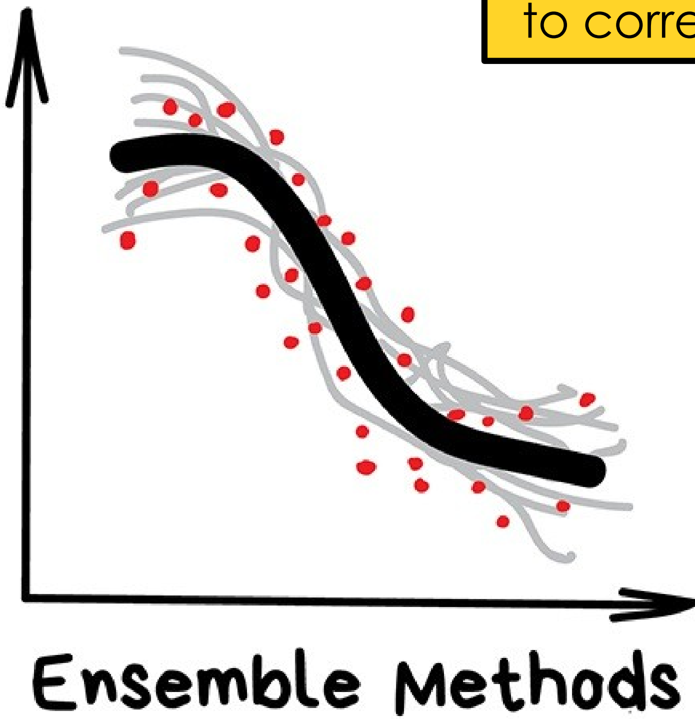
Ensemble methods

Vlad Gladkikh

IBS CMCM

Bunch of stupid trees learning to correct errors of each other



https://vas3k.com/blog/machine_learning/

https://www.kdnuggets.com/tag/ensemble-methods

Take many inefficient algorithms.

Force them to correct each other's mistakes.

The overall quality will be higher than even the best individual algorithms.

Types of ensemble learning:

Stacking

Bagging

Boosting

Popular in industry and in Kaggle competitions

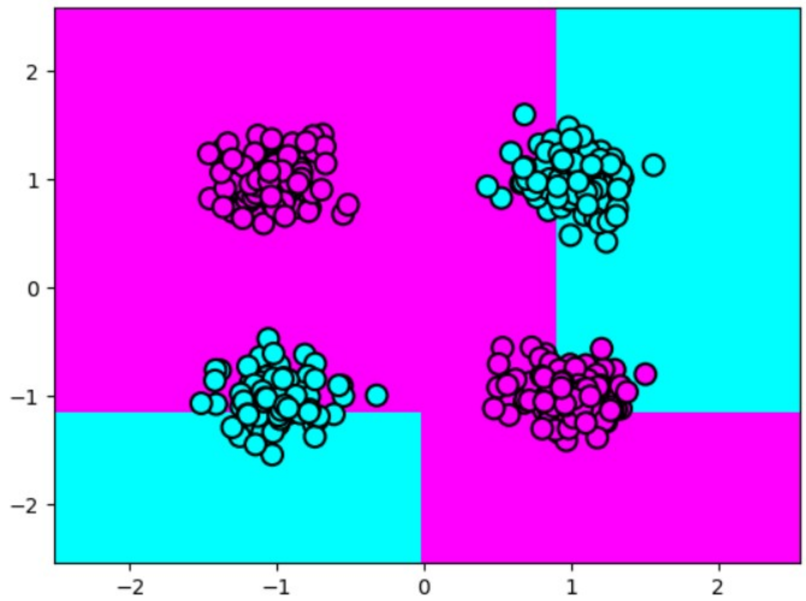
Useful when little training data, training time and little expertise for parameter tuning.

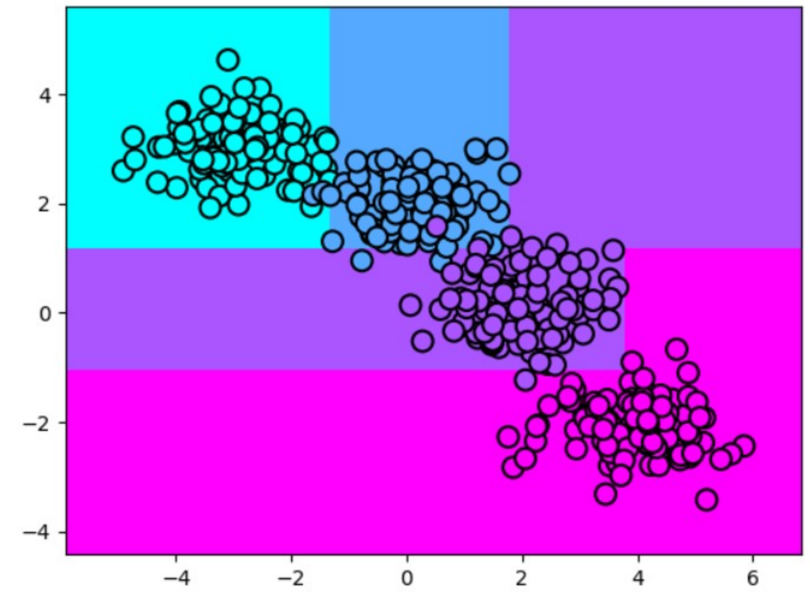
We can use any algorithm we know to create an ensemble.

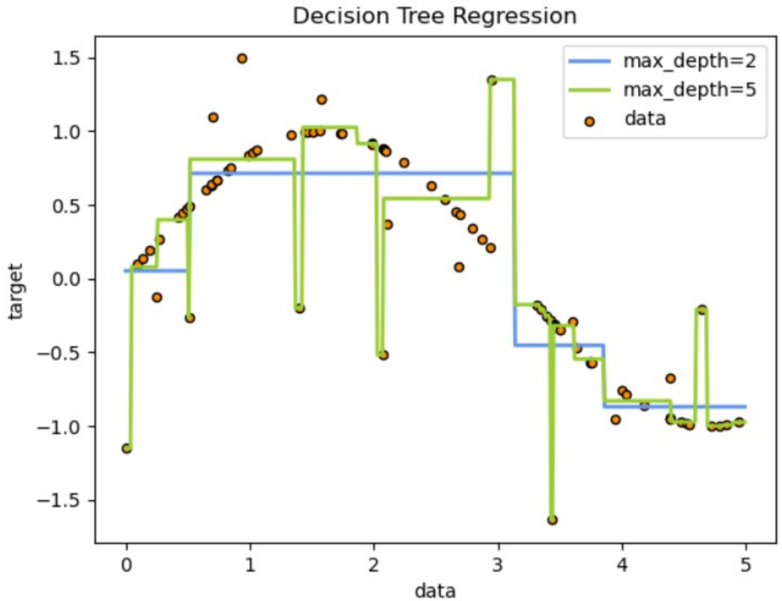
Even better results can be obtained with the most unstable algorithms that are predicting completely different results on small noise in input data, like regression and decision trees.

https://www.analyticsvidhya.com/blog/2018/06/comprehensive-guide-for-ensemble-models/

