CS 5350/6350: Machine Learning Fall 2023

Homework 1

Handed out: 3 Sep, 2024
Due: 11:59pm, 20 Sep, 2024

Name: Cameron Markovsky

1 Decision Tree [40 points + 10 bonus]

x_1	x_2	x_3	x_4	y
0	0	1	0	0
0	1	0	0	0
0	0	1	1	1
1	0	0	1	1
0	1	1	0.	0
1	1	0	0	0
0	1	0	1	0

Table 1: Training data for a Boolean classifier

1. [7 points] Decision tree construction.

   (a) [5 points] Use the ID3 algorithm with information gain to learn a decision tree from the training dataset in Table 1.

   For every split, the information gain is defined by the following:

$$Gain_{ent}(S, A) = entropy(S) \; - \sum_{v \in values(A)} \frac{|S_v|}{|S|} * entropy(S_v)$$

   where $entropy(S) = H(S) = -p_+ \log(p_+) - p_- \log(p_-)$

   i. **Split 1**
   Let S be the initial training data.

$$Gain_{ent}(S, A) \approx \begin{cases} 0.062 & \text{if } A = x_1 \\ 0.470 & \text{if } A = x_2 \\ 0.006 & \text{if } A = x_3 \\ 0.470 & \text{if } A = x_4 \end{cases}$$

Since splitting S on either x_2 and x_4 yields the maximum information gain, I choose to split on x_2.

Split 1: S split on x_2

$S_{x_2=0}$

x_1	x_3	x_4	y
0	1	0	0
0	1	1	1
1	0	1	1

$S_{x_2=1}$

x_1	x_3	x_4	y
0	0	0	0
0	1	0	0
1	0	0	0
0	0	1	0

Every label in $S_{x_2=1}$ is the same. Therefore, it is a leaf with label $y = 0$.

ii. **Split 2**

$$Gain_{ent}(S_{x_2=0}, A) \approx \begin{cases} 0.252 & \text{if } A = x_1 \\ 0.256 & \text{if } A = x_3 \\ 0.918 & \text{if } A = x_4 \end{cases}$$

Splitting $S_{x_2=0}$ on x_4 results in the maximum information gain.

Split 2: $S_{x_2=0}$ split on x_4

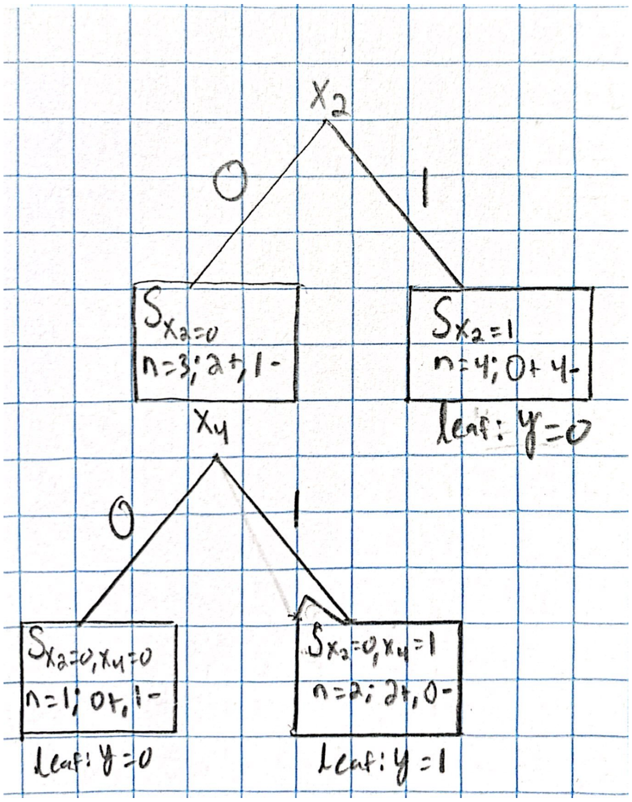
$S_{x_2=0,\ x_4=0}$

x_1	x_3	y
0	1	0

$S_{x_2=0,\ x_4=1}$

x_1	x_3	y
0	1	1
1	0	1

Every label in $S_{x_2=0,\ x_4=0}$ is the same. Therefore, it is a leaf with label $y = 0$.
Every label in $S_{x_2=0,\ x_4=1}$ is the same. Therefore, it is a leaf with label $y = 1$.



Full decision tree

(b) [2 points] Write the boolean function which your decision tree represents. Please use a table to describe the function — the columns are the input variables and label, i.e., x_1, x_2, x_3, x_4 and y; the rows are different input and function values.

x_1	x_2	x_3	x_4	y
1	1	1	1	0
1	1	1	0	0
1	1	0	1	0
1	1	0	0	0
1	0	1	1	1
1	0	1	0	0
1	0	0	1	1
1	0	0	0	0
0	1	1	1.	0
0	1	1	0	0
0	1	0	1	0
0	1	0	0	0
0	0	1	1.	1
0	0	1	0	0
0	0	0	1	1
0	0	0	0	0

Boolean function (truth table) that represents the decision tree.

