DataSci 420

lesson 5: decision trees

Seth Mottaghinejad

today's agenda

- decision trees
  - how decision trees work
  - tennis example
- pros and cons of decision trees
- two examples where decisions trees are not ideal
  - most features are continuous
  - target is continuous

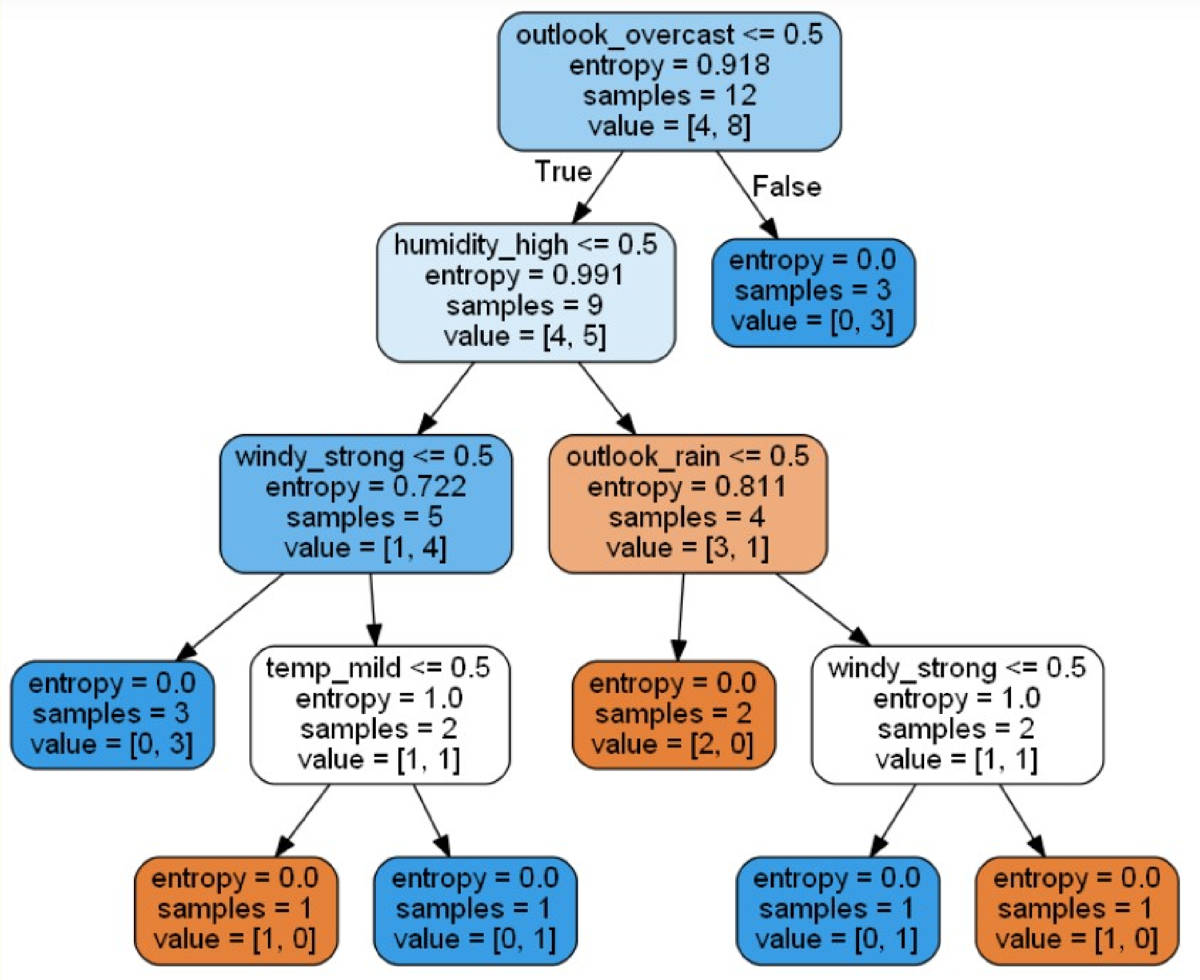
lab time

- suppose you don't know ML
- you have this data set
- you want we use outlook, temperature, humidity, and windiness to **predict if a game will be canceled or not** in the future?
- how would you do it?

