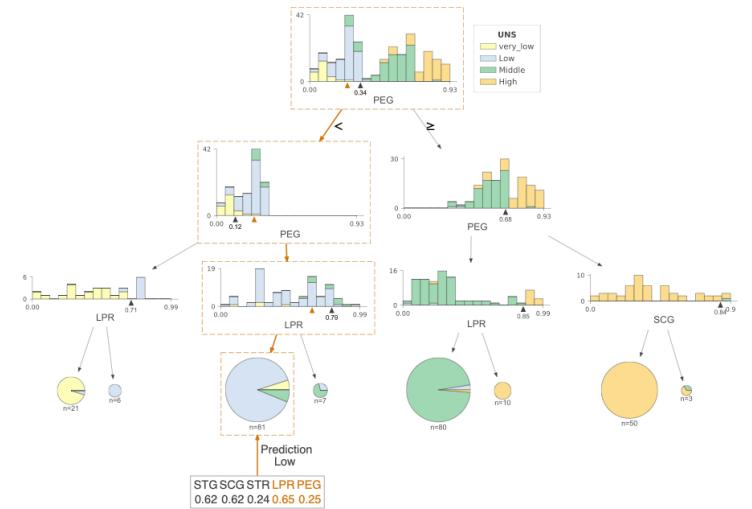
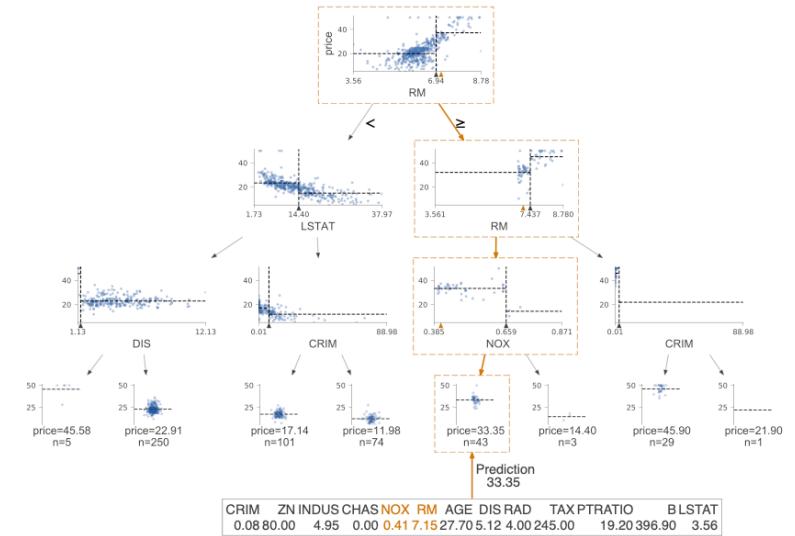


Decision trees

Terence Parr
MSDS program
University of San Francisco

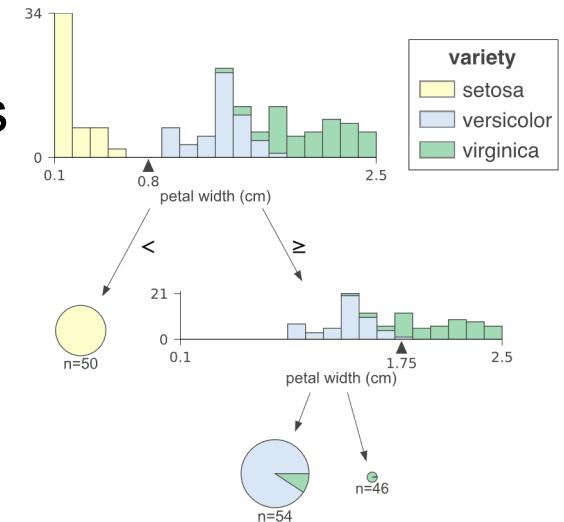


The essence of decision trees

- Decision trees are like kNN , but with rectangular not polygonal hypervolumes and dynamic k not fixed k
- Partition feature space into tight rectangular hypervolumes of feature space with constraint that we want y values to be as pure/similar as possible for records in that hypervolume
- Not so tight that the hypervolumes have too few feature vectors (records/samples), which tends to overfit the training data
- Prediction for unknown vector:
 - predict the mean y for training samples in that hypervolume (regression)
 - predict the mode (most common) y in that hypervolume (classification)
- Binary trees just happen to be an efficient implementation

Basic properties of decision tree models

- Decision trees consist of internal decision nodes and leaf nodes that make predictions
- Each input record (feature vector) is contained in exactly one leaf node
- Each leaf has 1 or more records whose y 's are as pure as possible (model hyperparameters affect number of records per leaf)
- Prediction proceeds from root to leaf, testing var/value combos
- Regressor: leaf predicts average of y for associated records
- Classifier: leaf predicts mode (most common) class

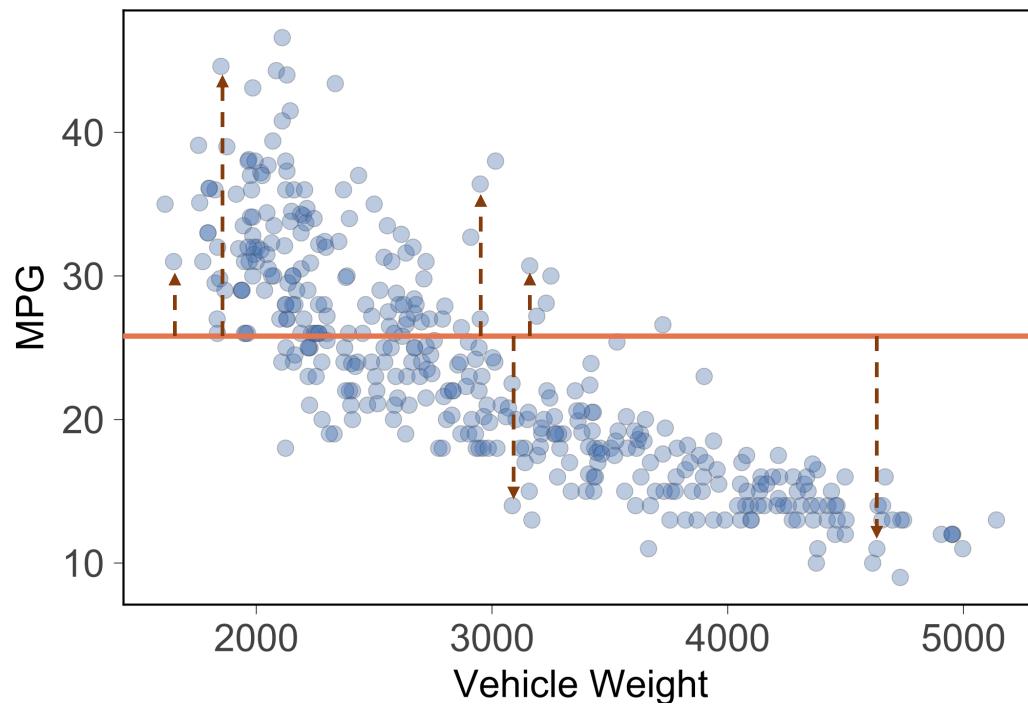


Most images in these slides from package <https://github.com/parrt/dtreeviz>

Let's reinvent decision trees

Let's create a simple regressor in 1D

Predict MPG
from weight



Predict a single
constant for
entire region
(the mean)

