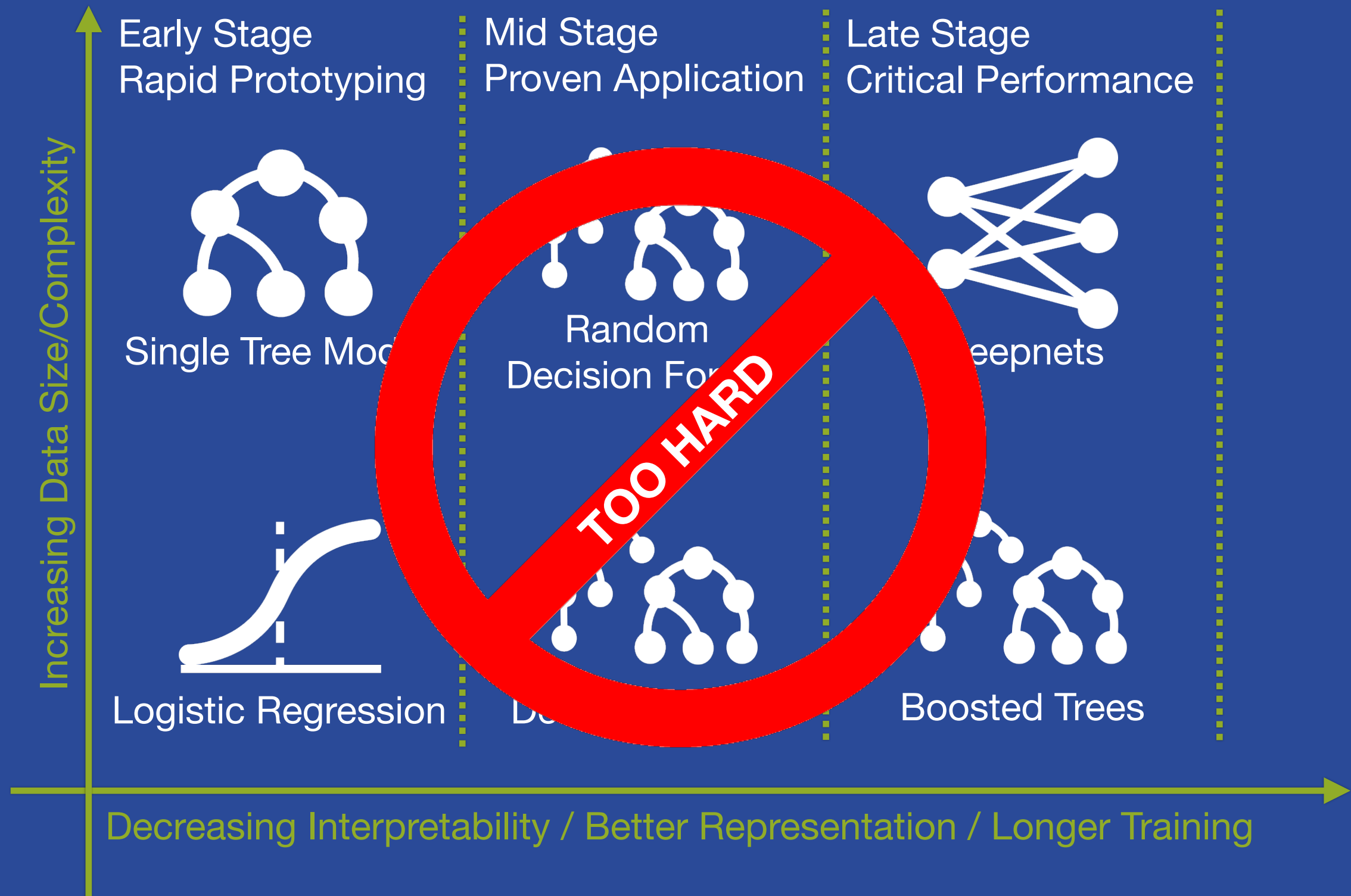


OptiML and Fusions

Automating Machine Learning

Charles Parker

VP ML Algorithms, BigML, Inc



Remember this?