2. [17 points] Let us use a training dataset to learn a decision tree about whether to play tennis (**Page 43, Lecture: Decision Tree Learning**, accessible by clicking the link http://www.cs.utah.edu/˜zhe/teach/pdf/3-decision-trees-learning.pdf). In the class, we have shown how to use information gain to construct the tree in ID3 framework.

   (a) [7 points] Now, please use majority error (ME) to calculate the gain, and select the best feature to split the data in ID3 framework. As in problem 1, please list every step in your tree construction, the attributes, how you calculate the gain of each attribute and how you split the dataset according to the selected attribute. Please also give a full structure of the tree.

   Let S be the initial data set provided.
   For every split, the information gain is defined by the following:

$$Gain_{ME}(S, A) = ME(S) \ - \sum_{v \in values(A)} \frac{|S_v|}{|S|} * ME(S_v)$$

   where $ME(S) = 1 - Max(p_+,\ p_-)$

   i. **Split 1:**

$$Gain_{ME}(S, A) = \begin{cases} \frac{1}{7} & \text{if } A = O \\ 0 & \text{if } A = T \\ \frac{1}{7} & \text{if } A = H \\ 0 & \text{if } A = W \end{cases}$$

Since splitting S on either O and H yields the maximum information gain, I choose to split on W.

Split 1: S split on W

$S_{W=W}$

O	T	H	P
S	H	H	0
O	H	H	1
R	M	H	1
R	C	N	1
S	M	H	0
S	C	N	1
R	M	N	1
O	H	N	1

$S_{W=S}$

O	T	H	P
S	H	H	0
R	C	N	0
O	C	N	1
S	M	N	1
O	M	H	1
R	M	H	0

ii. **Split 2:**

$$Gain_{ME}(S_{W=W}, A) = \begin{cases} \frac{1}{8} & \text{if } A = O \\ 0 & \text{if } A = T \\ 0 & \text{if } A = H \end{cases}$$

$$Gain_{ME}(S_{W=S}, A) = \begin{cases} \frac{1}{3} & \text{if } A = O \\ \frac{1}{6} & \text{if } A = T \\ \frac{1}{6} & \text{if } A = H \end{cases}$$

Splitting each subset, $S_{W=W}$ and $S_{W=S}$ on O results in the maximum information gain.

Split 2: $S_{W=W}$ split on O

$S_{W=W,\ O=S}$

T	H	P
H	H	0
M	H	0
C	N	1

$S_{W=W,\ O=R}$

T	H	P
M	H	1
C	N	1
M	N	1

$S_{W=W,\ O=O}$

T	H	P
H	H	1
H	N	1

Split 2: $S_{W=S}$ split on O

$S_{W=S,\ O=S}$

T	H	P
H	H	0
M	N	1

$S_{W=S,\ O=R}$

T	H	P
C	N	0
M	H	0

$S_{W=S,\ O=O}$

T	H	P
C	N	1
M	H	1

Every label in $S_{W=W,\ O=R}$ is the same. Therefore, it is a leaf with label $P = 1$.
Every label in $S_{W=W,\ O=O}$ is the same. Therefore, it is a leaf with label $P = 1$.
Every label in $S_{W=S,\ O=R}$ is the same. Therefore, it is a leaf with label $P = 0$.
Every label in $S_{W=S,\ O=O}$ is the same. Therefore, it is a leaf with label $P = 1$.

(b) **Split 3:**

$$Gain_{ME}(S_{W=W,\ O=S}, A) = \begin{cases} \frac{1}{2} & \text{if } A = T \\ \frac{1}{2} & \text{if } A = H \end{cases}$$

$$Gain_{ME}(S_{W=S,\ O=S}, A) = \begin{cases} \frac{1}{3} & \text{if } A = T \\ \frac{1}{3} & \text{if } A = H \end{cases}$$

Since splitting each subset on either T and H yields the maximum information gain, I choose to split on H.