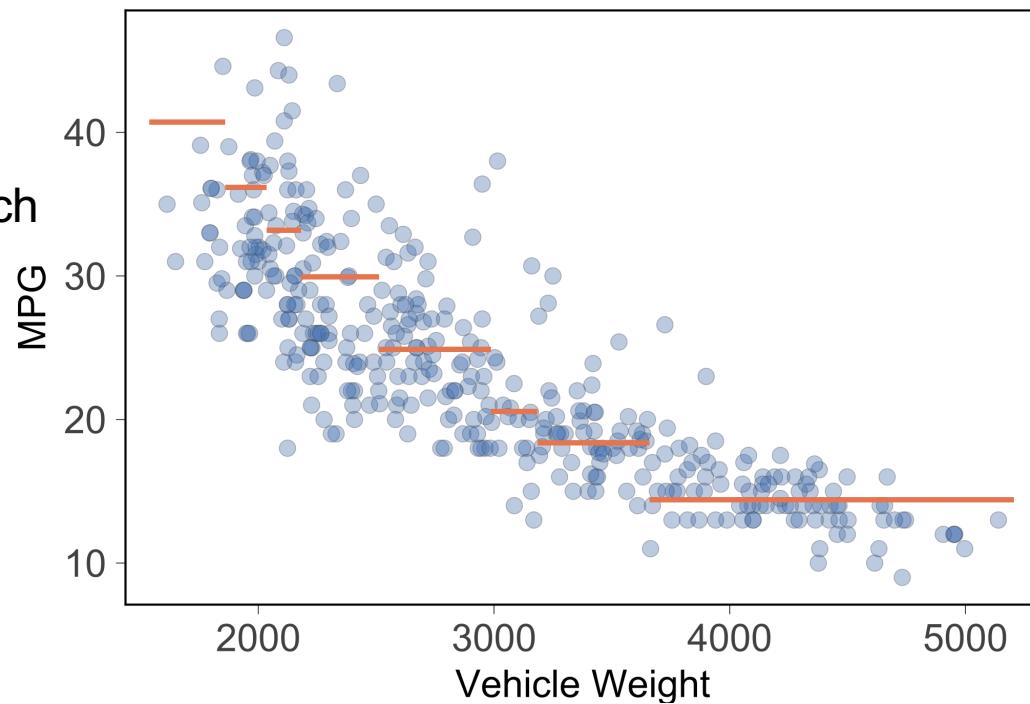
MSE is high,
 y variance high

↑ Residuals from
point to region
mean

Improve by partitioning, using multiple lines

*WHERE DO
WE SPLIT??*

Find ranges of WGT with similar MPG values, which yields a lower MSE

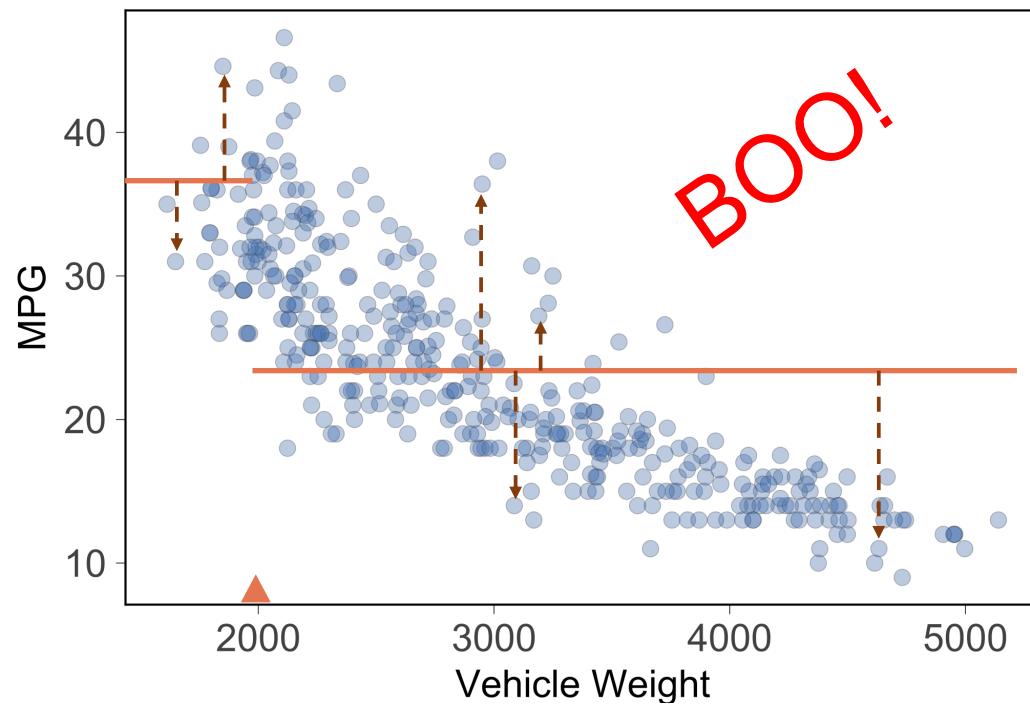


Can only use horizontal lines, but can use lots

Each region predicts mean in piecewise fashion

Strategy: find split point giving least MSE

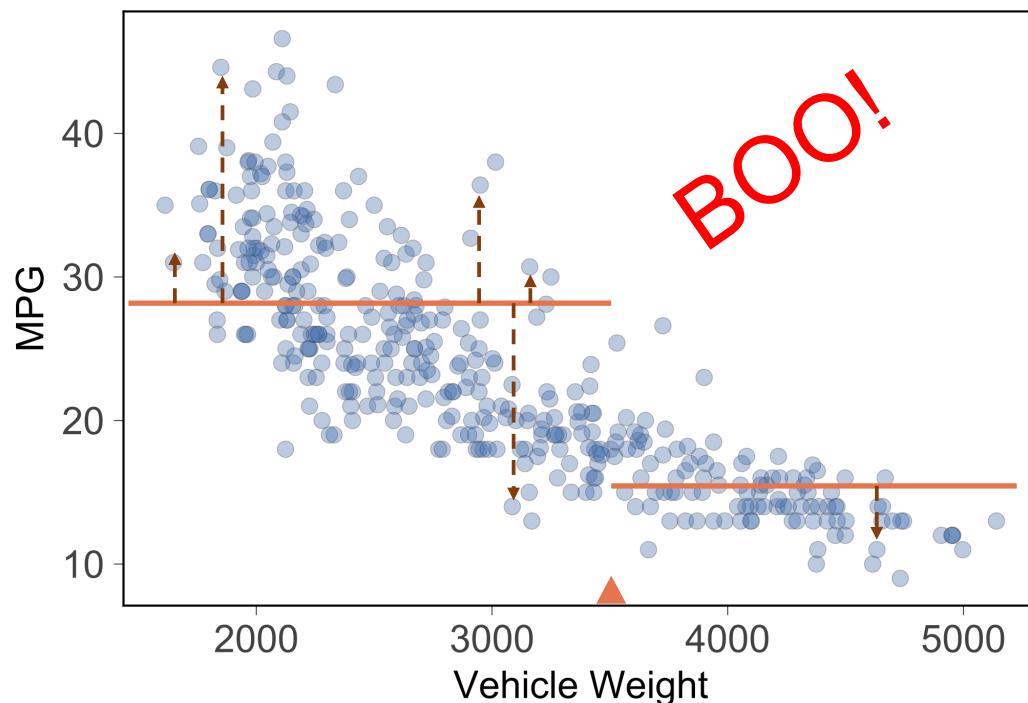
Split WGT
into two
subregions,
each
predicting
mean



WGT=2000 is
BAD CHOICE:
MSE very high
(subregion y 's
are still
dissimilar)

Strategy: find split point giving least MSE

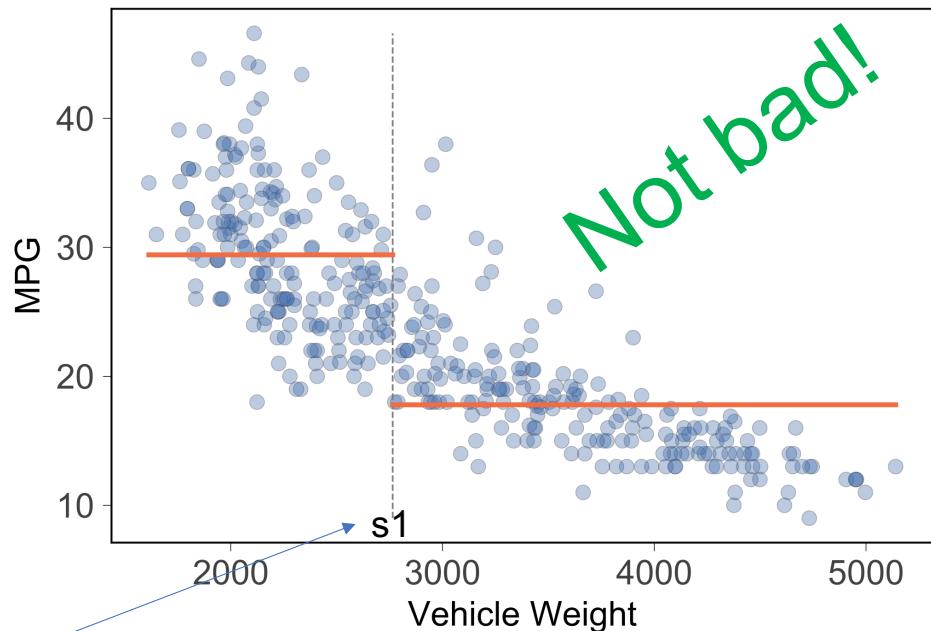
Split WGT
into two
subregions,
each
predicting
mean



WGT=3500 is
ANOTHER
BAD CHOICE:
MSE very high
(still dissimilar)

Note: MSE for
mean model is
same as variance
(average squared
difference from
mean)

A split exists that gives min MSE for regions



Technique:
Exhaustively
check all feature
values, computing
MSE or variance
of subregions for
each split

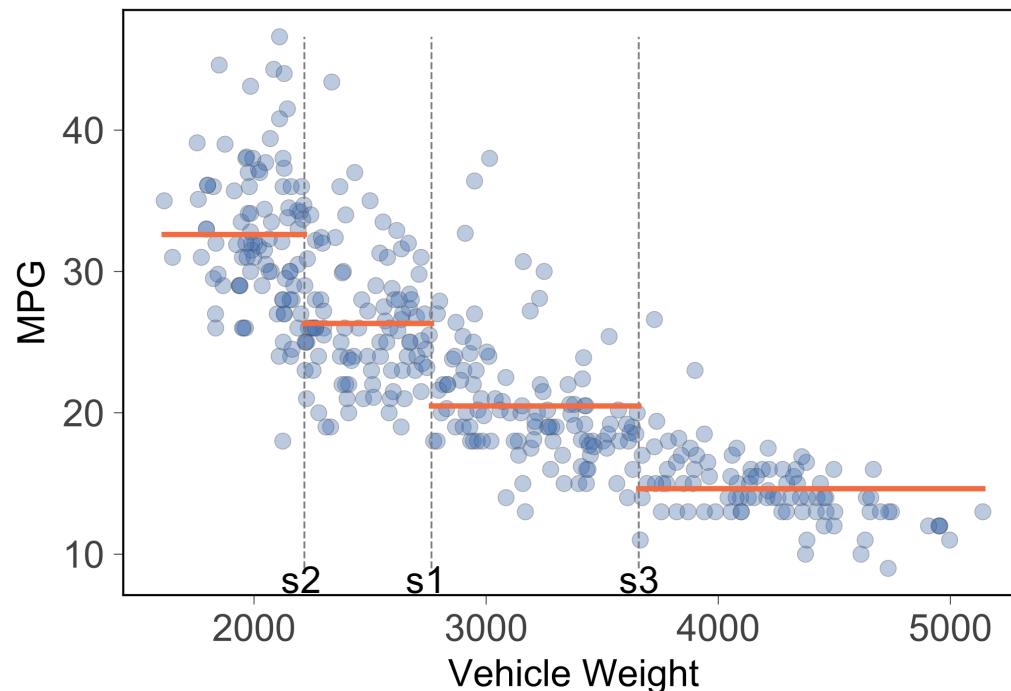
Choose split point
giving min MSE

Slide s_1 from left to right over x_i range, computing subregion MSE
Choose WGT location with min average MSE for subregions

Now split those 2 regions to get 4 regions

Split s1 stays,
recursively split
left/right regions to
get splits s2, s3

Kinda like binary
search or other
divide-and-conquer
strategy

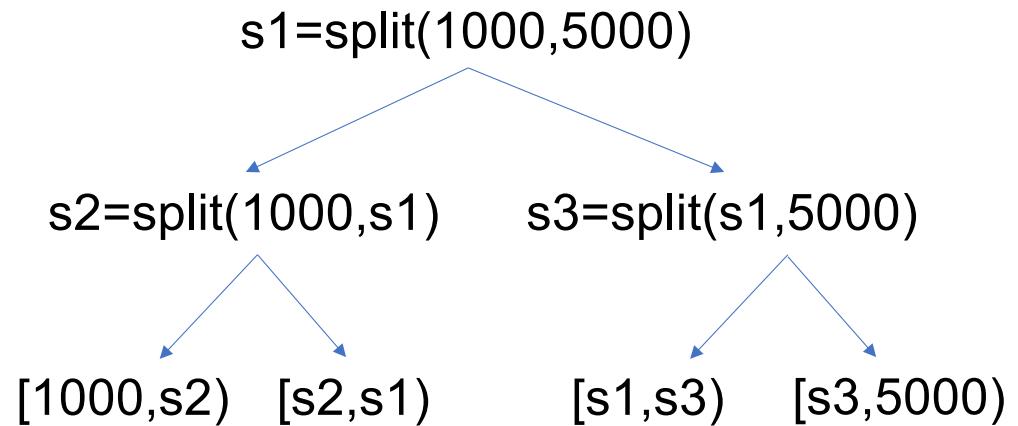
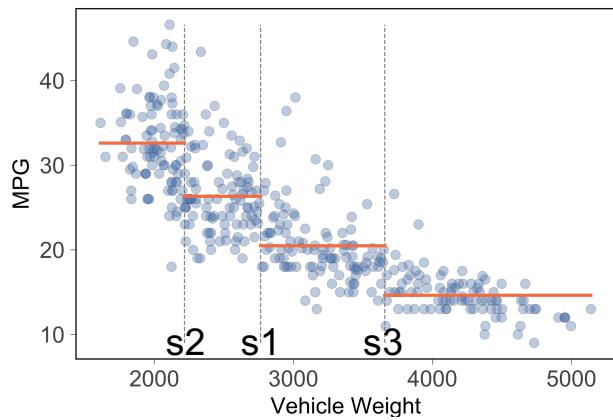


