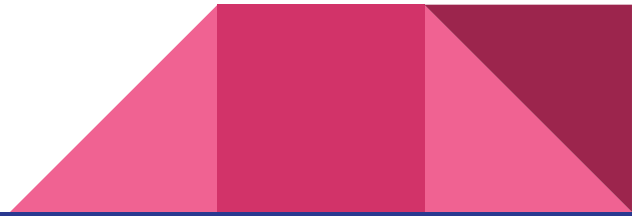
GOML2: ML in Go

Because…why not

Today

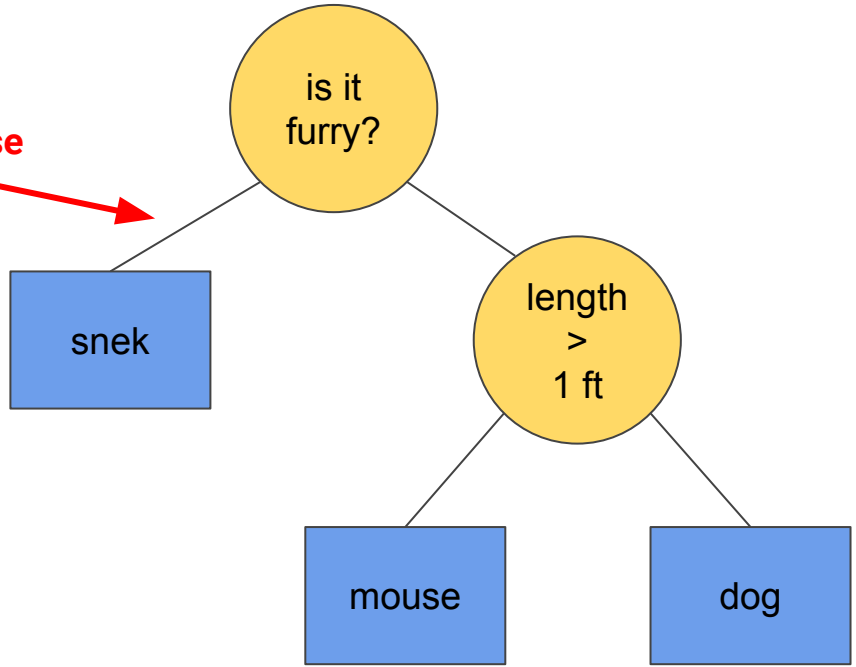
- **Classification trees**
- **Bias vs variance**
- **Cost-Complexity pruning**
- **Project timeline for GOML2**