Split 3: $S_{W=W,\ O=S}$ split on H

$S_{W=W,\ O=S,\ H=H}$

T	P
H	0
M	0

$S_{W=W,\ O=S,\ H=N}$

T	P
C	1

Split 3: $S_{W=S,\ O=S}$ split on H

$S_{W=S,\ O=S,\ H=H}$

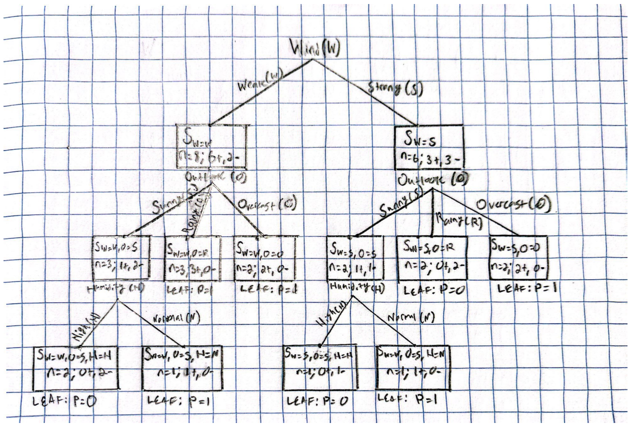
T	P
H	0

$S_{W=S,\ O=S,\ H=N}$

T	P
M	1

Every label in $S_{W=W,\ O=S,\ H=H}$ is the same. Therefore, it is a leaf with label $P = 0$.
Every label in $S_{W=W,\ O=S,\ H=N}$ is the same. Therefore, it is a leaf with label $P = 1$.

Every label in $S_{W=S,\ O=S,\ H=H}$ is the same. Therefore, it is a leaf with label $P = 0$.
Every label in $S_{W=S,\ O=S,\ H=N}$ is the same. Therefore, it is a leaf with label $P = 1$.



Full decision tree

(c) [7 points] Please use gini index (GI) to calculate the gain, and conduct tree learning with ID3 framework. List every step and the tree structure.

Let S be the initial data set provided.
For every split, the information gain is defined by the following:

$$Gain_{GI}(S, A) = GI(S) \ - \sum_{v \in values(A)} \frac{|S_v|}{|S|} * GI(S_v)$$

where $GI(S) = 1 - (p_+^2 + p_-^2)$

i. **Split 1:**

$$Gain_{GI}(S, A) \approx \begin{cases} 0.116 & \text{if } A = O \\ 0.019 & \text{if } A = T \\ 0.092 & \text{if } A = H \\ 0.031 & \text{if } A = W \end{cases}$$

Splitting S on O yields the maximum information gain.

Split 1: S split on O

$S_{O=S}$

T	H	W	P
H	H	W	0
H	H	S	0
M	H	W	0
C	N	W	1
M	N	S	1

$S_{O=R}$

T	H	W	P
M	H	W	1
C	N	W	1
C	N	S	0
M	N	W	1
M	H	S	0

$S_{O=O}$

T	H	W	P
H	H	W	1
C	N	S	1
M	H	S	1
H	N	W	1

Every label in $S_{O=O}$ is the same. Therefore, it is a leaf with label $P = 1$.

ii. **Split 2:**

$$Gain_{GI}(S_{O=S}, A) \approx \begin{cases} 0.280 & \text{if } A = T \\ 0.48 & \text{if } A = H \\ 0.013 & \text{if } A = W \end{cases}$$

$$Gain_{GI}(S_{O=R}, A) \approx \begin{cases} 0.013 & \text{if } A = T \\ 0.013 & \text{if } A = H \\ 0.48 & \text{if } A = W \end{cases}$$

Splitting $S_{O=S}$ on H and $S_{O=R}$ on W yields the maximum information gain for each subset.

Split 2: $S_{O=S}$ split on H

$S_{O=S,\ H=H}$

T	W	P
H	W	0
H	S	0
M	W	0

$S_{O=S,\ H=N}$

T	W	P
C	W	1
M	S	1

Split 2: $S_{O=R}$ split on W

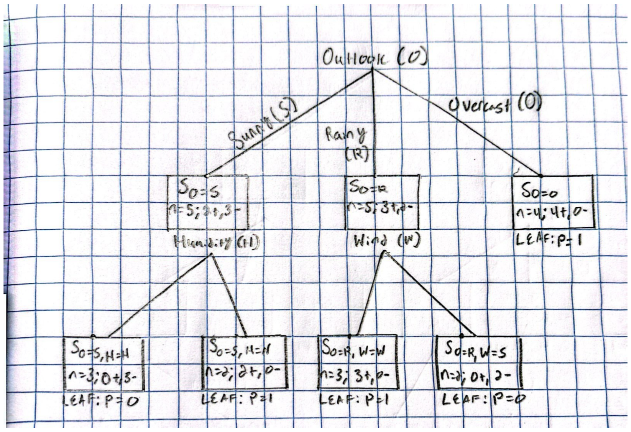
$S_{O=R,\ W=W}$

T	H	P
M	H	1
C	N	1
M	N	1

$S_{O=R,\ W=S}$

T	H	P
C	N	0
M	H	0

Every label in $S_{O=S,\ H=H}$ is the same. Therefore, it is a leaf with label $P = 0$.
Every label in $S_{O=S,\ H=N}$ is the same. Therefore, it is a leaf with label $P = 1$.
Every label in $S_{O=R,\ W=W}$ is the same. Therefore, it is a leaf with label $P = 1$.
Every label in $S_{O=R,\ W=S}$ is the same. Therefore, it is a leaf with label $P = 0$.



