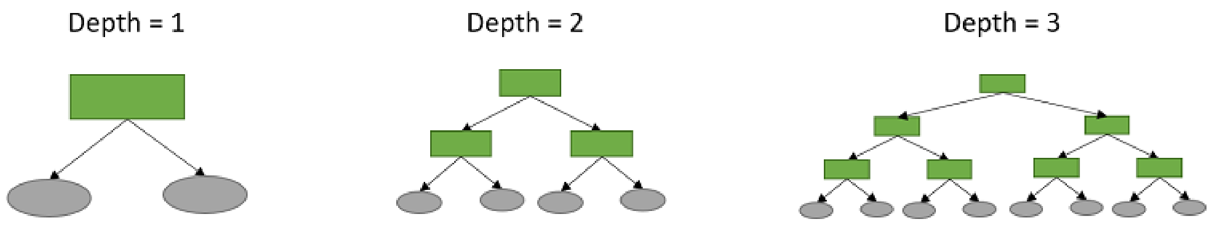
Assignment 1

Type	Homework
Materials	Assignment1 (1).pdf
Reviewed	☐
Date	@August 31, 2022
Topic	

1A

[Part 1] This problem asks for perfect binary trees of depth 3. When asked for clarification about the definition of depth, we were given the following example:



A decision tree can have the same attribute twice in the tree, provided it's not on the same path to a leaf. Therefore, we have the full selection of 5 nodes at the root. For each of its children, we have 4 choices, as it cannot re-use the attribute in the root. We have 3 choices then for each of their children. The number of possible trees is simply the product of all of those nodes:

$5 \cdot 4^2 \cdot 3^4 = 6480$

[Part 2] To generalize the above calculation, we make the following notes: 5 comes from the number of attributes we had to choose from, A, and the number of terms is equal to the depth $D$of the desired tree.

For a full tree, at level i of the tree there should be 2^i nodes. I consider the root to be on level 0. The 0th level has 1 node, the 1st has 2, the 2nd has 4, and so on.

On each level, there is one less choice of attributes to pick than the one above it. According to the question, the tree should not test the same attribute twice, so at each level you have $(A - i)$ choices of attribute to pick.

In this assignment, a tree of depth 1 only has attributes on level 0, a tree of depth 2 only has attributes on levels 0 and 1, and so on. Therefore, when doing the calculation, we must not consider the last level, which only contains leaves, not attributes.

The number of possible unique trees should be the product of the number of different arrangements at each level, which is the formula below.

$$\prod_{i=0}^{D-1} (A-i)^{2^i}$$

Note that the exponent on that term is 2^i. It’s hard to see.

1B

Note: I used a jupyter notebook to do this programmatically. It will be attached with the name “Assignment1_Question1.ipynb”. The result of the cells should resemble this:

```
Original had 8+, 6-
0.9852

Positive had 4+, 2-
Negative had 4+, 4-
IG = 0.0202
Early: 0.9650

Positive had 5+, 2-
Negative had 3+, 4-
IG = 0.0611
FinishedHM: 0.9242

Positive had 5+, 3-
Negative had 3+, 3-
IG = 0.0113
Senior: 0.9740

Positive had 3+, 1-
Negative had 5+, 5-
IG = 0.0391
LikesCoffee: 0.9461

Positive had 5+, 4-
Negative had 3+, 2-
IG = 0.0013
LikedTheLastJedi: 0.9839
```

Actual Answer

To determine the first attribute that we will put at the root of the decision tree, we calculate the information gain from each attribute

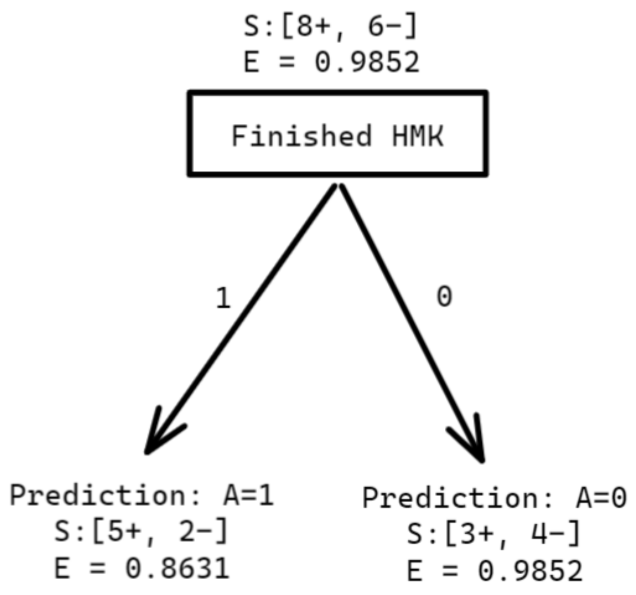
> We first use the formula $H(S) = -p_+ \log_2 p_+ - p_- \log_2 p_-$ to find the entropy of the entire dataset. There are 8 positive results and 6 negatives, making H(S) = 0.9852
>
> To find the information gain for an attribute, first partition the dataset with respect to that attribute. Since all of the attributes are binary, this will form 2 subsets (P_+ and P_-). Calculate the entropy

for each subset.