Slide s2 from left to s1, computing MSEs; choose x location with min avg MSE
Slide s3 from s1 to right, computing MSEs; choose x location with min avg MSE



UNIVERSITY OF SAN FRANCISCO

Recursive-call tree from model training gives regions defined by splits s_1, s_2, s_3

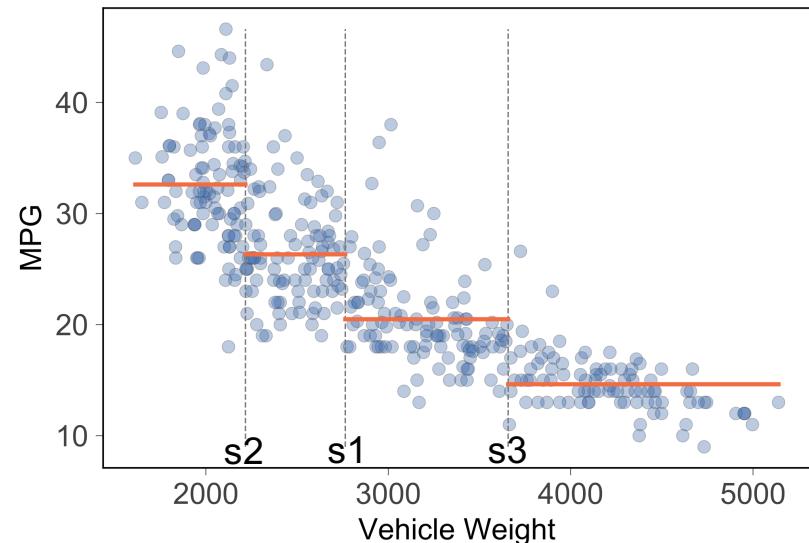


Split (recurse) until one of:

- All potential splits do not reduce MSE
- All nodes have min num samples
- Max number of splits reached
- Etc...

*Predictions are avg of MPG
(target) values in subregions*

Hardcoded non-tree model implementation

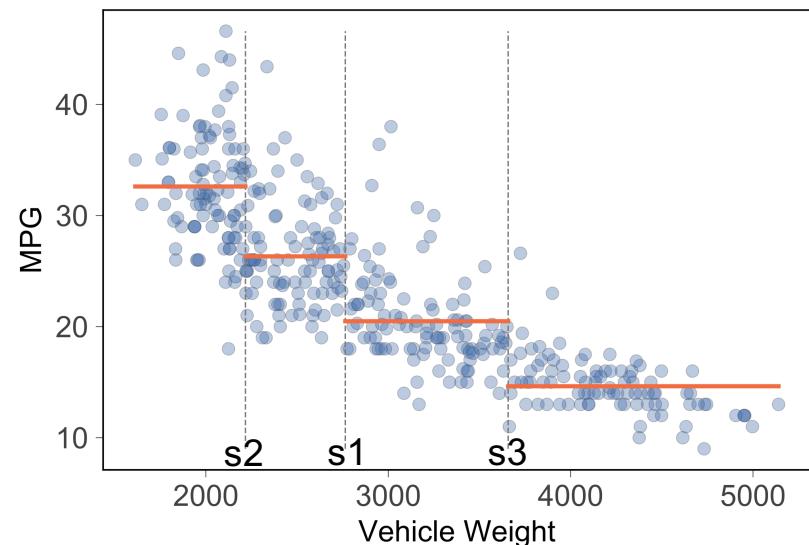


To partition space, test in recursion/split order

if $x < s_1$ and $x < s_2$: predict 32.6
if $x < s_1$ and $x \geq s_2$: predict 26.3
if $x \geq s_1$ and $x < s_3$: predict 20.5
if $x \geq s_1$ and $x \geq s_3$: predict 14.6

Note repeated comparisons!

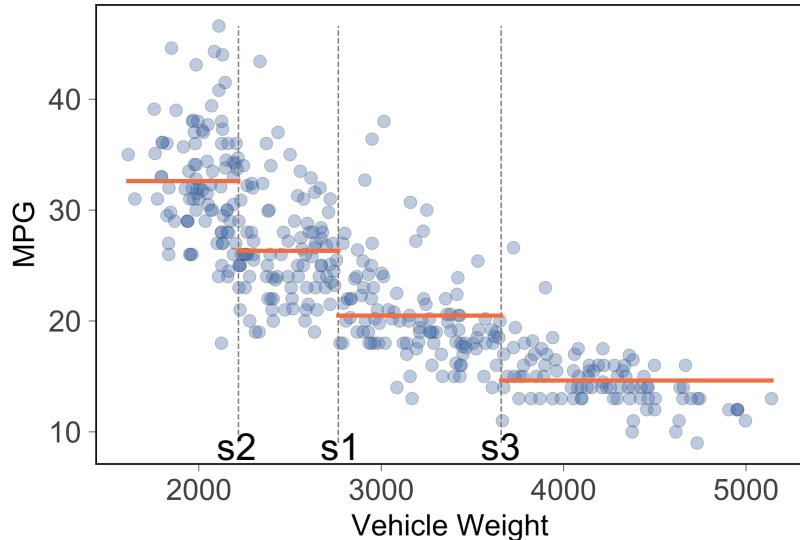
Factor the split comparisons for efficiency



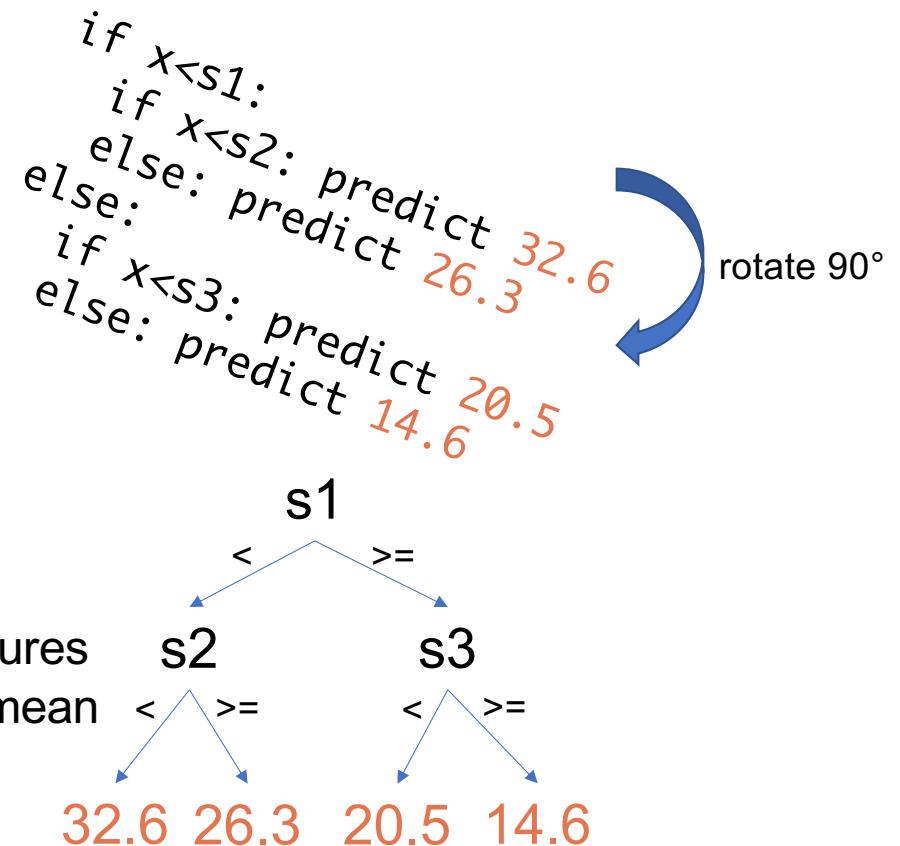
```
if x<s1:  
    if x<s2: predict 32.6  
    else: predict 26.3  
else:  
    if x<s3: predict 20.5  
    else: predict 14.6
```

But, don't want to hardcode model!