Reminder: problems with trees





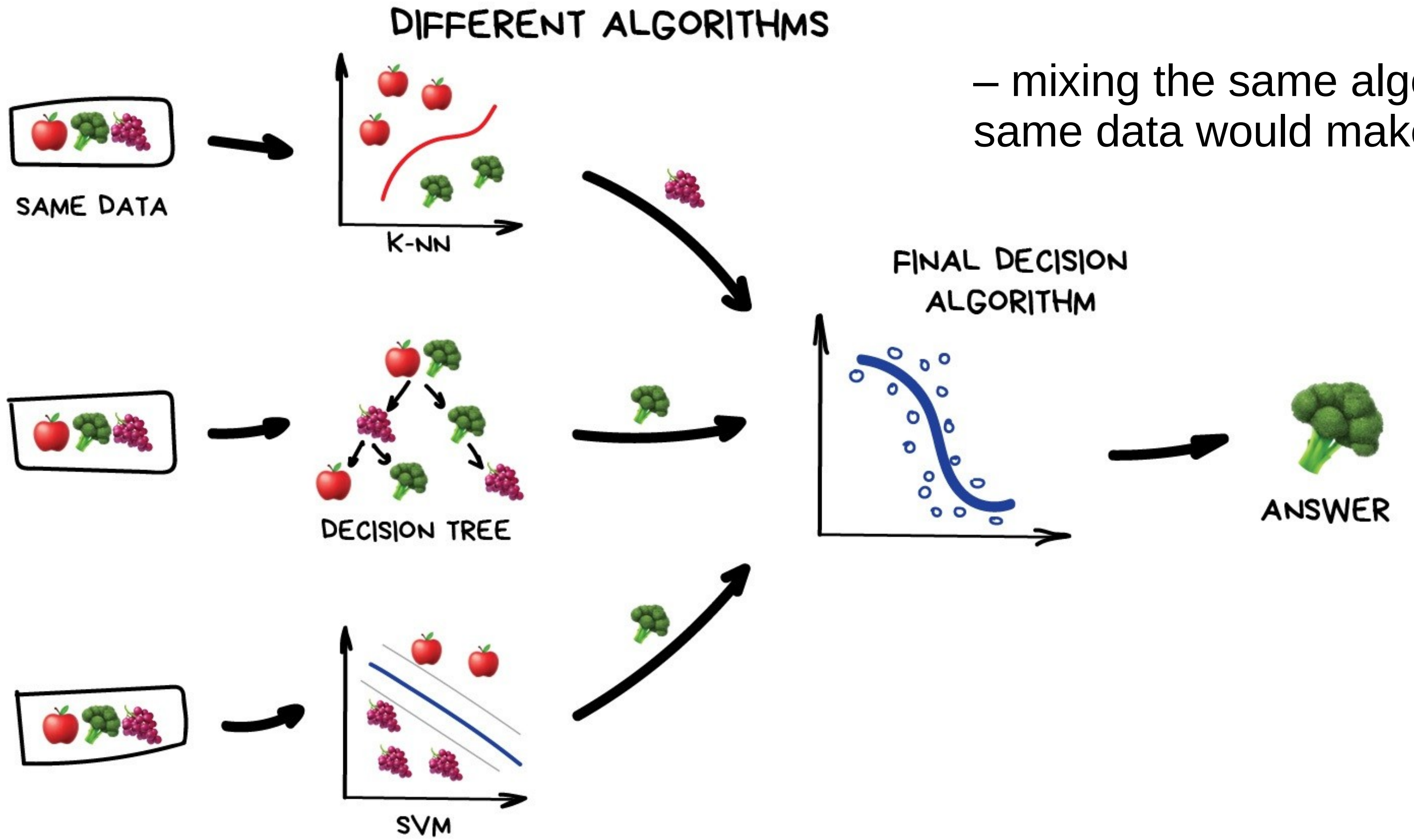


Learning an optimal decision tree is a global optimization problem.

A small change in the training data can result in a large change in the tree and consequently the final predictions.

Stacking

Output of several parallel models is passed as input to the last one which makes a final decision.



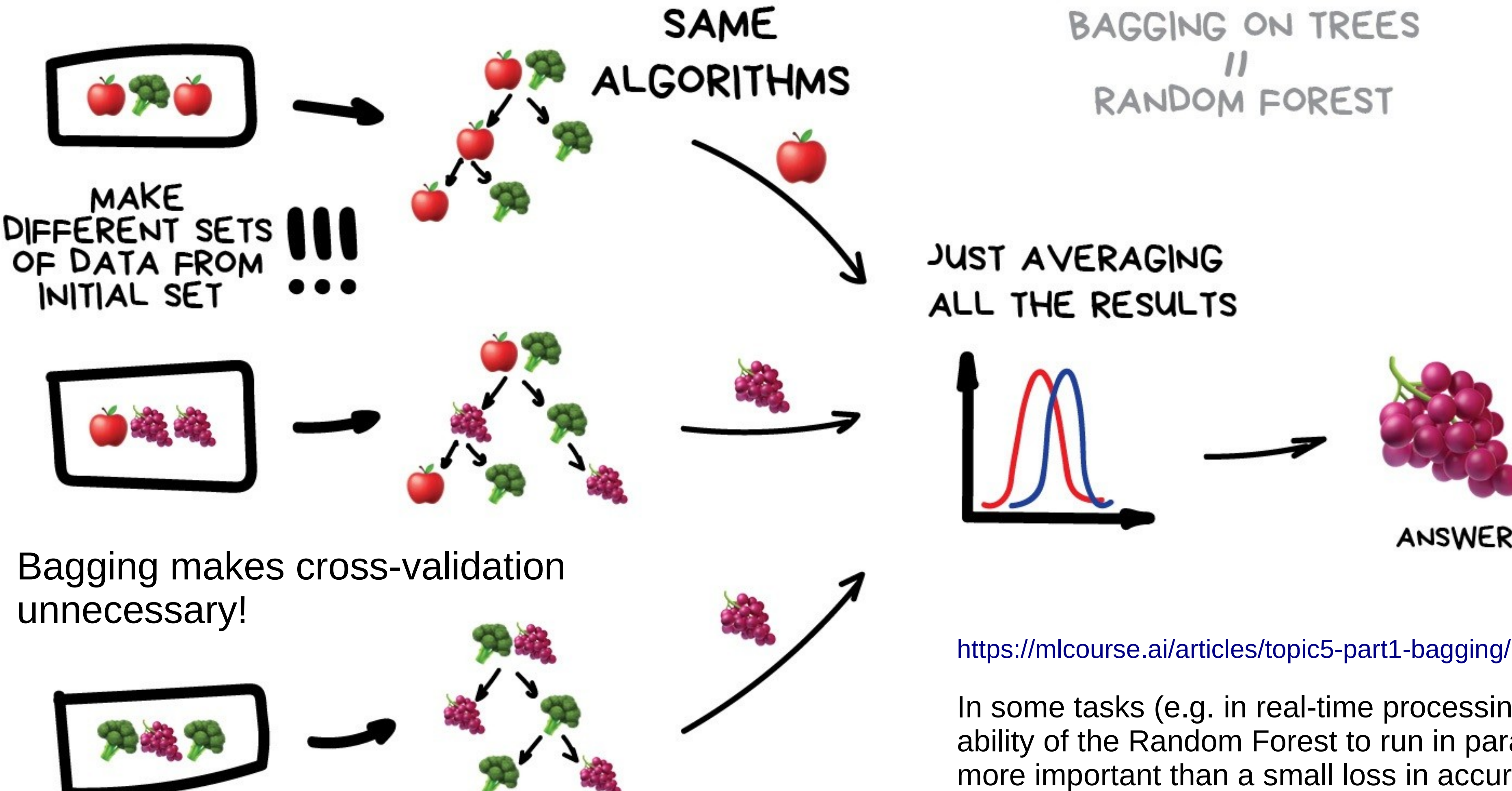
– mixing the same algorithms on the same data would make no sense.

https://vas3k.com/blog/machine_learning/

Bagging aka Bootstrap AGGregatING

Use the same algorithm but train it on different subsets of original data. Then average the answers.

