For Instructor/TA Use Only

Student Name:	
Student Last Name:	
Student Andrew ID:	

Q1 (max 5)	**Q2 (max 30)**	**Q3 (max 15)**	**Q4 (max 50)**		**Total (max 100)**

24-787 Artificial Intelligence and Machine Learning for Engineering Design

Homework 2

Decision Trees

Naming and Folder structure: Name a root folder "andrewid_hw#". Example: lkara_hw1. In that root folder, create subfolders such as q1, q2, q3 etc. each corresponding to a question in the assignment. The files and documents related to each question should go into the corresponding subfolder.

File types: Your folders and subfolders should not contain any files except .pdf, .doc, .docx, .m, .mat, .txt .zip files. If you want to take photos of your assignment, just make sure that combine all the jpg into a single pdf file and make it clear. Unorganized and illegible files will be penalized. Take care in arranging your illustrations, written solutions, photos. If any, do not scan the your hand-written solutions in the highest resolution.

What to include: Only submit the required files that you create or modify. For programming questions, include your source code (.m and/or .mat file rather than text) in your submission. All other files, including the ones we provide as supporting functionality, are unnecessary because we already have copies of them. Unless we tell you otherwise, make sure that you only submit the file you edited/changed. For .m files, we expect to click the "run" button and everything should work (obviously, after we include the additional support files and data that was given to you with the assignment).

Readme files: Provide a readme.txt within each questions' subfolders when necessary to aid grading. This is useful when there are many files of the same type. If your folder structure and your file names are logical, you should not need a readme.txt file.

Submission: Zip the entire assignment by compressing the root folder. Example: lkara_hw1.zip. Do not use .rar**.** Make sure your entire submission file is less than 20MB when zipped. You shall use the online homework submission platform. The submission entry can be found at the top right of homework/assignment page. Make sure your submission is in .zip format, otherwise it cannot be uploaded. Feel free to re-submit your homework before deadline if you find any mistake in your previous submission. Only the last submission will be graded.

1. *(5 points)*

Consider two binary variables x and y having the joint distribution given in the table below.

$$\begin{array}{cc|cc} & & \multicolumn{2}{c}{y} \\ & & 0 & 1 \\ \hline x & 0 & 1/3 & 1/3 \\ & 1 & 0 & 1/3 \end{array}$$

Evaluate the following distributions and/or quantities.

$$P(x),\quad P(y),\quad P(x|y),\quad P(y|x),\quad P(x,y),$$
$$H(x),\quad H(y),\quad H(x|y),\quad H(y|x),\quad IG(x|y).$$

2. *(30 points)*

You may solve this problem on paper, or use Matlab; both methods are acceptable. Suppose you are given the simple dataset listed in the table below and need to learn a decision tree from the data. There are 8 instances total, each with 3 binary attributes (X_1, X_2, X_3) and a binary class label (Y). Use the data to answer the following questions.

Instance	X_1	X_2	X_3	Y
1	0	0	0	0
2	0	0	1	0
3	0	1	0	1
4	0	1	1	1
5	1	0	1	1
6	1	0	1	1
7	1	1	0	0
8	1	1	0	0

(a) Compute the entropy of the class label.

(b) Calculate the information gains of Y with respect to each of the three attributes.

(c) Which attribute should be selected for the root of the decision tree? Why?

(d) After the root node is selected, the entire tree can be learned by recursively splitting the data into two subgroups, finding the next best attribute to split on, dividing the subgroup into smaller groups, and so forth. How do you know when to stop growing the tree? In other words, what are the stopping criteria? Describe in 1-2 sentences. You may consult the lecture slides and/or investigate this topic using online resources.

(e) By running the algorithm, compute your decisions tree? Draw your tree. Consult the lecture slides if you are unsure about drawing conventions.

(f) What is the training error on the training dataset? Consider all 8 instances to be training samples.

(g) Now suppose you are presented new instances for which the class Y is unknown. Use your decision tree to predict the label of each instance listed below.

