HOMEWORK 2
DECISION TREES[1]

10-301 / 10-601 INTRODUCTION TO MACHINE LEARNING (SPRING 2020)
https://mlcourse.org

OUT: Jan. 23, 2020
DUE: Feb. 5, 2020 11:59 PM
TAs: Sankalp Patro, Quentin Cheng, Vishal Baskar, Sana Lakdawala, Vinay Sameer Kadi, Hanyue Chai

Summary It's time to build your first end-to-end learning system! In this assignment, you will build a Decision Tree classifier and apply it to several binary classification problems. This assignment consists of several parts: In Section 1.1 and 1.2, you will work through some Information Theory basics in order to "learn" a Decision Tree on paper. Then in Section 2, you will implement Decision Tree learning, prediction, and evaluation. Using that implementation, you will answer a few empirical questions in Section 1.3

START HERE: Instructions

- **Collaboration Policy**: Collaboration on solving the homework is allowed, after you have thought about the problems on your own. It is also OK to get clarification (but not solutions) from books or online resources, again after you have thought about the problems on your own. There are two requirements: first, cite your collaborators fully and completely (e.g., "Jane explained to me what is asked in Question 3.4"). Second, write your solution *independently*: close the book and all of your notes, and send collaborators out of the room, so that the solution comes from you only. See the collaboration policy on the website for more information: http://www.cs.cmu.edu/~mgormley/courses/10601/about.html
- **Late Submission Policy:** See the late submission policy here: http://www.cs.cmu.edu/~mgormley/courses/10601/about.html
- **Submitting your work:** You will use Gradescope to submit answers to all questions, and Autolab to submit your code. Please follow instructions at the end of this PDF to correctly submit all your code to Autolab.
  - **Programming:** You will submit your code for programming questions on the homework to Gradescope (https://gradescope.com). After uploading your code, our grading scripts will autograde your assignment by running your program on a virtual machine (VM). When you are developing, check that the version number of the programming language environment (e.g. Python 3.6.9, Octave 4.2.2, OpenJDK 11.0.5, g++ 7.4.0) and versions of permitted libraries (e.g. `numpy` 1.17.0 and `scipy` 1.4.1) match those used on Gradescope. (Octave users: Please make sure you do not use any Matlab-specific libraries in your code that might make it fail against our tests.) You have a **total of 10 Gradescope programming submissions.** Use them wisely. In order to not waste code submissions, we recommend debugging your implementation on your local machine (or the linux servers) and making sure your code is running correctly first before any Gradescope coding submission.

[1]Compiled on Friday 7th February, 2020 at 16:40

  - **Written:** For written problems such as short answer, multiple choice, derivations, proofs, or plots, we will be using Gradescope (https://gradescope.com/). Please use the provided template. Submissions can be handwritten onto the template, but should be labeled and clearly legible. If your writing is not legible, you will not be awarded marks. Alternatively, submissions can be written in LaTeX. Regrade requests can be made, however this gives the TA the opportunity to regrade your entire paper, meaning if additional mistakes are found then points will be deducted. Each derivation/proof should be completed on a separate page. For short answer questions you **should not** include your work in your solution. If you include your work in your solutions, your assignment may not be graded correctly by our AI assisted grader.

- **Materials:** The data that you will need in order to complete this assignment is posted along with the writeup and template on Piazza.

For multiple choice or select all that apply questions, shade in the box or circle in the template document corresponding to the correct answer(s) for each of the questions. For LaTeXusers, use ■ and ● for shaded boxes and circles, and don't change anything else.

Instructions for Specific Problem Types

For "Select One" questions, please fill in the appropriate bubble completely:

Select One: Who taught this course?

● Matt Gormley

○ Marie Curie

○ Noam Chomsky

If you need to change your answer, you may cross out the previous answer and bubble in the new answer:

Select One: Who taught this course?

● Matt Gormley

○ Marie Curie
● Noam Chomsky

For "Select all that apply" questions, please fill in all appropriate squares completely:

Select all that apply: Which are scientists?

■ Stephen Hawking

■ Albert Einstein

■ Isaac Newton

□ I don't know

Again, if you need to change your answer, you may cross out the previous answer(s) and bubble in the new answer(s):

Select all that apply: Which are scientists?

■ Stephen Hawking

■ Albert Einstein

■ Isaac Newton
■ I don't know

For questions where you must fill in a blank, please make sure your final answer is fully included in the given space. You may cross out answers or parts of answers, but the final answer must still be within the given space.

