The visual interpretation of decision trees

(How to lead a fulfilling life by being dissatisfied)

Terence Parr
University of San Francisco

Work done with Prince Grover
manifold.ai

UNIVERSITY OF SAN FRANCISCO

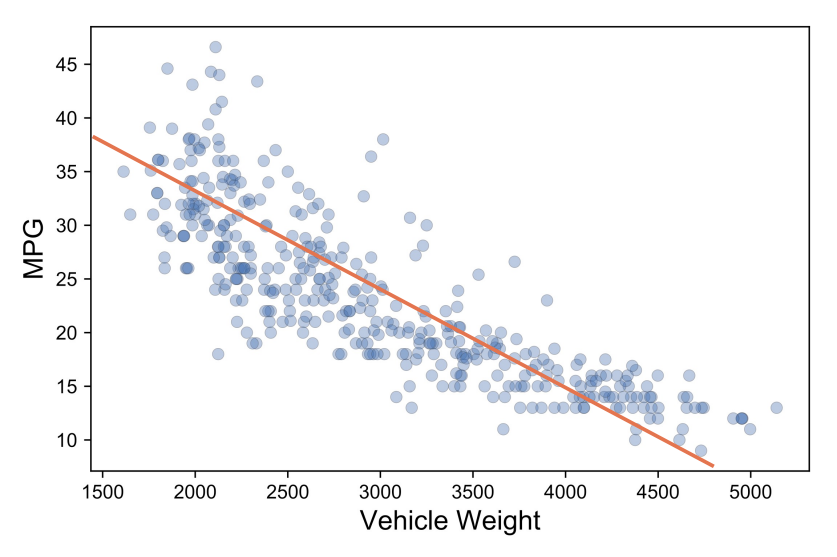
Topics

- `dtreeviz` 0.3: Visualizing training of regression, classification trees
- `dtreeviz` 0.2: Visualizing key elements of decision trees
- Implementation notes
- How to lead a fulfilling life by being dissatisfied

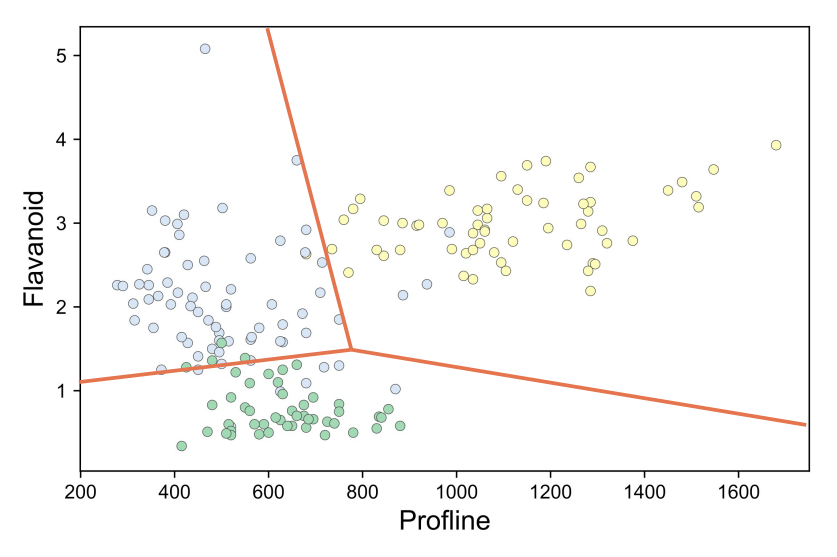
Training Decision trees

Regression versus classification

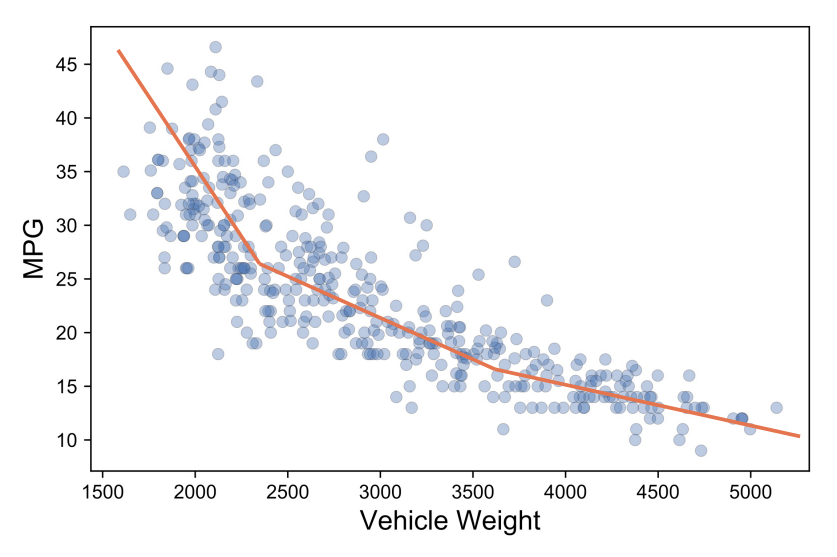
Regressors draw through data

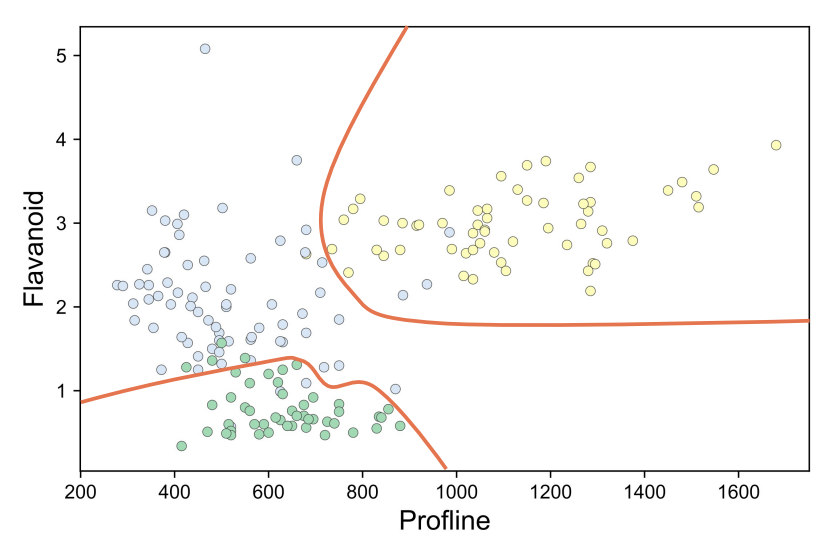


Classifiers draw between data clusters

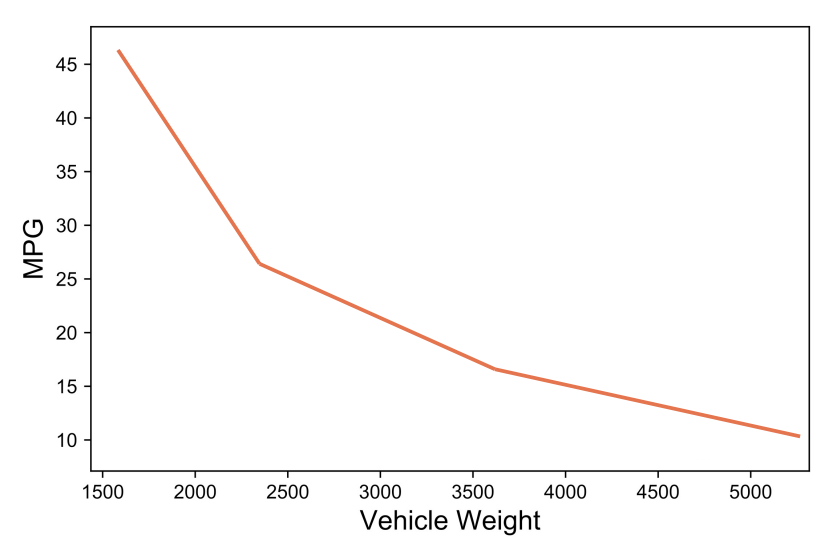


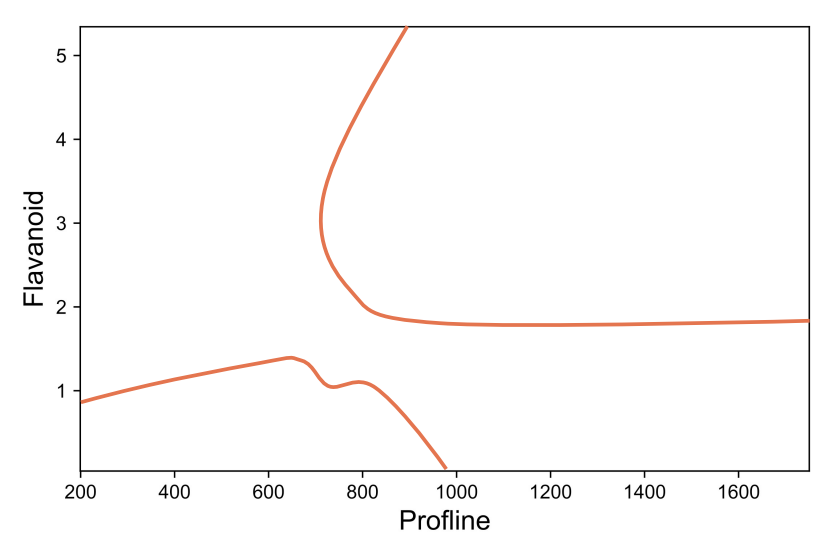
Different models have different surfaces





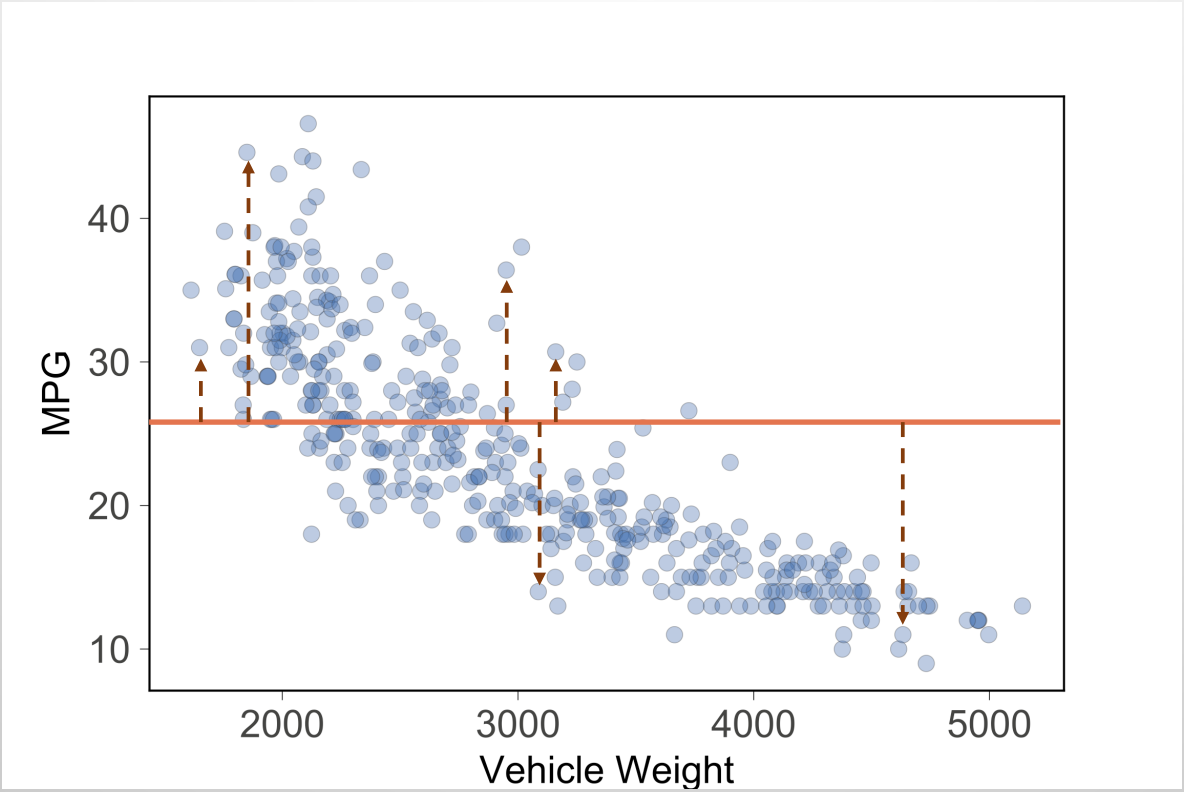
Model is just the decision surface(s)