	outlook	temp	humidity	windy	play
0	windy	hot	high	strong	no
1	sunny	hot	high	weak	no
2	overcast	hot	high	weak	yes
3	rain	mild	high	weak	yes
4	rain	cool	normal	weak	yes

decision tree

- start at the top: **the root**
- finish at bottom: **the leaves**
- each level is a **depth**
    - root has **depth = 0**
    - leaves have **depth = 4**
- we keep splitting until splitting doesn't make sense any more



recursive algorithm

- starting at the root
  - if you can't split, stop, otherwise find the best feature to split by (and the best way to split) and split the data
  - go to each leaf and repeat
- how do you find the best feature to split by? some statistical splitting criteria: **information gain** or **gini coefficient**, etc.
- how do you know if not to split? define a criterion such as **minimum leaf size** or **maximum depth**

break time

pros

- (+) one of the "white-box" algorithms: self-explanatory
- (+) classification and regression (but mostly classification)
- (+) you can do **post-pruning** based on what makes business sense, or to avoid overfitting
- (+) lend themselves well to **ensemble modeling**
- (+) they can be used to gauge **feature importance** (features split higer in the tree are more important)
- (+) **no one-hot encoding** needed for categorical features, **no normalization** needed for numeric features

cons

- (–) can be **unstable** (small changes in the data can change results)
- (–) lots of **hyper-parameters** to tune (max depth, splitting criteria)
- (–) not the best option when we have **continuous target**
- (–) not the best option when we have mostly **continuous features**
- (–) slow if we have **high-cardinality categorical features**

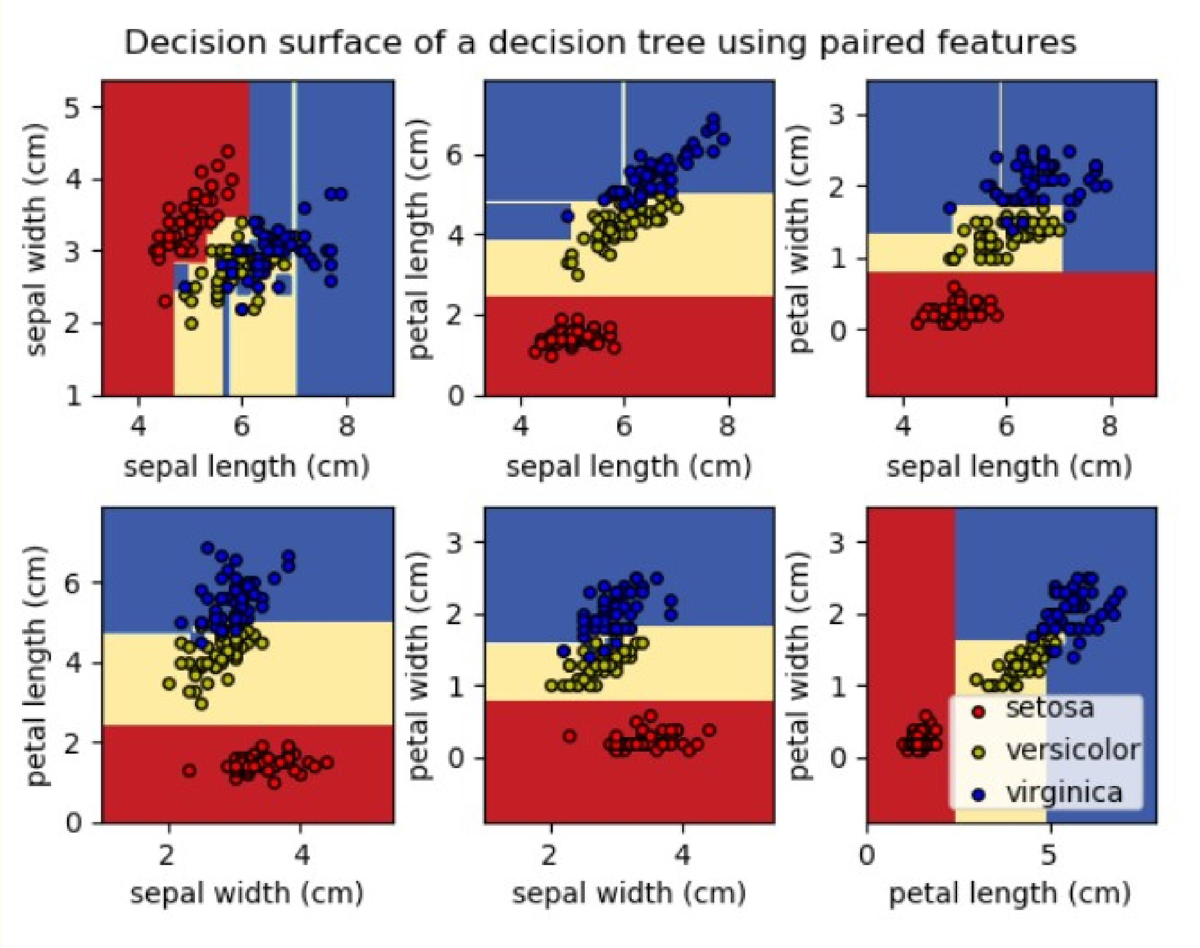
lab time

- return to the tennis data from the previous lab
- let's replace the four categorical features with **two numeric (and continuous) features**:
  - temperature in degrees
  - humidity as a percentage
- your target is the same binary target as before
- what would your decision tree look like? provide a short example
- what would **decision boundary** look like?