Fill in the blank: What is the course number?

10-601	10-7601

1 Written Questions [35pts]

Answer the following questions in the HW2 solutions template provided. DO NOT show your work. Then upload your solutions to Gradescope.

1.1 Gini Impurity

First, let's think a little bit about decision trees. The following dataset D consists of 7 examples, each with 3 attributes, (A, B, C), and a label, Y.

A	B	C	Y
1	1	0	0
1	1	2	1
1	0	0	1
1	1	2	1
0	0	2	0
0	1	1	0
0	0	0	0

Use the data above to answer the following questions.

A few important notes:

- *All calculations should be done without rounding!* After you have finished all of your calculations, write your rounded solutions in the boxes below.
- Note that, throughout this homework, we will use the convention that the leaves of the trees do not count as nodes, and as such are not included in calculations of depth and number of splits. (For example, a tree which classifies the data based on the value of a single attribute will have depth 1, and contain 1 split.)

1. (1 point) What is the Gini impurity of Y, $G(Y; D)$, before any splitting happens? (Please include one number rounded to the fourth decimal place, e.g. 0.1234)

   0.4898

2. (1 point) What is the resulting Gini gain on Y if we were to split on A, i.e. $G(Y, A; D)$? (Please include one number rounded to the fourth decimal place, e.g. 0.1234)

   0.2755

3. (1 point) What is the resulting Gini gain on Y if we were to split on B, i.e. $G(Y, B; D)$? (Please include one number rounded to the fourth decimal place, e.g. 0.1234)

   0.0136

4. (1 point) What is the resulting Gini gain on Y if we were to split on C, i.e. $G(Y, C; D)$? (Please include one number rounded to the fourth decimal place, e.g. 0.1234)

   0.1088

5. (1 point) Consider the dataset given above. Which attribute (A, B, or C) would a decision tree algorithm pick first to branch on, if its splitting criterion is Gini impurity?

   Select one:

   - ● A
   - ○ B
   - ○ C

6. (1 point) Consider the dataset given above. Which is the second attribute you would pick to branch on, if its splitting criterion is Gini impurity? (*Hint:* Notice that this question correctly presupposes that there is *exactly one* second attribute.)

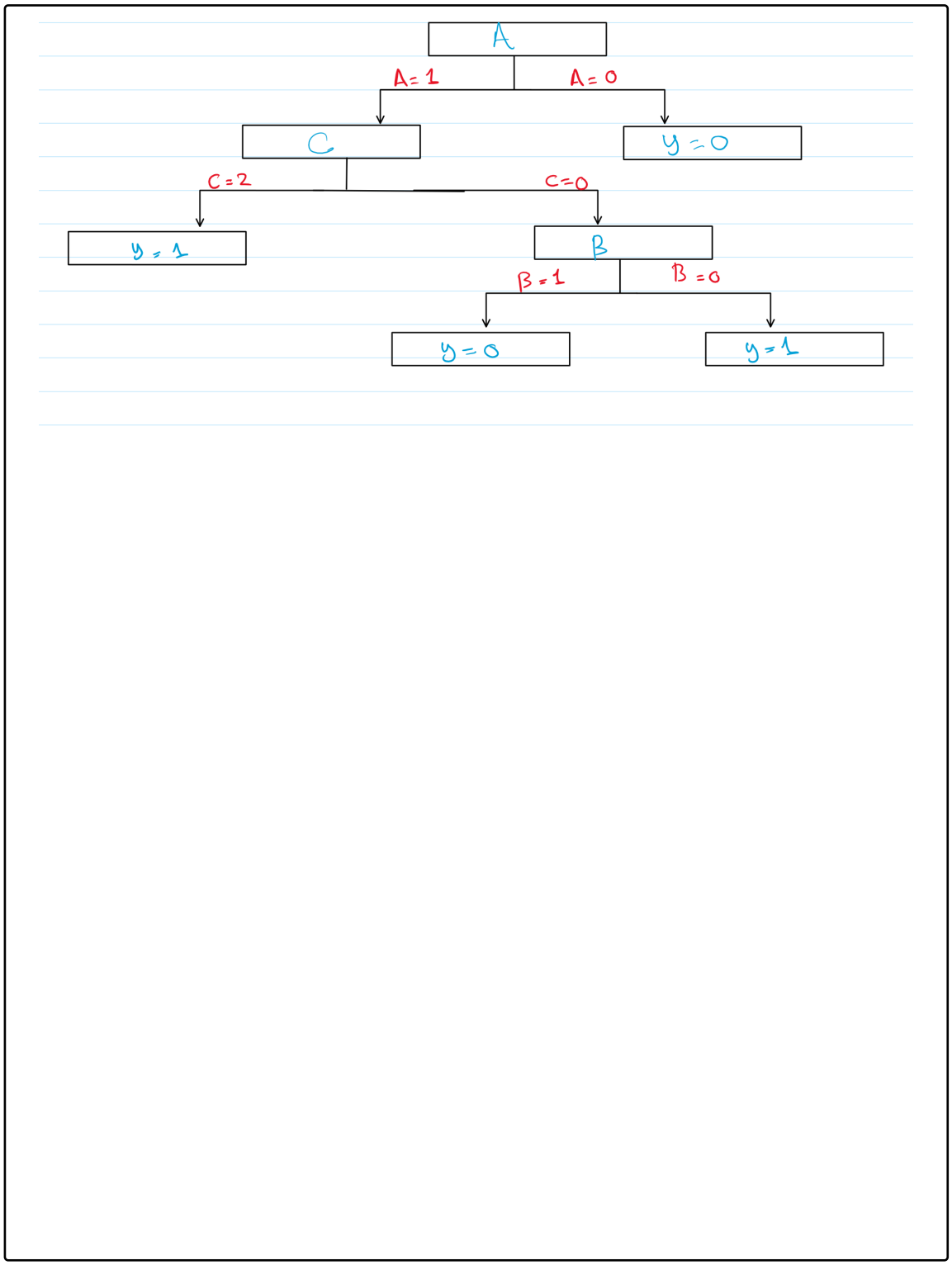
   Select one:

   - ○ A
   - ○ B
   - ● C

7. (1 point) If the same algorithm continues until the tree perfectly classifies the data, what would the depth of the tree be?

   3

8. (4 points) Draw your completed Decision Tree. Label the non-leaf nodes with which attribute the tree will split on (e.g. B), the edges with the value of the attribute (e.g. 1 or 0), and the leaf nodes with the classification decision (e.g. $Y = 0$).



1.2 Mutual Information

Now, using the same dataset, let's consider another splitting criterion, Mutual Information.

A	B	C	Y
1	1	0	0
1	1	2	1
1	0	0	1
1	1	2	1
0	0	2	0
0	1	1	0
0	0	0	0

Use the data above to answer the following questions.

1. (1 point) What is the entropy of Y in bits, $H(Y)$? In this and subsequent questions, when we request the units in *bits*, this simply means that you need to use $\log$ base 2 in your calculations.[2] (Please include one number rounded to the fourth decimal place, e.g. 0.1234)

   0.9852

2. (1 point) What is the mutual information[3] of Y and A in bits, $I(Y;A)$? (Please include one number rounded to the fourth decimal place, e.g. 0.1234)

   0.5216

3. (1 point) What is the mutual information of Y and B in bits, $I(Y;B)$? (Please include one number rounded to the fourth decimal place, e.g. 0.1234)

   0.0202

4. (1 point) What is the mutual information of Y and C in bits, $I(Y;C)$? (Please include one number rounded to the fourth decimal place, e.g. 0.1234)

   0.1981

[2] If instead you used $\log$ base e, the units would be *nats*; $\log$ base 10 gives *bats*.

[3] In the context of decision trees, and therefore in this assignment, the terms "information gain" and "mutual information" are synonymous (e.g. they have the same meaning as one another). However, in information theory and machine learning in general, "information gain" is synonymous with "Kullback-Leibler (K-L) divergence", often referred to as relative entropy, which is not the same as mutual information. For more information, go here.

5. (1 point) Consider the dataset given above. Which attribute (A, B, or C) would a decision tree algorithm pick first to branch on, if its splitting criterion is mutual information?

   Select one:

   - ● A
   - ○ B
   - ○ C

6. (3 points) Is the resulting tree the same as if we split using Gini Impurity? Explain in your own words what Gini Impurity and Mutual Information are each calculating.

Yes, the resulting tree is the same as if we split using Gini Impurity.
They both measure the probability of miss classification of a randomly chosen element in the data-set.