- The success of a Deepnet is dependent on getting the right network structure for the dataset
- But, there are too many parameters:
 - Nodes, layers, activation function, learning rate, etc...
- And setting them takes significant expert knowledge
- Solution:
 - Metalearning (a good initial guess)
 - Network search (try a bunch)

*Key Insight: We can solve **any** parameter selection problem in a similar way.*

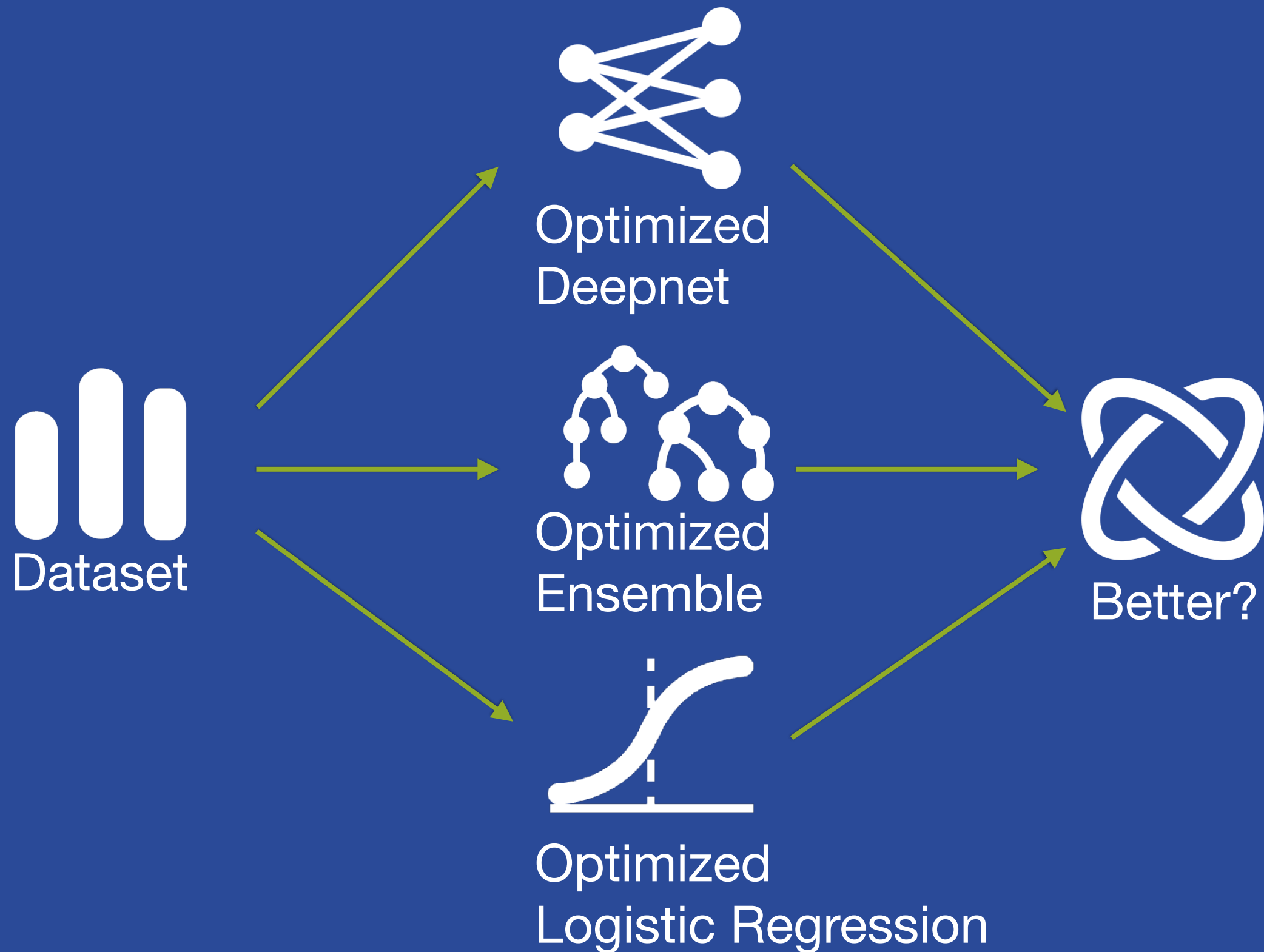
- Each resource has several parameters that impact quality
 - Number of trees, missing splits, nodes, weight
- Rather than trial and error, we can use ML to find ideal parameters
- Why not make the model type, Decision Tree, Boosted Tree, etc, a parameter as well?
- Similar to Deepnet network search, but finds the optimum machine learning **algorithm** and **parameters** for your data **automatically**

