CS 5350/6350: Machine Learining Fall 2023

Homework 1

Milena Belianovich
09/22/2023

- You are welcome to talk to other members of the class about the homework. I am more concerned that you understand the underlying concepts. However, you should write down your own solution. Please keep the class collaboration policy in mind.
- Feel free discuss the homework with the instructor or the TAs.
- Your written solutions should be brief and clear. You need to show your work, not just the final answer, but you do *not* need to write it in gory detail. Your assignment should be **no more than 20 pages**. Not that you do not need to include the problem description. Every extra page will cost a point.
- Handwritten solutions will not be accepted.
- *Your code should run on the CADE machines.* You should include a shell script, `run.sh`, that will execute your code in the CADE environment. Your code should produce similar output to what you include in your report.

  You are responsible for ensuring that the grader can execute the code using only the included script. If you are using an esoteric programming language, you should make sure that its runtime is available on CADE.
- Please do not hand in binary files! We will *not* grade binary submissions.
- The homework is due by **midnight of the due date**. Please submit the homework on Canvas.
- Note the bonus questions are for **both 5350 and 6350** students. If a question is mandatory for 6350, we will highlight it explicitly.

1 Decision Tree [40 points + 10 bonus]

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. Please list every step in your tree construction, including the data subsets, the attributes, and how you calculate the information gain of each attribute and how you split the dataset according to the selected attribute. Please also give a full structure of the tree. You can manually draw the tree structure, convert the picture into a PDF/EPS/PNG/JPG format and include it in your homework submission; or instead, you can represent the tree with a conjunction of prediction rules as we discussed in the lecture.

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

Solution: To construct a decision tree using the ID3 algorithm with information gain, we need to follow a step-by-step process that involves selecting the best attribute to split the dataset and recursively building the tree.
First, calculate the entropy of the target variable (y) for the entire dataset.
$Entropy(y) = -p(0) * log_2(p(0)) - p(1) * log_2(p(1))$
Where:
• $p(0)$ is the probability of $y = 0$ in the dataset.
• $p(1)$ is the probability of $y = 1$ in the dataset.
We know the following: $p = 2/7$ and $n = 5/7$. Therefore, $p(0) = 5/7$, $p(1) = 2/7$.
Then, $Entropy(y) = -(5/7)log_2(5/7) - (2/7)log_2(2/7) \approx 0.863$.
Now, we can calculate the information gain for each attribute (x_1, x_2, x_3, x_4) and split the dataset based on that attribute.
x_1: $x_1 = 0$: 5/7 examples
$p = 1/5$, $n = 4/5$
$Entropy(x_1 = 0) = -\frac{1}{5}(log_2(\frac{1}{5})) - \frac{4}{5}(log_2(\frac{4}{5})) = 0.7219$
$x_1 = 1$: 2/7 examples
$p = 1/2$, $n = 1/2$
$Entropy(x_1 = 1) = -\frac{1}{2}(log_2(\frac{1}{2})) - \frac{1}{2}(log_2(\frac{1}{2})) = 1$
$Entropy(x_1) = \frac{5}{7}0.7219 + \frac{2}{7}1 = 0.8014$
$IG = 0.8631 - 0.8014 = 0.0617$
x_2: $x_2 = 0$: 3/7 examples
$p = 2/3$, $n = 1/3$
$Entropy(x_2 = 0) = -\frac{2}{3}(log_2(\frac{2}{3})) - \frac{1}{3}(log_2(\frac{1}{3})) = 0.9183$
$x_2 = 1$: 4/7 examples
$p = 0$, $n = 1$
$Entropy(x_2 = 1) = -0 - 1(log_2(1)) = 0$
$Entropy(x_2) = \frac{4}{7}0 + \frac{3}{7}0.9183 = 0.3936$
$IG = 0.8631 - 0.3936 = 0.4695$
x_3: $x_3 = 0$: 4/7 examples
$p = 1/4$, $n = 3/4$
$Entropy(x_3 = 0) = -\frac{1}{4}(log_2(\frac{1}{4})) - \frac{3}{4}(log_2(\frac{3}{4})) = 0.8113$
$x_1 = 1$: 3/7 examples
$p = 1/3$, $n = 2/3$

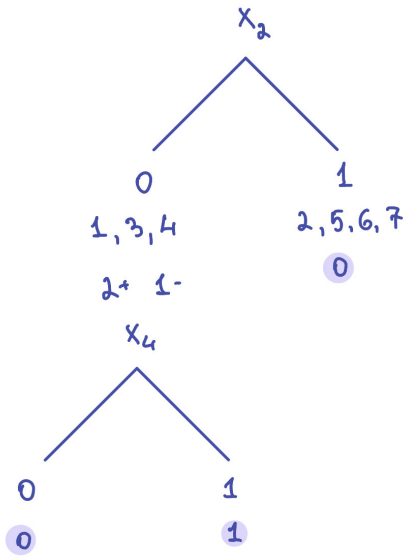
$Entropy(x_3 = 1) = -\frac{1}{3}(log_2(\frac{1}{3})) - \frac{2}{3}(log_2(\frac{2}{3})) = 0.9183$
$Entropy(x_3) = \frac{4}{7}0.8113 + \frac{3}{7}0.9183 = 0.8572$
$IG = 0.8631 - 0.8572 = 0.0059$
x_4: $x_4 = 0$: 4/7 examples
$p = 0$, $n = 1$
$Entropy(x_4 = 0) = 0$
$x_1 = 1$: 3/7 examples
$p = 2/3$, $n = 1/3$
$Entropy(x_4 = 1) = -\frac{1}{3}(log_2(\frac{1}{3})) - \frac{2}{3}(log_2(\frac{2}{3})) = 0.9183$
$Entropy(x_3) = \frac{4}{7}0 + \frac{3}{7}0.9183 = 0.3936$
$IG = 0.8631 - 0.3936 = 0.4695$
Now, we can compare the Information Gains for each attribute and select the one with the highest Information Gain as the root of the decision tree. According to the resuklts above, we can see that x_2 and x_4 have the same IG values, so we can pick x_2 as the root of the decision tree.
We are left with the subtree that needs splitting, a table for which can bee seen in table 2. We can first calculate the entropy of the target variable (y) for the