To get the information gain, multiply each subset's entropy by the size of that subset over the size of the original set. Sum them. The information gain is simply the difference between the original entropy and the weighted sums of the subsets' entropy. See the table below for the information gain of each attribute in the dataset.

Attribute	Entropy	Information Gain	Examples
None (original data)	0.9852	N/A	8+ 6-
Early	0.9650	0.0202	P_+ : 4+ 2- P_- : 4+ 4-
Finished HMK	**0.9242**	**0.0611**	P_+ : **5+ 2-** P_- : **3+ 4-**
Senior	0.9740	0.0113	P_+ : 5+ 3- P_- : 3+ 3-
Likes Coffee	0.9461	0.0391	P_+ : 3+ 1- P_- : 5+ 5-
Liked the Last Jedi	0.9839	0.0013	P_+ : 5+ 4- P_- : 3+ 2-

From this table, we can see, the attribute that gives the highest information gain is **Finished HMK**. Therefore, we select that attribute for the root of our depth-1 decision tree. The tree shown below is the depth 1 decision tree generated by ID3.



Now, the problem asks for the depth 2 tree. Because ID3 selects the locally optimal choice at each step, the depth 2 tree also contains **Finished HMK** at the root. Now, we determine which attribute gives the largest information gain for child node on the left side.

Attribute	Entropy	Information Gain	Examples

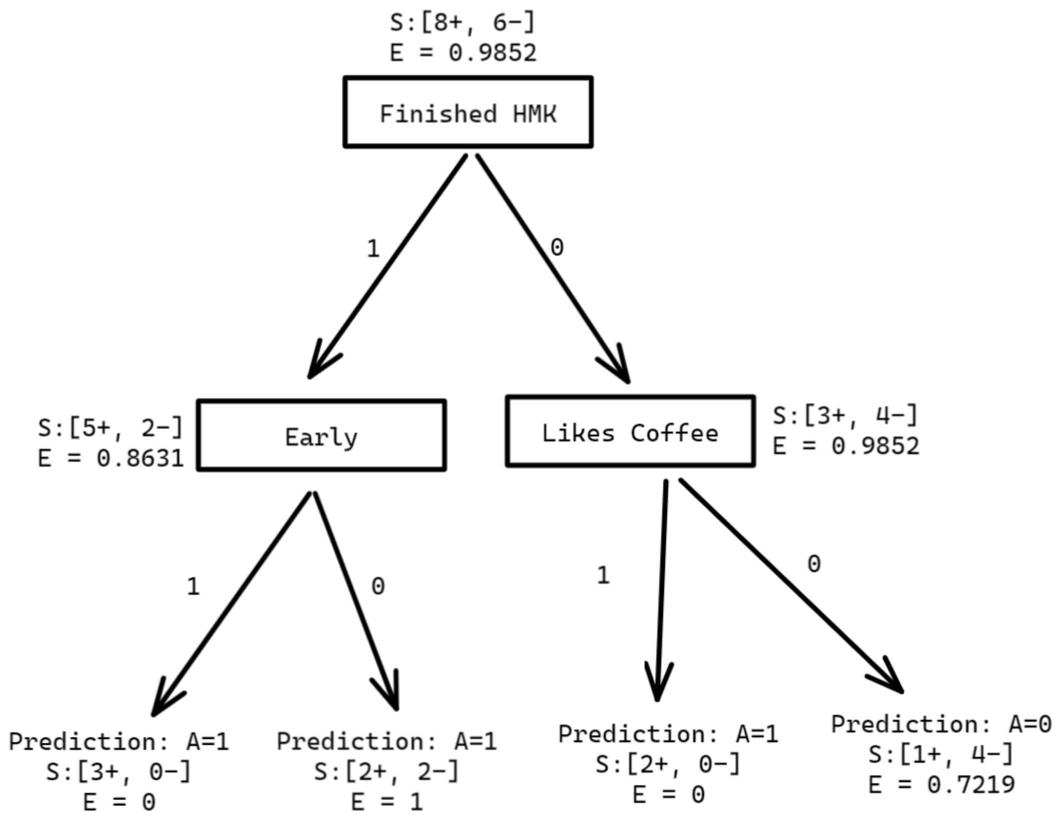
Attribute	Entropy	Information Gain	Examples
None (original data)	0.8631	N/A	5+ 2-
Early	**0.5714**	**0.2917**	P_+ **: 3+ 0-** P_- **: 2+ 2-**
Senior	0.6935	0.1696	P_+ : 3+ 2- P_- : 2+ 0-
Likes Coffee	0.8014	0.0617	P_+ : 1+ 1- P_- : 4+ 1-
Liked the Last Jedi	0.6935	0.1696	P_+ : 3+ 2- P_- : 2+ 0-