Full decision tree

(d) [3 points] Compare the two trees you just created with the one we built in the class (see Page 62 of the lecture slides). Are there any differences? Why?

3. [16 points] Continue with the same training data in Problem 2. Suppose before the tree construction, we receive one more training instance where Outlook's value is missing: {Outlook: Missing, Temperature: Mild, Humidity: Normal, Wind: Weak, Play: Yes}.

(a) [3 points] Use the most common value in the training data as the missing value, and calculate the information gains of the four features. Note that if there is a tie for the most common value, you can choose any value in the tie. Indicate the best feature.

For the Outlook feature, the most common values are sunny (S) and rainy (R), each with 5 instances. I choose to use the sunny value.
Let S be the new training data set and use the Gini Index to calculate gain, as defined above in problem 2c:

$$Gain_{GI}(S, A) \approx \begin{cases} 0.084 & \text{if } A = O \\ 0.021 & \text{if } A = T \\ 0.099 & \text{if } A = H \\ 0.037 & \text{if } A = W \end{cases}$$

Based on the calculated information gain, H is the best feature split on for the new training set.

(b) [3 points] Use the most common value among the training instances with the same label, namely, their attribute "Play" is "Yes", and calculate the information gains of the four features. Again if there is a tie, you can choose any value in the tie. Indicate the best feature.

For the Outlook feature, the most common value is O when $P = 1$, with 4 instances. Let S be the new training data set and use the Gini Index to calculate gain, as defined above in problem 2c:

$$Gain_{GI}(S, A) \approx \begin{cases} 0.124 & \text{if } A = O \\ 0.021 & \text{if } A = T \\ 0.099 & \text{if } A = H \\ 0.037 & \text{if } A = W \end{cases}$$

Based on the calculated information gain, O is now the best feature split on for the new training set.

(c) [3 points] Use the fractional counts to infer the feature values, and then calculate the information gains of the four features. Indicate the best feature.

Fractional Counts, Outlook

$$S\text{: } \frac{5}{14}$$
$$R\text{: } \frac{5}{14}$$
$$O\text{: } \frac{4}{14}$$

Using entropy, $H(S) = -p_+ \log(p_+) - p_- \log(p_-)$, to calculate gain:

$$\begin{aligned} entropy(S_{O=S}) &= -\frac{2+\frac{5}{14}}{5+\frac{5}{14}}\log(\frac{2+\frac{5}{14}}{5+\frac{5}{14}}) - \frac{3}{5+\frac{5}{14}}\log(\frac{3}{5+\frac{5}{14}})] \approx 0.990 \\ entropy(S_{O=R}) &= -\frac{3+\frac{5}{14}}{5+\frac{5}{14}}\log(\frac{3+\frac{5}{14}}{5+\frac{5}{14}}) - \frac{2}{5+\frac{5}{14}}\log(\frac{2}{5+\frac{5}{14}}) \approx 0.953 \\ entropy(S_{O=O}) &= -\frac{4+\frac{4}{14}}{4+\frac{4}{14}}\log(\frac{4+\frac{4}{14}}{4+\frac{4}{14}}) - \frac{0}{4+\frac{4}{14}}\log(\frac{0}{4+\frac{4}{14}}) = 0 \end{aligned}$$

$$\begin{aligned} Gain_{ent}(S,O) &= entropy(y) - (\frac{5}{14}entropy(S_{O=S}) + \frac{5}{14}entropy(S_{O=R}) + \frac{4}{14}entropy(S_{O=O})) \\ &= 0.918 - (\frac{5}{14}*0.990 + \frac{5}{14}*0.953 + \frac{4}{14}*0) \approx 0.224 \end{aligned}$$

$$Gain(S,A) = \begin{cases} 0.021 & \text{if } A = T \\ 0.099 & \text{if } A = H \\ 0.037 & \text{if } A = W \end{cases}$$

Based on the calculated information gain, O would be best to split on.

(d) [7 points] Continue with the fractional examples, and build the whole free with information gain. List every step and the final tree structure.

i. **Split 1:**

Split 1: S split on O

$S_{O=S}$

	T	H	W	P
	H	H	W	0
	H	H	S	0
	M	H	W	0
	C	N	W	1
	M	N	S	1
$\frac{5}{14}$	M	N	W	1

$S_{O=R}$

	T	H	W	P
	M	H	W	1
	C	N	W	1
	C	N	S	0
	M	N	W	1
	M	H	S	0
$\frac{5}{14}$	M	N	W	1

$S_{O=O}$

	T	H	W	P
	H	H	W	1
	C	N	S	1
	M	H	S	1
	H	N	W	1
$\frac{5}{14}$	M	N	W	1

Since every label in $S_{O=O}$ is the same, it is a leaf with value $P = 1$.