Let's invent a simple regressor

Predict MPG from weight



Predict a single constant for entire region (the mean)

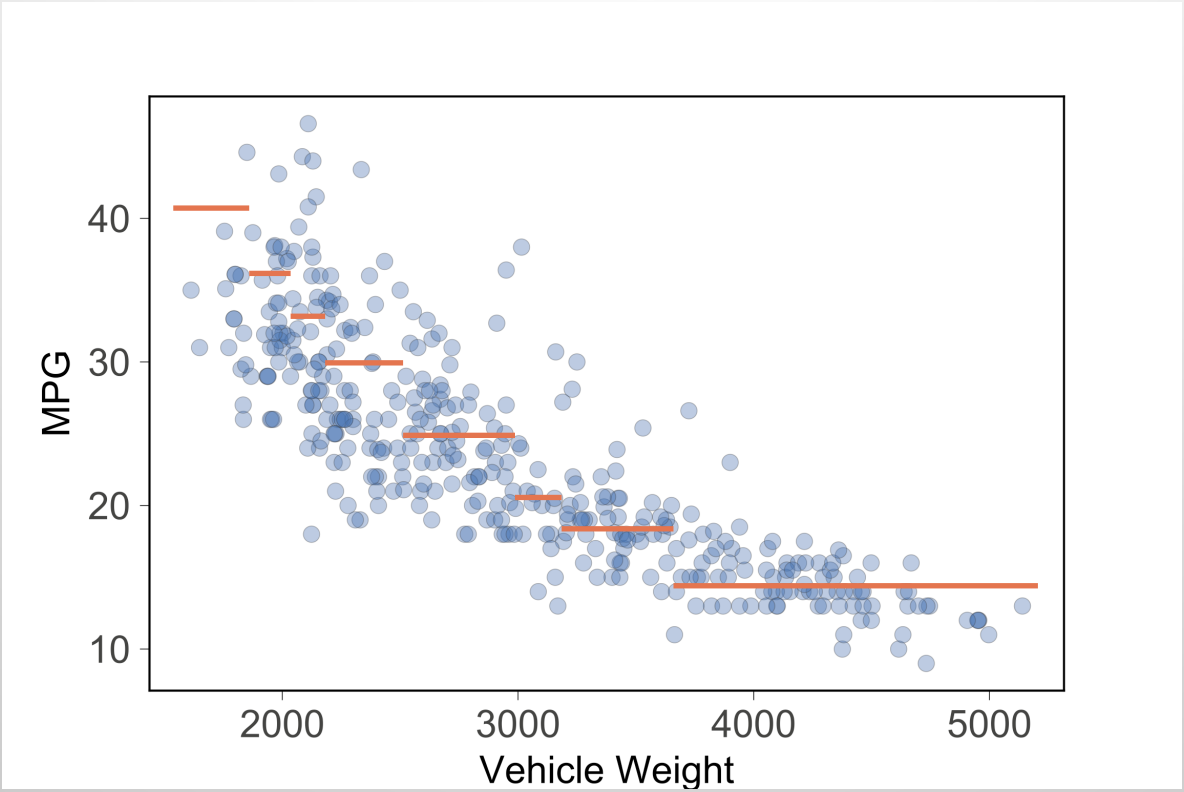
MSE is high

Residuals from point to region mean

Improve by partitioning, using multiple lines

WHERE DO WE SPLIT??

Find groups of similar MPG values, which yields a lower MSE

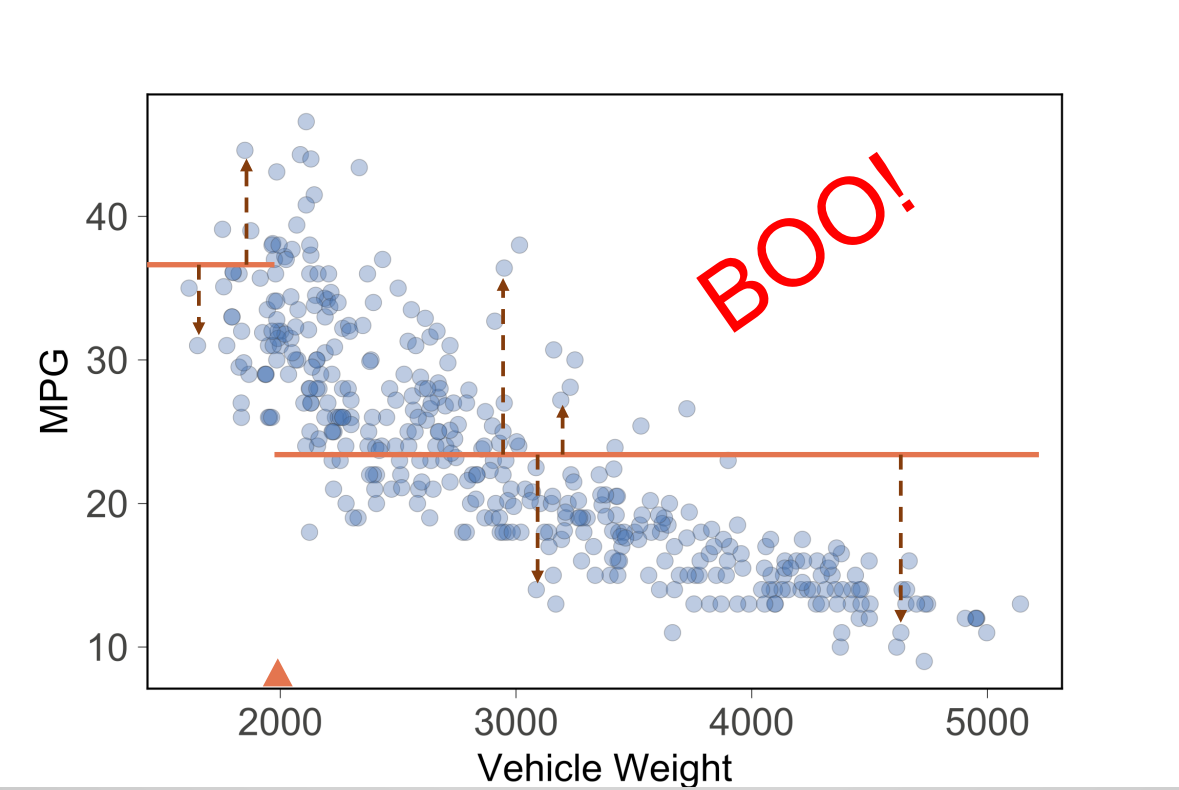


Can only use horizontal lines, but can use lots

Each region predicts mean in piecewise fashion

Strategy: find split point giving least MSE

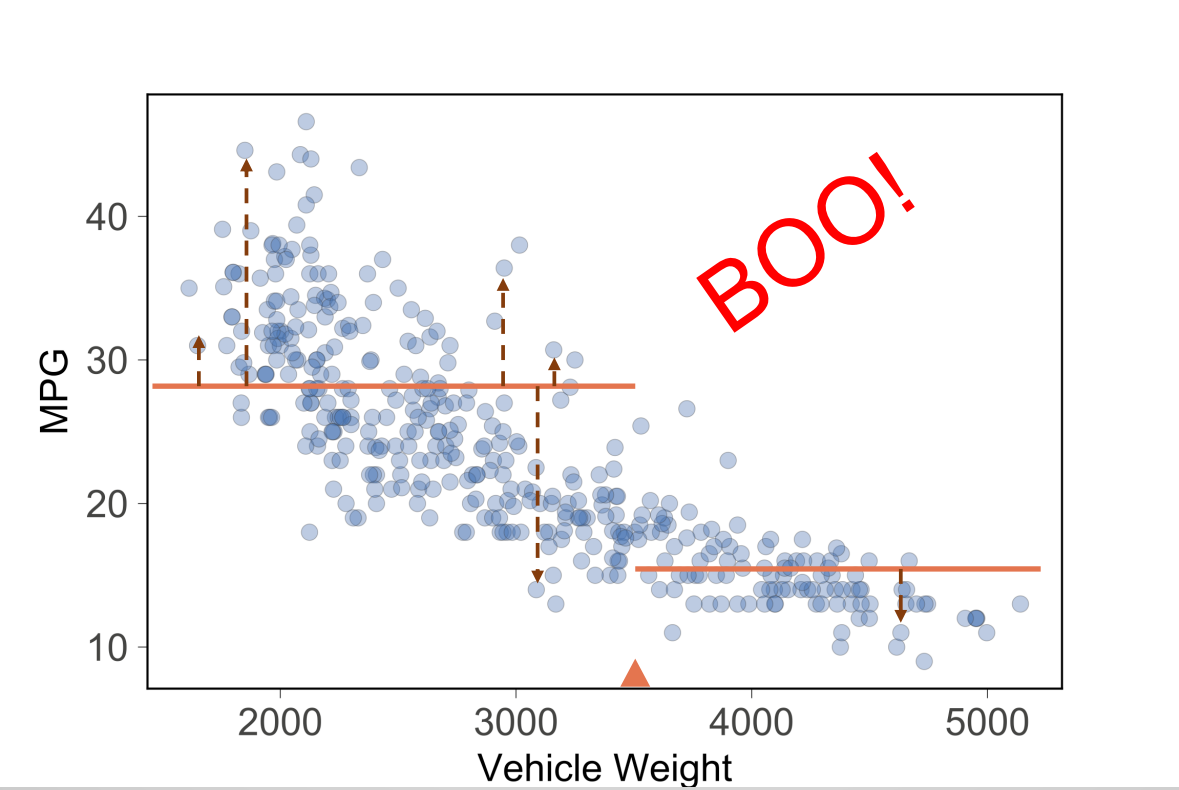
Split region into two subregions, each predicting mean



X=2000 is BAD CHOICE: MSE very high

Strategy: find split point giving least MSE

Split region into two subregions, each predicting mean



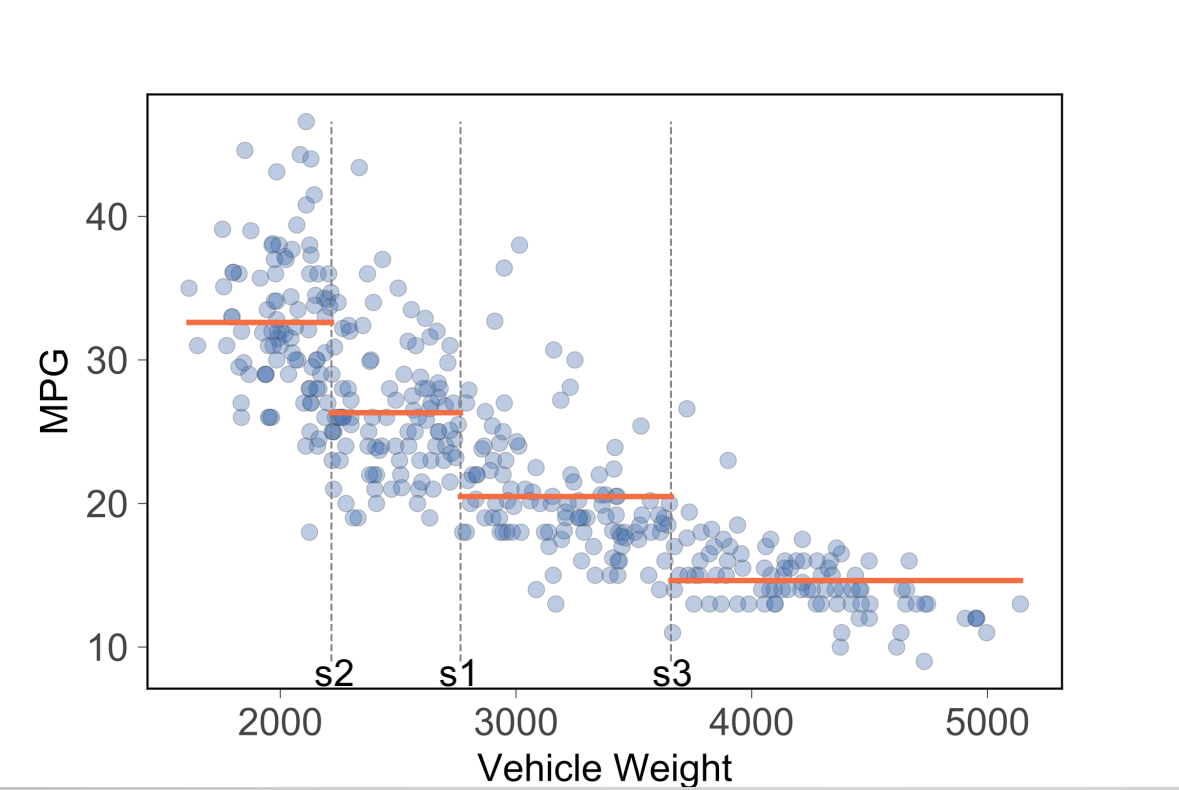
X=3500 is ANOTHER BAD CHOICE: MSE very high