classification example

- 3-class target
- 4 continuous features
- decision boundary seems "choppy" and "un-befitting"
- there are better options!

image: https://scikit-learn.org



regression example

- numeric target
- one numeric feature
- feature is **reused** (otherwise there is only a single split and we're done)
- prediction is like a **step function**
- there are better options!

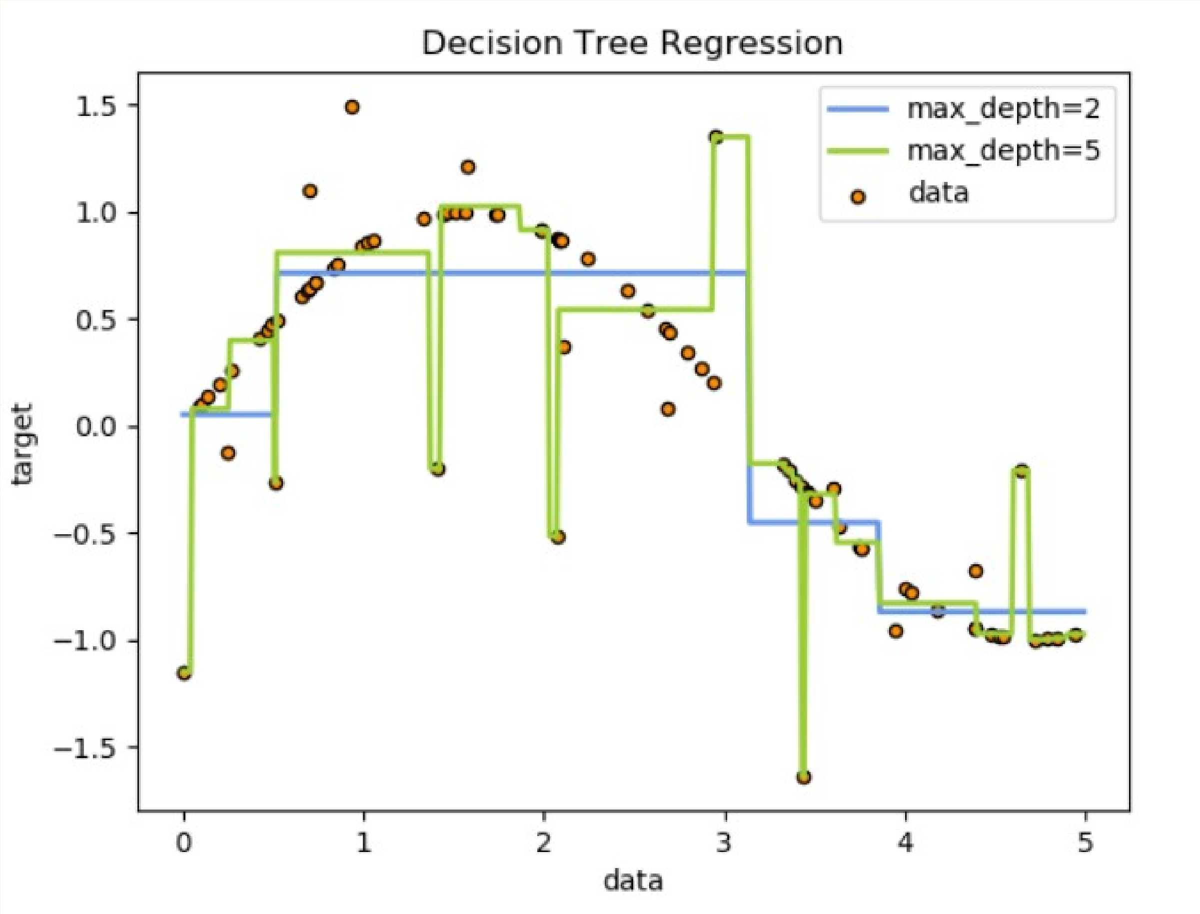


image: https://scikit-learn.org

the end