CAS CS 440
Lec 20

Supervised Learning II

1. Our Example Data
   a.

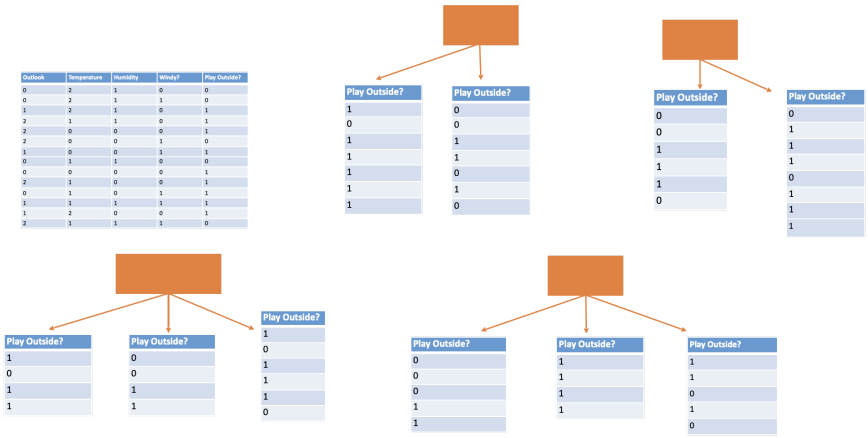
Outlook	Temperature	Humidity	Windy?	Play Outside?
Sunny	Hot	High	False	No
Sunny	Hot	High	True	No
Overcast	Hot	High	False	Yes
Rainy	Mild	High	False	Yes
Rainy	Cool	Normal	False	Yes
Rainy	Cool	Normal	True	No
Overcast	Cool	Normal	True	Yes
Sunny	Mild	High	False	No
Sunny	Cool	Normal	False	Yes
Rainy	Mild	Normal	False	Yes
Sunny	Mild	Normal	True	Yes
Overcast	Mild	High	True	Yes
Overcast	Hot	Normal	False	Yes
Rainy	Mild	High	True	No

      i. Goal is whether we will go out and play outside
      ii. Every row is a separate data with four features and one ground truth
2. First Convert into Numeric Representation
   a.

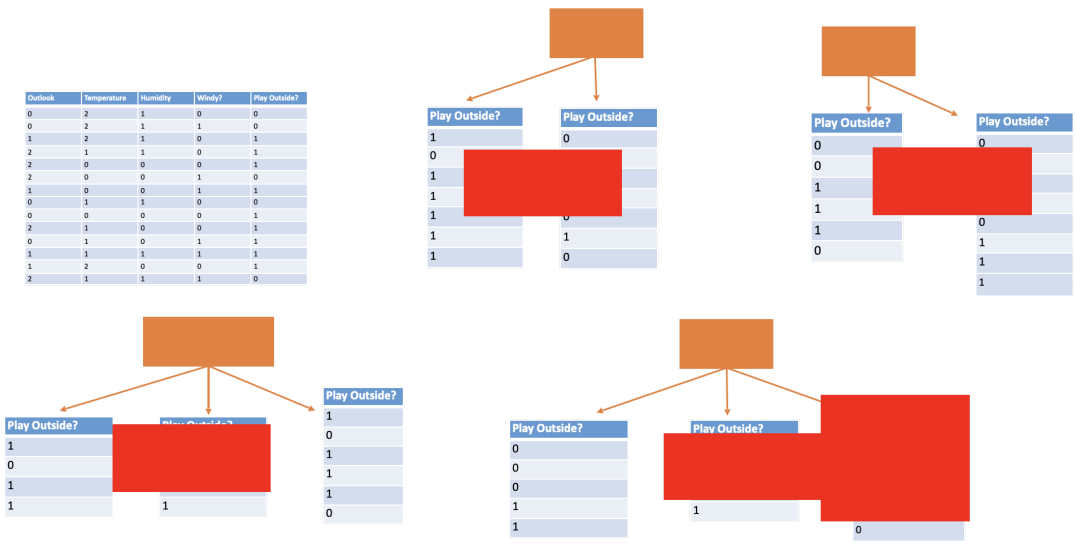
Outlook	Temperature	Humidity	Windy?	Play Outside?
0	2	1	0	0
0	2	1	1	0
1	2	1	0	1
2	1	1	0	1
2	0	0	0	1
2	0	0	1	0
1	0	0	1	1
0	1	1	0	0
0	0	0	0	1
2	1	0	0	1
0	1	0	1	1
1	1	1	1	1
1	2	0	0	1
2	1	1	1	0

   b. Computers do not like strings → change into bits
   c. For example, “sunny” is 0