A split exists that gives min MSE for regions

Technique: Exhaustively check all feature values, computing MSE for each split

Track split point giving min MSE

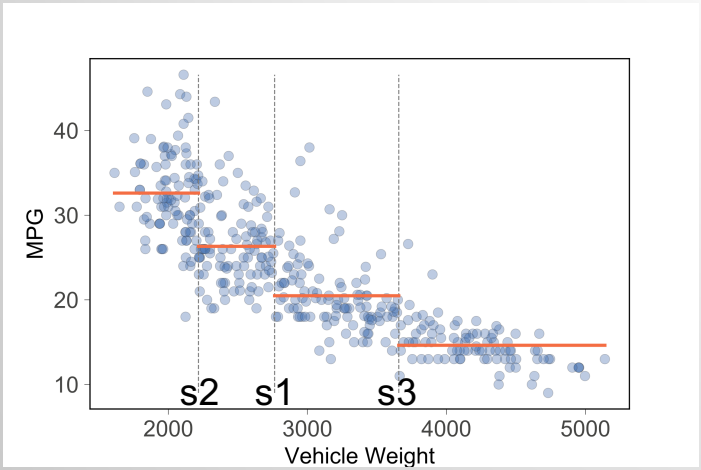
Now split those 2 regions into 4 regions

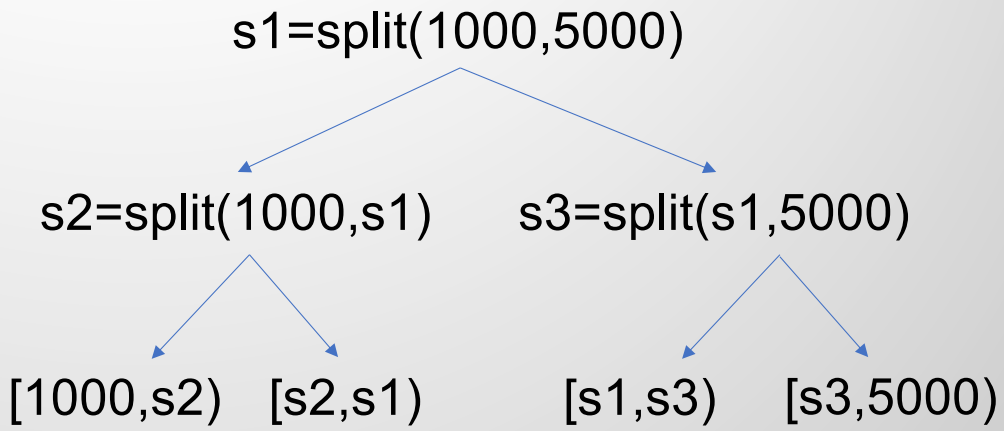


Split s1 stays, *recursively* split left/right regions to get splits s2, s3

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

Model training recursion tree gives regions defined by splits s1,s2,s3



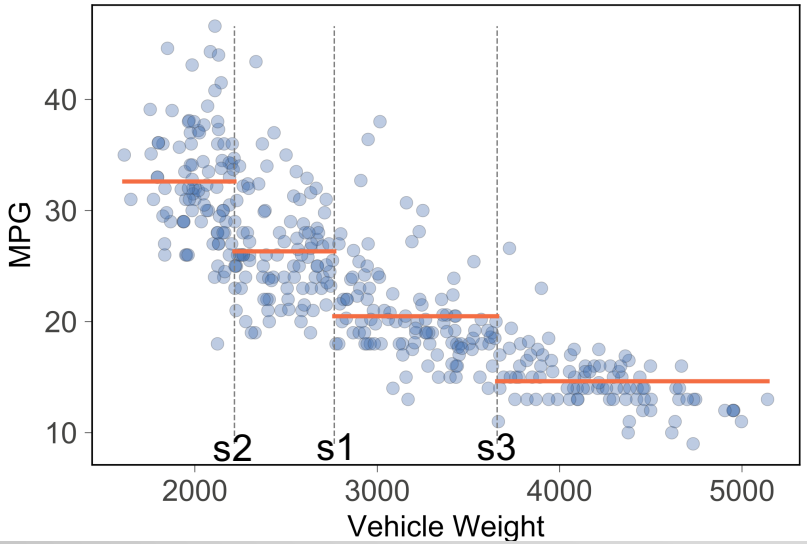


Split (recurse) until:

- All potential splits do not reduce MSE
- Acceptable MSE achieved
- Max number of splits reached
- Etc…

Predictions are avg of MPG (target) values in regions

Hardcoded model implementation

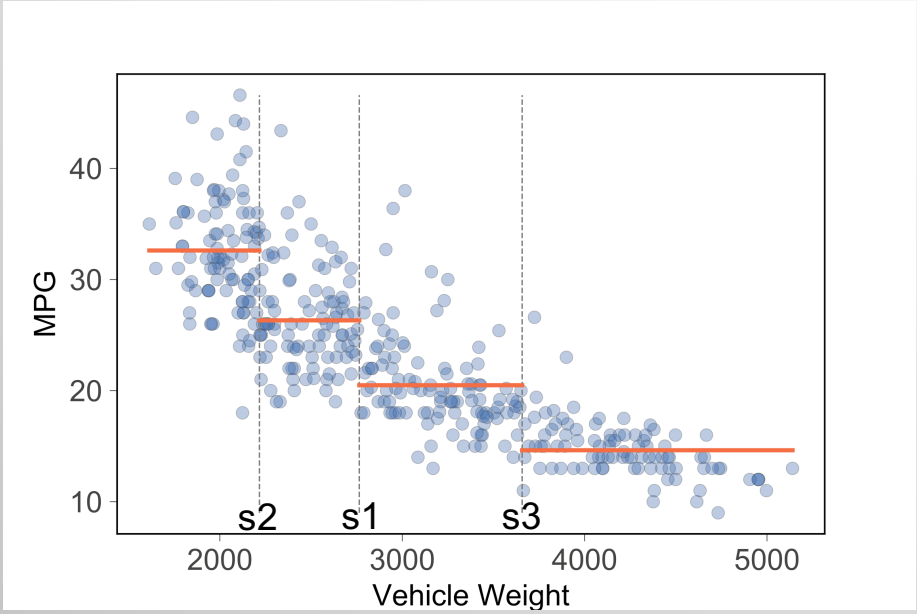


To partition space, test in recursion/split order

```
if x<s1 and x<s2: predict 32.6
if x<s1 and x>=s2: predict 26.3
if x>=s1 and x<s3: predict 20.5
if x>=s1 and x>=s3: predict 14.6
```

Note repeated comparisons!

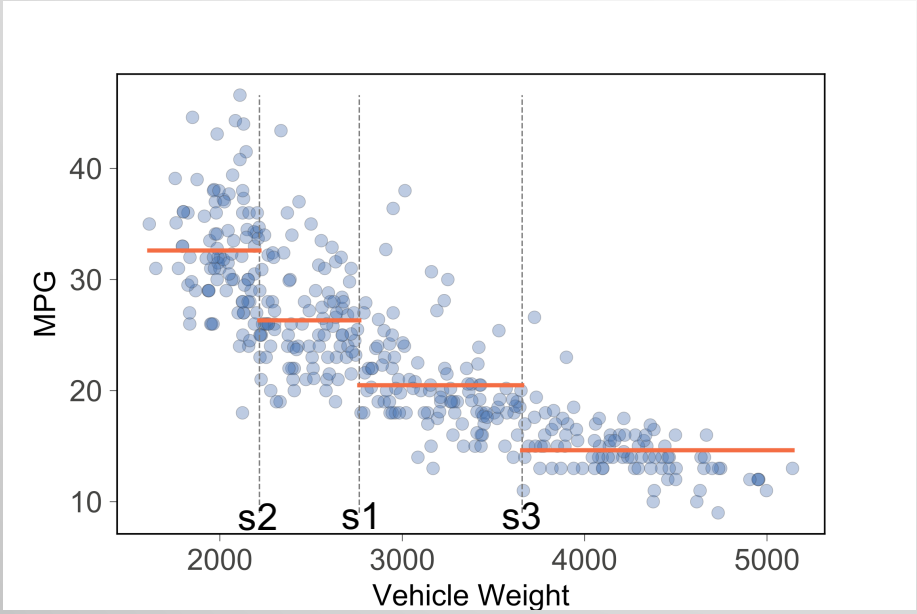
Factor the split comparisons for efficiency

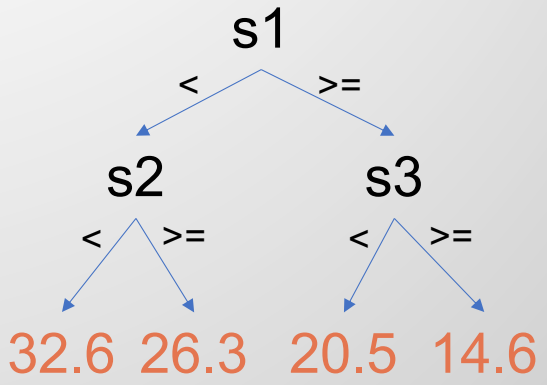



```
if x<s1:
  if x<s2: predict 32.6
  else: predict 26.3
else:
  if x<s3: predict 20.5
  else: predict 14.6
```

But, don't want to hardcode model!