Classification

Our decision tree

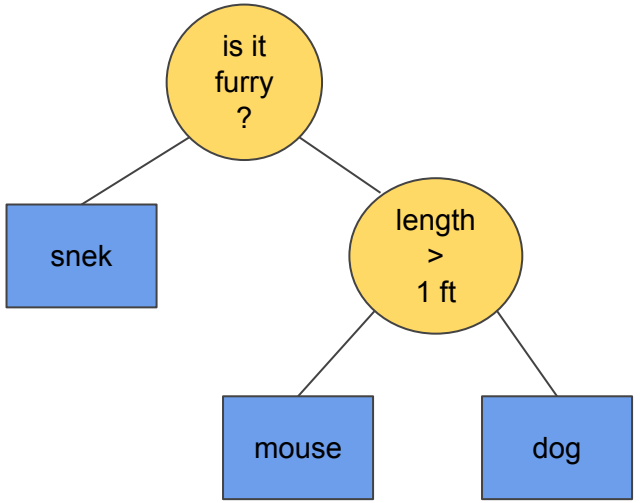
assume left is False



Classification

Some data - let's suppose our tree was trained on this data

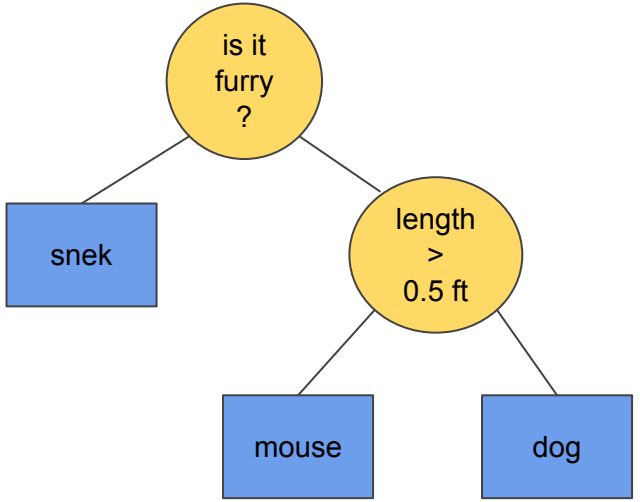
Furry?	Length	Class
X	3.0 ft	snek
✔	0.4 ft	mouse
✔	2.1 ft	dog
X	2.4 ft	snek
✔	3.0 ft	dog
✔	0.5 ft	dog
✔	0.6 ft	mouse



Classification

There are data points that are classified incorrectly

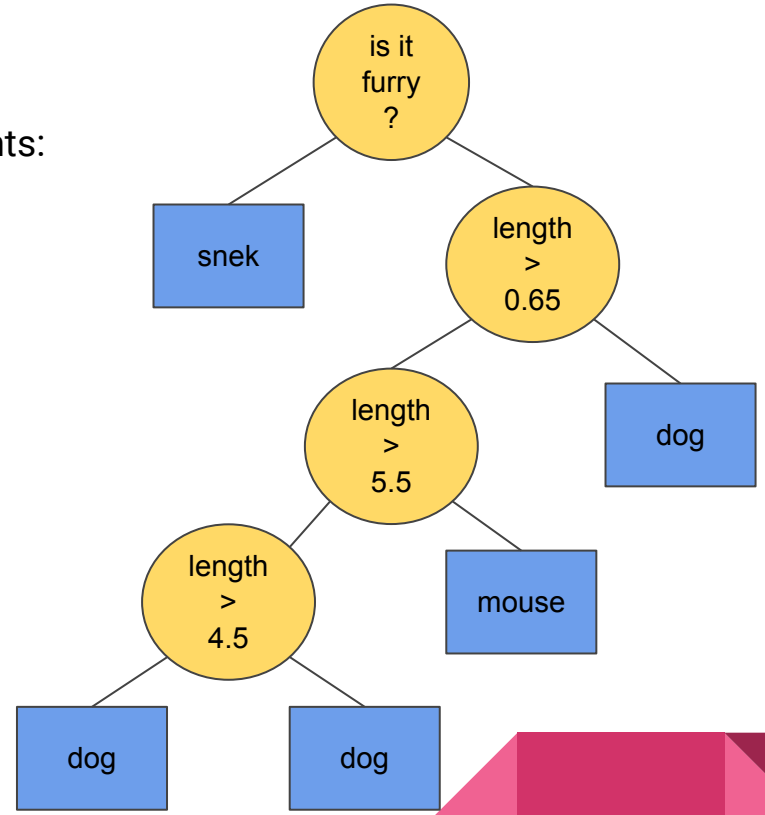
Furry?	Length	Class	
X	3.0 ft	snek	
✔	0.4 ft	mouse	
✔	2.1 ft	dog	
X	2.4 ft	snek	
✔	3.0 ft	dog	
✔	0.5 ft	dog	!!!
✔	0.6 ft	mouse	!!!