Represent nested conditionals as tree



Internal nodes test features
Leaves predict region mean

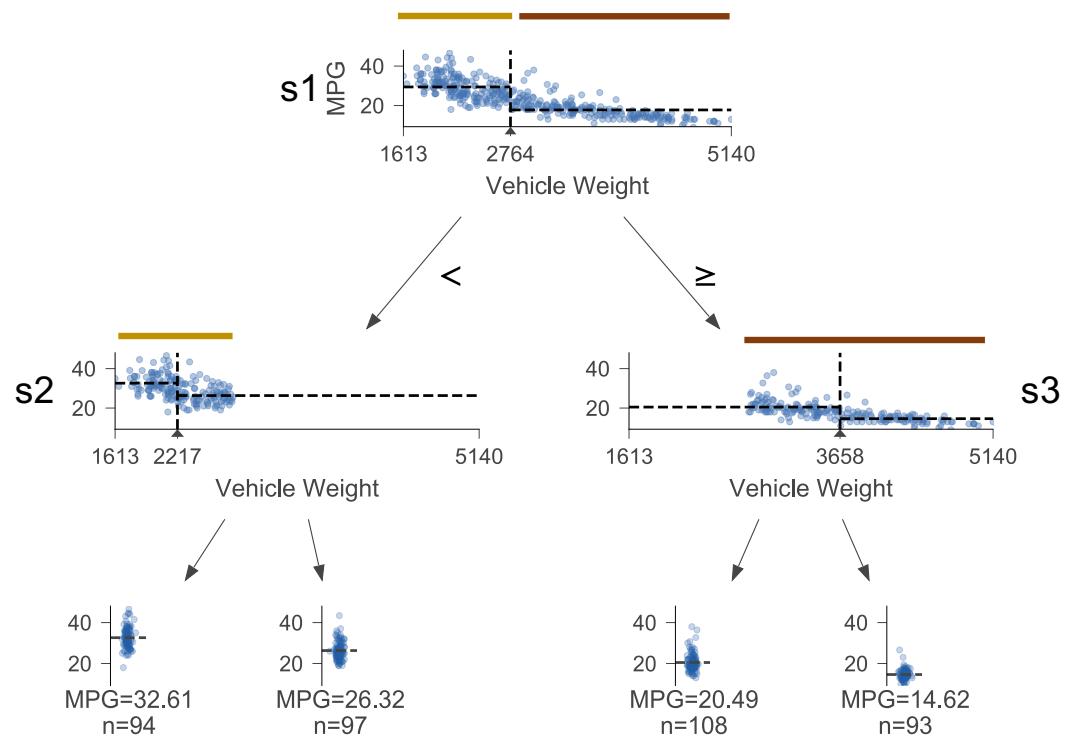
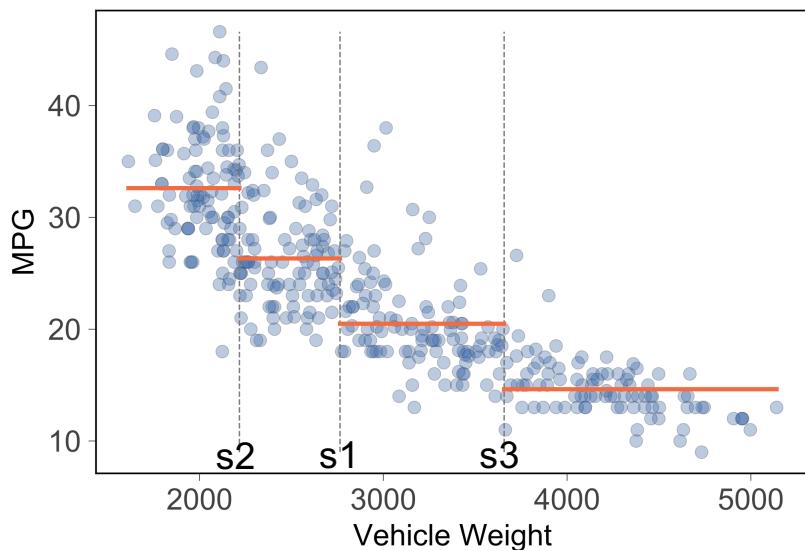


We morph tree of recursion from training into decision tree!



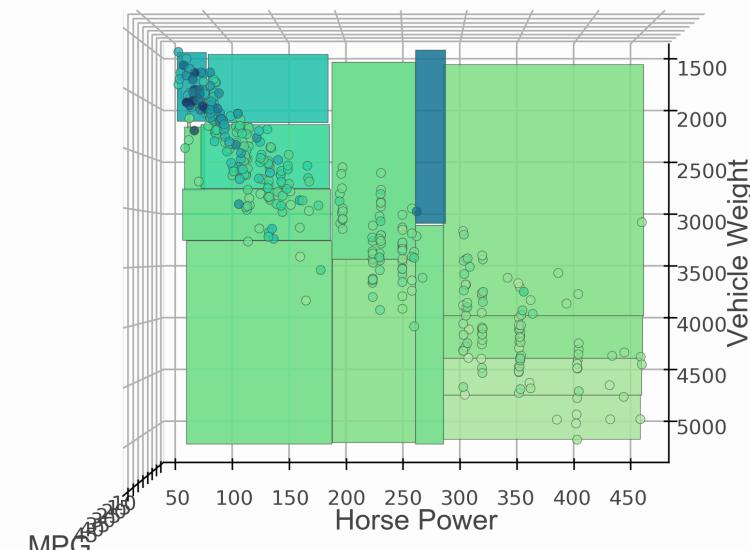
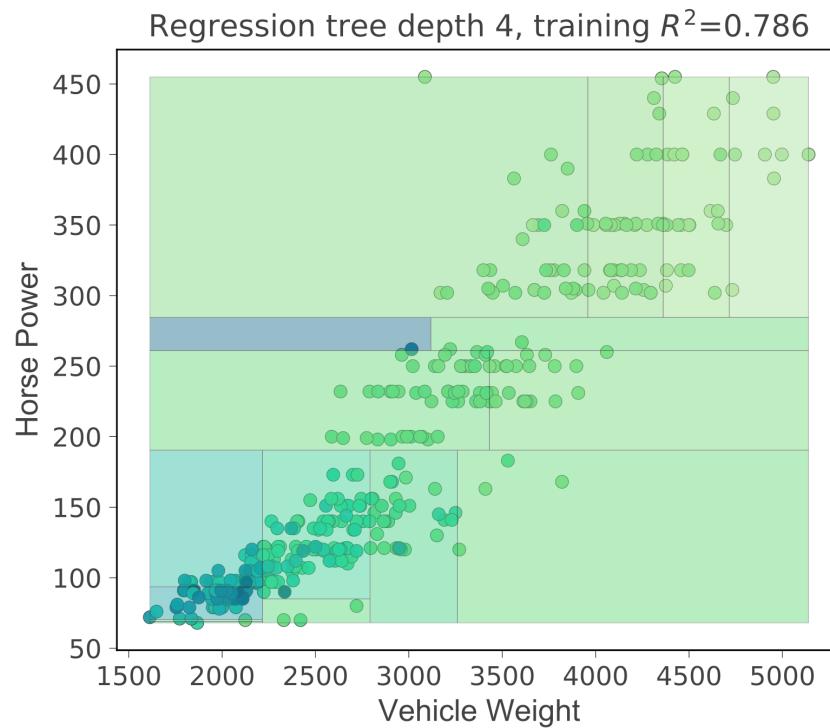
UNIVERSITY OF SAN FRANCISCO

1D feature space vs dtreeviz decision tree

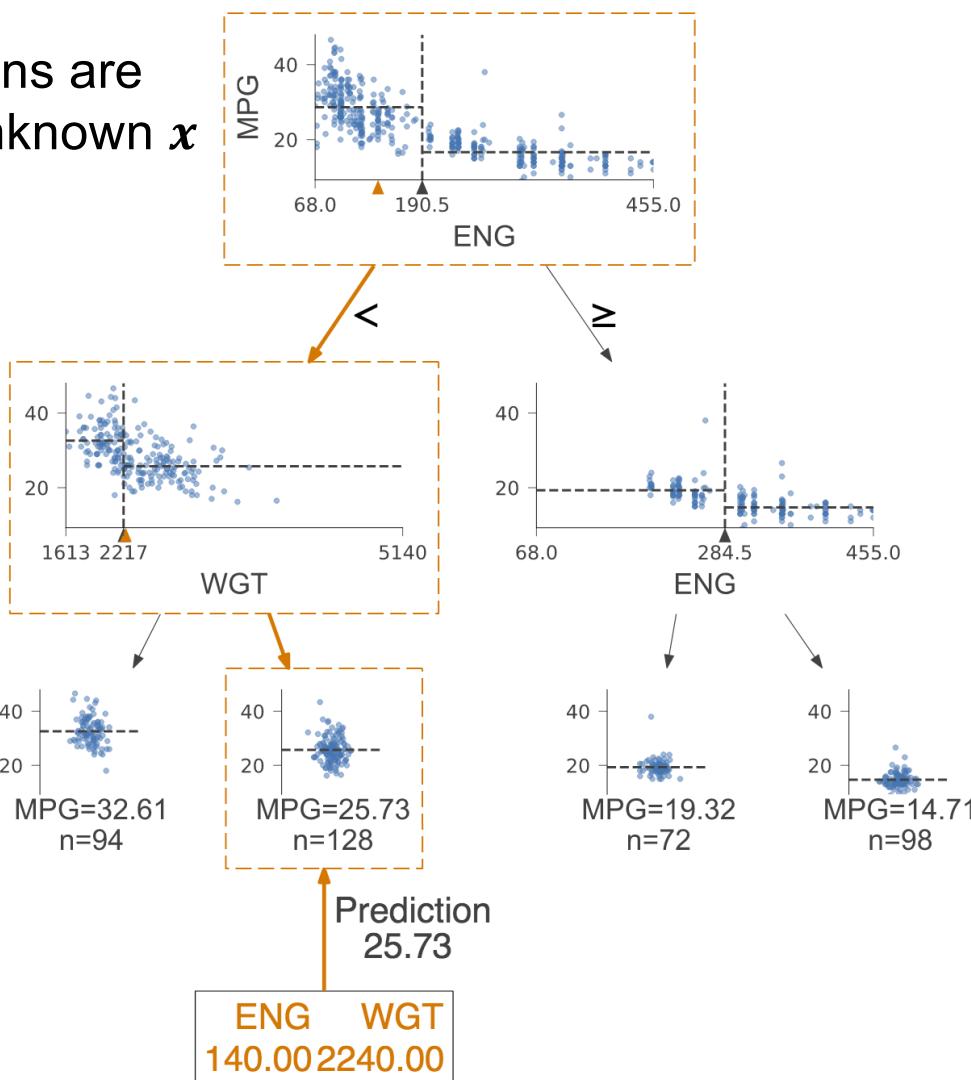
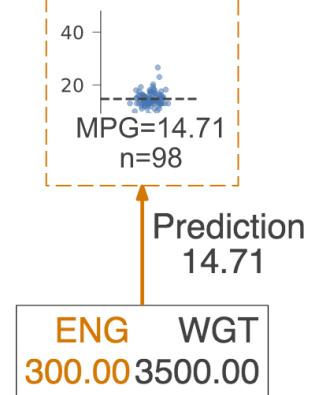
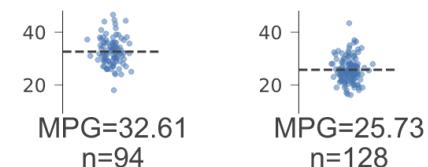
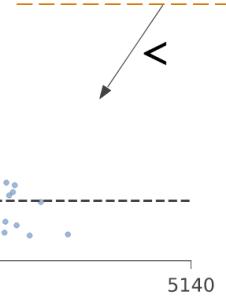
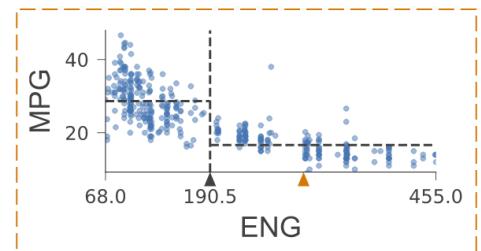


<https://github.com/parrt/msds621/blob/master/notebooks/trees/partitioning.ipynb>

2D regressor feature space (heatmap, 3D)