Represent nested conditionals as regression decision tree



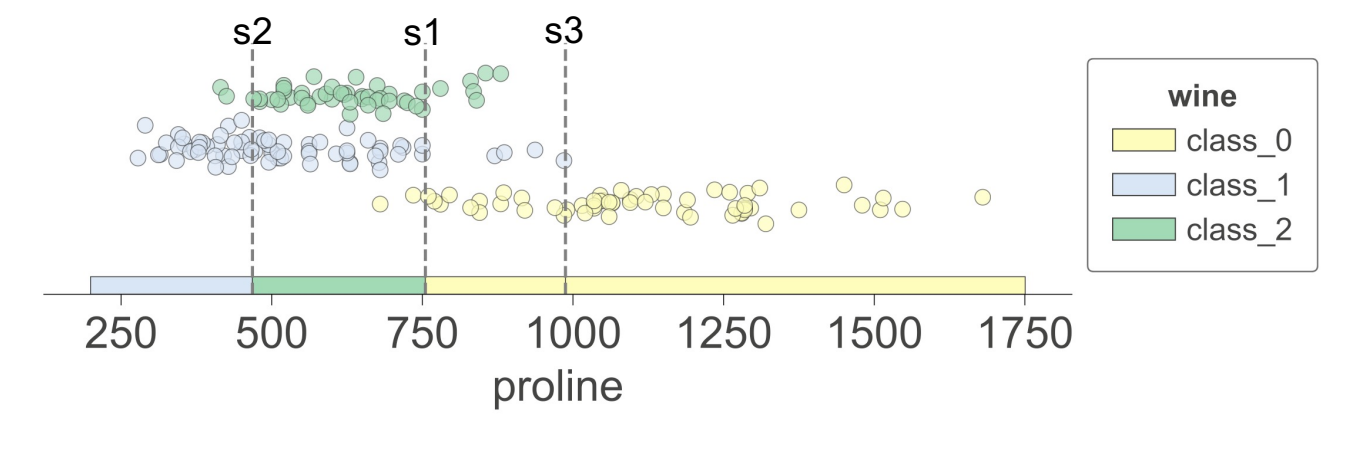


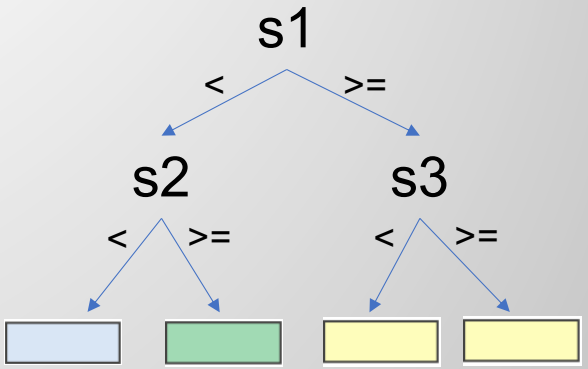
Internal nodes test features
Leaves predict region mean

Morph tree of recursion from training into decision tree!

Classifiers split feature space too

Predict wine from proline

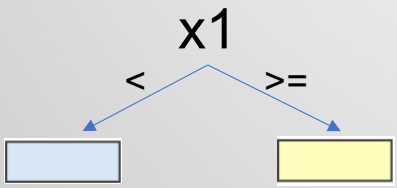


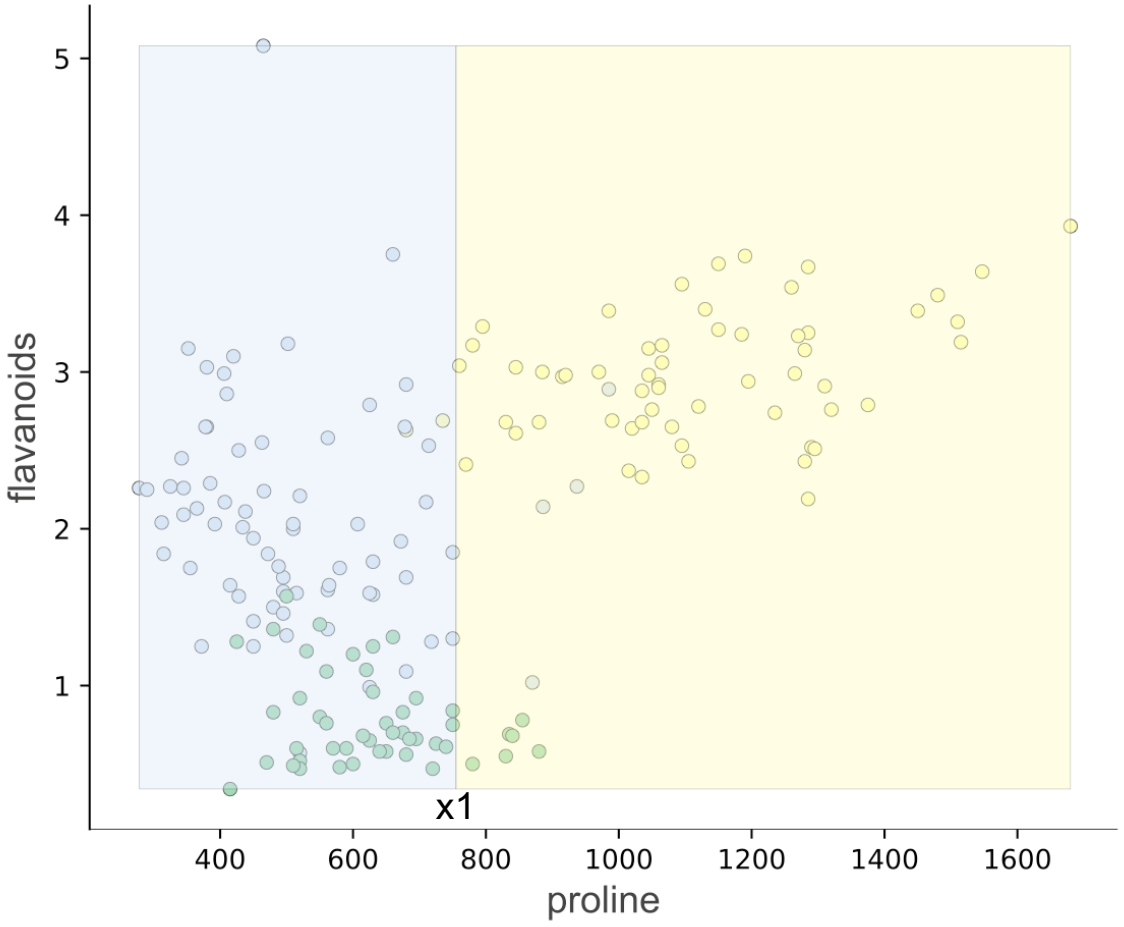


Internal decision nodes test features just like regressor trees

Leaves predict most common target category (mode) not mean

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

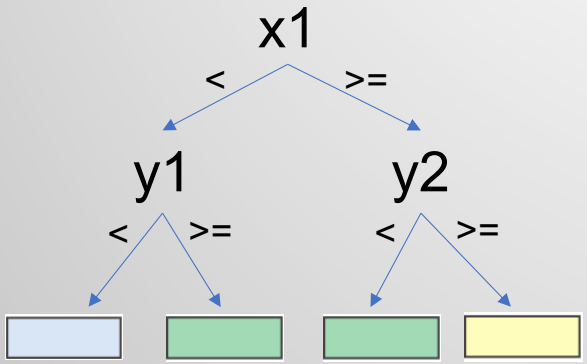




Training looks for (feature, split point) combos giving more pure subregions.

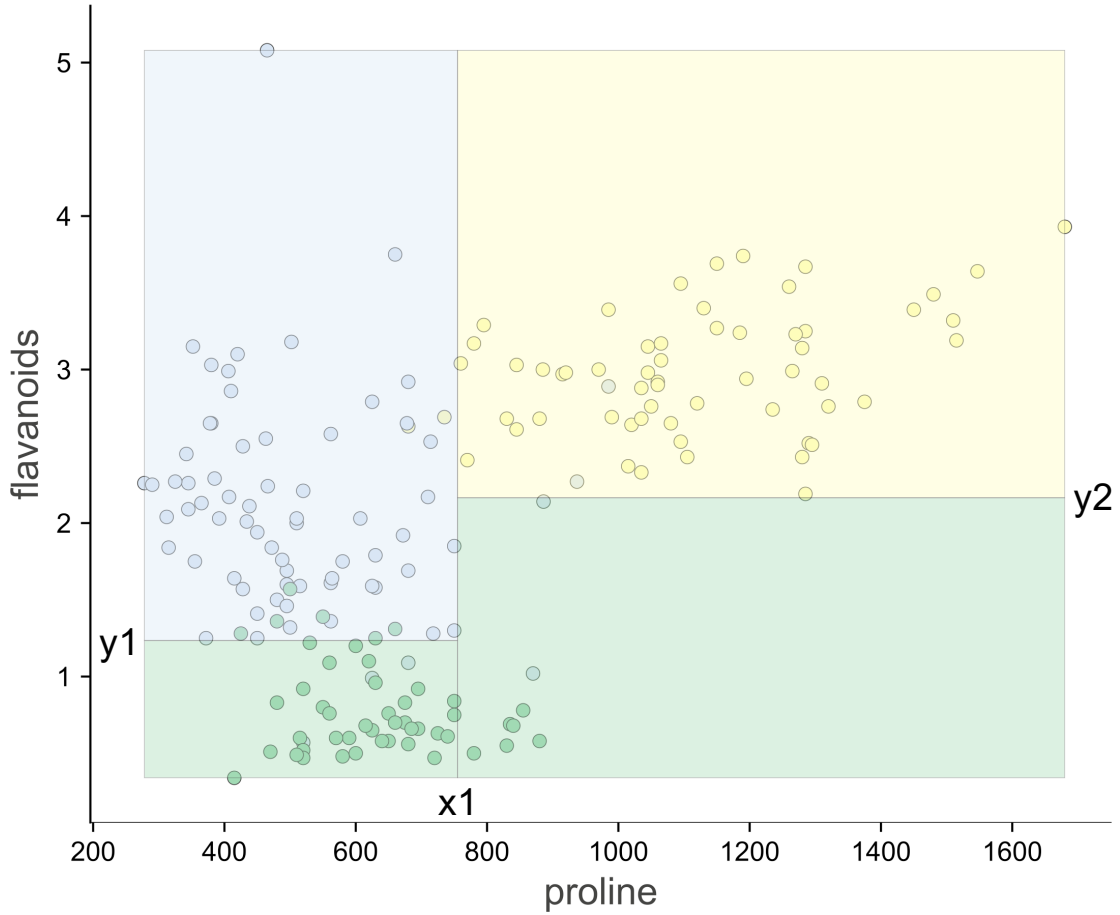
Decision nodes compare single feature value to split point

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

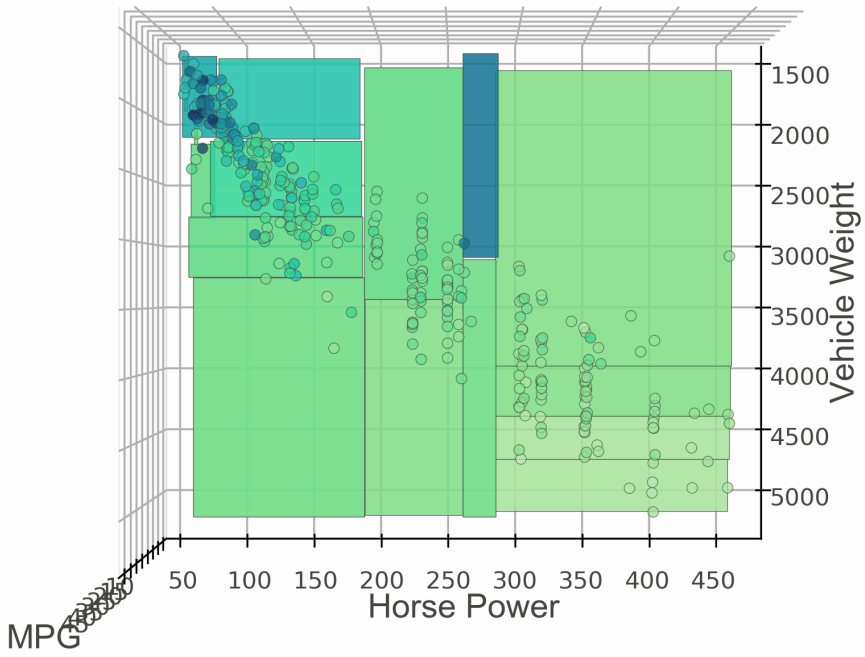


Training looks for (feature, split point) combos giving more pure subregions.