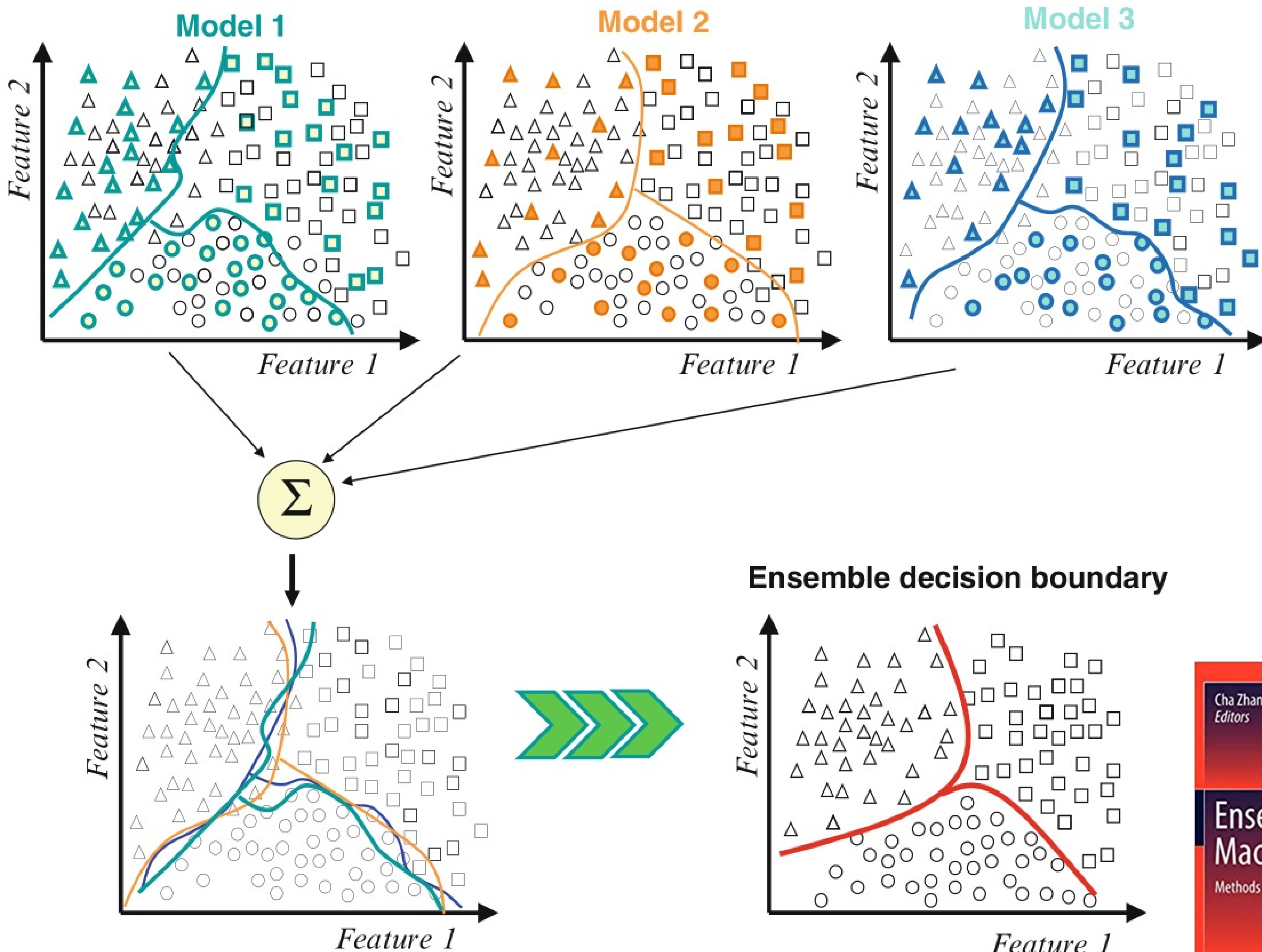
Data in random subsets may repeat. E.g., from a set "1-2-3" we get subsets like "2-2-3", "1-2-2", "3-1-2" etc. Use these new datasets to teach the same algorithm several times and then predict the final answer via majority voting.

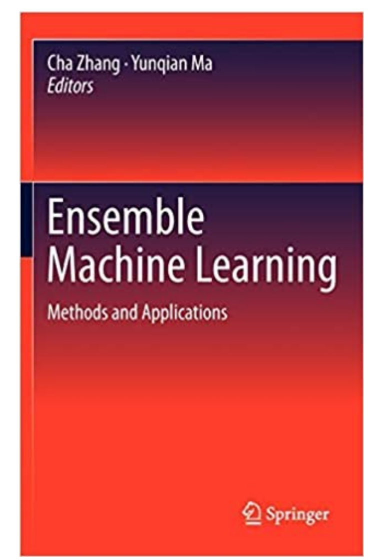


Bagging makes cross-validation unnecessary!

https://mlcourse.ai/articles/topic5-part1-bagging/

In some tasks (e.g. in real-time processing), the ability of the Random Forest to run in parallel is more important than a small loss in accuracy to the boosting, for example.

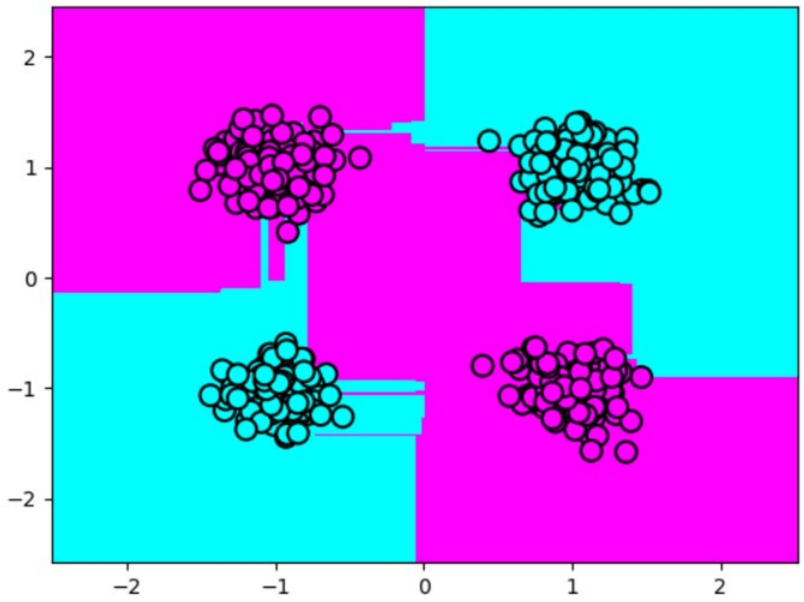

Model 1
Model 2
Model 3
Feature 2
Feature 1
Σ
Ensemble decision boundary


Cha Zhang · Yunqian Ma
Editors
Ensemble
Machine Learning
Methods and Applications
Springer

