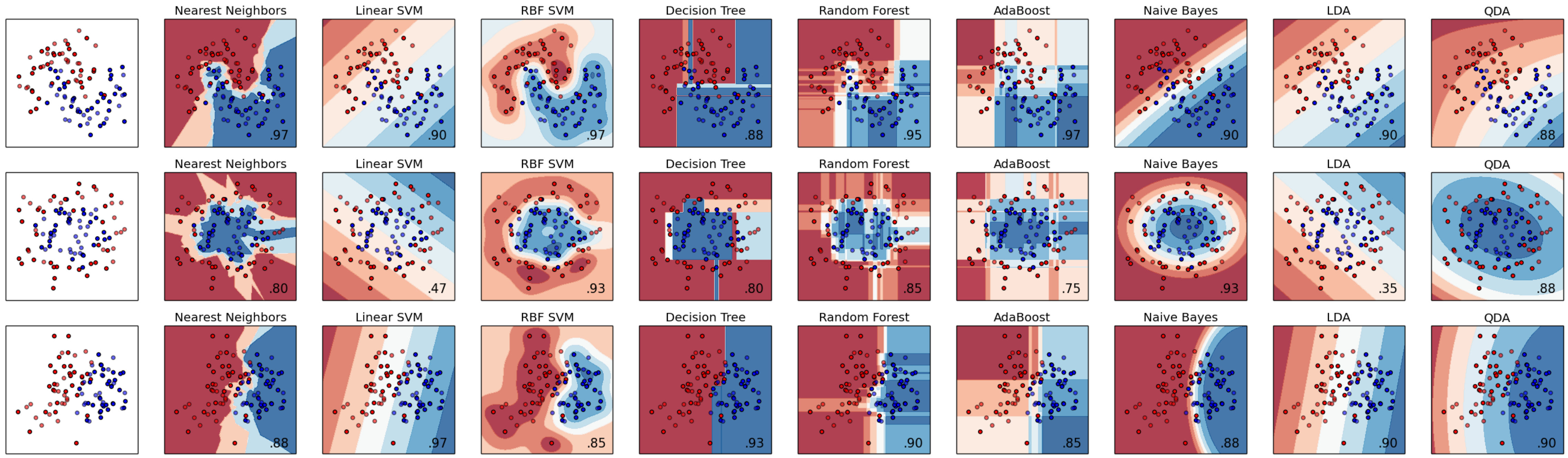


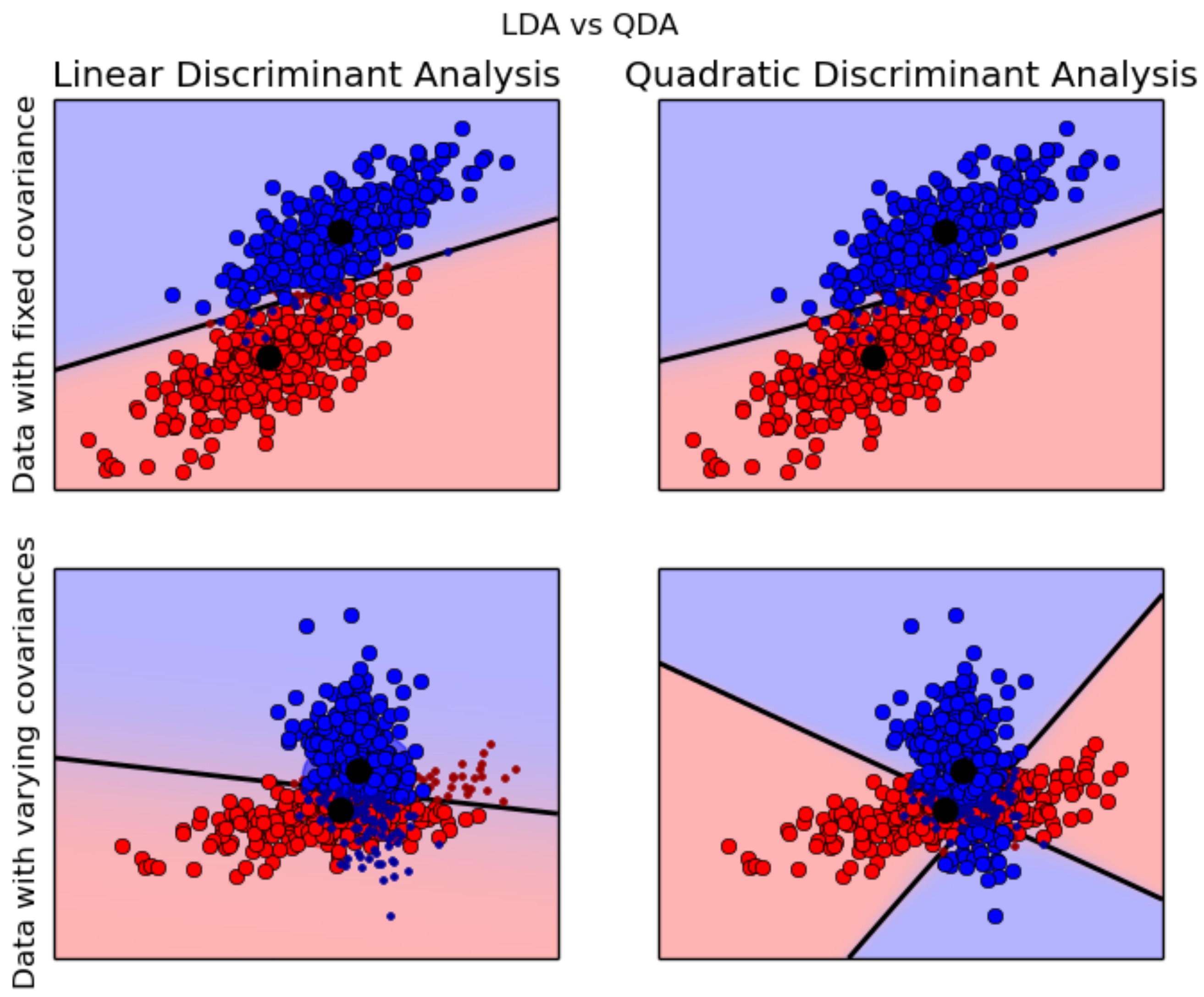
Algo-Palooza

(BIG) CAVEAT:

Often times choosing/creating good features or gathering more data will help more than changing algorithms…

