Деревья и композиции моделей

Случайный лес и его окресности

Михаил Андреев

сентябрь 2018

Roadmap

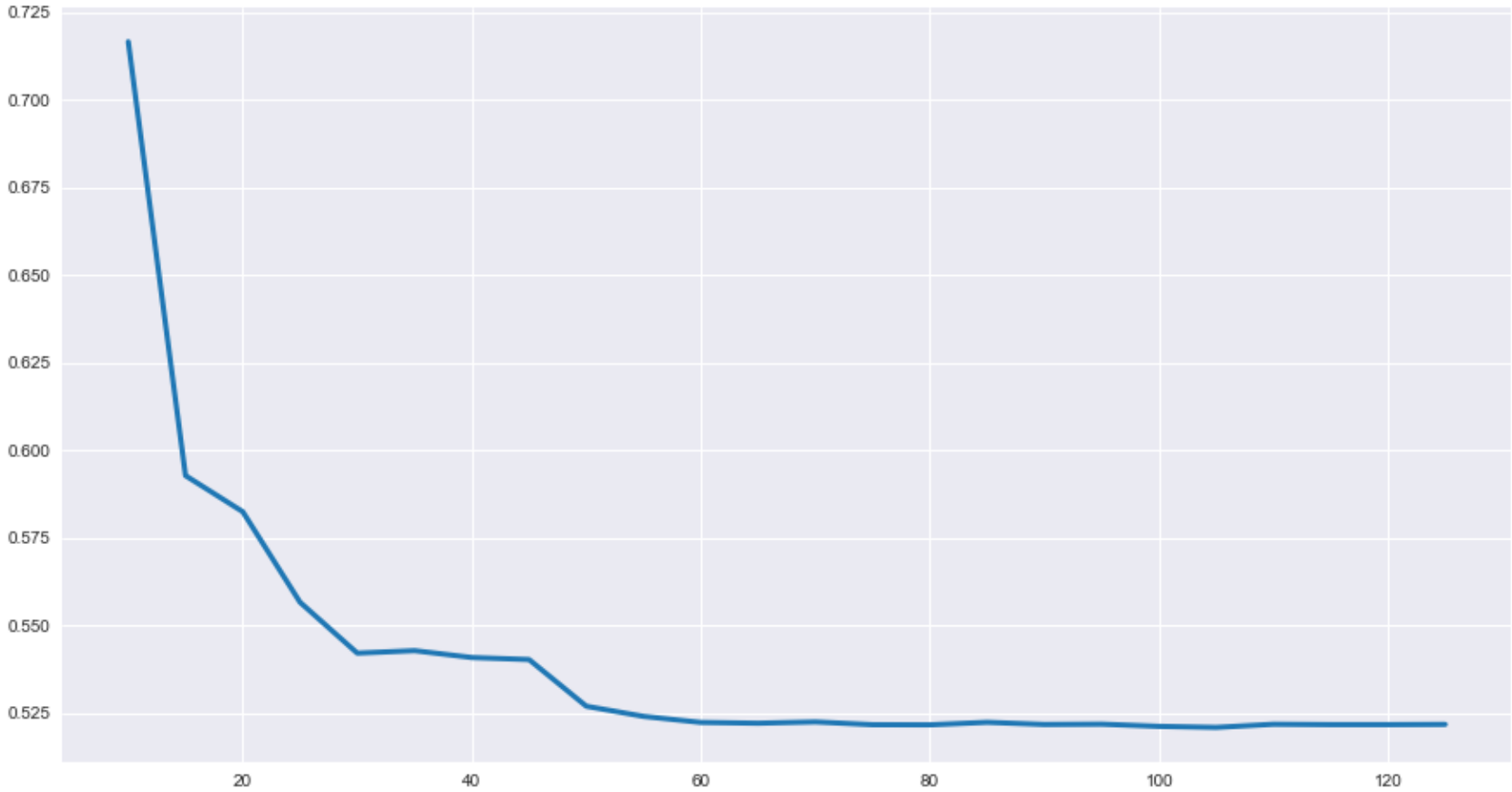
- Random Forest: preview
- Деревья - для построения композиций
- Bagging
- Random Forest: собираем деревья в лес
- Boosting

Random Forest: preview (1)