For the left node, the attribute that gives the most information is **Early,** so we select that node for the left side.

For the right side, we again determine largest information gain.

Attribute	Entropy	Information Gain	Examples
None (original data)	0.9852	N/A	3+, 4-
Early	0.9650	0.0202	P_+ : 1+ 2- P_- : 2+ 2-
Senior	0.8571	0.1281	P_+ : 2+ 1- P_- : 1+ 3-
Likes Coffee	**0.5157**	**0.4696**	P_+ **: 2+ 0-** P_- **: 1+ 4-**
Liked the Last Jedi	0.9650	0.0202	P_+ : 2+ 2- P_- : 1+ 2-

For the right node, the attribute that gives the most information is **Likes Coffee**. We select that node for the right side. The resulting depth 2 tree after picking those attributes is below.



1C

The depth 3 tree will have the same top 2 levels as the depth 2 tree, so we can use the above problem as a starting point. ID3 will stop on a leaf when the examples are perfectly classified at that point, so we do not need to do any more computations for the leaf at path [1,1] and at path [0,1].

For the leaf at path [1,0], or the right child of Early, we calculate the following information gain:

Attribute	Entropy	Information Gain	Examples
None (original data)	1	N/A	2+, 2-
Senior	**0.6887**	**0.3113**	P_+ **: 1+ 2-** P_- **: 1+ 0-**
Likes Coffee	1	0	P_+ : 1+ 1- P_- : 1+ 1-
Liked the Last Jedi	0.6887	0.3113	P_+ : 1+ 2- P_- : 1+ 0-

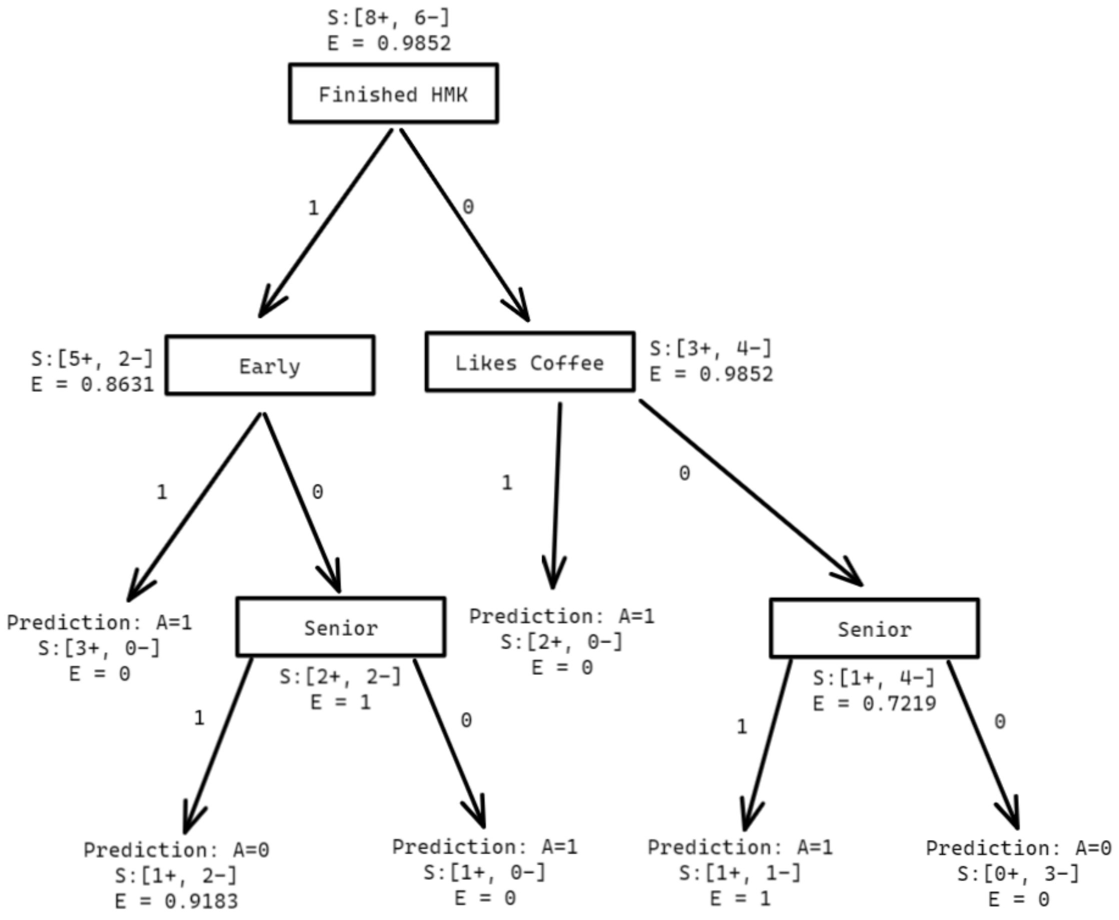
Therefore, at this position, we choose **Senior.** The information gain is same for Liked the Last Jedi, but we saw Senior first, so we arbitrarily pick that one.