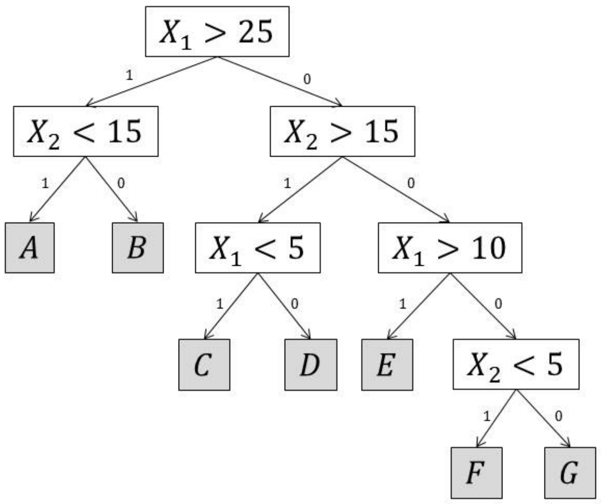
Instance	X_1	X_2	X_3	Y
9	1	1	1	?
10	1	0	0	?
11	0	1	1	?

(h) For the decision tree you have learned so far, do you have any basis on which to evaluate if the tree is overfitting? Why or why not? How might you combat overfitting in a decision tree? (Answers will not be unique. Describe your favorite one in one or two sentences.)

(i) If the labels on Instances 6 and 7 were changed to 0 and 1, respectively, would the structure of the learned decision tree change? Why or why not? Also, would any of the leaf nodes change? Explain your reasoning.

3. *(15 points)*

Consider the following decision tree. For the arrows: 1=True, 0=False.



(a) On a 2D plot whose axes are X_1 and X_2, draw the decision boundaries defined by this tree. Each leaf is labeled with a letter. Write this letter in the corresponding region of the instance space. Your plot should produce rectangular blocks.

(b) Draw another decision tree that is syntactically different than the one shown above but defines the same decision boundaries.

4. *(50 points)*

In this problem, you will find an executable file `trog-win.exe` is an oracle that decides whether a sentence you type bodes Joy or Despair. This program was written by Prof. Andrew Moore. You can invoke the program as follows:

```
Trog-win 1 play

Speak to the oracle> hello there
******* Despair *******
Speak to the oracle> whats your problem
******* Joy *******
Speak to the oracle> fair enough
******* Despair *******
Speak to the oracle> i am perplexed
******* Despair *******
```

There are many oracles. Each has a different oracle number (that's the 1 on the command line above). If you choose a different oracle, one of many possible different rules may be used. Some oracles are easy for a human to work out, while others are very cryptic. Your goal in this assignment is to write a MATLAB program that can predict an oracle's pronouncement given a set of inputs.

What your program should do

To begin with, you have been provided a set of support codes written in MATLAB. And your task is to complete the missing functions, so that the overall program runs in the following way.

The file `main.m` will be the starting point of the whole program. When invoked by pressing F5 in MATLAB, this file first calls a `trog_DataManager` which in turn calls `trog-win.exe` to obtain a collection of training and testing samples from the oracle. The requested data are stored in `trog.dat` and `trog.tst`. To complete this homework assignment, it is not required to study or understand the details of `trog_DataManager.m.`

In `trog.dat`, you will find a set of sentences generated by some oracle. The format of the file is:

<features1><sentence1> → *<value1>*
<features2><sentence2>→*<value2>*
.
.
.
<featuresN><sentenceN> → *<valueN>*

where each `<sentence>` consists of one, two, or three space-separated strings. Each string is 7 characters or less. Each `<value>` is either joy or despair. Each `<features>` is a string of binary digits, giving you information about the attributes if the sentences that the oracle might use in making is Joy/Despair decision.

Here is an example `trog.dat` file:

```
111011000000010       gloves    werent      sent  -> despair
110110110000111      attract      glad     grime  -> joy
111111110000011      insight     names   stained  -> joy
011001000000010        doggy   scooped            -> joy
111111100000001       turned    fuller   vaguely  -> joy
111101100000000       wildly    planet   begging  -> joy
```

Typically, you can expect many more lines. Note that given the 15 binary features, an oracle can perfectly make its Joy/Despair decision. So the actual words in this file can be ignored from a learning perspective.