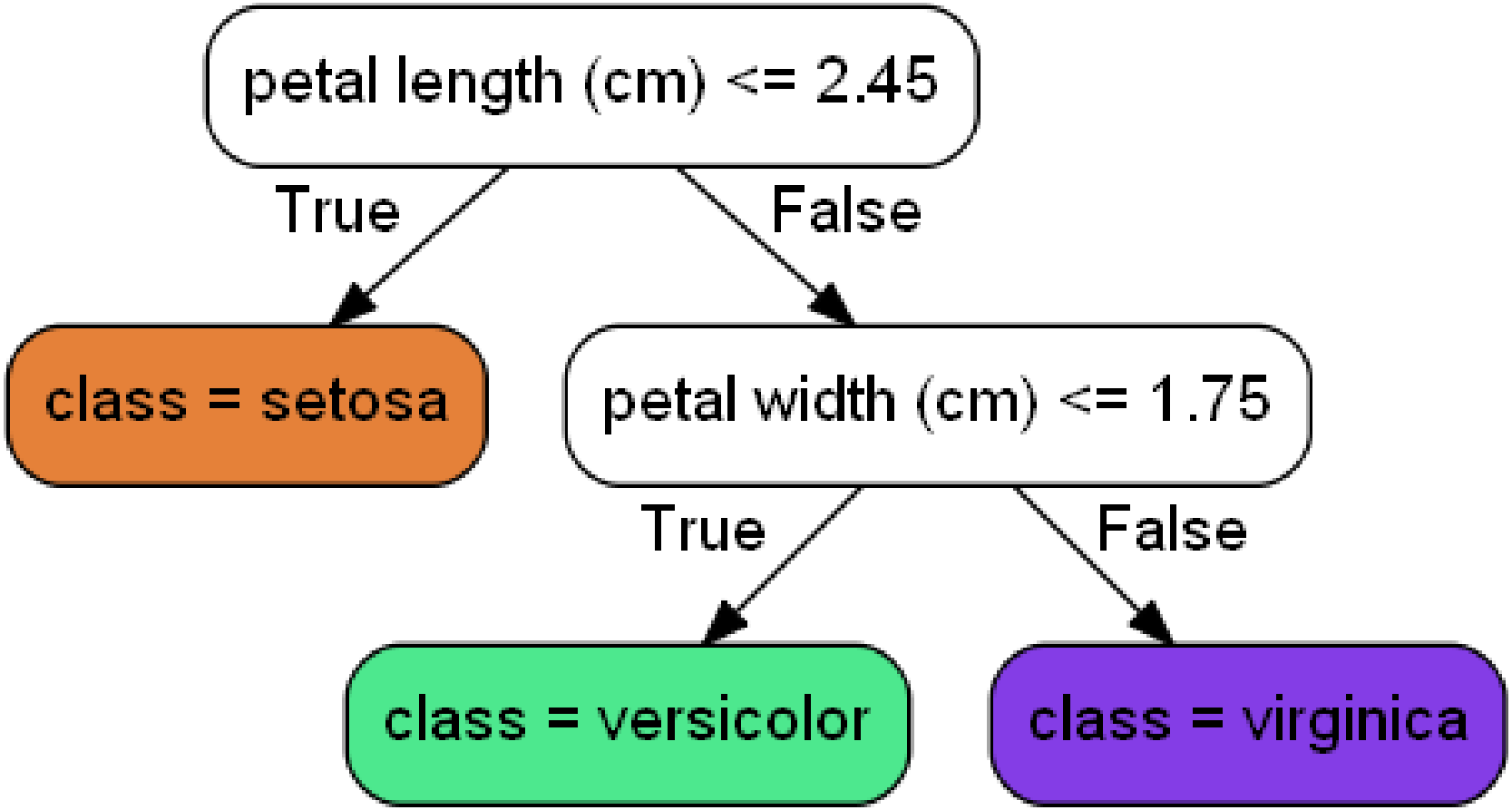
- Алгоритм классификации и регрессии. Часто используется как в иследовательских работах так и на практике.
- Прост в использовании: один-два основных параметра для настройки, нет переобучения; не требует масштабирования признаков; работает как с непрерывными, так и дискретными признаками; обучение распараллеливается.
- После обучения может выдавать дополнительную полезную информацию: вероятность пренадлежнасти к классу, out-of-bag score.

- RF основан на идеи постоения композиции простых алгоритмов (ensemble learning). RF это bagging над деревьями. Есть другие походы к построению композиции, но часто в качестве базовых алгоритмов берут именно деревья.