Num.	x_1	x_2	x_3	x_4	y
1	0	0	1	0	0
3	0	0	1	1	1
4	1	0	0	1	1

Table 2: Remaining data after the first split

remaining dataset. We know the following: $p = 2/3$ and $n = 1/3$. Therefore:
$Entropy(y) = -\frac{1}{3}(log_2(\frac{1}{3})) - \frac{2}{3}(log_2(\frac{2}{3})) \approx 0.9183$.
Now, we can calculate the information gain for each of the attributes left (x_1, x_3, x_4) and split the dataset based on that attribute.
x_1: $x_1 = 0$: 2/3 examples
$p = 1/2$, $n = 1/2$
$Entropy(x_1 = 0) = -\frac{1}{2}(log_2(\frac{1}{2})) - \frac{1}{2}(log_2(\frac{1}{2})) = 1$
$x_1 = 1$: 1/3 examples
$p = 0$, $n = 1/3$
$Entropy(x_1 = 1) = -1(log_2(1)) = 0$
$Entropy(x_1) = \frac{2}{3}1 + \frac{1}{3}0 = 0.6667$
$IG = 0.9183 - 0.6667 = 0.2516$
x_3: $x_3 = 0$: 1/3 examples
$p = 1$, $n = 0$
$Entropy(x_1 = 0) = 0$
$x_1 = 1$: 2/3 examples
$p = 1/2$, $n = 1/2$
$Entropy(x_1 = 1) = -\frac{1}{2}(log_2(\frac{1}{2})) - \frac{1}{2}(log_2(\frac{1}{2})) = 1$
$Entropy(x_1) = \frac{2}{3}1 + \frac{1}{3}0 = 0.6667$
$IG = 0.9183 - 0.6667 = 0.2516$

x_3: $x_3 = 0$: 1/3 examples
$p = 0$, $n = 1$
$Entropy(x_1 = 0) = -1(log_2(1)) = 0$
$x_1 = 1$: 2/3 examples
$p = 1$, $n = 0$
$Entropy(x_1 = 1) = -1(log_2(1)) = 0$
$Entropy(x_1) = \frac{1}{3}0 + \frac{2}{3}0 = 0$
$IG = 0.9183 - 0 = 0.9183$
According to the calculations of IG, the attribute with the highest IG is x_4. Therefore, the next split happens at x_4, forming the decision tree as seen below.



(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. ‘

Solution:
The decision tree in part (a) can be represented as a table below (table 3), where * can be replaced by either 0 or 1. (In order to save space the * was used.)

x_1	x_2	x_3	x_4	y
*	0	*	0	0
*	0	*	1	1
*	1	*	*	1

Table 3: Boolean representation

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/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.

Solution:

The dataset contains the following attributes: Outlook (O), Temperature (T), Humidity (H), Wind (W), and the target attribute Play. First, we can calculate the majority error for the target attribute Play and determine the majority class. The majority class in Play is No or - (since it appears more frequently). Now, we can select the attribute with the highest ME gain as the root node. First, calculate the ME for each attribute (O, T, H, W).

ME for Outlook (O):

ME(Sunny) = Majority Error(-, -, +) = 2/3

ME(Overcast) = Majority Error(-, +, +) = 1/3

ME(Rainy) = Majority Error(-, +, -, -) = 2/4 = 1/2

Weighted Average ME for Outlook (O):

P(O=S) = 3/14; P(O=O) = 4/14; P(O=R) = 7/14

ME(O) = [P(O=S) * ME(Sunny)] + [P(O=O) * ME(Overcast)] + [P(O=R) * ME(Rainy)]

ME(O) = (3/14) * (2/3) + (4/14) * (1/3) + (7/14) * (1/2) = 6/42 + 4/42 + 7/28 = 17/42

ME for Temperature (T):

ME(Hot) = Majority Error(-, -, +, +, +) = 3/5

ME(Medium) = Majority Error(-, -, +) = 2/3

ME(Cool) = Majority Error(-, +, +) = 1/3

Weighted Average ME for Temperature (T):

P(T=H) = 5/14; P(T=M) = 3/14; P(T=C) = 6/14

ME(T) = [P(T=H) * ME(Hot)] + [P(T=M) * ME(Medium)] + [P(T=C) * ME(Cool)]