For the leaf at path [0,0], or the right child of Likes Coffee,

Attribute	Entropy	Information Gain	Examples

Attribute	Entropy	Information Gain	Examples
None (original data)	0.7219	N/A	1+, 4-
Early	0.5510	0.1710	P_+ : 0+ 2- P_- : 1+ 2-
Senior	**0.4000**	**0.3219**	P_+ : **1+ 1-** P_- : **0+ 3-**
Liked the Last Jedi	0.5510	0.1710	P_+ : 1+ 2- P_- : 0+ 2-

At this position, we choose **Senior** as well. The final depth 3 decision tree is now below.



What tree should be picked to classify 10 new students?

Given that the tree is nicely human readable, I can use my intuition to guess that the tree with depth 2 will be the best one.

If the end goal is to predict if a student got an A, attributes like “Finished Homework” should have a high effect on that goal, as well as attributes like “Early.” For the depth 3 tree, the extra classification by “Senior” was found on a very small amount of data, meaning generalizations made on it are not reliable. Following Occam’s Razor, I believe that the tree with depth 2 will make the most accurate and generalizable predictions.

1D

A dataset with a sufficiently large number of attributes is unrealizable if and only if the dataset contains a conflict. A conflict occurs when two data points contain the same combination of attributes while having a different result. For example, on the above dataset, the following would be a conflict.

Early	Finished HMK	Senior	Likes Coffee	Liked the Last Jedi	A
1	0	1	1	0	0
1	0	1	1	0	1

In this case (and in general), attempting to form a decision tree on all of those attributes would lead to a leaf node with $(1+, 1-) \quad E \geq 0$ and no further way to classify them. This prevents the dataset from being realizable.

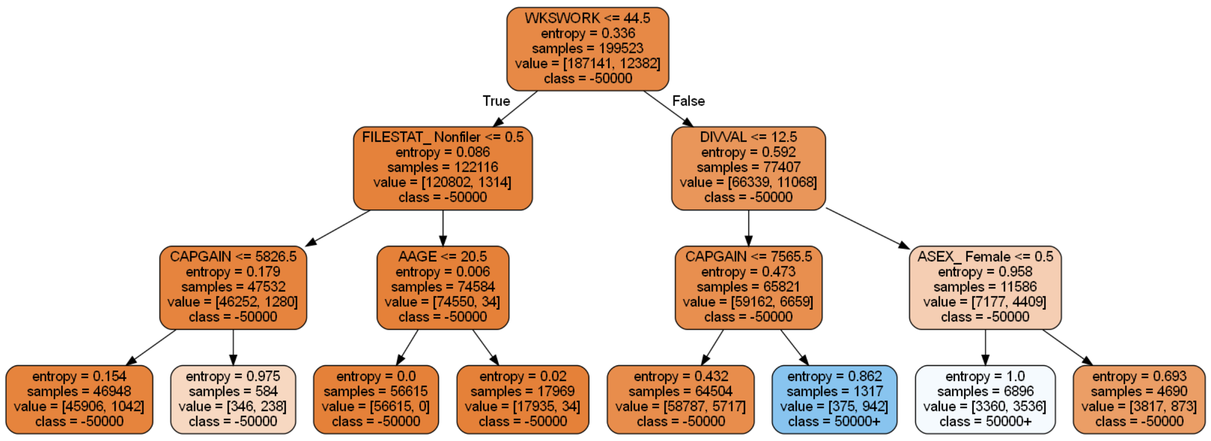
Section 2

2A

The code for section 2 will be attached as a jupyter notebook named “Assignment1_Question2.ipynb”. There is also text throughout that notebook that shows my development process.