How predictions are made from unknown x



An aside: partitioning, tree viz done with custom library

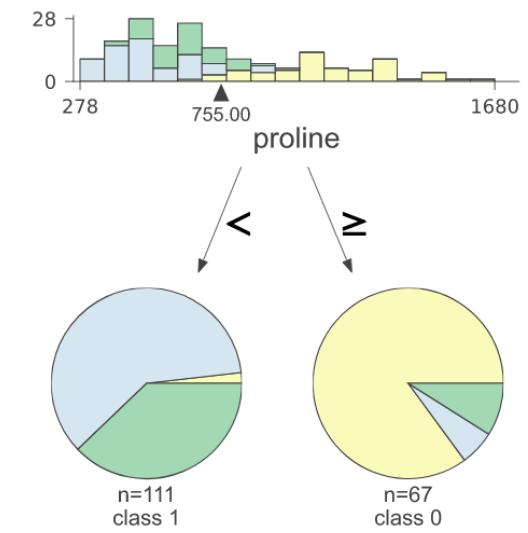
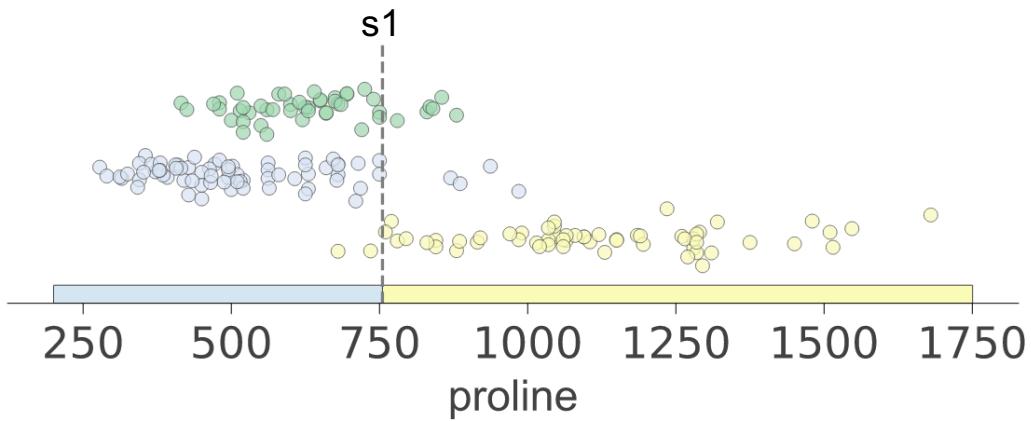
- Do “`pip install dtreeviz`”
- Partially built with Prince Grover, previous MSDS student
- See <https://github.com/parrt/dtreeviz> and the article for more detail: <https://explained.ai/decision-tree-viz/index.html>
- Advice: never accept status quo; always strive for more / better
- See “*How to lead a fulfilling life by being dissatisfied*” buried in my talk on decision tree viz
https://twitter.com/the_antlr_guy/status/1120359898062000128

Classifiers

Classifiers split feature space too

Predict wine
from proline

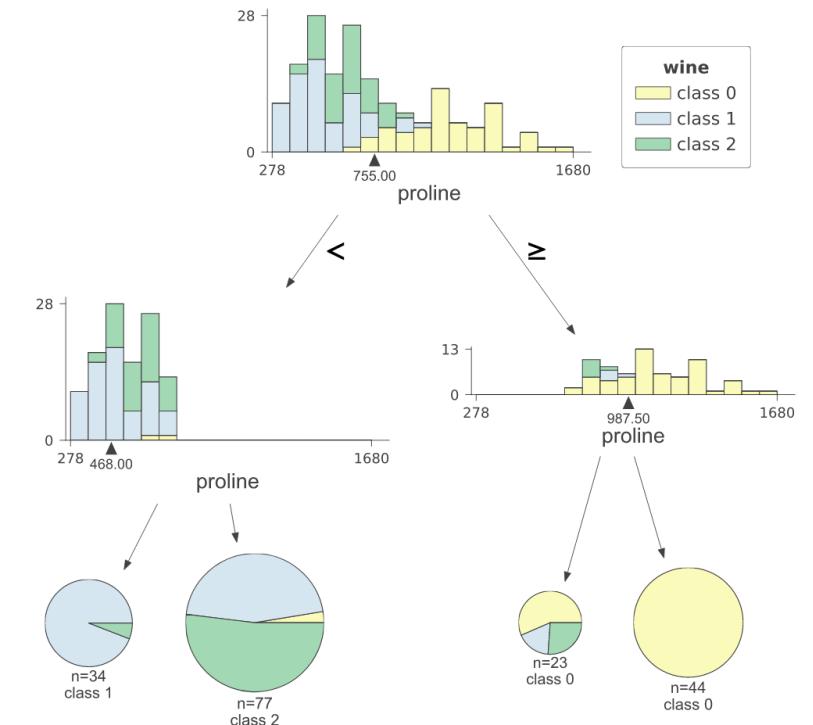
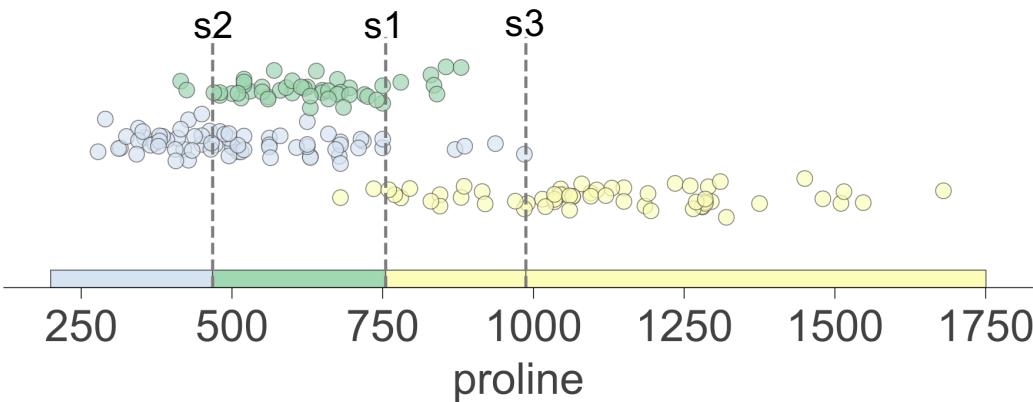
- Internal decision nodes test features just like regressor trees
- Leaves predict most common target category (mode) not mean
- Find split that decreases average impurity of left/right subregions (we'll need a definition of impurity for categories)



<https://github.com/parrt/msds621/blob/master/notebooks/trees/partitioning.ipynb>

Split s1 subregions into more subregions

All splits use same proline variable



Still not very pure though

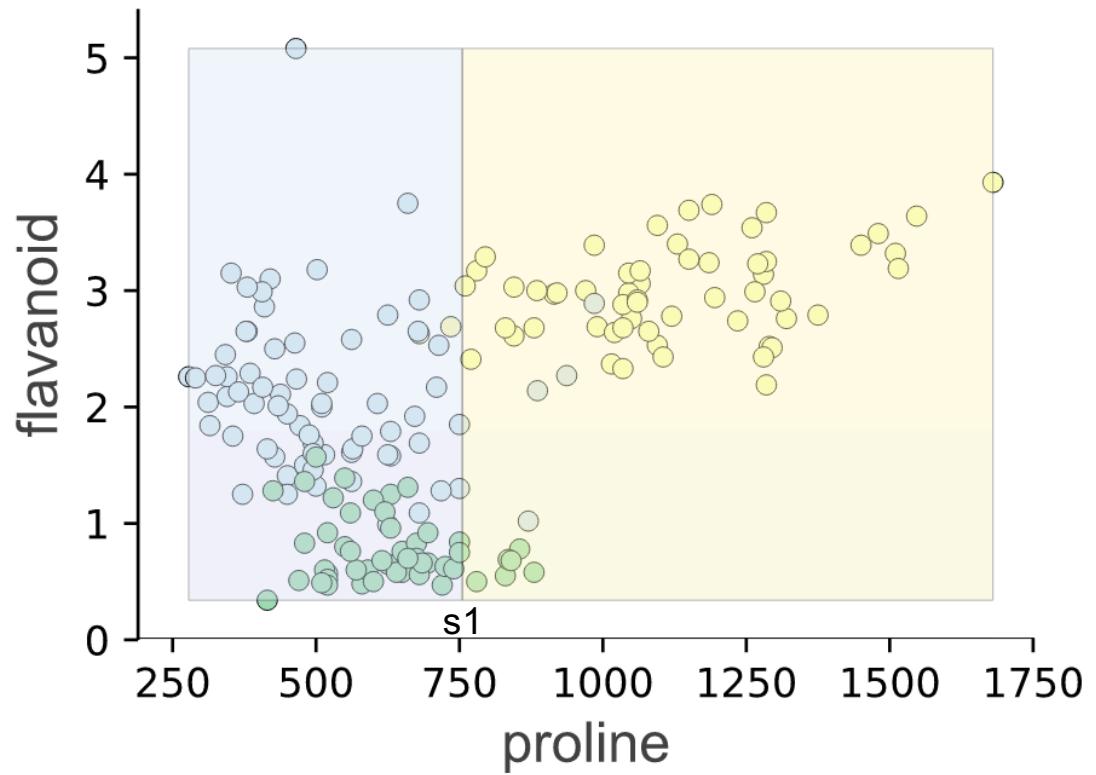


UNIVERSITY OF SAN FRANCISCO

To improve predictions:
Use 2 features and
split 2D feature
space into regions

*Training looks for (feature, split point)
 combos giving more **pure** subregions.*