ME(T) = (5/14) * (3/5) + (3/14) * (2/3) + (6/14) * (1/3) = 15/70 + 6/42 + 6/42 = 27/70

ME for Humidity (H):

ME(High) = Majority Error(-, -, +, +, +) = 3/5

ME(Normal) = Majority Error(-, -, -, +, +) = 2/5

ME(Low) = Majority Error(-, +, +) = 1/3

Weighted Average ME for Humidity (H):

P(H=H) = 5/14; P(H=N) = 5/14; P(H=L) = 4/14

ME(H) = [P(H=H) * ME(High)] + [P(H=N) * ME(Normal)] + [P(H=L) * ME(Low)]

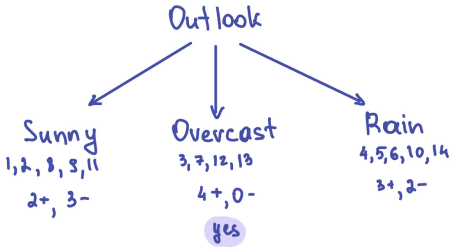
ME(H) = (5/14) * (3/5) + (5/14) * (2/5) + (4/14) * (1/3) = 15/70 + 10/70 + 4/42 = 29/70

ME for Wind (W):

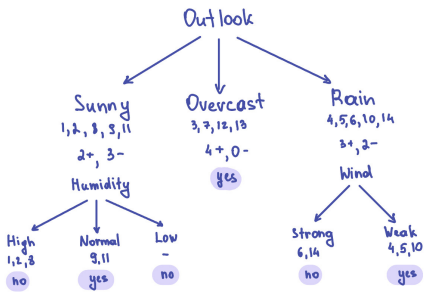
ME(Strong) = Majority Error(-, -, -, +, +, +) = 3/6 = 1/2

ME(Weak) = Majority Error(-, -, +, +, +) = 2/5

Weighted Average ME for Wind (W):
P(W=S) = 6/14; P(W=W) = 8/14
ME(W) = [P(W=S) * ME(Strong)] + [P(W=W) * ME(Weak)]
ME(W) = (6/14) * (1/2) + (8/14) * (2/5) = 3/14 + 16/70 = 41/70
Now, we select the attribute with the lowest ME, which is "Outlook" (O) with ME(O) = 17/42, as the root of the decision tree. This is our first split.



We start with the root node "Outlook (O)," which has three possible values: Sunny, Overcast, and Rainy.
Split on Sunny (O = Sunny)
For the first branch, we consider the subset of data where "Outlook" is Sunny. In this subset, we focus on the "Humidity (H)" attribute to make the next split decision. We calculate the Majority Error (ME) for each unique value of "Humidity (H)" for this subset.
ME(High) = Majority Error(-, -) = 0/2 = 0 (All instances are negative)
ME(Normal) = Majority Error(+) = 0 (All instances are positive)
ME(Low) = Majority Error(-) = 0 (All instances are negative)
Since all ME values for "Humidity (H)" are 0, we don't need further splits in this branch.
Split on Overcast (O = Overcast)
For the second branch, we consider the subset of data where "Outlook" is Overcast. In this subset, we have a pure subset where all instances are positive (Play = Yes). We don't need further splits in this branch since it's already pure.
Split on Rainy (O = Rainy)
For the third branch, we consider the subset of data where "Outlook" is Rainy. In this subset, we focus on the "Wind (W)" attribute to make the next split decision. We calculate the Majority Error (ME) for each unique value of "Wind (W)" for this subset.
ME(Strong) = Majority Error(-, -, -) = 0/3 = 0 (All instances are negative)
ME(Weak) = Majority Error(+, +) = 0/2 = 0 (All instances are positive)
Since all ME values for "Wind (W)" are 0, we don't need further splits in this branch.
The decision tree is now fully constructed with the following structure:



(b) [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.

Solution:

First, we calculate the Gini Index (GI) for the target attribute Play and determine the attribute with the lowest Gini Index.

Gini Index for Play:

$GI(\text{Play}) = 1 - [P(\text{Yes})^2 + P(\text{No})^2]$

$GI(\text{Play}) = 1 - [(9/14)^2 + (5/14)^2] = 0.4592$

We can now select the attribute with the lowest Gini Index as the root node by calculating the Gini Index for each attribute (O, T, H, W).

For Outlook (O):

Calculate the Gini Index for each unique value of O (S, O, R).

For Sunny (S):

Count(Play=Yes) = 2; Count(Play=No) = 3; Total instances (Sunny) = 2+3 = 5

Gini(Sunny) $= 1 - [(2/5)^2 + (3/5)^2] = 0.48$

For Overcast (O):

Count(Play=Yes) = 4; Count(Play=No)= 0; Total instances (Overcast) = 4+0 = 4

Gini(Overcast) $= 1 - [(4/4)^2 + (0/4)^2] = 0.00$

For Rainy (R):

Count(Play=Yes) = 3; Count(Play=No) = 2; Total instances (Rainy) = 3+2 = 5

Gini(Rainy) $= 1 - [(3/5)^2 + (2/5)^2] = 0.48$

Calculate the weighted average Gini Index for Outlook (O):

$P(O = S) = 5/14$; $P(O = O) = 4/14$; $P(O = R) = 5/14$

$GI(O)$ = Weighted average of Gini Index for each value of O.