For a brief summary:

> I first loaded data using pandas. I then formatted it using pandas and sklearn.preprocessing to turn categorical labels into one-hot attributes. This changed the number for columns from 42 to 509.
>
> I used sklearn’s DecisionTreeClassifier to generate the actual decision tree, and I used graphviz to visualize it. See the below picture of the depth 3 tree it generated.



I then generated trees for each depth.

From the notebook file, the accuracy on training data for each depth is as follows:

```
# Jupyter output
Max depth 2: 0.9379419916500854
Max depth 3: 0.941665873107361
Max depth 4: 0.9441267422803387
Max depth 5: 0.9461465595445137
Max depth 6: 0.946958496012991
Max depth 7: 0.9483217473674714
Max depth 8: 0.9507826165404489
Max depth 9: 0.9523663938493306
Max depth 10: 0.9537647288783749
```

Based on this data alone, the tree with depth 10 seems like the ideal choice with an accuracy of **95.4%.** Therefore, the best $k = 10$.

2B

In order to check the accuracy of my depth 10 model on the testing data, I first had to format the testing data to match that of the training data. You can find that process on the jupyter notebook.

On the testing data, I saw an accuracy of **0.9751,** or **97.5%.**

```
# Jupyter output
Accuracy on TESTING data: 0.975160758408805
```

2C

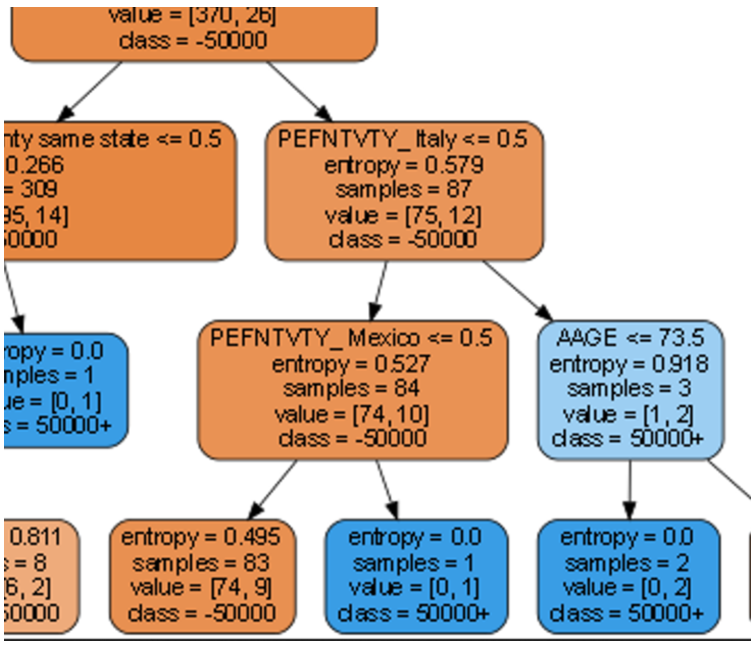
To determine if there was overfitting, we calculate the error on the training data and testing data. If we had more error on the testing data than the training data, the tree would be overfitting that training data. However, in my case, the classifier actually performed better on the testing set than the training set.

```
Error on TRAINING data: 0.04624
Error on TESTING data: 0.02484
Overfitting: -0.02140
```

Based on this, I believe that the generalizations made by the classifier were well picked, and they predict new data's result well.

Why didn't I see overfitting?

From what I could tell, some of the class found that their classifiers were less effective on the testing data than the training data — their models were overfitting the training set. I believe that has to do with my use of the OneHotEncoder. My model has effectively 509 attributes that can be picked at each node. For example, see this snippet from the depth 10 tree generated by the notebook:



The model was able to check PEFNTVTY (country of birth father) for two different values.

Say those values were encoded into integers, 5 for Italy and 15 for Mexico. If a decision tree put a split for PEFNTVTY ≥ 10, it would be impossible for both attributes to be checked on either side of that node.

Since my model was able to create these types of decision trees, it allowed my model to pick from a larger selection of possible trees, and evidently that lead to it creating more applicable generalizations.