Decision nodes compare single feature value to split point

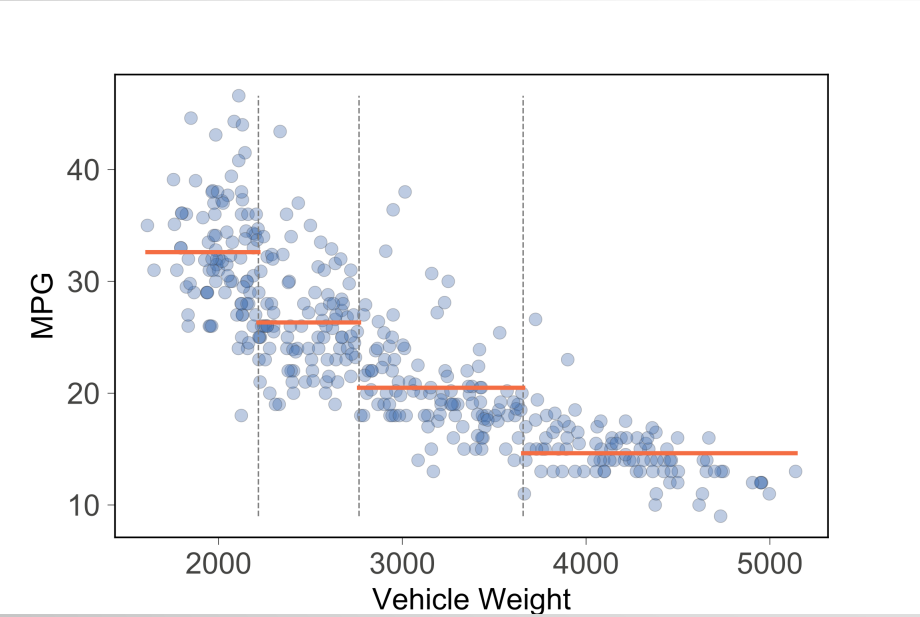


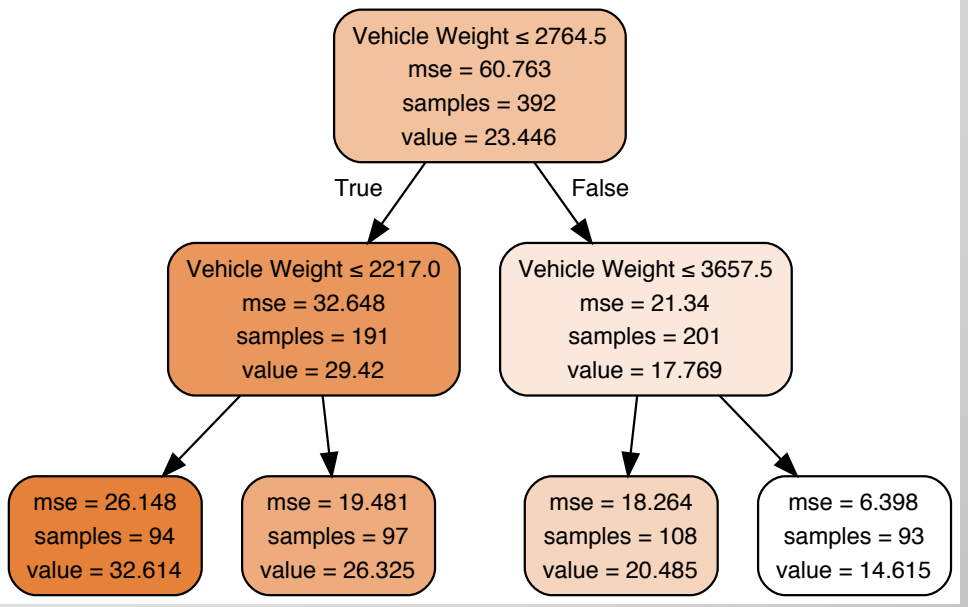
We can also split regressor 2D feature space into regions to improve accuracy



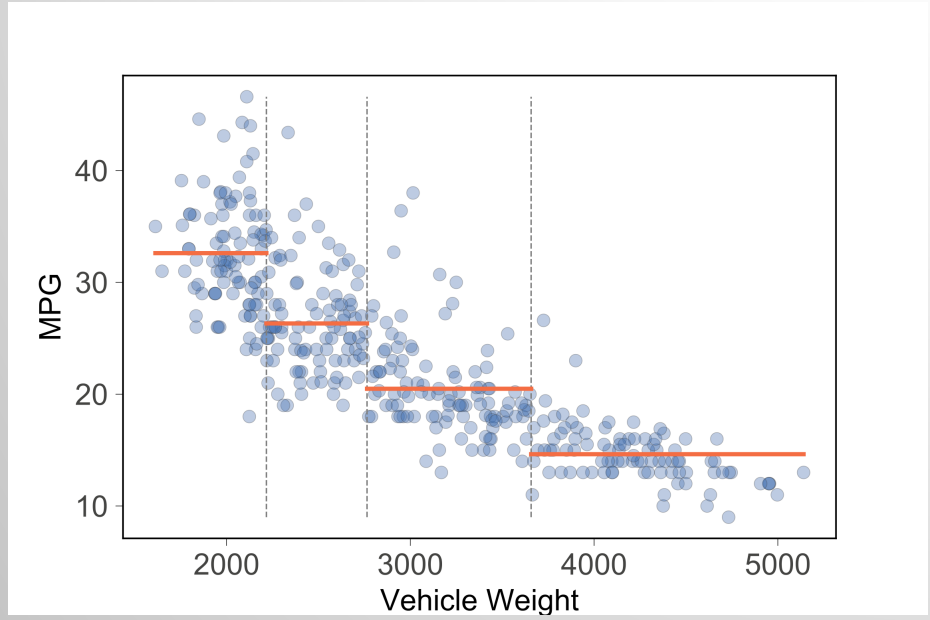
Visualizing the key elements

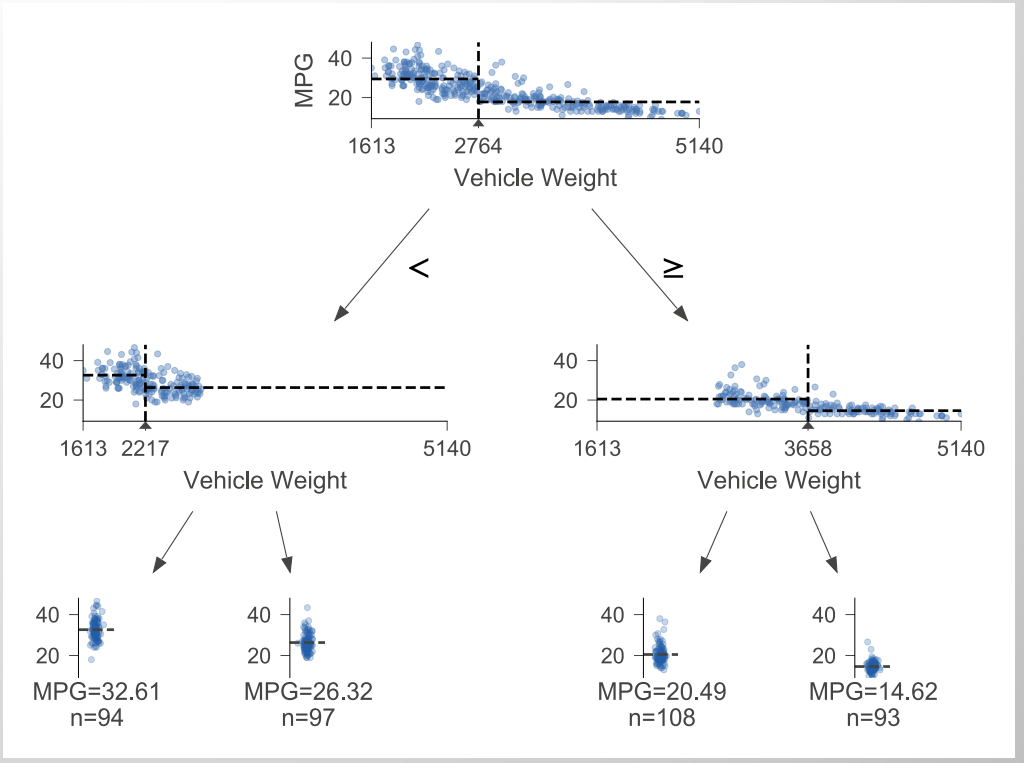
1D feature space vs sklearn regressor tree



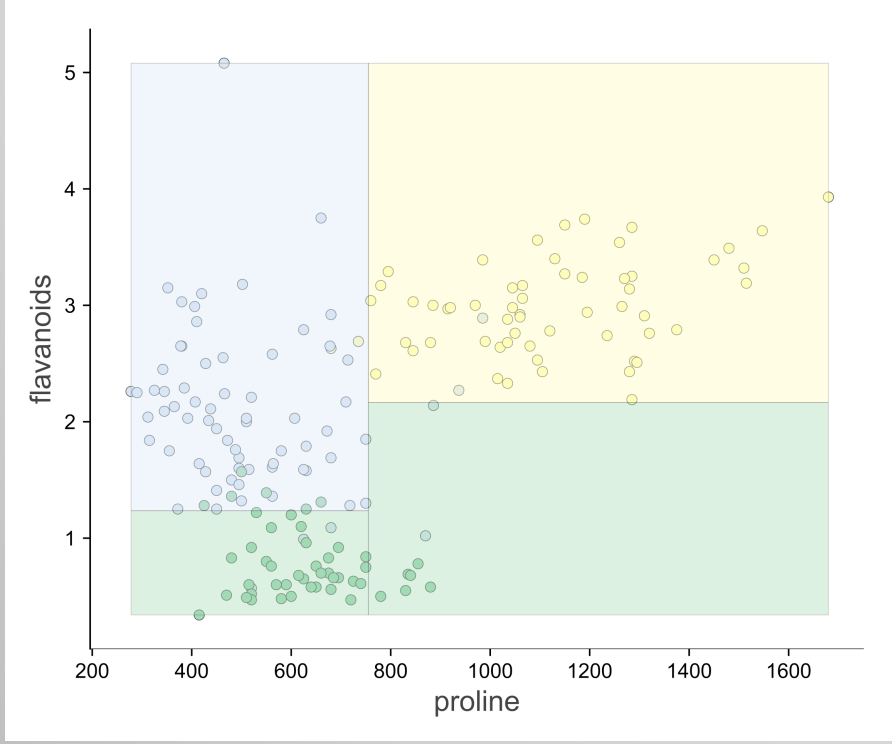


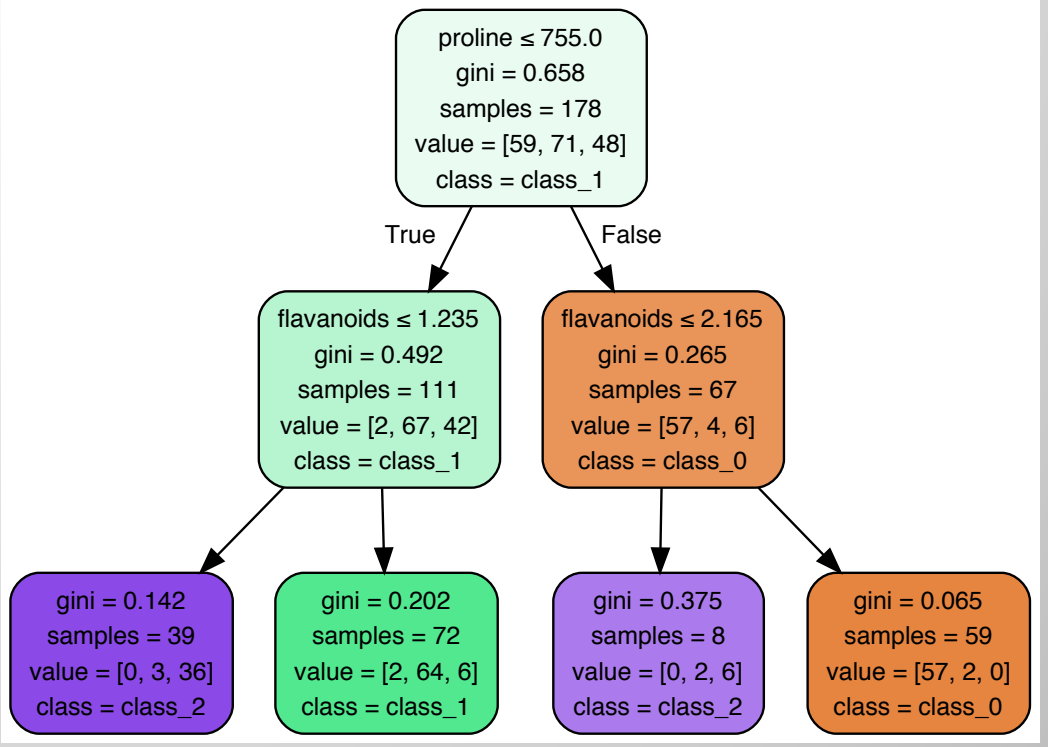
1D feature space vs `dtreeviz` decision tree