Classification

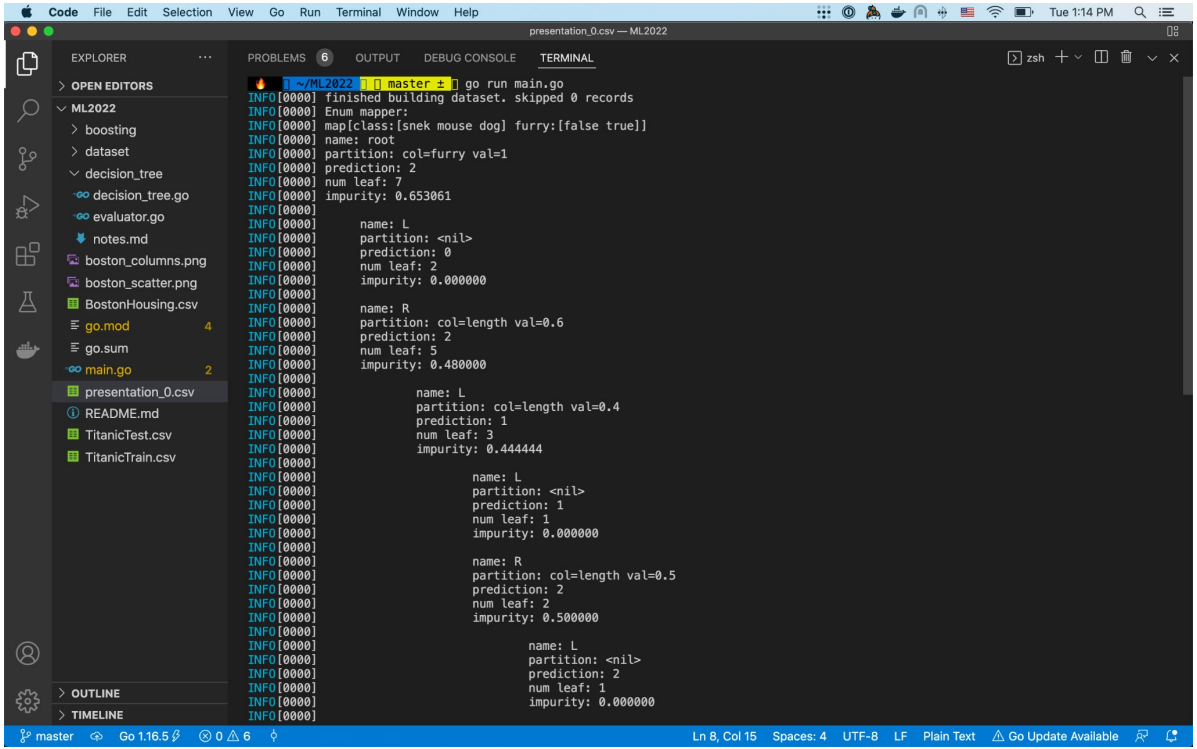
We could build a tree like this to get all of the data points:

Furry?	Length	Class
X	3.0 ft	snek
✔	0.4 ft	mouse
✔	2.1 ft	dog
X	2.4 ft	snek
✔	3.0 ft	dog
✔	0.5 ft	dog
✔	0.6 ft	mouse