ii. **Split 2:**

$$ent(S_{O=S}) = -\frac{2+\frac{5}{14}}{5+\frac{5}{14}}\log(\frac{2+\frac{5}{14}}{5+\frac{5}{14}}) - \frac{3}{5+\frac{5}{14}}\log(\frac{3}{5+\frac{5}{14}})] \approx 0.990$$

$$\begin{aligned} ent(S_{O=S,\ T=H}) &= -\frac{0}{2}\log(\frac{0}{2}) - \frac{2}{2}\log(\frac{2}{2}) = 0 \\ ent(S_{O=S,\ T=M}) &= -\frac{1+\frac{5}{14}}{2+\frac{5}{14}}\log(\frac{1+\frac{5}{14}}{2+\frac{5}{14}}) - \frac{1}{2+\frac{5}{14}}\log(\frac{1}{2+\frac{5}{14}}) \approx 0.983 \\ ent(S_{O=S,\ T=C}) &= -\frac{1}{1}\log(\frac{1}{1}) - \frac{0}{1}\log(\frac{0}{1}) = 0 \\ Gain(S_{O=S}, T) &\approx 0.990 - \frac{2+\frac{5}{14}}{5+\frac{5}{14}} * 0.983 = 0.556 \end{aligned}$$

$$ent(S_{O=S,\ H=H}) = -\frac{0}{3}\log(\frac{0}{3}) - \frac{3}{3}\log(\frac{3}{3}) = 0$$
$$ent(S_{O=S,\ H=N}) = -\frac{2+\frac{5}{14}}{2+\frac{5}{14}}\log(\frac{2+\frac{5}{14}}{2+\frac{5}{14}}) - \frac{0}{2+\frac{5}{14}}\log(\frac{0}{2+\frac{5}{14}}) = 0$$
$$Gain(S_{O=S}, H) \approx 0.990 - 0 = 0.990$$

$$ent(S_{O=S,\ W=W}) = -\frac{1+\frac{5}{14}}{3+\frac{5}{14}}\log(\frac{1+\frac{5}{14}}{3+\frac{5}{14}}) - \frac{2}{3+\frac{5}{14}}\log(\frac{2}{3+\frac{5}{14}}) \approx 0.973$$
$$ent(S_{O=S,\ W=S}) = -\frac{1}{2}\log(\frac{1}{2}) - \frac{1}{2}\log(\frac{1}{2}) = 1$$
$$Gain(S_{O=S}, W) \approx 0.990 - (\frac{3+\frac{5}{14}}{5+5/14} * 0.973 + \frac{2}{5+\frac{5}{14}} * 1) \approx 0.006$$

Based on the calculated information gain, split $S_{O=S}$ on H which results in 2 leaf nodes.

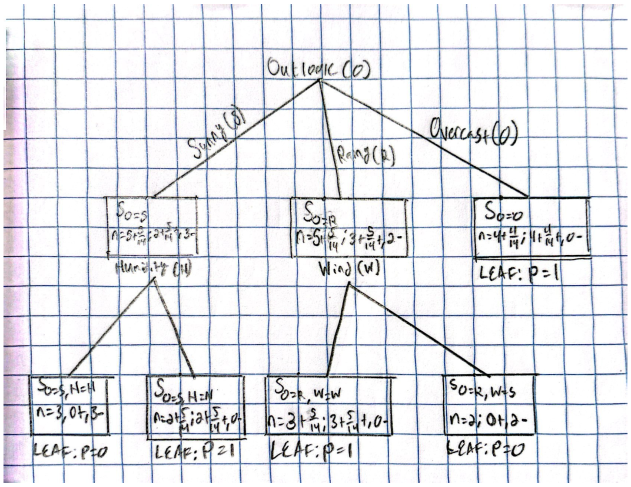
$$entropy(S_{O=R}) = -\frac{3+\frac{5}{14}}{5+\frac{5}{14}}\log(\frac{3+\frac{5}{14}}{5+\frac{5}{14}}) - \frac{2}{5+\frac{5}{14}}\log(\frac{2}{5+\frac{5}{14}}) \approx 0.953$$

$$ent(S_{O=R,\ T=M}) = -\frac{2+\frac{5}{14}}{3+\frac{5}{14}}\log(\frac{2+\frac{5}{14}}{3+\frac{5}{14}}) - \frac{1}{3+\frac{5}{14}}\log(\frac{1}{3+\frac{5}{14}}) \approx 0.879$$
$$ent(S_{O=R,\ T=C}) = -\frac{1}{2}\log(\frac{1}{2}) - \frac{1}{2}\log(\frac{1}{2}) = 1$$
$$Gain(S_{O=R}, T) \approx 0.953 - (\frac{3+\frac{5}{14}}{5+5/14} * 0.879 + \frac{2}{5+\frac{5}{14}} * 1) \approx 0.029$$