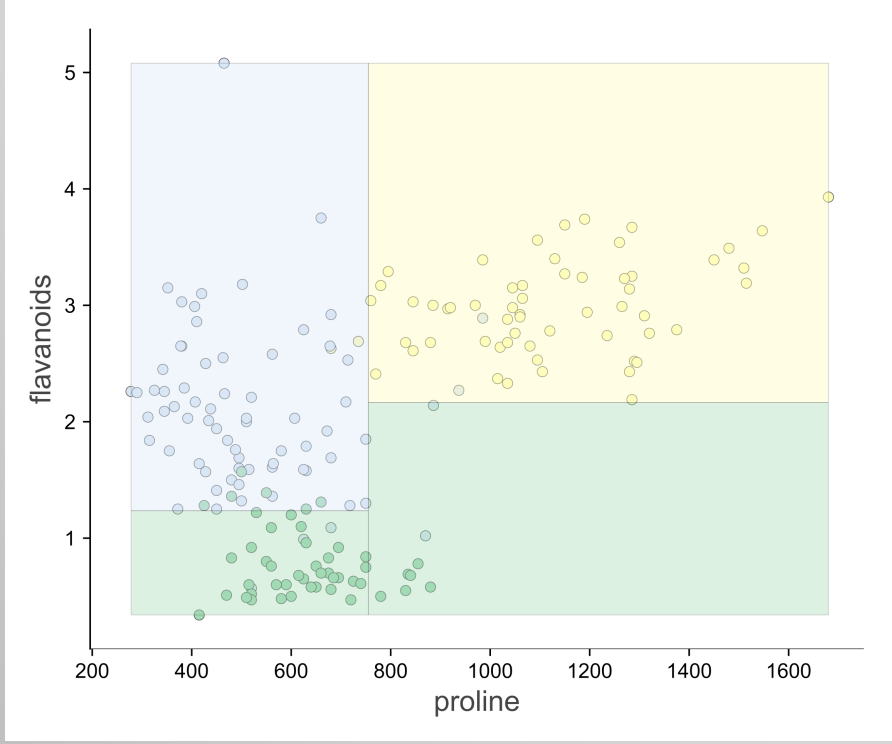


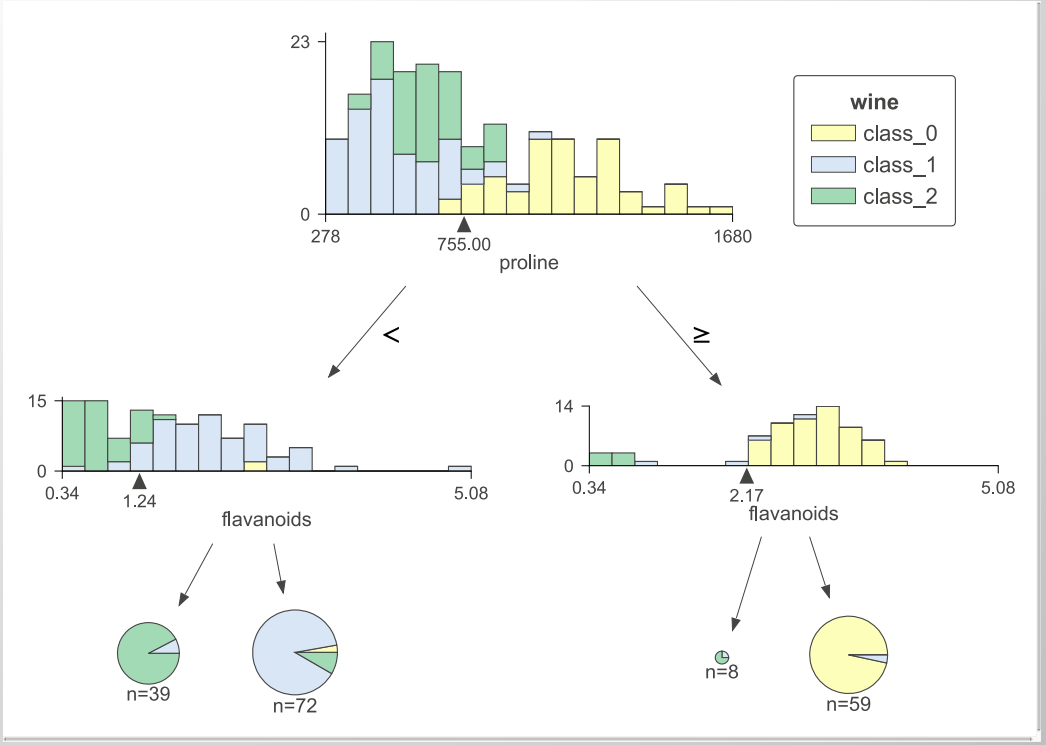
2D feature space vs sklearn classifier tree



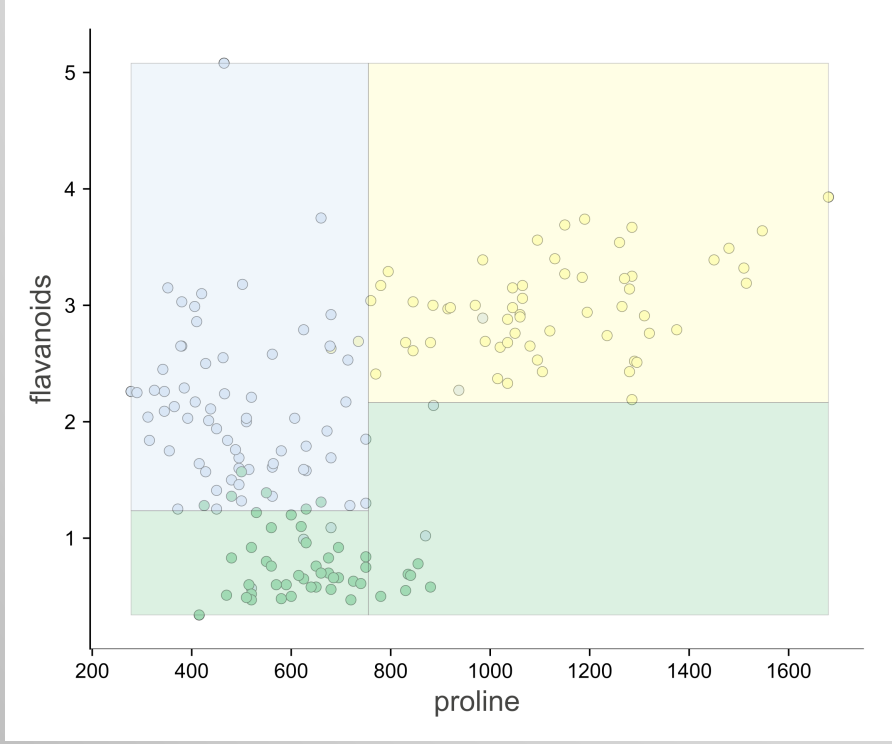


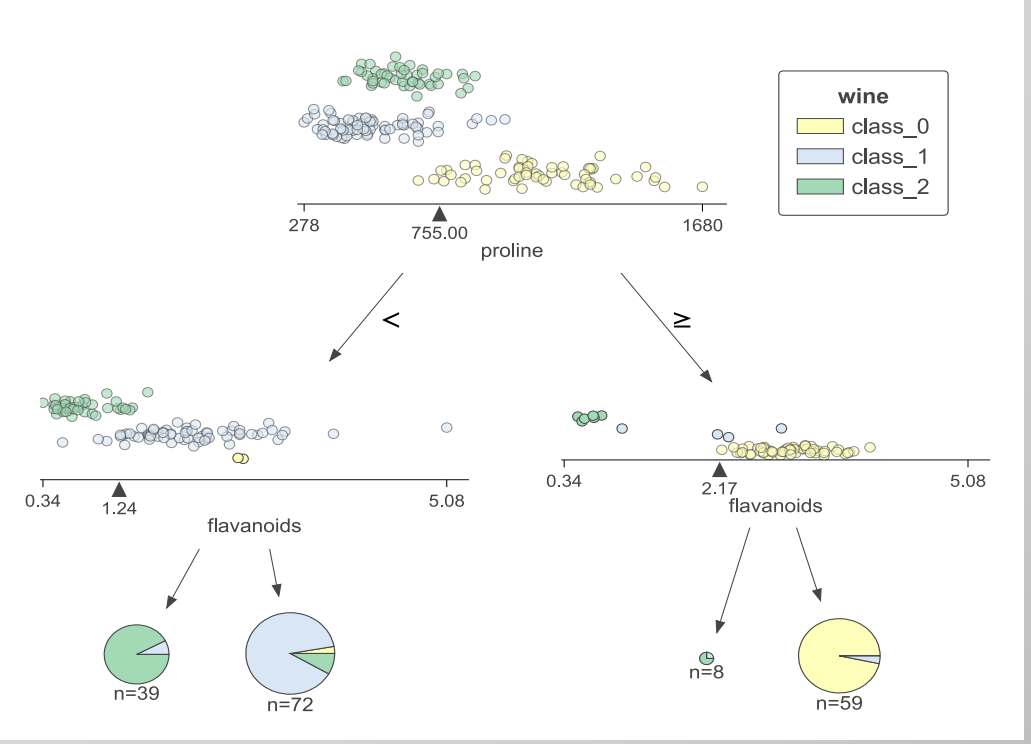
2D feature space vs `dtreeviz` decision tree