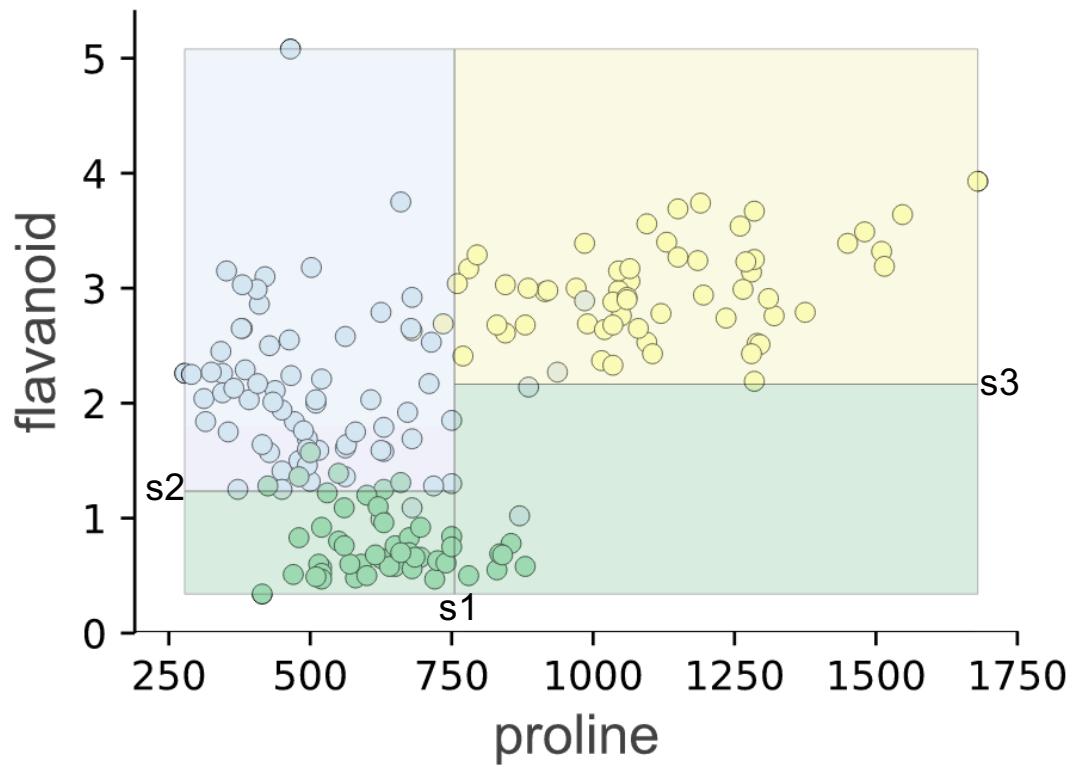
*To test: decision nodes compare single feature
 value in subset of records to split point*



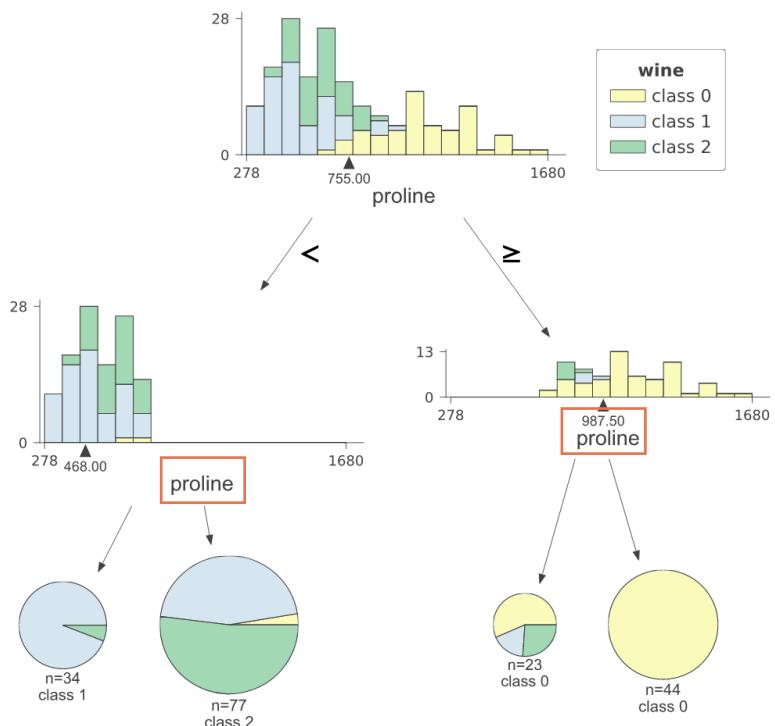
To improve predictions:
Use 2 features and
split 2D feature
space into regions

*Training looks for (feature, split point)
 combos giving more **pure** subregions.*

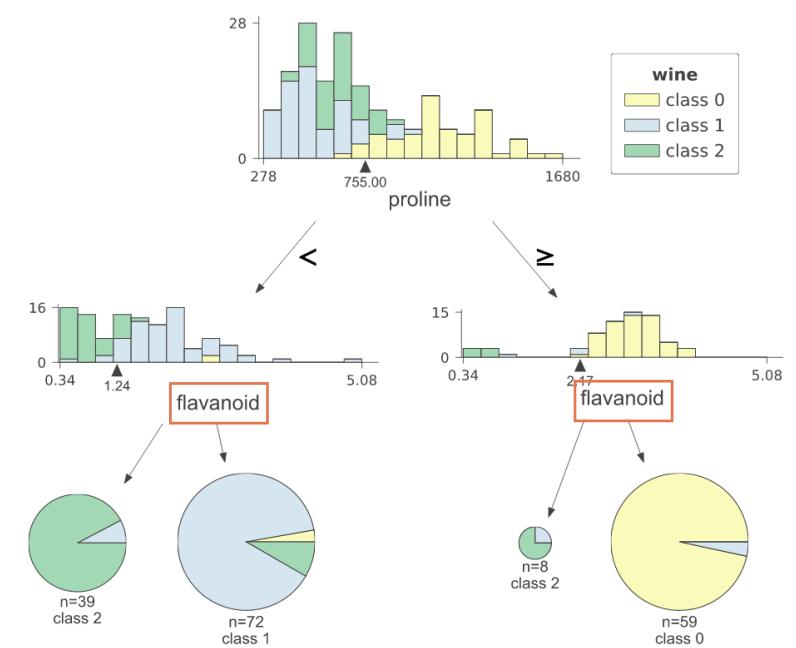
*To test: decision nodes compare single feature
 value in subset of records to split point*



Compare depth=2 trees for 1D, 2D vars

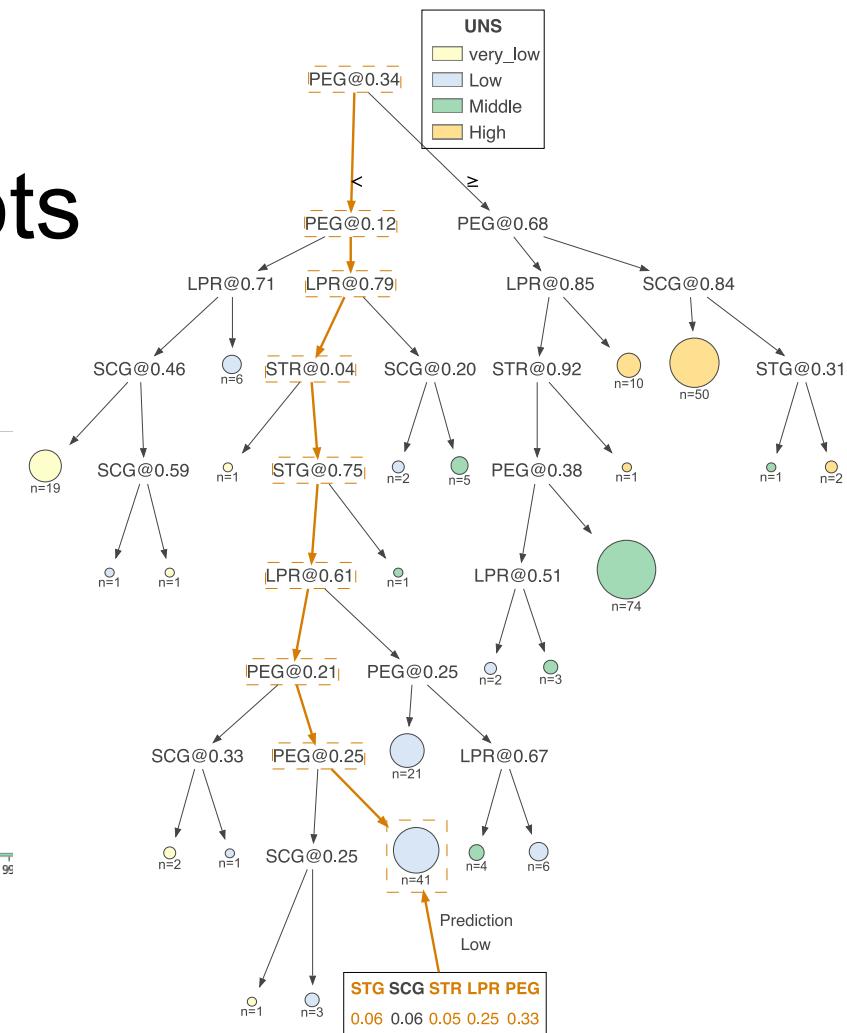
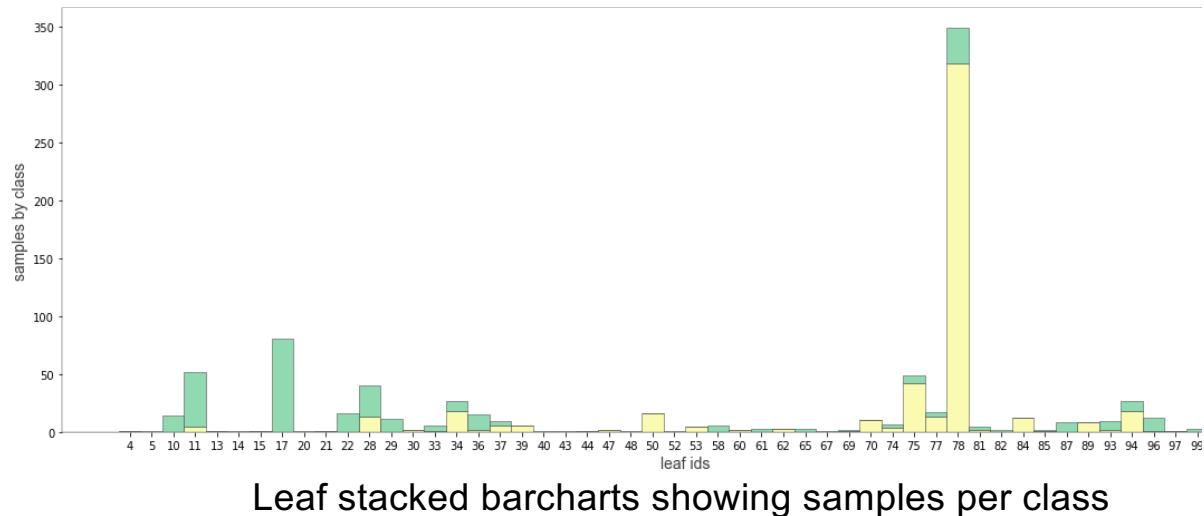


