 \mid C_1) \cdot P(\ x_3 \mid C_1) \cdot P(\ x_4 \mid C_1) \cdot P(\ x_5 \mid C_1)}{P(\vec{x})} \cdot P(C_1)$$

$$Big\ negative\ question\text{:}\quad P(\ C_2 \mid \vec{x}\ ) = \frac{P(\ x_1 \mid C_2) \cdot P(\ x_2 \mid C_2) \cdot P(\ x_3 \mid C_2) \cdot P(\ x_4 \mid C_2) \cdot P(\ x_5 \mid C_2)}{P(\vec{x})} \cdot P(C_2)$$

Your next task in Part B

1) Using matlab, code a naïve Bayes classifier that will classify these 5 new samples ! For *each of the 5 biopsy samples*, please echo the following:

a) The log of the probabilities of the big "positive" and "negative" questions… without the denominator term's contribution (only report the product of the gray-shaded boxes)

I want the log of products of the gray box terms !!

$$Big\ positive:\quad P(\ C_1 \mid \vec{x}\ ) = \frac{P(\ x_1 \mid C_1) \cdot P(\ x_2 \mid C_1) \cdot P(\ x_3 \mid C_1) \cdot P(\ x_4 \mid C_1) \cdot P(\ x_5 \mid C_1)}{P(\vec{x})} \cdot P(C_1)$$

$$Big\ negative:\ P(\ C_2 \mid \vec{x}\ ) = \frac{P(\ x_1 \mid C_2) \cdot P(\ x_2 \mid C_2) \cdot P(\ x_3 \mid C_2) \cdot P(\ x_4 \mid C_2) \cdot P(\ x_5 \mid C_2)}{P(\vec{x})} \cdot P(C_2)$$

I want the log of products of the gray box terms !!

b) Then, I want to know the final classifications for each of the 5 biopsy samples: They're either gonna be malignant (Class C_1)…… or benign (Class C_2) !

2) You will notice your log(probability) answer for Biopsy sample #5 looks a little bit weird !! Give me a plausible answer as to why your log(probability) answer looks the way it is.

Hints:

1) Try to do sketch out the calculations for Sample #2… or Sample #3 on paper first… =)
Then, try to think about how you would code it so that your script will automatically calculate these probabilities !!

** More hints on the next page ***

2) If you review the video for recitation #5, where I walked through how you would calculate the

"The big positive question" conditional probabilities, and
"The big negative question" conditional probabilities

Try to think about these concepts…. And think about how you would translate them into code:

a) Mutually-exclusive members ***** *This is the main operation you want to think about !!*

b) Sorting your data: *Would it help if you have sorted your data before you do any of your calculations (you may or may not have to do this)*

*** Try to look up the matlab command "sortrows" and see how It works!*

"String comparison-based" programming style

c) String comparisons: *You will definitely have the use the combination of "strcmp" and "find" commands in your code !*

Consider the 3 following lines (note the ~ (NOT logical operator) in the 3rd line):

```
shit = {'dammit'; 'crap'; 'bull'; 'dammit'}
my_indices = find(strcmp('dammit', shit))

my_opposite_logic_indices = find(   ~strcmp('dammit', shit)    )
```

** More hints on the next page ***

"Integer- encoded" programming style

d) Integer encoding your strings: *Maybe you might want to encode your attribute sub-categories into integers !! For instance, you can use the "strcmp" function to do the following task:*

Instance $\overrightarrow{x_1}$ $(shape)$	We can encode the sub-categories as integers !!
Circle	0
Irregular	1
Triangle	2

e) Using the "find" function on numerical vectors: *Consider the following lines:*

```
% -- Define a dummy vector
a = [ 3  3  20  60  80  100]

% -- Check out these lines...
my_indices_for3 = find(a == 3)
my_indices_for_other_than3 = find(a ~= 3)
```

```
% -- The one-hot encoding options (different scheme than the above 2 lines)
where_is_my3 = ismember(a, 3)
where_is_my_not3 = ~ismember(a, 3)

% -- This will give you the same answer as the previous line  (inverting the logic of "where_is_my3")
where_is_my_not3_version2 = ~ where_is_my3

% -- logical AND operations with your logical answers
my_logical_AND_between_logical_vectors  = (where_is_my3)     &     (where_is_my_not3)
% -- logical OR operations with your logical answers
my_logical_OR_between_logical_vectors  = (where_is_my3)     |     (where_is_my_not3)
```

```
% -- Turn logicals into true 8-bit integers (so you can do calculations on it)
where_is_my3_integer_format = uint8(where_is_my3)
```

I think that's enough hints for you to play around with !!! =)