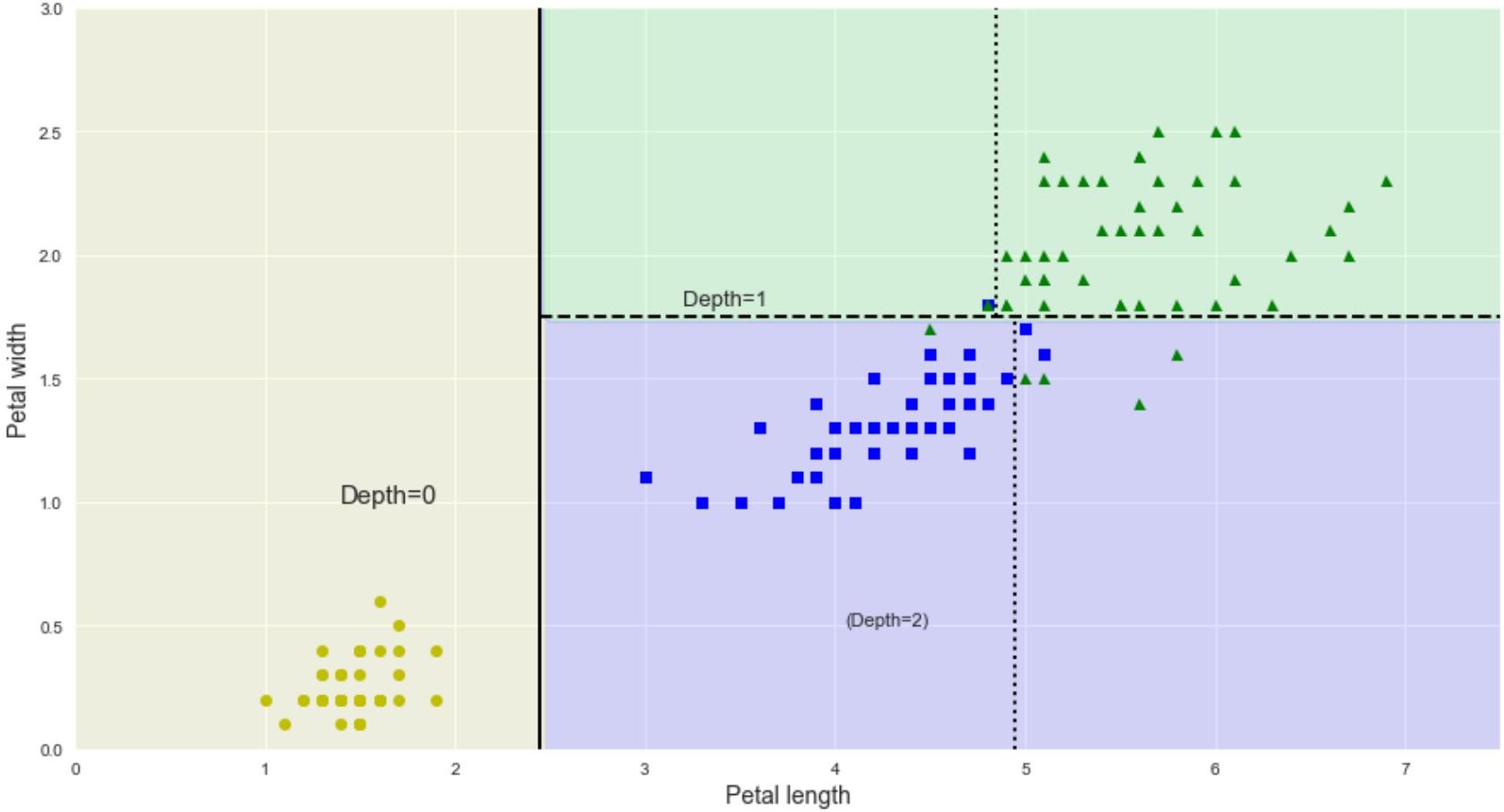
Random Forest: preview (2)



Решающие деревья



Решающие деревья: границы классификации



Автоматическое построение решающего дерева

- Поиск оптимального решение - NP-complete задача :(
- Но существуют быстрые алгоритмы, которые строят достаточно хорошие деревья :)
  - CART
  - C4.5
  - ID3

Алгоритм Classification And Regression Tree (CART) вкраце

Рекурсивное построение дерева. На каждом шаге есть множество объектов, которое мы хотим разделить на два подмножества выбрав фичу k и пороговое значение t_k. Ищем $\underset{k,t_k}{\operatorname{argmin}} J(k, t_k)$

$$J(k, t_k) = \frac{m_{left}}{m} G_{left} + \frac{m_{right}}{m} G_{right}$$

где $G_{left/right}$ – некоторая функция оценивающая неоднородность подмножества

sklearn

DecisionTreeClassifier: параметр criterion поддерживает значения 'gini' и 'entropy'.
DecisionTreeRegressor: 'mse', 'friedman_mse', 'mae'.