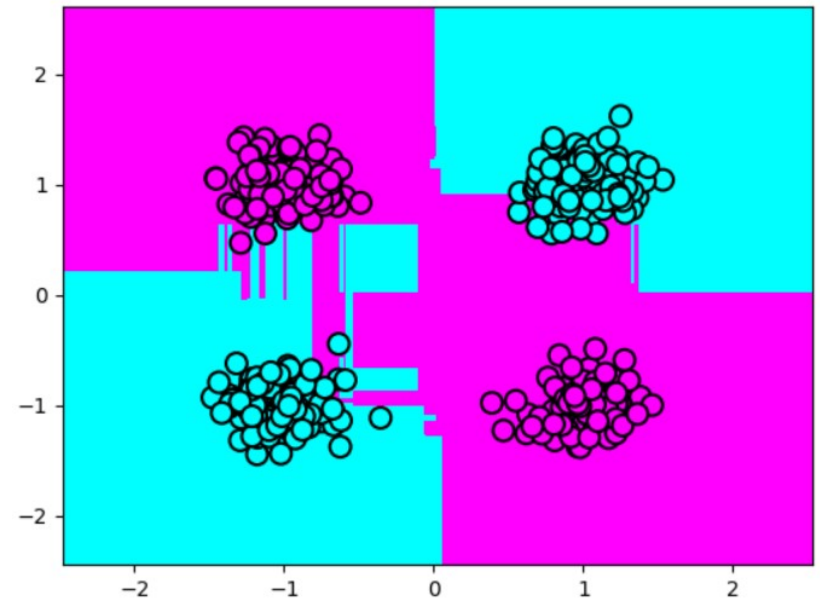
Random Forest

max_depth=2

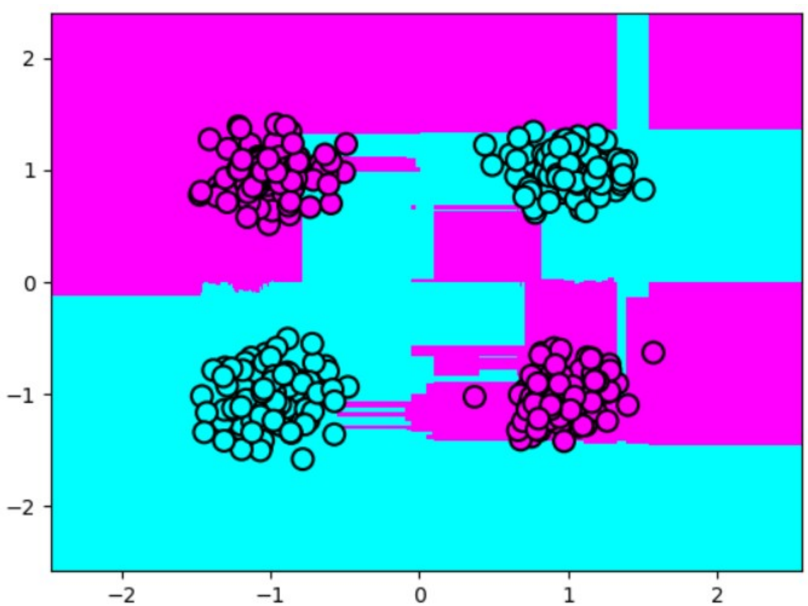
n_estimators=50



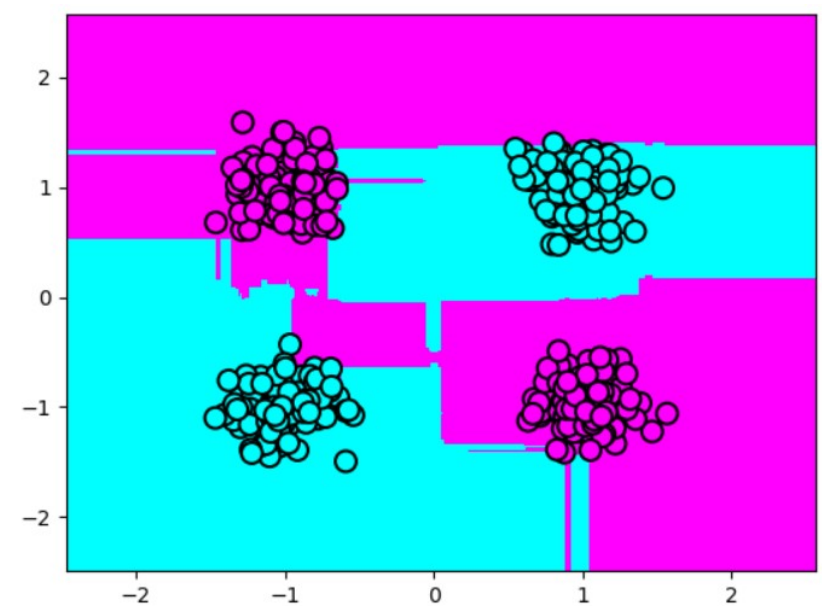
n_estimators=100



n_estimators=500

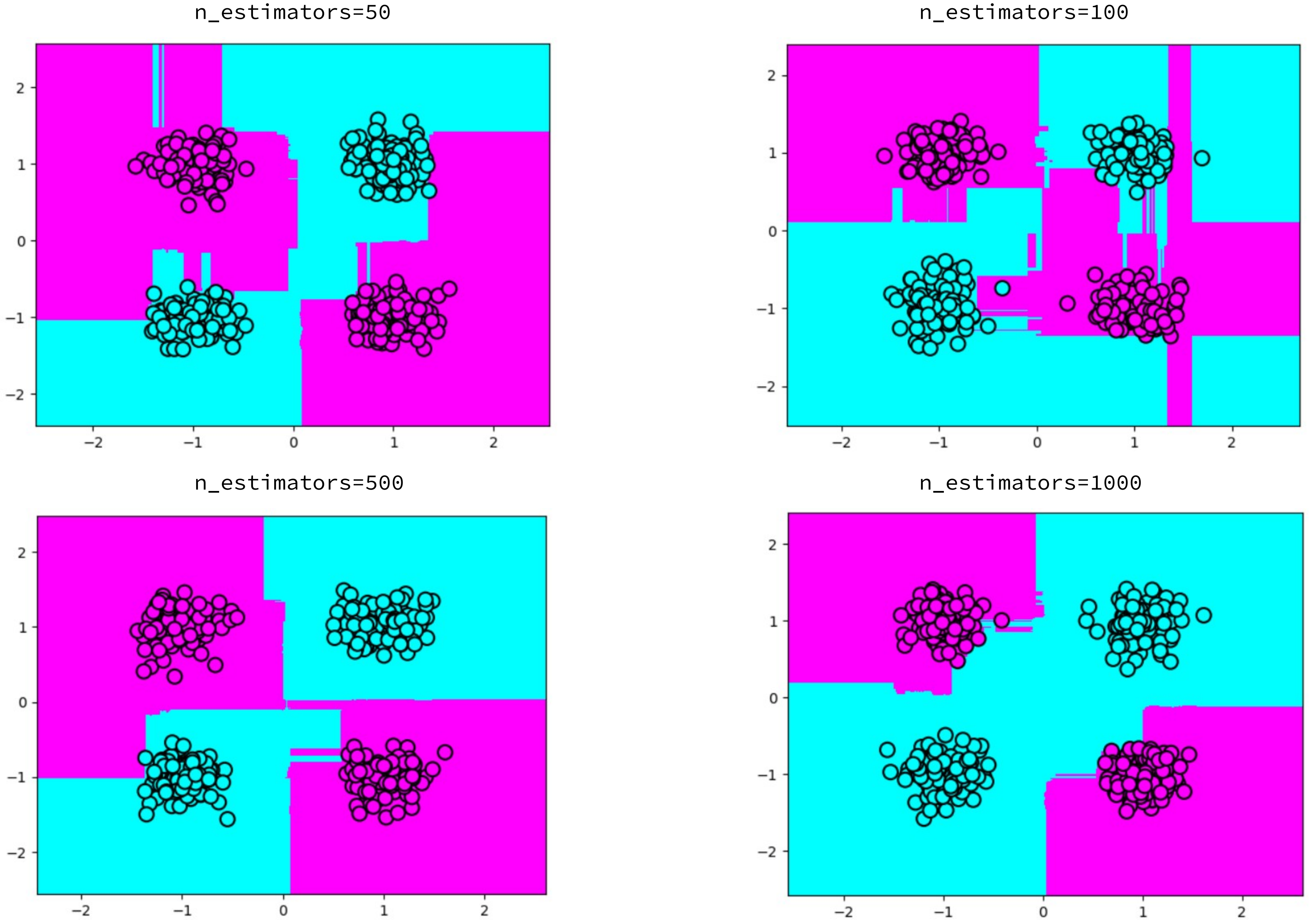


n_estimators=1000



Random Forest

max_depth=3



Boosting

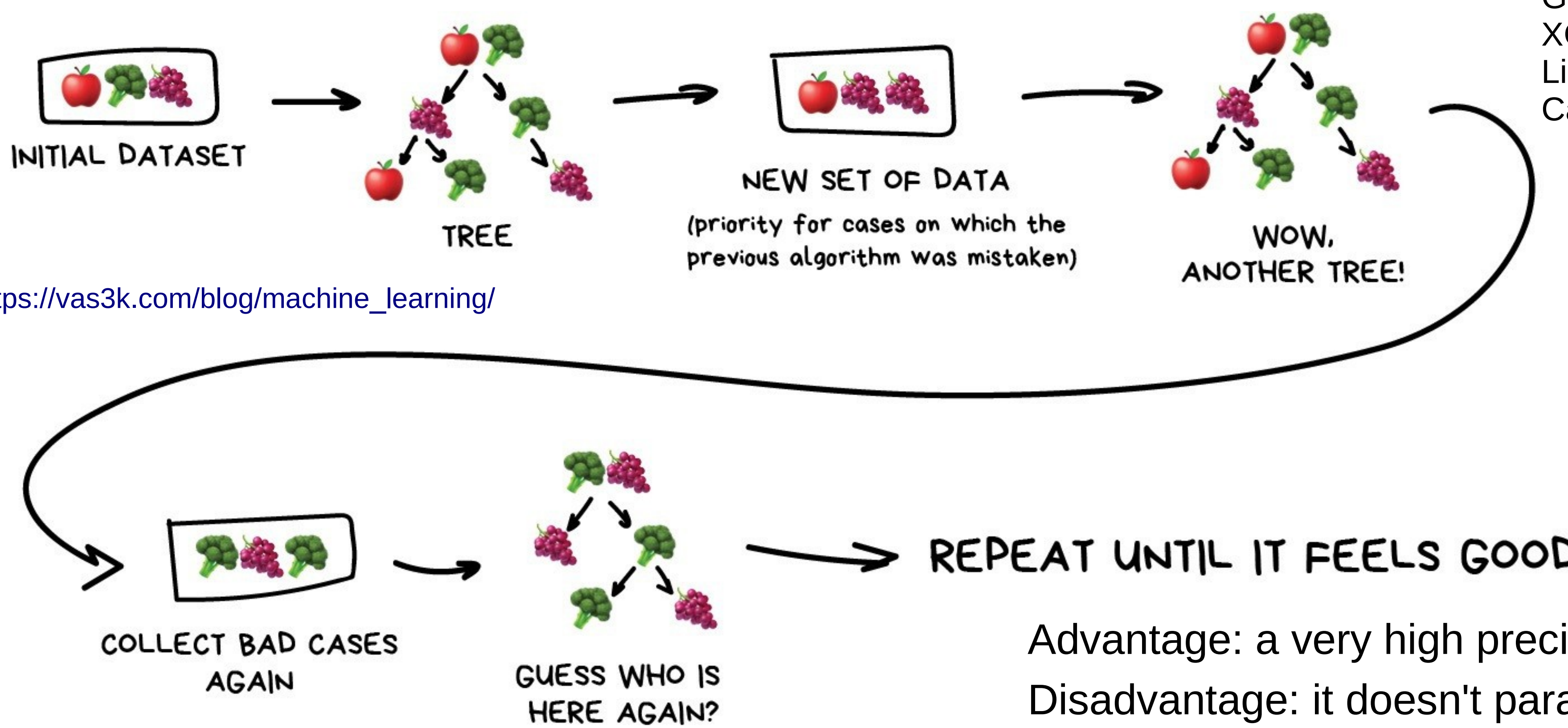
Algorithms are trained one by one sequentially. Each subsequent one paying most of its attention to data points that were mispredicted by the previous one.

Use subsets of the data like in bagging but this time they are not randomly generated.

In each subsample, we take a part of the data the previous algorithm failed to process.

Each next algorithm learns to fix the errors of the previous one.

Boosting algorithms:
- AdaBoost
- GBM
- XGBM
- LightGBM
- CatBoost



https://vas3k.com/blog/machine_learning/

Advantage: a very high precision.

Disadvantage: it doesn't parallelize.

https://towardsdatascience.com/catboost-vs-light-gbm-vs-xgboost-5f93620723db

https://github.com/Bixi81/R-ml

scikit-learn 0.24.1 https://scikit-learn.org/stable/modules/classes.html#module-sklearn.ensemble