3. Build a Decision Tree

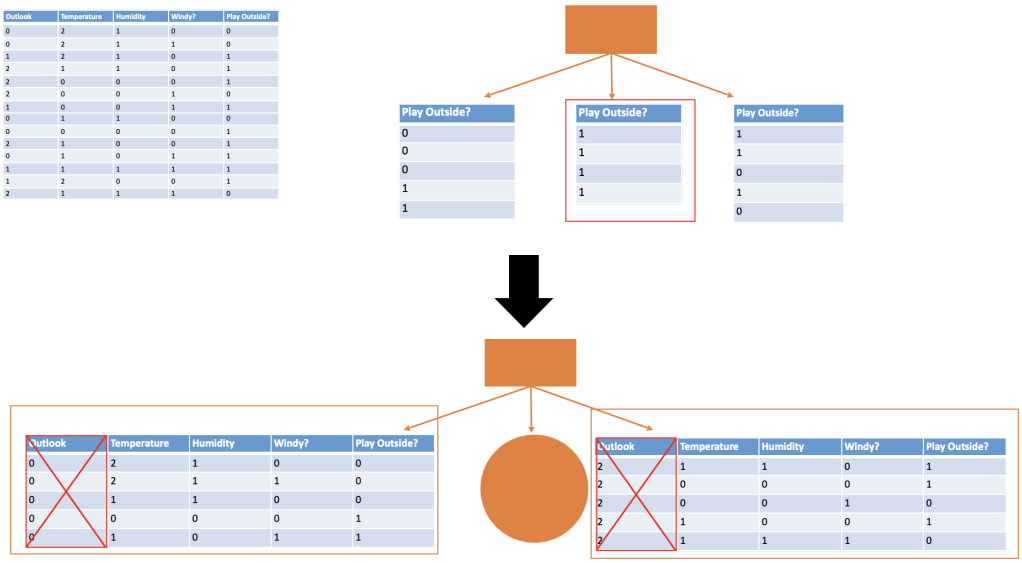
a. 

b. Now, we can construct flowcharts
c. Vertices ask questions and edges tell where to go
d. Questions are features and edges are the answers here
e. For example, “what happens to our predictions if we make a node for humidity?”
    i. If humidity == 0, go to a branch
    ii. If humidity == 1, go to another branch
f. Collect data for each branch
    i. If humidity==0, collect the columns of real data that have humidity of 0
    ii. If humidity ==1, collect the columns of real data that have humidity of 1
g. Repeat the same exact process for all features (Windy?, Temperature, Outlook)
    i. Notice that outlook and temperature have three branches since there are three values associated with it

h. 

    i. Pretend we have magical way of picking Quality
    ii. Example (higher quality values are better)

1. Quality of humidity is 0.152
2. Quality of Windy? Is 0.048
3. Quality of temperature is 0.029
4. Quality of outlook is 0.247 → the best node

i. 

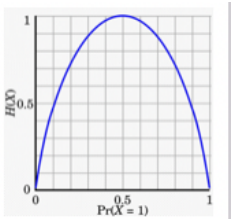
   i. We chose outlook as the best node
   ii. If outlook == 0, the outcomes are
      1. 0
      2. 0
      3. 0
      4. 1
      5. 1
      6. Therefore, we need to recurse
   iii. If outlook == 1, the outcomes are all 1s
      1. Perfect predictions! Don't need to recurse
   iv. If outlook == 2, the outcomes are
      1. 1
      2. 1
      3. 0
      4. 1
      5. 0
      6. Therefore, we need to recurse
   v. Now, we need to conduct trees for when outlook == 0 and outlook == 2 until we make leaves (where we do not need to recurse)
   vi. Get rid of data that is useless → the outlook feature is not useless
   vii. At most four layers of tree (four features)

4. What is “Quality?” of a Node?
    a. Measure how well a node splits the ground truth
    b. Want:
        i. Quality to be high when split is “pure”
        ii. Quality to be low when different classes are distributed within same child
    c. A few ways to do this:
        i. Information Gain (based on the information gain)
        ii. Gini Index
        iii. Etc.
5. Information Gain & Entropy
    a.

Play Outside?
1
0
1
1
1
1
1

    b. How good is a split?
    c. Idea: Can we measure how much info is inside the ground truth?
        i. Entropy function

$$H(X) = -\mathbb{E}\left[\log(\Pr[X])\right] = -\sum_{x \in X} \Pr[X = x]\log(\Pr[X = x])$$

        ii. 

    d. Entropy?
        i.

X	Pr[X]
0	1/7
1	6/7

→ turn the ground truth to PMF

        ii. $$-\Pr[X = 1]\log(\Pr[X = 1]) - \Pr[X = 0]\log(\Pr[X = 0])$$

    e. Entropy tells us how “well distributed” a distribution is
        i. Think of it like compression: how many bits needed to compress distribution?
            1. If distribution is skewed: fewer bits
            2. If distribution is balanced: more bits

f. How can we use this to determine the "quality" of a split?
   i. Measure entropy before & after!
   ii. Measure the difference → it is called (information gain)!