Вычислительная сложность

- Query time: $O(\log(N_s))$
- Learning time: $O(N_f N_s \log(N_s))$

http://scikit-learn.org/stable/modules/tree.html#complexity

CART: основные параметры (1)

max_depth.
min_samples_split (the minimum number of samples a node must have before it can be split), **min_samples_leaf** (the minimum number of samples a leaf node must have), **min_weight_fraction_leaf** (same as **min_samples_leaf** but expressed as a fraction of the total number of weighted instances), **max_leaf_nodes** (maximum number of leaf nodes), and **max_features** (maximum number of features that are evaluated for splitting at each node).
Increasing **min_*** hyperparameters or reducing **max_*** hyperparameters will regularize the model
http://scikit-learn.org/stable/modules/tree.html#tips-on-practical-use

Pruning

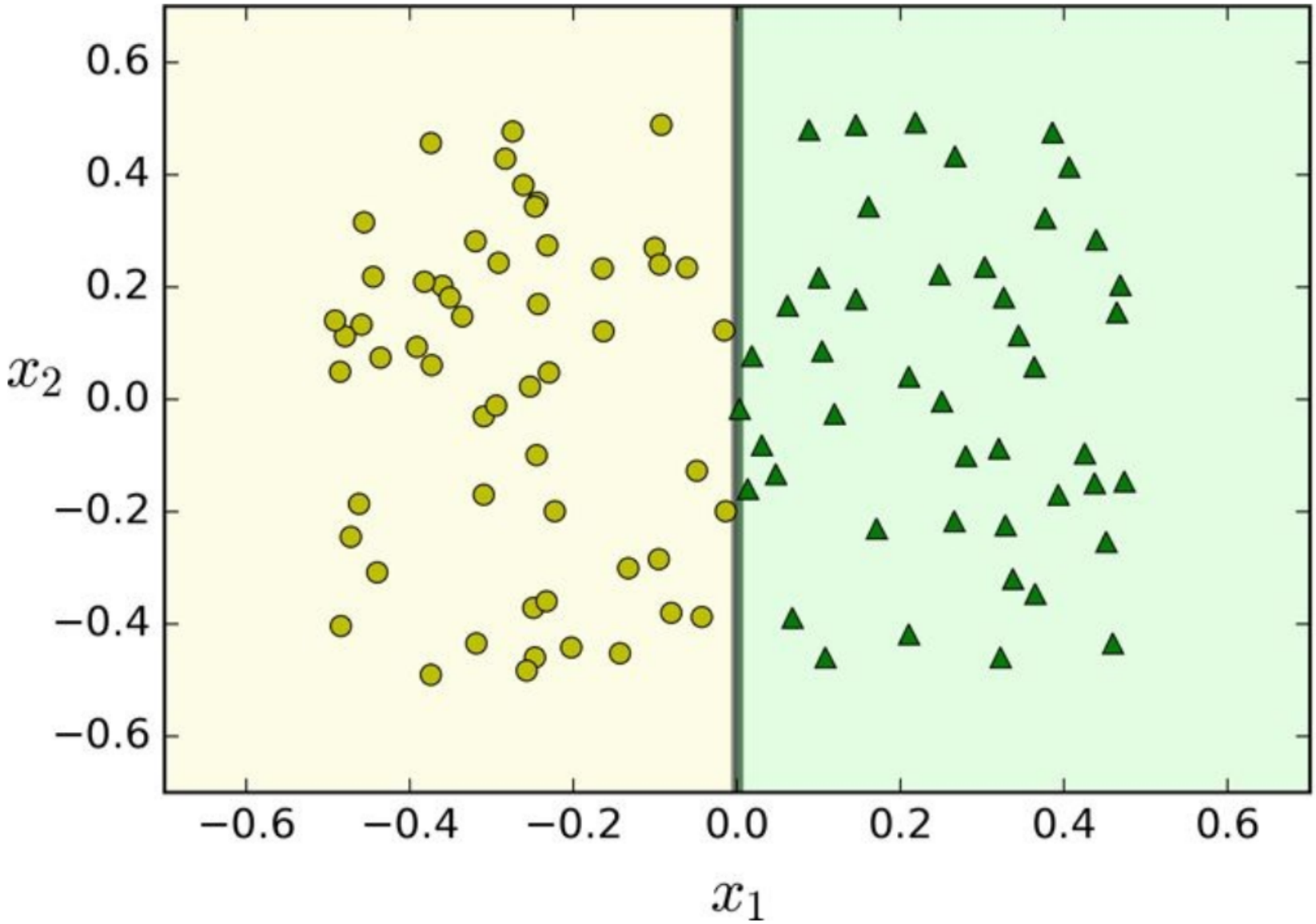
Не реализованно в sklearn

CART: основные параметры (2)