$GI(O) = [P(O = S) * Gini(S)] + [P(O = O) * Gini(O)] + [P(O = R) * Gini(R)]$

$GI(O) = [(5/14) * 0.48] + [(4/14) * 0.00] + [(5/14) * 0.48] = (2.14 + 0.00 + 2.14)/3 = 4.28/3 \approx 1.43$

For Temperature (T):

Calculate the Gini Index for each unique value of T (H, M, C).

For Hot (H):

Count(Play=Yes) = 2; Count(Play=No) = 2; Total instances (Hot) = 2 + 2 = 4

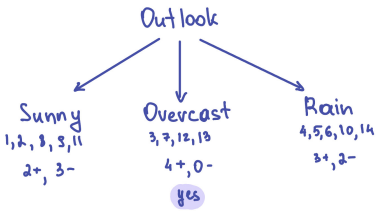
Gini(Hot)= $1 - [(2/4)^2 + (2/4)^2] = 0.50$

For Medium (M):

Count(Play=Yes) = 4; Count(Play=No) = 1; Total instances (Medium) = 4+1 = 5

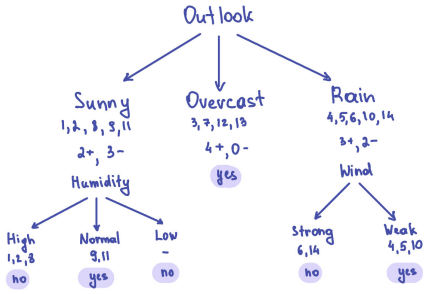
Gini(Medium) $= 1 - [(4/5)^2 + (1/5)^2] = 0.32$

For Cool (C):
Count(Play=Yes) = 3; Count(Play=No) = 2; Total instances (Cool) = $3+2=5$
Gini(Cool) = $1-[(3/5)^2+(2/5)^2]=0.48$
Calculate the weighted average Gini Index for Temperature (T):
P($T=H$) = 4/14; $P(T=M)=5/14$; $P(T=C)=5/14$
$GI(T)$ = Weighted average of Gini Index for each value of T.
$GI(T)=[P(T=H)*Gini(H)]+[P(T=M)*Gini(M)]+[P(T=C)*Gini(C)]$
$GI(T)=[(4/14)*0.50]+[(5/14)*0.32]+[(5/14)*0.48]=(0.50+0.1142857+0.1714286)/3=0.7857143/3\approx 0.262$
For Humidity (H):
Calculate the Gini Index for each unique value of H (H, N, L).
For High (H):
Count(Play=Yes) = 3; Count(Play=No) = 4; Total instances (High) = $3+4=7$
Gini(High) = $1-[(3/7)^2+(4/7)^2]=0.4898$
For Normal (N):
Count(Play=Yes) = 6; Count(Play=No) = 1; Total instances (Normal)= 6+1 = 7
Gini(Normal)= $1-[(6/7)^2+(1/7)^2]=0.2449$
For Low (L):
Count(Play=Yes) = 0; Count(Play=No) = 0; Total instances (Low) = $0+0=0$
Gini(Low) = 0
Calculate the weighted average Gini Index for Humidity (H):
$P(H=H)=7/14$; $P(H=N)=7/14$;$P(H=L)=0/14$
GI(H) = [P(H=H) * Gini(H)] + [P(H=N) * Gini(N)] + [P(H=L) * Gini(L)]
$GI(H)=[(7/14)*0.4898]+[(7/14)*0.2449]+[(0/14)*0]=0.4898/2\approx 0.2449$
For Wind (W):
Calculate the Gini Index for each unique value of W (S, W).
For Strong (S):
Count(Play=Yes) = 3; Count(Play=No) = 3; Total instances (Strong) = 3+3 = 6
Gini(Strong) = $1-[(3/6)^2+(3/6)^2]=0.5$
For Weak (W):
Count(Play=Yes) = 6; Count(Play=No) = 2; Total instances (Weak) = $6+2=8$
Gini(Weak) = $1-[(6/8)^2+(2/8)^2]=0.375$
Calculate the weighted average Gini Index for Wind (W):
$P(W=S)=6/14$; $P(W=W)=8/14$ GI(W) = [P(W=S) * Gini(S)] + [P(W=W) * Gini(W)]
$GI(W)=[(6/14)*0.5]+[(8/14)*0.375]=0.4285714/2\approx 0.2143$
The Outlook (O) attribute has the lowest Gini Index (0.3429). So, we select Outlook (O) as the root attribute.



Split the Data:

Split the dataset into subsets based on the values of Outlook (O):
Subset Sunny: Outlook = Sunny
Subset Overcast: Outlook = Overcast
Subset Rain: Outlook = Rain
For Subset S (Outlook = S):
Split Subset S based on Humidity (H):
Subset High: Humidity = High, Subset Normal: Humidity = Normal, Subset Low: Humidity = Low.
For Subset High (Humidity = High):
The majority class is Play: No (since it appears more frequently). Therefore, we can create a leaf node with the label "No" for this subset.
For Subset Normal (Humidity = Normal):
The majority class is Play: Yes (since it appears more frequently). Therefore, we can create a leaf node with the label "Yes" for this subset.
For Subset Low (Humidity = Low):
The majority class is Play: No (since it appears more frequently). Therefore, we can create a leaf node with the label "No" for this subset.
For Subset O (Outlook = O):
The majority class is Play: Yes (since it appears more frequently). Therefore, we can create a leaf node with the label "Yes" for this subset.
For Subset R (Outlook = R):
Split Subset R based on Wind (W):
Subset Strong: Wind = Strong, Subset Weak: Wind = Weak
For Subset Strong (Wind = Strong):
The majority class is Play: No (since it appears more frequently). Therefore, we can create a leaf node with the label "No" for this subset.
For Subset Weak (Wind = Weak):
The majority class is Play: Yes (since it appears more frequently). Therefore, we can create a leaf node with the label "Yes" for this subset.
The correct tree structure for the Gini Index approach will look like the following:



(c) [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?
Solution:
The rees acquired in parts (a) and (b) are the same as presented in class. Hoiwever, in the case of splitting at Humidity instead of Outlook when using the ME method, the decision tree would look different, but nevertheless have the same depth.

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.

   Solution:

   To handle the missing value for the "Outlook" attribute in the new training instance, we can use the most common value in the training data. First, let's calculate the Most Common Value (MCV) for the "Outlook" attribute in the training data:

   "Outlook" values in the training data: [S, S, O, R, R, R, O, S, S, R, S, O, O, R]
   Count of "Sunny" (S): 5
   Count of "Overcast" (O): 4
   Count of "Rainy" (R): 5
   Since "Sunny" and "Rainy" both have a count of 5, we can choose either one as the MCV. Let's choose "Sunny" as the MCV.
   Now, we have the following new instance with the missing "Outlook" value replaced by "Sunny":
   Outlook: Sunny, Temperature: Mild, Humidity: Normal, Wind: Weak, Play: Yes
   Now, we can calculate the information gain for all four variables (O, T, H, W).
   Target attribute Play: $p = 10/15$, $n = 5/15$
   $Entropy(P) = -\frac{10}{15}(log_2(\frac{10}{15})) - \frac{5}{15}(log_2(\frac{5}{15})) \approx 0.9183$
   Outlook (O):
   O=S: 6/15, $p = 1/2$, $n = 1/2$. $Entropy(O = S) = 1$
   O=O: 4/15, $p = 1$, $n = 0$. $Entropy(O = O) = 0$
   O=R: $5/15 = 1/3$, $p = 3/5$, $n = 2/5$. $Entropy(O = R) \approx 0.971$
   $Entropy(O) = \frac{6}{15}1 + \frac{4}{15}0 + \frac{1}{3}0.971 \approx 0.7237$
   $IG(O) = 0.9183 - 0.7237 = 0.1946$
   Temperature (T):
   T=H: 4/15, $p = 2/4 = 1/2$, $n = 2/4 = \text{`}/2$. $Entropy(T = H) = 1$
   T=M: 7/15, $p = 5/7$, $n = 2/7$. $Entropy(T = M) \approx 0.8631$
   T=C: 4/15, $p = 3/4$, $n = 1/4$. $Entropy(T = C) \approx 0.8113$
   $Entropy(T) \approx 0.8858$
   $IG(T) = 0.9183 - 0.8858 = 0.0325$
   Humidity (H):
   H=H: 7/15, $p = 3/7$, $n = 4/7$. $Entropy(H = H) \approx 0.9852$
   H=N: 8/15, $p = 7/8$, $n = 1/8$. $Entropy(H = N) \approx 0.5436$
   H=L: 0/15
   $Entropy(H) \approx 0.7497$
   $IG(H) = 0.1686$
   Wind (W):
   W=S: 6/15, $p = 1/2$, $n = 1/2$. $Entropy(H = H) = 1$

W=W: 9/15, $p = 7/9$, $n = 2/9$. $Entropy(T = M) \approx 0.7642$
$Entropy(W) \approx 0.8585$
$IG(W) = 0.0598$
Therefore, Outlook still has the highest IG value.

(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.

Solution:

To handle the missing value for the "Outlook" attribute in the new training instance, we can use the most common value in the training data. First, let's calculate the Most Common Value (MCV) for the "Play = Yes" attribute in the training data:
Count of "Sunny" (S): 2
Count of "Overcast" (O): 4
Count of "Rainy" (R): 3
Since "Overcast" has a count of 4, we can choose it as the MCV. Let's choose "Sunny" as the MCV.
Now, we have the following new instance with the missing "Outlook" value replaced by "Overcast":
Outlook: Overcast, Temperature: Mild, Humidity: Normal, Wind: Weak, Play: Yes
Now, we can calculate the information gain for all four variables (O, T, H, W). Since the only counts that changes are) = S and O = O, we only need to change the count for Outlook information gain, everything else stays the same.
Target attribute Play: $Entropy(P) = \approx 0.9183$
Outlook (O):
O=S: $5/15 = 1/3$, $p = 2/5$, $n = 3/5$. $Entropy(O = S) \approx 0.971$
O=O: $2/15 = 1/3$, $p = 1$, $n = 0$. $Entropy(O = O) = 0$
O=R: $5/15 = 1/3$, $p = 3/5$, $n = 2/5$. $Entropy(O = R) \approx 0.971$
$Entropy(O) = \frac{1}{3}0.971 + \frac{4}{15}0 + \frac{1}{3}0.971 \approx 0.6473$
$IG(O) = 0.9183 - 0.6473 = 0.271$
Temperature (T): $IG(T) = 0.9183 - 0.8858 = 0.0325$
Humidity (H): $IG(H) = 0.1686$
Wind (W): $IG(W) = 0.0598$
As we can see information gain for the Outlook attribute remains the highest.