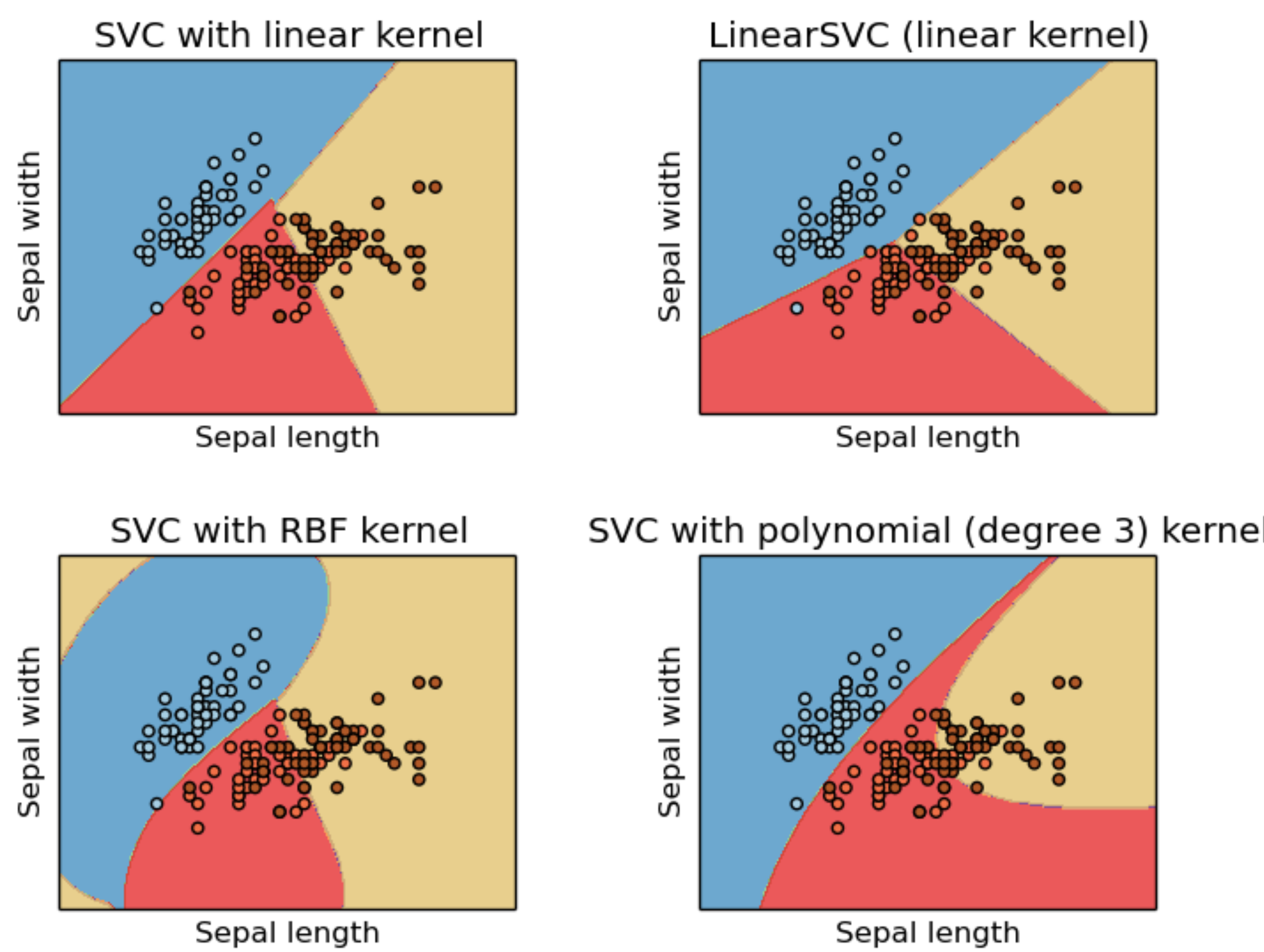
Linear Discriminants

"draw a line through it"



SVMs (Support Vector Machines)

"Advanced draw-a-line-through-it"



Pros:

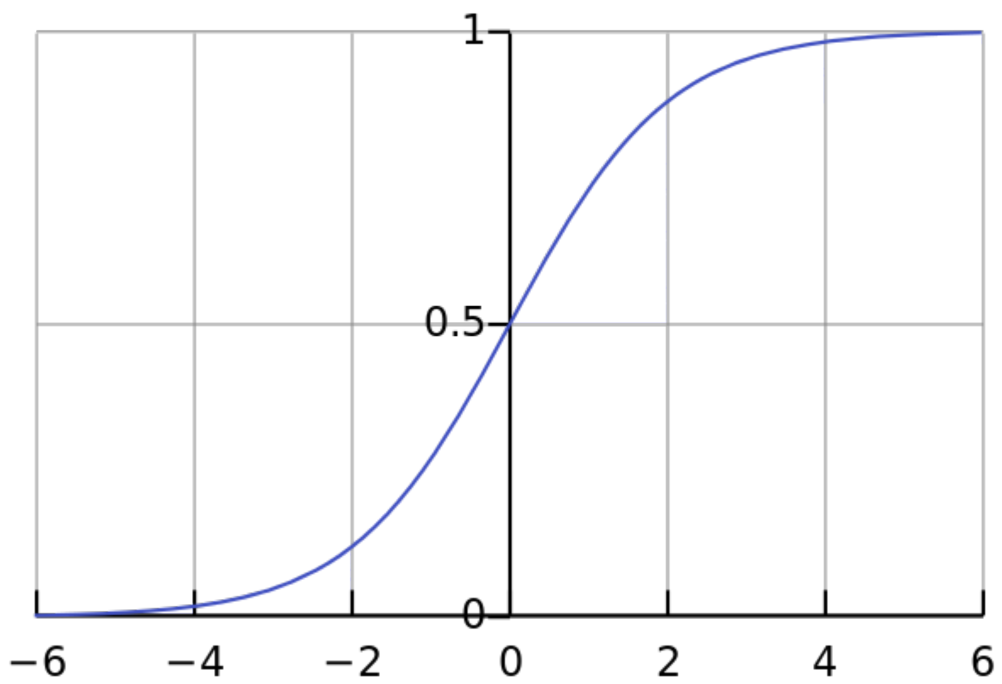
- can work well on many types of data (low or high-dimensional, linearly