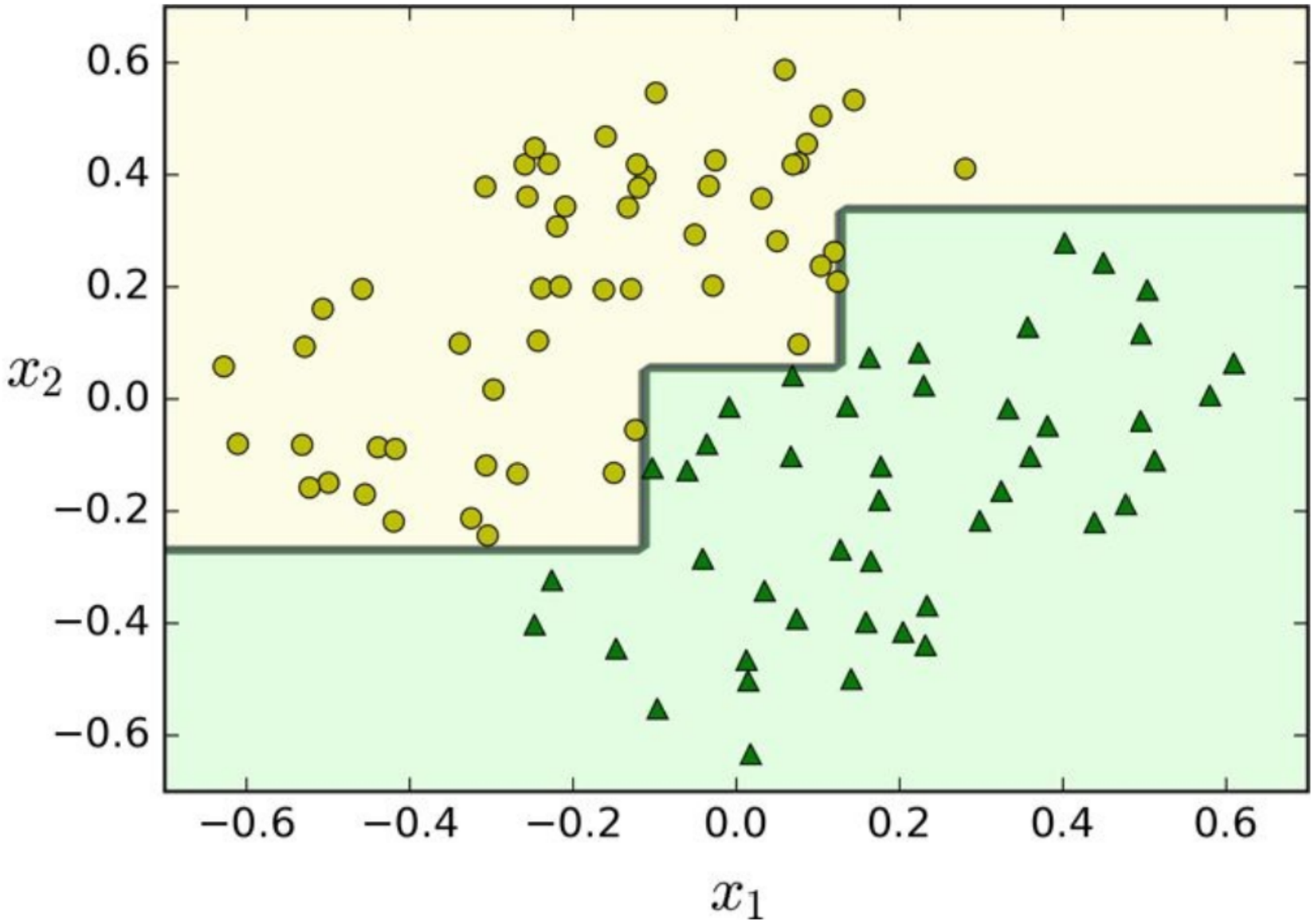
Gini impurity or entropy?