OptiML Demo

Key Insight: ML algorithms each have unique strengths and weaknesses

- Fuse any set of models into a new “fusion”
 - Must have the same objective type
 - Inputs and feature space **can** differ
- Weights can be added
 - Give more importance to individual models
- Fusions can be fused as well
- Especially useful for fusing OptiML models

Performance thru Diversity



Fusion Demo #1

Fusions: Under the Hood

Classification

Model Prediction Probability Weight

Ensemble	TRUE	%56	1
Deepnet	FALSE	%67	1
Model	TRUE	%78	2

Fusion

TRUE	%61
------	-----

$$P(\text{TRUE}) = [56 + (100 - 67) + 2 * 78] / 4$$

Regression

Model Prediction Error Weight

Ensemble	156.78	12.56	1
Deepnet	139.55	9.88	1
Model	172.10	23.76	2

Fusion

160.13	17.49
--------	-------

Fusions: Like any BigML Model



- Fully accessible thru API and WhizzML
- Bindings have support for local predictions

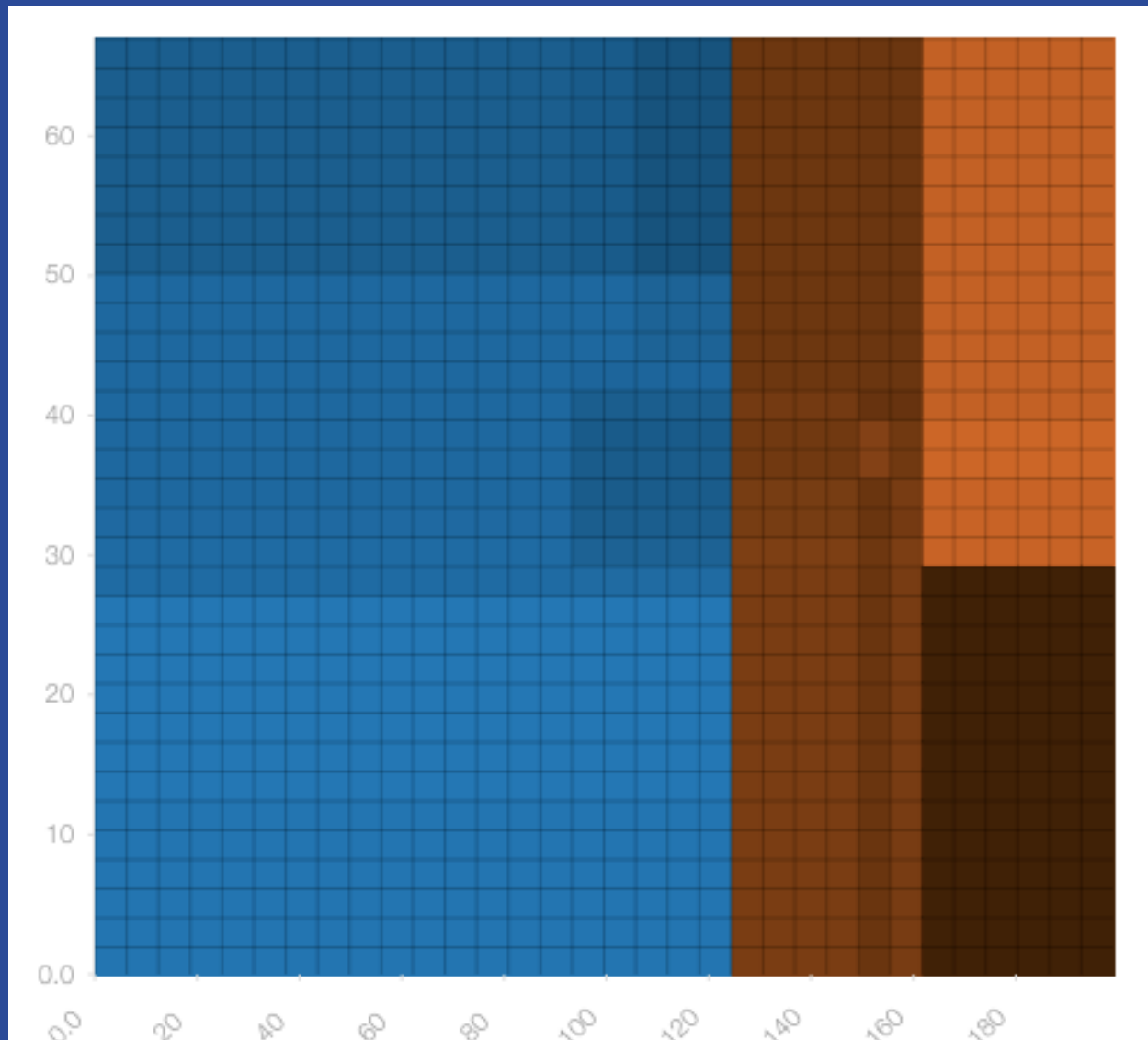
```
;; WhizzML - create a fusion
(define my-fusion (create-fusion {"models" my-best-models}))
```

```
# Python - create a fusion
fusion = api.create_fusion([ "model/5af06df94e17277501000010",
                             "logisticregression/5af06df84e17277502000019",
                             "deepnet/5af06df84e17277502000016",
                             "ensemble/5af06df74e1727750100000d" ]})
```