Classification

How to do this with GOML2… Sorry, not as pretty 😔

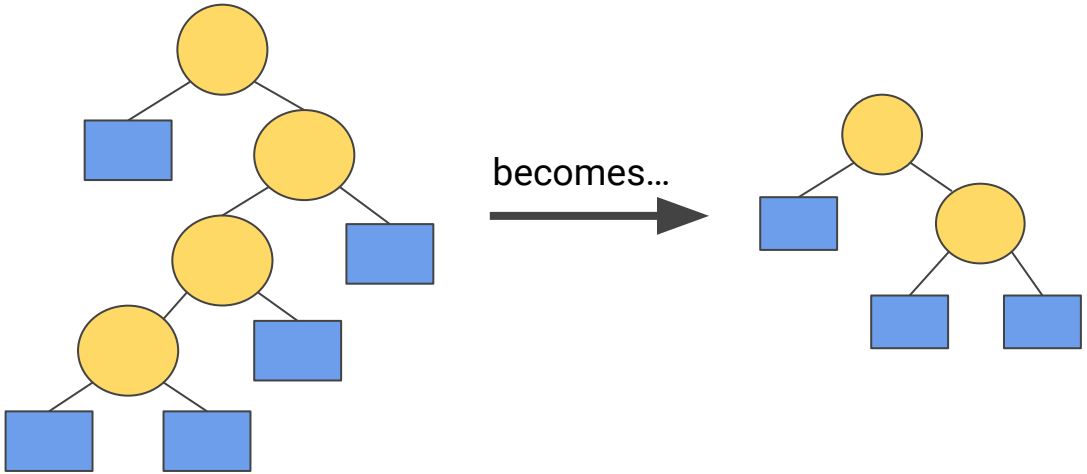


Bias vs variance

In most circumstances we want *some* error in our model to help decrease variance.

Pruning

By removing steps we fix the bias (which is already low) and decrease variance



Cost-complexity pruning

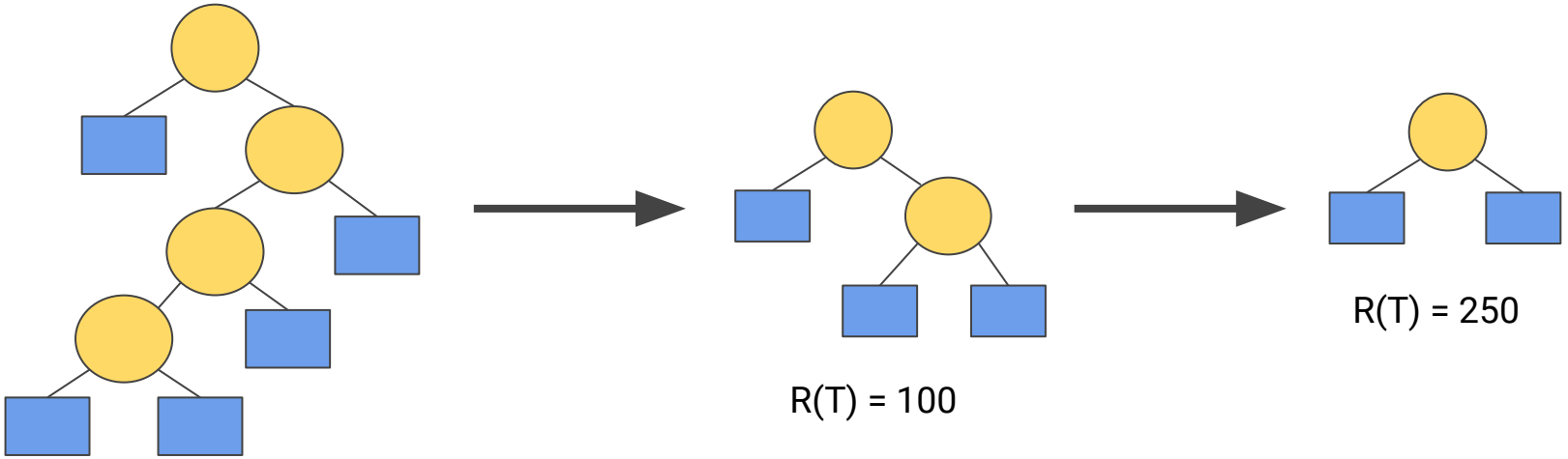
The best tree is the one that solves minimization problem:

$$R_\alpha(T) = R(T) + \alpha\,|f(T)|$$

R(T)	training error
α	hyperparameter
f(T)	returns set of leaves

Cost-complexity Pruning

Let’s suppose each tree has the following error



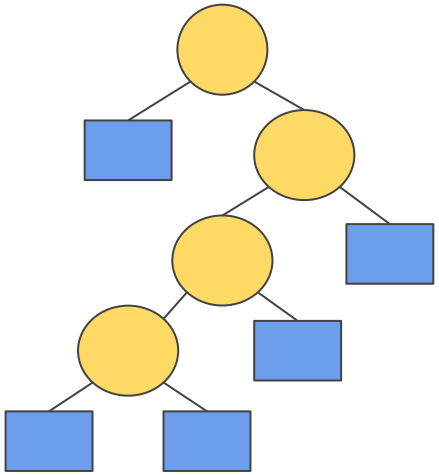
R(T) = 0

R(T) = 100

R(T) = 250

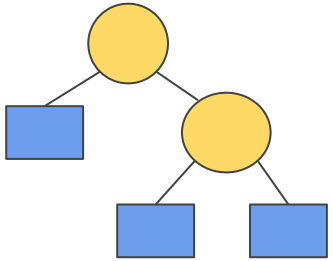
Cost-complexity Pruning

If α is fixed at 100, we get the following values for $R_\alpha(T)$:



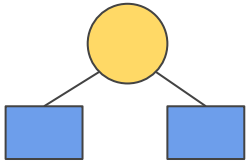
R(T) = 0
$R_\alpha(T)$ = 0 + 100* 5

= 500



R(T) = 100
$R_\alpha(T)$ = 100 + 100 * 3

= 400

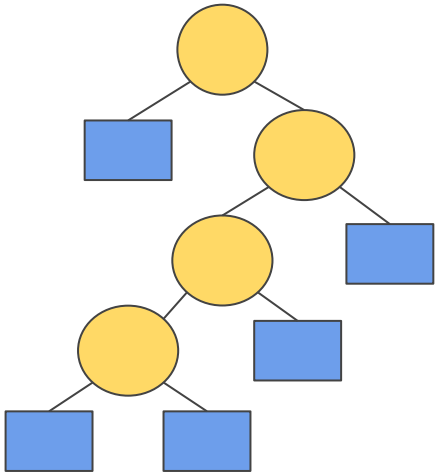


R(T) = 250
$R_\alpha(T)$ = 250 + 100 * 2

= 450

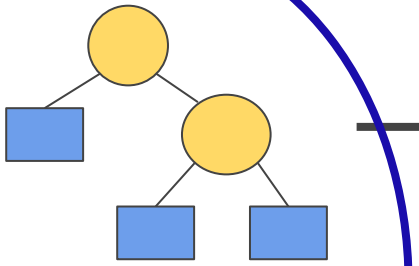
Cost-complexity Pruning

The lowest value for $R_\alpha(T)$ is the winner!