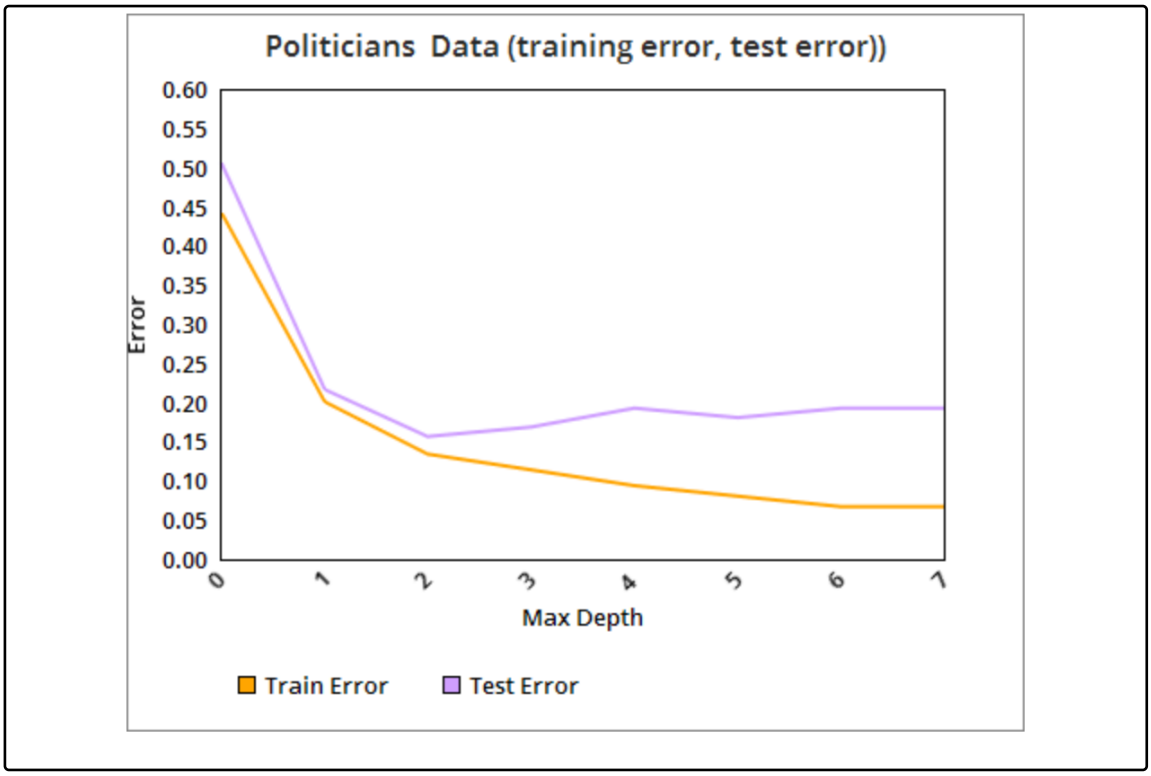
1.3 Empirical Questions

The following questions should be completed as you work through the programming portion of this assignment (Section 2).

1. (2 points) Train and test your decision tree on the politician dataset and the education dataset with four different values of max-depth, $\{0, 1, 2, 4\}$. Report your findings in the HW2 solutions template provided. A Decision Tree with max-depth 0 is simply a *majority vote classifier*; a Decision Tree with max-depth 1 is called a *decision stump*. If desired, you could even check that your answers for these two simple cases are correct using your favorite spreadsheet application (e.g. Excel, Google Sheets). (Please round each number to the fourth decimal place, e.g. 0.1234)

Dataset	Max-Depth	Train Error	Test Error
politician	0	0.4429	0.5060
politician	1	0.2013	0.2169
politician	2	0.1342	0.1566
politician	4	0.0939	0.1928
education	0	0.325	0.31
education	1	0.195	0.23
education	2	0.18	0.21
education	4	0.11	0.135

2. (3 points) For the politicians dataset, create a computer-generated plot showing error on the y-axis against depth of the tree on the x-axis. Plot *both* training error and testing error, clearly labeling which is which. That is, for each possible value of max-depth $(0, 1, 2, \ldots,$ up to the number of attributes in the dataset), you should train a decision tree and report train/test error of the model's predictions.



3. (2 points) Suppose your research advisor asks you to run some model selection experiments and then report your results. You select the Decision Tree model's max-depth to be the one with lowest test error in metrics.txt and then report that model's test error as the performance of our classifier on held out test data. Is this a good experimental setup? If so, why? If not, why not?