Decision Boundary Smoothness

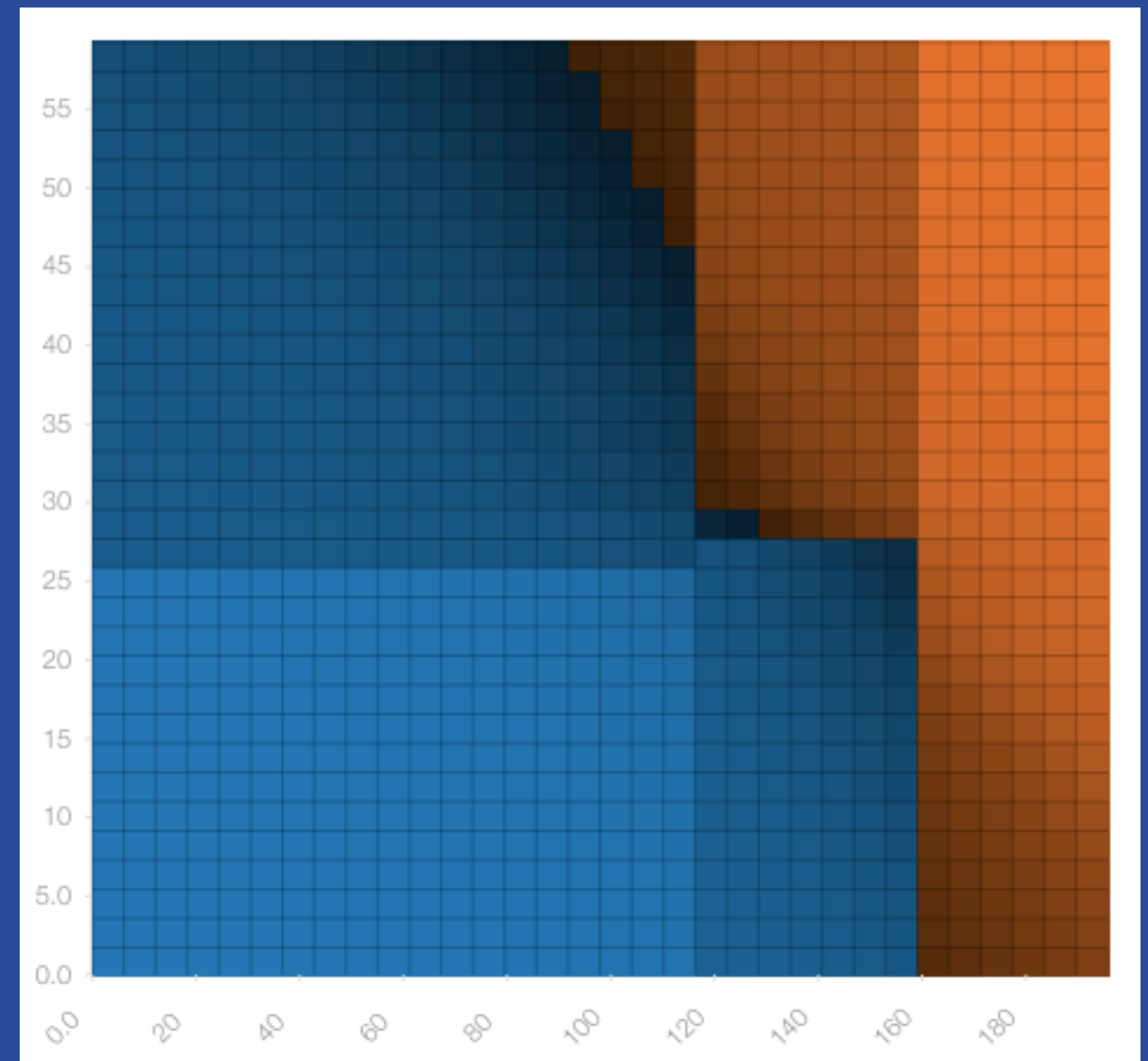
Single Tree:

- Outcome changes abruptly near decision boundary
- And not at all parallel to the boundary
- This can be “surprising”

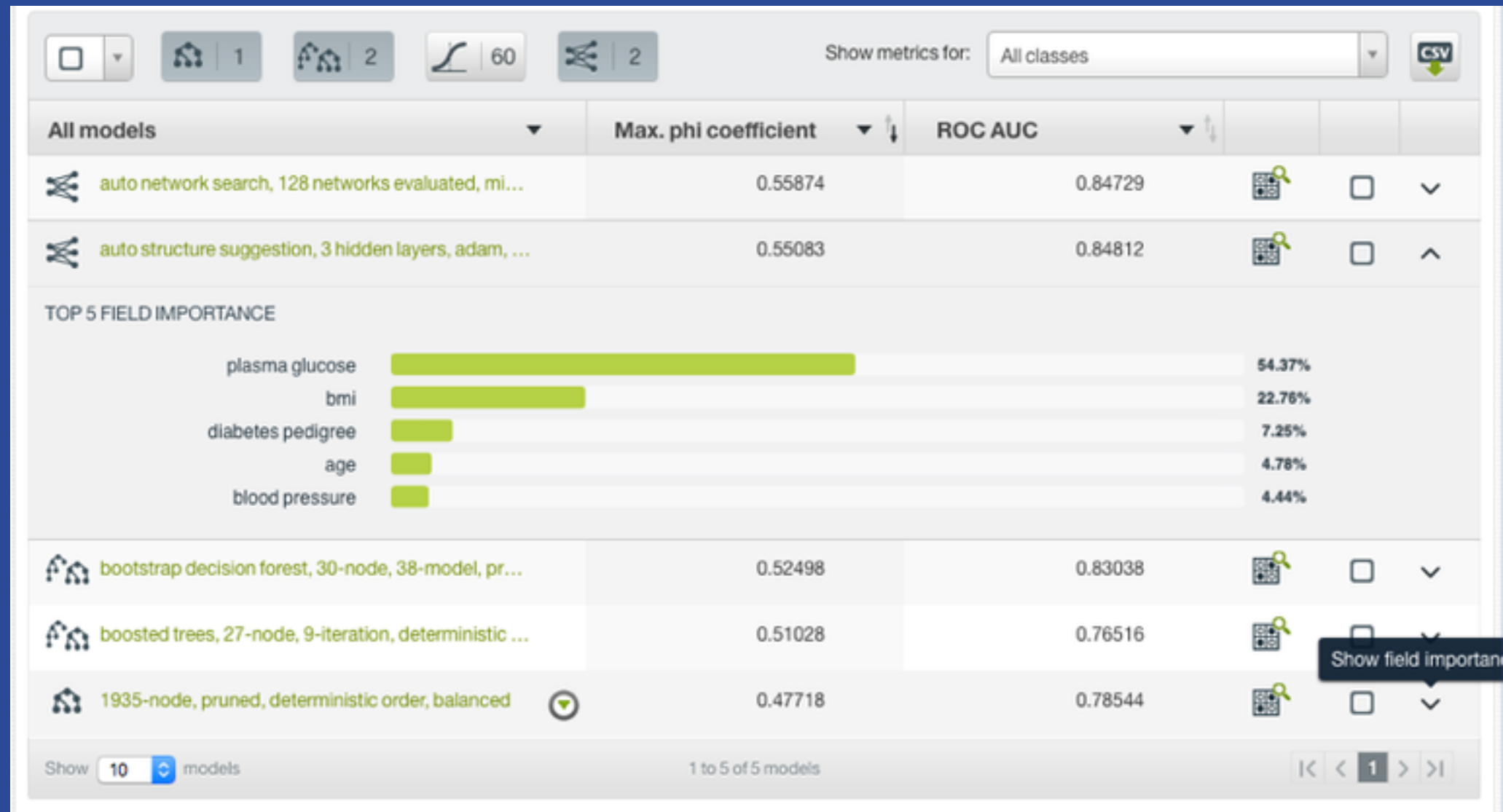


Single Tree + Deepnet:

- Keep the interpretability of the tree