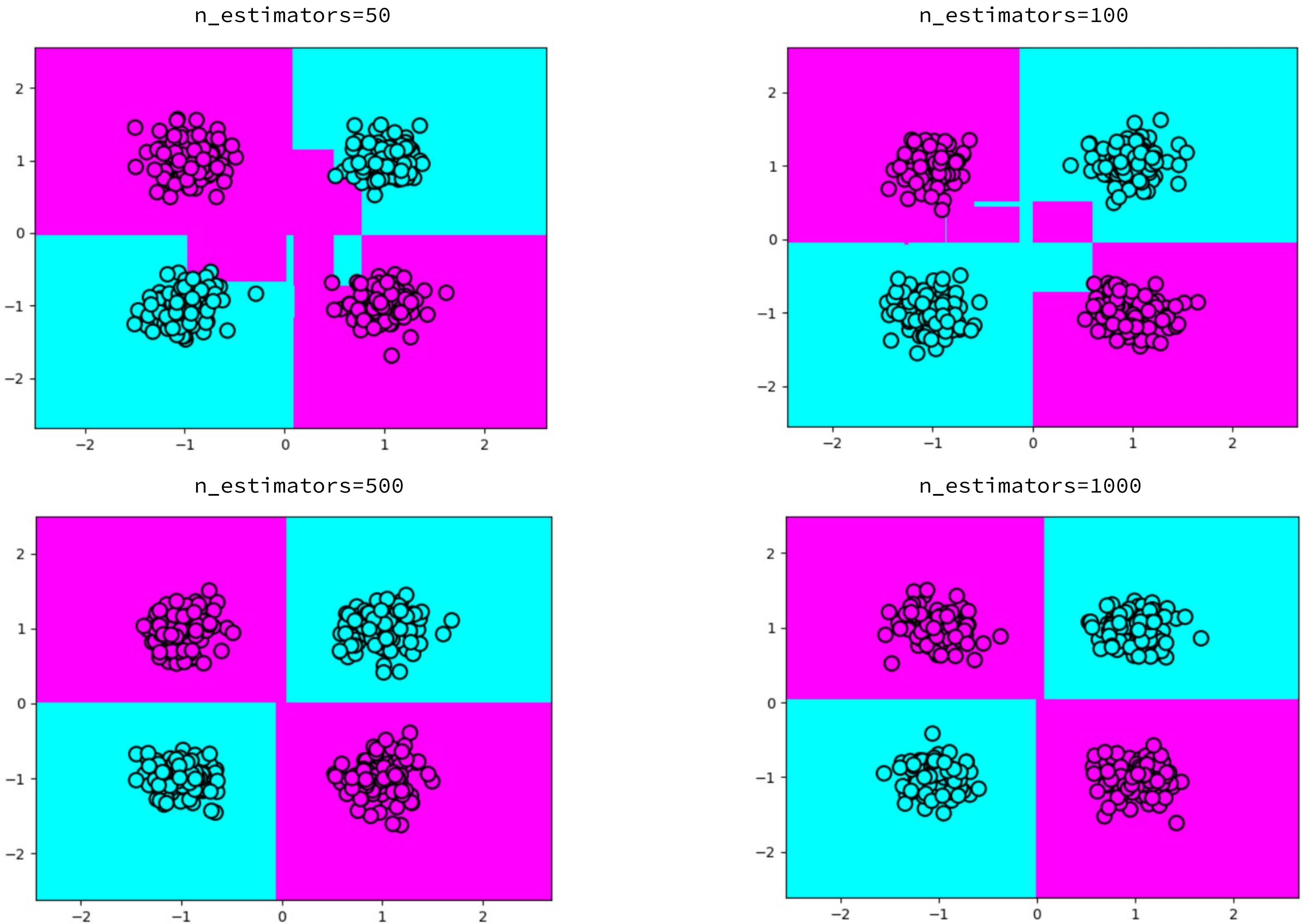
sklearn.ensemble: Ensemble Methods

The sklearn.ensemble module includes ensemble-based methods for classification, regression and anomaly detection.

User guide: See the Ensemble methods section for further details.

ensemble.AdaBoostClassifier([...])	An AdaBoost classifier.
ensemble.AdaBoostRegressor([base_estimator, ...])	An AdaBoost regressor.
ensemble.BaggingClassifier([base_estimator, ...])	A Bagging classifier.
ensemble.BaggingRegressor([base_estimator, ...])	A Bagging regressor.
ensemble.ExtraTreesClassifier([...])	An extra-trees classifier.
ensemble.ExtraTreesRegressor([n_estimators, ...])	An extra-trees regressor.
ensemble.GradientBoostingClassifier(*[, ...])	Gradient Boosting for classification.
ensemble.GradientBoostingRegressor(*[, ...])	Gradient Boosting for regression.
ensemble.IsolationForest(*[, n_estimators, ...])	Isolation Forest Algorithm.
ensemble.RandomForestClassifier([...])	A random forest classifier.
ensemble.RandomForestRegressor([...])	A random forest regressor.
ensemble.RandomTreesEmbedding([...])	An ensemble of totally random trees.
ensemble.StackingClassifier(estimators[, ...])	Stack of estimators with a final classifier.
ensemble.StackingRegressor(estimators[, ...])	Stack of estimators with a final regressor.
ensemble.VotingClassifier(estimators, *[, ...])	Soft Voting/Majority Rule classifier for unfitted estimators.
ensemble.VotingRegressor(estimators, *[, ...])	Prediction voting regressor for unfitted estimators.
ensemble.HistGradientBoostingRegressor([...])	Histogram-based Gradient Boosting Regression Tree.
ensemble.HistGradientBoostingClassifier([...])	Histogram-based Gradient Boosting Classification Tree.

Gradient boosting `max_depth=2`



https://dl.acm.org/doi/10.1145/2939672.2939785

XGBoost - eXtreme Gradient Boosting

https://github.com/dmlc/xgboost

https://xgboost.readthedocs.io/en/latest/index.html

https://xgboost.ai/

Python, R, Java, Scala, C, C++, Julia, Ruby, Swift

– winning in many machine learning competitions

Light Gradient Boosting Machine

https://github.com/Microsoft/LightGBM

https://lightgbm.readthedocs.io/en/latest/index.html

Python, Julia, .NET/C#, Java, Ruby, C, R

https://www.kdnuggets.com/2020/01/explaining-black-box-models-ensemble-deep-learning-lime-shap.html

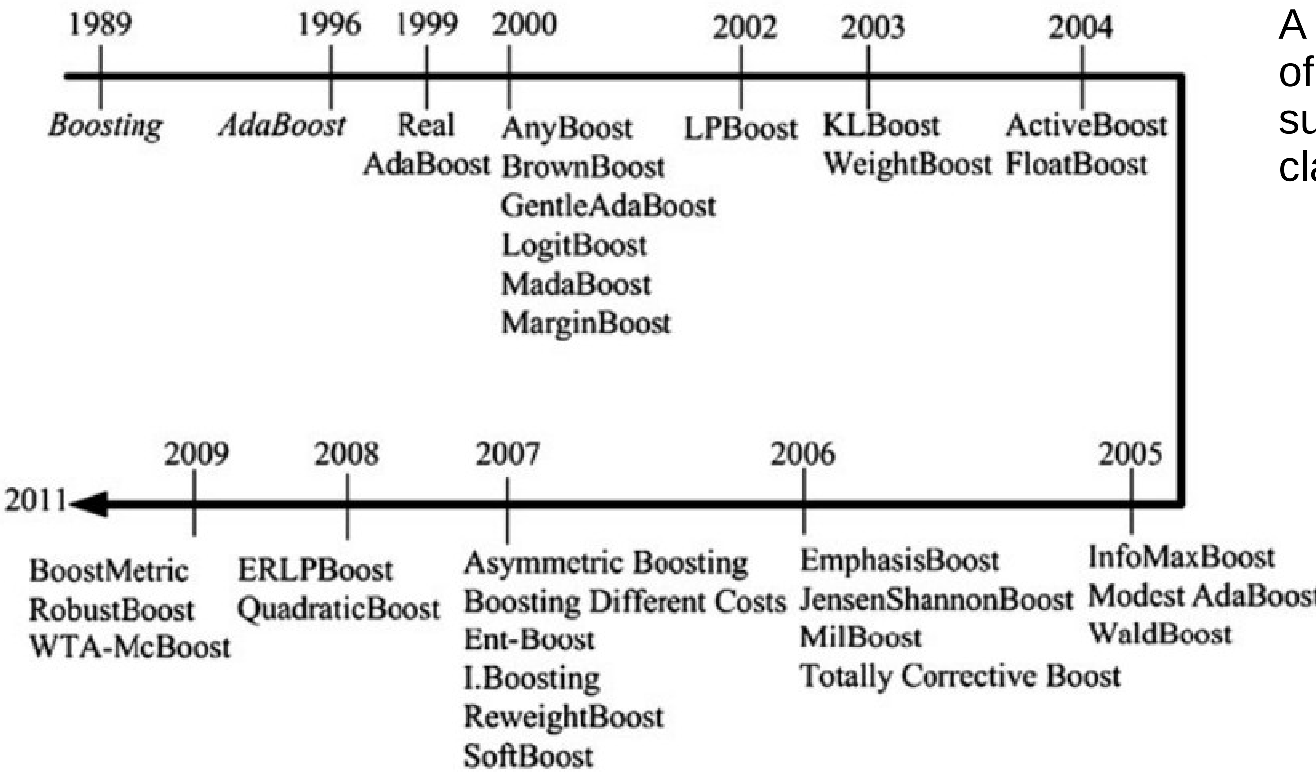
CatBoost
by Yandex

Python, R, Java, C++

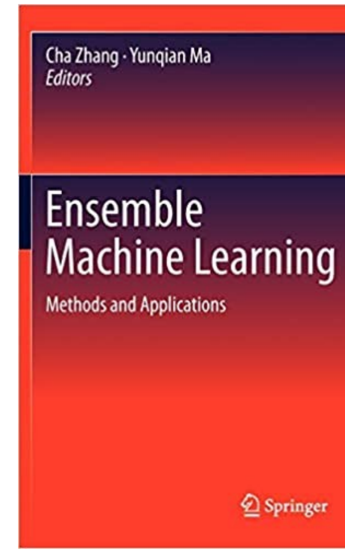
https://www.kdnuggets.com/2020/10/fast-gradient-boosting-catboost.html