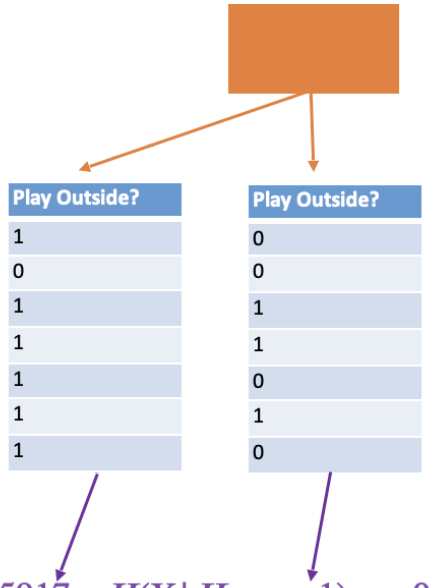
g. Before split:

Humidity	Play Outside?
1	0
1	0
1	1
1	1
0	1
0	0
0	1
1	0
0	1
0	1
0	1
1	1
0	1
1	0

X	Pr[Hum]
0	7/14
1	7/14

$$H(Y) = 0.9402$$

h. After split:



Play Outside?
1
0
1
1
1
1
1

Play Outside?
0
0
1
1
0
1
0

i. $H(Y \mid Hum = 0) = 0.5917 \quad H(Y \mid Hum = 1) = 0.9852$
   i. Multiply each by the probability that it appears in data

$$IG(Hum) = H(Y) - H(Y \mid Hum)$$
$$= H(Y) - \mathbb{E}[H(H \mid Hum)]$$

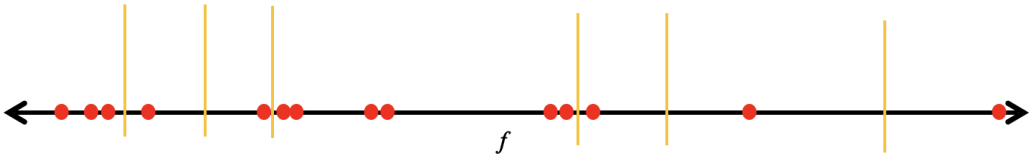
j. $$= H(Y) - \Pr[Hum = 0] H(Y \mid Hum = 0) \quad - \Pr[Hum = 1] H(Y \mid Hum = 1)$$
   i. This is the calculation

k. The final value
   i. 0.9402 - (7/14) * (0.5917) - (7/14) * 0.9852 = 0.152

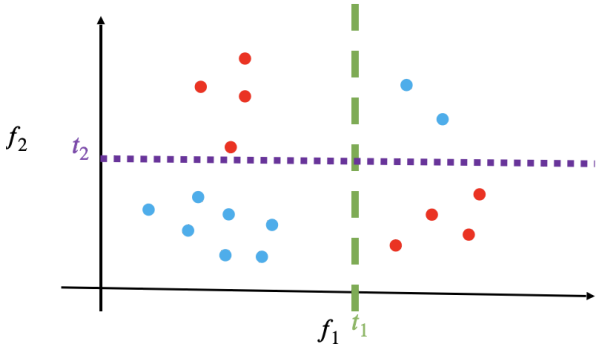
l. Information gain → how much does a feature tell me about the ground truth?
   i. The node that tells me the most information about the ground truth is the best

6. When to stop?
    a. Run out of features
    b. The labels of a node are “pure” enough (hyperparameter)
    c. Max depth of the tree is reached (hyperparameter)
    d. What about continuous features?
        i. Feature should still be considered in child nodes

e.
        i. Pick a threshold and binarize based on that
        ii. Does not guarantee that the feature is gone since that feature might still be affecting it
7. Continuous Features
    a. Sort the feature and ground truth by the feature value
    b. Potential values of ‘t’?
        i. Infinitely many
            1. can restrict to those within range of feature
    c. Observation:
        i. Only need to consider values of ‘t’ where the class changes!



d.
        i. Now only have six thresholds to consider
8. Decision Boundary
    a. Let’s think about models geometrically
        i. Consider some 2d data
            1. Our argument extends to higher dims



2.
        ii. Binary classification task
            1. Our argument extends to non-binary classification

b. How do Decision Trees chop up the feature space?
    i. Let's consider continuous features in this example (easier to draw)
        1. Argument applies to discrete features
    ii. Each node picks a features and makes a binary split
        1. axis-parallel (axis-aligned) cuts!
        2. The shape they draw is called a decision boundary
            a. Decision trees draw pixelated-looking rectangular boundaries

c.