"The truth is, most of the time it does not make a big difference: they lead to similar trees. Gini impurity is slightly faster to compute, so it is a good default. However, when they differ, Gini impurity tends to isolate the most frequent class in its own branch of the tree, while entropy tends to produce slightly more balanced trees"
Aurélien Géron – Hands-on Machine Learning with Scikit-Learn and TensorFlow – 2017
https://github.com/ageron/handson-ml

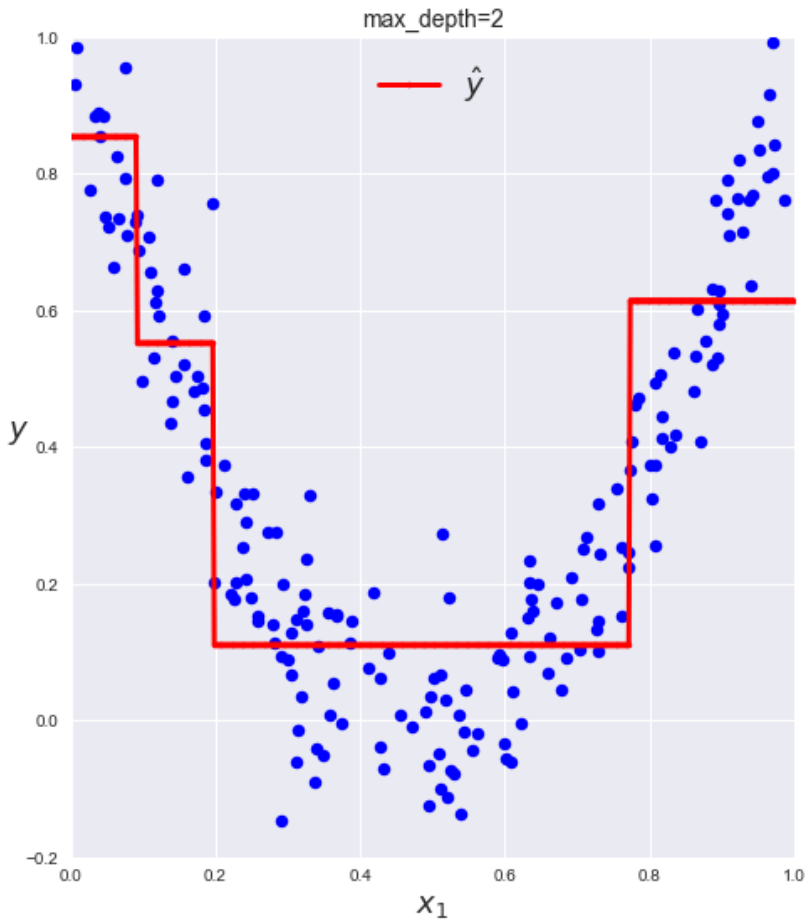
Решающие деревья: неустойчивость (1)

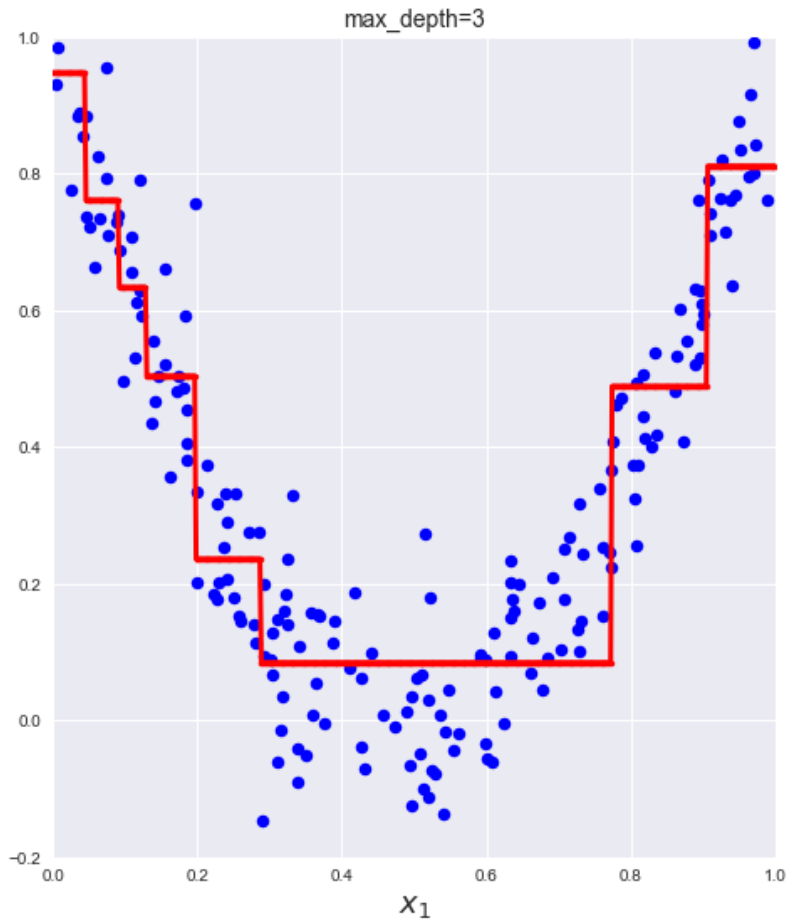


Решающие деревья: неустойчивость (2)