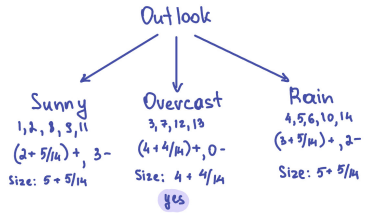
(c) [3 points] Use the fractional counts to infer the feature values, and then calculate the information gains of the four features. Indicate the best feature.

Solution:

Fractional Counts for Outlook (O):
Total instances: 14. Calculate the fraction of each unique value in the available data:
Fraction(S) = Count(S) / Total instances = 5 / 14
Fraction(O) = Count(O) / Total instances = 4 / 14

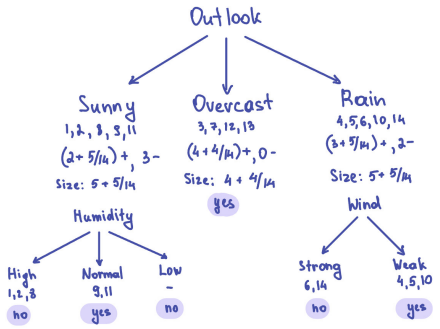
Fraction(R) = Count(R) / Total instances = 5 / 14
Then sizes of Sunny, Overcast, and Rain are the following, respectively: $5+5/14$, $4+4/14$, and $5+5/14$. Then, we can fin p_+ and p_- for all atrrbutes, which will help us calculate entropy and then information gains.
Outlook: O=S: $p_+ = \frac{2+5/14}{5+5/14}$, $p_- = \frac{3}{5+5/14}$
$Entropy(O = S) = -\frac{2+5/14}{5+5/14}(log_2(\frac{2+5/14}{5+5/14})) - \frac{3}{5+5/14}(log_2(\frac{3}{5+5/14})) \approx 0.9896$
Outlook: O=O: $p_+ = \frac{4+4/14}{4+4/14}$, $p_- = \frac{0}{4+5/14}$
$Entropy(O = O) = -\frac{4+4/14}{4+4/14}(log_2(\frac{4+4/14}{4+4/14})) - 0 = 0$
Outlook: O=R: $p_+ = \frac{3+5/14}{5+5/14}$, $p_- = \frac{2}{5+5/14}$
$Entropy(O = R) = -\frac{3+5/14}{5+5/14}(log_2(\frac{3+5/14}{5+5/14})) - \frac{2}{5+5/14}(log_2(\frac{2}{5+5/14})) \approx 0.9532$
Since the current entropy is still 0.9183, we get the following:
$Entropy(O) = \frac{5+5/14}{15}0.9896 + \frac{4+4/14}{15}0 + \frac{5+5/14}{15}0.9532 \approx 0.6939$
$IG(O) = 0.9183 - 0.6939 = 0.2244$
The infromation gains for other attributes remain the same since the missing value does not affect them. Therefore, the highest IG value is still the Outlook value.

(d) [7 points] Continue with the fractional examples, and build the whole free with information gain. List every step and the final tree structure.
Solution:
According to the previous parts, we know that Outlook has the highest information gain, so we can take it as a root node and have the first split there (as seen below).



For the subset "Sunny", entropy is already found and it is 0.9896. Now, we can find out which attribute should split here.
Temperature (T): T=H: $2/6 = 1/3$, $p = 0$, $n = 1$, $Entropy(T = H) = 0$.
Temperature (T): T=M: $3/6 = 1/2$, $p = 2/3$, $n = 1/3$, $Entropy(T = H) \approx 0.9183$.
Temperature (T): T=C: $1/6$, $p = 1$, $n = 0$, $Entropy(T = H) = 0$.
$Entropy(T) = \frac{1}{3}0 + \frac{1}{2}0.9183 + \frac{1}{6}0 \approx 0.4592$
$IG(T) = 0.4944$
Humidity (H): H=H: $3/6 = 1/2$, $p = 0$, $n = 1$, $Entropy(H = H) = 0$.
Humidity (H): H=N: $3/6 = 1/2$, $p = 1$, $n = 0$, $Entropy(H = N) = 0$.
Humidity (H): H=L: 0.
$Entropy(H) = 0$, $IG(H) = 0.9183$
There is no need to check Wind attribute, since Humidity has the highest possible information gain. Therefore, we split at Humidity.
For the subset "Rain", entropy is already found and it is 0.9532. Now, we can find out which attribute should split here.

Temperature (T): T=H: 0.
Temperature (T): T=M: 3/5, $p = 2/3$, $n = 1/3$, $Entropy(T = H) \approx 0.9183$.
Temperature (T): T=C: 2/5, $p = 1/2$, $n = 1/2$, $Entropy(T = H) = 1$.
$Entropy(T) = 0 + \frac{3}{5}0.9183 + \frac{2}{5}1 \approx 0.951$
$IG(T) = 0.0022$
Wind (W): H=H: 3/5, $p = 1$, $n = 0$, $Entropy(H = H) = 0$.
Wind (W): H=N: 2/5, $p = 0$, $n = 1$, $Entropy(H = N) = 0$.
$Entropy(W) = 0$, $IG(W) = 0.9532$
Wind has the highest information gain. Therefore, we split at Humidity.