All splits use same proline variable



Splits use proline and flavanoid

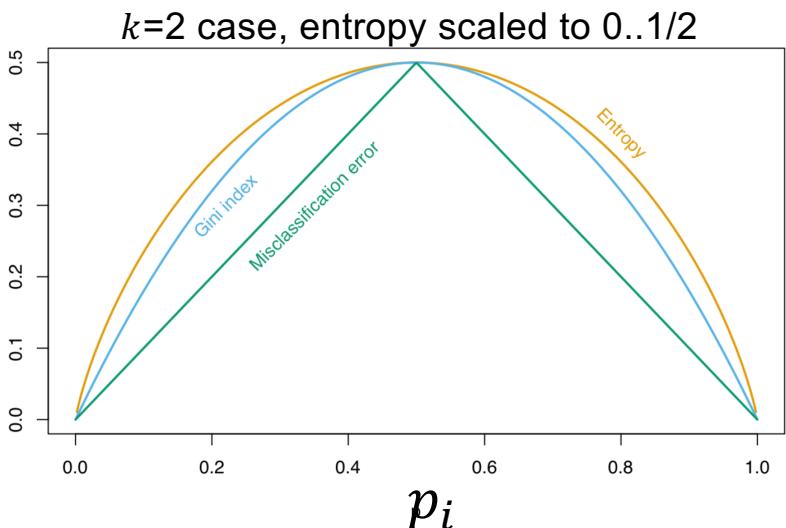
For bigger trees, can do nonfancy plot, leaf plots



Node impurity: Gini impurity

- Measures y 's uncertainty, like entropy except faster
- Minimize gini during node splitting
- Let p_i be the fraction of y values with class i and $k = \text{num of classes}$
- Gini range is $0..(k-1)/k$
- Max uncertainty when all $p_i = p_j$:
$$p_i = 1/k \text{ so gini} = 1 - \sum_{i=1}^k \frac{1}{k^2} = 1 - 1/k = (k - 1)/k$$
- Min uncertainty when a single $p_i = 1$ and other $p_j = 0$

$$Gini(y) = \sum_{i=1}^k p_i \left[\sum_{j \neq i}^k p_j \right] = \sum_{i=1}^k p_i(1 - p_i) = 1 - \sum_{i=1}^k p_i^2$$

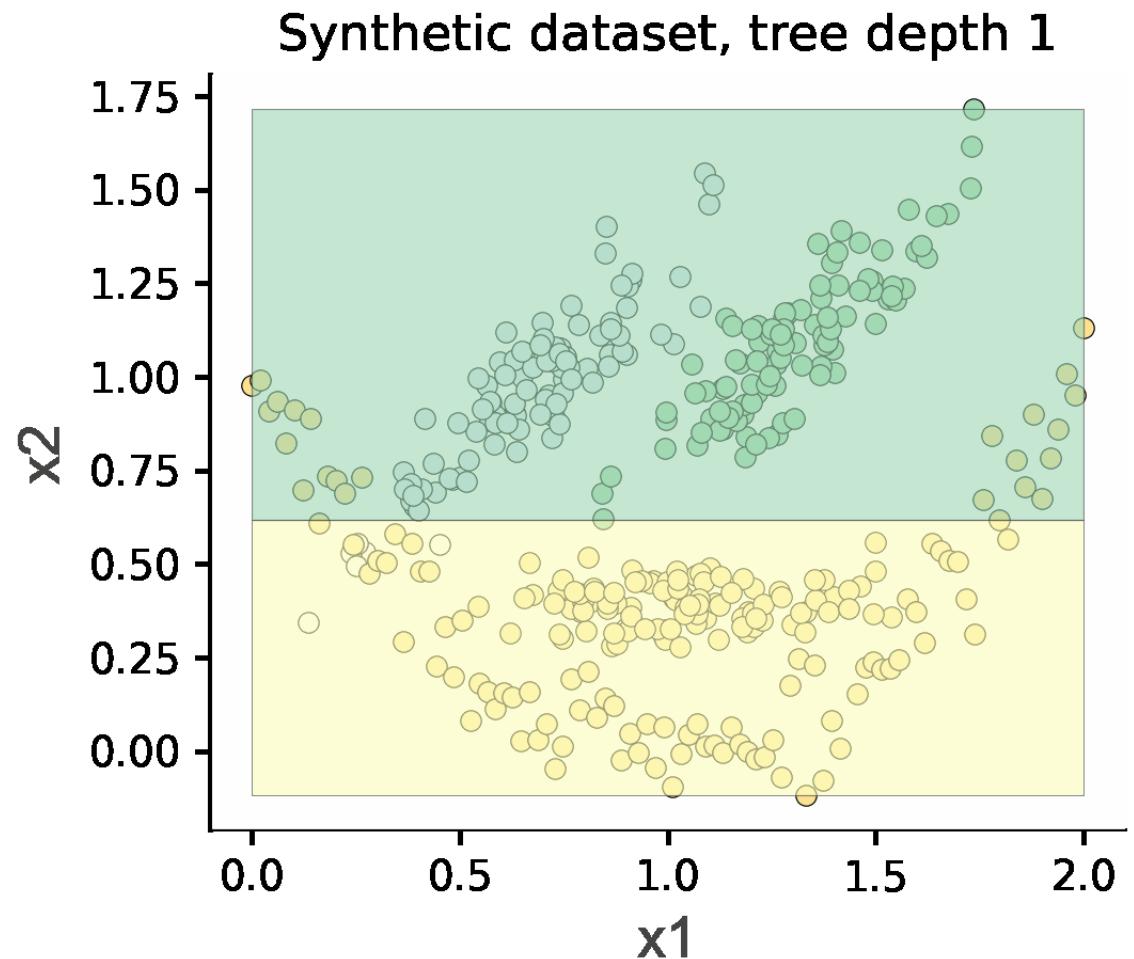


See <https://github.com/parrt/msds621/blob/master/notebooks/trees/gini-impurity.ipynb>

Tree structure's effect on prediction error

Hyperparameter max_depth

Restricts how many splits tree can make, preventing tree from getting too specific to training set (zeroing in on outliers)

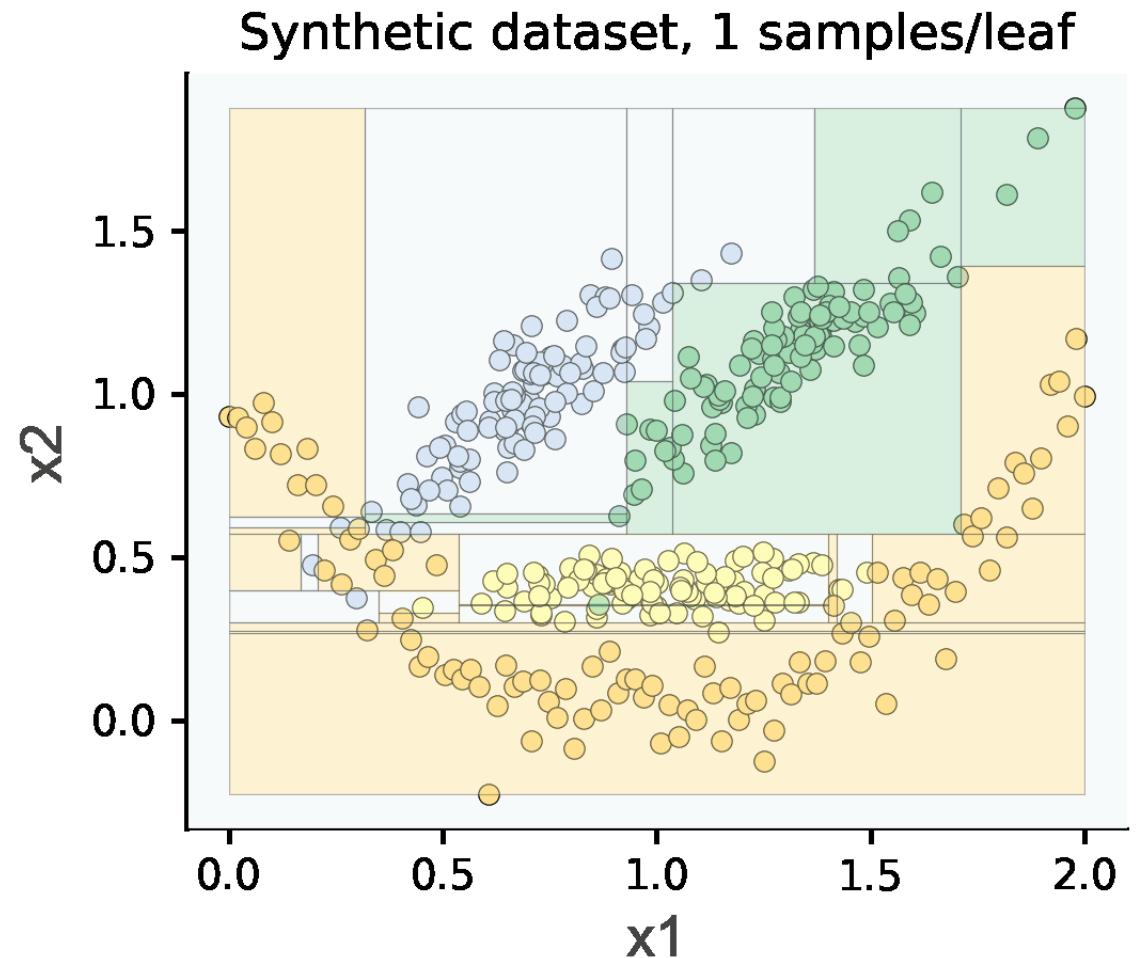


Hyperparameter min_samples_leaf

- Idea: don't split regions w/less than min_samples_leaf records
- Similar to limiting height of tree but finer granularity of control
- More direct control of generality than tree height
- Degenerate case where min_samples_leaf=n (sample size)
 - What does such a regressor predict?
 - What does such a classifier predict?
 - Describe accuracy of this extreme model
 - If we trained on many different training sets pulled from same data distribution, how stable would the test set prediction error be? (What does that say about variance/generality?)