Решающие деревья: регрессия





Решающие деревья: резюме

- Ненужно скейлить данные. Работают с числовыми и категориальными данными.
- white box: интерпритация, визуализация
- Переобучается, чувствительно к выбросам
- Может не повести с проекцией

Bagging: Bootstrap aggregation

Boostrap

Random sampling with replacement

$$P(item_i \in sample_b) = 1 - \left(1 - \frac{1}{N}\right)^N$$

$$\approx 1 - e^{-1} = 0.632$$

Bagging: sklearn (1)

Bagging methods come in many flavours but mostly differ from each other by the way they draw random subsets of the training set:

- When random subsets of the dataset are drawn as random subsets of the samples, then this algorithm is known as Pasting.
- When samples are drawn with replacement, then the method is known as Bagging.
- When random subsets of the dataset are drawn as random subsets of the features, then the method is known as Random Subspaces
- Finally, when base estimators are built on subsets of both samples and features, then the method is known as Random Patches.

http://scikit-learn.org/stable/modules/ensemble.html#bagging-meta-estimator

Bagging: sklearn (2)

Bagging

BaggingClassifier / BaggingRegressor
max_samples and **max_features** control the size of the subsets (in terms of samples and features), while **bootstrap** and **bootstrap_features** control whether samples and features are drawn with or without replacement.

VotingClassifier

voting= {'hard', 'soft' }

RF & Co : sklearn (1)

Classification / regression

- **RandomForestClassifier / RandomForestRegressor**
- In extremely randomized trees (see **ExtraTreesClassifier** and **ExtraTreesRegressor** classes), randomness goes one step further in the way splits are computed. As in random forests, a random subset of candidate features is used, but instead of looking for the most discriminative thresholds, thresholds are drawn at random for each candidate feature and the best of these randomly-generated thresholds is picked as the splitting rule.

Forests : sklearn : parameters

- **n_estimators** the larger the better, but also the longer it will take to compute
- **max_features** the size of the random subsets of features to consider when splitting a node. Empirical good default values are max_features=n_features for regression problems, and max_features=sqrt(n_features) for classification tasks.
- **n_jobs**=-1

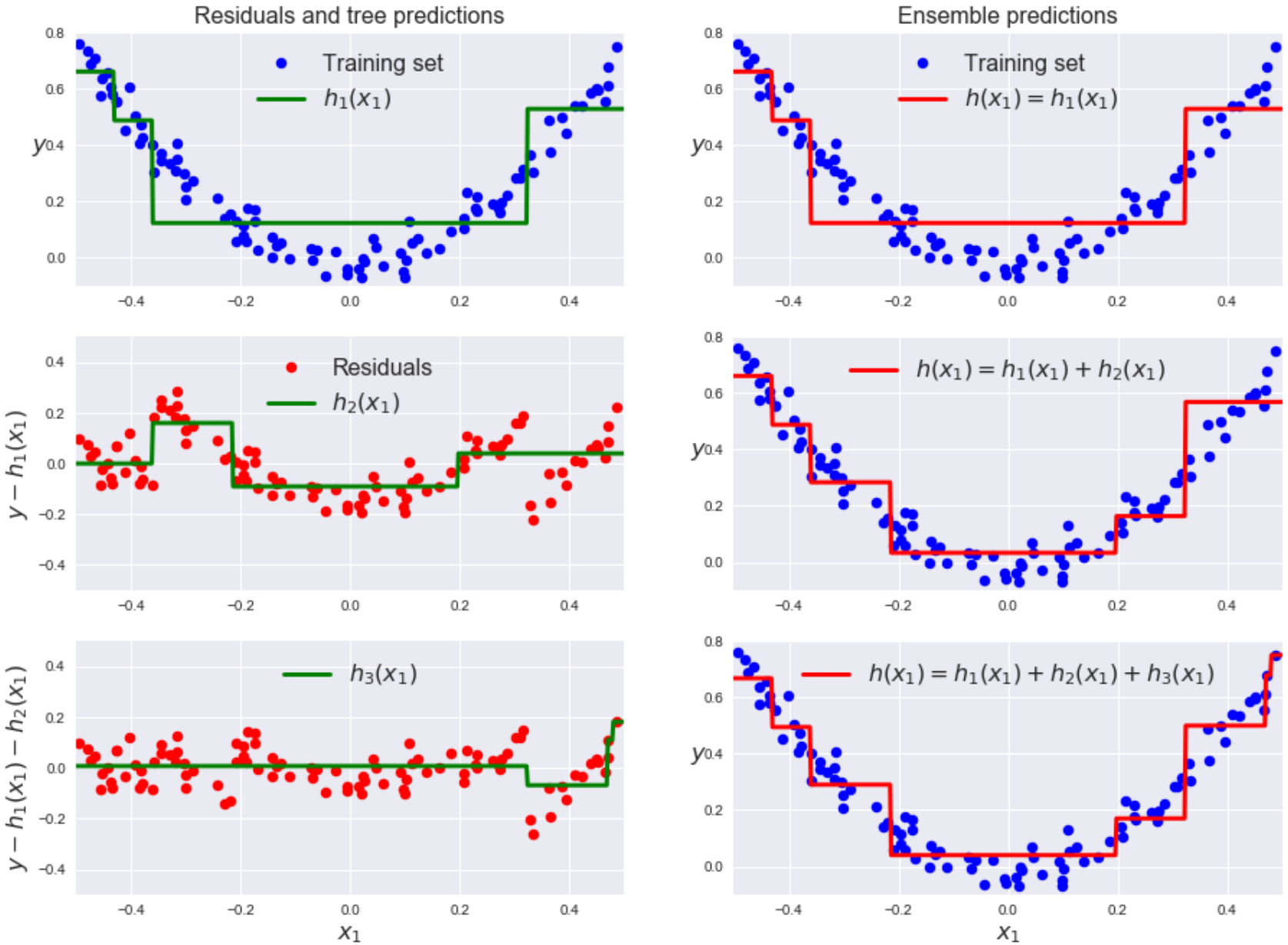
RF & Co : sklearn (2)