divisible or not) data thanks to the “kernel trick“

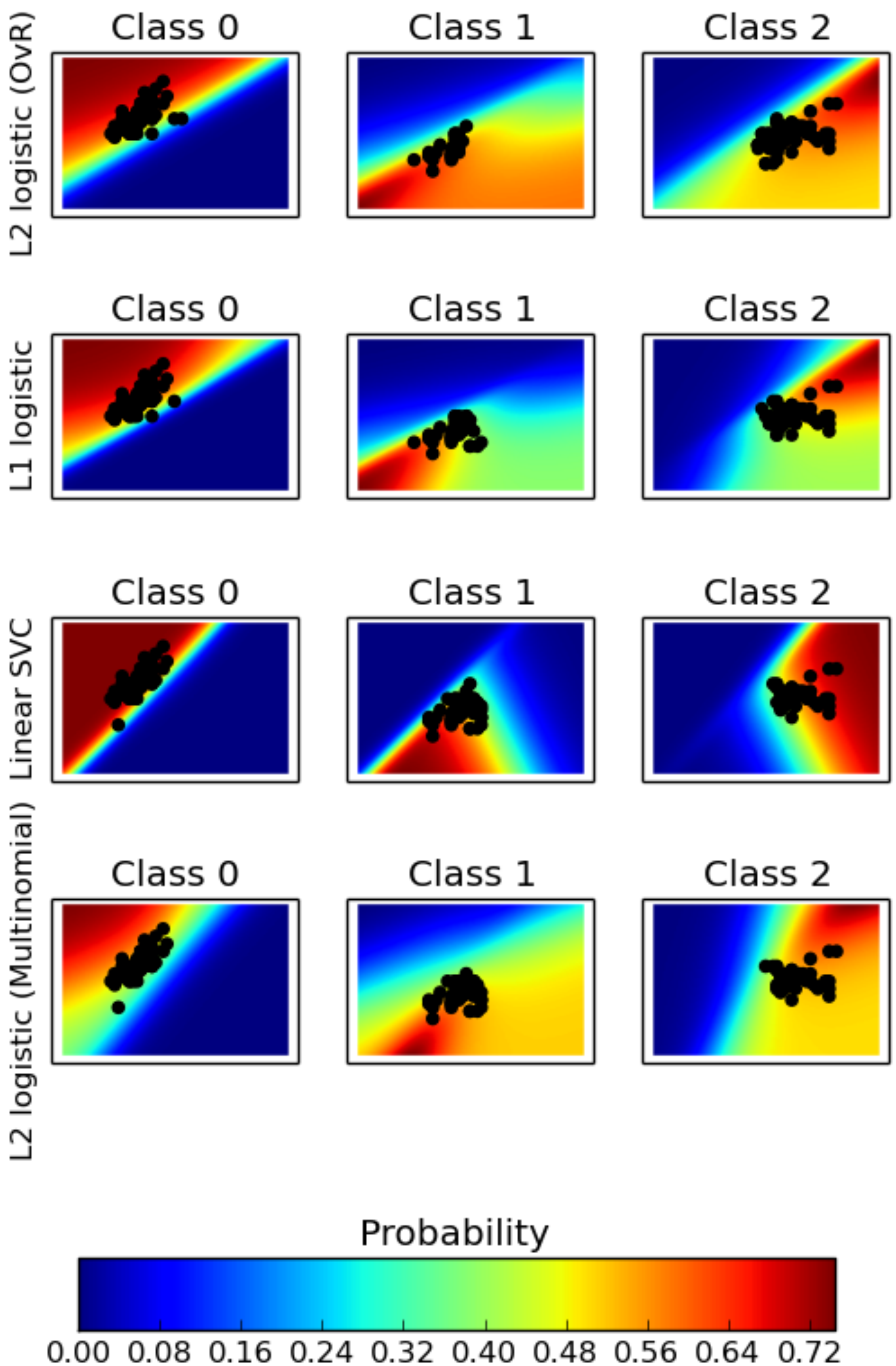
Cons:

- Not as easy to explain or tweak
- Can only *kinda* provide probability estimates—computationally intensive

Logistic Regression

"divide it with a logistic function"





Pros:

- Can provide probabilities (has “fuzzy edges“)
- Can update with new data

Cons:

- Limited to linear decision boundaries

Naïve Bayes

“calculate a probability of it”

- Uses the training data to build conditional probabilities for each feature in X
- When classifying, use the probabilities calculated in training + Bayes Rule to calculate chance of each possible result given the data

Pros:

- Fast + simple (and parallelizes pretty well)
- Can provide probability estimates
- Works well with small amounts of data
- Can add new data without “retraining”

Cons:

- Assumes independence between features (though it can do a good job even if this is a 💩-y assumption)
- Can’t learn interactions between features

Decision Tree

“make a flow chart to describe it”

Pros:

- Easy to interpret results, especially at low dimensions/simplistic models
- Very fast
- Can adapt to many shapes of data

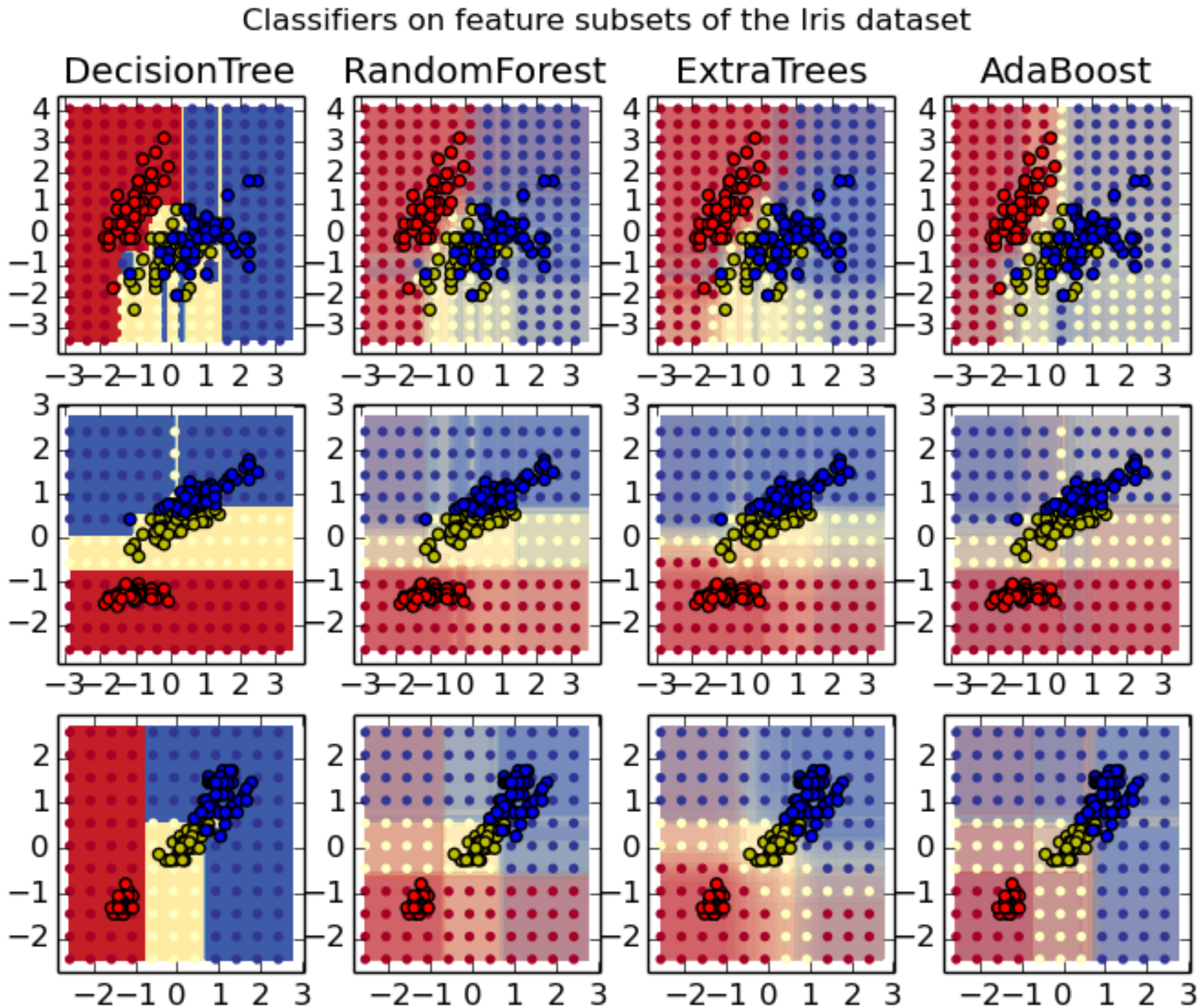
Cons:

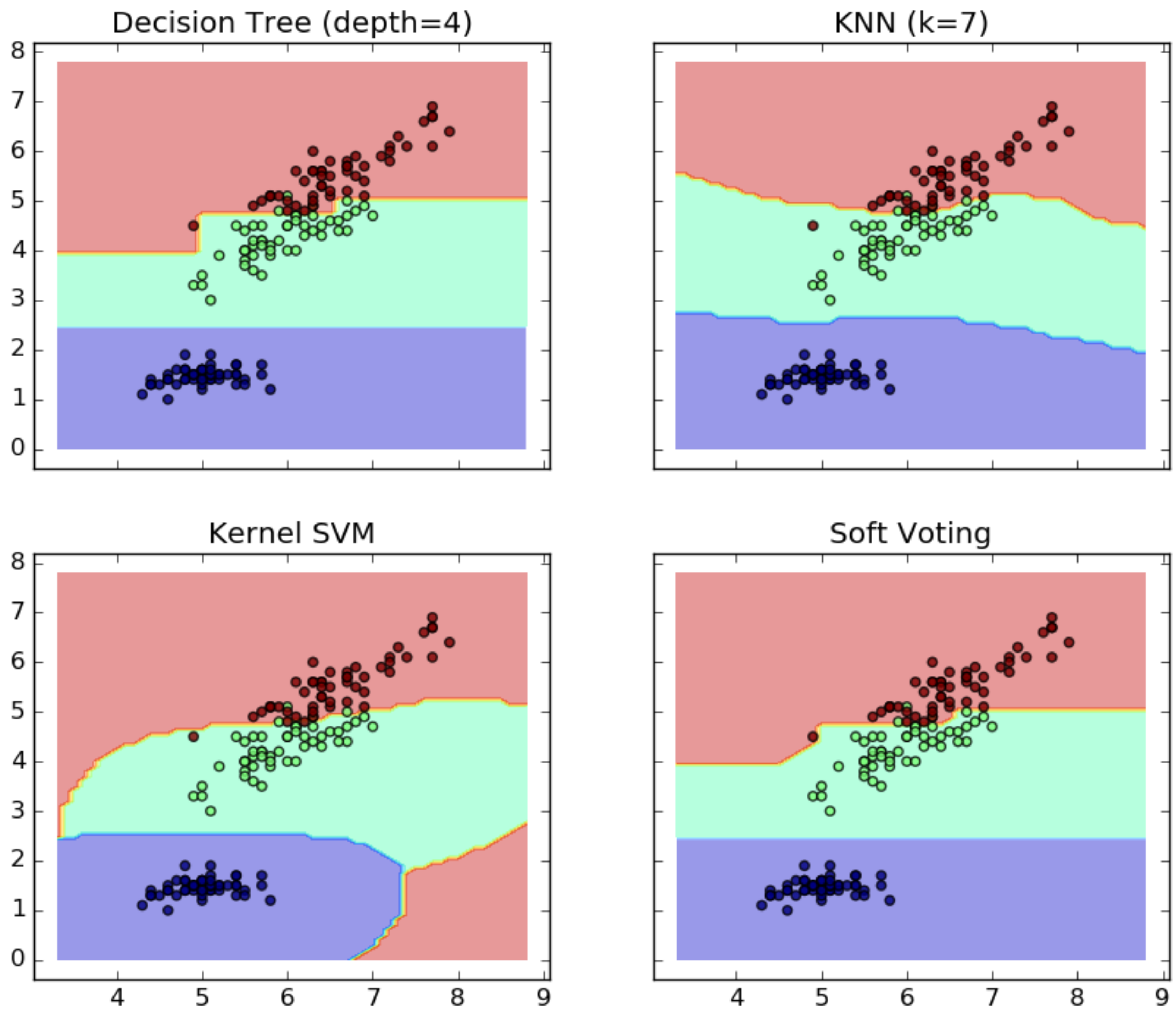
- Very easy to overfit with these if you aren’t careful.
- Have to rebuild any time you get new data.
- No probability estimates

Ensemble Methods

"combine results from a bunch of models"

- Bagging - bootstrap (sample) training set into multiple sets, train one model per set, take mode of results
- Random Forest - like bagging, but at each split randomly constrain features to choose from
- Extra Trees - for each split, make it randomly, non-optimally. Compensate by training a ton of trees
- Or build your own! See `VotingClassifier`


Classifiers on feature subsets of the Iris dataset
DecisionTree
RandomForest
ExtraTrees
AdaBoost



Decision Tree (depth=4)
KNN (k=7)
Kernel SVM
Soft Voting
0
1
2
3
4
5
6
7
8
9

Pros:

- Generally don’t require much parameter tweaking
- If data doesn’t change very often, you can make them semi-online by just adding new trees to the ensemble
- Can provide shades of gray and feature importances (number of cases where a feature was used in splits)
- Parallelize quite well

Cons

- Slower than their component parts (though if those are fast, it doesn't matter)

Boosting

“train a ‘team’ of classifiers step by step, combine results”

- Adaboost– after training each model, emphasize the data points we misclassified when training the next one.
- Gradient Boosting is a generalization that allows you to plug in different loss functions

Pros:

- Same niceties of other ensembles: probabilities, importances, “semi online” warm starts
- Reigning champ on a variety of classification tasks recently

Cons:

- Harder to parallelize
- Requires more parameter tweaking