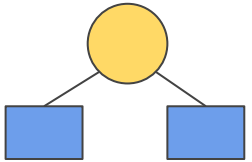
$R(T) = 0$
$R_\alpha(T) = 0 + 100 * 5$

= 500



$R(T) = 100$
$R_\alpha(T) = 100 + 100 * 3$

= 400

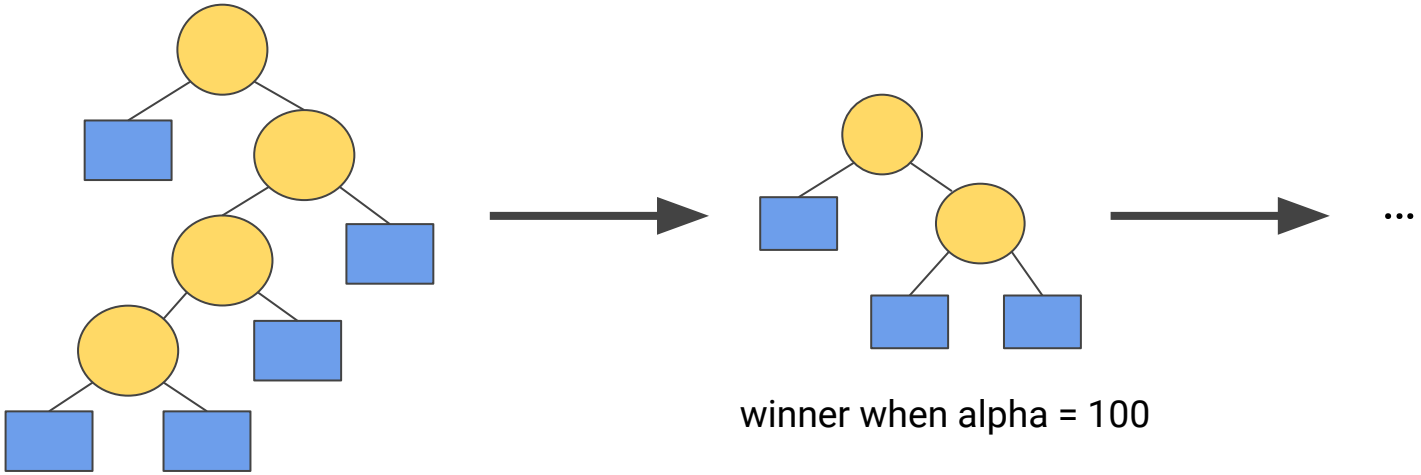


$R(T) = 250$
$R_\alpha(T) = 250 + 100 * 2$

= 450

Cost-complexity Pruning

But… any tree could be a winner, depending on alpha

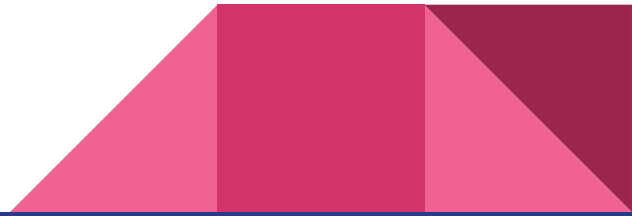


winner when alpha = 0

winner when alpha = 100

Cost-complexity pruning

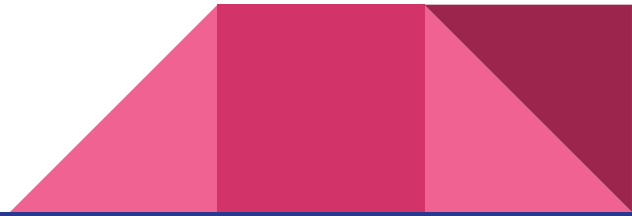
How do we know which alphas and subtrees to consider?



Cost-complexity pruning

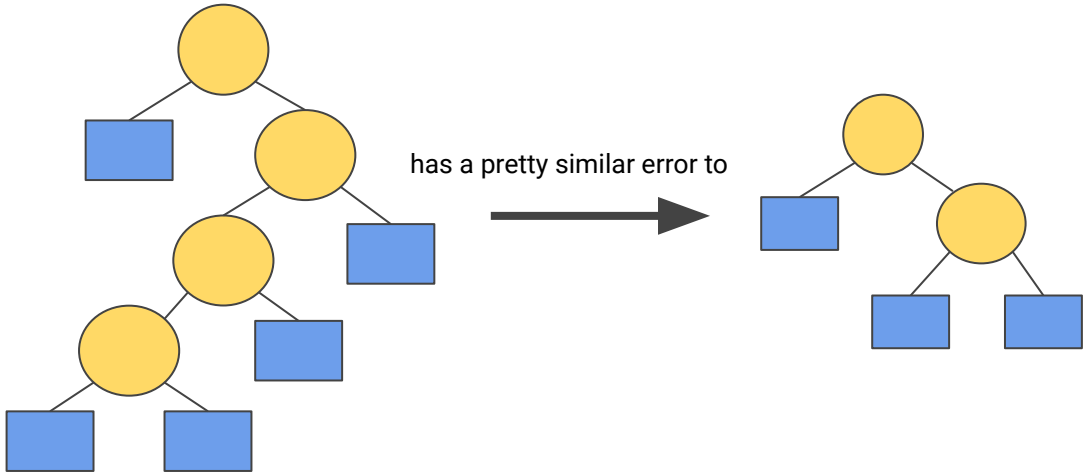
How do we know which alphas and subtrees to consider?