$$ent(S_{O=R,\ H=H}) = -\frac{1}{2}\log(\frac{1}{2}) - \frac{1}{2}\log(\frac{1}{2}) = 1$$
$$ent(S_{O=R,\ H=N}) = -\frac{2+\frac{5}{14}}{3+\frac{5}{14}}\log(\frac{2+\frac{5}{14}}{3+\frac{5}{14}}) - \frac{1}{3+\frac{5}{14}}\log(\frac{1}{3+\frac{5}{14}}) \approx 0.879$$
$$Gain(S_{O=R}, H) \approx 0.953 - (\frac{3+\frac{5}{14}}{5+5/14} * 0.879 + \frac{2}{5+\frac{5}{14}} * 1) \approx 0.029$$

$$ent(S_{O=S,\ W=W}) = -\frac{3+\frac{5}{14}}{3+\frac{5}{14}}\log(\frac{3+\frac{5}{14}}{3+\frac{5}{14}}) - \frac{0}{3+\frac{5}{14}}\log(\frac{0}{3+\frac{5}{14}}) = 0$$
$$ent(S_{O=S,\ W=S}) = -\frac{0}{2}\log(\frac{0}{2}) - \frac{2}{2}\log(\frac{2}{2}) = 0$$
$$Gain(S_{O=S}, W) \approx 0.990 - 0 = 0.990$$

Based on the calculated information gain, split $S_{O=R}$ on W which results in 2 leaf nodes.



Full decision tree

4. [**Bonus question 1**] [5 points]. Prove that the information gain is always non-negative. That means, as long as we split the data, the purity will never get worse! (Hint: use convexity)

5. [**Bonus question 2**] [5 points]. We have discussed how to use decision tree for regression (i.e., predict numerical values) — on the leaf node, we simply use the average of the (numerical) labels as the prediction. Now, to construct a regression tree, can you invent a gain to select the best attribute to split data in ID3 framework?

2 Decision Tree Practice [60 points]

1. [5 Points] Starting from this assignment, we will build a light-weighted machine learning library. To this end, you will first need to create a code repository in Github.com. Please refer to the short introduction in the appendix and the official tutorial to create an account and repository. Please commit a README.md file in your repository, and write one sentence: "This is a machine learning library developed by **Your Name** for CS5350/6350 in University of Utah". You can now create a first folder, "DecisionTree". Please leave the link to your repository in the homework submission. We will check if you have successfully created it.

2. [30 points] We will implement a decision tree learning algorithm for car evaluation task. The dataset is from UCI repository(`https://archive.ics.uci.edu/ml/datasets/car+evaluation`). Please download the processed dataset (car.zip) from Canvas. In this task, we have 6 car attributes, and the label is the evaluation of the car. The attribute and label values are listed in the file "data-desc.txt". All the attributes are categorical. The training data are stored in the file "train.csv", consisting of 1,000

examples. The test data are stored in “test.csv”, and comprise 728 examples. In both training and test datasets, attribute values are separated by commas; the file “data-desc.txt” lists the attribute names in each column.

(a) [15 points] Implement the ID3 algorithm that supports, information gain, majority error and gini index to select attributes for data splits. Besides, your ID3 should allow users to set the maximum tree depth. Note: you do not need to convert categorical attributes into binary ones and your tree can be wide here.

(b) [10 points] Use your implemented algorithm to learn decision trees from the training data. Vary the maximum tree depth from 1 to 6 — for each setting, run your algorithm to learn a decision tree, and use the tree to predict both the training and test examples. Note that if your tree cannot grow up to 6 levels, you can stop at the maximum level. Report in a table the average prediction errors on each dataset when you use information gain, majority error and gini index heuristics, respectively.

Training data

Depth	IG	GI	ME
1	0.626	0.626	0.85
2	0.302	0.302	0.652
3	0.331	0.336	0.594
4	0.294	0.298	0.588
5	0.299	0.299	0.598
6	0.302	0.302	0.6

Average prediction error for each gain metric

Test data

Depth	IG	GI	ME
1	0.628	0.628	0.864
2	0.297	0.297	0.633
3	0.323	0.335	0.571
4	0.253	0.271	0.541
5	0.297	0.297	0.567
6	0.299	0.299	0.567

Average prediction error for each gain metric

(c) [5 points] What can you conclude by comparing the training errors and the test errors?

Based on these results, you can conclude that this model is too simplistic. Since the error is similar for both the training and test sets, it is not "learning" enough about the data. Instead, it is generalized and performs similar across both datasets. In addition, we can tell that it is not overfitted since the error doesn't drastically increase on the test set.