- But with a more nuanced decision boundary



Feature Stability



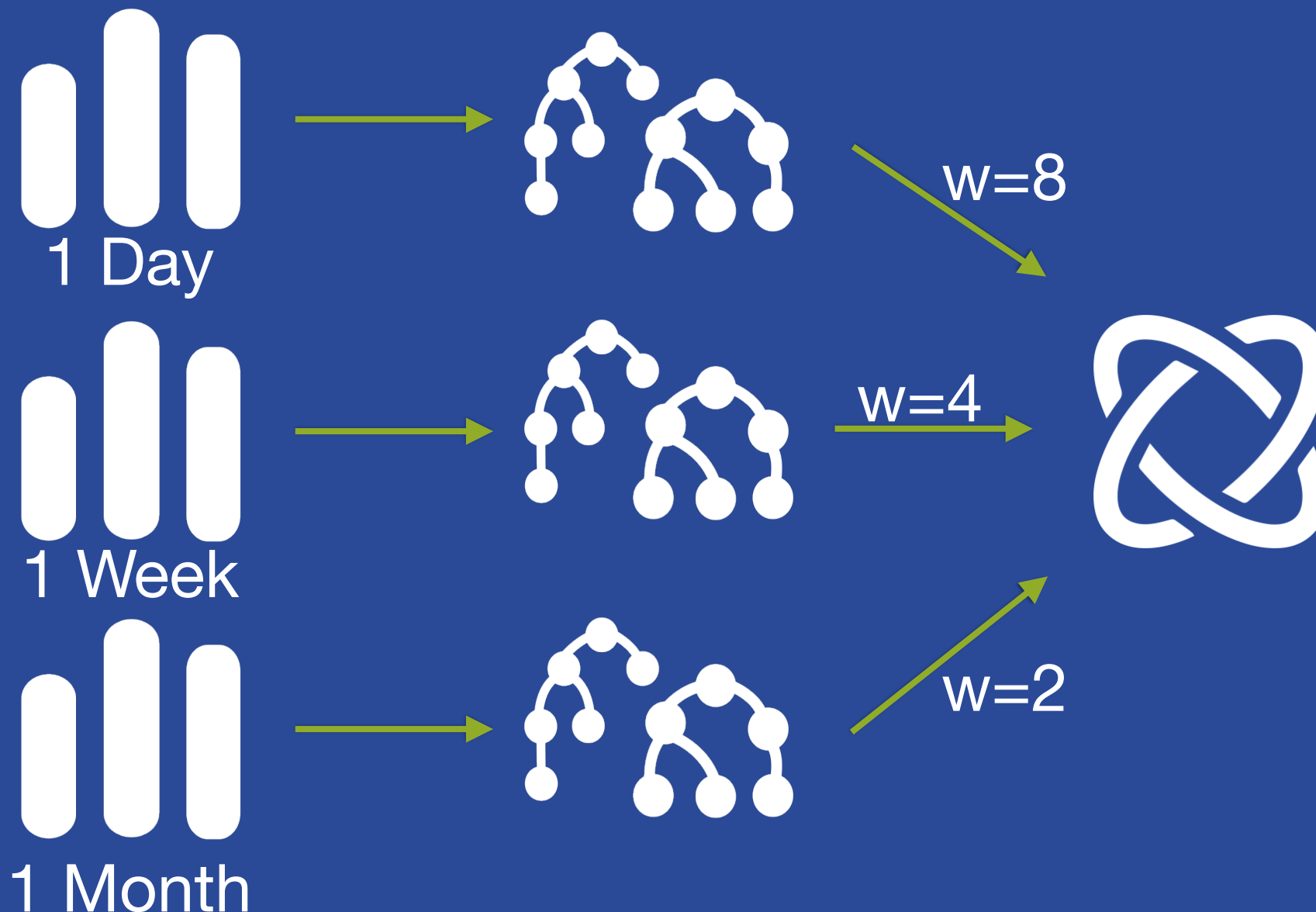
***Feature Importance:** Different subsets of features may have similar modeling performance*