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)

Solution:

Proof: We want to prove that $IG(A)$ is always non-negative. To do this, we will use the convexity of the entropy function. (Assuming that the entropy of a random variable Y is defined as $H(Y) = -\sum p(y)\log_2(p(y))$.)

Let $p_i = P(A = v_i)$ be the probabilities associated with each value v_i of attribute A. Then, the weighted probability distribution is $q(y) = \sum p_i P(Y|A = v_i)$, which is a weighted sum of conditional probability distributions. Therefore, by the convexity property of entropy:

$H(q(y)) \leq \sum p_i H(P(Y|A = v_i))$

We can now expand this formula by using the definition of $IG(A)$:

$H(q(y)) \leq \sum p_i H(Y|A = v_i) = \sum P(A = v_i)H(Y|A = v_i)$

We can subtract $H(q(y))$ from both sides of the $IG(A)$ equation, so we get:

$H(Y) - H(q(y)) \geq H(Y) - \sum P(A = v_i)H(Y|A = v_i)$

We also know that $H(Y) - H(q(y))$ represents the entropy of Y given attribute A, denoted as $H(Y|A)$, so we can substitute:

$H(Y|A) \geq H(Y) - \sum P(A = v_i)H(Y|A = v_i)$

Lastly, we can substitute this result back into the $IG(A)$ equation:

$IG(A) = H(Y) - \sum P(A = v_i)H(Y|A = v_i) \geq H(Y) - H(Y|A) \geq 0$

The last inequality follows from the fact that entropy is always non-negative, and $H(Y) \geq H(Y|A)$ because splitting the data based on attribute A reduces uncertainty (entropy) about Y. Therefore, we've proven that the information gain $IG(A)$ is always non-negative, meaning that as long as we split the data, the purity (entropy) will never get worse.

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?
Solution:
In the context of constructing a regression tree in the ID3 framework,we can use variance. The formula would look like the following:
$IG(A) = Var(S) - \sum_{v_i} \frac{|S_v|}{|S|} * Var(S_v)$
where v_i are values of A.
This formula works because the goal of a regression tree is to find attribute splits that result in reduced variance within the subsets. When variance is reduced after a split, it indicates that the predicted values in those subsets are more similar to each other, resulting in better predictions. By selecting the attribute A that maximizes the reduction in variance (i.e., maximizes $IG(A)$), we ensure that the chosen split leads to the greatest improvement in prediction accuracy.

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.
Solution: Here is the link to my repository: https://github.com/milenabel/CS6350-ML2023

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.

   Note: we highly recommend you to use Python for implementation, because it is very convenient to load the data and handle strings. For example, the following snippet reads the CSV file line by line and split the values of the attributes and the label into a list, "terms". You can also use "dictionary" to store the categorical attribute values. In the web are numerous tutorials and examples for Python. if you have issues, just

google it!

```
with open(CSVfile, 'r') as f:
    for line in f:
        terms = line.strip().split(',')
        process one training example
```

(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.
Solution: The implementation can be seen in my Github in the carIDE.py file inside the DecisionTree folder.

(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.
Solution: The solution for this part is represented through table 4.

Error Type	Max Depth	Avg Train Error	Avg Test Error
information gain	1	0.3020	0.2967
information gain	2	0.2220	0.2225
information gain	3	0.1810	0.1964
information gain	4	0.0820	0.1470
information gain	5	0.0270	0.0907
information gain	6	0.0000	0.0989
majority error	1	0.3020	0.2967
majority error	2	0.3020	0.2967
majority error	3	0.2590	0.2940
majority error	4	0.1690	0.2665
majority error	5	0.3020	0.2102
majority error	6	0.0600	0.2225
gini index	1	0.3020	0.2967
gini index	2	0.2220	0.2225
gini index	3	0.1760	0.1841
gini index	4	0.0890	0.1332
gini index	5	0.0270	0.0907
gini index	6	0.0000	0.0989

Table 4: Average Prediction Errors

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

Solution:
From the results of part (b), we can make the following observations regarding the training errors and test errors for different heuristics and different tree depths:
Information Gain Heuristic:
Training Error: Decreases as the tree depth increases.
Test Error: Initially decreases but then increases slightly after a certain depth.
Conclusion: The information gain heuristic results in relatively low training errors and reasonable test errors, suggesting that it effectively builds decision trees that generalize well. However, it's crucial to avoid overfitting by selecting an appropriate tree depth.
Majority Error Heuristic:
Training Error: Generally decreases as the tree depth increases.
Test Error: Initially decreases but then tends to plateau or increase slightly after a certain depth.
Conclusion: The majority error heuristic also results in low training errors but tends to have higher test errors compared to information gain. It may be less effective at generalization.
Gini Index Heuristic:
Training Error: Decreases as the tree depth increases.
Test Error: Shows a similar pattern to information gain, initially decreasing and then stabilizing or slightly increasing.
Conclusion: The Gini index heuristic performs similarly to information gain in terms of training and test errors, indicating its effectiveness in building decision trees that generalize well.
Overall, it appears that the Information Gain and Gini Index heuristics are more effective than Majority Error in terms of generalization performance, as they tend to produce lower test errors.