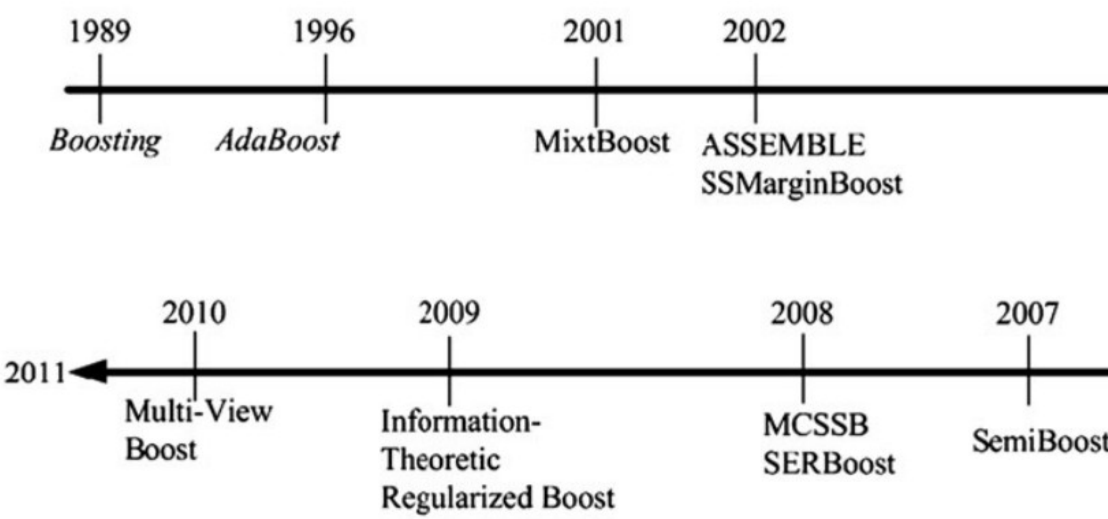
https://catboost.ai/
https://github.com/catboost

https://www.kdnuggets.com/2019/03/mastering-fast-gradient-boosting-google-colaboratory-free-gpu.html



A (possibly incomplete) timeline of AdaBoost variants for supervised learning of binary classifiers, as of 2011





A (possibly incomplete) timeline of AdaBoost variants for semi-supervised learning on binary and multiclass problems, as of 2011

ForEx++: A New Framework for Knowledge Discovery from Decision Forests

https://weka.sourceforge.io/packageMetaData/ForExPlusPlus/index.html

ForestPA: Constructs a Decision Forest by Penalizing Attributes used in Previous Trees.

https://weka.sourceforge.io/packageMetaData/ForestPA/index.html

J48Consolidated: Class for generating a pruned or unpruned C45 consolidated tree

https://weka.sourceforge.io/packageMetaData/J48Consolidated/index.html

OptimizedForest

https://weka.sourceforge.io/packageMetaData/OptimizedForest/index.html

SysFor: Systematically Developed Forest of Multiple Decision Trees.

https://weka.sourceforge.io/packageMetaData/SysFor/index.html

racedIncrementalLogitBoost: Classifier for incremental learning of large datasets by way of racing logit-boosted committees.

https://weka.sourceforge.io/packageMetaData/racedIncrementalLogitBoost/index.html

realAdaBoost: Class for boosting a 2-class classifier using the Real Adaboost method.

https://weka.sourceforge.io/packageMetaData/realAdaBoost/index.html

rotationForest: Ensembles of decision trees trained on rotated subsamples of the training data.

https://weka.sourceforge.io/packageMetaData/rotationForest/index.html

Bagging tips

https://www.coursera.org/learn/build-decision-trees-svms-neural-networks

Use out-of-bag error to evaluate the performance of the trees in the forest on the data they haven't seen during the bagging process.

Use most of the same pre-pruning hyperparameters in a random forest as you would on a single decision tree.

Consider limiting the number of trees to grow in the forest to around a few hundred, as growing more may not be worth the extra training time.

https://www.analyticsvidhya.com/blog/2020/03/beginners-guide-random-forest-hyperparameter-tuning/

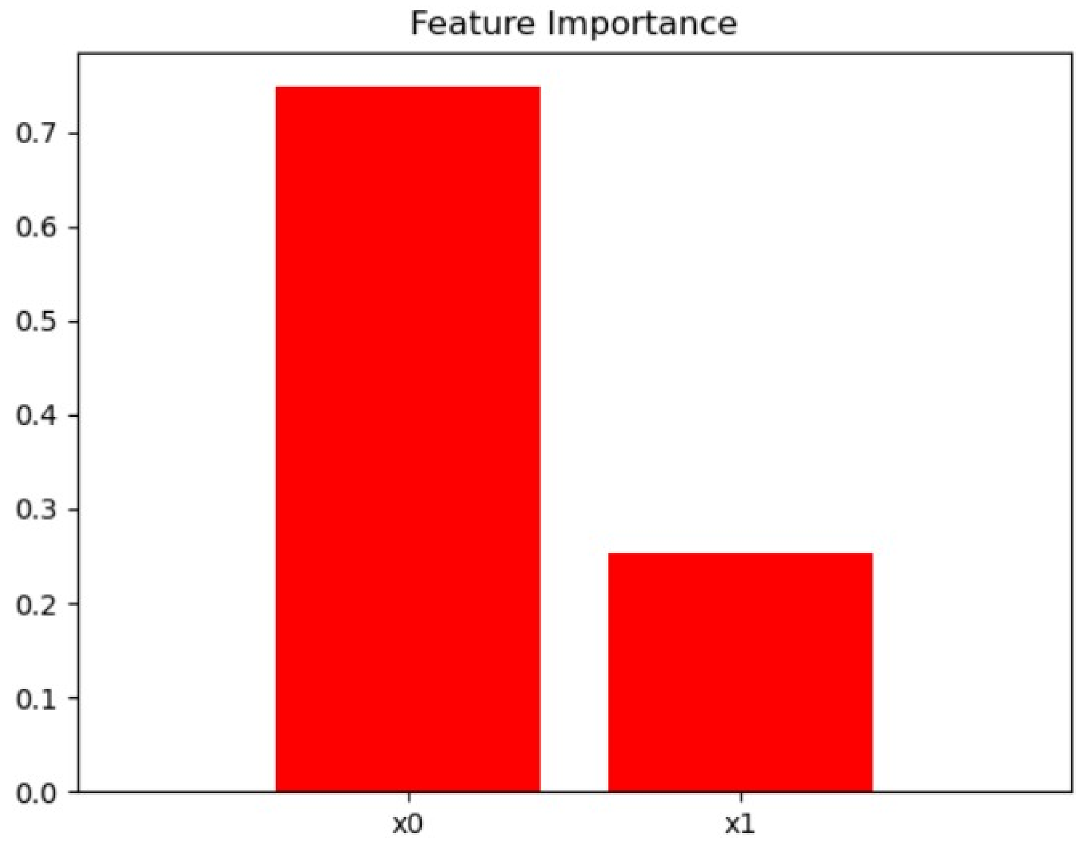
Use random forests for feature selection, culling the features that exhibit less importance and thereby reducing the dimensionality of the dataset.