- **IsolationForest**
- **RandomTreesEmbedding**

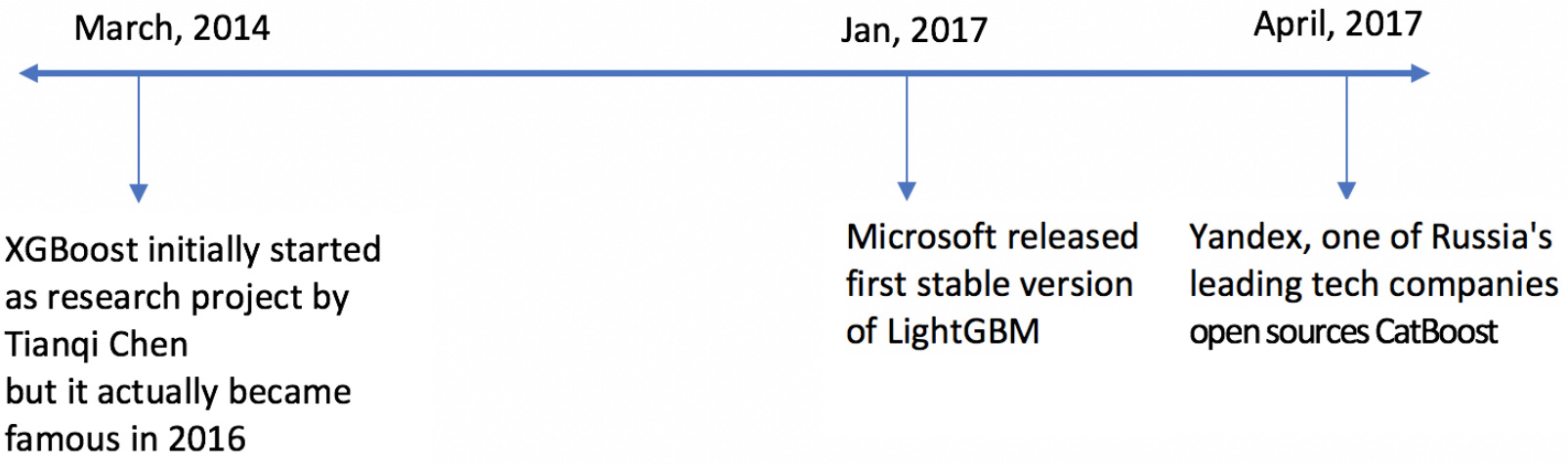
Boosting : sklearn

- **GradientBoostingClassifier / GradientBoostingRegressor**
- **AdaBoostClassifier / AdaBoostRegressor**

GradientBoostingRegressor



Boosting: libraries



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