No, this is actually a bad experiment as it is considered cheating. Also, it does not take the inductive bias of decision trees into account. The best way to do this experiment is by using the concept of cross-validation as max-depth is hyper-parameter.

4. (2 points) In this assignment, we used max-depth as our stopping criterion, and as a mechanism to prevent overfitting. Alternatively, we could stop splitting a node whenever the gini gain for the best attribute is lower than a threshold value. This threshold would be another hyperparameter. Theoretically, how would increasing this threshold value affect the number of nodes and depth of the learned trees?

Increasing the threshold value would in expectation decreases the number of nodes and tree depth. Previously, we used a threshold value ($t = 0$) and now, we are sitting $t = c$, where $c > 0$. This means if the gini-gain at some nodes is $\leq c$, the tree will stop splitting where before it would continue till the gini-gain is ≤ 0.

5. (2 points) From section 1.3 question 4, how would you set-up model training to choose the threshold value?

I would perform a random search over a set of possible discrete threshold values along with cross-validation to choose the best value based on the validation error.

6. (3 points) Print (do not handwrite!) the decision tree which is produced by your algorithm for the politician data with max depth 3. Instructions on how to print the tree could be found in section 2.3.4.

```
[83 democrat /66 republican]
|  Superfund_right_to_sue = y : [28 democrat /64 republican]
| |  Aid_to_nicaraguan_contras = n : [13 democrat /58 republican]
| | |  Duty_free_exports = y : [5 democrat /5 republican]
| | |  Duty_free_exports = n : [8 democrat /53 republican]
| |  Aid_to_nicaraguan_contras = y : [15 democrat /6 republican]
| | |  Mx_missile = y : [3 democrat /6 republican]
| | |  Mx_missile = n : [12 democrat /0 republican]
|  Superfund_right_to_sue = n : [55 democrat /2 republican]
| |  Export_south_africa = y : [55 democrat /1 republican]
| | |  Immigration = n : [46 democrat /0 republican]
| | |  Immigration = y : [9 democrat /1 republican]
| |  Export_south_africa = n : [0 democrat /1 republican]
```

7. (2 points) Below is the printed decision tree for the previous problem when the splitting criteria is Mutual Information instead of Gini impurity. Compare the two and comment on the differences.

```
[83 democrat/66 republican]
Superfund_right_to_sue = y: [28 democrat/64 republican]
| Aid_to_nicaraguan_contras = y: [15 democrat /6 republican]
|| Mx_missile = y: [3 democrat/6 republican]
|| Mx_missile = n: [12 democrat/0 republican]
| Aid_to_nicaraguan_contras = n: [13 democrat/58 republican]
|| Export_south_africa = y: [13 democrat/38 republican]
|| Export_south_africa = n: [0 democrat/20 republican]
Superfund_right_to_sue = n: [55 democrat/2 republican]
| Export_south_africa = y: [55 democrat/1 republican]
|| Immigration = y: [9 democrat/1 republican]
|| Immigration = n: [46 democrat/0 republican]
| Export_south_africa = n: [0 democrat/1 republican]
```

Almost both trees are split in the same way except that at the Aid_to_nicaraguan_contras sub-tree. The maximum gini gain of that node when it is (no) was by splitting on Duty_free _exports. However, the largest Information gain at the same node was Export_south_africa. Unlike the gini gain, mutual information yielded a perfect separation at the (no) branch which more appealing.

8. After you have completed all other components of this assignment, report your answers to the collaboration policy questions detailed in the Academic Integrity Policies found here. 1. Did you receive any help whatsoever from anyone in solving this assignment? Is so, include full details. 2. Did you give any help whatsoever to anyone in solving this assignment? Is so, include full details. 3. Did you find or come across code that implements any part of this assignment ? If so, include full details.

1. Did you receive any help whatsoever from anyone in solving this assignment?
   Yes / **No** .
   - If you answered 'yes', give full details: _____
   - (e.g. "Jane Doe explained to me what is asked in Question 3.4")
2. Did you give any help whatsoever to anyone in solving this assignment?
   Yes / **No**.
   - If you answered 'yes', give full details: _____
   - (e.g. "I pointed Joe Smith to section 2.3 since he didn't know how to proceed with Question 2")
3. Did you find or come across code that implements any part of this assignment ?
   Yes / **No**. (See below policy on "found code")
   - If you answered 'yes', give full details: .
   - (book & page, URL & location within the page, etc.).