Answer lies in the *other* name for cost-complexity pruning, “weakest-link pruning”



Cost-complexity pruning

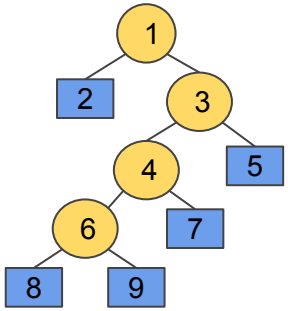
Intuition..

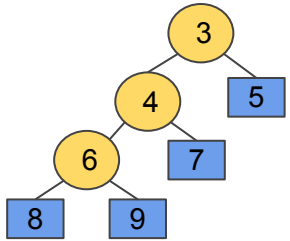


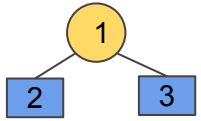
Cost-complexity pruning

Turning this into a minimization problem we get:

$$\min \left[R_\alpha(T - T_t) \quad - \quad R_\alpha(T) \right]$$







this is $T - T_3$

Cost-complexity pruning

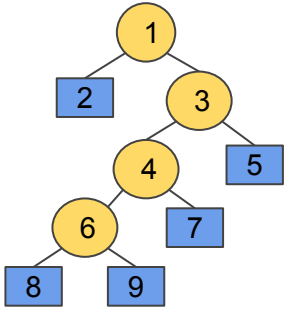
To find the value of α for weakest-link subtree (rooted at *t*):

$$\min \left[R_\alpha(T - T_t) - R_\alpha(T) \right]$$
$$= \min \left[R(T - T_t) - R(T) + \alpha \left(|f(T - T_t| - |f(T)| \right) \right]$$
$$= \min \left[R(t) - R(T_t) + \alpha \left(1 - |f(T)| \right) \right]$$

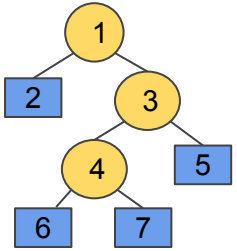
$$\alpha' = R(t) - R(T_t) \; / \; |f(T'_t)| - 1$$

Cost-complexity pruning

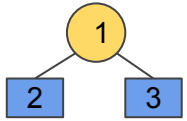
All of this to get a set of alphas and subtrees. Which do we choose?



a_1



a_2



a_3

K-fold cross validation

Suppose one row represents all observed data points

Furry?	Length	Class
X	3.0 ft	snek

K-fold cross validation

Generally we would split into two distinct sets, a **training** and a **testing** set.

Furry?	Length	Class
X	3.0 ft	snek

Furry?	Length	Class
✔	0.6 ft	???