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.
Solution: The solution for this part is represented through tables 5, 6, 7.

Max Depth	Avg Train Error	Avg Test Error
1	0.1192	0.1248
2	0.1060	0.1114
3	0.1006	0.1068
4	0.0800	0.1154
5	0.0624	0.1296
6	0.0480	0.1386
7	0.0372	0.1464
8	0.0292	0.1522
9	0.0222	0.1572
10	0.0182	0.1614
11	0.0150	0.1640
12	0.0136	0.1668
13	0.0132	0.1686
14	0.0132	0.1694
15	0.0132	0.1708
16	0.0132	0.1708

Table 5: Information Gain Average Prediction Errors without "Unknown" as Features

(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.
Solution: The solution for this part is represented through tables 8, 9, 10.

(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?

Solution:
From the results of parts (a) and (b), we can make the following observations regarding the training errors and test errors for different heuristics, different tree depths, as well as different ways to deal with "unknown" attribute values:
Without Handling "Unknown" as Missing Values:
As the tree depth increases, the training error tends to decrease for all three heuristics (information gain, majority error, gini index). This is expected because deeper trees can model the training data more closely, leading to lower training errors.

Max Depth	Avg Train Error	Avg Test Error
1	0.1088	0.1166
2	0.1042	0.1088
3	0.0964	0.1134
4	0.0790	0.1204
5	0.0648	0.1302
6	0.0544	0.1416
7	0.0428	0.1638
8	0.0372	0.1764
9	0.0328	0.1774
10	0.0262	0.1820
11	0.0232	0.1926
12	0.0192	0.2012
13	0.0154	0.2060
14	0.0132	0.2084
15	0.0132	0.2102
16	0.0132	0.2102

Table 6: Majority Error Average Prediction Errors without "Unknown" as Features

The test error, however, does not always follow the same trend. It initially decreases with increasing depth but starts to increase after reaching a certain depth. This indicates overfitting, where the model becomes too specialized in the training data and performs poorly on unseen test data.
Among the three heuristics, "information gain" generally performs better in terms of test error compared to "majority error" and "gini index."
With Handling "Unknown" as Missing Values:
Handling "unknown" values as missing values and completing them with the majority value of the respective attribute from the training set does not significantly affect the overall trend observed in the errors.
The same trends of decreasing training error and initial decrease followed by an increase in test error with increasing tree depth are still observed.
The choice of heuristic remains important, with "information gain" generally performing better in terms of test error.
Handling "unknown" values as missing values can help ensure that the decision tree algorithm can work with the dataset, but it doesn't fundamentally change the trade-off between tree complexity and overfitting.
Overall, deeper trees can fit the training data better but may lead to overfitting, while shallower trees may underfit.

Max Depth	Avg Train Error	Avg Test Error
1	0.1088	0.1166
2	0.1042	0.1088
3	0.0936	0.1124
4	0.0754	0.1232
5	0.0606	0.1338
6	0.0484	0.1474
7	0.0366	0.1568
8	0.0268	0.1614
9	0.0214	0.1660
10	0.0172	0.1690
11	0.0140	0.1736
12	0.0136	0.1744
13	0.0132	0.1758
14	0.0132	0.1760
15	0.0132	0.1780
16	0.0132	0.1780

Table 7: Gini Index Average Prediction Errors without "Unknown" as Features

Max Depth	Avg Train Error	Avg Test Error
1	0.1192	0.1248
2	0.1060	0.1114
3	0.1008	0.1068
4	0.0814	0.1168
5	0.0646	0.1310
6	0.0510	0.1426
7	0.0406	0.1508
8	0.0330	0.1572
9	0.0274	0.1600
10	0.0218	0.1636
11	0.0194	0.1648
12	0.0188	0.1668
13	0.0180	0.1694
14	0.0180	0.1706
15	0.0180	0.1738
16	0.0180	0.1738

Table 8: Information Gain Average Prediction Errors with "Unknown" as Features

Max Depth	Avg Train Error	Avg Test Error
1	0.1088	0.1166
2	0.1042	0.1088
3	0.0964	0.1134
4	0.0804	0.1196
5	0.0666	0.1286
6	0.0566	0.1396
7	0.0476	0.1584
8	0.0432	0.1662
9	0.0382	0.1688
10	0.0306	0.1768
11	0.0274	0.1824
12	0.0260	0.1918
13	0.0210	0.1978
14	0.0180	0.2018
15	0.0180	0.2048
16	0.0180	0.2048

Table 9: Majority Error Average Prediction Errors with "Unknown" as Features

Max Depth	Avg Train Error	Avg Test Error
1	0.1088	0.1166
2	0.1042	0.1088
3	0.0936	0.1124
4	0.0770	0.1232
5	0.0634	0.1326
6	0.0522	0.1454
7	0.0406	0.1556
8	0.0318	0.1600
9	0.0268	0.1622
10	0.0214	0.1676
11	0.0190	0.1686
12	0.0184	0.1718
13	0.0180	0.1730
14	0.0180	0.1740
15	0.0180	0.1772
16	0.0180	0.1772

Table 10: Gini Index Average Prediction Errors with "Unknown" as Features