But remember, they are based on the estimation of mutual information between the target and only one predictor.

https://scikit-plot.readthedocs.io/en/stable/estimators.html

XOR

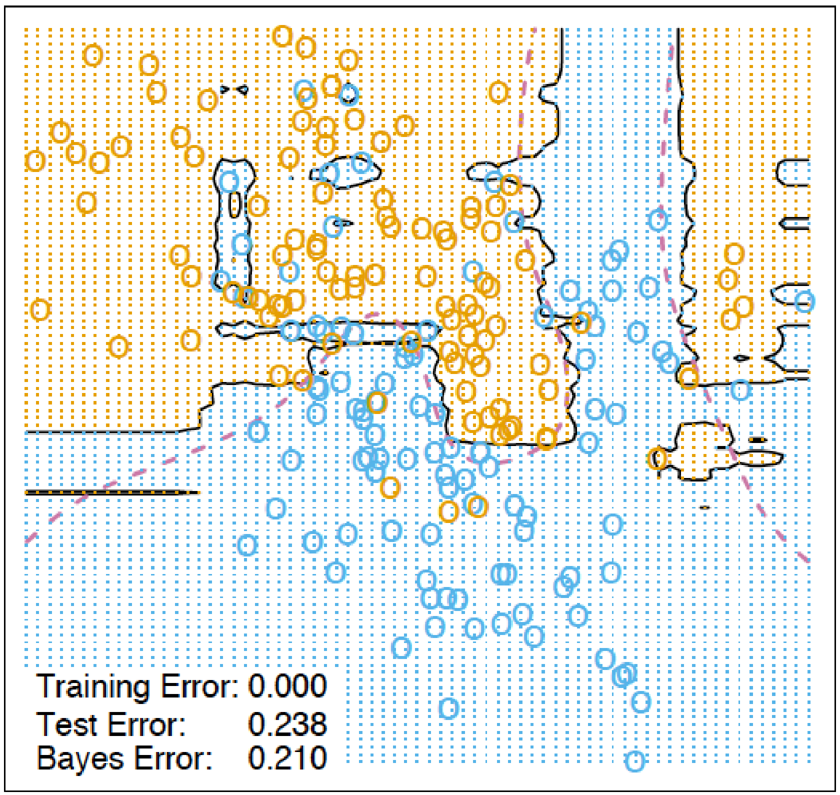
```
clf = GradientBoostingClassifier(n_estimators=1000, max_depth=2)

import scikitplot as skplt

skplt.estimators.plot_feature_importances(clf, feature_names=['x0', 'x1'])
plt.show()
```



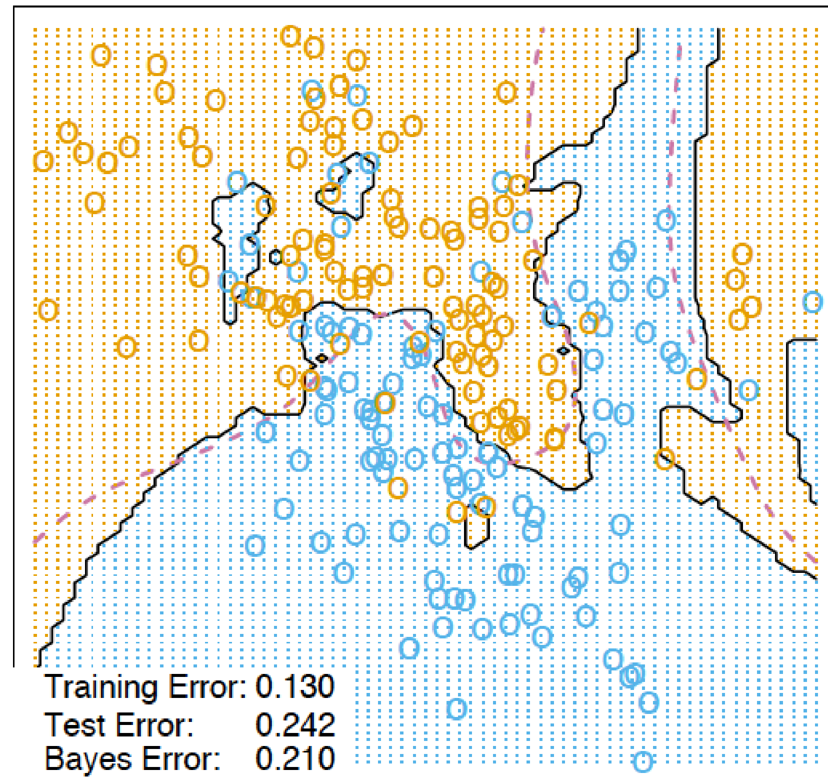
x_0 and x_1 should be equally important!

https://mlcourse.ai/articles/topic5-part2-rf/

Random Forest Classifier

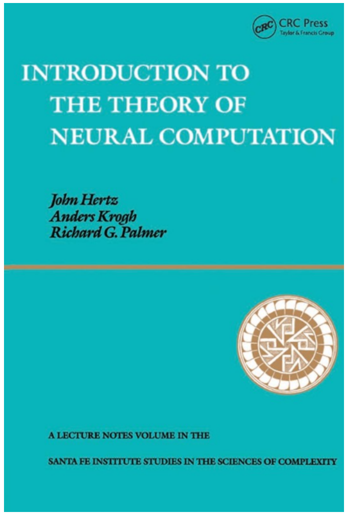


3−Nearest Neighbors



Ensemble learning for neural networks: the **dilution of weights** or **dropout**.

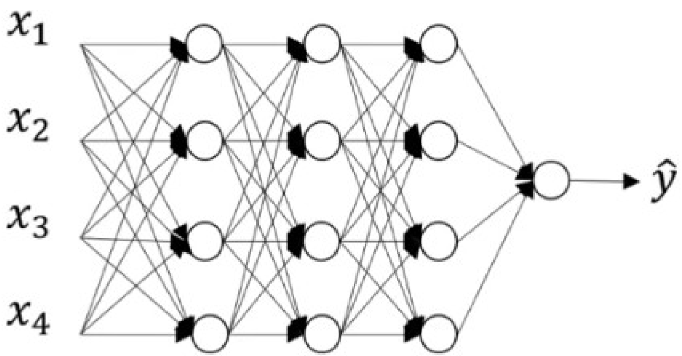
https://en.wikipedia.org/wiki/Dilution_(neural_networks))

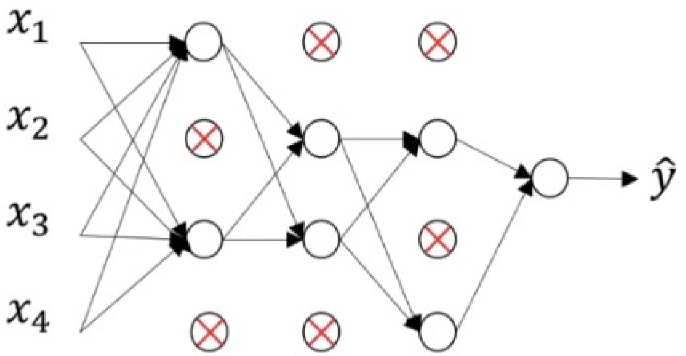


H. Sompolinsky. The Theory of Neural Networks: The Hebb Rules and Beyond (1987)

Hertz, John; Krogh, Anders; Palmer, Richard (1991). Introduction to the Theory of Neural Computation.

A. Canning, E. Gardner; Partially connected models of neural networks. 1988 J. Phys. A: Math. Gen. 21 3275
https://iopscience.iop.org/article/10.1088/0305-4470/21/15/016





https://www.coursera.org/learn/deep-neural-network

A few questions

How to make boosting parallelizable?

How to learn an interpretable model from ensembles?

Is it possible to extract a perfect tree from the forest?