3. [25 points] Next, modify your implementation a little bit to support numerical attributes. We will use a simple approach to convert a numerical feature to a binary one. We choose the media (NOT the average) of the attribute values (in the training set) as the threshold, and examine if the feature is bigger (or less) than the threshold. We will use another real dataset from UCI repository(`https://archive.ics.uci.edu/ml/datasets/Bank+Marketing`). This dataset contains 16 attributes, including both numerical and categorical ones. Please download the processed dataset from Canvas (bank.zip). The attribute and label values are listed in the file "data-desc.txt". The training set is the file "train.csv", consisting of $5,000$ examples, and the test "test.csv" with $5,000$ examples as well. In both training and test datasets, attribute values are separated by commas; the file "data-desc.txt" lists the attribute names in each column.

   (a) [10 points] Let us consider "unknown" as a particular attribute value, and hence we do not have any missing attributes for both training and test. Vary the maximum tree depth from 1 to 16 — for each setting, run your algorithm to learn a decision tree, and use the tree to predict both the training and test examples. Again, if your tree cannot grow up to 16 levels, stop at the maximum level. Report in a table the average prediction errors on each dataset when you use information gain, majority error and gini index heuristics, respectively.
   I implemented the $conv_numerics$ function to convert these features to categorical ones. It also stores the median value so that it can predict later.

Training data

Depth	IG	GI	ME
1	0.119	0.109	0.109
2	0.106	0.104	0.104
3	0.105	0.103	0.102
4	0.110	0.108	0.109
5	0.115	0.115	0.113
6	0.117	0.117	0.115
7	0.117	0.117	0.116
8	0.118	0.116	0.116
9	0.118	0.119	0.116
10	0.118	0.119	0.116
11	0.118	0.118	0.116
12	0.118	0.118	0.116
13	0.118	0.118	0.116
14	0.118	0.118	0.116
15	0.118	0.118	0.116
16	0.118	0.118	0.116

Average prediction error for each gain metric

Test data

Depth	IG	GI	ME
1	0.125	0.117	0.117
2	0.111	0.109	0.109
3	0.110	0.118	0.118
4	0.134	0.136	0.135
5	0.154	0.154	0.146
6	0.165	0.167	0.149
7	0.175	0.178	0.151
8	0.182	0.184	0.151
9	0.184	0.185	0.151
10	0.186	0.188	0.151
11	0.187	0.191	0.151
12	0.187	0.191	0.151
13	0.187	0.191	0.151
14	0.187	0.191	0.151
15	0.187	0.191	0.151
16	0.187	0.191	0.151

Average prediction error for each gain metric

(b) [10 points] Let us consider ”unknown” as attribute value missing. Here we simply complete it with the majority of other values of the same attribute in the training set. Vary the maximum tree depth from 1 to 16 — for each setting, run your algorithm to learn a decision tree, and use the tree to predict both the training and test examples. Report in a table the average prediction errors on each dataset when you use information gain, majority error and gini index heuristics, respectively.

Training data

Depth	IG	GI	ME
1	0.119	0.855	0.855
2	0.442	0.852	0.852
3	0.447	0.856	0.854
4	0.485	0.863	0.861
5	0.488	0.866	0.864
6	0.490	0.866	0.865
7	0.489	0.866	0.865
8	0.489	0.865	0.865
9	0.490	0.866	0.865
10	0.490	0.866	0.865
11	0.490	0.866	0.865
12	0.490	0.866	0.865
13	0.490	0.866	0.865
14	0.490	0.866	0.865
15	0.490	0.866	0.865
16	0.490	0.866	0.865

Average prediction error for each gain metric

Training data

Depth	*IG*	*GI*	*ME*
1	0.124	0.852	0.852
2	0.435	0.845	0.846
3	0.440	0.851	0.854
4	0.495	0.865	0.866
5	0.504	0.875	0.869
6	0.509	0.878	0.869
7	0.514	0.880	0.869
8	0.514	0.881	0.869
9	0.515	0.882	0.869
10	0.515	0.882	0.869
11	0.515	0.882	0.869
12	0.515	0.882	0.869
13	0.515	0.882	0.869
14	0.515	0.882	0.869
15	0.515	0.882	0.869
16	0.515	0.882	0.869

Average prediction error for each gain metric

(c) [5 points] What can you conclude by comparing the training errors and the test errors, with different tree depths, as well as different ways to deal with "unknown" attribute values?

We can conclude that this model is more adequately trained since we see a slight decrease in performance on the test set but over good performance across both the training and test set. As tree depth increases, the error tends to steady out, meaning that more depth doesn't necessarily increase performance. I would have expected to see a peak in performance at around depth 4-6, and then a gradual decrease after that, but either limitations in the data or implementation did not yield this. Filling unknown values with the most common value drastically decreased performance. This may be because one of the features had nearly all unknown values and this led the model to get false information from this filling.

Appendix

What is GitHub?

You may have contacted with GitHub long before you realized its existence, since a large part of open source code reside in GitHub nowadays. And whenever you google for some open source code snap, like code for a course project or a research paper, you would possibly be directed to GitHub.