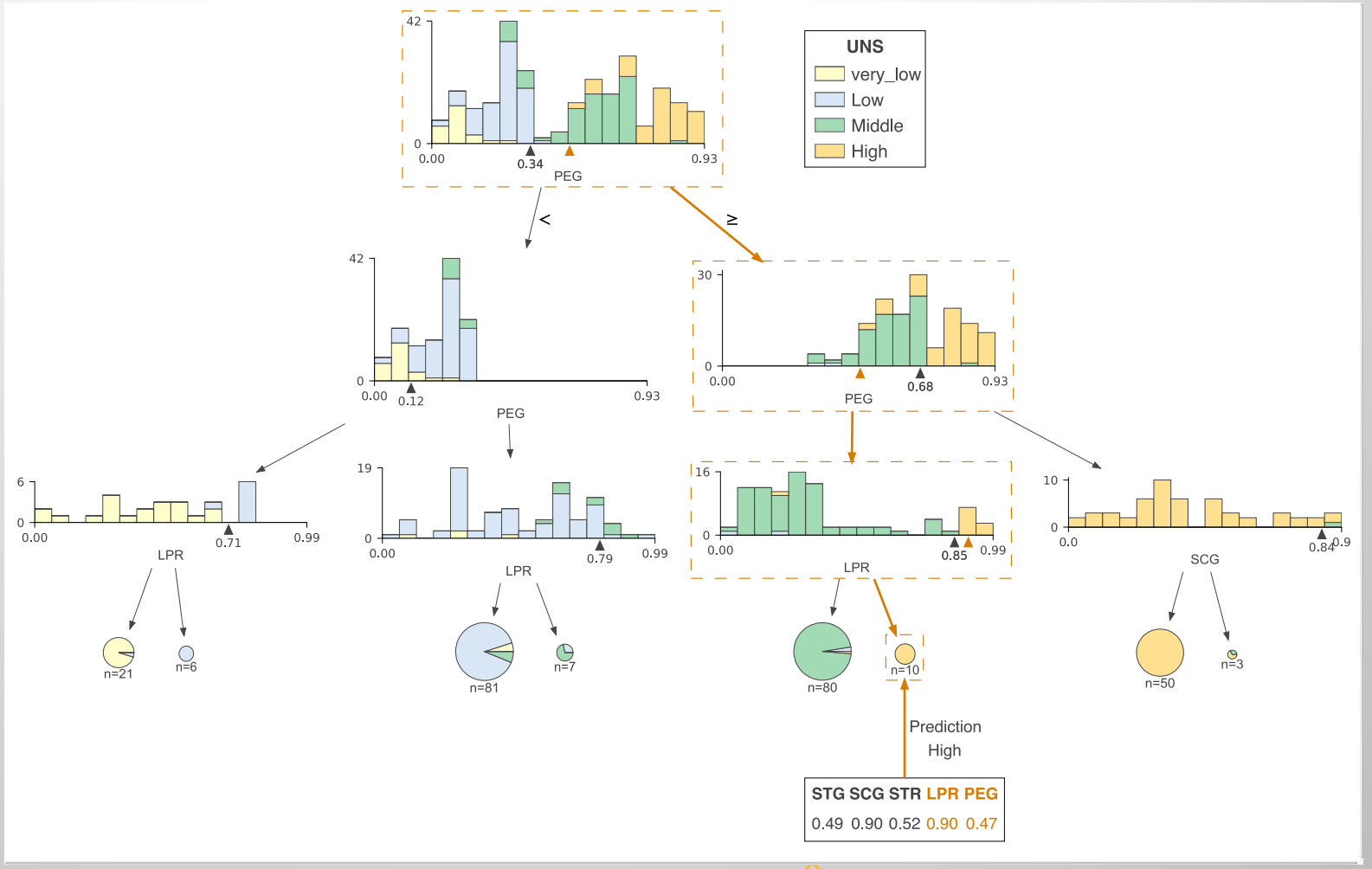


Option: `dtreeviz` decision tree (strip plot nodes)

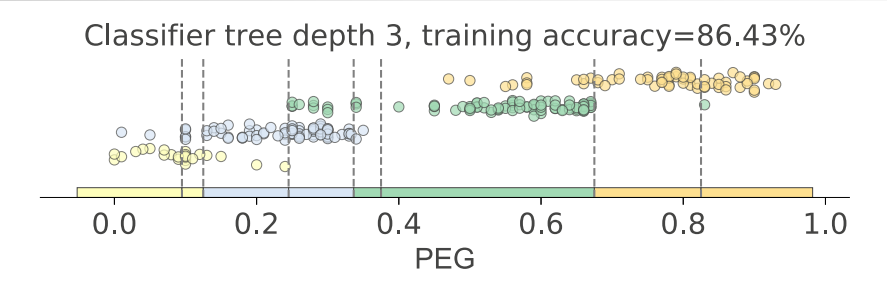


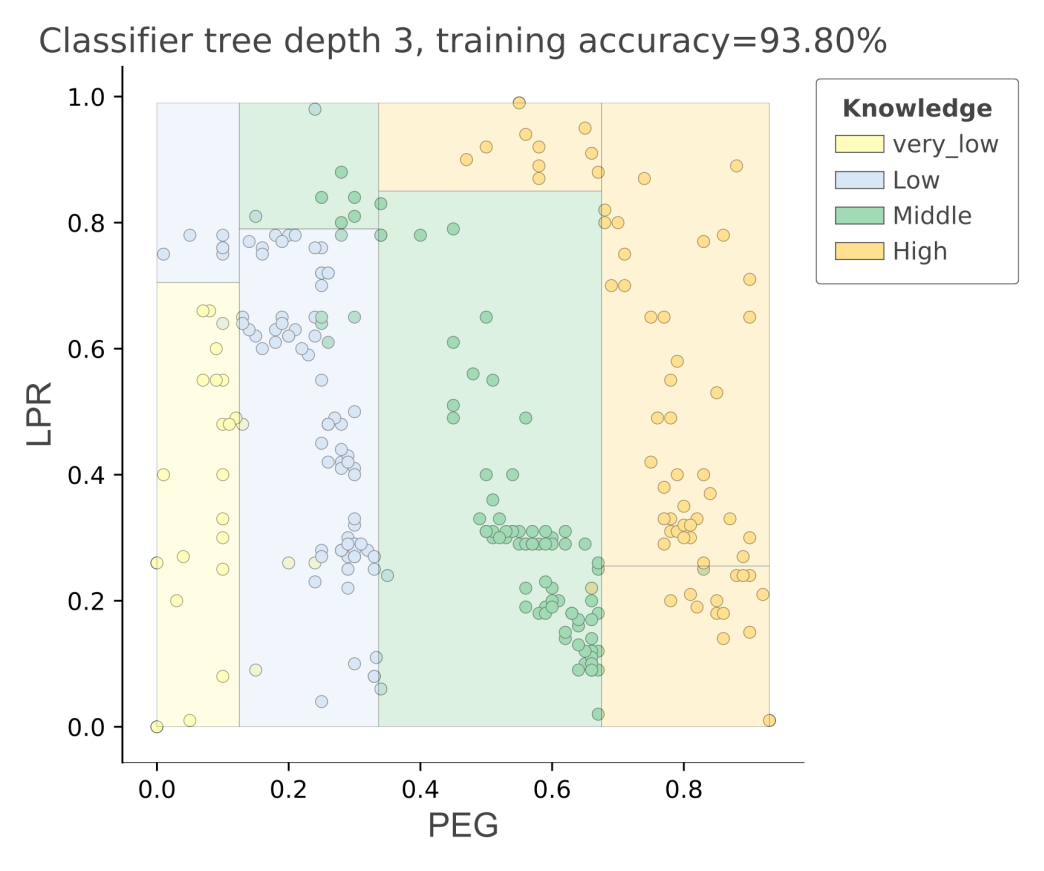


Test vector interpretation

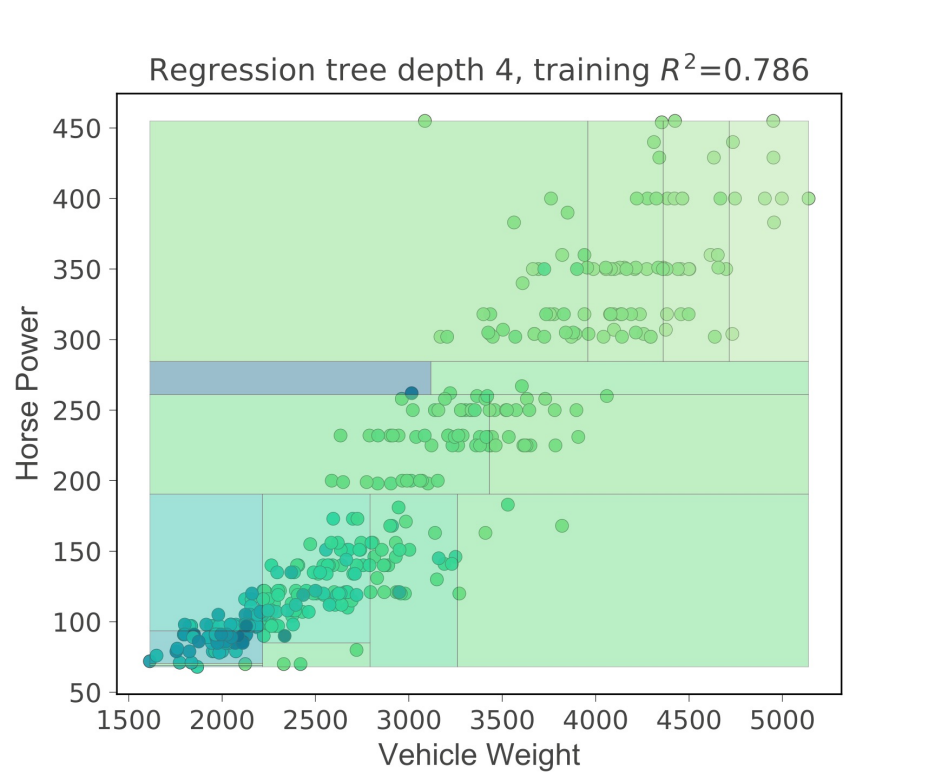


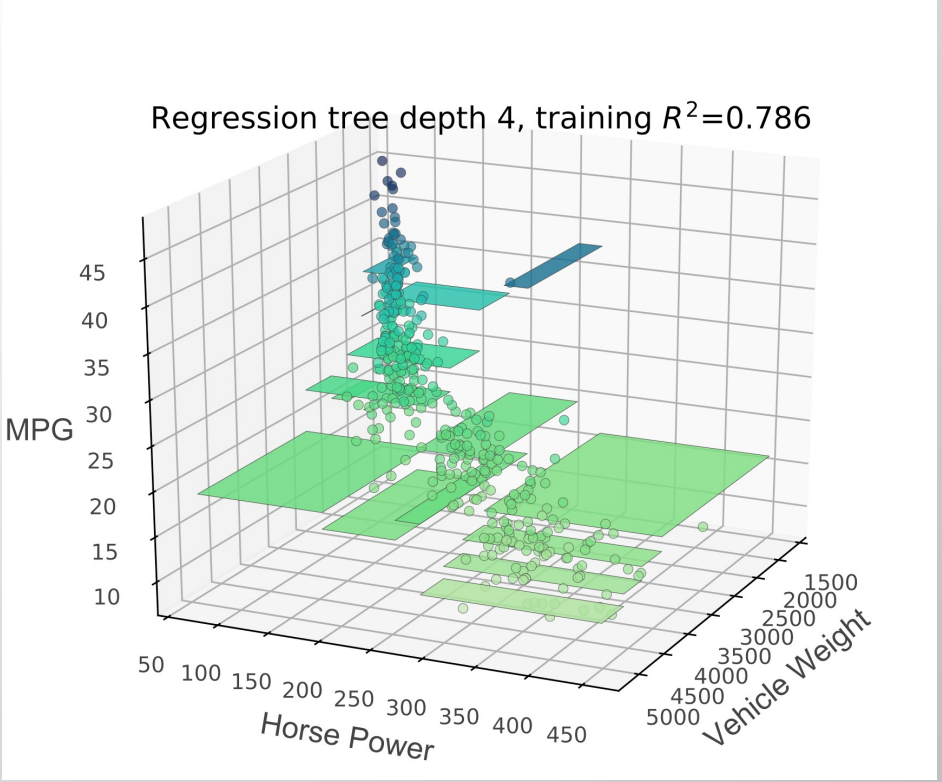
1D, 2D classifier feature space

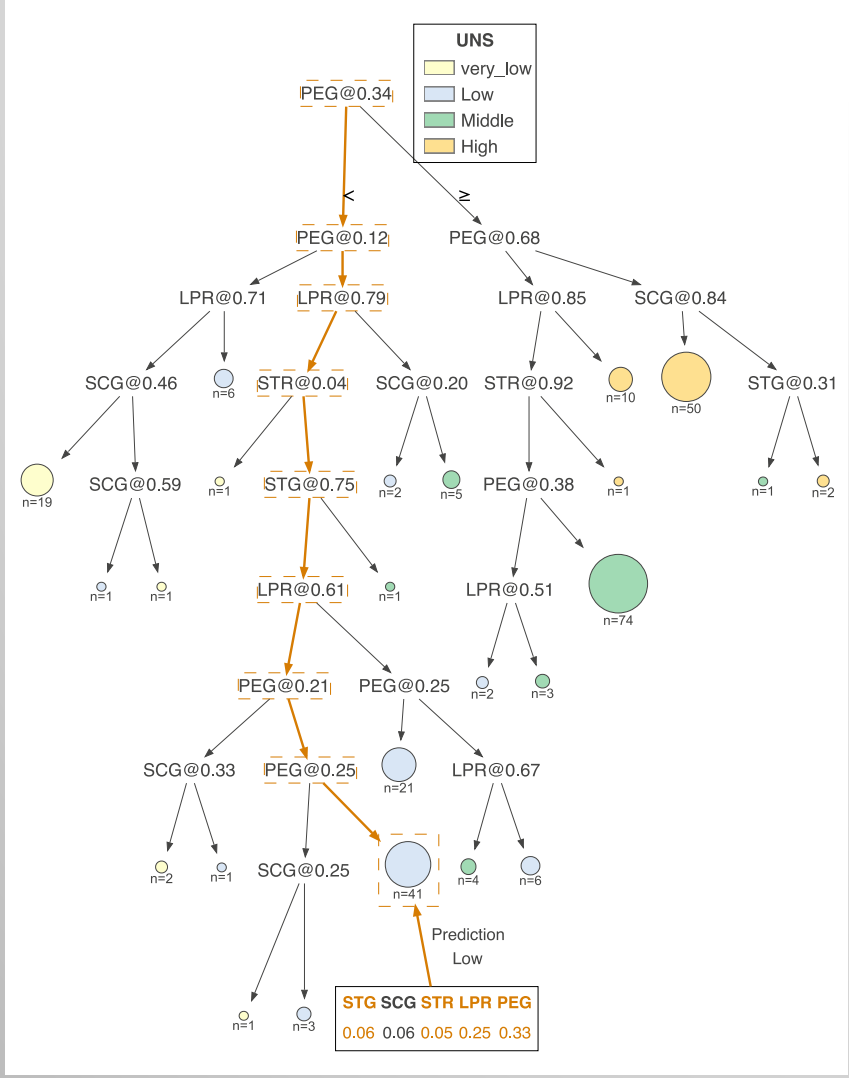




2D regressor feature space (heatmap, 3D)