2 Programming [75pts]

Your goal in this assignment is to implement a binary classifier, entirely from scratch–specifically a Decision Tree learner. In addition, we will ask you to run some end-to-end experiments on two tasks (predicting the party of a politician / predicting final grade for high school students) and report your results. You will write two programs: `inspection.{py|java|cpp|m}` (Section 2.2) and `decisionTree.{py|java|cpp|m}` (Section 2.3). The programs you write will be automatically graded using the Gradescope system. You may write your programs in **Octave**, **Python**, **Java**, or **C++**. However, you should use the same language for all parts below.

2.1 The Tasks and Datasets

Materials Download the zip file from Piazza ("Download handout"). The zip file will have a handout folder that contains all the data that you will need in order to complete this assignment.

Datasets The handout contains three datasets. Each one contains attributes and labels and is already split into training and testing data. The first line of each `.tsv` file contains the name of each attribute, and *the class is always the last column.*

1. **politician:** The first task is to predict whether a US politician is a member of the Democrat or Republican party, based on their past voting history. Attributes (aka. features) are short descriptions of bills that were voted on, such as *Aid_to_nicaraguan_contras* or *Duty_free_exports*. Values are given as *'y'* for yes votes and *'n'* for no votes. The training data is in `politicians_train.tsv`, and the test data in `politicians_test.tsv`.
2. **education:** The second task is to predict the final *grade* (A, not A) for high school students. The attributes (covariates, predictors) are student grades on 5 multiple choice assignments *M1* through *M5*, 4 programming assignments *P1* through *P4*, and the final exam *F*. The training data is in `education_train.tsv`, and the test data in `education_test.tsv`.
3. **small:** We also include `small_train.tsv` and `small_test.tsv`—a small, purely for demonstration version of the politicians dataset, with *only* attributes *Anti_satellite_test_ban* and *Export_south_africa*. For this small dataset, the handout tar file also contains the predictions from a reference implementation of a Decision Tree with max-depth 3 (see `small_3_train.labels`, `small_3_test.labels`, `small_3_metrics.txt`). You can check your own output against these to see if your implementation is correct.[4]

Note: For simplicity, all attributes are discretized into just two categories (i.e. each node will have at most two descendents). This applies to all the datasets in the handout, as well as the additional datasets on which we will evaluate your Decision Tree.

[4] Yes, you read that correctly: we are giving you the correct answers.

2.2 Program #1: Inspecting the Data [5pts]

Write a program `inspection.{py|java|cpp|m}` to calculate the overall Gini impurity (i.e. the Gini impurity of the labels for the entire dataset and before any splits) and the error rate (the percent of incorrectly classified instances) of classifying using a majority vote (picking the label with the most examples). You do not need to look at the values of any of the attributes to do these calculations, knowing the labels of each example is sufficient.

Command Line Arguments The autograder runs and evaluates the output from the files generated, using the following command:

For Python:	`$ python inspection.py <input> <output>`
For Java:	`$ javac inspection.java; java inspect <input> <output>`
For C++:	`$ g++ inspection.cpp; ./a.out <input> <output>`
For Octave:	`$ octave -qH inspection.m <input> <output>`

Your program should accept two command line arguments: an input file and an output file. It should read the `.tsv` input file (of the format described in Section 2.1), compute the quantities above, and write them to the output file so that it contains:

```
gini_impurity: <gini impurity value>
error: <error value>
```

Example For example, suppose you wanted to inspect the file `small_train.tsv` and write out the results to `small_inspect.txt`. For Python, you would run the command below:

```
$ python inspection.py small_train.tsv small_inspect.txt
```

Afterwards, your output file `small_inspect.txt` should contain the following:

```
gini_impurity: 0.4974.
error: 0.4643.
```

Our autograder will run your program on several input datasets to check that it correctly computes gini impurity and error, and will take minor differences due to rounding into account. You do not need to round your reported numbers! The Autograder will automatically incorporate the right tolerance for float comparisons.

For your own records, run your program on each of the datasets provided in the handout—this error rate for a *majority vote* classifier is a baseline over which we would (ideally) like to improve.