In `trog.tst` your will find something similar to the above, but without the Joy/Despair decisions:

```
101111101011001         wake     aside   greener
111101000000001        picks     gazes     threw
010000000000001       warmly
101101100000001         west    debris     floor
001011000000000         reek    gaping
011011000000000       dialed   caliber
110010000010011        budge      hums      says
```

Having obtained both training set and test set, `main.m` will instantiate a decision tree (`DecisionTree.m`) and initiate its training algorithm, so as to learn the decision rules from the training set from `trog.dat`, build the decision tree by adding nodes (`DecisionTreeNode.m`), and classify the test set in `trog.tst`. The classifications results will be stored in a new file called `trog.sub` which has same format as `trog.dat`. For example:

```
101111101011001     wake     aside   greener  -> Joy
111101000000001    picks     gazes     threw  -> Despair
010000000000001   warmly                      -> Despair
101101100000001     west    debris     floor  -> Joy
001011000000000     reek    gaping            -> Joy
011011000000000   dialed   caliber            -> Joy
110010000010011    budge      hums      says  -> Despair
```

After `trog.sub` is generated, `trog_DataManager` will submit it to `trog-win.exe`, so as to check the accuracy (or error rate) of your predictions. An output similar to the following will be printed:

```
The oracle says you have 97 out of 100 correct ( 97.0%)
Oracle     #Training Sample    #Test Sample   Error Rate (%)
   2              50               100             3.00
```

Finally, the `DecisionTree` class we provided would finally visualize the structure of the tree in a pop-up figure window. You could take snapshots of the tree structure to include in your report.

Code Support

To simplify your processing, we have provided `main.m` and the `trog_DataManager` class, which will convert between `trog.*` files and MATLAB data matrices and vectors. This means you won't have to write any of the code to parse various trog files. The 15 binary attributes of each sample are stored in n-by-15 matrices, where n is the sample size. The Joy/Despair decisions are encoded as a binary column vector with the size of n-by-1, where Joy=1 and Despair=0. The attributes are stored in variables containing the "attrib" substring, while the 1-0 decisions are stored in variables containing the word "class."

Moreover, `main.m` has been written to set up the workflow as described above.

Finally, `DecisionTree.m` is complete. You will need to study the first three member functions of `DecisionTree.m` to understand the data structure and how node-level training or classification is initiated from there.

You might find the enclosed MATLAB Quick Reference Card (PDF) useful.

Your Tasks

Your task is to fill in the missing parts of `DecisionTreeNode.m`, so that ultimately we should be able to check your code by running `main.m`. The code should proceed without the need of any human intervention. Specifically, you will need to complete the following functions in `DecisionTreeNode.m`:

```
entropy()
entropy_of_class()
find_decision_attrib()
train()
classify()
```

What should be in your report

A description of your approach, results showing how well the learning algorithm(s) perform(s). You should present informative results showing whether your algorithm, can ever learn: can it learn on a few oracles or all? How many examples does it usually need to be given in order to learn well? The report should include pseudo-code for functions and data structures you have written. Discuss if there are glaring implementation errors or bugs in your code. Even if you cannot resolve them in the allotted time, can you determine where these errors might be originating by diagnosing your code? Discuss such issues. Were there any innovations beyond the bare minimal implementation of a basic algorithm? For example, what solutions have been devised to overcome over-fitting? Discuss your results: Are you satisfied with your outcomes? What other improvements could you try?

In addition to the discussions above, provide sample results of your program similar to the following:

```
Oracle     #Training Sample     #Test Sample   Error Rate (%)
     2                   50              100   3.00
     2                  100              100   3.00
     2                  500              100   0.00
     7                   50              100    0.00
     7                  100              100   0.00
     7                  500              100   0.00
```

Also include screenshots showing the structure of the learned decision tree. A figure showing the structure of the learned decision tree should pop up automatically after running `main.m`.

Run similar tests for oracles: 1, 2, 3, 5, 7, 8, 12, and 15. You may vary number of training and tests samples as you desire. You can adjust the oracle ID and sample size in `main.m`.

For a maximum of 10% extra credits, implement a measure that could combat over-fitting. Refer to the AIMA book for inspirations.