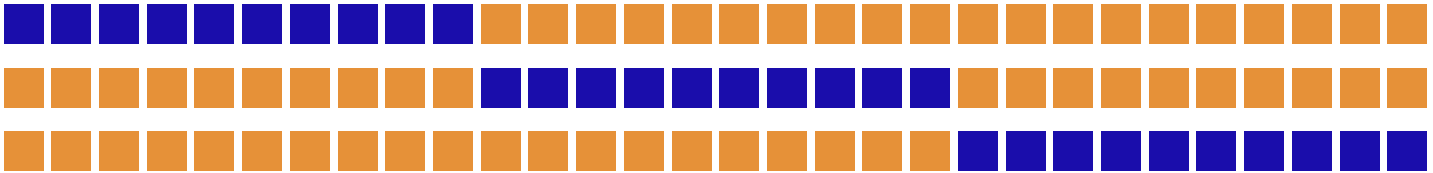
K-fold cross validation

But doing so reduces the amount of data that the model was trained on

Furry?	Length	Class
X	3.0 ft	snek

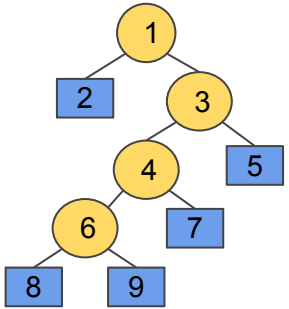
K-fold cross validation

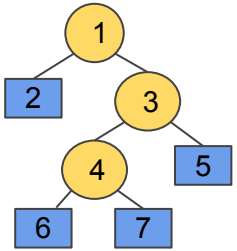
Instead we split the data points into K training / testing sets



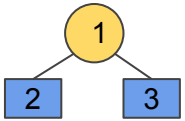
K-fold cross validation

We build new subtrees using the alphas that we found



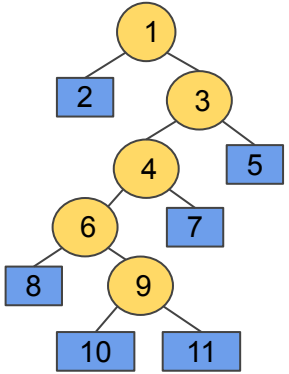


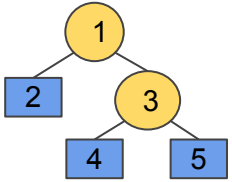
winner: a_1'



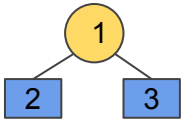
K-fold cross validation

Here's a new set of trees for the next training set



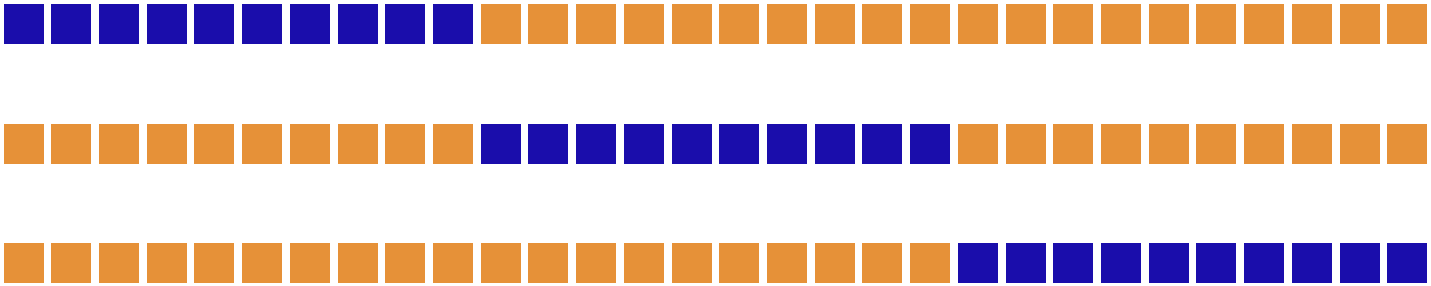


winner: a_2'



K-fold cross validation

Instead we split the data points into K training / testing sets



$$\frac{a_1' + a_2' + .. + a_K'}{K} = a'$$

this is the best value for alpha using entire dataset.

Current Scope

- **Classic ML**
  - **Datasets**
    - **Visualization**
  - **Unsupervised learning**
  - **Supervised learning**
    - **Decision trees (regression/classification)**
    - **Ensemble learning**
      - **Gradient tree boosting**
  - **Model selection and validation**
    - **K-fold cross validation**
- **Deep learning**
  - I few years ago I built simple NN w/ backprop from scratch 🙂
    - at www.robertkotcher.me

References

- StatQuest: https://www.youtube.com/watch?v=D0efHEJsfHo&t=779s
- MLWiki: http://mlwiki.org/index.php/Cost-Complexity_Pruning
- ScikitLearn source: https://github.com/scikit-learn/scikit-learn