2.3 Program #2: Decision Tree Learner [65pts]

In decisionTree.{py | java | cpp | m}, implement a Decision Tree learner. This file should learn a decision tree with a specified maximum depth, print the decision tree in a specified format, predict the labels of the training and testing examples, and calculate training and testing errors.

Your implementation must satisfy the following requirements:

- Use Gini impurity to determine which attribute to split on. You want to choose the attribute that maximizes Gini gain.
  - **Remember:** Gini gain is defined as $G(Y, X; D) = G(Y; D) - \sum_{i=1}^{n} P(X = i) * G(Y; D_{X=i})$
- Be sure you're correctly weighting your calculation of Gini impurity. For a split on attribute X, the weighted Gini impurity afterwards is $P(X = 0) * G(Y; D_{X=0}) + P(X = 1) * G(Y; D_{X=1})$.
- As a stopping rule, only split on an attribute if the Gini gain is > 0.
- Do not grow the tree beyond a max-depth specified on the command line. For example, for a maximum depth of 3, split a node only if the Gini gain is > 0 and the current level of the node is < 3.
- Use a majority vote of the labels at each leaf to make classification decisions. If the vote is tied, choose the attribute to split on that comes last in the lexicographical order (i.e. Republican should be chosen before Democrat)
- Do not hard-code any aspects of the datasets into your code. We may autograde your programs on hidden datasets that include different attributes and output labels.

Careful planning will help you to correctly and concisely implement your Decision Tree learner. Here are a few *hints* to get you started:

- Write helper functions to calculate Gini impurity and gain.
- Make sure to keep track of Gini impurities to calculate Gini gain at subsequent levels.
- Write a function to train a stump (tree with only one level). Then call that function recursively to create the sub-trees.
- In the recursion, keep track of the depth of the current tree so you can stop growing the tree beyond the max-depth.
- Implement a function that takes a learned decision tree and data as inputs, and generates predicted labels. You can write a separate function to calculate the error of the predicted labels with respect to the given (ground-truth) labels.
- Be sure to correctly handle the case where the specified maximum depth is greater than the total number of attributes.
- Be sure to handle the case where max-depth is zero (i.e. a majority vote classifier).
- Look under the FAQ's on Piazza for more useful clarifications about the assignment.

2.3.1 Command Line Arguments

The autograder runs and evaluates the output from the files generated, using the following command:

For Python:	`$ python decisionTree.py [args...]`
For Java:	`$ javac decisionTree.java; java decisionTree [args...]`
For C++:	`$ g++ decisionTree.cpp; ./a.out [args...]`
For Octave:	`$ octave -qH decisionTree.m [args...]`

Where above `[args...]` is a placeholder for six command-line arguments: `<train input> <test input> <max depth> <train out> <test out> <metrics out>`. These arguments are described in detail below:

1. `<train input>`: path to the training input `.tsv` file (see Section 2.1)
2. `<test input>`: path to the test input `.tsv` file (see Section 2.1)
3. `<max depth>`: maximum depth to which the tree should be built
4. `<train out>`: path of output `.labels` file to which the predictions on the *training* data should be written (see Section 2.3.2)
5. `<test out>`: path of output `.labels` file to which the predictions on the *test* data should be written (see Section 2.3.2)
6. `<metrics out>`: path of the output `.txt` file to which metrics such as train and test error should be written (see Section 2.3.3)

As an example, if you implemented your program in Python, the following command line would run your program on the politicians dataset and learn a tree with max-depth of two. The train predictions would be written to `pol_2_train.labels`, the test predictions to `pol_2_test.labels`, and the metrics to `pol_2_metrics.txt`.

```
$ python decisionTree.py politicians_train.tsv politicians_test.tsv \
        2 pol_2_train.labels pol_2_test.labels pol_2_metrics.txt
```

The following example would run the same learning setup except with max-depth three, and conveniently writing to analogously named output files, so you can can compare the two runs.

```
$ python decisionTree.py politicians_train.tsv politicians_test.tsv \
        3 pol_3_train.labels pol_3_test.labels pol_3_metrics.txt
```

2.3.2 Output: Labels Files

Your program should write two output `.labels` files containing the predictions of your model on training data (`<train out>`) and test data (`<test out>`). Each should contain the predicted labels for each example printed on a new line. Use '\n' to create a new line.

Your labels should exactly match those of a reference decision tree implementation—this will be checked by the autograder by running your program and evaluating your output file against the reference solution.