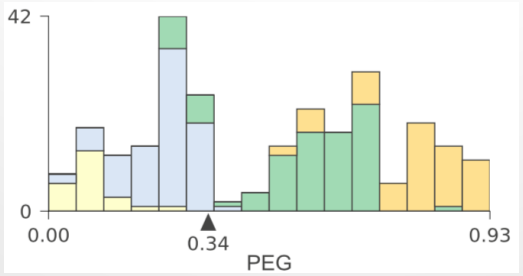


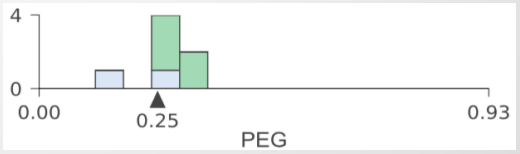
Plain layout for large trees

Details matter

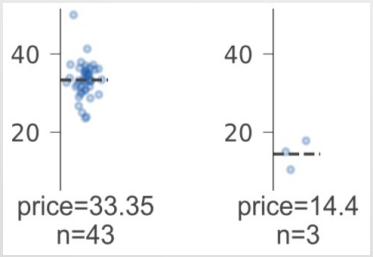
- Handpicked colors
- Gray not black text/lines
- Hairlines
- Outlines of bars in barchart

Histos shrink with sample size





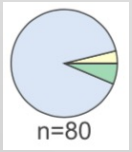
Target axes show same range



STG	SCG	STR	LPR	PEG
0.39	0.42	0.83	0.65	0.19

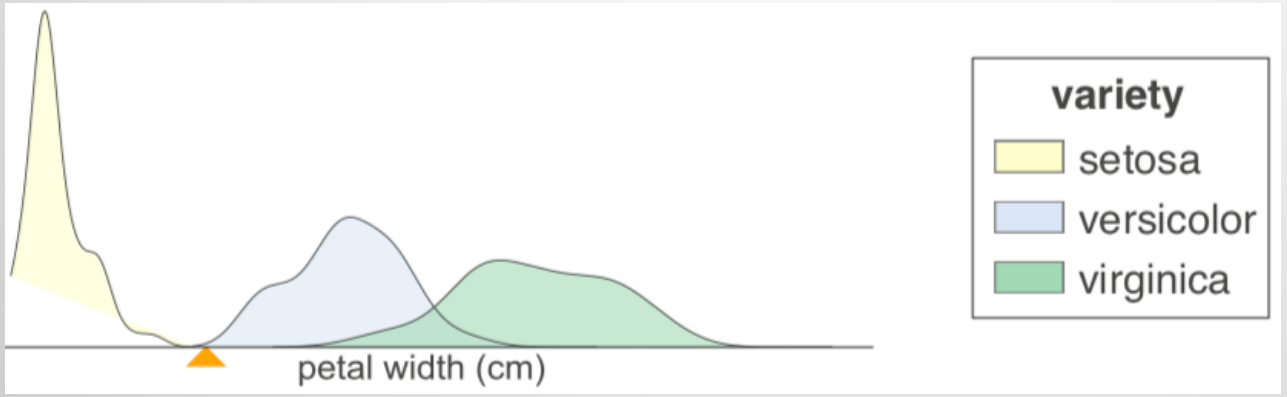
Highlight features Used in test vector

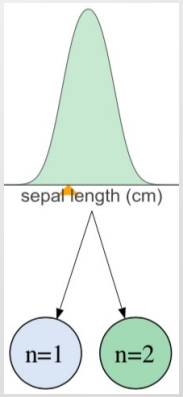
Pie charts shrink (nonlinearly) with sample size

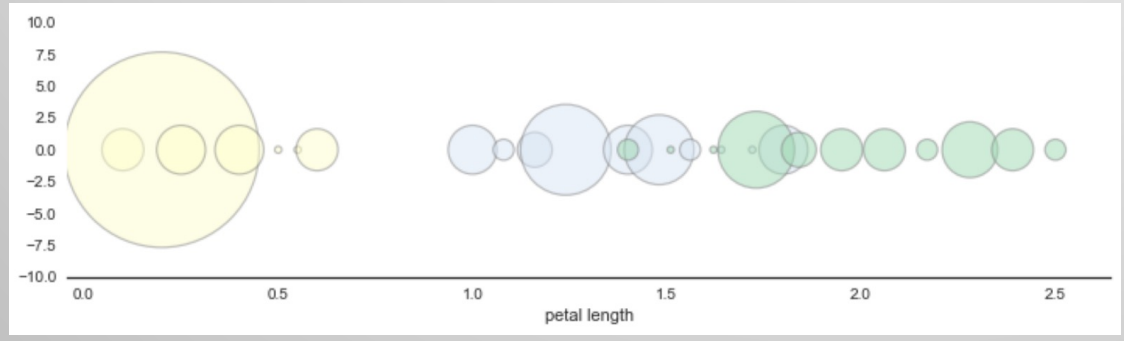


What we tried and rejected

Kernel density estimates (smooth histograms)

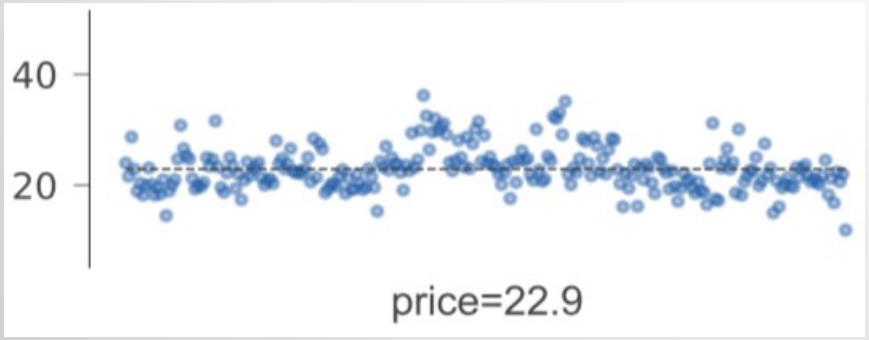


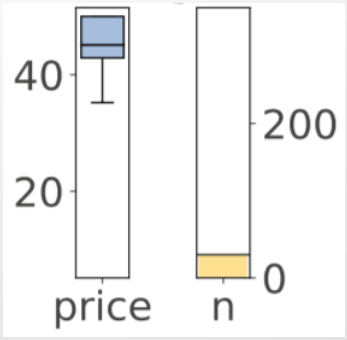


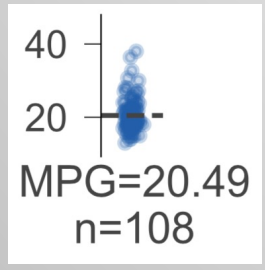


Bubble charts

Tried various regression leaf types







Settled on strip plot

Implementation overview

- We used matplotlib to generate images for decision, leaf nodes
- Combined nodes into a tree using graphviz
- Used HTML labels extensively in the graphviz for layout, fonts
- Single biggest headache: convincing all to produce high-quality vector graphics (SVG)
- Defined ShadowDecTree class to wrap sklearn trees

Generating high-quality graphics (SVG)

- Want scalable graphics like PDF or SVG, not PNG, GIF, JPG
- Import .svg or .pdf into graphviz? Nope.
- Many hours later, discovered magic incantation for graphviz
- Needed width/height of image; gotta parse SVG now
- Graphviz generates SVG that references other SVGs
  - Wrote manual SVG image embedding tool to get single meta-SVG file
- Uh oh. Graphviz doesn't handle fonts correctly with SVG output
  - Generate PDF then run pdf2svg tool to convert PDF to SVG
- Ah, but –Tsvg:cairo graphviz option does get fonts right
  - Switch code to straight SVG generation
- Final insult: Jupyter notebook doesn't show those SVG correctly

How to lead a fulfilling life by being dissatisfied

Be dissatisfied

- Be offended by poor software and unsatisfying results
- Crave beauty and quality
  - Quality must be baked in
  - Don’t put lipstick on a pig
- Make the world a better place for others as you move along
- Obsess about the details
  - Varying graph height, colors, box outlines, 3D plots, ...
- Warning:
  - Dissatisfied personalities are annoying to those in the “blast radius”
  - Can send you off on tangents
  - But, your dissatisfaction is a win for everyone else

Be tenacious

- Never let the computer win
  - Each loss would sap your confidence
  - Might require extraordinary lengths to solve
  - Lack of confidence leads to acceptance of status quo
- "Why program by hand in five days, what you can spend ~~five~~ **30** years of your life automating?"
  -- parrt on ANTLR
- Implementation of dtreeviz required pathological tenacity

Be courageous

- Slap your code up on github, push to pypi, ship it!
- Write articles or blog entries
- Otherwise, you're not helping the programming community
- Building code and writing articles exposes holes in your understanding; holes become chasms as you try to explain
- Face that fear; can't improve unless you identify weaknesses
- Students: these artifacts get you the job
- You might think you don't know anything (true at beginning)
  - Start with simple libraries or blogs, then stretch/reach
  - Practice expanding your abilities

Be a finisher

- Constant tension between perfectionism and finishing projects
- Details matter but must still ship it
- Go off on tangents to solve subproblems but always return to finish the original problem
- Make completion a key personality parameter to optimize
- I know researchers with only unpolished, unpublished projects
- How do you want to be measured?

How to contribute and make a difference

- Look for the pain
- Look for a hole in the market
- Erase the pain with reusable and general code
- Learn tactics from existing successful projects:
  - good name
  - website with domain devoted to project
  - useful doc
  - API design is hard; think hard about it
  - focus on conceptual integrity, consistent style
  - use social media to your advantage

“The reasonable [wo]man adapts [her|him]self to the world: the unreasonable one persists in trying to adapt the world to [her|him]self. Therefore all progress depends on the unreasonable [wo]man.”

— George Bernard Shaw

Resources

- Decision tree visualization article: https://explained.ai/decision-tree-viz/index.html
- My website for deep explanations of AI-related stuff: https://explained.ai
- fast.ai's Introduction to Machine Learning for Coders MOOC: https://course.fast.ai/ml
- U of San Francisco's MS Data Science program: https://www.usfca.edu/arts-sciences/graduate-programs/data-science
- https://matplotlib.org/
- http://graphviz.org/