2D tessellation varying min samples/leaf in action

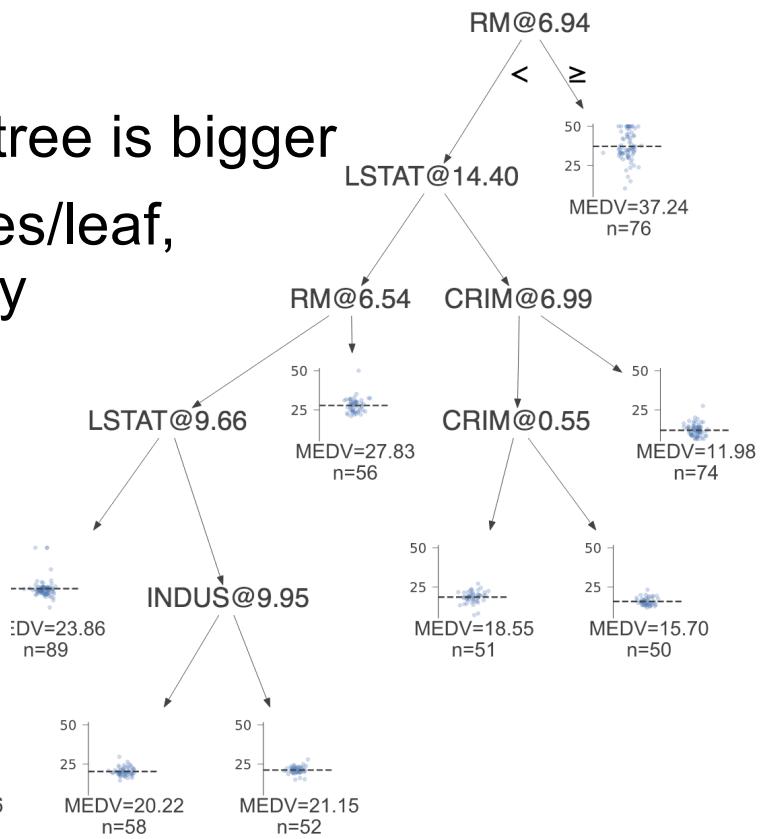
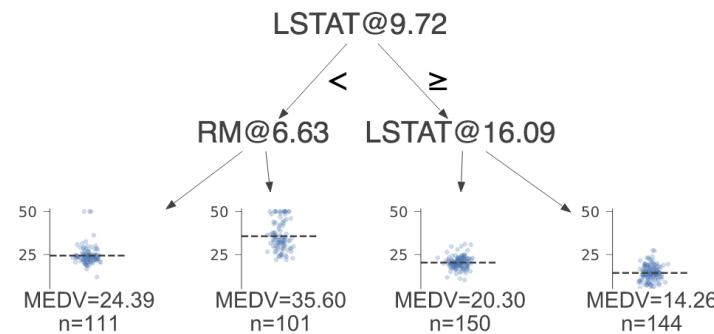
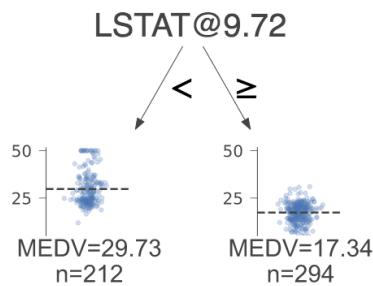
As min leaf size gets bigger,
more general but less
accurate



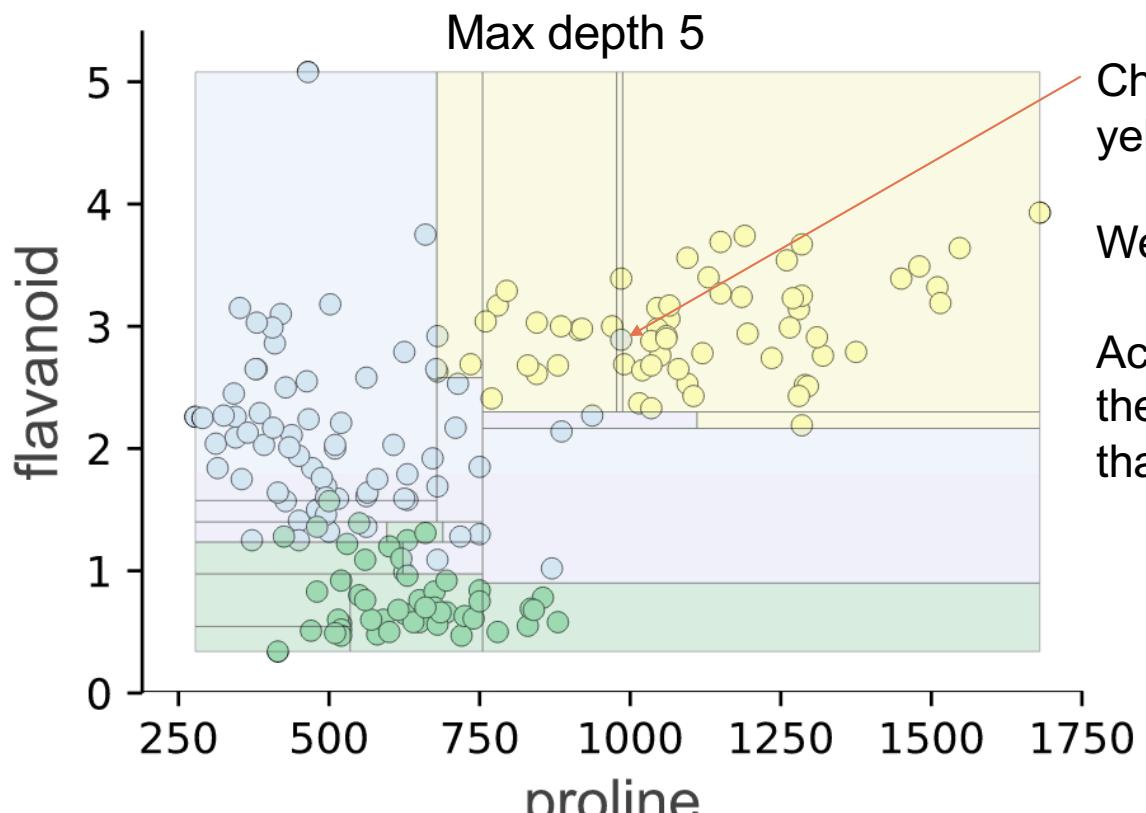
Trees for 200,100,50 samples per leaf

(Boston housing data, n=506 records)

- Leaves have fewer & fewer elements, but tree is bigger
- Variance of y in leaves shrinks with samples/leaf, which is training goal; increase accuracy by reducing impurity



What happens with very small leaves?

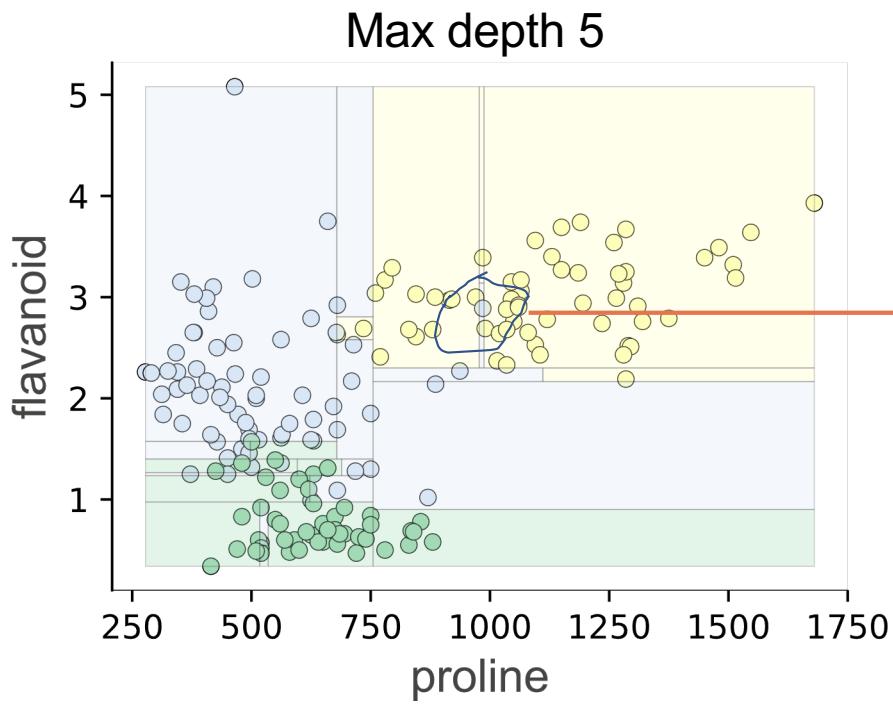


Check out that lonely blue dot in sea of yellow! (let's assume tiny region is blue)

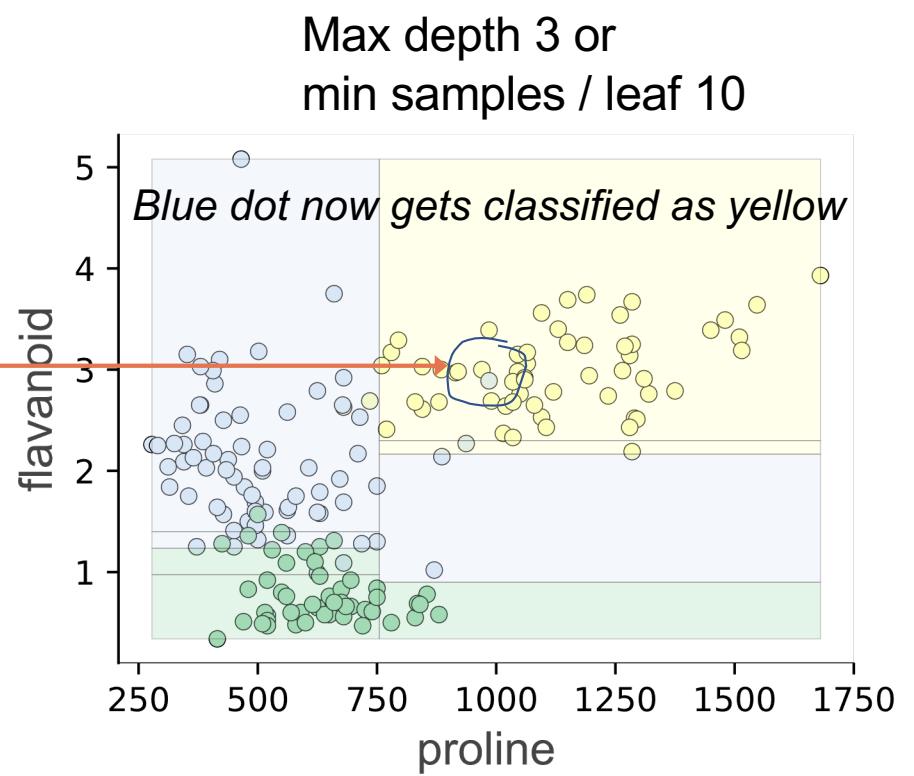
We let model get overly specific; it's overfit

Accuracy on training set is very high, but at the cost of generality; test set error is higher than necessary

How could we (likely) improve generality?



(Wine data set)



Key takeaways

- Trees partition feature space into tight rectangular hypervolumes with pure/similar y values for records in that hypervolume
- Decision trees have internal decision nodes that test variables at split points and leaf nodes that make predictions
- Leaves predict mean (regressor) or mode (classifier) of samples
- Partitioning subject to reducing MSE (y variance) or Gini impurity
- Limiting tree height or increasing leaf size reduces accuracy but improves generality
- (We'll have whole lecture on training these beasts)

Lab time

- Partitioning feature space

<https://github.com/parrt/msds621/blob/master/labs/trees/partitioning-feature-space.ipynb>