Note: You should output your predicted labels using the same string identifiers as the original training data: e.g., for the politicians dataset you should output democrat/republican and for the education dataset you should output A/notA. The first few lines of an example output file is given below for the politician dataset:

```
democrat
democrat
democrat
republican
democrat
...
```

2.3.3 Output: Metrics File

Generate another file where you should report the training error and testing error. This file should be written to the path specified by the command line argument `<metrics out>`. Your reported numbers should be within 0.01 of the reference solution. You do not need to round your reported numbers! The Autograder will automatically incorporate the right tolerance for float comparisons. The file should be formatted as follows:

```
error(train): 0.0714
error(test): 0.1429
```

The values above correspond to the results from training a tree of depth 3 on `small_train.tsv` and testing on `small_test.tsv`.

2.3.4 Output: Printing the Tree

Finally, you should write a function to pretty-print your learned decision tree. (You may find it more convenient to print the tree *as* you are learning it.) Each row should correspond to a node in the tree. They should be printed in a *depth-first-search* order (but you may print left-to-right or right-to-left). Print the attribute of the node's parent and the attribute value corresponding to the node. Also include the sufficient statistics (i.e. count of positive / negative examples) for the data passed to that node. The row for the root should include *only* those sufficient statistics. A node at depth d, should be prefixed by d copies of the string '| '.

Below, we have provided the recommended format for printing the tree (example for python). You can print it directly to standard out rather than to a file. **This functionality of your program will not be autograded**.

```
$ python decisionTree.py small_train.tsv small_test.tsv 2 \
small_2_train.labels small_2_test.labels small_2_metrics.txt

[15 democrat /13 republican]
| Anti_satellite_test_ban = y: [13 democrat /1 republican]
| | Export_south_africa = y: [13 democrat /0 republican]
| | Export_south_africa = n: [0 democrat /1 republican]
| Anti_satellite_test_ban = n: [2 democrat /12 republican]
| | Export_south_africa = y: [2 democrat /7 republican]
| | Export_south_africa = n: [0 democrat /5 republican]
```

However, you should be careful that the tree might not be full. For example, after swapping the train/test files in the example above, you could end up with a tree like the following.

```
$ python decisionTree.py small_test.tsv small_train.tsv 2 \
swap_2_train.labels swap_2_test.labels swap_2_metrics.txt

[13 democrat/15 republican]
| Anti_satellite_test_ban = y: [9 democrat/0 republican]
| Anti_satellite_test_ban = n: [4 democrat/15 republican]
| | Export_south_africa = y: [4 democrat/10 republican]
```

```
| | Export_south_africa = n: [0 democrat/5 republican]
```

The following pretty-print shows the education dataset with max-depth 3. Use this example to check your code before submitting your pretty-print of the politics dataset (asked in question 14 of the Empirical questions).

```
$ python decisionTree.py education_train.tsv education_test.tsv 3 \
edu_3_train.labels edu_3_test.labels edu_3_metrics.txt

[135 A/65 notA]
| F = A [119 A/23 notA]
| | M3 = A [95 A/10 notA]
| | | M5 = A [79 A/2 notA]
| | | M5 = notA [16 A/8 notA]
| | M3 = notA [24 A/13 notA]
| | | M1 = A [15 A/2 notA]
| | | M1 = notA [9 A/11 notA]
| F = notA [16 A/42 notA]
| | M4 = A [9 A/6 notA]
| | | M5 = A [8 A/1 notA]
| | | M5 = notA [1 A/5 notA]
| | M4 = notA [7 A/36 notA]
| | | M2 = A [7 A/14, notA]
| | | M2 = notA [0 A/22 notA]
```

The numbers in brackets give the number of positive and negative labels from the training data in that part of the tree.

At this point, you should be able to go back and answer questions 9-15 in the ”Written Questions” section of this handout. Write your solutions in the template provided.

2.4 Submission Instructions [5pts]

Programming Please ensure you have completed the following files for submission.

```
inspection.{py|java|cpp|m}
decisionTree.{py|java|cpp|m}
```

When submitting your solution, make sure to select and upload both files. Ensure the files have the exact same spelling and letter casing as above.

Note: Please make sure the programming language that you use is consistent within this assignment (e.g. don’t use C++ for inspect and Python for decisionTree).

Written Questions Make sure you have completed all questions from Section 1 (including the collaboration policy questions) in the template provided. When you have done so, please submit your document in **pdf format** to the corresponding assignment slot on Gradescope.