Fusing models gives better resilience against missing values as well as ensuring that all relevant features are utilized.

Weighting over Time

Data significance over time:

- Some data may change significance in different times
- Short-term user behavior versus long-term
- Weights can set to account for significance of time

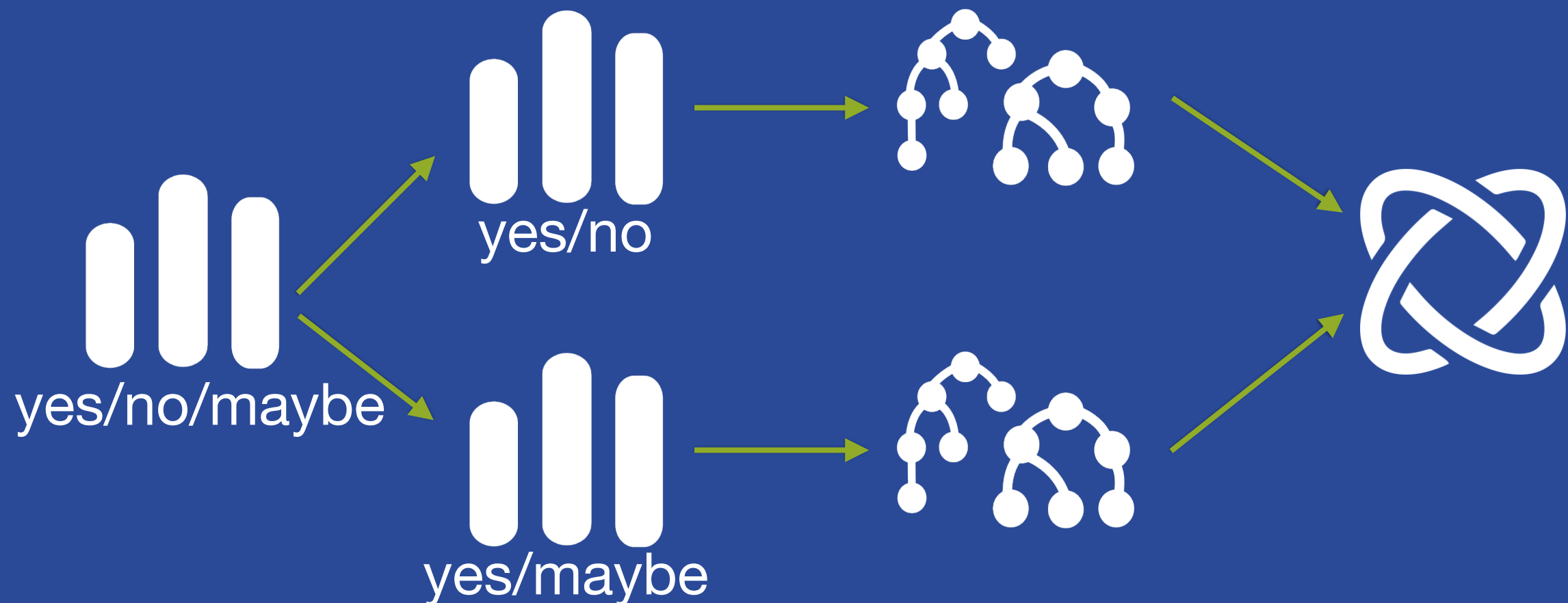


Improved Class Separation

Consider a 3-class objective

Yes	No	Maybe
-----	----	-------

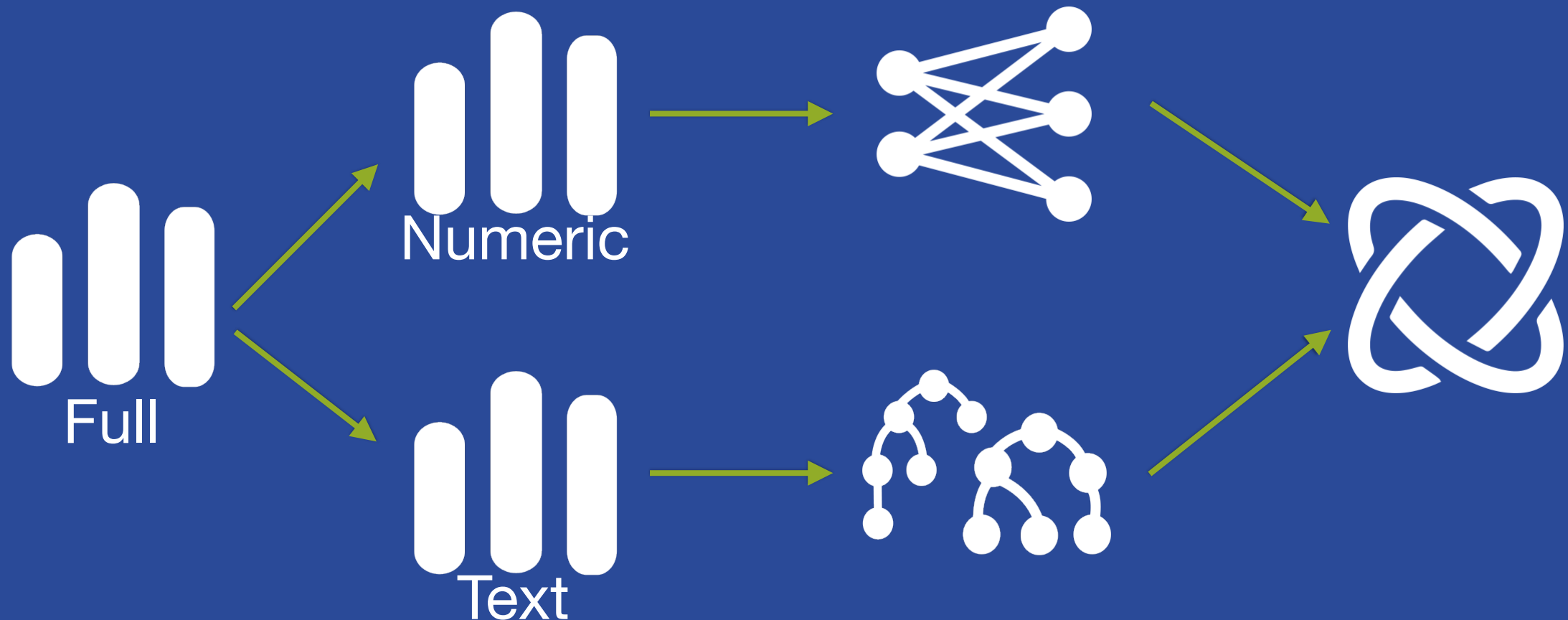
- Really only care about “yes” versus “not yes”
- A single model may struggle to separate the two negative classes



Feature Space Optimization

Model Skills: Some ML algorithms “generally” do better on some feature types:

- RDF for sparse text vectors
- LR/Deepnets for numeric features
- Trees for categorical features



Fusions Demo #2

Your Turn!



- *Note:* best to work in groups to limit computation time
- Configure an OptiML of the Diabetes 80%
 - Limit number of model candidates to 10
 - Disable Deepnets from the search
 - Optimize for identifying diabetes
- While the OptiML is running:
 - Build a Fusion from any set of Diabetes 80% models
 - Evaluate the fusion with the 20%
 - How does it compare to previous models?
- Returning to the OptiML
 - Evaluate the top performing model with the 20%
 - How does it perform?