GitHub, as well as many of its competitor like GitLab and BitBucket, is a so-called code hosting website, to which you upload and manage your code. For many first time user of GitHub, it's quite confusing that there is another software called git. Don't be confused now, git is a version control software and is the core of all these website. It can help you track the development of your code and manage them in a very organized way. Git works on your own local computer as other normal softwares do. Github, however, is just a website, or by its name, a hub, that you keep the code, just like a cloud storage space. How do we upload our code to GitHub? Yes, by Git!(or its variants). They are so dependent that when people say using GitHub, they mean they use git to manage their code, and keep their code in GitHub. That been said, as a stand-alone tool, git could work perfectly on your local computer without Internet access, as long as your do not want to keep your code on-line and access them everywhere, or share them with others.

Core concepts and operations of GitHub

Here we only state the basic concepts of GitHub and git. Specific commands vary slightly depending on the Platforms (Mac/Linux/WIN10) and command-line/GUI versions. Please refer to the link provided below for concrete examples and video tutorials. As you understand the whole working flow, those commands should be easy and straightforward to use.

There are two major parts we need to know about github. The on-line part of GitHub and local part of git. We start from GitHub.

GitHub

If you have never had a GitHub account, please follow this link to create one. It also provides tutorial on basic operations of GitHub.

https://guides.github.com/activities/hello-world/

Note that now you can create a private repository without paying to GitHub. In principle, we encourage you to create public repository. But if you somehow prefer a private one(i.e., can't be access by others), you must add TA's account as the collaborators in order to check your work.

These are some key concept you should know about:

- Repository: Repository is the place where you keep your whole project, including every version, every branch of the code. For example, you will need to create a repository named Final-Project (or other suitable name), which will contain all your code, report and results.

- Branch: Branch allows you (and your partners) to developed different version of a repository at the same time. For example, you and your partner are working on the final project. Suddenly, you want to try some crazy algorithm but not sure if it would work. Now you create a new branch of the repository and continue your trying without breaking the original (usually called master) branch. If successful, you then merge this branch with the master branch. Otherwise, you can simply give up and delete this branch, and nothing in the master branch will be affected.

- Pull Request: This is the heart of GitHub. Don't mistake this with PULL we will talk about later. Pull Request means when you finish your branch(like the crazy algorithm above), you make a request that the owner or manager of master branch to review your code, and merge them into the master branch. If accepted, any changes you make in your branch will also be reflected in the master branch. As the manager of master branch, you should also be careful to check the code about to be merged and address any potential conflicts this merge may introduce.

- Merge: This operation means to merge two different branches into a single one. Any inconsistency must be addressed before merging.

git

This link provides installation guides and video tutorial for basic git operation.
https://git-scm.com/doc

This is a cheating-sheet for common commands of git.
https://confluence.atlassian.com/bitbucketserver/basic-git-commands-776639767.html

As said before, you can always use git in a local repository. More frequently, we link this repository to the one you create in GitHub and then you can use git to push (upload) you code to and pull (fetch) them from GitHub. Beside the original command-line interface, there are many softwares with nice GUI to access the functionalities of git, such as GitHub Desktop and Source Tree.

There are also some core operations you should know about:

- clone: Download/Copy a repository from GitHub to your local disk. This would fetch everything of this repository. This is the most commonly used command to download someone else's code from GitHub.

- init: Initialize current fold a to local repository, which will generate some hidden files to track the changes.

- add: By default, no files in the repository folder are marked to be tracked. When you want to track the change of a file, use add operation to add this file to the tracking list. Normally, we only track the source code and report of our project, and we DON'T track datasets. As the datasets never change once downloaded and are usually big.

- commit: This is the most frequently used git operation. Commit means to make a LOCALLY check point. For example, you have done some change to the project, like adding a new complex function, and it works well. Then you can commit with a comment "adding new function, test well ***". Later when you try to modify this function but fail, you can roll back to this check point and start over. Hence you do not need to many copies before modification.

- checkout: After you commit checkpoints, you can use checkout to roll back to these checkpoints in case you mess up.
- push: When you complete current task and make check very thing is good, you use push(after commit) to upload the local repository to GitHub.
- pull: Fetch the content from GitHub. This is similar to Clone. But it only fetches content designated by the parameters to the pull command.

Work Flow

With concepts and operations introduced above, the work flow of using GitHub for a project is as follows:

1. Create a repository in GitHub.
2. Create a local repository in your local computer and link it to the remote repository in GitHub.
3. Create source code files and add them to the tracking list.
4. Edit, modify and test your code. Commit and checkout whenever mess up.
5. Push your code to GitHub.

If you start your work with an existed GitHub repository (like the one created by your partner), Just replace steps 1 to 3 by pull or clone.

You can play around with GitHub by creating some random repositories and files to track. Basic operation introduced above and in the links are more than enough to complete this course. If are you have further interest to master GitHub, there are several excellent on-line courses provided by Coursera and Udacity. Many tutorials are provided in the web as well.